Weblate

Component configuration:

.. //devel/integration
Weblate

settings.py

SINGLE_PROJECT
<table>
<thead>
<tr>
<th>Component</th>
<th>Translated</th>
<th>Untranslated</th>
<th>Untranslated words</th>
<th>Checks</th>
<th>Suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebateOrg/Android — Czech GNU GPL-3.0</td>
<td>76%</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebateOrg/Django — Hungarian GNU GPL-3.0</td>
<td>69%</td>
<td>8</td>
<td>109</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>WebateOrg/Django — Czech GNU GPL-3.0</td>
<td>96%</td>
<td>1</td>
<td>12</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>WebateOrg/Django — Hebrew GNU GPL-3.0</td>
<td>92%</td>
<td>2</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebateOrg/Djangojs — Hungarian GNU GPL-3.0</td>
<td>96%</td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebateOrg/Djangojs — Hebrew GNU GPL-3.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebateOrg/Djangojs — Czech GNU GPL-3.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebateOrg/Language names — Czech GNU GPL-3.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebateOrg/Language names — Hungarian GNU GPL-3.0</td>
<td>81%</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebateOrg/Language names — Hebrew GNU GPL-3.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebateOrg/WebateOrg — Hungarian GNU GPL-3.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebateOrg/WebateOrg — Czech GNU GPL-3.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebateOrg/WebateOrg — Hebrew GNU GPL-3.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Translated string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hebrew</td>
<td>קברות</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hungarian</td>
<td>Főkó</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English</td>
<td>Files</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**String information**

- **Screenshot context**: No screenshot currently associated.
- **Explanation**: No explanation currently provided.
- **Labels**: No labels currently set.
- **Flags**: No flags currently set.
- **Source string location**: [webate/templates/translation.html:45](https://webate/trans/forms.py:1404)
- **String age**: 23 seconds ago
- **Source string age**: 24 seconds ago
- **Translation file**: [webate/loc8or/cs/LC_MESSAGES/django.po](https://webate/loc8or/cs/LC_MESSAGES/django.po)
Component configuration

**Web**

VCS

```
editor://open/?file={{filename}}&line={{line}}
```

**URL**

Nette documentation

---

DE-FAULT_AUTO_WATCH

---
Watched projects

Automatically watch projects on contribution
Whenever you translate a string in a project, you will start watching it.

Available:  
- Webiite.org

Chosen:  
- Webiite.org

You can receive notifications for watched projects and they are shown on the dashboard by default.
Add all projects you want to translate to see them as watched projects on the dashboard.

Save

Notification settings

Other projects  Watched projects  Managed projects

Component wide notifications

You will receive a notification for every such event in your watched projects.

- Repository Failure
  - Do not notify
- Repository operation
  - Do not notify
- Component locking
  - Do not notify
- Changed license
  - Do not notify
- Parse error
  - Do not notify
- Comment on own translation
  - Instant notification
- Mentioned in comment
  - Instant notification
- New language
  - Do not notify
- New translation component
  - Do not notify
- New announcement
  - Instant notification
- New alert
  - Do not notify

Translation notifications

You will only receive these notifications for your translated languages in your watched projects.

- New string
  - Do not notify
- New contributor
  - Do not notify
- New suggestion
  - Do not notify
- New comment
  - Do not notify
- Changed string
  - Do not notify
- Translated string
  - Do not notify
- Approved string
  - Do not notify
- Pending suggestions
  - Do not notify
- Settings needing action
  - Do not notify

Save
<table>
<thead>
<tr>
<th>Platform</th>
<th>Web and Desktop</th>
<th>Mobile</th>
<th>Web</th>
<th>Desktop</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>GitLab</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GitHub</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Google</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Facebook</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bitbucket</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OAuth 2.0
Your profile

Languages  Preferences  Notifications  Account  Profile  Licenses  Audit log  API access

Account

Username

testuser

Username may only contain letters, numbers or the following characters: @ _ + -_

Full name

Weblate Test

E-mail

webmaster@example.org

You can add another e-mail address below.

Your name and e-mail will appear as commit authorship.

Save

Current user identities

<table>
<thead>
<tr>
<th>Identity</th>
<th>User ID</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>testuser</td>
<td>Change password</td>
</tr>
<tr>
<td>E-mail</td>
<td><a href="mailto:webmaster@example.org">webmaster@example.org</a></td>
<td>Disconnect</td>
</tr>
<tr>
<td>Google</td>
<td><a href="mailto:webmaster@example.org">webmaster@example.org</a></td>
<td>Disconnect</td>
</tr>
<tr>
<td>GitHub</td>
<td>123456</td>
<td>Disconnect</td>
</tr>
<tr>
<td>Bitbucket</td>
<td>webmaster</td>
<td>Disconnect</td>
</tr>
</tbody>
</table>

Add new association

E-mail

Removal

Account removal deletes all your private data.

Remove my account

User data

You can download all your private data.

Download user data
### Translation status

<table>
<thead>
<tr>
<th>Strings</th>
<th>Percentage</th>
<th>Completion Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>26 Strings</td>
<td>96%</td>
<td>Browse</td>
</tr>
<tr>
<td>183 Words</td>
<td>93%</td>
<td>Translate</td>
</tr>
</tbody>
</table>

### Strings status

- **26** Strings — 183 words
- **25** Translated strings — 171 words
- **1** Strings needing action — 12 words
- **1** Net translated strings — 12 words
- **1** Strings needing action without suggestions — 12 words
- **3** Strings with any failing checks — 11 words
- **3** Translated strings with any failing checks — 11 words
- **1** Failed check: Unchanged translation — 4 words
- **1** Failed check: Mismatched full stop — 4 words
- **1** Failed check: Python format — 3 words

### Other components

<table>
<thead>
<tr>
<th>Component</th>
<th>Translated</th>
<th>Untranslated</th>
<th>Untranslated words</th>
<th>Checks</th>
<th>Suggestions</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>76%</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language names</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glossary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Django</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Powered by [Webate](https://weblate.org)
Weblate

Markdown

@mention

report-source

variants

labels

Unicode Language Plural Rules
<table>
<thead>
<tr>
<th>English</th>
<th>Czech, One</th>
<th>Czech, Few</th>
<th>Czech, Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>%count{} word</td>
<td>%count{} slovo</td>
<td>%count{} slova</td>
<td>%count{} slov</td>
</tr>
<tr>
<td>%count{} words</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Plural formula: \(n=1\) ? 0; \(n=2 \& n\neq4\) ? 1:2

- Needs editing

---

**String information**
- Screenshot context: No screenshot currently associated.
- Add screenshot
- Explanation: No explanation currently provided.
- Labels: No labels currently set.
- Flags: python-format
- Source string location: web/website/templates/translation.html
- String age: 13 seconds ago
- Source string age: 14 seconds ago
- Translation file: web/website/website/co/NC_MESSAGE/django.py,string 1

---

**New comment**

Comment on this string for fellow translators and developers to read.

**Scope**
- Translation comment, discussions with other translators

Is your comment specific to this translation or generic for all of them?

**New comment**

You can use Markdown and mention users by @username.

---

**Glossary**

- English
- Czech
  - No related strings found in the glossary.
  - Add term to glossary
Weblate: Visual context for strings

<table>
<thead>
<tr>
<th>Language</th>
<th>Translated string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Czech</td>
<td>Soubory</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Fülfök</td>
</tr>
<tr>
<td>English</td>
<td>Files</td>
</tr>
</tbody>
</table>

String age
17 seconds ago

Source string age
28 seconds ago

Translation file
webate/local/he/LC_MESSAGE/5/django.po, string 1
Keeping translations same across components
Bulk edit addon

Translation types capabilities

gettext PO
gettext XLIFF
XLIFF 1.1
TermBase eXchange
Translation Memory eXchange
ggettext MO (gettext PO)
CSV
Excel Open XML
JSON
Android
iOS
GET /api/translations/(string:project)/(string:component)/(string:language)/file/
POST /api/translations/{string:project}/{string:component}/{string:language}/file/

Weblate 4.5

Weblate
Untranslatable terms

4.5. read-only

4.5. forbidden

4.5. terminology
Variants are a generic way to group strings together. All term variants are listed in the glossary sidebar when translating.

**variants:** You can use this to add abbreviations or shorter expressions for a term.
The translation has been saved, however there are some newly failing checks: Missing plurals, Python format

Translation

**English**

Singular

%{(count)}s word

Plural

%{(count)}s words

**Czech**

One

nekolik slov

Other

%{(count)}s slov

Plural formula: (n==1) ? 0 : (n==2 & & n<4) ? 1 : 2

**Needs editing**

Things to check

- **Python format**
  - Following format strings are missing: %{(count)}s
  - Dismiss

- **Missing plurals**
  - Some plural forms are not translated
  - Dismiss

Glossary

- English -> Czech
  - No related strings found in the glossary.
  - Add term to glossary

String information

- Screenshot context: No screenshot currently associated.
- Add screenshot

- Explanation: No explanation currently provided.

- Labels: No labels currently set.

- Flags: python-format

- Source string location
  - webiste/translations/translation.h

- String age: 20 seconds ago

- Source string age: 21 seconds ago

- Translation file
  - webiste/local/translations.django.py,string3
**BBcode**

```python
weblate.checks.markup.BBCodeCheck
ignore-bbcode
```

**4.1.2**

```python
weblate.checks.duplicate.DuplicateCheck
ignore-duplicate
```

**4.5**

```python
weblate.checks.glossary.GlossaryCheck
check-glossary
ignore-check-glossary
```
weblate.checks.chars.DoubleSpaceCheck
ignore-double-space

Weblate: Additional info on source strings
Component configuration:

Weblate: C 'c-format' Gettext PO xgettext
AngularJS

AngularJS

weblate.checks.angularjs.AngularJSInterpolationCheck

angularjs-format

ignore-angularjs-format

Your balance is {{amount}} {{ currency }}:

```
AngularJS [https://angular.jp/guide/interpolation]`
```
C format strings in C printf format

C# format strings in C# printf format

ECMAScript format strings in JavaScript

i18next format strings in i18next
Java

Java

weblate.checks.format.JavaFormatCheck
java-format
ignore-java-format
There are \%d apples
Your balance is \%1$d \%2$s

[Java](https://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html)

JavaScript

JavaScript

weblate.checks.format.JavaScriptFormatCheck
javascript-format
ignore-javascript-format
There are \{0\} apples


Lua

Lua

weblate.checks.format.LuaFormatCheck
lua-format
ignore-lua-format
There are \%d apples

Object Pascal

Object Pascal

weblate.checks.format.ObjectPascalFormatCheck
object-pascal-format
ignore-object-pascal-format
There are %d apples

Perl

Perl

weblate.checks.format.PerlFormatCheck
perl-format
ignore-perl-format
There are %d apples
Your balance is %1$d %2$s

PHP

PHP

weblate.checks.format.PHPFormatCheck
php-format
ignore-php-format
There are %d apples
Your balance is %1$d %2$s
Python

weblate.checks.format.PythonBraceFormatCheck
python-brace-format
ignore-python-brace-format
There are {} apples
Your balance is {amount} {currency}:

Python brace

Python

weblate.checks.format.PythonFormatCheck
python-format
ignore-python-format
There are %d apples
Your balance is %{amount}d %{currency}s

Python string formatting

Qt

weblate.checks.qt.QtFormatCheck
qt-format
ignore-qt-format
There are %1 apples

Qt QString::arg()

Qt

weblate.checks.qt.QtPluralCheck
qt-plural-format
ignore-qt-plural-format
There are %Ln apple(s)

Qt i18n
https://doc.qt.io/qt-5/i18n-source-translation.html#handling-plurals>_<
Ruby

weblate.checks.ruby.RubyFormatCheck
ruby-format
ignore-ruby-format
There are %d apples
Your balance is %1$f %2$s
Your balance is %+.2<amount>f %<currency>s
Your balance is %{amount} %{currency}

Scheme

weblate.checks.format.SchemeFormatCheck
scheme-format
ignore-scheme-format
There are ~d apples

Vue I18n

weblate.checks.format.VueFormattingCheck
vue-format
ignore-vue-format
There are {count} apples
There are %{count} apples
@:message.dio @:message.the_world!

Vue I18n Formatting, Srfi 28, Chicken Scheme format, Guile Scheme formatted output

Vue I18n Linked locale messages

all strings
weblate.checks.consistency.TranslatedCheck
ignore-translated

VCS
For performance reasons, the check might not find all inconsistencies, it limits number of matches.

---

### Keeping translations same across components

**Kashida**

<table>
<thead>
<tr>
<th>Version</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td><a href="https://en.wikipedia.org/wiki/Kashida">Wikipedia</a></td>
</tr>
</tbody>
</table>

**Markdown**

<table>
<thead>
<tr>
<th>Version</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td><a href="https://en.wikipedia.org/wiki/Kashida">Wikipedia</a></td>
</tr>
</tbody>
</table>

Markdown

Mark down 3.5.

Markdown

weblate.checks.markup.MarkdownRefLinkCheck
md-text
ignore-md-reflink

Markdown

Markdown links

Markdown

Mark down 3.5.

Markdown

weblate.checks.markup.MarkdownSyntaxCheck
md-text
ignore-md-syntax

Markdown

Markdown span elements

Markdown

weblate.checks.chars.MaxLengthCheck
max-length
ignore-max-length

max-length:100 key:value

replacements:

Markdown

weblate.checks.render.MaxSizeCheck
max-size
ignore-max-size

max-size: 3.7
Number of "\n in translation does not match source.

weblate.checks.chars.EscapedNewlineCountingCheck
ignore-escaped-newline

weblate.checks.chars.EndColonCheck
ignore-end-colon

weblate.checks.chars.EndEllipsisCheck
ignore-end-ellipsis

weblate.checks.chars.EndExclamationCheck
ignore-end-exclamation
weblate.checks.chars.EndStopCheck
ignore-end-stop

Wikipedia

weblate.checks.chars.EndQuestionCheck
ignore-end-question

Wikipedia

weblate.checks.chars.EndSemicolonCheck
ignore-end-semicolon

Wikipedia

weblate.checks.chars.NewLineCountCheck
ignore-newline-count

weblate.checks.consistency.PluralsCheck
ignore-plurals
weblate.checks.placeholders.PlaceholderCheck
placeholders
ignore-placeholders

weblate.checks.chars.PunctuationSpacingCheck
ignore-punctuation-spacing

weblate.checks.placeholders.RegexCheck
regex
ignore-regex

placeholders:$URL$:$TARGET$:"some long text"

placeholders:r"[^% ]%"

weblate.checks.consistency.SamePluralsCheck
ignore-same-plurals

weblate.checks.chars.BeginNewlineCheck
ignore-begin-newline

weblate.checks.chars.BeginSpaceCheck
ignore-begin-space

weblate.checks.chars.EndNewlineCheck
ignore-end-newline

weblate.checks.chars.EndSpaceCheck
ignore-end-space

39
weblate.checks.same.SameCheck
ignore-same

Component configuration

weblate.checks.markup.SafeHTMLCheck
safe-html
ignore-safe-html

HTML

weblate.checks.markup.URLCheck
url
ignore-url

URL

weblate.checks.markup.XMLTagsCheck
ignore-xml-tags

XML

---

This check is disabled by the safe-html flag as the HTML cleanup done by it can produce HTML markup which is not valid XML.
**XML**

Weblate checks mark up XML. Weblate.

**weblate.checks.markup.XMLValidityCheck**

ignore-xml-invalid

XML

This check is disabled by the `safe-html` flag as the HTML cleanup done by it can produce HTML markup which is not valid XML.

**weblate.checks.chars.ZeroWidthSpaceCheck**

ignore-zero-width-space

\[ \text{\textcopyright} \]

Wikipedia:

**weblate.checks.source.EllipsisCheck**

ignore-ellipsis

\[ \ldots \]

Wikipedia:

**weblate.checks.source.LongUntranslatedCheck**

ignore-long-untranslated

\[ \text{\textcopyright} \]
weblate.checks.source.MultipleFailingCheck
ignore-multiple-failures

weblate.checks.format.MultipleUnnamedFormatsCheck
ignore-unnamed-format

weblate.checks.source.OptionalPluralCheck
ignore-optional-plural

The string is used as a plural, but does not use plural forms. In case your translation system supports this, you should use the plural aware variant of it.

Python Gettext:

```python
from gettext import ngettext

print ngettext("Selected %d file", "Selected %d files", files) % files
```

3.9.1.42
"this is a quoted string" and 'another quoted string'"
Component slug or name case insensitive search, see \texttt{URL} and \texttt{TITLE}.

\texttt{TITLE: URL
\texttt{TITLE: source
\texttt{TITLE: change_time
\texttt{TITLE: change_action
\texttt{TITLE: state
\texttt{TITLE: source
\texttt{TITLE: change_time:2018 AND change_action:marked-for-edit
\texttt{TITLE: change_time:2018 AND change_action:"Marked for edit"

\texttt{AND OR NOT state:translated AND (source:hello OR source:bar)

\texttt{translated approved 2019

\texttt{= hello world

\texttt{4.4
2 ~ 5
source=r"[2-5]"
The string uses three dots (…) instead of an ellipsis character (…).
access control

Translation states

Gettext 

In case file format you use does not support storing states, you might want to use a addon to flag unchanged strings as needing editing.

Translation types capabilities

<table>
<thead>
<tr>
<th>Value</th>
<th>ON</th>
<th>OFF</th>
<th>OFF</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>per-project access control</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>
Weblate 2.18

<table>
<thead>
<tr>
<th>Value</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td></td>
</tr>
<tr>
<td>OFF</td>
<td></td>
</tr>
</tbody>
</table>

per-project access control

Language aliases

Comma-separated list of language code mappings, for example: en_GB, en_US

Enable source reviews

Requires dedicated reviewers to approve source strings.
Development process

- Developers
  - Intermediate file

Localization process

- Translators
  - Translation language file

- Editors
  - Monolingual base language file

1. Weblate Git
2. Weblate Component configuration
3. Weblate Component configuration

Avoiding merge conflicts
If you've already encountered a merge conflict, the easiest way to solve all conflicts locally on your machine, is to add Weblate as a remote repository, merge it into upstream and fix any conflicts. Once you push changes back, Weblate will be able to use the merged version without any other special actions.

```bash
# Commit all pending changes in Weblate, you can do this in the UI as well:
wlc commit
# Lock the translation in Weblate, again this can be done in the UI as well:
wlc lock
# Add Weblate as remote:
git remote add weblate https://hosted.weblate.org/git/project/component/
# You might need to include credentials in some cases:
git remote add weblate https://username:APIKEY@hosted.weblate.org/git/
# Update weblate remote:
git remote update weblate

# Merge Weblate changes:
git merge weblate/main

# Resolve conflicts:
edit ...
git add ...
... 
git commit

# Push changes to upstream repository, Weblate will fetch merge from there:
git push

# Open Weblate for translation:
wlc unlock

# Add and update Weblate remotes

git remote add weblate-one https://hosted.weblate.org/git/project/one/
git remote add weblate-second https://hosted.weblate.org/git/project/

git remote update weblate-one weblate-second

# Merge QA_4_7 branch:
git checkout QA_4_7
git merge weblate-one/QA_4_7
... # Resolve conflicts

git commit

# Merge main branch:
git checkout main
git merge weblates-second/main
... # Resolve conflicts

git commit
```
# Push changes to the upstream repository, Weblate will fetch the merge
from there:
git push

-gettext PO:

Weblate Git upstream
# Add remote:
git remote add weblate /path/to/weblate/snapshot/
# Update Weblate remote:
git remote update weblate
# Merge Weblate changes:
git merge weblate/main
# Resolve conflicts in the PO files:
for PO in `find . -name '*po'`; do
msgcat --use-first /path/to/weblate/snapshot/$PO
/path/to/upstream/snapshot/$PO -o $PO.merge
msgmerge --previous --lang=$(PO%.po) $PO.merge domain.pot -o $PO
rm $PO.merge
git add $PO
done
git commit

# Push changes to the upstream repository, Weblate will fetch merge from
there:
git push

## How to export the Git repository that Weblate uses?
Avoiding merge conflicts

Weblate Project configuration
Component configuration

Webtate:

git merge -s ours origin/maintenance

Keeping translations same across components

### How to translate multi-platform projects?

Weblate supports a wide range of file formats (see) and the easiest approach is to use the native format for each platform.

Once you have added all platform translation files as components in one project (see Adding translation projects and components), you can utilize the translation propagation feature (turned on by default, and can be turned off in the Component configuration) to translate strings for all platforms at once.

Keeping translations same across components
How to export the Git repository that Weblate uses?

There is nothing special about the repository, it lives under the `DATA_DIR` directory and is named `vcs/<project>/<component>/`. If you have SSH access to this machine, you can use the repository directly.

For anonymous access, you might want to run a Git server and let it serve the repository to the outside world. Alternatively, you can use `Git exporter` inside Weblate to automate this.

What are the options for pushing changes back upstream?

This heavily depends on your setup, Weblate is quite flexible in this area. Here are examples of some workflows used with Weblate:

- Weblate automatically pushes and merges changes (see `Performance report`).
- You manually tell Weblate to push (it needs push access to the upstream repository).
- Somebody manually merges changes from the Weblate git repository into the upstream repository.
- Somebody rewrites history produced by Weblate (e.g. by eliminating merge commits), merges changes, and tells Weblate to reset the content in the upstream repository.
- Of course you are free to mix all of these as you wish.

How can I limit Weblate access to only translations, without exposing source code to it?

You can use `git submodule` for separating translations from source code while still having them under version control.

1. Create a repository with your translation files.
2. Add this as a submodule to your code:

   ```bash
   git submodule add git@example.com:project-translations.git path/to/translations
   ```

3. Link Weblate to this repository, it no longer needs access to the repository containing your source code.
4. You can update the main repository with translations from Weblate by:

   ```bash
   git submodule update --remote path/to/translations
   ```

   Please consult the `git submodule` documentation for more details.

How can I check whether my Weblate is set up properly?

Weblate includes a set of configuration checks which you can see in the admin interface, just follow the `Performance report` link in the admin interface, or open the `/manage/performance/` URL directly.

Why are all commits committed by Weblate `<noreply@weblate.org>`?

This is the default committer name, configured when you create a translation component. You can change it in the administration at any time.

The author of every commit (if the underlying VCS supports it) is still recorded correctly as the user that made the translation.

Component configuration
Usage

How do I review the translations of others?

There are several review based workflows available in Weblate, see [review workfows here].
You can subscribe to any changes made in [see subscription] and then check others contributions as they come in by e-mail.
There is a review tool available at the bottom of the translation view, where you can choose to browse translations made by others since a given date.

How do I provide feedback on a source string?

On context tabs below translation, you can use the Comments tab to provide feedback on a source string, or discuss it with other translators.

How can I use existing translations while translating?

Weblate [import functionality] allows you to use existing translations. Use the import functionality to load compendium as translations, suggestions or translations needing review. This is the best approach for a one-time translation using a compendium or a similar translation database.
You can set up [tmserver] with all databases you have and let Weblate use it. This is good when you want to use it several times during translation.
Another option is to translate all related projects in a single Weblate instance, which will make it automatically pick up translations from other projects as well.

Does Weblate update translation files besides translations?

Weblate tries to limit changes in translation files to a minimum. For some file formats it might unfortunately lead to reformatting the file. If you want to keep the file formatted your way, please use a pre-commit hook for that.

Where do language definitions come from and how can I add my own?

The basic set of language definitions is included within Weblate and Translate-toolkit. This covers more than 150 languages and includes info about plural forms or text direction.
You are free to define your own languages in the administrative interface, you just need to provide info about it.
Can Weblate highlight changes in a fuzzy string?

Weblate supports this, however it needs the data to show the difference. For Gettext PO files, you have to pass the parameter `--previous` to `msgmerge` when updating PO files, for example:

```
msgmerge --previous -U po/cs.po po/phpmyadmin.pot
```

For monolingual translations, Weblate can find the previous string by ID, so it shows the differences automatically.

Why does Weblate still show old translation strings when I’ve updated the template?

Weblate does not try to manipulate the translation files in any way other than allowing translators to translate. So it also does not update the translatable files when the template or source code have been changed. You simply have to do this manually and push changes to the repository. Weblate will then pick up the changes automatically.

It is usually a good idea to merge changes done in Weblate before updating translation files, as otherwise you will usually end up with some conflicts to merge.

For example with gettext PO files, you can update the translation files using the `msgmerge` tool:

```
msgmerge -U locale/cs/LC_MESSAGES/django.mo locale/django.pot
```

In case you want to do the update automatically, you can install addon `POT PO (msgmerge)`.

Troubleshooting

Requests sometimes fail with "too many open files" error

This happens sometimes when your Git repository grows too much and you have many of them. Compressing the Git repositories will improve this situation.

The easiest way to do this is to run:

```
# Go to DATA_DIR directory
cd data/vcs
# Compress all Git repositories
for d in */* ; do
    pushd $d
    git gc
    popd
done
```

When accessing the site I get a "Bad Request (400)" error

This is most likely caused by an improperly configured `ALLOWED_HOSTS`. It needs to contain all hostnames you want to access on your Weblate. For example:

```
ALLOWED_HOSTS = ["weblate.example.com", "weblate", "localhost"]
```
What does mean "There are more files for the single language (en)"?

This typically happens when you have translation file for source language. Weblate keeps track of source strings and reserves source language for this. The additional file for same language is not processed.

You might get similar error message for other languages as well. In that case the most likely reason is that several files map to single language in Weblate.

This can be caused by using obsolete language codes together with new one (ja and jp for Japanese) or including both country specific and generic codes (fr and fr_FR). See wiki for more details.

Does Weblate support other VCSes than Git and Mercurial?

Weblate currently does not have native support for anything other than Git (with extended support for GitHub, Gerrit and Subversion) and Mercurial, but it is possible to write backends for other VCSes.

You can also use Git in Git to access other VCSes.

Weblate also supports VCS-less operation, see wiki.

For native support of other VCSes, Weblate requires using distributed VCS, and could probably be adjusted to work with anything other than Git and Mercurial, but somebody has to implement this support.

How does Weblate credit translators?

Every change made in Weblate is committed into VCS under the translators name. This way every single change has proper authorship, and you can track it down using the standard VCS tools you use for code.

Additionally, when the translation file format supports it, the file headers are updated to include the translator’s name.

list_translators

Why does Weblate force showing all PO files in a single tree?

Weblate was designed in a way that every PO file is represented as a single component. This is beneficial for translators, so they know what they are actually translating.

Why does Weblate use language codes such sr_Latn or zh_Hant?

These are language codes defined by RFC 5646 to better indicate that they are really different languages instead previously wrongly used modifiers (for @latin variants) or country codes (for Chinese).

Weblate still understands legacy language codes and will map them to current one - for example sr@latin will be handled as sr_Latn or zh@CN as zh_Hans.

Weblate defaults to POSIX style language codes with underscore, see wiki for more details.
Weblate supports most translation format understood by translate-toolkit, however each format being slightly different, some issues with formats that are not well tested can arise.

### Translation Related File Formats

When choosing a file format for your application, it’s better to stick some well established format in the toolkit/platform you use. This way your translators can additionally use whatever tools they are used to, and will more likely contribute to your project.

For correct use of monolingual files, Weblate requires access to a file containing complete list of strings to translate with their source—this file is called within Weblate, though the naming might vary in your paradigm.

Additionally this workflow can be extended by utilizing to include strings provided by developers, but not to be used as is in the final strings.

Weblate can automatically detect several widespread file formats, but this detection can harm your performance and will limit features specific to given file format (for example automatic addition of new translations).

### Translation types capabilities

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GNUgettext</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>needs editing</td>
</tr>
<tr>
<td>Monolingual gettext</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>needs editing</td>
</tr>
<tr>
<td>XLIFF</td>
<td>both</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>needs editing, approved</td>
</tr>
<tr>
<td>Java proper-both</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>mil8n lang</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>WGT</td>
<td>mono</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Joomla translations</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Qt Linguist.ts both</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>needs editing</td>
</tr>
<tr>
<td>Android string</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Apple iOS strings</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>PHP</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>JSON files</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>JSON i18next.files</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>go-i18n JSON files</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ARB File</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>WebExtension JSON</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Format</td>
<td>Linguality</td>
<td>Plurals</td>
<td>Comments</td>
<td>Context</td>
<td>Location</td>
<td>Flags</td>
<td>Additional states</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>---------</td>
<td>----------</td>
<td>---------</td>
<td>----------</td>
<td>-------</td>
<td>-------------------</td>
</tr>
<tr>
<td>XML resource files</td>
<td>mono</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSV files</td>
<td>both</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>needs editing</td>
</tr>
<tr>
<td>YAML files</td>
<td>mono</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Ruby YAML files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DTD files</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Flat XML files</td>
<td>mono</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Windows RC files</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Excel OpenXML</td>
<td>mono</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>needs editing</td>
</tr>
<tr>
<td>Stringsdict format</td>
<td>mono</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Fluent format</td>
<td>mono</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

Plurals are necessary to properly localize strings with variable count. Comments can be used to pass additional info about the string to translate. Context is used to differentiate identical strings used in different scopes (for example "Sun" can be used as an abbreviated name of the day "Sunday" or as the name of our closest star). Location of a string in source code might help proficient translators figure out how the string is used.

Additional states supported by the file format in addition to "Not translated" and "Translated". The gettext type comments are used as flags. The flags are extracted from the non-standard attribute `weblate-flags` for all XML based formats. Additionally `max-length:N` is supported through the `maxwidth` attribute as defined in the XLIFF standard, see `Specifying translation flags`.

XML comment placed before the `<string>` element, parsed as a developer comment. The plurals are supported only for Laravel which uses in string syntax to define them, see `Localization in Laravel`. Plurals are handled in the syntax of the strings and not exposed as plurals in Weblate.
3.10. Read-only strings from translation files will be included, but cannot be edited in Weblate. This feature is natively supported by few formats (XLIFF and Android string resources), but can be emulated in others by adding a read-only flag, see [Android string resources](https://developer.android.com/training/dalvik-vm/resources).

**GNU gettext**

Libre [GNU gettext](https://www.gnu.org/software/gettext/)

Contextual info stored in the file is supported by adjusting its headers or linking to corresponding source files.

The bilingual gettext PO file typically looks like this:

```plaintext
#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "Monday"
msgstr "Pondělí"

#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "Tuesday"
msgstr "Úterý"

#: weblate/accounts/avatar.py:163
msgctxt "No known user"
msgid "None"
msgstr "Žádný"
```

**Typical Weblate Component configuration**

```
po/*.po

po/messages.pot

Gettext PO file
```

See also: [Gettext Wikipedia](https://en.wikipedia.org/wiki/Gettext)
**Monolingual gettext**

Some projects decide to use gettext as monolingual formats—they code just the IDs in their source code and the string then needs to be translated to all languages, including English. This is supported, though you have to choose this file format explicitly when importing components into Weblate.

The monolingual gettext PO file typically looks like this:

```
#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-monday"
msgstr "Pondělí"

#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-tuesday"
msgstr "Úterý"

#: weblate/accounts/avatar.py:163
msgid "none-user"
msgstr "Žádný"
```

While the base language file will be:

```
#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-monday"
msgstr "Monday"

#: weblate/media/js/bootstrap-datepicker.js:1421
msgid "day-tuesday"
msgstr "Tuesday"

#: weblate/accounts/avatar.py:163
msgid "none-user"
msgstr "None"
```

Typical Weblate Component configuration

```
po/*.po
po/en.po
po/messages.pot
```

**XLIFF**

XML-based format created to standardize translation files, but in the end it is one of many standards, in this area. XML Localization Interchange File Format (XLIFF) is usually used as bilingual, but Weblate supports it as monolingual as well.

**Note:**

XML Localization Interchange File Format (XLIFF) specification

**Translation states**

**Note:**

3.3: Weblate ignored the state attribute prior to the 3.3 release.

The state attribute in the file is partially processed and mapped to the "Needs edit" state in Weblate (the following states are used to flag the string as needing edit if there is a target present: new, needs-translation, needs-adaptation, needs-l10n). Should the state attribute be missing, a string is considered translated as soon as a <target> element exists.

If the translation string has approved="yes", it will also be imported into Weblate as "Approved", anything else will be imported as "Waiting for review" (which matches the XLIFF specification).

While saving, Weblate doesn’t add those attributes unless necessary:

The state attribute is only added in case string is marked as needing edit.

The approved attribute is only added in case string has been reviewed.

In other cases the attributes are not added, but they are updated in case they are present.
That means that when using the XLIFF format, it is strongly recommended to turn on the Weblate review process, in order to see and change the approved state of strings.

Similarly upon importing such files (in the upload form), you should choose Import as translated under Processing of strings needing edit.

### Whitespace and newlines in XLIFF

Generally types or amounts of whitespace is not differentiated between in XML formats. If you want to keep it, you have to add the `xml:space="preserve"` flag to the string.

```xml
<trans-unit id="10" approved="yes">
  <source xml:space="preserve">hello</source>
  <target xml:space="preserve">Hello, world!</target>
</trans-unit>
```

### Specifying translation flags

You can specify additional translation flags (see [XLIFF 1.2 font attribute](https://www.w3.org/TR/xliff-2.0/#font-attribute)) by using the `weblate-flags` attribute. Weblate also understands `maxwidth` and `font` attributes from the XLIFF specification:

```xml
<trans-unit id="10" maxwidth="100" size-unit="pixel" font="ubuntu;22;bold">
  <source>Hello %s</source>
</trans-unit>
<trans-unit id="20" maxwidth="100" size-unit="char" weblate-flags="c-format">
  <source>Hello %s</source>
</trans-unit>
```

The `font` attribute is parsed for font family, size and weight, the above example shows all of that, though only font family is required. Any whitespace in the font family is converted to underscore, so `Source Sans Pro` becomes `Source_Sans_Pro`, please keep that in mind when naming the font group (see [XLIFF 1.2 font attribute](https://www.w3.org/TR/xliff-2.0/#font-attribute)).

Weblate identifies the units in the XLIFF file by `resname` attribute in case it is present and falls back to `id` (together with `file` tag if present).

The `resname` attribute is supposed to be human friendly identifier of the unit making it more suitable for Weblate to display instead of `id`. The `resname` has to be unique in the whole XLIFF file. This is required by Weblate and is not covered by the XLIFF standard - it does not put any uniqueness restrictions on this attribute.

**Typical Weblate Component configuration for bilingual XLIFF**

```
localizations/*.xliff
Empty
localizations/en-US.xliff
XLIFF Translation File
```

**Typical Weblate Component configuration for monolingual XLIFF**

```
localizations/*.xliff
localizations/en-US.xliff
localizations/en-US.xliff
XLIFF Translation File
```

---

XLIFF Wikipedia [XML](https://www.w3.org/2002/xhtmlź/), [XLIFF: font attribute in XLIFF 1.2](https://www.w3.org/TR/xliff-2.0/#font-attribute), [maxwidth attribute in XLIFF 1.2](https://www.w3.org/TR/xliff-2.0/#maxwidth-attribute)
Java properties

Native Java format for translations.
Java properties are usually used as monolingual translations.

Weblate supports ISO-8859-1, UTF-8 and UTF-16 variants of this format. All of them support storing all Unicode characters, it is just differently encoded. In the ISO-8859-1, the Unicode escape sequences are used (for example zkou\u0161ka), all others encode characters directly either in UTF-8 or UTF-16.

Loading escape sequences works in UTF-8 mode as well, so please be careful choosing the correct encoding set to match your application needs.

Typical Weblate Component configuration

```properties
src/app.Bundle_*.properties
src/app.Bundle.properties
Empty
Java Properties (ISO-8859-1)
```

mi18n lang

File format used for JavaScript localization by mi18n. Syntactically it matches Java properties.

Typical Weblate Component configuration

```lang
*.lang
en-US.lang
Empty
mi18n lang
```

GWT

Native GWT format for translations.
GWT properties are usually used as monolingual translations.

Typical Weblate Component configuration

```properties
src/app(Bundle_*.properties
src/app.Bundle.properties
Empty
GWT Properties
```

GWT localization guide, GWT Internationalization Tutorial, Mozilla and Java properties files
INI translations

INI file format for translations.
INI translations are usually used as monolingual translations.

Typical Weblate Component configuration

```
language/*.ini
language/en.ini
Empty
INI File
```

INI File:

Weblate only extracts keys from sections within an INI file. In case your INI file lacks sections, you might want to use Joomla translations or Java properties instead.

INI Files:

Java properties
Joomla translations
Inno Setup INI

Inno Setup INI

INI file format for translations.

Inno Setup INI

The only notable difference to INI translations is in supporting %n and %t placeholders for line break and tab.

Typical Weblate Component configuration

```
language/*.islu
language/en.islu
Empty
Inno Setup INI File
```

INI File:

Only Unicode files (.islu) are currently supported, ANSI variant (.isl) is currently not supported.

INI Files:

Joomla translations
INI translations

Joomla translations

Native Joomla format for translations.
Joomla translations are usually used as monolingual translations.

Typical Weblate Component configuration

```
language/*/com_foobar.ini
language/en-GB/com_foobar.ini
Empty
Joomla Language File
```

INI File:

Specification of Joomla language files
Mozilla and Java properties files
INI translations
Inno Setup INI
Qt Linguist .ts
Translation format used in Qt based applications.
Qt Linguist files are used as both bilingual and monolingual translations.

Typical Weblate Component configuration when using as bilingual

```
# i18n/app.*.ts
Empty
# i18n/app.de.ts
Qt Linguist Translation File
```

Typical Weblate Component configuration when using as monolingual

```
i18n/app.*.ts
i18n/app.en.ts
i18n/app.en.ts
Qt Linguist Translation File
```

Qt Linguist manual

Android string resources

Android specific file format for translating applications.
Android string resources are monolingual, the `res/values/strings.xml` is stored in a different location from the others.

Typical Weblate Component configuration

```
# res/values-*/strings.xml
# res/values/strings.xml
Empty
Android String Resource
```

Android string resources documentation

Android string resources documentation

Android string-array structures are not currently supported. To work around this, you can break your string arrays apart:

```
<string-array name="several_strings">
  <item>First string</item>
  <item>Second string</item>
</string-array>
```

become:

```
<string-array name="several_strings">
  <item>@string/several_strings_0</item>
  <item>@string/several_strings_1</item>
</string-array>
<string name="several_strings_0">First string</string>
<string name="several_strings_1">Second string</string>
```

The `string-array` that points to the `string` elements should be stored in a different file, and not be made available for translation.

This script may help pre-process your existing strings.xml files and translations: https://gist.github.com/paour/11291062
Apple iOS strings

Apple specific file format for translating applications, used for both iOS and iPhone/iPad application translations. Apple iOS strings are usually used as bilingual translations.

Typical Weblate Component configuration

<table>
<thead>
<tr>
<th>Resources/*.lproj/Localizable.strings</th>
<th>Resources/en.lproj/Localizable.strings</th>
<th>Resources/Base.lproj/Localizable.strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>Empty</td>
<td>Empty</td>
</tr>
</tbody>
</table>

iOS Strings (UTF-8)

Typical Weblate Component configuration

| lang/*/texts.php | lang/en/texts.php | PHP strings |

PHP

PHP translations are usually monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Example file:

```php
<?php
$LANG['foo'] = 'bar';
$LANG['foo1'] = 'foo bar';
$LANG['foo2'] = 'foo bar baz';
$LANG['foo3'] = 'foo bar baz bag';
```

Laravel PHP

Laravel 4.1

The Laravel PHP localization files are supported as well with plurals:

```php
return [
    'welcome' => 'Welcome to our application',
    'apples' => 'There is one apple|There are many apples',
];
```

JSON files

Since Weblate 2.16 and with translate-toolkit at-least 2.2.4, nested structure JSON files are supported as well.

4.3: The structure of JSON file is properly preserved even for complex situations which were broken in prior releases.

JSON format is used mostly for translating applications implemented in JavaScript.

Weblate currently supports several variants of JSON translations:

Simple key / value files, used for example by vue-i18n or react-intl.

Files with nested keys.
JSON i18next files

go-i18n JSON files

WebExtension JSON

ARB File

JSON translations are usually monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

Example file:

```json
{
  "Hello, world!": "Ahoj světe!",
  "Orangutan has %d banana": ",
  "Try Weblate at https://demo.weblate.org/!
  "Thank you for using Weblate.": 
}
```

Nested files are supported as well (see above for requirements), such a file can look like:

```json
{
  "$weblate": {
    "$hello": "Ahoj světe!
    "$orangutan": "",
    "$try": "",
    "$thanks": ""
  }
}
```

**Note:** The JSON file and JSON nested structure file can both handle same type of files. Both preserve existing JSON structure when translating.

The only difference between them is when adding new strings using Weblate. The nested structure format parses the newly added key and inserts the new string into the matching structure. For example `app.name` key is inserted as:

```json
{
  "$app": {
    "$name": "Weblate"
  }
}
```

**Typical Weblate Component configuration**

langs/translation-*.json
langs/translation-en.json
Empty

JSON nested structure file

**Note:** Since Weblate 2.17 and with translate-toolkit at-least 2.2.5, i18next JSON files with plurals are supported as well.

i18next is an internationalization framework written in and for JavaScript. Weblate supports its localization files with features such as plurals.

i18next translations are monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

**Note:** Weblate & i18next JSON v3

Example file:
```json
{
  "hello": "Hello",
  "apple": "I have an apple",
  "apple_plural": "I have {count} apples",
  "apple_negative": "I have no apples"
}
```

**Typical Weblate Component configuration**

```
langs/*.json
langs/en.json
Empty
```

**go-i18n JSON files**

**4.1.**

Go-i18n translations are monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

**Weblate** supports the go-i18n JSON v1 format, for flat JSON formats please use JSON files. The v2 format with hash is currently not supported.

```
langs/*.json
langs/en.json
Empty
```

**ARB File**

**4.1.**

ARB translations are monolingual, so it is recommended to specify a base file with (what is most often the) English strings.

```
lib/l10n/intl_*.arb
lib/l10n/intl_en.arb
Empty
```

**Application Resource Bundle Specification**

**Internationalizing Flutter apps**

```
lib/l10n/intl_*.arb
lib/l10n/intl_en.arb
Empty
```

67
WebExtension JSON

This is supported since Weblate 2.16 and with translate-toolkit at least 2.2.4.
File format used when translating extensions for Mozilla Firefox or Google Chromium.

While this format is called JSON, its specification allows to include comments, which are not part of JSON specification. Weblate currently does not support file with comments.

Example file:

```json
{
    "hello": {
        "message": "Ahoj světe!\n",
        "description": "Description",
        "placeholders": {
            "url": {
                "content": "$1",
                "example": "https://developer.mozilla.org"
            }
        }
    },
    "orangutan": {
        "message": "",
        "description": "Description"
    },
    "try": {
        "message": "",
        "description": "Description"
    },
    "thanks": {
        "message": "",
        "description": "Description"
    }
}
```

Typical Weblate Component configuration

- locales/*/messages.json
- locales/en/messages.json
- Empty
- WebExtension JSON file

**Note:**

JSON

Google Chrome.i18n

Mozilla Extensions Internationalization

.XML resource files

This .XML resource (.resx) file employs a monolingual XML file format used in Microsoft .NET applications. It is interchangeable with .resw, when using identical syntax to .resx.

Typical Weblate Component configuration

- Resources/Language.*.resx
- Resources/Language.resx
- Empty
- .NET resource file

**Note:**

.NET Resource files (.resx) updating-target-files: addon-weblate.cleanup.generic
CSV files

2.4 CSV files can contain a simple list of source and translation. Webate supports the following files:

- Files with header defining fields (location, source, target, ID, fuzzy, context, translator_comments, developer_comments). This is the recommended approach, as it is the least error prone. Choose CSV file as a file format.
- Files with two fields—source and translation (in this order). Choose Simple CSV file as a file format.
- Headerless files with fields in order defined by the translate-toolkit: location, source, target, ID, fuzzy, context, translator_comments, developer_comments. Choose CSV file as a file format.

Remember to define when your files are monolingual (see instructions).

Note: The CSV format currently automatically detects the dialect of the CSV file. In some cases the automatic detection might fail and you will get mixed results. This is especially true for CSV files with newlines in the values. As a workaround it is recommended to omit quoting characters.

Example file:

| Thank you for using Weblate.,Děkujeme za použití Weblate. |

Typical Weblate Component configuration for bilingual CSV

locale/*.csv
locale/en.csv
locale/en.csv

Typical Weblate Component configuration for monolingual CSV

locale/*.csv
locale/en.csv
locale/en.csv

YAML files

2.9 YAML files

The plain YAML files with string keys and values. Webate also extract strings from lists or dictionaries.

Example of a YAML file:

```
weblate:  
  hello: "Thank you for using Weblate.",  
  orangutan": "Děkujeme za použití Weblate.",  
  try": "Try this instead",  
  thanks": ""  
```

Typical Weblate Component configuration

translations/messages.*.yml
translations/messages.en.yml
Empty

YAML file

Ruby YAML files
Ruby YAML files

Example Ruby i18n YAML file:

```ruby
.cs:
  weblate:
    hello: ""
    orangutan: ""
    try: ""
    thanks: ""
```

Typical Weblate Component configuration

- translations/messages.*.yml
- translations/messages.en.yml

Typical Weblate Component configuration

Ruby YAML file

YAML files

DTD files

Example DTD file:

```xml
<!ENTITY hello "" >
<!ENTITY orangutan "" >
<!ENTITY try "" >
<!ENTITY thanks "" >
```

Typical Weblate Component configuration

locale/*.dtd
locale/en.dtd

DTD file

Mozilla DTD format

Flat XML files

Example of a flat XML file:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<root>
  <str key="hello_world">Hello World!</str>
  <str key="resource_key">Translated value.</str>
</root>
```

Typical Weblate Component configuration

locale/*.xml
locale/en.xml

Flat XML file

Flat XML
Windows RC files

Support for Windows RC files has been rewritten.

Support for this format is currently in beta, feedback from testing is welcome.

Example Windows RC file:

```
LANGUAGE LANG_CZECH, SUBLANG_DEFAULT
STRINGTABLE
BEGIN
  IDS_MSG1  "Hello, world!\n"
  IDS_MSG2  "Orangutan has %d banana.\n"
  IDS_MSG3  "Try Weblate at http://demo.weblate.org/!\n"
  IDS_MSG4  "Thank you for using Weblate."
END
```

Typical Weblate Component configuration

```
lang/*.rc
lang/en-US.rc
```

Metadata used for publishing apps in various app stores can be translated. Currently the following tools are compatible:

- Triple-T gradle-play-publisher
- Fastlane
- F-Droid

The metadata consists of several textfiles, which Weblate will present as separate strings to translate.

Typical Weblate Component configuration

```
fastlane/android/metadata/*
fastlane/android/metadata/en-US
```

Subtitle files

Webmate SubRip subtitle file (*.srt)
MicroDVD subtitle file (*.sub)
Advanced Substation Alpha subtitles file (*.ass)
Substation Alpha subtitle file (*.ssa)
Typical Weblate Component configuration

/*srt*/
en.srt
/en.srt
SubRip subtitle file

### Subtitles

**Excel Open XML**

**3.2**

Excel Open XML (.xlsx) files can be imported and exported. When uploading XLSX files for translation, be aware that only the active worksheet is considered, and there must be at least a column called `source` (which contains the source string) and a column called `target` (which contains the translation). Additionally there should be the column called `context` (which contains the context path of the translation string). If you use the XLSX download for exporting the translations into an Excel workbook, you already get a file with the correct file format.

**HTML**

**4.1**

---

Support for this format is currently in beta, feedback from testing is welcome.

---

The translatable content is extracted from the HTML files and offered for the translation.

**4.6**

---

Support for this format is currently in beta, feedback from testing is welcome.

---

The translatable content is extracted from the plain text files and offered for the translation. Each paragraph is translated as a separate string.

**3**

DokuWiki

MediaWiki

---

Simple Text Documents
OpenDocument Format

Support for this format is currently in beta, feedback from testing is welcome.

The translatable content is extracted from the OpenDocument files and offered for the translation.

OpenDocument Format

IDML Format

Support for this format is currently in beta, feedback from testing is welcome.

The translatable content is extracted from the Adobe InDesign Markup Language files and offered for the translation.

TermBase eXchange

TBX Wikipedia

Stringsdict format

Support for this format is currently in beta, feedback from testing is welcome.

XML based format used by Apple which is able to store plural forms of a string.

Stringsdict File Format
Fluent format

Support for this format is currently in beta, feedback from testing is welcome.

Fluent is a monolingual text format that focuses on asymmetric localization: a simple string in one language can map to a complex multi-variant translation in another language.

Typical Weblate Component configuration

```
locales/*/messages.ftl
locales/empty
Empty Fluent file
```

Translation Related File Formats

VCS: Weblate URL: https://github.com/WeblateOrg/weblate.git

Hosted Weblate

GitHub Bitbucket Codeberg GitLab

Weblate push user git@github.com:WeblateOrg/weblate.git
SSH

Weblate

SSH keys

Weblate uses SSH key to access remote repositories. The corresponding public key is found below, you can use it to grant Weblate access to a repository.

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAADQc8oVJi3RpfHwkJkMT2iQpTZsYqPmpjMxum4M0RfSi
zr7vzRyLJlqSnAmYIMIYbERn9wOId/YKwR03Q8w2yeCU05FQlu1g86U2uQKk/iMzPyU+v4a6
BUra/NNQh1R5pS3j/4wMTp8R2w3Mm6ygs1EJvR4+XeMjPAaCtR2xLkGk7lRq6ZnGkV15
```

Download private key

Known host keys

```
Hostname   Key type   Fingerprint
github.com ssh-rsa   nThbq6XuiJGW/ITEgOCspRbimTvdCARIuNhGW6ESS9B
```

Add host key

To access SSH hosts, its host key needs to be verified. You can get the host key by entering a domain name or IP for the host in the form below.

<table>
<thead>
<tr>
<th>Hostname</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Submit

Weblate SSH

Weblate About GitHub 1

Hosted Weblate

SSH About GitHub 1 Weblate

SSH keys

Weblate
SSH

Weblate SSH

SSH:
gitlab.com

GitHub SSH:

Creating an access token for command-line use

Hosted Weblate
Removing main component also removes linked components.

HTTPS

GitHub URL: https://user:your_access_token@github.com/WeblateOrg/weblate.git

HTTP/HTTPS VCS:
```
git config --global http.proxy http://user:password@proxy.example.com:80
```

The cURL manpage

Git

Weblate needs Git 2.12 or newer.
Git

Weblate VCS

Home=$DATA_DIR/home

DATA_DIR

DATA_DIR/home/.git

remote helpers

Bazaar & Mercurial & GitHub & GitHub:
git-remote-hg & git-remote-bzr

Bazaar & Launchpad & gnuhhello

bzr::lp:gnuhello

Mercurial selenic.com & hello

hg::http://selenic.com/repo/hello

GitHub

2.3 API

Git GitHub API

GitHub API

GitHub API

Pushing changes from Weblate

GitHub

API

1 API

API

GITHUB_USERNAME=GITHUB_TOKEN=GITHUB_CREDENTIALS
GitLab

GitLab 3.9

GitLab API

GitLab: Pushing changes from Weblate

GitLab

Git

GitLab

Pagure

Pagure 4.3.2

Pagure API

Pagure: Pushing changes from Weblate

Pagure

Git

Pagure

Gerrit

Gerrit 2.2

git-review

Gerrit

Gerrit

Mercurial

Mercurial 2.1

Mercurial Weblate VCS

Mercurial

Mercurial

79
Subversion

2.8

Weblate ➔ git-svn ➔ subversion ➔ Git ➔ Perl

Weblate ➔ Subversion ➔ URL

branches/tags/trunk/ URL

git-svn documentation

2.19

Weblate Subversion URL

branches/tags/trunk/ URL

Git

Underneath, this uses Git. It requires Git installed and allows you to switch to using Git natively with full history of your translations.

3.8

Weblate ➔ VCS

Weblate ➔ Git ➔ VCS ➔ Weblate

WEBLATE ➔ REST API

2.6

API ➔ /api/ ➔ URL ➔ Django REST framework

API ➔ /api/ ➔ URL ➔ Django REST framework

ANY /

format -- REST ➔ Accept — json ➔ REST ➔ Accept — json

page -- Returns given page of paginated results (use next and previous fields in response to automate the navigation).

Accept --

Authorization -- optional token to authenticate as Authorization: Token YOUR-TOKEN

80
Content-Type --
Accept --
Allow --
detail(string) --
count(int) --
next(string) --
previous(string) --
results(array) --
url(string) --
web_url(string) --

200 OK --

GET /api/ HTTP/1.1
Host: example.com
Accept: application/json, text/javascript
Authorization: Token YOUR-TOKEN

HTTP/1.0 200 OK
Date: Fri, 25 Mar 2016 09:46:12 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, HEAD, OPTIONS

{
    "projects":"http://example.com/api/projects/",
    "components":"http://example.com/api/components/",
    "translations":"http://example.com/api/translations/",
    "languages":"http://example.com/api/languages/"
}

CURL command:
curl \
   -H "Authorization: Token TOKEN" \
   https://example.com/api/
POST `application/x-www-form-urlencoded` JSON `application/json`

```
POST /api/projects/hello/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Token TOKEN
operation=pull
```

```
POST /api/projects/hello/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"operation":"pull"}
```

CURL:

```
```

CURL JSON:

```
```

API

```
API 100 / 5000 / settings.py
Throttling in Django REST framework documentation
```

```
X-RateLimit-Limit
X-RateLimit-Remaining
X-RateLimit-Reset
```

82
GET /api/

API: GET /api/ HTTP/1.1
Host: example.com
Accept: application/json, text/javascript
Authorization: Token YOUR-TOKEN

HTTP/1.0 200 OK
Date: Fri, 25 Mar 2016 09:46:12 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, HEAD, OPTIONS

{
    "projects": "http://example.com/api/projects/",
    "components": "http://example.com/api/components/",
    "translations": "http://example.com/api/translations/",
    "languages": "http://example.com/api/languages/"
}

GET /api/users/

GET /api/users/(str:username)/
POST /api/users/

username(string) -- JSON
full_name(string) -- JSON
email(string) -- JSON
is_superuser(boolean) -- JSON
is_active(boolean) -- JSON
date_joined(string) -- JSON
groups(array) -- JSON

GET /api/groups/(int:id)/
JSON
```json
{
    "email": "user@example.com",
    "full_name": "Example User",
    "username": "exampleusername",
    "groups": [
        "http://example.com/api/groups/2/",
        "http://example.com/api/groups/3/"
    ],
    "is_superuser": true,
    "is_active": true,
    "date_joined": "2020-03-29T18:42:42.617681Z",
    "url": "http://example.com/api/users/exampleusername/",
    "statistics_url": "http://example.com/api/users/exampleusername/
statistic"
}
```

**PUT /api/users/**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>username</code></td>
<td>string</td>
<td>JSON URL</td>
</tr>
</tbody>
</table>

**PATCH /api/users/**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>username</code></td>
<td>string</td>
<td>JSON URL</td>
</tr>
</tbody>
</table>

**DELETE /api/users/**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>username</code></td>
<td>string</td>
<td>JSON URL</td>
</tr>
</tbody>
</table>

**POST /api/users/**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>username</code></td>
<td>string</td>
<td>JSON URL</td>
</tr>
<tr>
<td><code>group_id</code></td>
<td>string</td>
<td>ID</td>
</tr>
</tbody>
</table>

**GET /api/users/**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>username</code></td>
<td>string</td>
<td>JSON URL</td>
</tr>
</tbody>
</table>

translated (int) -- |

suggested (int) -- |

uploaded (int) -- |

84
commented (int) -- データ形式
languages (int) -- データ形式
GET /api/users/(str: username)/notifications/
  
username (string) -- データ形式
POST /api/users/(str: username)/notifications/
  
username (string) -- データ形式
  JSON format
notification (string) -- データ形式
scope (int) -- データ形式
frequency (int) -- データ形式
GET /api/users/(str: username)/notifications/
int: subscription_id/
  
username (string) -- データ形式
subscription_id (int) -- ID
PUT /api/users/(str: username)/notifications/
int: subscription_id/
  
username (string) -- データ形式
subscription_id (int) -- ID
  JSON format
notification (string) -- データ形式
scope (int) -- データ形式
frequency (int) -- データ形式
PATCH /api/users/(str: username)/notifications/
int: subscription_id/
  
username (string) -- データ形式
subscription_id (int) -- ID
  JSON format
notification (string) -- データ形式
scope (int) -- データ形式
frequency (int) -- データ形式
DELETE /api/users/(str: username)/notifications/
int: subscription_id/
  
username (string) -- データ形式
subscription_id -- 
subscription_id -- int
GET /api/groups/

POST /api/groups/

name(string)--

project_selection(int)--

language_selection(int)--

GET /api/groups/(int:id)/

id(int)-- ID

name(string)--

project_selection(int)--

language_selection(int)--

roles(array)--

projects(array)--

components(array)--

componentlist (array) --

JSON:

```
{
  "name": "Guests",
  "project_selection": 3,
  "language_selection": 1,
  "url": "http://example.com/api/groups/1/",
  "roles": [
    "http://example.com/api/roles/1/",
    "http://example.com/api/roles/2/
  ],
  "languages": [
    "http://example.com/api/languages/en/",
    "http://example.com/api/languages/cs/",
  ],
  "projects": [
    "http://example.com/api/projects/demo1/",
    "http://example.com/api/projects/demo/
  ],
  "componentlist": "http://example.com/api/component-lists/new/",
  "components": [
    "http://example.com/api/components/demo/weblate/
  ]
}
```
language_selection(int) -- PATCH /api/groups/(int: id)/

id(int) -- ID

name(string) --

project_selection(int) -- DELETE /api/groups/(int: id)/

id(int) -- ID

POST /api/groups/(int: id)/roles/

id(int) -- ID

string role_id --

POST /api/groups/(int: id)/components/

id(int) -- ID

string component_id --

DELETE /api/groups/(int: id)/components/

id(int) -- ID

component_id(int) --

POST /api/groups/(int: id)/projects/

id(int) -- ID

string project_id --

DELETE /api/groups/(int: id)/projects/

id(int) -- ID

project_id(int) --

POST /api/groups/(int: id)/languages/

id(int) -- ID

string language_code --

DELETE /api/groups/(int: id)/languages/

string: language_code

id(int) -- ID

language_code(string) --

POST /api/groups/(int: id)/componentlists/
id(int) -- ID

string component_list_id -- ID

DELETE /api/groups/(int: id)/componentlists/
    int: component_list_id

id(int) -- ID

component_list_id(int) -- ID

GET /api/roles/
    GET /api/roles/(int:id)/
POST /api/roles/
    name(string)
    permissions(array)

GET /api/roles/(int: id)/
    id(int) -- ID
    JSON
    name(string)
    permissions(array)

JSON:

```json
{
    "name": "Access repository",
    "permissions": [
        "vcs.access",
        "vcs.view"
    ],
    "url": "http://example.com/api/roles/1/"
}
```

PUT /api/roles/(int: id)/
    id(int) -- ID
    name(string)
    permissions(array)

PATCH /api/roles/(int: id)/
    id(int) -- ID
    name(string)
    permissions(array)

DELETE /api/roles/(int: id)/
    id(int) -- ID

88
GET /api/languages/

POST /api/languages/

code (string) --
name (string) --
direction (string) --
plural (object) --

GET /api/languages/(string: language)/

language (string) --

code (string) --
direction (string) --
plural (object) --

aliases (object) --

JSON: 

```json
{
    "code": "en",
    "direction": "ltr",
    "name": "English",
    "plural": {
        "id": 75,
        "source": 0,
        "number": 2,
        "formula": "n \neq 1",
        "type": 1
    },
    "aliases": [
        "english",
        "en_en",
        "base",
        "source",
        "eng"
    ],
    "url": "http://example.com/api/languages/en/",
    "web_url": "http://example.com/languages/en/",
}
```

PUT /api/languages/(string: language)/

language (string) --

name (string) --
direction (string) --
plural (object) --

PATCH /api/languages/(string: language)/

language (string) --
DELETE /api/languages/(string: language) /
language(string)--
GET /api/languages/(string: language)/statistics/
language(string)--

JSON

name(string) --
direction(string)--
plural(object)--
delete JSON

DELETE /api/languages/(string: language) /
language(string)--
GET /api/languages/(string: language)/statistics/
language(string)--

JSON

total(int)--
total_words(int)--
last_change(timestamp)--
recent_changes(int)--
translated(int)--
translated_percent(float)--
translated_words(int)--
translated_words_percent(int)--
translated_chars(int)--
translated_chars_percent(int)--
total_chars(int)--
fuzzy(int)--
fuzzy_percent(int)--
failing(int)--
failing--

GET /api/projects/

GET /api/projects/(string:project) /

POST /api/projects/

name(string)--
slug(string)--
web(string)--

GET /api/projects/(string: project) /

project(string)--

name(string)--
slug(string)--
web(string)--

90
components_list_url (string) -- URL: GET /api/projects/(string:project)/components/
repository_url (string) -- URL: GET /api/projects/(string:project)/repository/
changes_list_url (string) -- URL: GET /api/projects/(string:project)/changes/
translation_review (boolean) --
source_review (boolean) --
set_language_team (boolean) -- "Language-Team"
enable_hooks (boolean) --
instructions (string) --
language_aliases (string) --

JSON:

```
{
    "name": "Hello",
    "slug": "hello",
    "url": "http://example.com/api/projects/hello/",
    "web": "https://weblate.org/",
    "web_url": "http://example.com/projects/hello/"
}
```

PATCH /api/projects/(string: project) /
PATCH /api/projects/(string: project)/repository/
PUT /api/projects/(string: project) /
DELETE /api/projects/(string: project) /

GET /api/projects/(string: project)/changes/
GET /api/changes/(int:id)/
GET /api/projects/(string: project)/repository/
GET /api/components/(string:project)/(string:component)/repository/

needs_commit (boolean) --
needs_merge (boolean) -- upstream
needs_push (boolean) --

JSON:

```
91
```
null

POST /api/projects/(string: project)/repository/

VCS

- project (string) -- URL
- operation (string) -- push, pull, commit, reset, cleanup, file-sync

result (boolean)

CURL:

curl \
-d operation=pull \
-H "Authorization: Token TOKEN" \
http://example.com/api/projects/hello/repository/

JSON:

POST /api/projects/hello/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"operation":"pull"}

HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{"result":true}

GET /api/projects/(string: project)/components/

- results (array) -- GET /api/components/(string:project)/(string:component)/

POST /api/projects/(string: project)/components/

4.6: The cloned repositories are now automatically shared within a project using Weblate URL. Use disable_autoshare to turn off this.
project (string) -- URL
file zipfile -- Weblate ZIP
file docfile --
boolean disable_autoshare -- Disables automatic repository sharing via Weblate URL.
JSON result (object) -- GET /api/components/(string:project)/

JSON can not be used when uploading the files using the zipfile and docfile parameters. The data has to be uploaded as multipart/form-data.

CURL:
```
curl \
---form docfile=@strings.html \
---form name=Weblate \
---form slug=weblate \
---form file_format=html \
---form new_lang=add \
-H "Authorization: Token TOKEN" \
http://example.com/api/projects/hello/components/
```

CURL JSON:
```
curl \
---data-binary '{
  "branch": "main",
  "file_format": "po",
  "filemask": "po/*.po",
  "git_export": ",",
  "license": ",",
  "license_url": ",",
  "name": "Weblate",
  "slug": "weblate",
  "repo": "file:///home/nijel/work/weblate-hello",
  "template": "",
  "new_base": "",
  "vcs": "git"
}' \
-H "Content-Type: application/json" \
-H "Authorization: Token TOKEN" \
http://example.com/api/projects/hello/components/
```

JSON:
```
POST /api/projects/hello/components/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{
  "branch": "main",
  "file_format": "po",
  "filemask": "po/*.po",
  "git_export": "",
  "license": "",
  "license_url": "",
  "name": "Weblate",
  "slug": "weblate",
```
GET /api/projects/(string: project)/languages/

HTTP 3.8 OK

project (string) -- JSON

results (array) -- JSON

language (string) -- JSON

code (string) -- JSON

total (int) -- JSON

translated (int) -- JSON

translated_percent (float) -- JSON

total_words (int) -- JSON

translated_words (int) -- JSON

words_percent (float) -- JSON
GET /api/projects/(string: project)/statistics/

project (string) -- URL

GET /api/components/

GET /api/components/(string:project)/

GET /apicomponents/(string:project)/

GET /api/components/(string:project)/

PROJECT (string) -- URL

project (string) -- URL

GET /api/languages/(string:language)/

push (string) -- URL

check_flags (string) --

priority (string) --

enforced_checks (string) --

95
restricted (string) --
repoweb (string) --
report_source_bugs (string) --
merge_style (string) --
commit_message (string) -- Commit, add, delete, merge and addon messages
add_message (string) -- Commit, add, delete, merge and addon messages
delete_message (string) -- Commit, add, delete, merge and addon messages
merge_message (string) -- Commit, add, delete, merge and addon messages
addon_message (string) -- Commit, add, delete, merge and addon messages
allow_translation_propagation (string) --
able_suggestions (string) --
suggestion_voting (string) --
suggestion_autoaccept (string) --
push_on_commit (string) --
commitPendingAge (string) --

auto_lock_error (string) --
language_regex (string) --
variant_regex (string) --
repository_url (string) -- URL: GET /api/components/(string:project)/(string:component)/repository/
translations_url (string) -- URL: GET /api/components/(string:project)/(string:component)/translations/
lock_url (string) -- URL: GET /api/components/(string:project)/(string:component)/lock/
changes_list_url (string) -- URL: GET /api/components/(string:project)/(string:component)/changes/
task_url (string) -- URL: GET /api/tasks/(str:uuid)/

JSON:

```json
{
"branch": "main",
"file_format": "po",
"filemask": "po/*.po",
"git_export": "",
"license": "",
"license_url": "",
"name": "Weblate",
"slug": "weblate",
"project": {
  "name": "Hello",
  "slug": "hello",
  "source_language": {
    "code": "en",
    "direction": "ltr",
    "name": "English",
    "url": "http://example.com/api/languages/en/",
    "web_url": "http://example.com/languages/en/"
  },
  "url": "http://example.com/api/projects/hello/",
  "web": "https://weblate.org/",
  "web_url": "http://example.com/projects/hello/"
},
"source_language": {
  "code": "en",
  "direction": "ltr",
  "name": "English",
  "url": "http://example.com/api/languages/en/"
}
}
```
PATCH /api/components/(string: project)/
  string: component/ PATCH

project (string) -- URL
  component (string) -- URL

source_language (string) -- JSON
  name (string)
  slug (string)

repo (string) -- VCS URL

CURL:

curl \
  --data-binary '{"name": "new name"}' \
  -H "Content-Type: application/json" \
  -H "Authorization: Token TOKEN" \
  PATCH http://example.com/api/projects/hello/components/

JSON:

 PATCH /api/projects/hello/components/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

  "name": "new name"

JSON:

HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGI Server/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

  "branch": "main",
  "file_format": "po",
  "filemask": "po/*.po",
  "git_export": "",
  "license": "",
  "license_url": "",
  "name": "new name",
  "slug": "weblate",
  "project": {
    "name": "Hello",
    "slug": "hello",
  }
GET /api/components/(string: project)/
string: component/lock/

project (string) -- URL
component (string) -- URL
JSON
locked (boolean) --

JSON response:

```json
{
  "locked": false
}
```

POST /api/components/(string: project)/
string: component/lock/

```
GET /api/components/(string:project)/(string:component)/lock/

project (string) -- URL
component (string) -- URL
JSON
lock --
```

CURL:

```bash
curl 
  -d lock=true 
  -H "Authorization: Token TOKEN" 
  http://example.com/api/components/hello/weblate/repository/
```

JSON response:

```
POST /api/components/hello/weblate/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"lock": true}
```

HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{"locked":true}

GET /api/components/(string: project)/
string: component/repository/

```
GET /api/projects/(string:project)/repository/

project (string) -- URL
component (string) -- URL
JSON
needs_commit (boolean) --
needs_merge (boolean) --
needs_push (boolean) --
```

99
remote_commit (string) -- URL
status (string) -- VCS
merge_failure -- null

POST /api/components/(string: project)/
string: component/repository/VCS

POST /api/projects/(string:project)/repository/
project (string) -- URL
component (string) -- URL

operation (string) -- push, pull, commit, reset, cleanup

result (boolean) --

CURL例:
curl -d operation=pull 
    -H "Authorization: Token TOKEN"
    http://example.com/api/components/hello/weblate/repository/

JSON头部:
POST /api/components/hello/weblate/repository/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"operation":"pull"}

JSON头部:
HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{"result":true}
results (array) -- GET /api/translations/(string:project)/(string:component)/(string:language)/

POST /api/components/(string: project)/
string: component/translations/

project (string) -- URL

component (string) -- URL

language_code (string) -- GET /api/languages/(string:language)/

result (object) --

CURL:

curl \
-d language_code=cs \ 
-H "Authorization: Token TOKEN" \ 
http://example.com/api/projects/hello/components/

JSON SUCCESS:

POST /api/projects/hello/components/ HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Token TOKEN
Content-Length: 20

{"language_code": "cs"}

JSON RESPONSE:

HTTP/1.0 200 OK
Date: Tue, 12 Apr 2016 09:32:50 GMT
Server: WSGIServer/0.1 Python/2.7.11+
Vary: Accept, Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Language: en
Allow: GET, POST, HEAD, OPTIONS

{
    "failing_checks": 0,
    "failing_checks_percent": 0,
    "failing_checks_words": 0,
    "filename": "po/cs.po",
    "fuzzy": 0,
    "fuzzy_percent": 0.0,
    "fuzzy_words": 0,
    "have_comment": 0,
    "have_suggestion": 0,
    "is_template": false,
    "is_source": false,
    "language": {
        "code": "cs",
        "direction": "ltr",
        "name": "Czech",
        "url": "http://example.com/api/languages/cs/",
        "web_url": "http://example.com/languages/cs/"
    },
    "language_code": "cs",
    "id": 125,
    "last_author": null,
    "last_change": null,
    "share_url": "http://example.com/engage/hello/cs/",
    "total": 4,
    "total_words": 15,
}
GET /api/components/(string: project)/
string: component/statistics/ [GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/]

project (string) -- URL
component (string) -- URL

GET /api/components/(string: project)/
string: component/links/ [GET /api/translations/(string:project)/(string:component)/(string:language)/]

project (string) -- URL
component (string) -- URL

DELETE /api/components/(string: project)/
string: component/links/string: project_slug

project (string) -- URL
component (string) -- URL

GET /api/translations/ [GET /api/translations/(string:project)/(string:component)/(string:language)/]
GET /api/translations/(string: project)/
string: component/string: language

project (string) -- URL
component (string) -- URL
language (string) -- URL

102
```json
{
    "component": {
        "branch": "main",
        "file_format": "po",
        "filemask": "po/*.po",
        "git_export": "",
        "license": "",
        "license_url": "",
        "name": "Weblate",
        "new_base": "",
        "project": {
            "name": "Hello",
            "slug": "hello",
            "source_language": {
                "code": "en",
                "direction": "ltr",
                "name": "English",
                "url": "http://example.com/api/languages/en/",
                "web_url": "http://example.com/languages/en/"
            }
        }
    }
}
```
**POST /api/translations/**

* `project` (string) -- **URL**

* `component` (string) -- **URL**

* `language` (string) -- **URL**

* `q` (string) -- **JSON**

Results (array) -- **see GET /api/units/(int:id)/**

**POST /api/translations/(string: project)/**

* `string` (project/component/language/units) -- **GET**

**POST /api/translations/**

* `project` (string) -- **URL**

* `component` (string) -- **URL**

* `language` (string) -- **URL**

**POST /api/translations/(string: project)/**

* `string` (project/component/language/autotranslate) -- **GET**

**POST /api/translations/**

* `project` (string) -- **URL**

* `component` (string) -- **URL**

* `language` (string) -- **URL**

**mode** (string) -- **options**

**filter_type** (string) -- **options**

**auto_source** (string) -- **mt** or **others**

**POST /api/translations/(string: project)/**

* `string` (project/component/language/file) -- **Download current translation file as it is stored in the VCS (without the format parameter) or converted to another format (see **format**).**

---

**Note:** This API endpoint uses different logic for output than rest of API as it operates on whole file rather than on data. Set of accepted **format** parameter differs and without such parameter you get translation file as stored in VCS.

**format** -- File format to use; if not specified no format conversion happens; supported file formats: **po**,**mo**,**xliff**,**xliff11**,**tbx**,**csv**,**xlsx**,**json**,**aresource**,**strings**

**POST /api/translations/**

* `project` (string) -- **URL**

* `component` (string) -- **URL**

* `language` (string) -- **URL**

**POST /api/translations/(string: project)/**

* `string` (project/component/language/file) -- **GET**

**string conflict** -- **ignore, replace-translated, replace-approved**
file file
string email --
string author --
string method -- translate, approve, suggest, fuzzy, replace, source, add
string fuzzy -- `process`` approve``
CURL:

curl -X POST \
- F file=@strings.xml \ 
- H "Authorization: Token TOKEN" \
 http://example.com/api/translations/hello/android/cs/file/

GET /api/translations/(string: project) /
string: component/string: language/repository/ 

GET /api/components/(string:project)/(string:component)/repository/

project (string) -- URL 
component (string) -- URL 
language (string) -- URL

POST /api/translations/(string: project) /
string: component/string: language/repository/VCS

POST /api/projects/(string:project)/repository/

project (string) -- URL 
component (string) -- URL 
language (string) -- JSON 

operation (string) -- push, pull, commit, reset, cleanup 

result (boolean) -- 

GET /api/translations/(string: project) /
string: component/string: language/statistics/ 

2.7 2.7

project (string) -- URL 
component (string) -- URL 
language (string) -- 

code (string) -- 
failing (int) -- 
failing_percent (float) -- 
fuzzy (int) -- 
fuzzy_percent (float) -- 
total_words (int) -- 
translated_words (int) -- 
last_author (string) -- 
last_change (timestamp) -- 
name (string) -- 
total (int) -- 
translated (int) -- 
translated_percent (float) -- 

106
url(string) -- URL
url_translate(string) -- URL

unit

GET /api/units/

GET /api/units/(int:id)/

id(int) -- ID

translation(string) -- URL

source(array) --

previous_source(string)

target(array)

id_hash(string)

content_hash(string)

location(string)

context(string)

note(string)

flags(string)

state(int) -- 0 - 10 - 20 - 30 - 100 -

fuzzy(boolean) -- "fuzzy"

translated(boolean)

approved(boolean)

position(int)

has_suggestion(boolean)

has_comment(boolean)

has_failing_check(boolean)

num_words(int)

priority(int) -- 100

id(int)

explanation(string) -- Additional info on source strings

extra_flags(string) --

web_url(string) -- URL

source_unit(string)

PATCH /api/units/(int:id)/

id(int) -- ID

state(int) -- 0 - 10 - 20 - 30 - 100 -
PUT /api/units/(int:id)/

DELETE /api/units/(int:id)/

GET /api/changes/(int:id)/

GET /api/changes/

id(int) -- ID

state(int) -- 0 - 10 - 20 - 30 -

extra_flags(string) --

user(string) --

action(int) --

timestamp_after(timestamp) -- ISO 8601

timestamp_before(timestamp) -- ISO 8601

unit(string) -- URL

translation(string) -- URL

component(string) -- URL

author(string) -- URL

 timestamp(timestamp) --

action(int) --

action_name(string) --

target(string) --

id(int) --
GET /api/screenshots/

GET /api/screenshots/(int:id)/

id(int) -- ID
name(string) -- JSON Name
component(string) -- URL
file_url(string) -- URL
units(array) -- 

GET /api/screenshots/(int:id)/file/

id(int) -- ID

POST /api/screenshots/(int:id)/file/

file image--

curl -X POST
-F image=@image.png
-H "Authorization: Token TOKEN"
http://example.com/api/screenshots/1/file/

POST /api/screenshots/(int:id)/units/

id(int) -- ID
name(string) --
unit_id--

GET /api/screenshots/(int:id)/file/

units(array) -- 

DELETE /api/screenshots/(int:id)/units/

id(int)--
unit_id--

POST /api/screenshots/

file image--
string name--
string project_slug--
string component_slug

string language_code

name(string)

component(string)

file_url(string)

units(array)

PATCH /api/screenshots/(int:id)/

id(int) -- ID

PUT /api/screenshots/(int:id)/

DELETE /api/screenshots/(int:id)/

id(int) -- ID

GET /api/addons/

id(int) -- ID

name(string)

component(string)

file_url(string)

units(array)

DELETE /api/screenshots/(int:id)/

id(int) -- ID

POST /api/components/(string: project)/

name (string) -- required
configuration (object) -- required

PATCH /api/addons/(int: id)/

id (int) -- required ID

configuration (object) -- required

PUT /api/addons/(int: id)/

id (int) -- required ID

configuration (object) -- required

DELETE /api/addons/(int: id)/

id (int) -- required ID

4.0 API

GET /api/component-lists/

GET /api/component-lists/(str: slug)/

slug (string) -- required
name (string) -- required

GET /api/component-lists/(str: slug)/

slug (string) -- required
name (string) -- required

show_dashboard (boolean) -- required

components (array) -- required

auto_assign (array) -- required

PUT /api/component-lists/(str: slug)/

slug (string) -- required
name (string) -- required

show_dashboard (boolean) -- required

PATCH /api/component-lists/(str: slug)/

slug (string) -- required
name (string) -- required

slug (string) -- required

111
show_dashboard(boolean) --

DELETE /api/component-lists/(str: slug)/
slug(string) --

POST /api/component-lists/(str: slug)/components/
slug(string) --
string component_id -- ID

DELETE /api/component-lists/(str: slug)/components/
str: component_slug
slug(string) --
component_slug(string) --

GET /api/tasks/
GET /api/tasks/(str: uuid)/

uuid(string) -- UUID
completed(boolean) --
progress(int) --
result(object) --
log(string) --

Metrics

GET /api/metrics/
Returns server metrics.

units(int) --
units_translated(int) --
users(int) --
changes(int) --
projects(int) --
components"(int) --
translations"(int) --
languages"(int) --
checks"(int) --
configuration_errors"(int) --
suggestions"(int) --
Notification hooks allow external applications to notify Weblate that the VCS repository has been updated. You can use repository endpoints for projects, components and translations to update individual repositories; see POST /api/projects/(string:project)/repository/ for documentation. 

**GET /hooks/update/(string: project)/**

String: component/ Please use POST /api/components/(string:project)/(string:component)/repository/ instead which works properly with authentication for ACL limited projects.

**GET /hooks/update/(string: project)/**

VCS 2.6. Please use POST /api/projects/(string:project)/repository/ instead which works properly with authentication for ACL limited projects.

**POST /hooks/github/**

Special hook for handling GitHub notifications and automatically updating matching components.

**POST /hooks/gitlab/**

Special hook for handling GitLab notifications and automatically updating matching components.

**POST /hooks/bitbucket/**

Special hook for handling Bitbucket notifications and automatically updating matching components.

**POST /hooks/pagure/**

Pagure 3.3. Pagure

**POST /hooks/azure/**

Pagure 3.8. Pagure
Special hook for handling Azure Repos notifications and automatically updating matching components.

**How:**

*Automatically receiving changes from Azure Repos*

For instruction on setting up Azure integration

Generic information about Azure Repos Web Hooks
**ENABLE_HOOKS**

For enabling hooks for whole Weblate

**POST /hooks/gitea/**

Special hook for handling Gitea Webhook notifications and automatically updating matching components.

**How:**

*Automatically receiving changes from Gitea Repos*

For instruction on setting up Gitea integration
https://docs.gitea.io/en-us/webhooks/

Generic information about Gitea Webhooks
**ENABLE_HOOKS**

For enabling hooks for whole Weblate

**POST /hooks/gitee/**

Special hook for handling Gitee Webhook notifications and automatically updating matching components.

**How:**

*Automatically receiving changes from Gitee Repos*

For instruction on setting up Gitee integration
https://gitee.com/help/categories/40

Generic information about Gitee Webhooks
**ENABLE_HOOKS**

For enabling hooks for whole Weblate

**Exports**

Weblate provides various exports to allow you to further process the data.

**GET /exports/stats/**

(string: project) /

(string: component) /

(format (string) -- Output format: either json or csv)

**2.6:** Please use **GET /api/components/** (string:project) /

(string:component) /

/statistics/ and **GET /api/translations/** (string:project) /

(string:component) /

(string:language) /statistics/ instead; it allows access to ACL controlled projects as well.

Retrieves statistics for given component in given format.

**Example request:**

<table>
<thead>
<tr>
<th>GET /exports/stats/weblate/main/</th>
<th>HTTP 1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host: example.com</td>
<td></td>
</tr>
<tr>
<td>Accept: application/json, text/javascript</td>
<td></td>
</tr>
</tbody>
</table>

**Example response:**

<table>
<thead>
<tr>
<th>HTTP 1.1</th>
<th>200 OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vary: Accept</td>
<td></td>
</tr>
<tr>
<td>Content-Type: application/json</td>
<td></td>
</tr>
</tbody>
</table>

```json
[
  {
    "code": "cs",
    "failing": 0,
    "failing_percent": 0.0,
    "fuzzy": 0,
    "fuzzy_percent": 0.0,
    "last_author": "Michal Čihař",
    "last_change": "2012-03-28T15:07:38+00:00",
  }
]```
Changes in translations are exported in RSS feeds.

GET /exports/rss/(string: project) / string: component/string: language/ Retrieves RSS feed with recent changes for a translation.
GET /exports/rss/(string: project) / string: component/ Retrieves RSS feed with recent changes for a component.
GET /exports/rss/(string: project) / Retrieves RSS feed with recent changes for a project.
GET /exports/rss/language/(string: language) / Retrieves RSS feed with recent changes for a language.
GET /exports/rss/ Retrieves RSS feed with recent changes for Weblate instance.

RSS Wikipedia [1]
Weblate

2.7 : Weblate 2.7 wlc API

Weblate Python

pip3 install wlc

Docker

Weblate Docker

Docker Hub: https://hub.docker.com/r/weblate/wlc

docker pull weblate/wlc

docker run --rm weblate/wlc

You might want to pass your [WLC_ARGS] to the Docker container, the easiest approach is to add your current directory as /home/weblate volume:

docker run --volume $PWD:/home/weblate --rm weblate/wlc show

wlc

~/.config/weblate for other locations

[weblate]
url = https://hosted.weblate.org/api/

[keys]
https://hosted.weblate.org/api/ = APIKEY

wlc ls
wlc commit sandbox/hello-world
wlc [arguments] <command> [options]

Weblate Python Weblate REST API Weblate
Weblate wlc wlc

--format {csv, json, text, html}
--url URL
--key KEY
--config PATH
--config-section SECTION

version
list-languages
list-projects
list-components
list-translations
show
ls
commit
pull
push
reset
cleanup
repo
Weblate
statistics
lock-status
lock
unlock
changes
download
--convert
--output
upload
--overwrite
--input

wlc ls --help

---

1.6 : .ini

Windows
XDG_CONFIG_HOME
XDG_CONFIG_DIRS
APPDATA

[weblate]
--config-section
key
Weblate
url
API URL
translation
INI
```json
[weblate]
url = https://hosted.weblate.org/api/
key = APIKEY
translation = weblate/application

API keys:
https://hosted.weblate.org/api/ = APIKEY

VCS command: $ wlc version
version: 0.1

VCS command: $ wlc list-projects
name: Hello
slug: hello
url: http://example.com/api/projects/hello/
web: https://weblate.org/
web_url: http://example.com/projects/hello/

VCS command: $ cat .weblate
[weblate]
url = https://hosted.weblate.org/api/
translation = weblate/application

VCS command: $ wlc show
branch: main
file_format: po
source_language: en
filemask: weblate/locale/*/LC_MESSAGES/django.po
git_export: https://hosted.weblate.org/git/weblate/application/
license: GPL-3.0+
license_url: https://spdx.org/licenses/GPL-3.0+
name: Application
new_base: weblate/locale/django.pot
project: weblate
repo: git://github.com/WeblateOrg/weblate.git
slug: application
template:
url: https://hosted.weblate.org/api/components/weblate/application/
vcs: git
web_url: https://hosted.weblate.org/projects/weblate/application/

VCS command: $ wlc commit
```
Weblate Python API

```bash
pip install wlc
```

`wlc`

`WeblateException`

```python
exception wlc.WeblateException
```

`Weblate`

```python
class wlc.Weblate (key='', url=None, config=None)
```

- `key` (str) -- key
- `url` (str) -- API URL
- `config` (wlc.config.WeblateConfig) -- API URL

```python
get (path)
```

- `path` (str) -- object

```python
post (path, **kwargs)
```

- `path` (str) -- object

`wlc.config`

`WeblateConfig`

```python
class wlc.config.WeblateConfig (section='wlc')
```

- `section` (str) -- XDG
- `load (path=None)`

```python
path (str) -- XDG
```

- `path` (str) -- XDG
Weblate

Docker

With dockerized Weblate deployment you can get your personal Weblate instance up and running in seconds. All of Weblate’s dependencies are already included. PostgreSQL is set up as the default database.

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- **2GB** RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Note: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

The following examples assume you have a working Docker environment, with `docker-compose` installed. Please check the Docker documentation for instructions.

1. Clone the weblate-docker repo:

   ```bash
git clone https://github.com/WeblateOrg/docker-compose.git weblate-docker
cd weblate-docker
```

2. Create a `docker-compose.override.yml` file with your settings. See [Docker environment variables](#) for full list of environment variables.
version: '3'

services:
  weblate:
    ports:
    - 80:8080
    environment:
      WEBLATE_EMAIL_HOST: smtp.example.com
      WEBLATE_EMAIL_HOST_USER: user
      WEBLATE_EMAIL_HOST_PASSWORD: pass
      WEBLATE_SERVER_EMAIL: weblate@example.com
      WEBLATE_DEFAULT_FROM_EMAIL: weblate@example.com
      WEBLATE_SITE_DOMAIN: weblate.example.com
      WEBLATE_ADMIN_PASSWORD: password for the admin user
      WEBLATE_ADMIN_EMAIL: weblate.admin@example.com

**Note:** If WEBLATE_ADMIN_PASSWORD is not set, the admin user is created with a random password shown on first startup.

The provided example makes Weblate listen on port 80, edit the port mapping in the docker-compose.override.yml file to change it.

3. Start Weblate containers:

```bash
docker-compose up
```

Enjoy your Weblate deployment, it’s accessible on port 80 of the weblate container.

**Note:**

- 2.15-2: The setup has changed recently, priorly there was separate web server container, since 2.15-2 the web server is embedded in the Weblate container.
- 3.7.1-6: In July 2019 (starting with the 3.7.1-6 tag), the containers are not running as a root user. This has changed the exposed port from 80 to 8080.

**Note:**

### Invoking management commands

### Choosing Docker hub tag

You can use following tags on Docker hub, see [https://hub.docker.com/r/weblate/weblate/tags/](https://hub.docker.com/r/weblate/weblate/tags/) for full list of available ones.

<table>
<thead>
<tr>
<th>Tag name</th>
<th>Use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>latest</td>
<td>Weblate stable release, matches latest tagged release Rolling updates in a production environment</td>
</tr>
<tr>
<td>&lt;VERSION&gt;-</td>
<td>Weblate stable release Well defined deploy in a production environment</td>
</tr>
<tr>
<td>&lt;PATCH&gt;-</td>
<td>Weblate stable release with development changes in the DockerRolling updates in a staging environment</td>
</tr>
<tr>
<td>edge</td>
<td>Weblate stable release with development changes in the DockerWell defined deploy in a staging environment</td>
</tr>
<tr>
<td>edge-&lt;DATE&gt;-</td>
<td>Weblate stable release with development changes in the DockerWell defined deploy in a staging environment</td>
</tr>
<tr>
<td>&lt;SHA&gt;-</td>
<td>Development version Weblate from GitRolling updates to test upcoming Weblate features</td>
</tr>
<tr>
<td>bleeding</td>
<td>Development version Weblate from GitWell defined deploy to test upcoming Weblate features</td>
</tr>
<tr>
<td>bleeding-&lt;DATE&gt;-&lt;SHA&gt;</td>
<td>Development version Weblate from GitWell defined deploy to test upcoming Weblate features</td>
</tr>
</tbody>
</table>

Every image is tested by our CI before it gets published, so even the bleeding version should be quite safe to use.
Docker container with HTTPS support

Please see [link](#) for generic deployment instructions, this section only mentions differences compared to it.

Using own SSL certificates

In case you have own SSL certificate you want to use, simply place the files into the Weblate data volume (see Docker container volumes):

- ssl/fullchain.pem containing the certificate including any needed CA certificates
- ssl/privkey.pem containing the private key

Both of these files must be owned by the same user as the one starting the docker container and have file mask set to 600 (readable and writable only by the owning user).

Additionally, Weblate container will now accept SSL connections on port 4443, you will want to include the port forwarding for HTTPS in docker compose override:

```yaml
version: '3'
services:
  weblate:
    ports:
      - 80:8080
      - 443:4443
```

If you already host other sites on the same server, it is likely ports 80 and 443 are used by a reverse proxy, such as NGINX. To pass the HTTPS connection from NGINX to the docker container, you can use the following configuration:

```nginx
server {
    listen 443;
    listen [::]:443;

    server_name <SITE_URL>;
    ssl_certificate /etc/letsencrypt/live/<SITE>/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/<SITE>/privkey.pem;

    location / {
        proxy_set_header HOST $host;
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_pass https://127.0.0.1:<EXPOSED_DOCKER_PORT>;
    }
}
```

Replace `<SITE_URL>`, `<SITE>` and `<EXPOSED_DOCKER_PORT>` with actual values from your environment.

Automatic SSL certificates using Let’s Encrypt

In case you want to use Let’s Encrypt automatically generated SSL certificates on public installation, you need to add a reverse HTTPS proxy an additional Docker container, `https-portal` will be used for that. This is made use of in the `docker-compose-https.yml` file. Then create a `docker-compose-https.override.yml` file with your settings:

```yaml
version: '3'
services:
  weblate:
    environment:
      WEBLATE_EMAIL_HOST: smtp.example.com
      WEBLATE_EMAIL_HOST_USER: user
      WEBLATE_EMAIL_HOST_PASSWORD: pass
      WEBLATE_SITE_DOMAIN: weblate.example.com
      WEBLATE_ADMIN_PASSWORD: password for admin user
```

(continued)
Whenever invoking `docker-compose` you need to pass both files to it, and then do:

```bash
docker-compose -f docker-compose-https.yml -f docker-compose-https.override.yml build
docker-compose -f docker-compose-https.yml -f docker-compose-https.override.yml up
```

### Upgrading the Docker container

Usually it is a good idea to only update the Weblate container and keep the PostgreSQL container at the version you have, as upgrading PostgreSQL is quite painful and in most cases does not bring many benefits.

You can do this by sticking with the existing docker-compose and just pull the latest images and then restart:

```bash
docker-compose stop
docker-compose pull
docker-compose up
```

The Weblate database should be automatically migrated on first startup, and there should be no need for additional manual actions.

**Note:** Upgrades across 3.0 are not supported by Weblate. If you are on 2.x series and want to upgrade to 3.x, first upgrade to the latest 3.0.1-x (at time of writing this it is the 3.0.1-7) image, which will do the migration and then continue upgrading to newer versions.

You might also want to update the `docker-compose` repository, though it's not needed in most cases. Please beware of PostgreSQL version changes in this case as it's not straightforward to upgrade the database, see GitHub issue for more info.

#### **FAQ**

After container setup, you can sign in as `admin` user with password provided in `WEBLATE_ADMIN_PASSWORD`, or a random password generated on first start if that was not set.

To reset `admin` password, restart the container with `WEBLATE_ADMIN_PASSWORD` set to new password.

**Note:**

`WEBLATE_ADMIN_PASSWORD`, `WEBLATE_ADMIN_NAME`, `WEBLATE_ADMIN_EMAIL`

### Number of processes and memory consumption

The number of worker processes for both uWSGI and Celery is determined automatically based on number of CPUs. This works well for most cloud virtual machines as these typically have few CPUs and good amount of memory.

In case you have a lot of CPU cores and hit out of memory issues, try reducing number of workers:

```bash
WEBLATE_WORKERS: 2
```

You can also fine-tune individual worker categories:

```bash
WEBLATE_WORKERS, CELERY_MAIN_OPTIONS, CELERY_NOTIFY_OPTIONS, CELERY_MEMORY_OPTIONS, CELERY_TRANSLATE_OPTIONS, CELERY_BACKUP_OPTIONS, CELERY_BEAT_OPTIONS, UWSGI_WORKERS
```

**Note:**

`WEBLATE_WORKERS`, `CELECY_MAIN_OPTIONS`, `CELECY_NOTIFY_OPTIONS`, `CELECY_MEMORY_OPTIONS`, `CELECY_TRANSLATE_OPTIONS`, `CELECY_BACKUP_OPTIONS`, `CELECY_BEAT_OPTIONS`, `UWSGI_WORKERS`
Scaling horizontally

You can run multiple Weblate containers to scale the service horizontally. The `/app/data` volume has to be shared by all containers, it is recommended to use cluster filesystem such as GlusterFS for this. The `/app/cache` volume should be separate for each container.

Each Weblate container has defined role using `WEBLATE_SERVICE` environment variable. Please follow carefully the documentation as some of the services should be running just once in the cluster and the ordering of the services matters as well.

You can find example setup in the docker-compose repo as `docker-compose-split.yml`.

Docker environment variables

Many of Weblate's can be set in the Docker container using environment variables:

Generic settings

`WEBLATE_DEBUG`
Configures Django debug mode using `DEBUG`.

```
environment:
  WEBLATE_DEBUG: 1
```

`WEBLATE_LOGLEVEL`
Configures the logging verbosity.

`WEBLATE_SITE_TITLE`
Changes the site-title shown in the header of all pages.

`WEBLATE_SITE_DOMAIN`

`WEBLATE_ADMIN_NAME`
`WEBLATE_ADMIN_EMAIL`
Configures the site-admin's name and e-mail. It is used for both `ADMINS` setting and creating `admin` user (see `WEBLATE_ADMIN_PASSWORD` for more info on that).

```
environment:
  WEBLATE_ADMIN_NAME: Weblate admin
  WEBLATE_ADMIN_EMAIL: noreply@example.com
```

`WEBLATE_ADMIN_PASSWORD`
Sets the password for the `admin` user.

If not set and `admin` user does not exist, it is created with a random password shown on first container startup.
If not set and `admin` user exists, no action is performed.
If set the `admin` user is adjusted on every container startup to match `WEBLATE_ADMIN_PASSWORD`, `WEBLATE_ADMIN_NAME` and `WEBLATE_ADMIN_EMAIL`. 
It might be a security risk to store password in the configuration file. Consider using this variable only for initial setup (or let Weblate generate random password on initial startup) or for password recovery.

**WEBLATE_ADMIN_PASSWORD**
Sets the path to a file containing the password for the admin user.

**WEBLATE_ADMIN_PASSWORD_FILE**

**WEBLATE_SERVER_EMAIL**
Configures the address for outgoing e-mails.

**WEBLATE_CONTACT_FORM**
Configures contact form behavior, see CONTACT_FORM.

**WEBLATE_ALLOWED_HOSTS**
Configures allowed HTTP hostnames using ALLOWED_HOSTS.
Defaults to * which allows all hostnames.

```bash
environment:
  WEBLATE_ALLOWED_HOSTS: weblate.example.com,example.com
```

**WEBLATE_REGISTRATION_OPEN**
Configures whether registrations are open by toggling REGISTRATION_OPEN.

```bash
environment:
  WEBLATE_REGISTRATION_OPEN: 0
```

**WEBLATE_REGISTRATION_ALLOW_BACKENDS**
Configure which authentication methods can be used to create new account via REGISTRATION_ALLOW_BACKENDS.

```bash
environment:
  WEBLATE_REGISTRATION_OPEN: 0
  WEBLATE_REGISTRATION_ALLOW_BACKENDS: azuread-oauth2,azuread-tenant-oauth2
```

**WEBLATE_TIME_ZONE**
Configures the used time zone in Weblate, see TIME_ZONE.

```bash
environment:
  WEBLATE_TIME_ZONE: Europe/Prague
```

**WEBLATE_ENABLE_HTTPS**
Makes Weblate assume it is operated behind a reverse HTTPS proxy, it makes Weblate use HTTPS in e-mail and API links or set secure flags on cookies.

```bash
environment:
  WEBLATE_TIME_ZONE: Europe/Prague
```

Please see ENABLE_HTTPS documentation for possible caveats.
This does not make the Weblate container accept HTTPS connections, you need to configure that as well, see Docker container with HTTPS support for examples.

```yaml
environment:
  WEBLATE_ENABLE_HTTPS: 1
```

**WEBLATE_ENABLE_HTTPS**

This environment variable and `ENABLE_HTTPS` need to be set to enable HTTPS connections. They are used for configuring HTTPS connection for the Weblate container.

**WEBLATE_SECURE_PROXY_SSL_HEADER**

A tuple representing a HTTP header/value combination that signifies a request is secure. This is needed when Weblate is running behind a reverse proxy doing SSL termination which does not pass standard HTTPS headers.

```yaml
environment:
  WEBLATE_SECURE_PROXY_SSL_HEADER: HTTP_X_FORWARDED_PROTO,https
```

**WEBLATE_IP_PROXY_HEADER**

Lets Weblate fetch the IP address from any given HTTP header. Use this when using a reverse proxy in front of the Weblate container.

Enables `IP_BEHIND_REVERSE_PROXY` and sets `IP_PROXY_HEADER`.

```yaml
environment:
  WEBLATE_IP_PROXY_HEADER: HTTP_X_FORWARDED_FOR
```

**WEBLATE_REQUIRE_LOGIN**

Enables `REQUIRE_LOGIN` to enforce authentication on whole Weblate.

```yaml
environment:
  WEBLATE_REQUIRE_LOGIN: 1
```

**WEBLATE_LOGIN_REQUIRED_URLS_EXCEPTIONS**

You can either replace whole settings, or modify default value using ADD and REMOVE variables.

**WEBLATE_GITHUB_USERNAME**

Configures GitHub username for GitHub pull-requests by changing `GITHUB_USERNAME`.

**WEBLATE_GITHUB_TOKEN**

Configures GitHub personal access token for GitHub pull-requests via API by changing `GITHUB_TOKEN`. 4.3+

The format must conform to Django’s expectations. Django transforms raw HTTP header names as follows:

- converts all characters to uppercase
- replaces any hyphens with underscores
- prepends `HTTP_` prefix

So `X-Forwarded-For` would be mapped to `HTTP_X_FORWARDED_FOR`.

```yaml
environment:
  WEBLATE_IP_PROXY_HEADER: HTTP_X_FORWARDED_FOR
```

**WEBLATE_SECURE_PROXY_SSL_HEADER**

A tuple representing a HTTP header/value combination that signifies a request is secure. This is needed when Weblate is running behind a reverse proxy doing SSL termination which does not pass standard HTTPS headers.

```yaml
environment:
  WEBLATE_SECURE_PROXY_SSL_HEADER: HTTP_X_FORWARDED_PROTO,https
```

**WEBLATE_REQUIRE_LOGIN**

Enables `REQUIRE_LOGIN` to enforce authentication on whole Weblate.

```yaml
environment:
  WEBLATE_REQUIRE_LOGIN: 1
```

**WEBLATE_LOGIN_REQUIRED_URLS_EXCEPTIONS**

You can either replace whole settings, or modify default value using ADD and REMOVE variables.

**WEBLATE_GITHUB_USERNAME**

Configures GitHub username for GitHub pull-requests by changing `GITHUB_USERNAME`.

**WEBLATE_GITHUB_TOKEN**

Configures GitHub personal access token for GitHub pull-requests via API by changing `GITHUB_TOKEN`. 4.3+

The format must conform to Django’s expectations. Django transforms raw HTTP header names as follows:

- converts all characters to uppercase
- replaces any hyphens with underscores
- prepends `HTTP_` prefix

So `X-Forwarded-For` would be mapped to `HTTP_X_FORWARDED_FOR`.

```yaml
environment:
  WEBLATE_IP_PROXY_HEADER: HTTP_X_FORWARDED_FOR
```
GitHub

**WEBLATE_GITLAB_USERNAME**
Configures GitLab username for GitLab merge-requests by changing `GITLAB_USERNAME`.

**WEBLATE_GITLAB_TOKEN**
Configures GitLab personal access token for GitLab merge-requests via API by changing `GITLAB_TOKEN`.

Pagure

**WEBLATE_PAGURE_USERNAME**
Configures the pagure merge-requests username.

**WEBLATE_PAGURE_TOKEN**
Configures the pagure merge-requests via API.

**WEBLATE_SIMPLIFY_LANGUAGES**
Configures the language simplification policy, see `SIMPLOY_LANGUAGES`.

**WEBLATE_DEFAULT_ACCESS_CONTROL**
Configures the default access for new projects, see `DEFAULT_ACCESS_CONTROL`.

**WEBLATE_DEFAULT_RESTRICTED_COMPONENT**
Configures the default value for restricted component, see `DEFAULT_RESTRICTED_COMPONENT`.

**WEBLATE_DEFAULT_TRANSLATION_PROPAGATION**
Configures the default value for translation propagation, see `DEFAULT_TRANSLATION_PROPAGATION`.

**WEBLATE_DEFAULT_COMMITER_EMAIL**
Configures the default committer email.

**WEBLATE_DEFAULT_COMMITER_NAME**
Configures the default committer name.

**WEBLATE_DEFAULT_SHARED_TM**
Configures the default shared translation.

**WEBLATE_AKISMET_API_KEY**
Configures the Akismet API key, see `AKISMET_API_KEY`.

**WEBLATE_GPG_IDENTITY**
Configures GPG signing of commits, see `WEBLATE_GPG_IDENTITY`.

Signing Git commits with GnuPG

**WEBLATE_URL_PREFIX**
Configures URL prefix where Weblate is running, see `URL_PREFIX`.

**WEBLATE_SILENCED_SYSTEM_CHECKS**
Configures checks which you do not want to be displayed, see `SILENCED_SYSTEM_CHECKS`.

**WEBLATE_CSP_SCRIPT_SRC**
**WEBLATE_CSP_IMG_SRC**
**WEBLATE_CSP_CONNECT_SRC**
**WEBLATE_CSP_STYLE_SRC**
**WEBLATE_CSP_FONT_SRC**
Allows to customize Content-Security-Policy HTTP header.
You can set configuration for any rate limiter scopes. To do that add WEBLATE_ prefix to any of setting described in 129.

WEBLATE_ENABLE_AVATARS

Machine translation settings

WEBLATE_MT_APERTIUM_API
Enables Apertium machine translation and sets MT_APERTIUM_API

WEBLATE_MT_AWS_REGION
WEBLATE_MT_AWS_ACCESS_KEY_ID
WEBLATE_MT_AWS_SECRET_ACCESS_KEY
Configures AWS machine translation.

WEBLATE_MT_DEEPL_KEY
Enables DeepL machine translation and sets MT_DEEPL_KEY

WEBLATE_MT_DEEPL_API_URL

WEBLATE_MT_LIBRETRANSLATE_KEY
Enables LibreTranslate machine translation and sets MT_LIBRETRANSLATE_KEY

WEBLATE_MT_LIBRETRANSLATE_API_URL
Configures LibreTranslate API instance to use, see MT_LIBRETRANSLATE_API_URL.

WEBLATE_MT_GOOGLE_KEY
Enables Google Translate and sets MT_GOOGLE_KEY

WEBLATE_MT_MICROSOFT_COGNITIVE_KEY
Enables Microsoft Cognitive Services Translator and sets MT_MICROSOFT_COGNITIVE_KEY
WEBLATE_MT_MICROSOFT_ENDPOINT_URL
Sets MT_MICROSOFT_ENDPOINT_URL, please note this is supposed to contain domain name only.

WEBLATE_MT_MICROSOFT_REGION
Sets MT_MICROSOFT_REGION

WEBLATE_MT_MICROSOFT_BASE_URL
Sets MT_MICROSOFT_BASE_URL

WEBLATE_MT_MODERNMT_KEY
Enables ModernMT and sets MT_MODERNMT_KEY.

WEBLATE_MT_MYMEMORY_ENABLED
Enables MyMemory machine translation and sets MT_MYMEMORY_EMAIL to WEBLATE_ADMIN_EMAIL.

```
environment:
  WEBLATE_MT_MYMEMORY_ENABLED: 1
```

WEBLATE_MT_GLOSBE_ENABLED
Glosbe

```
environment:
  WEBLATE_MT_GLOSBE_ENABLED: 1
```

WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED
Microsoft Terminology Service

```
environment:
  WEBLATE_MT_MICROSOFT_TERMINOLOGY_ENABLED: 1
```

WEBLATE_MT_SAP_BASE_URL
WEBLATE_MT_SAP_SANDBOX_APIKEY
WEBLATE_MT_SAP_USERNAME
WEBLATE_MT_SAP_PASSWORD
WEBLATE_MT_SAP_USE_MT
Configures SAP Translation Hub machine translation.

```
environment:
  WEBLATE_MT_SAP_BASE_URL: "https://example.hana.ondemand.com/translationhub/api/v1/"
  WEBLATE_MT_SAP_USERNAME: "user"
  WEBLATE_MT_SAP_PASSWORD: "password"
  WEBLATE_MT_SAP_USE_MT: 1
```

Authentication settings

LDAP

WEBLATE_AUTH_LDAP_SERVER_URI
WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE
WEBLATE_AUTH_LDAP_USER_ATTR_MAP
WEBLATE_AUTH_LDAP_BIND_DN
WEBLATE_AUTH_LDAP_BIND_PASSWORD
WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS
WEBLATE_AUTH_LDAP_USER_SEARCH
WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER
WEBLATE_AUTH_LDAP_USER_SEARCH_UNION
WEBLATE_AUTH_LDAP_USER_SEARCH_UNION_DELIMITER
LDAP authentication configuration.

Example for direct bind:
**Environment:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEBLATE_AUTH_LDAP_SERVER_URI</td>
<td>ldap://ldap.example.org</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_USER_DN_TEMPLATE</td>
<td>uid=%(user)s,ou=People,dc=example,dc=net</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_USER_ATTR_MAP</td>
<td>full_name:name,email:mail</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_BIND_DN</td>
<td>CN=ldap,CN=Users,DC=example,DC=com</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_BIND_PASSWORD</td>
<td>password</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_USER_SEARCH</td>
<td>CN=Users,DC=example,DC=com</td>
</tr>
</tbody>
</table>

**Example for search and bind:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEBLATE_AUTH_LDAP_SERVER_URI</td>
<td>ldap://ldap.example.org</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_BIND_DN</td>
<td>CN=ldap,CN=Users,DC=example,DC=com</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_BIND_PASSWORD</td>
<td>password</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_USER_SEARCH</td>
<td>CN=Users,DC=example,DC=com</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_USER_ATTR_MAP</td>
<td>full_name:name,email:mail</td>
</tr>
</tbody>
</table>

**Example for union search and bind:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEBLATE_AUTH_LDAP_SERVER_URI</td>
<td>ldap://ldap.example.org</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_BIND_DN</td>
<td>CN=ldap,CN=Users,DC=example,DC=com</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_BIND_PASSWORD</td>
<td>password</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_USER_SEARCH</td>
<td>CN=Users,DC=example,DC=com</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_USER_ATTR_MAP</td>
<td>full_name:name,email:mail</td>
</tr>
</tbody>
</table>

**Example with search and bind against Active Directory:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEBLATE_AUTH_LDAP_BIND_DN</td>
<td>CN=ldap,CN=Users,DC=example,DC=com</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_BIND_PASSWORD</td>
<td>password</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_SERVER_URI</td>
<td>ldap://ldap.example.org</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_CONNECTION_OPTION_REFERRALS</td>
<td>0</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_USER_SEARCH</td>
<td>CN=Users,DC=example,DC=com</td>
</tr>
<tr>
<td>WEBLATE_AUTH_LDAP_USER_SEARCH_FILTER</td>
<td>(sAMAccountName=%(user)s)</td>
</tr>
</tbody>
</table>

**GitHub**

- WEBLATE_SOCIAL_AUTH_GITHUB_KEY
- WEBLATE_SOCIAL_AUTH_GITHUB_SECRET

**Bitbucket**

- WEBLATE_SOCIAL_AUTH_BITBUCKET_KEY
- WEBLATE_SOCIAL_AUTH_BITBUCKET_SECRET
Facebook

WEBLATE_SOCIAL_AUTH_FACEBOOK_KEY
WEBLATE_SOCIAL_AUTH_FACEBOOK_SECRET
Facebook OAuth 2

Google

WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_KEY
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_DOMAINS
WEBLATE_SOCIAL_AUTH_GOOGLE_OAUTH2_WHITELISTED_EMAILS
Google OAuth 2

GitLab

WEBLATE_SOCIAL_AUTH_GITLAB_KEY
WEBLATE_SOCIAL_AUTH_GITLAB_SECRET
WEBLATE_SOCIAL_AUTH_GITLAB_API_URL
GitLab OAuth 2

Azure Active Directory

WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_KEY
WEBLATE_SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET
Enables Azure Active Directory authentication, see Microsoft Azure Active Directory.

Azure Active Directory with Tenant support

WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY
WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET
WEBLATE_SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_ID
Enables Azure Active Directory authentication with Tenant support, see Microsoft Azure Active Directory.

Keycloak

WEBLATE_SOCIAL_AUTH_KEYCLOAK_KEY
WEBLATE_SOCIAL_AUTH_KEYCLOAK_SECRET
WEBLATE_SOCIAL_AUTH_KEYCLOAK_PUBLIC_KEY
WEBLATE_SOCIAL_AUTH_KEYCLOAK_ALGORITHM
WEBLATE_SOCIAL_AUTH_KEYCLOAK_AUTHORIZATION_URL
WEBLATE_SOCIAL_AUTH_KEYCLOAK_ACCESS_TOKEN_URL
Enables Keycloak authentication, see documentation.
Linux vendors

You can enable authentication using Linux vendors authentication services by setting following variables to any value.

```text
WEBLATE_SOCIAL_AUTH_FEDORA
WEBLATE_SOCIAL_AUTH_OPENSUSE
WEBLATE_SOCIAL_AUTH_UBUNTU
```

Slack

```text
WEBLATE_SOCIAL_AUTH_SLACK_KEY
SOCIAL_AUTH_SLACK_SECRET
```

Enables Slack authentication, see Slack.

SAML

Self-signed SAML keys are automatically generated on first container startup. In case you want to use own keys, place the certificate and private key in `/app/data/ssl/saml.crt` and `/app/data/ssl/saml.key`.

```text
WEBLATE_SAML_IDP_ENTITY_ID
WEBLATE_SAML_IDP_URL
WEBLATE_SAML_IDP_X509CERT
```

SAML Identity Provider settings, see SAML.

Other authentication settings

```text
WEBLATE_NO_EMAIL_AUTH
```

Disables e-mail authentication when set to any value.

PostgreSQL database setup

The database is created by `docker-compose.yml`, so these settings affect both Weblate and PostgreSQL containers.

### POSTGRES_PASSWORD

PostgreSQL password.

### POSTGRES_PASSWORD_FILE

Path to the file containing the PostgreSQL password. Use as an alternative to POSTGRES_PASSWORD.

### POSTGRES_USER

PostgreSQL username.

### POSTGRES_DATABASE

PostgreSQL database name.

### POSTGRES_HOST

PostgreSQL server hostname or IP address. Defaults to database.

### POSTGRES_PORT

PostgreSQL server port. Defaults to none (uses the default value).

### POSTGRES_SSL_MODE

Configure how PostgreSQL handles SSL in connection to the server, for possible choices see SSL Mode Descriptions

### POSTGRES_ALTER_ROLE

Configures name of role to alter during migrations, see `PostgreSQL` 4.8.1 and Webrate.

### POSTGRES_CONN_MAX_AGE

The lifetime of a database connection, as an integer of seconds. Use 0 to close database connections at the end of each request (this is the default behavior).
Enabling connection persistence will typically cause more open connection to the database. Please adjust your database configuration prior enabling.

This is currently experimental, not widely tested and not supported by the Weblate team.

**CONN_MAX_AGE**, Persistent connections

### Database backup settings

**Dumped data for backups**

**WEBLATE_DATABASE_BACKUP**

Configures the daily database dump using **DATABASE_BACKUP**. Defaults to plain.

### Caching server setup

Using Redis is strongly recommended by Weblate and you have to provide a Redis instance when running Weblate in Docker.

**REDIS_HOST**
The Redis server hostname or IP address. Defaults to cache.

**REDIS_PORT**
The Redis server port. Defaults to 6379.

**REDIS_DB**
The Redis database number, defaults to 1.

**REDIS_PASSWORD**
The Redis server password, not used by default.

**REDIS_TLS**
Enables using SSL for Redis connection.

**REDIS_VERIFY_SSL**
Can be used to disable SSL certificate verification for Redis connection.

### Email server setup

To make outgoing e-mail work, you need to provide a mail server.

Example TLS configuration:

```
environment:
    WEBLATE_EMAIL_HOST: smtp.example.com
    WEBLATE_EMAIL_HOST_USER: user
    WEBLATE_EMAIL_HOST_PASSWORD: pass
```

Example SSL configuration:

```
environment:
    WEBLATE_EMAIL_HOST: smtp.example.com
    WEBLATE_EMAIL_PORT: 465
    WEBLATE_EMAIL_HOST_USER: user
    WEBLATE_EMAIL_HOST_PASSWORD: pass
    WEBLATE_EMAIL_USE_TLS: 0
    WEBLATE_EMAIL_USE_SSL: 1
```

**WEBLATE_EMAIL_HOST**
Mail server hostname or IP address.
WEBLATE_EMAIL_PORT
Mail server port, defaults to 25.

EMAIL_PORT

WEBLATE_EMAIL_HOST_USER

EMAIL_HOST_USER

WEBLATE_EMAIL_HOST_PASSWORD

EMAIL_HOST_PASSWORD

WEBLATE_EMAIL_HOST_PASSWORD_FILE

WEBLATE_EMAIL_HOST_PASSWORD

WEBLATE_EMAIL_USE_SSL
Whether to use an implicit TLS (secure) connection when talking to the SMTP server. In most e-mail documentation, this type of TLS connection is referred to as SSL. It is generally used on port 465. If you are experiencing problems, see the explicit TLS setting WEBLATE_EMAIL_USE_TLS.

WEBLATE_EMAIL_USE_TLS
Whether to use a TLS (secure) connection when talking to the SMTP server. This is used for explicit TLS connections, generally on port 587 or 25. If you are experiencing connections that hang, see the implicit TLS setting WEBLATE_EMAIL_USE_SSL.

WEBLATE_EMAIL_BACKEND
Configures Django back-end to use for sending e-mails.

WEBLATE_GET_HELP_URL

WEBLATE_STATUS_URL

WEBLATE_LEGAL_URL

WEBLATE_PRIVACY_URL
Configures PRIVACY_URL.
Error reporting

It is recommended to collect errors from the installation systematically, see [error reporting](#). To enable support for Rollbar, set the following:

**ROLLBAR_KEY**
Your Rollbar post server access token.

**ROLLBAR_ENVIRONMENT**
Your Rollbar environment, defaults to production.

To enable support for Sentry, set following:

**SENTRY_DSN**
Your Sentry DSN.

**SENTRY_ENVIRONMENT**
Your Sentry Environment (optional).

CDN

**WEBLATE_LOCALIZE_CDN_URL**
**WEBLATE_LOCALIZE_CDN_PATH**
Configuration for JavaScript CDN.

The **WEBLATE_LOCALIZE_CDN_PATH** is path within the container. It should be stored on the persistent volume and not in the transient storage.

One of possibilities is storing that inside the Weblate data dir:

```plaintext
environment:
WEBLATE_LOCALIZE_CDN_URL: https://cdn.example.com/
WEBLATE_LOCALIZE_CDN_PATH: /app/data/l10n-cdn
```

You are responsible for setting up serving of the files generated by Weblate, it only does stores the files in configured location.

weblate-cdn**LOCALIZE_CDN_URL**,**LOCALIZE_CDN_PATH**

Changing enabled apps, checks, addons or autofixes

**WEBLATE_ADD_APPS**
**WEBLATE_REMOVE_APPS**
**WEBLATE_ADD_CHECK**
**WEBLATE_REMOVE_CHECK**
**WEBLATE_ADD_AUTOFIX**
**WEBLATE_REMOVE_AUTOFIX**
**WEBLATE_ADD_ADDONS**
**WEBLATE_REMOVE_ADDONS**

You can change enabled apps, checks, addons or autofixes with these variables:

```plaintext
environment:
WEBLATE_ADD_AUTOFIX: weblate.trans.autofixes.whitespace.
WEBLATE_ADD_ADDONS: customize.addons.MyAddon,customize.addons.OtherAddon
```

**CHECK_LIST**, **AUTOFIX_LIST**, **WEBLATE_ADDONS**, **INSTALLED_APPS**
WEBLATE_WORKERS

Base number of worker processes running in the container. When not set it is determined automatically on container startup based on number of CPU cores available.

It is used to determine CELERY_MAIN_OPTIONS, CELERY_NOTIFY_OPTIONS, CELERY_MEMORY_OPTIONS, CELERY_TRANSLATE_OPTIONS, CELERY_BACKUP_OPTIONS, CELERY_BEAT_OPTIONS, and UWSGI_WORKERS. You can use these settings to fine-tune.

CELERY_MAIN_OPTIONS
CELERY_NOTIFY_OPTIONS
CELERY_MEMORY_OPTIONS
CELERY_TRANSLATE_OPTIONS
CELERY_BACKUP_OPTIONS
CELERY_BEAT_OPTIONS

These variables allow you to adjust Celery worker options. It can be useful to adjust concurrency (\`--concurrency 16\`) or use different pool implementation (\`--pool=gevent\`).

By default, the number of concurrent workers is based on WEBLATE_WORKERS.

| environment: |
| CELERY_MAIN_OPTIONS: --concurrency 16 |

CELERY worker options

UWSGI_WORKERS

Configure how many uWSGI workers should be executed.

It defaults to WEBLATE_WORKERS.

| environment: |
| UWSGI_WORKERS: 32 |

WEBLATE_SERVICE

Defines which services should be executed inside the container. Use this for Scaling horizontally.

| environment: |
| WEBLATE_SERVICE: |
| celery-beat |
| celery-backup |
| celery-celery |
| celery-memory |
| celery-notify |
| celery-translate |
| web |
| Web |

137
Docker container volumes

There are two volumes (data and cache) exported by the Weblate container. The other service containers (PostgreSQL or Redis) have their data volumes as well, but those are not covered by this document.

The data volume is used to store Weblate persistent data such as cloned repositories or to customize Weblate installation.

The placement of the Docker volume on host system depends on your Docker configuration, but usually it is stored in /var/lib/docker/volumes/weblate-docker_weblate-data/_data/ (the path consist of name of your docker-compose directory, container, and volume names). In the container it is mounted as /app/data.

The cache volume is mounted as /app/cache and is used to store static files. Its content is recreated on container startup and the volume can be mounted using ephemeral filesystem such as tmpfs.

When creating the volumes manually, the directories should be owned by UID 1000 as that is user used inside the container.

Docker volumes documentation

Further configuration customization

You can further customize Weblate installation in the data volume, see Docker container volumes.

Custom configuration files

You can additionally override the configuration in /app/data/settings-override.py (see Docker container volumes). This is executed at the end of built-in settings, after all environment settings are loaded, and you can adjust or override them.

Replacing logo and other static files

The static files coming with Weblate can be overridden by placing into /app/data/python/customize/static (see Docker container volumes). For example creating /app/data/python/customize/static/favicon.ico will replace the favicon.

The files are copied to the corresponding location upon container startup, so a restart of Weblate is needed after changing the content of the volume.

Alternatively you can also include own module (see Customizing Weblate) and add it as separate volume to the Docker container, for example:

```yaml
weblate:
  volumes:
    - weblate-data:/app/data
    - ./weblate_customization/weblate_customization:/app/data/python/
  environment:
    WEBLATE_ADD_APPS: weblate_customization
```

Adding own Python modules

You can place own Python modules in /app/data/python/ (see Docker container volumes) and they can be then loaded by Weblate, most likely by using Custom configuration files.
Installing on Debian and Ubuntu

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- **2GB RAM**
- **2 CPU cores**
- **1 GB of storage space**

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

System requirements

Install the dependencies needed to build the Python modules (see [documentation](#)):

```bash
apt install \n   libxml2-dev libxslt-dev libfreetype6-dev libjpeg-dev libz-dev libyaml-dev \n   libcairo-dev gir1.2-pango-1.0 libgirepository1.0-dev libacl1-dev libssl-dev \n   build-essential python3-gdbm python3-dev python3-pip python3-virtualenv \n   virtualenv git
```

Install wanted optional dependencies depending on features you intend to use (see [documentation](#)):

```bash
apt install tesseract-ocr libtesseract-dev libleptonica-dev
```

Optionally install software for running production server, see [documentation](#), [Weblate](#), [Celery](#). Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```bash
# Web server option 1: NGINX and uWSGI
apt install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with `mod_wsgi`
apt install apache2 libapache2-mod-wsgi-py3

# Caching backend: Redis
apt install redis-server

# Database server: PostgreSQL
apt install postgresql postgresql-contrib

# SMTP server
apt install exim4
```
**Python modules**

**Notes:** We're using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:
   ```bash
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:
   ```bash
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:
   ```bash
   pip install Weblate
   ```

4. Install database driver:
   ```bash
   pip install psycopg2-binary
   ```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check [documentation](#)):
   ```bash
   pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
   ```

**Configuring Weblate**

**Notes:** Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to `weblate` command as `~/weblate-env/bin/weblate`.


2. Adjust the values in the new `settings.py` file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see [documentation](#).

3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check [Weblate](#) for production ready setup):
   ```bash
   weblate migrate
   ```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:
   ```bash
   weblate createadmin
   ```

5. Collect static files for web server (see [documentation](#) and [example](#)):
   ```bash
   weblate collectstatic
   ```

6. Compress JavaScript and CSS files (optional, see [documentation](#)):
   ```bash
   weblate compress
   ```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See [Celery](#) for more info:
   ```bash
   ~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
   ```

8. Start the development server (see [documentation](#) for production setup):
   ```bash
   weblate runserver
   ```
After installation

Congratulations, your Weblate server is now running and you can start using it.
You can now access Weblate on http://localhost:8000/.
Login with admin credentials obtained during installation or register with new users.
You can now run Weblate commands using weblate command when Weblate virtualenv is active, see /manage/entrypoints.
You can stop the test server with Ctrl+C.
Review potential issues with your installation either on /manage/performance/ URL or using weblate check --deploy, see /manage/entrypoints.

Adding translation

1. Open the admin interface (http://localhost:8000/create/project/) and create the project you want to translate. See Project configuration for more details.
   All you need to specify here is the project name and its website.
2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See Component configuration for more details.
   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see /manage/entrypoints for more details.
3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on SUSE and openSUSE

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):
2GB RAM
2 CPU cores
1 GB of storage space
The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).
Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.
The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Note: Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

System requirements

Install the dependencies needed to build the Python modules (see /manage/entrypoints):

```bash
zypper install \n   libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel\n   cairo-devel typelib-1_0-Pango-1_0 gobject-introspection-devel libacl-\n   devel \n   python3-pip python3-virtualenv python3-devel git
```

Install wanted optional dependencies depending on features you intend to use (see /manage/entrypoints):
zypper install tesseract-ocr tesseract-devel leptonica-devel

Optionally install software for running production server, see Weblate, Celery, Weblate, Weblate. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
zypper install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with `mod_wsgi`
zypper install apache2 apache2-mod_wsgi

# Caching backend: Redis
zypper install redis-server

# Database server: PostgreSQL
zypper install postgresql postgresql-contrib

# SMTP server
zypper install postfix
```

**Python modules**

**Note:** We’re using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:

   ```
   pip install Weblate
   ```

4. Install database driver:

   ```
   pip install psycopg2-binary
   ```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check Weblate for production ready setup):

   ```
   pip install ruamel.yaml aeidon boto3 zeep chardet tesserocr
   ```

**Configuring Weblate**

**Note:** Following steps assume virtualenv used by Weblate is active (what can be done by . ~/weblate-env/bin/activate). In case this is not true, you will have to specify full path to weblate command as ~/weblate-env/bin/weblate.


2. Adjust the values in the new settings.py file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see Weblate.

3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check Weblate for production ready setup):
4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate migrate
```

5. Collect static files for web server (see `weblate` and `collectstatic`):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see `weblate` and `compress`):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See `Celery` for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see `weblate` for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.

You can now access Weblate on `http://localhost:8000/`.

Login with admin credentials obtained during installation or register with new users.

You can now run Weblate commands using `weblate` command when Weblate virtualenv is active, see `weblate`.

You can stop the test server with Ctrl+C.

Review potential issues with your installation either on `/manage/performance/` URL or using `weblate check --deploy`, see `weblate`.

Adding translation

1. Open the admin interface (`http://localhost:8000/create/project/`) and create the project you want to translate. See `Project configuration` for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See `Component configuration` for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see `weblate` for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on RedHat, Fedora and CentOS

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

- 2GB RAM
- 2 CPU cores
- 1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.
The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

System requirements

Install the dependencies needed to build the Python modules (see [User Guide](#)):  

```
dnf install \  
  libxslt-devel libxml2-devel freetype-devel libjpeg-devel zlib-devel_  
  libyaml-devel \  
  cairo-devel pango-devel gobject-introspection-devel libacl-devel \  
  python3-pip python3-virtualenv python3-devel git
```

Install wanted optional dependencies depending on features you intend to use (see [User Guide](#)):

```
dnf install tesseract-langpack-eng tesseract-devel leptonica-devel
```

Optionally install software for running production server, see [User Guide](#). Weblate, [Celery](#). Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
dnf install nginx uwsgi uwsgi-plugin-python3

# Web server option 2: Apache with `mod_wsgi`
dnf install apache2 apache2-mod_wsgi

# Caching backend: Redis
dnf install redis

# Database server: PostgreSQL
dnf install postgresql postgresql-contrib

# SMTP server
dnf install postfix
```

Python modules

We’re using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:

   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:

   ```
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:

   ```
   pip install Weblate
   ```

4. Install database driver:

   ```
   pip install psycopg2-binary
   ```
5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check `pip list`):

```
pip install ruamel.yaml aeidon boto3 zeep char det tess erocr
```

**Configuring Weblate**

### Notes

Following steps assume virtualenv used by Weblate is active (what can be done by `. ~/weblate-env/bin/activate`). In case this is not true, you will have to specify full path to `weblate` command as `~/weblate-env/bin/weblate`.


2. Adjust the values in the new settings.py file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see `weblate check --deploy`.

3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check Weblate for production ready setup):

```
weblate migrate
```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:

```
weblate createadmin
```

5. Collect static files for web server (see `weblate runserver` and `weblate check --deploy`):

```
weblate collectstatic
```

6. Compress JavaScript and CSS files (optional, see `weblate runserver`):

```
weblate compress
```

7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See `weblate runserver` for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see `weblate runserver` for production setup):

```
weblate runserver
```

**After installation**

Congratulations, your Weblate server is now running and you can start using it.

You can now access Weblate on `http://localhost:8000/`.

Login with admin credentials obtained during installation or register with new users.

You can now run Weblate commands using `weblate` command when Weblate virtualenv is active, see `weblate runserver`.

You can stop the test server with Ctrl+C.

Review potential issues with your installation either on `/manage/performance/` URL or using `weblate check --deploy`, see `weblate check --deploy`.

145
Adding translation

1. Open the admin interface (http://localhost:8000/create/project/) and create the project you want to translate. See Project configuration for more details.

   All you need to specify here is the project name and its website.

2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See Component configuration for more details.

   The important fields here are: Component name, VCS repository address and mask for finding translatable files. Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see Component configuration for more details.

3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on macOS

Hardware requirements

Weblate should run on any contemporary hardware without problems, the following is the minimal configuration required to run Weblate on a single host (Weblate, database and webserver):

2GB RAM
2 CPU cores
1 GB of storage space

The more memory the better - it is used for caching on all levels (filesystem, database and Weblate).

Many concurrent users increases the amount of needed CPU cores. For hundreds of translation components at least 4 GB of RAM is recommended.

The typical database storage usage is around 300 MB per 1 million hosted words. Storage space needed for cloned repositories varies, but Weblate tries to keep their size minimal by doing shallow clones.

Actual requirements for your installation of Weblate vary heavily based on the size of the translations managed in it.

System requirements

Install the dependencies needed to build the Python modules (see System requirements):

```
brew install python pango cairo gobject-introspection libffi glib libyaml
pip3 install virtualenv
```

Make sure pip will be able to find the libffi version provided by homebrew — this will be needed during the installation build step.

```
export PKG_CONFIG_PATH="/usr/local/opt/libffi/lib/pkgconfig"
```

Install wanted optional dependencies depending on features you intend to use (see System requirements):

```
brew install tesseract
```

Optionally install software for running production server, see System requirements, Weblate System requirements, Celery System requirements. Depending on size of your installation you might want to run these components on dedicated servers.

The local installation instructions:

```
# Web server option 1: NGINX and uWSGI
brew install nginx uwsgi
# Web server option 2: Apache with `mod_wsgi`
```
brew install httpd
# Caching backend: Redis
brew install redis
# Database server: PostgreSQL
brew install postgresql

Python modules

**Note:** We’re using virtualenv to install Weblate in a separate environment from your system. If you are not familiar with it, check virtualenv User Guide.

1. Create the virtualenv for Weblate:
   
   ```
   virtualenv --python=python3 ~/weblate-env
   ```

2. Activate the virtualenv for Weblate:
   
   ```
   . ~/weblate-env/bin/activate
   ```

3. Install Weblate including all dependencies:
   
   ```
   pip install Weblate
   ```

4. Install database driver:
   
   ```
   pip install psycopg2-binary
   ```

5. Install wanted optional dependencies depending on features you intend to use (some might require additional system libraries, check [here](#)):
   
   ```
   pip install ruamel.yaml aidon boto3 zeep chardet tesserocr
   ```

Configuring Weblate

**Note:** Following steps assume virtualenv used by Weblate is active (what can be done by . ~/weblate-env/bin/activate). In case this is not true, you will have to specify full path to weblate command as ~/weblate-env/bin/weblate.


2. Adjust the values in the new settings.py file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see [here](#).

3. Create the database and its structure for Weblate (the example settings use PostgreSQL, check Weblate [here](#) for production ready setup):
   
   ```
   weblate migrate
   ```

4. Create the administrator user account and copy the password it outputs to the clipboard, and also save it for later use:
   
   ```
   weblate createadmin
   ```

5. Collect static files for web server (see [here](#) and [here](#)):
   
   ```
   weblate collectstatic
   ```

6. Compress JavaScript and CSS files (optional, see [here](#) and [here](#)):
   
   ```
   weblate compress
   ```
7. Start Celery workers. This is not necessary for development purposes, but strongly recommended otherwise. See Celery for more info:

```
~/weblate-env/lib/python3.7/site-packages/weblate/examples/celery start
```

8. Start the development server (see for production setup):

```
weblate runserver
```

After installation

Congratulations, your Weblate server is now running and you can start using it.
You can now access Weblate on http://localhost:8000/.
Login with admin credentials obtained during installation or register with new users.
You can now run Weblate commands using weblate command when Weblate virtualenv is active, see .
You can stop the test server with Ctrl+C.
Review potential issues with your installation either on /manage/performance/ URL or using weblate check --deploy, see .

Adding translation

1. Open the admin interface (http://localhost:8000/create/project/) and create the project you want to translate. See Project configuration for more details.
   All you need to specify here is the project name and its website.
2. Create a component which is the real object for translation - it points to the VCS repository, and selects which files to translate. See Component configuration for more details.
   The important fields here are: Component name, VCS repository address and mask for finding translatable files.
   Weblate supports a wide range of formats including gettext PO files, Android resource strings, iOS string properties, Java properties or Qt Linguist files, see for more details.
3. Once the above is completed (it can be lengthy process depending on the size of your VCS repository, and number of messages to translate), you can start translating.

Installing on Debian and Ubuntu
Installing on SUSE and openSUSE
Installing on RedHat, Fedora and CentOS

2. Git Webblate tarball:

```
git clone https://github.com/WeblateOrg/weblate.git weblate-src
```

3. Weblate virtualenv:

```
. ~/weblate-env/bin/activate
pip install -e weblate-src
```

4. Weblate/settings_example.py Webblate/settings.py
5. Adjust the values in the new settings.py file to your liking. You will need to provide at least the database credentials and Django secret key, but you will want more changes for production setup, see .
6. Weblate
7. Django
weblate migrate
weblate collectstatic
weblate compress
weblate compilemessages

Installing on OpenShift

With the OpenShift Weblate template you can get your personal Weblate instance up and running in seconds. All of Weblate’s dependencies are already included. PostgreSQL is set up as the default database and persistent volume claims are used.

You can find the template at <https://github.com/WeblateOrg/openshift/>.

The following examples assume you have a working OpenShift v3.x environment, with oc client tool installed. Please check the OpenShift documentation for instructions.

The template.yml is suited for running all components in OpenShift. There is also template-external-postgresql.yml which does not start a PostgreSQL server and allows you to configure external PostgreSQL server.

Web Console

Copy the raw content from template.yml and import them into your project, then use the Create button in the OpenShift web console to create your application. The web console will prompt you for the values for all of the parameters used by the template.

CLI

To upload the Weblate template to your current project's template library, pass the template.yml file with the following command:

```bash
$ oc create -f https://raw.githubusercontent.com/WeblateOrg/openshift/main/template.yml \
    -n <PROJECT>
```

The template is now available for selection using the web console or the CLI.

The parameters that you can override are listed in the parameters section of the template. You can list them with the CLI by using the following command and specifying the file to be used:

```bash
# If the template is already uploaded
$ oc process --parameters -n <PROJECT> weblate
```
You can also use the CLI to process templates and use the configuration that is generated to create objects immediately.

```
  -p APPLICATION_NAME=weblate
  -p WEBLATE_VERSION=4.3.1-1
  -p WEBLATE_SITE_DOMAIN=weblate.app-openshift.example.com
  -p POSTGRESQL_IMAGE=docker-registry.default.svc:5000/openshift/postgresql
  -p REDIS_IMAGE=docker-registry.default.svc:5000/openshift/redis:3.2
  | oc create -f
```

The Weblate instance should be available after successful migration and deployment at the specified WEBLATE_SITE_DOMAIN parameter.

After container setup, you can sign in as admin user with password provided in WEBLATE_ADMIN_PASSWORD, or a random password generated on first start if that was not set.

To reset admin password, restart the container with WEBLATE_ADMIN_PASSWORD set to new password in the respective Secret.

```
$ oc delete all -l app=<APPLICATION_NAME>
$ oc delete configmap -l app=<APPLICATION_NAME>
$ oc delete secret -l app=<APPLICATION_NAME>
  # ATTENTION! The following command is only optional and will permanently delete all of your data.
  $ oc delete pvc -l app=<APPLICATION_NAME>
$ oc delete all -l app=weblate
  || oc delete secret -l app=weblate
  || oc delete configmap -l app=weblate
  || oc delete pvc -l app=weblate
```

By processing the template a respective ConfigMap will be created and which can be used to customize the Weblate image. The ConfigMap is directly mounted as environment variables and triggers a new deployment every time it is changed. For further configuration options, see Docker environment variables for full list of environment variables.

**Installing on Kubernetes**

This guide is looking for contributors experienced with Kubernetes to cover the setup in more details.

With the Kubernetes Helm chart you can get your personal Weblate instance up and running in seconds. All of Weblate’s dependencies are already included. PostgreSQL is set up as the default database and persistent volume claims are used.

You can find the chart at <https://github.com/WeblateOrg/helm/> and it can be displayed at <https://artifacthub.io/packages/helm/weblate/weblate>.
helm repo add weblate https://helm.weblate.org
helm install my-release weblate/weblate

For further configuration options, see Docker environment variables for full list of environment variables.

Docker

Virtualenv

Installing on Debian and Ubuntu
Installing on SUSE and openSUSE
Installing on RedHat, Fedora and CentOS
Installing on macOS
Installing on OpenShift
Installing on Kubernetes

Weblate

PostgreSQL

Redis

SMTP

Celery

Python

Weblate

requirements.txt

Python

151
Weblate

- PostgreSQL
- MySQL
- MariaDB

Pango

- Cairo

Python

- pip
- Wheels

**Pango and Cairo**

Weblate: promotion

Michal Čihař: PGP: 63CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D

https://keybase.io/nijel

$ gpg --verify Weblate-3.5.tar.xz.asc

gpg: assuming signed data in 'Weblate-3.5.tar.xz'
gpg: Signature made Ne 3. března 2019, 16:43:15 CET
gpg:    using RSA key 87E673A8F83F6C3A0C344C8C3F4AA229D4D58C245

gpg: Can't check signature: public key not found
wkd:

$ gpg --auto-key-locate wkd --locate-keys michal@cihar.com
pub rsa4096 2009-06-17 [SC]
    63CB1DF1EF12CF2AC0EE5A329C27B31342B7511D
uid [ultimate] Michal Čihař <michal@cihar.com>
uid [ultimate] Michal Čihař <nijel@debian.org>
uid [ultimate] [jpeg image of size 8848]
sub rsa4096 2009-06-17 [E]
sub rsa4096 2015-09-09 [S]

Michal Čihař:

$ gpg --import wmxth3chu9jfxdxwyj1skpmhsj311mzm

$ gpg --keyserver hkp://pgp.mit.edu --recv-keys
    87E673AF83F6C3A0C344C8C3F4AA229D4D58C245
gpg: key 9C27B31342B7511D: "Michal Čihař <michal@cihar.com>" imported
gpg: Total number processed: 1
    unchange: 1

$ gpg --verify Weblate-3.5.tar.xz.asc
Primary key fingerprint: 63CB 1DF1 EF12 CF2A C0EE 5A32 9C27 B313 42B7 511D

$ gpg --verify Weblate-3.5.tar.xz.asc
   Good signature from "Michal Čihař <michal@cihar.com>" [ultimate]
   aka "Michal Čihař <nijel@debian.org>" [ultimate]
   aka "[jpeg image of size 8848]" [ultimate]
   aka "Michal Čihař (Braiins) <michal.cihar@braiins.cz>

$ gpg --verify Weblate-3.5.tar.xz.asc
   Good signature from "Michal Čihař <michal@cihar.com>" [ultimate]
   aka "Michal Čihař <nijel@debian.org>" [ultimate]
   aka "[jpeg image of size 8848]" [ultimate]
   aka "Michal Čihař (Braiins) <michal.cihar@braiins.cz>

$ gpg --verify Weblate-3.5.tar.xz.asc
   BAD signature from "Michal Čihař <michal@cihar.com>" [ultimate]
POSTGRESQL

**Migrating from other databases to PostgreSQL**

**POSTGRESQL**

- **Django**

**POSTGRESQL**

-- **postgresql-contrib**

**POSTGRESQL notes**

**POSTGRESQL**

---

```sql
# If PostgreSQL was not installed before, set the main password
sudo -u postgres psql postgres -c "\password postgres"
```

```sql
# Create a database user called "weblate"
sudo -u postgres createuser --superuser --pwprompt weblate
```

```sql
# Create the database "weblate" owned by "weblate"
sudo -u postgres createdb -E UTF8 -O weblate weblate
```

```sql
CREATE EXTENSION IF NOT EXISTS pg_trgm WITH SCHEMA weblate;
```
settings.py PostgreSQL

```python
DATABASES = {
    "default": {
        # Database engine
        "ENGINE": "django.db.backends.postgresql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Name of role to alter to set parameters in PostgreSQL,
        # use in case role name is different than user used for...
        # authentication.
        # "ALTER_ROLE": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "database.example.com",
        # Set to empty string for default
        "PORT": "",
    }
}
```

Weblate ALTER ROLE username

errors.UndefinedObject: role "weblate@hostname" does not exist

Azure Database for PostgreSQL

MySQL

MariaDB

Weblate MySQL

MySQL notes

MariaDB notes

PostgreSQL

MySQL 8.x, MariaDB 10.5.x or newer have reasonable default configuration so that no server tweaking should be necessary and all what is needed can be configured on the client side.

MySQL

MariaDB

[mysqld]
character-set-server = utf8mb4
character-set-client = utf8mb4
collation-server = utf8mb4_unicode_ci
datadir=/var/lib/mysql
log-error=/var/log/mariadb/mariadb.log
innodb_large_prefix=1

8 GB RAM /etc/my.cnf.d/server.cnf

MySQL

MariaDB

innodb_file_per_table

innodb_large_prefix

READ COMMITTED

SQL

STRICT_TRANS_TABLES

utf8mb4 Unicode

innodb_large_prefix

innodb_file_per_table
innodb_file_format=Barracuda
innodb_file_per_table=1
innodb_buffer_pool_size=2G
sql_mode=STRICT_TRANS_TABLES

#1071 - Specified key was too long; max key length is 767 bytes
innodb

#2006 - MySQL server has gone away error, configuring CONN_MAX_AGE might help.

MySQL/MariaDB settings.py

```
DATABASES = {
    "default": {
        # Database engine
        "ENGINE": "django.db.backends.mysql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "127.0.0.1",
        # Set to empty string for default
        "PORT": "3306",
        # In case you wish to use additional
        # connection options
        "OPTIONS": {},
    }
}
```

GRANT ALL ON weblate.* to 'weblate'@'localhost' IDENTIFIED BY 'password'; FLUSH PRIVILEGES;

SMTP AUTH extension not supported by server
SMTP AUTH extension not supported by server

Not receiving e-mails from Weblate Configuring outgoing e-mail in Docker container
HTTP

Weblate [VCS][settings.py]

```
import os

os.environ['http_proxy'] = 'http://proxy.example.com:8080'
os.environ['HTTPS_PROXY'] = 'http://proxy.example.com:8080'
```

### Proxy Environment Variables

**ADMINS**

Django

```python
ADMINS = [('admin', 'admin@example.com')]
```

**ALLOWED_HOSTS**

- `demo.weblate.org`
- `*`

**SESSION_ENGINE**

- `django.contrib.sessions.backends.cache`
- `weblate_cleansessions`

- Redis

```python
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
```
DEBUG
ADMINS

Django Weblate

DEBUG
DEFAULT_FROM_EMAIL

SECRET_KEY

Django Cookie

SECRET_KEY

SERVER_EMAIL

weblate migrate
django_secretkey

weblate check --deploy

weblate migrate --noinput

createadmin admin

Performance report
Django DEBUG settings:

```
DEBUG = False
```

ADMINS settings:

```
ADMINS = (("Your Name", "your_email@example.com"),)
```

HTTPS settings:

```
ENABLE_HTTPS = True
```

SECURE_HSTS_SECONDS setting:

```
SECURE_HSTS_SECONDS = 0
```

.readFileSync('path/to/settings.py')
Use adjacent location for running the database server, otherwise the networking performance or reliability might ruin your Weblate experience.

Check the database server performance or tweak its configuration, for example using PGTune.

---

### CACHES

**Django Redis**

```python
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/0",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "PARSER_CLASS": "redis.connection.HiredisParser",
        },
    },
}
```

**Django’s cache framework**

```python
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "unix:///var/run/redis/redis.sock?db=0",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "PARSER_CLASS": "redis.connection.HiredisParser",
        },
    },
}```

**Avatar cache**

```python
"avatar": {
    "BACKEND": "django.core.cache.backends.filebased.FileBasedCache",
    "LOCATION": os.path.join(DATA_DIR, "avatar-cache"),
    "TIMEOUT": 604800,
    "OPTIONS": {
        "MAX_ENTRIES": 1000,
    },
},
```

**Django’s cache framework**

```python
ENABLE_AVATARS = "AVATAR_URL_PREFIX"
```

---
Weblate

SERVER_EMAIL

SERVER_EMAIL = "admin@example.org"
DEFAULT_FROM_EMAIL = "weblate@example.org"

EMAIL_BACKEND = django.core.mail.backends.dummy.EmailBackend

ALLOWED_HOSTS = ['1.1.1.1']. You may need to add '1.1.1.1' to ALLOWED_HOSTS.

SECRET_KEY

os.environ["HOME"] = os.path.join(BASE_DIR, "configuration")
Django loaders

```python
TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [os.path.join(BASE_DIR, "templates")],
        "OPTIONS": {
            "context_processors": [
                "django.contrib.auth.context_processors.auth",
                "django.template.context_processors.debug",
                "django.template.context_processors.i18n",
                "django.template.context_processors.request",
                "django.contrib.messages.context_processors.messages",
                "weblate.trans.context_processors.weblate_context",
            ],
            "loaders": [
                "django.template.loaders.cached.Loader",
                "django.template.loaders.filesystem.Loader",
                "django.template.loaders.app_directories.Loader",
            ],
        },
    },
]
```

django.template.loaders.cached.Loader

**Celery**

Lazy commits automatic commit_pending

AUTO_UPDATE

JSON dump_memory

cleanuptrans

3.2 3.2 3.2 3.2 3.2 Celery Weblate

UTF-8 Linux UTF-8

```bash
LANG="C.UTF-8"
```

Apache on Ubuntu uses /etc/apache2/envvars:

```bash
export LANG='en_US.UTF-8'
export LC_ALL='en_US.UTF-8'
```
Apache on CentOS uses `/etc/sysconfig/httpd` (or `/opt/rh/httpd24/root/etc/sysconfig/httpd`):

```plaintext
LANG='en_US.UTF-8'
```

Weblate uses HTTP, SSL,

```plaintext
Debian CA files
usr/local/share/ca-certificates/
update-ca-certificates
```

Python settings.py (Debian):

```python
import os
os.environ["REQUESTS_CA_BUNDLE"] = "/etc/ssl/certs/ca-certificates.crt"
```

Weblate uses JavaScript and CSS:

Weblate:

```python
django.conf.settings.COMPRESS_OFFLINE = True
```

```python
weblate compress
```

Docker:

Common Deployment Scenarios:

In case you are not experienced with services described below, you might want to try Docker.

Weblate:

```
SSL
Web
NGINX
uWSGI
```

Celery

```
DATA_DIR
```

WSGI

```
```

163
**Web**

Web: Weblate Django uWSGI nginx Apache Gunicorn

Weblate: Django uWSGI nginx

```
weblate runserver
```

```
runserver @ Django
```

```
Django DEBUG
```

```
NGINX uWSGI nginx Apache Gunicorn
```

```
2.4 2.4 2.4 Weblate Django
```

```
weblate collectstatic --noinput
```

```
STATIC_ROOT
DATA_DIR
```

```
/static/favicon.ico
```

```
NGINX uWSGI nginx Apache Gunicorn
```

```
virtual env
```

```
weblate.middleware.SecurityMiddleware
```

```
Content-Security-Policy
X-XSS-Protection
HTTP
```

```
CSP_SCRIPT_SRC CSP_IMG_SRC CSP_CONNECT_SRC CSP_STYLE_SRC CSP_FONT_SRC
```

**NGINX**

```
NGINX uWSGI nginx Apache Gunicorn
```

```
/server {
    listen 80;
}
```

# This example assumes Weblate is installed in virtualenv in /home/weblate/
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your setup.

```
server_name weblate;
# Not used
root /var/www/html;

location ~ ^/favicon.ico$ {
    # DATA_DIR/static/favicon.ico
    alias /home/weblate/data/static/favicon.ico;
    expires 30d;
}

location /static/ {
    # DATA_DIR/static/
    alias /home/weblate/data/static/;
    expires 30d;
}

location /media/ {
    # DATA_DIR/media/
    alias /home/weblate/data/media/;
    expires 30d;
}

location / {
    include uwsgi_params;
    # Needed for long running operations in admin interface
    uwsgi_read_timeout 3600;
    # Adjust based to uwsgi configuration:
    uwsgi_pass unix:///run/uwsgi/app/weblate/socket;
    # uwsgi_pass 127.0.0.1:8080;
}

uWSGI [weblate/examples/weblate.uwsgi.ini]

# This example assumes Weblate is installed in virtualenv in /home/weblate/
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match...
# your setup.
[uwsgi]
plugins = python3
master = true
protocol = uwsgi
socket = 127.0.0.1:8080
wsgi-file = /home/weblate/weblate-env/lib/python3.9/site-packages/weblate/wsgi.py

# Add path to Weblate checkout if you did not install
# Weblate by pip
# python-path = /path/to/weblate

# In case you're using virtualenv uncomment this:
virtualenv = /home/weblate/weblate-env

# Needed for OAuth/OpenID
buffer-size = 8192

# Reload when consuming too much of memory
reload-on-rss = 250

# Increase number of workers for heavily loaded sites
workers = 8

# Enable threads for Sentry error submission
enable-threads = true

# Child processes do not need file descriptors
close-on-exec = true
# Avoid default 0000 umask
umask = 0022

# Run as weblate user
uid = weblate
gid = weblate

# Enable harakiri mode (kill requests after some time)
harakiri = 3600
harakiri-verbose = true

# Enable uWSGI stats server
stats = :1717
stats-http = true

# Do not log some errors caused by client disconnects
ignore-sigpipe = true
ignore-write-errors = true
disable-write-exception = true

## Django <-> uWSGI

## Apache

## Weblate <-> WSGI

<table>
<thead>
<tr>
<th>Django</th>
<th>uWSGI</th>
<th>Apache</th>
<th>Weblate</th>
<th>WSGI</th>
<th>mod_wsgi</th>
<th>weblate/examples/apache.conf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
# VirtualHost for Weblate
#
# This example assumes Weblate is installed in virtualenv in /home/weblate/
# and DATA_DIR is set to /home/weblate/data, please adjust paths to match your setup.
#
<VirtualHost *:80>
    ServerAdmin admin@weblate.example.org
    ServerName weblate.example.org

    # DATA_DIR/static/favicon.ico
    Alias /favicon.ico /home/weblate/data/static/favicon.ico

    # DATA_DIR/static/
    Alias /static/ /home/weblate/data/static/
    <Directory /home/weblate/data/static/>
        Require all granted
    </Directory>

    # DATA_DIR/media/
    Alias /media/ /home/weblate/data/media/
    <Directory /home/weblate/data/media/>
        Require all granted
    </Directory>

    # Path to your Weblate virtualenv
    WSGIDaemonProcess weblate python-home=/home/weblate/weblate-env_
    user=weblate
    WSGIProcessGroup weblate
    WSGIApplicationGroup %{GLOBAL}

    WSGIScriptAlias / /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/wsgi.py process-group=weblate request-timeout=600
```
WSGISPassAuthorization On

    <Directory /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/>
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>
</VirtualHost>

# Weblate Python3 modwsgi Python 3 variant libapache2-mod-wsgi-py3

Apache Gunicorn

Apache Gunicorn Apache 2.4 Weblate weblate/examples/apache.gunicorn.conf

# # VirtualHost for Weblate using gunicorn on localhost:8000
# # This example assumes Weblate is installed in virtualenv in /home/weblate/
# # and DATA_DIR is set to /home/weblate/data, please adjust paths to match...
# # your setup.
# <VirtualHost *:443>
  ServerAdmin admin@weblate.example.org
  ServerName weblate.example.org

  # # DATA_DIR/static/favicon.ico
  Alias /favicon.ico /home/weblate/data/static/favicon.ico

  # # DATA_DIR/static/
  Alias /static/ /home/weblate/data/static/
  <Directory /home/weblate/data/static/>
      Require all granted
  </Directory>

  # # DATA_DIR/media/
  Alias /media/ /home/weblate/data/media/
  <Directory /home/weblate/data/media/>
      Require all granted
  </Directory>

  SSLEngine on
  SSLCertificateFile /etc/apache2/ssl/https_cert.cert
  SSLCertificateKeyFile /etc/apache2/ssl/https_key.pem
  SSLProxyEngine On
  ProxyPass /favicon.ico !
  ProxyPass /static/ !
  ProxyPass /media/ !
  ProxyPass / http://localhost:8000/
  ProxyPassReverse / http://localhost:8000/
  ProxyPreserveHost On
</VirtualHost>

---

How to use Django with Gunicorn

---

167
Weblate 1.3 WSGI prefork MPM

Weblate /weblate Apache mod_wsgi weblate/examples/apache-path.conf

# # VirtualHost for Weblate, running under /weblate path
# # This example assumes Weblate is installed in virtualenv in /home/weblate/
# # and DATA_DIR is set to /home/weblate/data, please adjust paths to match...
# # your setup.
#<VirtualHost *:80>
ServerAdmin admin@weblate.example.org
ServerName weblate.example.org

# DATA_DIR/static/favicon.ico
Alias /weblate/favicon.ico /home/weblate/data/static/favicon.ico

# DATA_DIR/static/
Alias /weblate/static/ /home/weblate/data/static/
<Directory /home/weblate/data/static/>
  Require all granted
</Directory>

# DATA_DIR/media/
Alias /weblate/media/ /home/weblate/data/media/
<Directory /home/weblate/data/media/>
  Require all granted
</Directory>

# Path to your Weblate virtualenv
WSGIDaemonProcess weblate python-home=/home/weblate/weblate-env/
<user>weblate</user>
WSGIProcessGroup weblate
WSGIApplicationGroup %{GLOBAL}

WSGIScriptAlias /weblate /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/wsgi.py process-group=weblate request-timeout=600
WSGIPassAuthorization On

<Directory /home/weblate/weblate-env/lib/python3.7/site-packages/weblate/>
  <Files wsgi.py>
    Require all granted
  </Files>
</Directory>
</VirtualHost>

# URL_PREFIX = "/weblate"

168
Weblate

Celery

CELERY_TASK_ALWAYS_EAGER = False
CELERY_BROKER_URL = "redis://localhost:6379"
CELERY_RESULT_BACKEND = CELERY_BROKER_URL

---

Redis broker configuration in Celery

CELERY_TASK_ALWAYS_EAGER = True
CELERY_BROKER_URL = "memory://"
CELERY_TASK_EAGER_PROPAGATES = True

---

```
./weblate/examples/celery start
./weblate/examples/celery stop
```

---

Celery: Weblate

```
CELERY_APP = "weblate"
CELERY_BROKER_URL = "redis://localhost:6379"
CELERY_RESULT_BACKEND = CELERY_BROKER_URL
CELERY_TASK_ALWAYS_EAGER = True
CELERY_TASK_EAGER_PROPAGATES = True
```

---

```
[Unit]
Description=Celery Service (Weblate)
After=network.target

[Service]
Type=forking
User=weblate
Group=weblate
EnvironmentFile=/etc/default/celery-weblate
WorkingDirectory=/home/weblate
RuntimeDirectory=celery
RuntimeDirectoryPreserve=restart
LogsDirectory=celery
ExecStart=/bin/sh -c "${CELERY_BIN} multi start ${CELERYD_NODES} \n -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \n --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}" ExecStop=/bin/sh -c "${CELERY_BIN} multi stopwait ${CELERYD_NODES} \n --pidfile=${CELERYD_PID_FILE}" ExecReload=/bin/sh -c "${CELERY_BIN} multi restart ${CELERYD_NODES} \n -A ${CELERY_APP} --pidfile=${CELERYD_PID_FILE} \n --logfile=${CELERYD_LOG_FILE} --loglevel=${CELERYD_LOG_LEVEL} ${CELERYD_OPTS}" [Install] WantedBy=multi-user.target
```

---

```
/etc/systemd/system/celery-weblate.service
```

---

**Systemd**

```
[Service]
```

---

```
[Install]
```
CELERYD_NODES="celery notify memory backup translate"

CELDY_BIN="/home/weblate/weblate-env/bin/celery"

CELERY_APP="weblate.utils"


CELERYD_PID_FILE="/run/celery/weblate-%n.pid"

CELERYD_LOG_FILE="/var/log/celery/weblate-%n%I.log"

CELERYD_LOG_LEVEL="INFO"

CELERY_WORKER_RUNNING="1"

/etc/logrotate.d/celery

/var/log/celery/*/log {
    weekly
    missingok
    rotate 12
    compress
   notifempty
}

Celery beats

Webate Lazy commits

Celery queues
For monitoring metrics of Weblate you can use `GET /api/metrics/` API endpoint:

```
`Munin Weblate <https://github.com/WeblateOrg/munin>`_
```

**Sentry**

Weblate Sentry `<https://sentry.io>`_

```
settings.py:

```
SENTRY_DSN = "https://id@your.sentry.example.com/
```

**Rollbar**

Weblate Rollbar Rollbar notifier for Python

```
settings.py:

```
# Add rollbar as last middleware:
MIDDLEWARE = [
    # ... other middleware classes ...
    "rollbar.contrib.django.middleware.RollbarNotifierMiddleware",
]

# Configure client access
ROLLBAR = {
    "access_token": "POST_SERVER ITEM ADDRESS_TOKEN",
    "client_token": "POST_CLIENT ITEM ADDRESS_TOKEN",
    "environment": "development" if DEBUG else "production",
    "branch": "main",
    "root": "/absolute/path/to/code/root",
}
```

**mysql dump command:**

```
# Export current data
weblate dumpdata > /tmp/weblate.dump
# Import dump
weblate loaddata /tmp/weblate.dump
```
Weblate

Weblate: Docker
Installing on OpenShift
Installing on Kubernetes

Bitnami Weblate


Weblate Cloudron

Cloudron Web Weblate Cloudron Weblate package repo

YunoHost Weblate

YunoHost <https://github.com/YunoHost-Apps/weblate_ynh>

yunohost app install https://github.com/YunoHost-Apps/weblate_ynh
Weblate

Docker

Always check Version specific instructions before upgrade. In case you are skipping some versions, please follow instructions for all versions you are skipping in the upgrade. Sometimes it's better to upgrade to some intermediate version to ensure a smooth migration. Upgrading across multiple releases should work, but is not as well tested as single version upgrades.

It is recommended to perform a full database backup prior to upgrade so that you can roll back the database in case upgrade fails, see Weblate.

1. Stop wsgi and Celery processes. The upgrade can perform incompatible changes in the database, so it is always safer to avoid old processes running while upgrading.

2. Upgrade Weblate code.
   For pip installs it can be achieved by:
   ```
   pip install -U Weblate
   ```
   With Git checkout you need to fetch new source code and update your installation:
   ```
   cd weblate-src
   git pull
   # Update Weblate inside your virtualenv
   . ~/weblate-env/bin/pip install -e .
   # Install dependencies directly when not using virtualenv
   pip install --upgrade -r requirements.txt
   ```

3. Upgrade configuration file, refer to settings_example.py or Version specific instructions for needed steps.

4. Upgrade database structure:
   ```
   weblate migrate --noinput
   ```

5. Collect updated static files (see and ):
   ```
   weblate collectstatic --noinput
   ```

6. Compress JavaScript and CSS files (optional, see ):
   ```
   weblate compress
   ```

7. Git:
   ```
   weblate compilemessages
   ```

8. Verify that your setup is sane (see also ):
   ```
   weblate check --deploy
   ```

9. Restart Celery worker (see Celery).

173
Version specific instructions

Upgrade from 2.x

If you are upgrading from 2.x release, always first upgrade to 3.0.1 and then continue upgrading in the 3.x series. Upgrades skipping this step are not supported and will break.

Related: Upgrade from 2.20 to 3.0 in Weblate 3.0 documentation

Upgrade from 3.x

If you are upgrading from 3.x release, always first upgrade to 4.0.4 or 4.1.1 and then continue upgrading in the 4.x series. Upgrades skipping this step are not supported and will break.

Related: Upgrade from 3.11 to 4.0 in Weblate 4.0 documentation

Upgrade from 4.0 to 4.1

Please follow [upgrade instructions](#) in order to perform update.

Notable configuration or dependencies changes:

- There are several changes in `settings_example.py`, most notable middleware changes, please adjust your settings accordingly.
- There are new file formats, you might want to include them in case you modified the `WEBLATE_FORMATS`.
- There are new quality checks, you might want to include them in case you modified the `CHECK_LIST`.
- `DEFAULT_THROTTLE_CLASSES` and `API`
- `WEBLATE_SITE_DOMAIN`

There are some new and updated requirements.

Related: [new requirements](#)

Upgrade from 4.1 to 4.2

Please follow [upgrade instructions](#) in order to perform update.

Notable configuration or dependencies changes:

- Upgrade from 3.x releases is no longer supported, please upgrade to 4.0 or 4.1 first.
- There are some new and updated requirements.
- There are several changes in `settings_example.py`, most notable new middleware and changed application ordering.
- The keys for JSON based formats no longer include leading dot. The strings are adjusted during the database migration, but external components might need adjustment in case you rely on keys in exports or API.
- The Celery configuration was changed to no longer use `memory` queue. Please adjust your startup scripts and `CELERY_TASK_ROUTES` setting.
- The Weblate domain is now configured in the settings, see `SITE_DOMAIN` (or `WEBLATE_SITE_DOMAIN`). You will have to configure it before running Weblate.
- The username and email fields on user database now should be case insensitive unique. It was mistakenly not enforced with PostgreSQL.

Related: [new requirements](#)
Upgrade from 4.2 to 4.3

Please follow [Upgrade](#) in order to perform update.
Notable configuration or dependencies changes:
There are some changes in quality checks, you might want to include them in case you modified the `CHECK_LIST`. The source language attribute was moved from project to a component what is exposed in the API. You will need to update Weblate [Upgrade](#) in case you are using it.
The database migration to 4.3 might take long depending on number of strings you are translating (expect around one hour of migration time per 100,000 source strings).
There is a change in `INSTALLED_APPS`.
There is a new setting `SESSION_COOKIE_AGE_AUTHENTICATED` which complements `SESSION_COOKIE_AGE`.
In case you were using hub or lab to integrate with GitHub or GitLab, you will need to reconfigure this, see [GITHUB_CREDENTIALS](#) and [GITLAB_CREDENTIALS](#).

**4.3.1**

The Celery configuration was changed to add memory queue. Please adjust your startup scripts and `CELERY_TASK_ROUTES` setting.

**4.3.2**

The post_update method of addons now takes extra `skip_push` parameter.

Upgrade from 4.3 to 4.4

Please follow [Upgrade](#) in order to perform update.
Notable configuration or dependencies changes:
There is a change in `INSTALLED_APPS`, `weblate.configuration` has to be added there.
Django 3.1 is now required.
In case you are using MySQL or MariaDB, the minimal required versions have increased, see [MySQL](#) [MariaDB](#).

**4.4.1**

`Monolingualgettext` now uses both `msgid` and `msgctxt` when present. This will change identification of translation strings in such files breaking links to Weblate extended data such as screenshots or review states. Please make sure you commit pending changes in such files prior upgrading and it is recommended to force loading of affected component using `loadpo`.
Increased minimal required version of translate-toolkit to address several file format issues.

Upgrade from 4.4 to 4.5

Please follow [Upgrade](#) in order to perform update.
Notable configuration or dependencies changes:
The migration might take considerable time if you had big glossaries.
Glossaries are now stored as regular components.
The glossary API is removed, use regular translation API to access glossaries.
There is a change in `INSTALLED_APPS`, `weblate.metrics` should be added.

**4.5.1**

There is a new dependency on the `pyahocorasick` module.
Upgrade from 4.5 to 4.6

Please follow [Upgradefrom4.5to4.6](#) in order to perform update.
Notable configuration or dependencies changes:
There are new file formats, you might want to include them in case you modified the `WEBLATE_FORMATS`

![API URL](#)

There is a change in dependencies and `PASSWORD_HASHERS` to prefer Argon2 for passwords hashing.

![API URL](#)

Upgrade from 4.6 to 4.7

Please follow [Upgradefrom4.6to4.7](#) in order to perform update.
Notable configuration or dependencies changes:
There are several changes in `settings_example.py`, most notable middleware changes (`MIDDLEWARE`),
please adjust your settings accordingly.
The DeepL machine translation now has a generic `MT_DEEPL_API_URL` setting to adapt to different subscription models more flexibly. The `MT_DEEPL_API_VERSION` setting is no longer used.
Django 3.2 is now required.

![API URL](#)

Upgrade from 4.7 to 4.8

Please follow [Upgradefrom4.7to4.8](#) in order to perform update.
There are no additional upgrade steps needed in this release.

![API URL](#)

Upgrading from Python 2 to Python 3

Weblate no longer supports Python older than 3.5. In case you are still running on older version, please perform migration to Python 3 first on existing version and upgrade later. See [Upgrading from Python 2 to Python 3](#) in the Weblate 3.11.1 documentation.

Migrating from other databases to PostgreSQL

If you are running Weblate on other databas than PostgreSQL, you should consider migrating to PostgreSQL as Weblate performs best with it. The following steps will guide you in migrating your data between the databases. Please remember to stop both web and Celery servers prior to the migration, otherwise you might end up with inconsistent data.

PostgreSQL

```
# If PostgreSQL was not installed before, set the main password
sudo -u postgres psql postgres -c "\password postgres"
# Create a database user called "weblate"
sudo -u postgres createuser -D -P weblate
# Create the database "weblate" owned by "weblate"
sudo -u postgres createdb -E UTF8 -O weblate weblate
```
Migrating using Django JSON dumps

The simplest approach for migration is to utilize Django JSON dumps. This works well for smaller installations. On bigger sites you might want to use pgloader instead, see Migrating to PostgreSQL using pgloader.

1. Add PostgreSQL as additional database connection to the settings.py:

```python
DATABASES = {
    "default": {
        # Database engine
        "ENGINE": "django.db.backends.mysql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "database.example.com",
        # Set to empty string for default
        "PORT": "",
        # Additional database options
        "OPTIONS": {
            # In case of using an older MySQL server, which has MyISAM as-
            # a default storage
            "init_command": 'SET storage_engine=INNODB',
            # Uncomment for MySQL older than 5.7:
            "init_command": "SET sql_mode='STRICT_TRANS_TABLES'",
            # If your server supports it, see the Unicode issues above
            "charset": "utf8mb4",
            # Change connection timeout in case you get MySQL gone away-
            "connect_timeout": 28800,
        },
    },
    "postgresql": {
        # Database engine
        "ENGINE": "django.db.backends.postgresql",
        # Database name
        "NAME": "weblate",
        # Database user
        "USER": "weblate",
        # Database password
        "PASSWORD": "password",
        # Set to empty string for localhost
        "HOST": "database.example.com",
        # Set to empty string for default
        "PORT": "",
    },
}
```

2. Run migrations and drop any data inserted into the tables:

```bash
weblate migrate --database=postgresql
weblate sqlflush --database=postgresql | weblate dbshell --
database=postgresql
```

3. Dump legacy database and import to PostgreSQL

```bash
weblate dumpdata --all --output weblate.json
weblate loaddata weblate.json --database=postgresql
```

4. Adjust DATABASES to use just PostgreSQL database as default, remove legacy connection.

Weblate should be now ready to run from the PostgreSQL database.
Migrating to PostgreSQL using pgloader

The pgloader is a generic migration tool to migrate data to PostgreSQL. You can use it to migrate Weblate database.

1. Adjust your settings.py to use PostgreSQL as a database.
2. Migrate the schema in the PostgreSQL database:

```
weblate migrate
weblate sqlflush | weblate dbshell
```

3. Run the pgloader to transfer the data. The following script can be used to migrate the database, but you might want to learn more about pgloader to understand what it does and tweak it to match your setup:

```
LOAD DATABASE FROM mysql://weblate:password@localhost/weblate INTO postgresql://weblate:password@localhost/weblate
WITH include no drop, truncate, create no tables, create no indexes, no
    foreign keys, disable triggers, reset sequences, data only
ALTER SCHEMA 'weblate' RENAME TO 'public';
```

Migrating from Pootle

As Weblate was originally written as replacement from Pootle, it is supported to migrate user accounts from Pootle. You can dump the users from Pootle and import them using importusers.
Backup service: /tmp/tmpsflwftcwweblate

Backup service credentials

Backup repository: /tmp/tmpsflwftcwweblate

Passphrase: g613vr13zvu0跌幅n8pbzvffg1vswRQKSAJbc5htVqk8j

The passphrase is used to encrypt the backups and is necessary to restore them.

SSH key

Download private key

The private key is needed to access the remote backup repository.

Deleted the oldest backups

Backup performed

Repository initialization

Activate support package

The support packages include priority email support, or cloud backups of your Weblate installation.

Activation token

Please enter the activation token obtained when making the subscription.

Activate

Purchase support package

Add backup service

Backup repository URL

Use /path/to/repo for local backups or user@host:/path/to/repo for remote SSH backups.

Add
Borg

BorgBackup

Weblate

SSH:

`borg init`

Weblate

backup service at weblate.org:

1. https://weblate.org/support/#backup

2.

3.

4.

5. Borg

Weblate

Docker

Docker

`/app/data/borgbackup`

Docker Compose:

```
services:
  weblate:
    volumes:
      - /home/weblate/data:/app/data
      - /home/weblate/borgbackup:/borgbackup
```

180
1. Prepare a server where your backups will be stored.
2. Install the SSH server on it (you will get it by default with most Linux distributions).
3. Install BorgBackup on that server; most Linux distributions have packages available (see Installation).
4. Choose an existing user or create a new user that will be used for backing up.
5. Add Weblate SSH key to the user so that Weblate can SSH to the server without a password (see Weblate SSH).
6. Configure the backup location in Weblate as `user@host:/path/to/backups`.

**Weblate SSH**

**BorgBackup**

1. `borg list REPOSITORY`
2. `borg extract REPOSITORY::ARCHIVE`
3. `Weblate backup SQL` Dumped data for backups
4. `Weblate backups/settings.py`: Dumped data for backups

When using Docker container, the settings file is already included in the container and you should restore the original environment variables. The `environment.yml` file might help you with this (see Dumped data for backups).

5. `DATA_DIR`

When using Docker container place the data into the data volume, see Docker container volumes.

Please make sure the files have correct ownership and permissions, see `DATA_DIR`.

**Borg**

```
$ borg list /tmp/xxx
Enter passphrase for key /tmp/xxx:
2019-09-26T14:56:08 Thu, 2019-09-26 14:56:08...
$ borg extract /tmp/xxx::2019-09-26T14:56:08
Enter passphrase for key /tmp/xxx:
```

**Note:** If you are doing the manual backups, you might want to silence Weblate's warning about a lack of backups by adding `weblate.I028` to `SILENCED_SYSTEM_CHECKS` in `settings.py` or `WEBBLATE_SILENCED_SYSTEM_CHECKS` for Docker.

```
SILENCED_SYSTEM_CHECKS.append("weblate.I028")
```
You can restore this backup in a newer Weblate release, it will perform all the necessary migrations when running in `migrate`. Please consult Weblate documentation on more detailed info on how to upgrade between versions.

Alternatively, you can back up your database using Django’s `dumpdata` command. That way the backup is database agnostic and can be used in case you want to change the database backend.

Prior to restoring the database you need to be running exactly the same Weblate version the backup was made on. This is necessary as the database structure does change between releases and you would end up corrupting the data in some way. After installing the same version, run all database migrations using `migrate`.

Afterwards some entries will already be created in the database and you will have them in the database backup as well. The recommended approach is to delete such entries manually using the management shell (see Invoking management commands):

```
weblate shell
>>> from weblate.auth.models import User
>>> User.objects.get(username='anonymous').delete()
```

If you have enough backup space, simply back up the whole `DATA_DIR`. This is a safe bet even if it includes some files you don’t want. The following sections describe what you should back up and what you can skip in detail.

### Dumped data for backups

**4.7: The environment dump was added as `environment.yml` to help in restoring in the Docker environments.**

Stored in `DATA_DIR/backups`.

Weblate dumps various data here, and you can include these files for more complete backups. The files are updated daily (requires a running Celery beats server, see Celery documentation). Currently, this includes:

- Weblate settings as `settings.py` (there is also expanded version in `settings-expanded.py`).
- PostgreSQL database backup as `database.sql`.
- Environment dump as `environment.yml`.

The database backups are saved as plain text by default, but they can also be compressed or entirely skipped using `DATABASE_BACKUP`.

To restore the database backup load it using database tools, for example:

```
psql --file=database.sql weblate
```
Version control repositories

Stored in `DATA_DIR/vcs`.

The version control repositories contain a copy of your upstream repositories with Weblate changes. If you have enabling for all your translation components, all Weblate changes are included upstream. No need to back up the repositories on the Weblate side as they can be cloned again from the upstream location(s) with no data loss.

SSH and GPG keys

Stored in `DATA_DIR/ssh` and `DATA_DIR/home`.

If you are using SSH or GPG keys generated by Weblate, you should back up these locations. Otherwise you will lose the private keys and you will have to regenerate new ones.

User uploaded files

Stored in `DATA_DIR/media`.

Celery tasks

The Celery task queue might contain some info, but is usually not needed for a backup. At most you will lose updates not yet been processed to translation memory. It is recommended to perform the fulltext or repository update upon restoration anyhow, so there is no problem in losing these.

Command line for manual backup

Using a cron job, you can set up a Bash command to be executed on a daily basis, for example:

```bash
$ XZ_OPT="-9" tar -Jcf ~/backup/weblate-backup-$\{date -u +%Y-%m-%d_%H%M%S\}.
```

The string between the quotes after `XZ_OPT` allows you to choose your xz options, for instance the amount of memory used for compression; see [https://linux.die.net/man/1/xz](https://linux.die.net/man/1/xz)

You can adjust the list of folders and files to your needs. To avoid saving the translation memory (in backups folder), you can use:

```bash
$ XZ_OPT="-9" tar -Jcf ~/backup/weblate-backup-$\{date -u +%Y-%m-%d_%H%M%S\}.
```

Restoring manual backup

1. Restore all data you have backed up.
2. Update all repositories using `updategit`.

```bash
weblate updategit --all
```
Moving a Weblate installation

Relocate your installation to a different system by following the backing up and restoration instructions above.

- Upgrading from Python 2 to Python 3
- Migrating from other databases to PostgreSQL

Welcome to Python Social Auth's documentation!

Django Framework

**REGISTRATION_OPEN**

**FORCE_EMAIL_VALIDATION**

<table>
<thead>
<tr>
<th>socially-authenticating</th>
<th>socially-authenticating</th>
<th>socially-authenticating</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGISTRATION_OPEN</td>
<td>FORCE_EMAIL_VALIDATION</td>
<td></td>
</tr>
<tr>
<td>AUTHENTICATION_BACKENDS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SOCIAL_AUTH_OPENSUSE_FORCE_EMAIL_VALIDATION = True**

<table>
<thead>
<tr>
<th>Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHENTICATION_BACKENDS</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Most of the authentication backends require HTTPS. Once HTTPS is enabled in your web server please configure Weblate to report it properly using `ENABLE_HTTPS`, or by `WEBLATE_ENABLE_HTTPS` in the Docker container.

**Python Social Auth backend**
# Authentication configuration
AUTHENTICATION_BACKENDS = (  
    "social_core.backends.email.EmailAuth",  
    "social_core.backends.suse.OpenSUSEOpenId",  
    "social_core.backends.ubuntu.UbuntuOpenId",  
    "social_core.backends.fedora.FedoraOpenId",  
    "weblate.accounts.auth.WeblateUserBackend",  
)

GitHub

# Authentication configuration
AUTHENTICATION_BACKENDS = (  
    "social_core.backends.github.GithubOAuth2",  
    "social_core.backends.email.EmailAuth",  
    "weblate.accounts.auth.WeblateUserBackend",  
)

# Social auth backends setup
SOCIAL_AUTH_GITHUB_KEY = "GitHub Client ID"
SOCIAL_AUTH_GITHUB_SECRET = "GitHub Client Secret"
SOCIAL_AUTH_GITHUB_SCOPE = ["user:email"]

Bitbucket

# Authentication configuration
AUTHENTICATION_BACKENDS = (  
    "social_core.backends.bitbucket.BitbucketOAuth",  
    "social_core.backends.email.EmailAuth",  
    "weblate.accounts.auth.WeblateUserBackend",  
)

# Social auth backends setup
SOCIAL_AUTH_BITBUCKET_KEY = "Bitbucket Client ID"
SOCIAL_AUTH_BITBUCKET_SECRET = "Bitbucket Client Secret"
SOCIAL_AUTH_BITBUCKET_VERIFIED_EMAILS_ONLY = True
Google OAuth 2

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (  
    "social_core.backends.google.GoogleOAuth2",  
    "social_core.backends.email.EmailAuth",  
    "weblate.accounts.auth.WeblateUserBackend",  
)

# Social auth backends setup
SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = "Client ID"
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = "Client secret"
```

Facebook OAuth 2

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (  
    "social_core.backends.facebook.FacebookOAuth2",  
    "social_core.backends.email.EmailAuth",  
    "weblate.accounts.auth.WeblateUserBackend",  
)

# Social auth backends setup
SOCIAL_AUTH_FACEBOOK_KEY = "key"
SOCIAL_AUTH_FACEBOOK_SECRET = "secret"
SOCIAL_AUTH_FACEBOOK_SCOPE = ["email", "public_profile"]
```

GitLab OAuth 2

```
# Authentication configuration
AUTHENTICATION_BACKENDS = (  
    "social_core.backends.gitlab.GitLabOAuth2",  
    "social_core.backends.email.EmailAuth",  
    "weblate.accounts.auth.WeblateUserBackend",  
)

# Social auth backends setup
```
SOCIAL_AUTH_GITLAB_KEY = "Application ID"
SOCIAL_AUTH_GITLAB_SECRET = "Secret"
SOCIAL_AUTH_GITLAB_SCOPE = ["read_user"]

# If you are using your own GitLab
# SOCIAL_AUTH_GITLAB_API_URL = 'https://gitlab.example.com/'

### Microsoft Azure Active Directory

Weblate URL URL

GitLab

# Azure AD common
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.azuread.AzureADOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# OAuth2 keys
SOCIAL_AUTH_AZUREAD_OAUTH2_KEY = 
SOCIAL_AUTH_AZUREAD_OAUTH2_SECRET = 

# Azure AD Tenant
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.azuread_tenant.AzureADTenantOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# OAuth2 keys
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_KEY = 
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_SECRET = 

# Tenant ID
SOCIAL_AUTH_AZUREAD_TENANT_OAUTH2_TENANT_ID = 

Microsoft Azure Active Directory
Slack

Slack OAuth 2 URL: [https://api.slack.com/apps](https://api.slack.com/apps)

URL: [https://WEBLATE SERVER/accounts/complete/slack/](https://WEBLATE SERVER/accounts/complete/slack/)

```python
# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.slack.SlackOAuth2",
    "social_core.backends.email.EmailAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Social auth backends setup
SOCIAL_AUTH_SLACK_KEY = ""
SOCIAL_AUTH_SLACK_SECRET = ""
```

---

**Overriding authentication method names and icons**

You can override the authentication method display name and icon using settings as `SOCIAL_AUTH_<NAME>_IMAGE` and `SOCIAL_AUTH_<NAME>_TITLE`. For example overriding naming for Auth0 would look like:

```python
SOCIAL_AUTH_AUTH0_IMAGE = "custom.svg"
SOCIAL_AUTH_AUTH0_TITLE = "Custom auth"
```

---

**openSUSE Open ID**

```python
# Authentication configuration
AUTHENTICATION_BACKENDS = (   "social_core.backends.suse.OpenSUSEOpenId",
    "weblate.accounts.auth.WeblateUserBackend",
)
```

---

**settings.py**

```
AUTH_PASSWORD_VALIDATORS = [    # A sample password validation settings
    
]
```

---

**188**
SAML

4.1.1.

Python Social Auth

Webate IDP

SOCIAL_AUTH_SAML_ENABLED_IDPS

weblate

SAML XML URL /

SOCIAL_AUTH_SAML_SP_ENTITY_ID

SOCIAL_AUTH_SAML_TECHNICAL_CONTACT

SOCIAL_AUTH_SAML_SUPPORT_CONTACT

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.email.EmailAuth",
    "social_core.backends.saml.SAMLAuth",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Social auth backends setup
SOCIAL_AUTH_SAML_SP_ENTITY_ID = f"https://{SITE_DOMAIN}/accounts/metadata/saml/

SOCIAL_AUTH_SAML_SP_PUBLIC_CERT = "-----BEGIN CERTIFICATE-----"

SOCIAL_AUTH_SAML_SP_PRIVATE_KEY = "-----BEGIN PRIVATE KEY-----"

SOCIAL_AUTH_SAML_ENABLED_IDPS = {
    "weblate": {
        "entity_id": "https://idp.testshib.org/idp/idibboleth",
        "url": "https://idp.testshib.org/idp/profile/SAML2/Redirect/SSO",
        "x509cert": "MIIEjDCCAqAwIBAgIBADAlBgkqhkiG9w0BAQsFADBTVzCCAR8wEQQD
        "attr_name": "full_name",
        "attr_username": "username",
        "attr_email": "email",
    }
}

SOCIAL_AUTH_SAML_ORG_INFO = {
    "en-US": {
        "name": "example",
        "displayname": "Example Inc.",
        "url": "http://example.com"
    }
}

SOCIAL_AUTH_SAML_TECHNICAL_CONTACT = {
    "givenName": "Tech Gal",
    "emailAddress": "technical@example.com"
}

SOCIAL_AUTH_SAML_SUPPORT_CONTACT = {
    "givenName": "Support Guy",
    "emailAddress": "support@example.com"
}

The default configuration extracts user details from following attributes, configure your IDP to provide them:

SAML URI

urn:oid:2.5.4.3

urn:oid:2.5.4.42

urn:oid:2.5.4.4

urn:oid:0.9.2342.19200300.100.1.3

urn:oid:0.9.2342.19200300.100.1.1

Configuring SAML in Docker

189
LDAP

LDAP

django-auth-ldap

# Using PyPI
pip install django-auth-ldap>=1.3.0

# Using apt-get
apt-get install python-django-auth-ldap

Docker

Docker

Python LDAP 3.1.0

AttributeError: 'module' object has no attribute '_trace_level'

Django

# Add LDAP backed, keep Django one if you want to be able to sign in
# even without LDAP for admin account
AUTHENTICATION_BACKENDS = (
    "django_auth_ldap.backend.LDAPBackend",
    "weblate.accounts.auth.WeblateUserBackend",
)

# LDAP server address
AUTH_LDAP_SERVER_URI = "ldaps://ldap.example.net"

# DN to use for authentication
AUTH_LDAP_USER_DN_TEMPLATE = "cn=%(user)s, o=Example"

# Depending on your LDAP server, you might use a different DN
# like:
# AUTH_LDAP_USER_DN_TEMPLATE = 'ou=users, dc=example, dc=com'

# List of attributes to import from LDAP upon sign in
# Weblate stores full name of the user in the full_name attribute
AUTH_LDAP_USER_ATTR_MAP = {
    "full_name": "name",
    # Use the following if your LDAP server does not have full name
    # Weblate will merge them later
    # 'first_name': 'givenName',
    # 'last_name': 'sn',
    # Email is required for Weblate (used in VCS commits)
    "email": "mail",
}

# Hide the registration form
REGISTRATION_OPEN = False

AUTHENTICATION_BACKENDS = 'social_core.backends.email.EmailAuth'
accounts.auth.WeblateUserBackend'
createadmin'
import ldap
from django_auth_ldap.config import LDAPSearch

AUTH_LDAP_BIND_DN = ""
AUTH_LDAP_BIND_PASSWORD = ""
AUTH_LDAP_USER_SEARCH = LDAPSearch("ou=users,dc=example,dc=com", ldap.SCOPE_SUBTREE, "(uid=$(user)s)")

Active Directory

import ldap
from django_auth_ldap.config import LDAPSearch, NestedActiveDirectoryGroupType

AUTH_LDAP_BIND_DN = "CN=ldap,CN=Users,DC=example,DC=com"
AUTH_LDAP_BIND_PASSWORD = "password"

# User and group search objects and types
AUTH_LDAP_USER_SEARCH = LDAPSearch("CN=Users,DC=example,DC=com", ldap.SCOPE_SUBTREE, "(sAMAccountName=$(user)s)")

# Make selected group a superuser in Weblate
AUTH_LDAP_USER_FLAGS_BY_GROUP = {
    "is_superuser": "CN=weblate_AdminUsers,OU=Groups,DC=example,DC=com",
}

# Map groups from AD to Weblate
AUTH_LDAP_GROUP_SEARCH = LDAPSearch("OU=Groups,DC=example,DC=com", ldap.SCOPE_SUBTREE, "(objectClass=group)")

AUTH_LDAP_GROUP_TYPE = NestedActiveDirectoryGroupType()

# Optionally enable group mirroring from LDAP to Weblate
# AUTH_LDAP_MIRROR_GROUPS = True

Django Authentication Using LDAP

CAS

django-cas-ng

1. CAS

2. Weblate

settings.py:

# Add CAS backed, keep the Django one if you want to be able to sign in
# even without LDAP for the admin account
AUTHENTICATION_BACKENDS = (}
"django_cas_ng.backends.CASBackend",
"weblate.accounts.auth.WeblateUserBackend",
)

# CAS server address
CAS_SERVER_URL = "https://cas.example.net/cas/"

# Add django_cas_ng somewhere in the list of INSTALLED_APPS
INSTALLED_APPS = (... "django_cas_ng")

from django_cas_ng.signals import cas_user_authenticated
from django.dispatch import receiver

@receiver(cas_user_authenticated)
def update_user_email_address(sender, user=None, attributes=None, **kwargs):
    # If your CAS server does not always include the email attribute
    # you can wrap the next two lines of code in a try/catch block.
    user.email = attributes["email"]
    user.save()
If you are not administrating the whole Weblate installation and just have access to manage certain projects (like on Hosted Weblate), your access control management options are limited to following settings. If you don’t need any complex setup, those are sufficient for you.

You can limit user’s access to individual projects by selecting a different Access control setting. Available options are:

- **Public**: Visible to everybody. Any authenticated user can contribute. VCS repository might be exposed to everybody.
- **Protected**: Visible only to chosen users. Only chosen users can contribute. Only chosen users can access VCS repository.
- **Private**: Visible only to chosen users. Only chosen users can contribute. Only chosen users can access VCS repository.
- **Custom**: Permissions are not managed in Weblate. Only use this if you know what you are doing, enabling it might revoke your access to this project.

**User management** features will be disabled; by default all users are forbidden to performed any actions on the project. You will have to set up all the permissions using custom settings.

Access control can be changed in the **Access** tab of the configuration (**Manage ↓ Settings**) of each respective project.

Even for Private projects, some info about your project will be exposed: statistics and language summary for the whole instance will include counts for all projects despite the access control setting. Your project name and other information can’t be revealed through this.

The actual set of permissions available for users by default in Public, Protected, and Private projects can be redefined by Weblate instance administrator using custom settings.
For Public, Protected and Private projects:

For Protected and Private projects only:

VCS

Unfortunately, it’s not possible to change this predefined set of groups for now. Also this way it’s not possible to give just some additional permissions to all users.

For non-Custom access control an instance of each group described above is actually defined for each project. The actual name of those groups will be Project@Group, also displayed in the Django admin interface this way. Although they can’t be edited from Weblate user-interface.
These features are available on the Access control page, which can be accessed from the project’s menu Manage ↓ Users.

Also, besides adding an existing user to the project, it is possible to invite new ones. Any new user will be created immediately, but the account will remain inactive until signing in with a link in the invitation sent via an e-mail. It is not required to have any site-wide privileges in order to do so, access management permission on the project’s scope (e.g. a membership in the Administration group) would be sufficient.

**Note:** If the invited user missed the validity of the invitation, they can set their password using invited e-mail address in the password reset form as the account is created already.

**Note:** If the invited user missed the validity of the invitation, they can set their password using invited e-mail address in the password reset form as the account is created already.

The same kind of invitations are available site-wide from the management interface on the Users tab.
In case some users behave badly in your project, you have an option to block them from contributing. The blocked user still will be able to see the project if he has permissions for that, but he won't be able to contribute.

By default this prevents Weblate from granting access provided by Users and Viewers default groups due to these groups' own configuration. This doesn't prevent you from granting permissions to those projects site-wide by altering default groups, creating a new one, or creating additional custom settings for individual component as described in below.

One of the main benefits of managing permissions through the Weblate user interface is that you can delegate it to other users without giving them the superuser privilege. In order to do so, add them to the Administration group of the project.

The most powerful features of the Weblate's access control system for now are available only through the Django admin interface. You can use it to manage permissions of any project. You don’t necessarily have to switch it to Custom access control to utilize it. However you must have superuser privileges in order to use it.

If you are not interested in details of implementation, and just want to create a simple-enough configuration based on the defaults, or don’t have a site-wide access to the whole Weblate installation (like on Hosted Weblate), please refer to the section.

This section contains an overview of some common configurations you may be interested in.

To manage permissions for a whole instance at once, add users to appropriate default groups:

Users (this is done by default by the automatic group assignment).

Reviewers (if you are using review workflow with dedicated reviewers).

Managers (if you want to delegate most of the management operations to somebody else).

You should keep all projects configured as Public (see ), otherwise the site-wide permissions provided by membership in the Users and Reviewers groups won’t have any effect.

You may also grant some additional permissions of your choice to the default groups. For example, you may want to give a permission to manage screenshots to all the Users.

You can define some new custom groups as well. If you want to keep managing your permissions site-wide for these groups, choose an appropriate value for the Project selection (e.g. All projects or All public projects).
You can create your own dedicated groups to manage permissions for distinct objects such as languages, components, and projects. Although these groups can only grant additional privileges, you can’t revoke any permission granted by site-wide or per-project groups by adding another custom group.

If you want (for whatever reason) to allow translation to a specific language (let’s say Czech) only to a closed set of reliable translators while keeping translations to other languages public, you will have to:

1. Remove the permission to translate Czech from all the users. In the default configuration this can be done by altering the Users default group.

   Table2

<table>
<thead>
<tr>
<th>Role</th>
<th>Project</th>
<th>Language</th>
<th>Components</th>
<th>Component list</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All but Czech</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Add a dedicated group for Czech translators.

   Table3

<table>
<thead>
<tr>
<th>Role</th>
<th>Project</th>
<th>Language</th>
<th>Components</th>
<th>Component list</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Add users you wish to give the permissions to into this group.

   As you can see, permissions management this way is powerful, but can be quite a tedious job. You can’t delegate it to another user, unless granting superuser permissions.

---

**Note:** A group can have no roles assigned to it, in that case access to browse the project by anyone is assumed (see below).
The scope of the permission assigned by the roles in the groups are applied by the following rules:

If the group specifies any Component list, all the permissions given to members of that group are granted for all the components in the component lists attached to the group, and an access with no additional permissions is granted for all the projects these components are in. Components and Projects are ignored.

If the group specifies any Components, all the permissions given to the members of that group are granted for all the components attached to the group, and an access with no additional permissions is granted for all the projects these components are in. Projects are ignored.

Otherwise, if the group specifies any Projects, either by directly listing them or by having Projects selection set to a value like All public projects, all those permissions are applied to all the projects, which effectively grants the same permissions to access all projects unrestricted components.

The restrictions imposed by a group’s Languages are applied separately, when it’s verified if a user has an access to perform certain actions. Namely, it’s applied only to actions directly related to the translation process itself like reviewing, saving translations, adding suggestions, etc.

Let’s say there is a project foo with the components: foo/bar and foo/baz and the following group:

Table4Group Spanish Admin-Reviewers

Members of that group will have following permissions (assuming the default role settings):

General (browsing) access to the whole project foo including both components in it: foo/bar and foo/baz.

Review strings in foo/bar Spanish translation (not elsewhere).

Manage VCS for the whole foo/bar repository e.g. commit pending changes made by translators for all languages.

Django admin interface

The most common use-case for the feature is to assign all new users to some default group. In order to do so, you will probably want to keep the default value (.*$) in the regular expression field. Another use-case for this option might be to give some additional privileges to employees of your company by default. Assuming all of them use corporate e-mail addresses on your domain, this can be accomplished with an expression like .*@mycompany.com.

As for now, there is no way to bulk-add already existing users to some group via the user interface. For that, you may resort to using the REST API.
After installation, a default set of groups is created (see Administration).

[Administration, Billing]

[Administration, Edit source, Power user, Review strings, Translate]

[Administration, Manage glossary, Power user]

[Administration, Manage glossary, Power user]

[Administration, Manage glossary, Power user]

[Administration, Edit source, Power user, Review strings, Translate]

[Administration, Manage translation memory]

[Administration, Manage translation memory]

[Administration, Manage translation memory]

[Administration]

[Administration]

[Administration, Manage screenshots]

[Administration, Manage screenshots]

[Administration, Manage screenshots]

[Administration, Edit source]

[Administration]

Dismiss failing check [Administration, Edit source, Power user, Review strings, Translate]

[Administration, Edit source, Power user, Review strings, Translate]

[Administration, Review strings]

[Administration, Review strings]

[Administration, Edit source, Power user]

[Administration, Edit source, Power user, Review strings, Translate]

[Administration, Edit source, Power user, Review strings, Translate]

[Administration, Power user]

[Administration, Edit source, Power user, Review strings, Translate]

[Administration, Power user, Manage languages]

[Administration, Manage languages]

[Administration, Manage languages]

[Administration, Manage languages]

[Administration, Manage languages]

[Administration, Edit source, Power user, Review strings, Translate]

[Administration, Edit source, Power user, Review strings, Translate]

[Administration, Access repository, Power user, Manage repository]

[Administration, Manage repository]
Weblate

setupgroups

Add suggestion

automatic group assignment: none

automatic group assignment: Power user

Review strings

Administration
If you want to use your Weblate installation in a less public manner, i.e. allow new users on an invitational basis only, it can be done by configuring Weblate in such a way that only known users have an access to it. In order to do so, you can set `REGISTRATION_OPEN` to `False` to prevent registrations of any new users, and set `REQUIRE_LOGIN` to `/.*` to require logging-in to access all the site pages. This is basically the way to lock your Weblate installation.

---

You can use built-in invitations to add new users.

---

**Translation organization**

Weblate organizes translatable VCS content of project/components into a tree-like structure. The bottom level object is *Project configuration*, which should hold all translations belonging together (for example translation of an application in several versions and/or accompanying documentation).

On the level above, *Component configuration*, which is actually the component to translate, you define the VCS repository to use, and the mask of files to translate.

Above *Component configuration* there are individual translations, handled automatically by Weblate as translation files (which match *File mask* defined in *Component configuration*) appear in the VCS repository.

Weblate supports a wide range of translation formats (both bilingual and monolingual ones) supported by Translate Toolkit, see [documentation](https://translate Toolkit.readthedocs.io).

---

You can share cloned VCS repositories using [Weblate URL](https://weblate.org). Using this feature is highly recommended when you have many components sharing the same VCS. It improves performance and decreases required disk space.

---

**Adding translation projects and components**

**3.2**: An interface for adding projects and components is included, and you no longer have to use Django.

**3.4**: The process of adding components is now multi staged, with automated discovery of most parameters.

Based on your permissions, new translation projects and components can be created. It is always permitted for users with the *Add new projects* permission, and if your instance uses billing (e.g. like [https://hosted.weblate.org/](https://hosted.weblate.org/)), you can also create those based on your plans allowance from the user account that manages billing.

You can view your current billing plan on a separate page:
The project creation can be initiated from there, or using the menu in the navigation bar, filling in basic info about the translation project to complete addition of it:

After creating the project, you are taken directly to the project page:
Creating a new translation component can be initiated via a single click there. The process of creating a component is multi-staged and automatically detects most translation parameters. There are several approaches to creating component:

- Creates component from remote version control repository.
- Creates additional component to existing one by choosing different files.
- Creates additional component to existing one, just for different branch.
- Upload translation files to Weblate in case you do not have version control or do not want to integrate it with Weblate.
  You can later update the content using the web interface or Weblate REST API.
- Upload single document or translation file and translate that.
- Create blank translation project and add strings manually.
- Once you have existing translation components, you can also easily add new ones for additional files or branches using same repository.
- First you need to fill in name and repository location:
On the next page, you are presented with a list of discovered translatable resources:

As a last step, you review the translation component info and fill in optional details:
Project configuration

These basic attributes set up and inform translators of a project:

Verbose project name, used to display the project name.

URL

Project name suitable for URLs.

Web

URL where translators can find more info about the project.
This is a required parameter unless turned off by `WEBSITE_REQUIRED`.

Text describing localization process in the project, and any other information useful for translators. Markdown can be used for text formatting or inserting links.

"Language-Team"

Whether Weblate should manage the `Language-Team` header (this is a GNU gettext only feature right now).

Whether to use shared translation memory, see [for more details.](#)

Whether to contribute to shared translation memory, see [for more details.](#)
Configure per project access control, see [DEFAULT_ACCESS_CONTROL] for more details. Default value can be changed by DEFAULT_ACCESS_CONTROL.

Enable review workflow for translations, see [报告].

Enable review workflow for source strings, see [报告].

Whether unauthenticated [报告] are to be used for this repository.

Define language codes mapping when importing translations into Weblate. Use this when language codes are inconsistent in your repositories and you want to get a consistent view in Weblate or in case you want to use non-standard naming of your translation files.

The typical use case might be mapping American English to English: en_US:en
Multiple mappings to be separated by comma: en_GB:en,en_US:en
Using non standard code: ia_FOO:ia

The language codes are mapped when matching the translation files and the matches are case sensitive, so make sure you use the source language codes in same form as used in the filenames.

Component configuration

A component is a grouping of something for translation. You enter a VCS repository location and file mask for which files you want translated, and Weblate automatically fetches from this VCS, and finds all matching translatable files.

/devel/integration
You can find some examples of typical configurations in the [报告].

It is recommended to keep translation components to a reasonable size - split the translation by anything that makes sense in your case (individual apps or addons, book chapters or websites).

Weblate easily handles translations with 10000s of strings, but it is harder to split work and coordinate among translators with such large translation components.

Should the language definition for a translation be missing, an empty definition is created and named as "cs_CZ (generated)". You should adjust the definition and report this back to the Weblate authors, so that the missing languages can be included in next release.

The component contains all important parameters for working with the VCS, and for getting translations out of it:
Verbose component name, used to display the component name.

Component name suitable for URLs.

**Component project**

*Project configuration* where the component belongs.

VCS to use, see [URL] for details.

*Pushing changes from Weblate*

VCS repository used to pull changes.

See [URL] for more details on specifying URLs.

**Note:** This can either be a real VCS URL or weblate://project/component indicating that the repository should be shared with another component. See *Weblate URL* for more details.

**URL**

Repository URL used for pushing. This setting is used only for *Git* and *Mercurial* and push support is turned off for these when this is empty.

See [URL] for more details on how to specify a repository URL and *Pushing changes from Weblate* for more details on pushing changes from Weblate.

**URL of repository browser** used to display source files (location of used messages). When empty, no such links will be generated. You can use [URL].

For example on GitHub, use something like: https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename}}#L{{line}}

In case your paths are relative to different folder (path contains ..), you might want to strip leading directory by `parentdir` filter (see [URL]): https://github.com/WeblateOrg/hello/blob/{{branch}}/{{filename|parentdir}}#L{{line}}
URL

URL where changes made by Weblate are exported. This is important when **[URL]** is not used, or when there is a need to manually merge changes. You can use **Git exporter** to automate this for Git repositories.

Which branch to checkout from the VCS, and where to look for translations.

**push**

Branch for pushing changes, leave empty to use **[push]**.

**push**

This is currently only supported for Git, GitLab and GitHub, it is ignored for other VCS integrations.

**push**

*Pushing changes from Weblate*

**File mask**

Mask of files to translate, including path. It should include one "*" replacing language code (see **[Filemask]** for info on how this is processed). In case your repository contains more than one translation file (e.g. more gettext domains), you need to create a component for each of them.

**push**

*po/*.po, locale/*/LC_MESSAGES/django.po*

In case your filename contains special characters such as [ ], these need to be escaped as [ ] or [ ].

**push**

*What does mean "There are more files for the single language (en)"?*

**Base file**

Base file containing string definitions for **[Base file]**.

**Base file**

*What does mean "There are more files for the single language (en)"?*

**Allow editing base file**

Whether to allow editing the base file for **[Allow editing base file]**.

**Intermediate language file**

Intermediate language file for **[Intermediate language file]**. In most cases this is a translation file provided by developers and is used when creating actual source strings.

When set, the source strings are based on this file, but all other languages are based on **[Intermediate language file]**. In case the string is not translated into the source language, translating to other languages is prohibited. This provides

**Intermediate language file**

*What does mean "There are more files for the single language (en)"?*
Base file used to generate new translations, e.g. `.pot` file with gettext.

In many monolingual formats Weblate starts with blank file by default. Use this in case you want to have all strings present with empty value when creating new translation.

What does mean “There are more files for the single language (en)”?

Translation file format, see also [here].

Email address used for reporting upstream bugs. This address will also receive notification about any source string comments made in Weblate.

You can turn off propagation of translations to this component from other components within same project. This really depends on what you are translating, sometimes it’s desirable to have make use of a translation more than once. It’s usually a good idea to turn this off for monolingual translations, unless you are using the same IDs across the whole project.

Default value can be changed by `DEFAULT_TRANSLATION_PROPAGATION`.

Keeping translations same across components

Whether translation suggestions are accepted for this component.

Turns on vote casting for suggestions, see [here].

Automatically accept voted suggestions, see [here].

Weblate [here].
List of checks which can not be ignored, see \texttt{Enforcing the check does not automatically enable it, you still should enabled it using \texttt{in Additional info on source strings}.}

License of the translation (does not need to be the same as the source code license).

How to handle requests for creation of new languages. Available options:

- User can select desired language and the project maintainers will receive a notification about this. It is up to them to add (or not) the language to the repository.
- User is presented a link to page which describes process of starting new translations. Use this in case more formal process is desired (for example forming a team of people before starting actual translation).
- User can select language and Weblate automatically creates the file for it and translation can begin.

There will be no option for user to start new translation.

The project admins can add new translations even if it is disabled here when it is possible (either or the file format supports starting from an empty file).

Configures whether users in Weblate will be allowed to add new strings and remove existing ones. Adjust this to match your localization workflow - how the new strings are supposed to be introduced.

For bilingual formats, the strings are typically extracted from the source code (for example by using \texttt{xgettext}) and adding new strings in Weblate should be disabled (they would be discarded next time you update the translation files). In Weblate you can manage strings for every translation and it does not enforce the strings in all translations to be consistent.

For monolingual formats, the strings are managed only on source language and are automatically added or removed in the translations. The strings appear in the translation files once they are translated.
You can configure how updates from the upstream repository are handled. This might not be supported for some VCSs. See Merge or rebase for more details.

Default value can be changed by DEFAULT_MERGE_STYLE.

**Commit, add, delete, merge and addon messages**

Message used when committing a translation, see DEFAULT_COMMIT_MESSAGE.

Message used when adding a translation, see DEFAULT_ADD_MESSAGE.

Message used when adding to an existing translation, see DEFAULT_ADDON_MESSAGE.

Whether committed changes should be automatically pushed to the upstream repository. When enabled, the push is initiated once Weblate commits changes to its underlying repository (see Lazy commits). To actually enable pushing Repository push URL has to be configured as well.

Sets how old (in hours) changes have to be before they are committed by background task or the commit_pending management command. All changes in a component are committed once there is at least one change older than this period.

Default value can be changed by COMMIT_PENDING_HOURS.

There are other situations where pending changes might be committed, see Lazy commits.

Locks the component (and linked components, see Weblate URL) upon the first failed push or merge into its upstream repository, or pull from it. This avoids adding another conflicts, which would have to be resolved manually.

The component will be automatically unlocked once there are no repository errors left.

Language used for source strings. Change this if you are translating from something else than English.

In case you are translating bilingual files from English, but want to be able to do fixes in the English translation as well, choose English (Developer) as a source language to avoid conflict between the name of the source language and the existing translation.

For monolingual translations, you can use intermediate translation in this case, see DEFAULT_LANGUAGE.
You need to list language codes as they appear in the filename.

Some examples of filtering:

<table>
<thead>
<tr>
<th>Filter description</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected languages only</td>
<td>^([cs</td>
</tr>
<tr>
<td>Exclude languages</td>
<td>^(?!(it</td>
</tr>
<tr>
<td>Filter two letter codes only</td>
<td>^..$</td>
</tr>
<tr>
<td>Exclude non language files</td>
<td>^(?!(blank)$).+$</td>
</tr>
<tr>
<td>Include all files (default)</td>
<td>^[^.]$</td>
</tr>
</tbody>
</table>

variants:

Does Weblate support other VCSes than Git and Mercurial?

Restricting access at a component, or component-list level takes over access permission to a component, regardless of project-level permissions. You will have to grant access to it explicitly. This can be done through granting access to a new user group and putting users in it, or using the default custom or private access control groups.
4.5 Web.
The glossary will be accessible in all projects defined by `glossary`.

It is recommended to enable `glossary` on glossaries in order to allow adding new words to them.

Display color for a glossary used when showing word matches.

Weblate The Django Template Language Component configuration

```python
parentdir: {{filename|parentdir}}
```
{% component|title %}

{% if stats.translated_percent > 80 %}Well translated!{% endif %}

{% replace component "-" " " %}

{% replace component|capfirst "-" " " %}

Directory of a file: {{ filename|dirname }}
File without extension: {{ filename|stripext }}
File in parent dir: {{ filename|parentdir }}
It can be used multiple times: {{ filename|parentdir|parentdir }}

... boyfriend Django ...

VCS Weblate: Weblate

Weblate ref: production
Celery

— Weblate

CPU

CHECK_LIST

215
Weblate ISO 639-1:

-- cs_CZ (cs, cs)

Weblate (xx_XX (generated)): Weblate

Extending built-in language definitions
This is the process:

1. Developers make changes and push them to the VCS repository.
2. Optionally the translation files are updated (this depends on the file format, see Why does Weblate still show old translation strings when I’ve updated the template?).
3. Weblate pulls changes from the VCS repository, see Updating repositories.
4. Once Weblate detects changes in translations, translators are notified based on their subscription settings.
5. Translators submit translations using the Weblate web interface, or upload offline changes.
6. Once the translators are finished, Weblate commits the changes to the local repository (see Lazy commits) and pushes them back if it has permissions to do so (see Pushing changes from Weblate).
Updating repositories

You should set up some way of updating backend repositories from their source.

Automatically receiving changes from GitHub
Automatically receiving changes from GitLab
Automatically receiving changes from Bitbucket
Pagure

Automatically receiving changes from Azure Repos

Weblate REST API

Avoiding merge conflicts

The merge conflicts from Weblate arise when same file was changed both in Weblate and outside it. There are two approaches to deal with that - avoid edits outside Weblate or integrate Weblate into your updating process, so that it flushes changes prior to updating the files outside Weblate.

The first approach is easy with monolingual files - you can add new strings within Weblate and leave whole editing of the files there. For bilingual files, there is usually some kind of message extraction process to generate translatable files from the source code. In some cases this can be split into two parts - one for the extraction generates template (for example gettext POT is generated using xgettext) and then further process merges it into actual translations (the gettext PO files are updated using msgmerge). You can perform the second step within Weblate and it will make sure that all pending changes are included prior to this operation.

The second approach can be achieved by using Weblate REST API to force Weblate to push all pending changes and lock the translation while you are doing changes on your side.

The script for doing updates can look like this:

```bash
# Lock Weblate translation
wlc lock
# Push changes from Weblate to upstream repository
wlc push
# Pull changes from upstream repository to your local copy
git pull
# Update translation files, this example is for Django
./manage.py makemessages --keep-pot -a
git commit -m 'Locale updates' -- locale
# Push changes to upstream repository
git push
# Tell Weblate to pull changes (not needed if Weblate follows your repo
# automatically)
wlc pull
# Unlock translations
wlc unlock
```

If you have multiple components sharing same repository, you need to lock them all separately:

```bash
wlc lock foo/bar
wlc lock foo/baz
wlc lock foo/baj
```

The example uses Weblate, which needs configuration (API keys) to be able to control Weblate remotely. You can also achieve this using any HTTP client instead of wlc, e.g. curl, see Weblate REST API.

Weblate
Automatically receiving changes from GitHub

Weblate comes with native support for GitHub.

If you are using Hosted Weblate, the recommended approach is to install the Weblate app, that way you will get the correct setup without having to set much up. It can also be used for pushing changes back.

To receive notifications on every push to a GitHub repository, add the Weblate Webhook in the repository settings (Webhooks) as shown on the image below:

For the payload URL, append /hooks/github/ to your Weblate URL, for example for the Hosted Weblate service, this is https://hosted.weblate.org/hooks/github/.

You can leave other values at default settings (Weblate can handle both content types and consumes just the push event).

POST /hooks/github/

Automatically receiving changes from Bitbucket

Weblate has support for Bitbucket webhooks, add a webhook which triggers upon repository push, with destination to /hooks/bitbucket/ URL on your Weblate installation (for example https://hosted.weblate.org/hooks/bitbucket/).
Automatically receiving changes from GitLab

Weblate has support for GitLab hooks, add a project webhook with destination to /hooks/gitlab/ URL on your Weblate installation (for example https://hosted.weblate.org/hooks/gitlab/).

Pagure

Pagure has support for Weblate webhooks, add a project webhook with destination to /hooks/pagure/ URL on your Weblate installation (for example https://hosted.weblate.org/hooks/pagure/).
POST /hooks/pagure/

Automatically receiving changes from Azure Repos

Weblate has support for Azure Repos web hooks, add a webhook for Code pushed event with destination to /hooks/azure/ URL on your Weblate installation (for example https://hosted.weblate.org/hooks/azure/). This can be done in Service hooks under Project settings.

Web hooks in Azure DevOps manual
Automatically receiving changes from Gitea Repos

Weblate has support for Gitea webhooks, add a Gitea Webhook for Push events event with destination to /hooks/gitea/ URL on your Weblate installation (for example https://hosted.weblate.org/hooks/gitea/). This can be done in Webhooks under repository Settings.

Webhooks in Gitea manual

Automatically receiving changes from Gitee Repos

Weblate has support for Gitee webhooks, add a WebHook for Push event with destination to /hooks/gitee/ URL on your Weblate installation (for example https://hosted.weblate.org/hooks/gitee/). This can be done in WebHooks under repository Management.

Webhooks in Gitee manual

Automatically updating repositories nightly

Weblate automatically fetches remote repositories nightly to improve performance when merging changes later. You can optionally turn this into doing nightly merges as well, by enabling AUTO_UPDATE.

Pushing changes from Weblate

Each translation component can have a push URL set up (see URL), and in that case Weblate will be able to push change to the remote repository. Weblate can be also be configured to automatically push changes on every commit (this is default, see AUTO_UPDATE). If you do not want changes to be pushed automatically, you can do that manually under Repository maintenance or using API via wlc push.

The push options differ based on the component used, more details are found in that chapter.

<table>
<thead>
<tr>
<th>Desired setup</th>
<th>URL</th>
<th>push</th>
</tr>
</thead>
<tbody>
<tr>
<td>No push</td>
<td></td>
<td>empty</td>
</tr>
<tr>
<td>Push directly</td>
<td>Git</td>
<td>SSH URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>empty</td>
</tr>
<tr>
<td>SSH URL push</td>
<td>Git</td>
<td>SSH URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>empty</td>
</tr>
<tr>
<td>GitHub pull request from fork</td>
<td>GitHub</td>
<td>SSH URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>empty</td>
</tr>
<tr>
<td>GitHub pull request from branch</td>
<td>GitHub</td>
<td>SSH URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>empty</td>
</tr>
<tr>
<td>GitLab pull request from fork</td>
<td>GitLab</td>
<td>SSH URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>empty</td>
</tr>
<tr>
<td>GitLab merge request from branch</td>
<td>GitLab</td>
<td>SSH URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>empty</td>
</tr>
<tr>
<td>Pagure pull request from fork</td>
<td>Pagure</td>
<td>SSH URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>empty</td>
</tr>
<tr>
<td>Pagure pull request from branch</td>
<td>Pagure</td>
<td>SSH URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>empty</td>
</tr>
<tr>
<td>Pagure merge request from branch</td>
<td>Pagure</td>
<td>SSH URL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>empty</td>
</tr>
</tbody>
</table>

You can also enable automatic pushing of changes after Weblate commits, this can be done in Lazy commits.

See for setting up SSH keys, and Lazy commits for info about when Weblate decides to commit changes.

Can be empty in case supports pushing.
Protected branches

If you are using Weblate on protected branch, you can configure it to use pull requests and perform actual review on the translations (what might be problematic for languages you do not know). An alternative approach is to waive this limitation for the Weblate push user.

For example on GitHub this can be done in the repository configuration:

Merge or rebase

By default, Weblate merges the upstream repository into its own. This is the safest way in case you also access the underlying repository by other means. In case you don't need this, you can enable rebasing of changes on upstream, which will produce a history with fewer merge commits.

 Rebasing can cause you trouble in case of complicated merges, so carefully consider whether or not you want to enable them.

Interacting with others

Weblate makes it easy to interact with others using its API.

Weblate REST API
Lazy commits

The behaviour of Weblate is to group commits from the same author into one commit if possible. This greatly reduces the number of commits, however you might need to explicitly tell it to do the commits in case you want to get the VCS repository in sync, e.g. for merge (this is by default allowed for the Managers group, see Managers).

The changes in this mode are committed once any of the following conditions are fulfilled:
- Somebody else changes an already changed string.
- A merge from upstream occurs.
- An explicit commit is requested.
- Change is older than period defined as on Component configuration.

Note: Commits are created for every component. So in case you have many components you will still see lot of commits. You might utilize Git addon in that case.

If you want to commit changes more frequently and without checking of age, you can schedule a regular task to perform a commit:

CELERY_BEAT_SCHEDULE = {
    # Unconditionally commit all changes every 2 minutes
    "commit": {
        "task": "weblate.trans.tasks.commit_pending",
        # Omitting hours will honor per component settings,
        # otherwise components with no changes older than this
        # won't be committed
        "kwargs": {"hours": 0},
        # How frequently to execute the job in seconds
        "schedule": 120,
    }
}

Processing repository with scripts

The way to customize how Weblate interacts with the repository is . Consult for info on how to execute external scripts through addons.

Keeping translations same across components

Once you have multiple translation components, you might want to ensure that the same strings have same translation. This can be achieved at several levels.

Translation propagation

With enabled (what is the default, see Component configuration), all new translations are automatically done in all components with matching strings. Such translations are properly credited to currently translating user in all components.

Note: The translation propagation requires the key to be match for monolingual translation formats, so keep that in mind when creating translation keys.
Consistency check

The check fires whenever the strings are different. You can utilize this to review such differences manually and choose the right translation.

Automatic translation based on different components can be way to synchronize the translations across components. You can either trigger it manually (see ) or make it run automatically on repository update using addon (see ).

Component configuration

URL Libre Component configuration

Libre
Everyone can add suggestions by default, to be accepted by signed in users. Suggestion voting can be used to make use of a string when more than one signed-in user agrees, by setting up the Component configuration with Suggestion voting to turn on voting, and Autoaccept suggestions to set a threshold for accepted suggestions (this includes a vote from the user making the suggestion if it is cast).

You can combine these with access control into one of the following setups:

Users suggest and vote for suggestions and a limited group controls what is accepted. - Turn on voting. - Turn off automatic acceptance. - Don’t let users save translations.

Users suggest and vote for suggestions with automatic acceptance once the defined number of them agree. - Turn on voting. - Set the desired number of votes for automatic acceptance.

Optional voting for suggestions. (Can optionally be used by users when they are unsure about a translation by making multiple suggestions.) - Only turn on voting.

---

Please pay attention to the licensing info, as this specifies how translations can be used.

By registering you agree to use your name and e-mail in the commits, and provide your contribution under the license defined by each localization project.

You have agreed to the following as a contributor:

- WebatceOrg/language names

Licenses for individual translations

GNU General Public License v3.0 or later

MIT License

WebatceOrg/Android
Additional info on source strings

Enhance the translation process by adding additional info to the strings including explanations, string priorities, check flags and visual context. Some of that info may be extracted from the translation files and some may be added by editing the additional string info:

Access this directly from the translation interface by clicking the "Edit" icon next to Screenshot context or Flags.
Automatic translation via machine translation uses active machine translation engines to get the best possible translations and applies them in this project.

Automatic překlad prostřednictvím strojového překladu používá aktuální strojový překlad pro získání nejlepších možných překladů a použije je na tento projekt.
Strings prioritization

2.0

String priority can be changed to offer higher priority strings for translation earlier by using the *priority* flag.

This can be used to order the flow of translation in a logical manner.

2.4

3.3: Previously called *Quality checks flags*, it no longer configures only checks.

The default set of translation flags is determined by the translation *Component configuration* and the translation file. However, you might want to use it to customize this per source string.

4.1: In previous versions this has been called *Extra context*.

Use the explanation to clarify scope or usage of the translation. You can use Markdown to include links and other markup.

Visual context for strings

2.9

You can upload a screenshot showing a given source string in use within your program. This helps translators understand where it is used, and how it should be translated.

The uploaded screenshot is shown in the translation context sidebar:
In addition to *Additional info on source strings*, screenshots have a separate management interface under the *Tools* menu. Upload screenshots, assign them to source strings manually, or use optical character recognition to do so. Once a screenshot is uploaded, this interface handles management and source string association:
Screenshot has been uploaded, you can now assign it to source strings.

### Assigned source strings

<table>
<thead>
<tr>
<th>English</th>
<th>Location</th>
<th>Assigned screenshots</th>
<th>Actions</th>
</tr>
</thead>
</table>

No matching strings found.

Screenshot is shown to add visual context for all listed source strings.

### Assign source strings

<table>
<thead>
<tr>
<th>English</th>
<th>Location</th>
<th>Assigned screenshots</th>
<th>Actions</th>
</tr>
</thead>
</table>

No matching strings found.

### Image

```
Source string

Hello, world:

One. Crocodile has 1% banana.

Other. Crocodile has 1% bananas.

Try Webate at <http://demo.weblate.org/>

Thank you for using Webate.

Screenshot is shown to add visual context for all listed source strings.
```

### Edit screenshot

**Screenshot name**

Automatic translation

**Image**

Currently `screenshots/screenshot.png`

Upload: JPEG or PNG images up to 2000x2000 pixels.

**Save**

### Screenshot details

**Created**

new

**Uploaded by**

Testuser

**Language**

English

### Delete screenshot

Deleting screenshot will remove it from all associated source strings.

**Delete**
from django.utils.translation import gettext_lazy as _

from weblate.trans.autofixes.base import AutoFix

class ReplaceFooWithBar(AutoFix):
    ""
    Replace foo with bar."
    ""
    name = _("Foobar")

    def fix_single_target(self, target, source, unit):
        if "foo" in target:
            return target.replace("foo", "bar"), True
        return target, False
Indicates forbidden translation in a glossary, see [variants](#).

Enable the [quality check].
Enable the AngularJS [quality check].
Enable the C [quality check].
Enable the C# [quality check].
Enable the ECMA Script [quality check].
Enable the `i18next` [quality check].
Enable the JavaScript [quality check].
Enable the Lua [quality check].
Enable the Object Pascal [quality check].
Enable the Perl [quality check].
Enable the PHP [quality check].
Enable the Python [quality check].
Enable the Qt [quality check].
Enable the Ruby [quality check].
Enable the Scheme [quality check].
Enable the Vue i18n [quality check].
Enable the HTML [quality check].
The string should consist of only a URL. Enable the URL [quality check].
Skip the BBCode [quality check].
Skip the [quality check].
Skip the AngularJS [quality check].
Skip the C [quality check].
Skip the C# [quality check].
Skip the ECMA Script [quality check].
Skip the `i18next` [quality check].
Skip the JavaScript [quality check].
Skip the Java MessageFormat [quality check].
Skip the JavaScript [quality check].
Skip the Lua [quality check].
Skip the Object Pascal [quality check].

234
Skip the quality check.
Skip the Perl quality check.
Skip the PHP quality check.
Skip the Python quality check.
Skip the Qt quality check.
Skip the Ruby quality check.
Skip the Scheme quality check.
Skip the Vue I18n quality check.
Skip the Kashida quality check.
Skip the Markdown quality check.
Skip the HTML quality check.
Skip the URL quality check.
Skip the XML quality check.
Component configuration: GNU gettext

3.11

Component configuration: Weblate. Translation states:

3.7

<table>
<thead>
<tr>
<th></th>
<th>Weblate</th>
<th>TrueTypeOpenType</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

236
### Font group

<table>
<thead>
<tr>
<th>Name</th>
<th>default-font</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default font</td>
<td>Source Sans 3 Bold</td>
</tr>
<tr>
<td>Japanese language override</td>
<td>Droid Sans Fallback Regular</td>
</tr>
<tr>
<td>Korean language override</td>
<td>Droid Sans Fallback Regular</td>
</tr>
</tbody>
</table>

**Add language override**

<table>
<thead>
<tr>
<th>Language</th>
<th>Font override language override</th>
<th>Font override Droid Sans Fallback Regular</th>
<th>Remove</th>
</tr>
</thead>
</table>

**Edit font group**

<table>
<thead>
<tr>
<th>Font group name</th>
<th>default-font</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier you will use in checks to select this font group. Avoid whitespaces and special characters.</td>
<td></td>
</tr>
<tr>
<td>Default Font</td>
<td>Source Sans 3 Bold</td>
</tr>
</tbody>
</table>

Default font is used unless per language override matches.
<table>
<thead>
<tr>
<th>Group name</th>
<th>Default font</th>
<th>Language overrides</th>
</tr>
</thead>
<tbody>
<tr>
<td>default-font</td>
<td>Source Sans 3 Bold</td>
<td>Japanese: Droid Sans Fallback Regular, Korean: Droid Sans Fallback Regular</td>
</tr>
</tbody>
</table>

Add font group

**Font group name**

Identifier you will use in checks to select this font group. Avoid whitespaces and special characters.

**Default font**

Default font is used unless per language override matches.

Save
1. weblate.checks.Check

2. check

3. check_single

Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>

This file is part of Weblate <https://weblate.org/>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see <https://www.gnu.org/licenses/>.
from django.utils.translation import gettext_lazy as _
from weblate.checks.base import TargetCheck

class FooCheck(TargetCheck):
    # Used as identifier for check, should be unique
    # Has to be shorter than 50 characters
    check_id = "foo"

    # Short name used to display failing check
    name = _("Foo check")

    # Description for failing check
    description = _("Your translation is foo")

    # Real check code
    def check_single(self, source, target, unit):
        return "foo" in target

# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
# This file is part of Weblate <https://weblate.org/>
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.
"""Quality check example for Czech plurals."""
from django.utils.translation import gettext_lazy as _
from weblate.checks.base import TargetCheck

class PluralCzechCheck(TargetCheck):
    # Used as identifier for check, should be unique
    # Has to be shorter than 50 characters
    check_id = "foo"

    # Short name used to display failing check
    name = _("Foo check")

    # Description for failing check
    description = _("Your translation is foo")

    # Real check code
def check_target_unit(self, sources, targets, unit):
    if self.is_language(unit, ("cs")):
        return targets[1] == targets[2]
    return False

def check_single(self, source, target, unit):
    
    return False

Project configuration

amaGama

weblate.machinery.tmserver.AmagamaTranslation

Installing amaGama

Apertium

weblate.machinery.apertium.ApertiumAPYTranslation

MT_APERTIUM_APY

AWS

Amazon Translate

1. Turn on this service by adding weblate.machinery.aws.AWSTranslation to MT_SERVICES.
2. boto3

Baidu API machine translation

Baidu

This service uses an API and you need to obtain an ID and API key from Baidu to use it.

Turn on this service by adding weblate.machinery.baidu.BaiduTranslation to MT_SERVICES and set MT_BAIDU_ID and MT_BAIDU_SECRET.

MT_BAIDU_ID MT_BAIDU_SECRET Baidu Translate API
DeepL

**2.20 API.**

DeepL is a paid service providing good machine translation for a few languages. You need to purchase DeepL API subscription or use legacy DeepL Pro (classic) plan.

Turn on this service by adding `weblate.machinery.deepl.DeepLTranslation` to `MT_SERVICES` and set `MT_DEEPL_KEY`.

---

**NOTES:** In case you have subscription for CAT tools, you are supposed to use "v1 API" instead of default "v2" used by Weblate (it is not really an API version in this case). In case you are on a free instead of a paid plan, you have to use `https://api-free.deepl.com/` instead of `https://api.deepl.com/` You can adjust both parameters by `MT_DEEPL_API_URL`.

---

**MT_DEEPL_KEY**, **MT_DEEPL_API_URL**

DeepL website DeepL pricing DeepL API documentation

LibreTranslate

**4.7.1 API.**

LibreTranslate is a free and open-source service for machine translations. The public instance requires an API key, but LibreTranslate can be self-hosted and there are several mirrors available to use the API for free.

Turn on this service by adding `weblate.machinery.libretranslate.LibreTranslateTranslation` to `MT_SERVICES` and set `MT_LIBRETRANSLATE_API_URL`. If your instance requires an API key, you must also set `MT_LIBRETRANSLATE_KEY`.

---

**MT_LIBRETRANSLATE_KEY**, **MT_LIBRETRANSLATE_API_URL**, LibreTranslate website, LibreTranslate repository, LibreTranslate mirrors

Glosbe

Free dictionary and translation memory for almost every living language.

The API is gratis to use, but subject to the used data source license. There is a limit of calls that may be done from one IP in a set period of time, to prevent abuse.

Turn on this service by adding `weblate.machinery.glosbe.GlosbeTranslation` to `MT_SERVICES`.

---

Glosbe website

Google Translate

This service uses the Google Translation API, and you need to obtain an API key and turn on billing in the Google API console.

To turn on this service, add `weblate.machinery.google.GoogleTranslation` to `MT_SERVICES` and set `MT_GOOGLE_KEY`.

---

**MT_GOOGLE_KEY**

Google translate documentation
Google Translate API V3 (Advanced)

Google Cloud

This service differs from the former one in how it authenticates. To enable service, add `weblate.machinery.googlev3.GoogleV3Translation` to `MT_SERVICES` and set `MT_GOOGLE_CREDENTIALS` and `MT_GOOGLE_PROJECT`.

If `location` fails, you may also need to specify `MT_GOOGLE_LOCATION`.

Microsoft Cognitive Services Translator

Cognitive Services - Azure portal

Weblate implements Translator API V3.

To enable this service, add `weblate.machinery.microsoft.MicrosoftCognitiveTranslation` to `MT_SERVICES` and set `MT_MICROSOFT_COGNITIVE_KEY`.

Translator Text API V2

The key you use with Translator API V2 can be used with API 3.

Translator Text API V3

Azure

`MT_MICROSOFT_REGION` and `MT_MICROSOFT_PROJECT` are set automatically.

Microsoft Terminology Service

The Microsoft Terminology Service API is available for Web.

The API is available in `weblate.machinery.microsoftterminology`.

ModernMT

Turn this service on by adding `weblate.machinery.modernmt.ModernMTTranslation` to `MT_SERVICES` and configure `MT_MODERNMT_KEY`.

ModernMT API, `MT_MODERNMT_KEY`, `MT_MODERNMT_URL`
MyMemory

Huge translation memory with machine translation.
Free, anonymous usage is currently limited to 100 requests/day, or to 1000 requests/day when you provide a contact e-mail address in `MT_MYMEMORY_EMAIL`. You can also ask them for more.

Turn on this service by adding `weblate.machinery.mymemory.MyMemoryTranslation` to `MT_SERVICES` and set `MT_MYMEMORY_EMAIL`.

`MT_MYMEMORY_EMAIL`  `MT_MYMEMORY_USER`  `MT_MYMEMORY_KEY`  MyMemory website

NetEase Sight API machine translation

3.3

NetEase

This service uses an API, and you need to obtain key and secret from NetEase.

Turn on this service by adding `weblate.machinery.youdao.NeteaseSightTranslation` to `MT_SERVICES` and set `MT_NETEASE_KEY` and `MT_NETEASE_SECRET`.

`MT_NETEASE_KEY`  `MT_NETEASE_SECRET`  NetEase Sight Translation Platform

tmserver

You can run your own translation memory server by using the one bundled with Translate-toolkit and let Weblate talk to it. You can also use it with an amaGama server, which is an enhanced version of tmserver.

1. First you will want to import some data to the translation memory:

2. Turn on this service by adding `weblate.machinery.tmserver.TMServerTranslation` to `MT_SERVICES`.

```
build_tmdb -d /var/lib/tm/db -s en -t cs locale/cs/LC_MESSAGES/django.po
build_tmdb -d /var/lib/tm/db -s en -t de locale/de/LC_MESSAGES/django.po
build_tmdb -d /var/lib/tm/db -s en -t fr locale/fr/LC_MESSAGES/django.po
```

3. Start tmserver to listen to your requests:

```
tmserver -d /var/lib/tm/db
```

4. Configure Weblate to talk to it:

```
MT_TMSERVER = "http://localhost:8888/tmserver/
```

tmserver  Installing amaGama  Amagama  Amagama Translation Memory

Yandex Translate

Yandex

This service uses a Translation API, and you need to obtain an API key from Yandex.

Turn on this service by adding `weblate.machinery.yandex.YandexTranslation` to `MT_SERVICES`, and set `MT_YANDEX_KEY`.

`MT_YANDEX_KEY`  Yandex Translate API  Powered by Yandex.Translate
Youdao Zhiyun API machine translation

This service uses an API, and you need to obtain an ID and an API key from Youdao.

Turn on this service by adding `weblate.machinery.youdao.YoudaoTranslation` to `MT_SERVICES` and set `MT_YOUDAO_ID` and `MT_YOUDAO_SECRET`.

Youdao Zhiyun Natural Language Translation Service

Webate

Webate can be the source of machine translations as well. It is based on the Woosh fulltext engine, and provides both exact and inexact matches.

Turn on these services by adding `weblate.machinery.weblatetm.WeblateTranslation` to `MT_SERVICES`.

Webate Translation Memory

The `memory` can be used as a source for machine translation suggestions as well.

Turn on these services by adding `weblate.memory.machine.WeblateMemory` to the `MT_SERVICES`. This service is turned on by default.

SAP Translation Hub

You need to have a SAP account (and the SAP Translation Hub enabled in the SAP Cloud Platform) to use this service.

Turn on this service by adding `weblate.machinery.saptranslationhub.SAPTranslationHub` to `MT_SERVICES`.

SAP Translation Hub API

Custom machine translation

You can also implement your own machine translation services using a few lines of Python code. This example implements machine translation in a fixed list of languages using the `dictionary` Python module.
(
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.
#
"""Machine translation example."""

)

import dictionary
from weblate.machinery.base import MachineTranslation
class SampleTranslation(MachineTranslation):
"""Sample machine translation interface."""
name = "Sample"
def download_languages(self):
"""Return list of languages your machine translation supports."""
return {"cs"}
def download_translations(
self,
source,
language,
text: str,
unit,
user,
search: bool,
threshold: int = 75,
):
"""Return tuple with translations."""
for t in dictionary.translate(text):
yield {"text": t, "quality": 100, "service": self.name, "source
,→": text}
You can list your own class in MT_SERVICES and Weblate will start using that.
����
2.19

.
↓

: You can also conﬁgure add-ons using API, DEFAULT_ADDONS, or install_addon.

246


### Installed add-ons

There are no add-ons currently installed.

### Available add-ons

- **Automatic translation**
  Automatically translates strings using machine translation or other components.
  - **Install**

- **Add missing languages**
  Ensures a consistent set of languages is used for all components within a project.
  - **Project wide**
  - **Install**

- **Component discovery**
  Automatically adds or removes project components based on file changes in the version control system.
  - **Repository wide**
  - **Install**

- **Bulk edit**
  Bulk edit flags, labels, or states of strings.
  - **Install**

- **Statistics generator**
  Generates a file containing detailed info about the translation status.
  - **Install**

- **Pseudolocale generation**
  Generates a translation by adding prefix and suffix to source strings automatically.
  - **Install**

- **Contributors in comment**
  Updates the comment part of the PO file header to include contributor names and years of contributions.
  - **Install**

- **Customize gettext output**
  Allows customization of gettext output behavior, for example line wrapping.
  - **Install**

- **Generate MO files**
  Automatically generates a MO file for every changed PO file.
  - **Install**

- **Update PO files to match POT (msgmerge)**
  Updates all PO files (as configured by "Filemask") to match the POT file (as configured by "Template for new translations") using msgmerge.
  - **Install**

- **Squash Git commits**
  Squash Git commits prior to pushing changes.
  - **Repository wide**
  - **Install**

- **Stale comment removal**
  Set a timeframe for removal of comments.
  - **Project wide**
  - **Install**

- **Stale suggestion removal**
  Set a timeframe for removal of suggestions.
  - **Project wide**
  - **Install**

Some add-ons will ask for additional configuration during installation.
weblate.autotranslate.autotranslate

mode -- Available choices:
suggest -- Add as suggestion
translate -- Add as translation
fuzzy -- Add as needing edit

fil-
ter_type -- Available choices:
all -- All strings
nottranslated -- Not translated strings
todo -- Strings needing action
fuzzy -- Strings marked for edit
check:inconsistent -- Failed check: Inconsistent

auto_ -- Available choices:
others -- Other translation components
mt -- Machine translation

component update, daily
Keeping translations same across components

JavaScript CDN

weblate.cdn.cdnjs

threshold  CSS

css_selector  CSS

cookie_name  Cookie

files  Cookie

JavaScript HTML

HTML JavaScript HTML

HTML JavaScript HTML

cdn-addon-config weblate-cdn cdn-addon-extract cdn-addon-html
Does Weblate update translation files besides translations?
Please review and confirm the matched components

Component | Matched files
---|---
Django | `webseite/locale/cs/LC_MESSAGES/django.po` (cs)
| `webseite/locale/it/LC_MESSAGES/django.po` (it)
| `webseite/locale/de/LC_MESSAGES/django.po` (de)

Django | `webseite/locale/it/LC_MESSAGES/django.po` (it)
| `webseite/locale/de/LC_MESSAGES/django.po` (de)

I confirm the above matches look correct.

Regular expression to match translation files against
```
webseite/locale/([^\/]*)/LC_MESSAGES/([^\/]*)\1.po
```

File format
- gettext PO file

Customize the component name
```
{component-title}
```

Define the monolingual base filename
Leave empty for bilingual translation files.

Define the base file for new translations
```
webseite/locale/[^/]*\1.po
```

Filename of file used for creating new translations. For gettext choose .po file.

Language filter
```
[^0-9a-z]
```

Regular expression to filter translation files against when scanning for filename.

Clone addons from the main component to the newly created ones

Remove components for nonexistent files

The regular expression to match translation files has to contain two named groups to match component and language, some examples:

<table>
<thead>
<tr>
<th>Regular expression</th>
<th>Example matched files</th>
<th>Description</th>
</tr>
</thead>
</table>
| `([^/]*\1)\2` | `cs/application.po
de/application.po
de/webseite.po` | One folder per language containing translation files for components. |
| `locale/[^/]*\1` | `locale/cs/LC_MESSAGES/application.po` | Usual structure for storing gettext PO files. |
| `src/[^/]*\1` | `src/locale/application.cs.po` | Using both component and language name within filename. |
| `locale/[^/]*\1` | `locale/cs/application.cs.po` | Using language in both path and filename. |
| `res/values/[^/]*\1` | `res/values/cs/strings.xml` | Android resource strings, split into several files. |

You can use Django template markup in both component name and the monolingual base filename, for example:
```
{{ component }}
```
Component filename match
```
{{ component-title }}
```
Component filename with uppercase first letter

Save
Component discovery add-on uses Weblate URL. It’s a convenient way to share VCS setup between multiple components. Linked components use the local repository of the main component set up by filling weblate:/project/main-component into the field (in Manage ↓ Settings ↓ Version control system) of each respective component. This saves time with configuration and system resources too.

Weblate flags

Available choices:
-1 -- Do not change
10 -- Needs editing
20 -- Translated
30 -- Approved

add_flags
re-move_flags
add_labels
re-move_labels

component update

NOT has:label

Table5 Label new strings automatically

NOT has:label

Table6 language:en AND key:changelogs/read-only

You might also want to tighten the check by adding strict-same flag to.

Translation states
weblate.flags.source_edit

unit post-create

VCS Weblate "

Translation states

weblate.flags.target_edit

unit post-create

VCS Weblate "

Translation states

weblate.generate.generate

repository pre-commit

Django:

locale/{{ language_code }}.json

```
{
    "language": "{{ language_code }}",
    "strings": "{{ stats.all }}",
    "translated": "{{ stats.translated }}",
    "last_changed": "{{ stats.last_changed }}",
    "last_author": "{{ stats.last_author }}"
}
```

weblate.generate.pseudolocale

component update, daily

253
Finding strings whose localized counterparts might not fit the layout is also possible.

You can use this add-on to start translation to a new locale of an existing language or similar language. Once you add the translation to the component, follow to the add-on. Example: If you have fr and want to start fr_CA translation, simply set fr as the source, fr_CA as the target, and leave the prefix and suffix blank. Uninstall the add-on once you have the new translation filled to prevent Weblate from changing the translations made after the copying.

```
weblate.gettext.authors
```

weblate.gettext.configure

```
configure configure ALL_LINGUAS
```

```
repository pre-commit
PO
```

The PO file header will look like this:

```
# Pavel Boreckí <pavel@example.com>, 2018, 2019.
# Filip Hron <filip@example.com>, 2018, 2019.
# anonymous <noreply@weblate.org>, 2019.
```

```
configure configure.in
```

```
repository post-add, daily
configure.in configure.ac ALL_LINGUAS
```

```
gettext 77 --no-wrap
```

Available choices:

```
77 -- Wrap lines at 77 characters and at newlines
65535 -- Only wrap lines at newlines
-1 -- No line wrapping
```

```
storage post-load
```

```
```

```
```
Does Weblate update translation files besides translations?

Available choices:
- all -- All commits into one
- language -- Per language
- file -- Per file
- author -- Per author
- append_trailers
- RFC 822   "Co-authored-by: …"
- commit_message
3.5 ¶

Per author

3.5 ¶

Co-authored-by: ...

Co-authored-by: ...

JSON

weblate.json.customize

```
sort_keys: JSON
indent: JSON
style: JSON
```

Available choices:

- spaces -- Spaces
- tabs -- Tabs

storage post-load

Java

weblate.properties.sort

repository pre-commit

Java

3.7 ¶

weblate.removal.comments

```
age: daily
```

daily

3.7 ¶

weblate.removal.suggestions

```
age: daily
votes:
```

daily

256
RESX

weblate.resx.update
repository post-update

---

---

Does Weblate update translation files besides translations?

YAML

weblate.yaml.customize

in-

dent |

width

Available choices:
80 -- Wrap lines at 80 chars
100 -- Wrap lines at 100 chars
120 -- Wrap lines at 120 chars
180 -- Wrap lines at 180 chars
65535 -- No line wrapping

line_break

Available choices:
dos -- DOS (\r\n)
unix -- UNIX (\n)
mac -- MAC (\r)

storage post-load

WEBLATE_ADDONS

---

---
```python
from django.utils.translation import gettext_lazy as _
from weblate.addons.events import EVENT_PRE_COMMIT
from weblate.addons.scripts import BaseScriptAddon

class ExamplePreAddon(BaseScriptAddon):
    # Event used to trigger the script
    events = (EVENT_PRE_COMMIT,)
    # Name of the addon, has to be unique
    name = "weblate.example.pre"
    # Verbose name and long description
    verbose = _("Execute script before commit")
    description = _("This addon executes a script.")

    # Script to execute
    script = "/bin/true"
    # File to add in commit (for pre commit event)
    does not have to be set
    add_file = "po/{{ language_code }}.po"
```

---

**VCS**: WL_VCS

**Upstream URL**: WL_REPO

**VCS**: WL_PATH

**Branch**: WL_BRANCH

**Filemask**: WL_FILEMASK

**Template**: WL_TEMPLATE

**New Base**: WL_NEW_BASE

---

258
Component configuration

VCS upstream Weblate VCS Gulp:

```bash
#!/bin/sh
gulp --gulpfile gulp-i18n-extract.js
git commit -m 'Update source strings' src/languages/en.lang.json
```

Weblate comes with a built-in translation memory consisting of the following:

Manually imported translation memory (see User interface).

Automatically stored translations performed in Weblate (depending on Translation memory scopes).

Automatically imported past translations.

Content in the translation memory can be applied one of two ways:

Manually, view while translating.
Automatically, by translating strings using [addon], or [addon] addon.
For installation tips, see Weblate Translation Memory, which is turned on by default.

**Translation memory scopes**

**3.2** In earlier versions translation memory could be only loaded from a file corresponding to the current imported translation memory scope.
The translation memory scopes are there to allow both privacy and sharing of translations, to suit the desired behavior.

**Imported translation memory**

Importing arbitrary translation memory data using the `import_memory` command makes memory content available to all users and projects.

**Per user translation memory**

Stores all user translations automatically in the personal translation memory of each respective user.

**Per project translation memory**

All translations within a project are automatically stored in a project translation memory only available for this project.

All translation within projects with shared translation memory turned on are stored in a shared translation memory available to all projects.

Please consider carefully whether to turn this feature on for shared Weblate installations, as it can have severe implications:
The translations can be used by anybody else.
This might lead to disclosing secret information.

**Managing translation memory**

**User interface**

**3.2**

In the basic user interface you can manage per user and per project translation memories. It can be used to download, wipe or import translation memory.

- **Note:** Translation memory in JSON can be imported into Weblate, TMX is provided for interoperability with other tools.

See Weblate [addon]
There are several management commands to manipulate the translation memory content. These operate on the translation memory as whole, unfiltered by scopes (unless requested by parameters):

- Exports the memory into JSON
- Imports TMX or JSON files into the translation memory

---

**settings.py**

`Django`s documentation`
AUDITLOG_EXPIRY
3.6
Weblate
180

AUTH_LOCK_ATTEMPTS
2.14

AUTO_UPDATE
3.2
3.11:

AVATAR_URL_PREFIX
URL: ${AVATAR_URL_PREFIX}/avatar/${MAIL_HASH}?${PARAMS}:
AVATAR_URL_PREFIX = 'https://www.gravatar.com/
AVATAR_URL_PREFIX = 'https://www.libravatar.org/

ENABLE_AVATARS: Avatars

262
AUTH_TOKEN_VALID

2.14

AUTH_PASSWORD_DAYS

2.15

AUTOFIX_LIST

autofixer Python

safe-html HTML

```python
AUTOFIX_LIST = (  
    "weblate.trans.autofixes.whitespace.SameBookendingWhitespace",  
    "weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis",  
)
```

BACKGROUND_TASKS

4.5.2

`monthly` `weekly` `daily` `never`

Weblate 263
BASE_DIR

Webate

DATA_DIR

Weblate

BASIC_LANGUAGES

4.4

List of languages to offer users for starting new translation. When not specified built-in list is used which includes all commonly used languages, but without country specific variants.

```
BASIC_LANGUAGES = ("cs", "it", "ja", "en")
```

CSP_SCRIPT_SRC

```
# Enable Cloudflare Javascript optimizations
CSP_SCRIPT_SRC = ["ajax.cloudflare.com"]
```

CHECK_LIST

```
CHECK_LIST = ()
```

```
CHECK_LIST = (  
    "weblate.checks.chars.BeginNewlineCheck",  
    "weblate.checks.chars.EndNewlineCheck",  
    "weblate.checks.chars.MaxLengthCheck",  
)
```
COMMENT_CLEANUP_DAYS

3.6

COMMIT_PENDING_HOURS

2.10

Component configuration

CONTACT_FORM

4.6

Configures how e-mail from the contact form is being sent. Choose a configuration that matches your mail server configuration.

The sender is used in as Reply-To, this is the default behaviour.

The sender is used in as From. Your mail server needs to allow sending such e-mails.

DATA_DIR

Webate

SSH

STATIC_ROOT

The Docker container uses a separate volume for this, see Docker container volumes.

MEDIA_ROOT

User-uploaded fonts, see.

Celery

Dumped data for backups

User-uploaded fonts, see.

Weblate uWSGI

User-uploaded fonts, see.

sudo chown www-data:www-data -R $DATA_DIR

BASE_DIR/data

sudo chown www-data:www-data $BASE_DIR/data

$DATA_DIR
DATABASE_BACKUP

3.1

Weblate provides the following options for backup:

- "plain"
- "compressed"
- "none"

DEFAULT_ACCESS_CONTROL

3.3

ACL configuration

DEFAULT_AUTO_WATCH

4.5

Configures whether `Automatically watch projects on contribution` should be turned on for new users. Defaults to `True`.

DEFAULT_RESTRICTED_COMPONENT

4.1

Component configuration

DEFAULT_ADD_MESSAGE | DEFAULT_ADDON_MESSAGE | DEFAULT_COMMIT_MESSAGE | DEFAULT_DELETE_MESSAGE | DEFAULT_MERGE_MESSAGE

Component configuration

Commit, add, delete, merge and addon messages
**DEFAULT_ADDONS**

```python
DEFAULT_ADDONS = {
    # Add-on with no parameters
    "weblate.flags.target_edit": {},
    # Add-on with parameters
    "weblate.autotranslate.autotranslate": {
        "mode": "suggest",
        "filter_type": "todo",
        "auto_source": "mt",
        "component": "",
        "engines": ["weblate-translation-memory"],
        "threshold": 80,
    },
}
```

**install_addon**

**DEFAULT_COMMITER_EMAIL**

```text
2.4.
noreply@weblate.org
```

**DEFAULT_COMMITER_NAME**

```text
2.4.
Weblate
```

**DEFAULT_COMMITER_EMAIL**

**DEFAULT_LANGUAGE**

```text
4.3.2.
en
```
DEFAULT_MERGE_STYLE

3.4

rebase - merge

Component configuration

DEFAULT_SHARED_TM

3.2

Configures default value of and .

DEFAULT_TRANSLATION_PROPAGATION

2.5

True

Component configuration

DEFAULT_PULL_MESSAGE

'Update from Weblate'

ENABLE_AVATARS

Gravatar -

AVATAR_URL_PREFIX

ENABLE_HOOKS

ENABLE_HTTPS

Weblate

HTTPS URL

HTTPS Django SSL

HTTP Forwarded-Proto

X-Forwarded-Proto

SESSION_COOKIE_SECURE

CSRF_COOKIE_SECURE

SECURE_SSL_REDIRECT

SECURE_PROXY_SSL_HEADER

268
ENABE_SHARING

GET_HELP_URL

4.5.2 URL
Weblate URL

GITLAB_CREDENTIALS

4.3
GitLab

GITLAB_CREDENTIALS = { 
  "gitlab.com": { 
    "username": "weblate", 
    "token": "your-api-token", 
  }, 
  "gitlab.example.com": { 
    "username": "weblate", 
    "token": "another-api-token", 
  }, 
}

GITLAB_USERNAME

GitLab

GITLAB_TOKEN

4.3
API GitLab

GITLAB_CREDENTIALS{GitLab}

GITHUB_CREDENTIALS

4.3
GitHub

GITHUB_CREDENTIALS = { 
  "api.github.com": { 
    "username": "weblate", 
    "token": "your-api-token", 
  }, 
}

269
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GitHub_USERNAME</td>
<td>GitHub</td>
</tr>
<tr>
<td>GitHub_CREDENTIALS</td>
<td>GitHub</td>
</tr>
<tr>
<td>GitHub_TOKEN</td>
<td>GitHub</td>
</tr>
<tr>
<td>Google_ANALYTICS_ID</td>
<td>Google</td>
</tr>
<tr>
<td>HIDE_REPO_CREDENTIALS</td>
<td>Web URL</td>
</tr>
<tr>
<td>HIDE_VERSION</td>
<td>4.3.1</td>
</tr>
<tr>
<td>IP_BEHIND_REVERSE_PROXY</td>
<td>2.14</td>
</tr>
</tbody>
</table>

270
IP_PROXY_HEADER

2.14 (EN)

HTTP_X_FORWARDED_FOR: a, b, client-ip

IP_PROXY_OFFSET

2.14 (EN)

SECURE_PROXY_SSL_HEADER, IP_BEHIND_REVERSE_PROXY, IP_PROXY_OFFSET

LEGAL_URL

3.5 (EN)

URL: https://weblate.org/terms/

LICENSE_EXTRA

[ "AGPL-3.0", "GNU Affero General Public License v3.0", "https://www.gnu.org/licenses/agpl-3.0-standalone.html", ]
LICENSE_FILTER

```python
LICENSE_FILTER = ("AGPL-3.0", "GPL-3.0-or-later")
LICENSE_FILTER = set()
```

alerts

LICENSE_REQUIRED

Component configuration

```python
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH = 10000
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH = 10
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH = True
```

LOCALIZE_CDN_URL = LOCALIZE_CDN_PATH = JavaScript CDN

```python
LOCALIZE_CDN_URL = LOCALIZE_CDN_PATH = "https://weblate-cdn.com/
```

JavaScript CDN
LOGIN_REQUIRED_URLS

```
LOGIN_REQUIRED_URLS = (r"/(.*)$",)
REST_FRAMEWORK["DEFAULT_PERMISSION_CLASSES"] = [
    "rest_framework.permissions.IsAuthenticated"
]
```

LOGIN_REQUIRED_URLS_EXCEPTIONS

```
LOGIN_REQUIRED_URLS_EXCEPTIONS = (  
    r"/accounts/(.*)$",  # Required for sign in  
    r"/static/(.*)$",  # Required for development mode  
    r"/widgets/(.*)$",  # Allowing public access to widgets  
    r"/data/(.*)$",  # Allowing public access to data exports  
    r"/hooks/(.*)$",  # Allowing public access to notification hooks  
    r"/api/(.*)$",  # Allowing access to API  
    r"/js/i18n/$",  # JavaScript localization
)
```

MATOMO_SITE_ID

```
MATOMO_SITE_ID = 1
MATOMO_URL = "https://example.matomo.cloud/
```

MATOMO_URL
**MT_SERVICES**

When using Docker container, this configuration is automatically generated based on provided API keys, see Machine translation settings.

```
MT_SERVICES = {
    "weblate.machinery.apertium.ApertiumAPYTranslation",
    "weblate.machinery.deepl.DeepLTranslation",
    "weblate.machinery.glosbe.GlosbeTranslation",
    "weblate.machinery.google.GoogleTranslation",
    "weblate.machinery.libretranslate.LibreTranslateTranslation",
    "weblate.machinery.microsoft.MicrosofCognitiveTranslation",
    "weblate.machinery.microsoftterminology.MicrosoftTerminologyService",
    "weblate.machinery.mymemory.MyMemoryTranslation",
    "weblate.machinery.tmserver.AmagamaTranslation",
    "weblate.machinery.tmserver.TMServerTranslation",
    "weblate.machinery.yandex.YandexTranslation",
    "weblate.machinery.weblatetm.WeblateTranslation",
    "weblate.memory.machine.WeblateMemory",
    "weblate.machinery.saptranslationhub.SAPTranslationHub",
}
```

---

**MT_APERTIUM_APY**

Apertium-APy URL: https://wiki.apertium.org/wiki/Apertium-apy

---

**MT_AWS_ACCESS_KEY_ID**

Amazon ID

---

**MT_AWS_SECRET_ACCESS_KEY**

AWS API key

---
MT_AWS_REGION

Amazon
AWS Region

MT_BAIDU_ID

Baidu Zhiyun API ID
https://api.fanyi.baidu.com/api/trans/product/index

Baidu API machine translation

MT_BAIDU_SECRET

Baidu Zhiyun API Secret Key
https://api.fanyi.baidu.com/api/trans/product/index

Baidu API machine translation

MT_DEEPL_API_URL

DeepL API URL v1 API v2 API

CAT Weblate DeepL v1 v2 API v2 v1 API

MT_DEEPL_API_VERSION

API DeepL API v1 API v2 API

CAT Weblate DeepL v1 v2 v1 API

MT_DEEPL_KEY

DeepL API Key
https://www.deepl.com/pro.html

DeepL API
**MT_LIBRETRANSLATE_API_URL**

4.7.1 API URL for the LibreTranslate instance to use.
Requires an API key to use outside of the website.
MIRRORS are documented on the LibreTranslate GitHub repository, some of which can be used without authentication:
https://github.com/LibreTranslate/LibreTranslate#user-content-mirrors
LibreTranslate.

**MT_LIBRETRANSLATE_KEY**

4.7.1 API key for the LibreTranslate instance specified in **MT_LIBRETRANSLATE_API_URL**.
LibreTranslate.

**MT_GOOGLE_KEY**

Google Translate API v2 API register at https://cloud.google.com/translate/docs
Google Translate.

**MT_GOOGLE_CREDENTIALS**

Google API v3 JSON OS https://cloud.google.com/docs/authentication/getting-started
Google Translate API V3 (Advanced)

**MT_GOOGLE_PROJECT**

Google API v3 ID https://cloud.google.com/appengine/docs/standard/nodejs/building-app/creating-project
Google Translate API V3 (Advanced)

**MT_GOOGLE_LOCATION**

API v3 Google global https://cloud.google.com/appengine/docs/locations
Google Translate API V3 (Advanced)

**MT_MICROSOFT_BASE_URL**

"Base URLs" Azure api.cognitive.microsofttranslator.com
Azure China api.tranlator.azure.cn
MT_MICROSOFT_COGNITIVE_KEY
Microsoft Cognitive Services Translator API

MT_MICROSOFT_REGION

MT_MICROSOFT_ENDPOINT_URL
Azure api.cognitive.microsoft.com
Azure China Azure

MT_MODERNMT_KEY
ModernMT API

MT_MODERNMT_URL
ModernMT URL https://api.modernmt.com/

MT_MYMEMORY_EMAIL
MyMemory API technical specifications

MT_MYMEMORY_KEY
MyMemory API key generator

MT_MYMEMORY_USER
MyMemory API key generator
**MT_NETEASE_KEY**

NetEase Sight API App key
https://sight.youdao.com/

**MT_NETEASE_SECRET**

NetEase Sight API App secret
https://sight.youdao.com/

**MT_TMSERVER**

tmserver URL

**MT_YANDEX_KEY**

Yandex Translate API App key
https://yandex.com/dev/translate/

**MT_YOUDAO_ID**

Youdao Zhiyun API ID
https://ai.youdao.com/product-fanyi-text.s

**MT_YOUDAO_SECRET**

Youdao Zhiyun API Client secret
https://ai.youdao.com/product-fanyi-text.s

**MT_SAP_BASE_URL**

SAP Translation Hub API URL
MT_SAP_SANDBOX_APIKEY

MT_SAP_USERNAME

MT_SAP_PASSWORD

MT_SAP_USE_MT

NEARBY_MESSAGES

DEFAULT_PAGE_LIMIT

PAGURE_CREDENTIALS

```
PAGURE_CREDENTIALS = {
    "pagure.io": {
        "username": "weblate",
        "token": "your-api-token",
    },
    "pagure.example.com": {
        "username": "weblate",
        "token": "another-api-token",
    },
}
```
PAGURE_USERNAME

4.3.2

Pagure

PAGURE_CREDENTIALS

Pagure

PAGURE_TOKEN

4.3.2

Pagure

Pagure API

PRIVACY_URL

4.8.1

URL where your Weblate instance shows its privacy policy.

PRIVACY_URL = "https://weblate.org/terms/"

LEGAL_URL

RATELIMIT_ATTEMPTS

3.2

RATELIMIT_WINDOW

RATELIMIT_LOCKOUT

RATELIMIT_WINDOW

3.2

RATELIMIT_ATTEMPTS

RATELIMIT_LOCKOUT
RATELIMIT_LOCKOUT

3.2

600 10

RATELIMIT_ATTEMPTS RATELIMIT_WINDOW

REGISTRATION_ALLOW_BACKENDS

4.1

REGISTRATION_OPEN Weblate

REGISTRATION_ALLOW_BACKENDS = ["azuread-oauth2", "azuread-tenant-oauth2"]

REGISTRATION_OPEN

REGISTRATION_CAPTCHA

CAPTCHA True False True

REGISTRATION_EMAIL_MATCH

REGISTRATION_EMAIL_MATCH = r"^.*@weblate.org$"

REGISTRATION_OPEN

LDAP

REGISTRATION_ALLOW_BACKENDS REGISTRATION_EMAIL_MATCH

281
REPOSITORY_ALERT_THRESHOLD

 alerts

REQUIRE_LOGIN

 LOGIN_REQUIRED_URLS

 REST API

 SESSION_COOKIE_AGE

 SESSION_COOKIE_AGE

SENTRY_DSN

 Sentry DSN

 SESSION_COOKIE_AGE

 SIMPLIFY_LANGUAGES

 Turn this off if you want to different translations for each variant.

 SITE_DOMAIN

 # Production site with domain name
 SITE_DOMAIN = "weblate.example.com"

 # Local development with IP address and port
 SITE_DOMAIN = "127.0.0.1:8000"

 HTTPS ENABLE_HTTPS URL_URL_PREFIX

 WEBLATE_ALLOWED_HOSTS

 Docker
HTTPS WEBSITE_DOMAIN: ENABLE_HTTPS

SITE_TITLE
Web

SPECIAL_CHARS

SPECIAL_CHARS = ("\t", "\n", "\u00a0", "...")

SINGLE_PROJECT

3.8 True
3.11

SINGLE_PROJECT = "test"

STATUS_URL

Weblate URL

SUGGESTION_CLEANUP_DAYS

3.2.1 None

UPDATE_LANGUAGES

4.3.2 setuplang

URL_PREFIX

Weblate WSGI WSGIScriptAlias

URL: /
URL_PREFIX = "/translations"

VCS_BACKENDS

VCS_BACKENDS = ("weblate.vcs.git.GitRepository",)

WEBLATE_ADDONS

WEBLATE_ADDONS = (  
  # Built-in addons  
  "weblate.addons.gettext.GenerateMoAddon",
  "weblate.addons.gettext.UpdateLinguasAddon",
  "weblate.addons.gettext.UpdateConfigureAddon",
  "weblate.addons.gettext.MsgmergeAddon",
  "weblate.addons.gettext.GettextCustomizeAddon",
  "weblate.addons.gettext.GettextAuthorComments",
  "weblate.addons.cleanup.CleanupAddon",
  "weblate.addons.consistency.LanguageConsistencyAddon",
  "weblate.addons.discovery.DiscoveryAddon",
  "weblate.addons.flags.SourceEditAddon",
  "weblate.addons.flags.TargetEditAddon",
  "weblate.addons.flags.SameEditAddon",
  "weblate.addons.flags.BulkEditAddon",
  "weblate.addons.generate.GenerateFileAddon",
  "weblate.addons.json.JSONCustomizeAddon",
  "weblate.addons.properties.PropertiesSortAddon",
  "weblate.addons.git.GitSquashAddon",
  "weblate.addons.removal.RemoveComments",
  "weblate.addons.removal.RemoveSuggestions",
)
# Add-on you want to include
"weblate.addons.example.ExampleAddon",
"

## DEFAULT_ADDONS

### WEBLATE_EXPORTERS

#### 4.2

### WEBLATE_FORMATS

#### 3.0

### WEBLATE_GPG_IDENTITY

#### 3.1

Webate Git GPG IDENTITY:

WEBLATE_GPG_IDENTITY = "Webate <weblate@example.com>"

Signing Git commits with GnuPG

### WEBSITE_REQUIRED

Defines whether Webate has to be specified when creating a project. Turned on by default as that suits public server setups.
Weblate

weblate/settings_example.py

# Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
# This file is part of Weblate <https://weblate.org/>
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.

import os
import platform
from logging.handlers import SysLogHandler

# Title of site to use
SITE_TITLE = "Weblate"

# Site domain
SITE_DOMAIN = ""

# Whether site uses https
ENABLE_HTTPS = False

# Django settings for Weblate project.

# DEBUG = True

ADMINS = [
    # ("Your Name", "your_email@example.com"),
]

MANAGERS = ADMINS

DATABASES = {
    "default": {
        # Use "postgresql" or "mysql".
        "ENGINE": "django.db.backends.postgresql",
        # Database name.
        "NAME": "weblate",
        # Database user.
        "USER": "weblate",
        # Name of role to alter to set parameters in PostgreSQL, in case role name is different than user used for authentication.
        "ALTER_ROLE": "weblate",
        # Database password.
        "PASSWORD": "",
        # Set to empty string for localhost.
        "HOST": "127.0.0.1",
        # Set to empty string for default.
        "PORT": "",
        # Customizations for databases.
    },
}
"OPTIONS": {
    # In case of using an older MySQL server,
    # which has MyISAM as a default storage
    # "init_command": "SET storage_engine=INNODB",
    # Uncomment for MySQL older than 5.7:
    # "init_command": "SET sql_mode='STRICT_TRANS_TABLES'",
    # Set emoji capable charset for MySQL:
    # "charset": "utf8mb4",
    # Change connection timeout in case you get MySQL gone away-
    "error":
        # "connect_timeout": 28800,
        # Persistent connections
        "CONN_MAX_AGE": 0,
    }

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__))))

# Data directory
DATA_DIR = os.path.join(BASE_DIR, "data")

# Local time zone for this installation. Choices can be found here:
# http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
# although not all choices may be available on all operating systems.
# In a Windows environment this must be set to your system time zone.
TIME_ZONE = "UTC"

# Language code for this installation. All choices can be found here:
# http://www.i18nguy.com/unicode/language-identifiers.html
LANGUAGE_CODE = "en-us"

LANGUAGES = {
    ("ar", "عربية"),
    ("az", "Azerbaijani"),
    ("be", "Belaruxskaja"),
    ("bg", "Bългарски"),
    ("br", "Brezhoneg"),
    ("ca", "Català"),
    ("cs", "Čeština"),
    ("da", "Dansk"),
    ("de", "Deutsch"),
    ("en", "English"),
    ("el", "Ελληνικά"),
    ("en-gb", "English (United Kingdom)"),
    ("es", "Español"),
    ("fi", "Suomi"),
    ("fr", "Français"),
    ("gl", "Galego"),
    ("he", "עברית"),
    ("hu", "Magyar"),
    ("hr", "Hrvatski"),
    ("id", "Indonesia"),
    ("is", "Íslenska"),
    ("it", "Italiano"),
    ("ja", "日本"),
    ("kab", "Taqbaylit"),
    ("kk", "Қазақ тілі"),
    ("ko", "한국"),
    ("nb", "Norsk bokmål"),
    ("nl", "Nederlands"),
    ("pl", "Polski"),
    ("pt", "Português"),
    ("pt-br", "Português brasileiro"),
    ("ro", "Română"),
    ("ru", "Русский"),
    ("zh", "简体中文"),
    ("zh-hant", "繁體中文"),
    ("zh-tw", "台湾中文"),
    ("zh-hk", "香港中文"),
    ("zh-mo", "澳门中文"),
    ("zh-mo", "澳门中文")}
SITE_ID = 1

# If you set this to False, Django will make some optimizations so as not
# to load the internationalization machinery.
USE_I18N = True

# If you set this to False, Django will not format dates, numbers and
# calendars according to the current locale.
USE_L10N = True

# If you set this to False, Django will not use timezone-aware datetimes.
USE_TZ = True

# Type of automatic primary key, introduced in Django 3.2
DEFAULT_AUTO_FIELD = "django.db.models.AutoField"

# URL prefix to use, please see documentation for more details
URL_PREFIX = ""

# Absolute filesystem path to the directory that will hold user-uploaded_
# files.
MEDIA_ROOT = os.path.join(DATA_DIR, "media")

# URL that handles the media served from MEDIA_ROOT. Make sure to use a
# trailing slash.
MEDIA_URL = f"{URL_PREFIX}/media/"

# Absolute path to the directory static files should be collected to.
# Don't put anything in this directory yourself; store your static files
# in apps' "static/" subdirectories and in STATICFILES_DIRS.
STATIC_ROOT = os.path.join(DATA_DIR, "static")

# URL prefix for static files.
STATIC_URL = f"{URL_PREFIX}/static/"

# Additional locations of static files
STATICFILES_DIRS = (
    # Put strings here, like "/home/html/static" or "C:/www/django/static".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative paths.
)

# List of finder classes that know how to find static files in
# various locations.
STATICFILES_FINDERS = (
    "django.contrib.staticfiles.finders.FileSystemFinder",
    "django.contrib.staticfiles.finders.AppDirectoriesFinder",
    "compressor.finders.CompressorFinder",
)

# Make this unique, and don't share it with anybody.
# You can generate it using weblate/examples/generate-secret-key
SECRET_KEY = ""

288
_TEMPLATE_LOADERS = [
    "django.template.loaders.filesystem.Loader",
    "django.template.loaders.app_directories.Loader",
]

if not DEBUG:
    _TEMPLATE_LOADERS = [("django.template.loaders.cached.Loader", _
                        TEMPLATE_LOADERS)]

TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "OPTIONS": {
            "context_processors": [
                "django.contrib.auth.context_processors.auth",
                "django.template.context_processors.debug",
                "django.template.context_processors.i18n",
                "django.template.context_processors.request",
                "django.contrib.messages.context_processors.messages",
                "weblate.trans.context_processors.weblate_context",
            ],
            "loaders": _TEMPLATE_LOADERS,
        },
    },
]

# GitHub username and token for sending pull requests.
# Please see the documentation for more details.
GITHUB_USERNAME = None
GITHUB_TOKEN = None

# GitLab username and token for sending merge requests.
# Please see the documentation for more details.
GITLAB_USERNAME = None
GITLAB_TOKEN = None

# Authentication configuration
AUTHENTICATION_BACKENDS = (
    "social_core.backends.email.EmailAuth",
    # "social_core.backends.google.GoogleOAuth2",
    # "social_core.backends.github.GithubOAuth2",
    # "social_core.backends.bitbucket.BitbucketOAuth",
    # "social_core.backends.suse.OpenSUSEOpenId",
    # "social_core.backends.ubuntu.UbuntuOpenId",
    # "social_core.backends.fedora.FedoraOpenId",
    # "social_core.backends.facebook.FacebookOAuth2",
    "weblate.accounts.auth.WeblateUserBackend",
)

# Custom user model
AUTH_USER_MODEL = "weblate_auth.User"

# Social auth backends setup
SOCIAL_AUTH_GITHUB_KEY = ""
SOCIAL_AUTH_GITHUB_SECRET = ""
SOCIAL_AUTH_GITHUB_SCOPE = ["user:email"]
SOCIAL_AUTH_BITBUCKET_KEY = ""
SOCIAL_AUTH_BITBUCKET_SECRET = ""
SOCIAL_AUTH_BITBUCKET_VERIFIED_EMAILS_ONLY = True
SOCIAL_AUTH_FACEBOOK_KEY = ""
SOCIAL_AUTH_FACEBOOK_SECRET = ""
SOCIAL_AUTH_FACEBOOK_SCOPE = ["email", "public_profile"]
SOCIAL_AUTH_FACEBOOK_PROFILE_EXTRA_PARAMS = {"fields": "id,name,email"}
SOCIAL_AUTH_GOOGLE_OAUTH2_KEY = ""
SOCIAL_AUTH_GOOGLE_OAUTH2_SECRET = ""

# Social auth settings
SOCIAL_AUTH_PIPELINE = (
    "social_core.pipeline.social_auth.social_details",
    "social_core.pipeline.social_auth.social_uid",
    "social_core.pipeline.social_auth.auth_allowed",
    "social_core.pipeline.social_auth.social_user",
    "weblate.accounts.pipeline.store_params",
    "weblate.accounts.pipeline.verify_open",
    "social_core.pipeline.user.get_username",
    "weblate.accounts.pipeline.require_email",
    "social_core.pipeline.mail.mail_validation",
    "weblate.accounts.pipeline.revoke_mail_code",
    "weblate.accounts.pipeline.ensure_valid",
    "weblate.accounts.pipeline.remove_account",
    "social_core.pipeline.social_auth.associate_by_email",
    "weblate.accounts.pipeline.reauthenticate",
    "social_core.pipeline.user.create_user",
    "social_core.pipeline.social_auth.associate_user",
    "social_core.pipeline.load_extra_data",
    "weblate.accounts.pipeline.store_email",
    "weblate.accounts.pipeline.notify_connect",
    "weblate.accounts.pipeline.cleanup_next",
    "weblate.accounts.pipeline.user_full_name",
    "weblate.accounts.pipeline.password_reset",
)

SOCIAL_AUTH_DISCONNECT_PIPELINE = (
    "social_core.pipeline.disconnect.allowed_to_disconnect",
    "social_core.pipeline.disconnect.get_entries",
    "social_core.pipeline.disconnect.revoke_tokens",
    "weblate.accounts.pipeline.cycle_session",
    "weblate.accounts.pipeline.adjust_primary_mail",
    "social_core.pipeline.disconnect.disconnect",
    "weblate.accounts.pipeline.cleanup_next",
)

# Custom authentication strategy
SOCIAL_AUTH_STRATEGY = "weblate.accounts.strategy.WeblateStrategy"

# Raise exceptions so that we can handle them later
SOCIAL_AUTH_RAISE_EXCEPTIONS = True

SOCIAL_AUTH_EMAIL_VALIDATION_FUNCTION = "weblate.accounts.pipeline.send_email_validation"
SOCIAL_AUTH_EMAIL_VALIDATION_URL = f"{URL_PREFIX}/accounts/email-sent/"
SOCIAL_AUTH_LOGIN_ERROR_URL = f"{URL_PREFIX}/accounts/login/"
SOCIAL_AUTH_EMAIL_FORM_URL = f"{URL_PREFIX}/accounts/email/"
SOCIAL_AUTH_NEW_ASSOCIATION_REDIRECT_URL = f"{URL_PREFIX}/accounts/profile/"

SOCIAL_AUTH_PROTECTED_USER_FIELDS = ("email",)
SOCIAL_AUTH_SLUGIFY_USERNAMES = True
SOCIAL_AUTH_SLUGIFY_FUNCTION = "weblate.accounts.pipeline.slugify_username"

# Password validation configuration
AUTH_PASSWORD_VALIDATORS = [
    {"NAME": "django.contrib.auth.password_validation.
        UserAttributeSimilarityValidator" # noqa: E501, pylint: disable=line-too-long
    },
    {"NAME": "django.contrib.auth.password_validation.
        MinimumLengthValidator",
        "OPTIONS": {"min_length": 10},
]
"NAME": "django.contrib.auth.password_validation.CommonPasswordValidator"},
{"NAME": "django.contrib.auth.password_validation.NumericPasswordValidator"},
{"NAME": "weblate.accounts.password_validation_CHARSPasswordValidator"},
{"NAME": "weblate.accounts.password_validationPastPasswordsValidator"}

# Optional password strength validation by django-zxcvbn-password
# {
# "NAME": "zxcvbn_password.ZXCVBNValidator",
# "OPTIONS": {
# "min_score": 3,
# "user_attributes": ["username", "email", "full_name"]
# }
# }

# Password hashing (prefer Argon)
PASSWORD_HASHERS = [
  "django.contrib.auth.hashers.Argon2PasswordHasher",
  "django.contrib.auth.hashers.PBKDF2PasswordHasher",
  "django.contrib.auth.hashers.PBKDF2SHA1PasswordHasher",
  "django.contrib.auth.hashers.BCryptSHA256PasswordHasher",
]

# Allow new user registrations
REGISTRATION_OPEN = True

# Shortcut for login required setting
REQUIRE_LOGIN = False

# Middleware
MIDDLEWARE = [
  "weblate.middleware.RedirectMiddleware",
  "weblate.middleware.ProxyMiddleware",
  "django.middleware.security.SecurityMiddleware",
  "django.contrib.sessions.middleware.SessionMiddleware",
  "django.middleware.csrf.CsrfViewMiddleware",
  "weblate.accounts.middleware.AuthenticationMiddleware",
  "django.middleware.clickjacking.XFrameOptionsMiddleware",
  "social_django.middleware.SocialAuthExceptionMiddleware",
  "weblate.accounts.middleware.RequireLoginMiddleware",
  "weblate.api.middleware.ThrottlingMiddleware",
  "weblate.middleware.SecurityMiddleware",
  "weblate.wladmin.middleware.ManageMiddleware",
]

ROOT_URLCONF = "weblate.urls"

# Django and Weblate apps
INSTALLED_APPS = [
  # Weblate apps on top to override Django locales and templates
  "weblate.addons",
  "weblate.auth",
  "weblate.checks",
  "weblate.formats",
  "weblate.glossary",
  "weblate.machinery",
  "weblate.trans",
  "weblate.lang",
  "weblate_language_data",
  "weblate.memory",
  "weblate.screenshots",
  "weblate.fonts",
]
"weblate.accounts",
"weblate.configuration",
"weblate.utils",
"weblate.vcs",
"weblate.wladmin",
"weblate.metrics",
"weblate",
# Optional: Git exporter
"weblate.gitexport",
# Standard Django modules
"django.contrib.auth",
"django.contrib.contenttypes",
"django.contrib.sessions",
"django.contrib.messages",
"django.contrib.staticfiles",
"django.contrib.admin.apps.SimpleAdminConfig",
"django.contrib.admindocs",
"django.contrib.sitemaps",
"django.contrib.humanize",
# Third party Django modules
"social_django",
"crispy_forms",
"compressor",
"rest_framework",
"rest_framework.authtoken",
"django_filters",
]

# Custom exception reporter to include some details
DEFAULT_EXCEPTION_REPORTER_FILTER = "weblate.trans.debug.

# Default logging of Weblate messages
# - to syslog in production (if available)
# - otherwise to console
# you can also choose "logfile" to log into separate file
# after configuring it below

# Detect if we can connect to syslog
HAVE_SYSLOG = False
if platform.system() != "Windows":
    try:
        handler = SysLogHandler(address="/dev/log", facility=SysLogHandler.

LOG_LOCAL2)
    handler.close()
    HAVE_SYSLOG = True
except OSError:
    HAVE_SYSLOG = False

if DEBUG or not HAVE_SYSLOG:
    DEFAULT_LOG = "console"
else:
    DEFAULT_LOG = "syslog"

DEFAULT_LOGLEVEL = "DEBUG" if DEBUG else "INFO"

# A sample logging configuration. The only tangible logging
# performed by this configuration is to send an email to
# the site admins on every HTTP 500 error when DEBUG=False.
# See http://docs.djangoproject.com/en/stable/topics/logging for
# more details on how to customize your logging configuration.
LOGGING = {
    "version": 1,
    "disable_existing_loggers": True,
    "filters": {
        "require_debug_false": {
            (): "django.utils.log.

RequireDebugFalse"}
    },
    "formatters": {
        "syslog": {
            "format": "weblate[%(process)d]: %(levelname)s

%(message)s"]
        },
        ""root": {
            "level": "DEBUG",
            "handlers": ["console", "syslog"],
            "filters": [],
            "styles": []
        }
    },
    "handlers": {
        "console": {
            "class": "logging.StreamHandler",
            "formatter": "syslog",
            "level": "DEBUG"
        },
        "syslog": {
            "class": "logging.handlers.SysLogHandler",
            "address": "/dev/log",
            "facility": SysLogHandler.

LOG_LOCAL2",
            "formatter": "syslog"
        }
    },
    "loggers": {
        "weblate": {
            "level": "DEBUG",
            "handlers": ["console", "syslog"],
            "propagate": False
        }
    }
}
"simple": { "format": "%(asctime)s %(levelname)s/%(process)s $message$s" },
"logfile": { "format": "%(asctime)s %(levelname)s $message$s" },
"django.server": { "()": "django.utils.log.ServerFormatter", "format": "[$(server_time)s] $message$s" },

"mail_admins": { "level": "ERROR", "filters": ["require_debug_false"], "class": "django.utils.log.AdminEmailHandler", "include_html": True, },
"console": { "level": "DEBUG", "class": "logging.StreamHandler", "formatter": "simple" },
"django.server": { "level": "INFO", "class": "logging.StreamHandler", "formatter": "django.server" },
"syslog": { "level": "DEBUG", "class": "logging.handlers.SysLogHandler", "formatter": "syslog", "address": "/dev/log", "facility": SysLogHandler.LOG_LOCAL2, },

# Logging to a file
# "logfile": { "level": "DEBUG", "class": "logging.handlers.RotatingFileHandler", "filename": "/var/log/weblate/weblate.log", "maxBytes": 100000, "backupCount": 3, "formatter": "logfile", },

"loggers": {
"django.request": { "handlers": ["mail_admins", DEFAULT_LOG], "level": "ERROR", "propagate": True, },
"django.server": { "handlers": ["django.server"], "level": "INFO", "propagate": False, },

# Logging database queries
# "django.db.backends": { "handlers": [DEFAULT_LOG], "level": "DEBUG", },
"weblate": { "handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL, "Logging VCS operations" },
"weblate.vcs": { "handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL, },

# Python Social Auth
"social": { "handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL, "Django Authentication Using LDAP" },
"django_auth_ldap": { "handlers": [DEFAULT_LOG], "level": DEFAULT_LOGLEVEL, },
SAML IdP

djangosaml2idp:
"handlers": [DEFAULT_LOG], "level": DEFAULT_LOG,
LOGLEVEL},
}

# Remove syslog setup if it's not present
if not HAVE_SYSLOG:
    del LOGGING["handlers"]["syslog"]

# List of machine translations
MT_SERVICES = [
    # "weblate.machinery.apertium.ApertiumAPYTranslation",
    # "weblate.machinery.baidu.BaiduTranslation",
    # "weblate.machinery.deepL.DeepLTranslation",
    # "weblate.machinery.glosbe.GlosbeTranslation",
    # "weblate.machinery.google.GoogleTranslation",
    # "weblate.machinery.googlev3.GoogleV3Translation",
    # "weblate.machinery.libretranslate.LibreTranslateTranslation",
    # "weblate.machinery.microsoft.MicrosoftCognitiveTranslation",
    # "weblate.machinery.microsoftterminology.MicrosoftTerminologyService",
    # "weblate.machinery.modernmt.ModernMTTranslation",
    # "weblate.machinery.mymemory.MyMemoryTranslation",
    # "weblate.machinery.netease.NeteaseSightTranslation",
    # "weblate.machinery.tmserver.AmagamaTranslation",
    # "weblate.machinery.tmserver.TMServerTranslation",
    # "weblate.machinery.yandex.YandexTranslation",
    # "weblate.machinery.saptranslationhub.SAPTranslationHub",
    "weblate.machinery.weblatetm.WeblateTranslation",
    "weblate.memory.machine.WeblateMemory",
]

# Machine translation API keys

# URL of the Apertium APy server
MT_APERTIUM_APY = None

# DeepL API key
MT_DEEPL_KEY = None

# LibreTranslate
MT_LIBRETRANSLATE_API_URL = None
MT_LIBRETRANSLATE_KEY = None

# Microsoft Cognitive Services Translator API, register at
# https://portal.azure.com/
MT_MICROSOFT_COGNITIVE_KEY = None
MT_MICROSOFT_REGION = None

# ModernMT
MT_MODERNMT_KEY = None

# MyMemory identification email, see
# https://mymemory.translated.net/doc/spec.php
MT_MYMEMORY_EMAIL = None

# Optional MyMemory credentials to access private translation memory
MT_MYMEMORY_USER = None
MT_MYMEMORY_KEY = None

# Google API key for Google Translate API v2
MT_GOOGLE_KEY = None

# Google Translate API3 credentials and project id
MT_GOOGLE_CREDENTIALS = None
MT_GOOGLE_PROJECT = None

# Baidu app key and secret
MT_BAIDU_ID = None
MT_BAIDU_SECRET = None

# Youdao Zhiyun app key and secret
MT_YOUDAO_ID = None
MT_YOUDAO_SECRET = None

# Netease Sight (Jianwai) app key and secret
MT_NETEASE_KEY = None
MT_NETEASE_SECRET = None

# API key for Yandex Translate API
MT_YANDEX_KEY = None

# tmserver URL
MT_TMSERVER = None

# SAP Translation Hub
MT_SAP_BASE_URL = None
MT_SAP_SANDBOX_APIKEY = None
MT_SAP_USERNAME = None
MT_SAP_PASSWORD = None
MT_SAP_USE_MT = True

# Use HTTPS when creating redirect URLs for social authentication, see
documentation for more details:
# https://python-social-auth-docs.readthedocs.io/en/latest/configuration/
# SOCIAL_AUTH_REDIRECT_IS_HTTPS = ENABLE_HTTPS

SOCIAL_AUTH_REDIRECT_IS_HTTPS = ENABLE_HTTPS

# SAP Translation Hub
MT_SAP_BASE_URL = None
MT_SAP_SANDBOX_APIKEY = None
MT_SAP_USERNAME = None
MT_SAP_PASSWORD = None
MT_SAP_USE_MT = True

# Use HTTPS when creating redirect URLs for social authentication, see
documentation for more details:
# https://python-social-auth-docs.readthedocs.io/en/latest/configuration/
# SOCIAL_AUTH_REDIRECT_IS_HTTPS = ENABLE_HTTPS

SOCIAL_AUTH_REDIRECT_IS_HTTPS = ENABLE_HTTPS

# Make CSRF cookie HttpOnly, see documentation for more details:
# https://docs.djangoproject.com/en/1.11/ref/settings/#csrf-cookie-httponly
CSRF_COOKIE_HTTPONLY = True
CSRF_COOKIE_SECURE = ENABLE_HTTPS

# Store CSRF token in session
CSRF_USE_SESSIONS = True

# Customize CSRF failure view
CSRF_FAILURE_VIEW = "weblate.trans.views.error.csrf_failure"
SESSION_COOKIE_SECURE = ENABLE_HTTPS
SESSION_COOKIE_HTTPONLY = True

# SSL redirect
SECURE_SSL_REDIRECT = ENABLE_HTTPS

# Sent referrer only for same origin links
SECURE_REFERRER_POLICY = "same-origin"

# SSL redirect URL exemption list
SECURE_REDIRECT_EXEMPT = (r"healthz/$",) # Allowing HTTP access to health_

# cookie age (in seconds)
SESSION_COOKIE_AGE = 1000
SESSION_COOKIE_AGE_AUTHENTICATED = 1209600
SESSION_COOKIE_SAMESITE = "Lax"

# Increase allowed upload size
DATA_UPLOAD_MAX_MEMORY_SIZE = 5000000

# Apply session cookie settings to language cookie as ewll
LANGUAGE_COOKIE_SECURE = SESSION_COOKIE_SECURE
LANGUAGE_COOKIE_HTTPONLY = SESSION_COOKIE_HTTPONLY
LANGUAGE_COOKIE_AGE = SESSION_COOKIE_AGE_AUTHENTICATED * 10
LANGUAGE_COOKIE_SAMESITE = SESSION_COOKIE_SAMESITE

# Some security headers
SECURE_BROWSER_XSS_FILTER = True
X_FRAME_OPTIONS = "DENY"
SECURE_CONTENT_TYPE_NOSNIFF = True
# Optionally enable HSTS
SECURE_HSTS_SECONDS = 31536000 if ENABLE_HTTPS else 0
SECURE_HSTS_PRELOAD = ENABLE_HTTPS
SECURE_HSTS_INCLUDE_SUBDOMAINS = ENABLE_HTTPS

# HTTPS detection behind reverse proxy
SECURE_PROXY_SSL_HEADER = None

# URL of login
LOGIN_URL = f"{URL_PREFIX}/accounts/login/

# URL of logout
LOGOUT_URL = f"{URL_PREFIX}/accounts/logout/

# Default location for login
LOGIN_REDIRECT_URL = f"{URL_PREFIX}/

# Anonymous user name
ANONYMOUS_USER_NAME = "anonymous"

# Reverse proxy settings
IP_PROXY_HEADER = "HTTP_X_FORWARDED_FOR"
IP_BEHIND_REVERSE_PROXY = False
IP_PROXY_OFFSET = 0

# Sending HTML in mails
EMAIL_SEND_HTML = True

# Subject of emails includes site title
EMAIL_SUBJECT_PREFIX = f"[{SITE_TITLE}]"

# Enable remote hooks
ENABLE_HOOKS = True

# By default the length of a given translation is limited to the length of
# the source string * 10 characters. Set this option to False to allow...

# longer translations (up to 10.000 characters)
LIMIT_TRANSLATION_LENGTH_BY_SOURCE_LENGTH = True

# Use simple language codes for default language/country combinations
SIMPLIFY_LANGUAGES = True

# Render forms using bootstrap
CRISPY_TEMPLATE_PACK = "bootstrap3"

# List of quality checks
# CHECK_LIST = (∋
# "weblate.checks.chars.BeginNewlineCheck",
# "weblate.checks.chars.BeginSpaceCheck",
# "weblate.checks.chars.EndNewlineCheck",
# "weblate.checks.chars.EndSpaceCheck",
# "weblate.checks.chars.DoubleSpaceCheck",
# "weblate.checks.chars.EndStopCheck",
# "weblate.checks.chars.EndColonCheck",
# "weblate.checks.chars.EndQuestionCheck",
# "weblate.checks.chars.EndExclamationCheck",
# "weblate.checks.chars.EndEllipsisCheck",
# "weblate.checks.chars.EndSemicolonCheck",
# "weblate.checks.chars.MaxLengthCheck",
# "weblate.checks.chars.KashidaCheck",
# "weblate.checks.chars.PunctuationSpacingCheck",
# "weblate.checks.format.PythonFormatCheck",
# "weblate.checks.format.PHPFormatCheck",
# "weblate.checks.format.PythonBraceFormatCheck",
# "weblate.checks.format.PythonQuotedFormatCheck",
# "weblate.checks.format.PythonUnquotedFormatCheck",..."
# List of automatic fixups
# AUTOFIX_LIST = (
    "weblate.trans.autofixes.whitespace.SameBookendingWhitespace",
    "weblate.trans.autofixes.chars.ReplaceTrailingDotsWithEllipsis",
    "weblate.trans.autofixes.chars.RemoveZeroSpace",
    "weblate.trans.autofixes.chars.RemoveControlChars",
)
# E-mail address that error messages come from.
SERVER_EMAIL = "noreply@example.com"

# Default email address to use for various automated correspondence from # the site managers. Used for registration emails.
DEFAULT_FROM_EMAIL = "noreply@example.com"

# List of URLs your site is supposed to serve
ALLOWED_HOSTS = ["*"]

# Configuration for caching
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",
        # If redis is running on same host as Weblate, you might
        # want to use unix sockets instead:
        # "LOCATION": "unix:///var/run/redis/redis.sock?db=1",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
            "PARSER_CLASS": "redis.connection.HiredisParser",
            # If you set password here, adjust CELERY_BROKER_URL as well
            "PASSWORD": None,
            "CONNECTION_POOL_KWARGS": {},
        },
        "KEY_PREFIX": "weblate",
    },
    "avatar": {
        "BACKEND": "django.core.cache.backends.filebased.FileBasedCache",
        "LOCATION": os.path.join(DATA_DIR, "avatar-cache"),
        "TIMEOUT": 86400,
        "OPTIONS": {"MAX_ENTRIES": 1000},
    },
}

# Store sessions in cache
SESSION_ENGINE = "django.contrib.sessions.backends.cache"

# Store messages in session
MESSAGE_STORAGE = "django.contrib.messages.storage.session.SessionStorage"

# REST framework settings for API
REST_FRAMEWORK = {
    # Use Django's standard `django.contrib.auth` permissions,
    # or allow read-only access for unauthenticated users.
    "DEFAULT_PERMISSION_CLASSES": [
        # Require authentication for login required sites
        rest_framework.permissions.IsAuthenticated
        if REQUIRE_LOGIN
        else "rest_framework.permissions.IsAuthenticatedOrReadOnly"
    ],
    "DEFAULT_AUTHENTICATION_CLASSES": [
        "rest_framework.authentication.TokenAuthentication",
        "weblate.api.authentication.BearerAuthentication",
    ],
}
"rest_framework.authentication.SessionAuthentication",
"DEFAULT_THROTTLE_CLASSES": {
    "weblate.api.throttling.UserRateThrottle",
    "weblate.api.throttling.AnonRateThrottle",
},
"DEFAULT_THROTTLE_RATES": {"anon": "100/day", "user": "5000/hour"},
"DEFAULT_PAGINATION_CLASS": "rest_framework.pagination.PageNumberPagination",
"PAGE_SIZE": 20,
"VIEW_DESCRIPTION_FUNCTION": "weblate.api.views.get_view_description",
"UNAUTHENTICATED_USER": "weblate.auth.models.get_anonymous",
}

# Fonts CDN URL
FONTS_CDN_URL = None

# Django compressor offline mode
COMPRESS_OFFLINE = False
COMPRESS_OFFLINE_CONTEXT = [
    {"fonts_cdn_url": FONTS_CDN_URL, "STATIC_URL": STATIC_URL, "LANGUAGE_BIDI": True},
    {"fonts_cdn_url": FONTS_CDN_URL, "STATIC_URL": STATIC_URL, "LANGUAGE_BIDI": False},
]

# Require login for all URLs
if REQUIRE_LOGIN:
    LOGIN_REQUIRED_URLS = (r"/(.*)"),

# In such case you will want to include some of the exceptions
# LOGIN_REQUIRED_URLS_EXCEPTIONS = (  
#     rf"/(.*)$", # Required for login  
#     rf"/(.*)$", # Required for admin login  
#     rf"/(.*)$", # Required for development mode  
#     rf"/(.*)$", # Allowing public access to widgets  
#     rf"/(.*)$", # Allowing public access to data exports  
#     rf"/(.*)$", # Allowing public access to...
# )

# Silence some of the Django system checks
SILENCED_SYSTEM_CHECKS = [
    # We have modified django.contrib.auth.middleware.
    "AuthenticationMiddleware",
    "as weblate.accounts.middleware.AuthenticationMiddleware"
]

# Celery worker configuration for testing
# CELERY_TASK_ALWAYS_EAGER = True
# CELERY_BROKER_URL = "memory://"
# CELERY_TASK_EAGER_PROPAGATES = True
# Celery worker configuration for production
CELERY_TASK_ALWAYS_EAGER = False
CELERY_BROKER_URL = "redis://localhost:6379"
CELERY_RESULT_BACKEND = CELERY_BROKER_URL

# Celery settings, it is not recommended to change these
CELERY_WORKER_MAX_MEMORY_PER_CHILD = 200000
CELERY_BEAT_SCHEDULE_FILENAME = os.path.join(DATA_DIR, "celery", "beat-"
CELERY_TASK_ROUTES = {
    "weblate.trans.tasks.auto_translate*": {"queue": "translate"},
    "weblate.accounts.tasks.notify_*": {"queue": "notify"},
    "weblate.accounts.tasks.send_mails": {"queue": "notify"},
    "weblate.utils.tasks.settings_backup": {"queue": "backup"},
    "weblate.utils.tasks.database_backup": {"queue": "backup"},
    "weblate.wladmin.tasks.backup_service": {"queue": "backup"},
    "weblate.memory.tasks.*": {"queue": "memory"},
}

# Enable plain database backups
DATABASE_BACKUP = "plain"

# Enable auto updating
AUTO_UPDATE = False

# PGP commits signing
WEBLATE_GPG_IDENTITY = None

# Third party services integration
MATOMO_SITE_ID = None
MATOMO_URL = None
GOOGLE_ANALYTICS_ID = None
SENTRY_DSN = None
SENTRY_ENVIRONMENT = SITE_DOMAIN
AKISMET_API_KEY = None

You will find basic management commands (available as ./manage.py in the Django sources, or as an extended set in a script called weblate installable atop Weblate).

Invoking management commands

As mentioned before, invocation depends on how you installed Weblate.
If using virtualenv for Weblate, you can either specify the full path to weblate, or activate the virtualenv prior to invoking it:

# Direct invocation
~/.weblate-env/bin/weblate

# Activating virtualenv adds it to search path
. ~/.weblate-env/bin/activate
weblate

If you are using source code directly (either from a tarball or Git checkout), the management script is ./manage.py available in the Weblate sources. To run it:

python ./manage.py list_versions

If you’ve installed Weblate using the pip or pip3 installer, or by using the ./setup.py script, the weblate is installed to your path (or virtualenv path), from where you can use it to control Weblate:

weblate list_versions

For the Docker image, the script is installed like above, and you can run it using docker exec:

docker exec --user weblate <container> weblate list_versions

For docker-compose the process is similar, you just have to use docker-compose exec:
```bash
docker-compose exec --user weblate weblate weblate list_versions
```

In case you need to pass it a file, you can temporary add a volume:

```bash
docker-compose exec --user weblate /tmp:/tmp weblate weblate importusers /tmp/users.json
```

---

**Docker**

**Installing on Debian and Ubuntu**

**Installing on SUSE and openSUSE**

**Installing on RedHat, Fedora and CentOS**

---

### add_suggestions

```bash
weblate add_suggestions <project> <component> <language> <file>
```

Imports a translation from the file to use as a suggestion for the given translation. It skips duplicated translations; only different ones are added.

```bash
--author USER@EXAMPLE.COM
```

E-mail of author for the suggestions. This user has to exist prior to importing (you can create one in the admin interface if needed).

```bash
weblate --author michal@cihar.com add_suggestions weblate application cs /tmp/suggestions-cs.po
```

---

### auto_translate

```bash
weblate auto_translate <project> <component> <language>
```

```bash
--source PROJECT/COMPONENT
```

Performs automatic translation based on other component translations.

```bash
--user USERNAME
```

Specify username listed as author of the translations. "Anonymous user" is used if not specified.

```bash
--overwrite
```

Whether to overwrite existing translations.

```bash
--inconsistent
```

Whether to overwrite existing translations that are inconsistent (see `--inconsistent`).

```bash
--add
```

Automatically add language if a given translation does not exist.

```bash
--mt MT
```

Use machine translation instead of other components as machine translations.

```bash
--threshold THRESHOLD
```

Similarity threshold for machine translation, defaults to 80.

```bash
--mode MODE
```

Specify translation mode, default is `translate` but `fuzzy` or `suggest` can be used.

```bash
weblate auto_translate --user nijel --inconsistent --source weblate/ website application weblate cs
```

---

```bash
```

301
celery_queues

weblate celery_queues

Displays length of Celery task queues.

checkgit

weblate checkgit <project|project/component>

Prints current state of the back-end Git repository.
You can either define which project or component to update (for example weblate/application), or use --all to update all existing components.

commitgit

weblate commitgit <project|project/component>

Commits any possible pending changes to the back-end Git repository.
You can either define which project or component to update (for example weblate/application), or use --all to update all existing components.

commit_pending

weblate commit_pending <project|project/component>

Commits pending changes older than a given age.
You can either define which project or component to update (for example weblate/application), or use --all to update all existing components.

--age HOURS
Age in hours for committing. If not specified the value configured in Component configuration is used.

Note: This is automatically performed in the background by Weblate, so there no real need to invoke this manually, besides forcing an earlier commit than specified by Component configuration.

commit_pending

--age COMMIT_PENDING_HOURS

cleanuptrans

weblate cleanuptrans

Cleans up orphaned checks and translation suggestions. There is normally no need to run this manually, as the cleanups happen automatically in the background.

createadmin

weblate createadmin

Creates an admin account with a random password, unless it is specified.

--password PASSWORD
Provides a password on the command-line, to not generate a random one.

--no-password
Do not set password, this can be useful with --update.

--username USERNAME
Use the given name instead of admin.
--email USER@EXAMPLE.COM
Specify the admin e-mail address.

--name
Specify the admin name (visible).

--update
Update the existing user (you can use this to change passwords).

2.9  Added parameters --username, --email, --name and --update.

dump_memory

debate dump_memory

2.20
Export a JSON file containing Weblate Translation Memory content.

Note:
Weblate dump_memory

dumpuserdata

debate dumpuserdata <file.json>
Dumps userdata to a file for later use by importuserdata

Note: This comes in handy when migrating or merging Weblate instances.

import_demo

debate import_demo

4.1
Creates a demo project with components based on <https://github.com/WeblateOrg/demo>.
This can be useful when developing Weblate.

import_json

debate import_json <json-file>

2.7
Batch import of components based on JSON data.
The imported JSON file structure pretty much corresponds to the component object (see GET /api/components/(string:project)/(string:component)/). You have to include the name and filemask fields.

--project PROJECT
Specifies where the components will be imported from.

--main-component COMPONENT
Use the given VCS repository from this component for all of them.

--ignore
Skip (already) imported components.

--update
Update (already) imported components.

2.9  The parameters --ignore and --update are there to deal with already imported components.
Example of JSON file:
import_memory

**weblate import_memory** `<file>`

**2.20+**

Imports a TMX or JSON file into the Weblate translation memory.

--language-map **LANGMAP**

Allows mapping languages in the TMX to the Weblate translation memory. The language codes are mapped after normalization usually done by Weblate.

--language-map **en_US**:en will for example import all en_US strings as en ones.

**TMX IMPORT**

**Weblate**

**import_project**

**weblate import_project** `<project>` `<gitrepo>` `<branch>` `<filemask>`

**3.0+**

The import_project command is now based on the **add-on**, leading to some changes in behavior and what parameters are accepted.

Batch imports components into project based on filemask.

`<project>` names an existing project, into which the components are to be imported.

The `<gitrepo>` defines the Git repository URL to use, and `<branch>` signifies the Git branch. To import additional translation components from an existing Weblate component, use a `weblate://<project>/<component>` URL for the `<gitrepo>`.

The `<filemask>` defines file discovery for the repository. It can be either be made simple using wildcards, or it can use the full power of regular expressions.

The simple matching uses `**` for component name and `*` for language, for example: `**/*.po`

The regular expression has to contain groups named `component` and `language`. For example: `(\?P<language>[^/])*/(\?P<component>[^-/]+)\.po`

The import matches existing components based on files and adds the ones that do not exist. It does not change already existing ones.

--name-template **TEMPLATE**

Customize the name of a component using Django template syntax.

Documentation: `{{ component }}`

--base-file-template **TEMPLATE**

Customize the base file for monolingual translations.

{{ component }}/res/values/string.xml
**--new-base-template** TEMPLATE
Customize the base file for addition of new translations.
```ts
{{ component }}/ts/en.ts
```

**--file-format** FORMAT
You can also specify the file format to use (see Component configuration), the default is auto-detection.

**--language-regex** REGEX
You can specify language filtering (see Component configuration) with this parameter. It has to be a valid regular expression.

**--main-component**
You can specify which component will be chosen as the main one—the one actually containing the VCS repository.

**--license** NAME
Specify the overall, project or component translation license.

**--license-url** URL
Specify the URL where the translation license is to be found.

**--vcs** NAME
In case you need to specify which version control system to use, you can do it here. The default version control is Git.

To give you some examples, let’s try importing two projects.

First The Debian Handbook translations, where each language has separate a folder with the translations of each chapter:

```bash
weblate import_project
  debian-handbook
  git://anonscm.debian.org/debian-handbook/debian-handbook.git
  squeeze/master
  */**.po
```

Then the Tanaguru tool, where the file format needs be specified, along with the base file template, and how all components and translations are located in single folder:

```bash
weblate import_project
  --file-format=properties
  --base-file-template=web-app/tgol-web-app/src/main/resources/i18n/%s-I18N.properties
  tanaguru
  https://github.com/Tanaguru/Tanaguru
  master
  web-app/tgol-web-app/src/main/resources/i18n/**-I18N_*.properties
```

More complex example of parsing of filenames to get the correct component and language out of a filename like `src/security/Numerous_security_holes_in_0.10.1.de.po`:

```bash
weblate import_project
  tails
  git://git.tails.boum.org/tails master
  'wiki/src/security/([^<component>].*)\.(^<language>[^.]*)\..po$'
```

Filtering only translations in a chosen language:

```bash
./manage import_project
  --language-regex '^<cs|sk>$'
  weblate
  https://github.com/WeblateOrg/weblate.git
  'weblate/locale/*/LC_MESSAGES/**.po'
```

Importing Sphinx documentation split to multiple files:

```bash
$ weblate import_project --name-template 'Documentation: %s'
  --file-format=po
  project https://github.com/project/docs.git master
  'docs/locale/*/LC_MESSAGES/**.po'
```

Importing Sphinx documentation split to multiple files and directories:

305
import_project

```bash
$ weblate import_project --name-template 'Directory 1: %s' \ 
  --file-format po \ 
  project https://github.com/project/docs.git master \ 
  'docs/locale/*/LC_MESSAGES/dir1/**.po'
$ weblate import_project --name-template 'Directory 2: %s' \ 
  --file-format po \ 
  project https://github.com/project/docs.git master \ 
  'docs/locale/*/LC_MESSAGES/dir2/**.po'
```

**Note:**
More detailed examples can be found in the starting chapter, alternatively you might want to use `import_json`.

**importuserdata**

`weblate importuserdata <file.json>`

Imports user data from a file created by `dumpuserdata`.

**importusers**

`weblate importusers --check <file.json>`

Imports users from JSON dump of the Django auth_users database.

--check

With this option it will just check whether a given file can be imported and report possible conflicts arising from usernames or e-mails.

You can dump users from the existing Django installation using:

```bash
weblate dumpdata auth.User > users.json
```

**install_addon**

**3.2 New.**

`weblate install_addon --addon ADDON <project|project/component>`

--addon ADDON
Name of the add-on to install. For example `weblate.gettext.customize`.

--configuration CONFIG

--update

You can either define which project or component to install the add-on in (for example `weblate/application`), or use `--all` to include all existing components.

To install `gettext` for all components:

```bash
weblate install_addon --addon weblate.gettext.customize --config '{"width": -1}' --update --all
```

**Note:**

```bash
weblate
```
list_languages

weblate list_languages <locale>
Lists supported languages in MediaWiki markup - language codes, English names and localized names. This is used to generate <https://wiki.l10n.cz/Slovn%C3%ADk_s_n%C3%A1zvy_jazyk%C5%AF>.

list_translators

weblate list_translators <project|project/component>
Lists translators by contributed language for the given project:

<table>
<thead>
<tr>
<th>[French]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jean Dupont <a href="mailto:jean.dupont@example.com">jean.dupont@example.com</a></td>
</tr>
<tr>
<td>[English]</td>
</tr>
<tr>
<td>John Doe <a href="mailto:jd@example.com">jd@example.com</a></td>
</tr>
</tbody>
</table>

--language-code
List names by language code instead of language name.
You can either define which project or component to use (for example weblate/application), or use --all to list translators from all existing components.

list_versions

weblate list_versions
Lists all Weblate dependencies and their versions.

loadpo

weblate loadpo <project|project/component>
Reloads translations from disk (for example in case you have done some updates in the VCS repository).

--force
Force update, even if the files should be up-to-date.

--lang LANGUAGE
Limit processing to a single language.
You can either define which project or component to update (for example weblate/application), or use --all to update all existing components.

Loadpo: You seldom need to invoke this, Weblate will automatically load changed files for every VCS update. This is needed in case you manually changed an underlying Weblate VCS repository or in some special cases following an upgrade.

lock_translation

weblate lock_translation <project|project/component>
Prevents further translation of a component.

---lock Translation
Useful in case you want to do some maintenance on the underlying repository.
You can either define which project or component to update (for example weblate/application), or use --all to update all existing components.

unlock_translation
move_language

weblate move_language source target

3.0

Allows you to merge language content. This is useful when updating to a new version which contains aliases for previously unknown languages that have been created with the (generated) suffix. It moves all content from the source language to the target one.

```bash
weblate move_language cze cs
```

After moving the content, you should check whether there is anything left (this is subject to race conditions when somebody updates the repository meanwhile) and remove the (generated) language.

pushgit

weblate pushgit <project|project/component>

Pushes committed changes to the upstream VCS repository.

```
--force-commit
Force commits any pending changes, prior to pushing.
```

You can either define which project or component to update (for example weblate/application), or use --all to update all existing components.

```bash
: Weblate pushes changes automatically if in Component configuration is turned on, which is the default.
```

unlock_translation

weblate unlock_translation <project|project/component>

Unlocks a given component, making it available for translation.

```bash
: Useful in case you want to do some maintenance on the underlying repository.
```

You can either define which project or component to update (for example weblate/application), or use --all to update all existing components.

```bash
lock_translation
```

setupgroups

weblate setupgroups

Configures default groups and optionally assigns all users to that default group.

```
--no-privs-update
Turns off automatic updating of existing groups (only adds new ones).
```

```
--no-projects-update
Prevents automatic updates of groups for existing projects. This allows adding newly added groups to existing projects, see
```

```bash
```
setuplang

weblate setuplang
Updates list of defined languages in Weblate.

--no-update
Turns off automatic updates of existing languages (only adds new ones).

updatechecks

weblate updatechecks <project|project/component>
Updates all checks for all strings.

Notes: Useful for upgrades which do major changes to checks.

You can either define which project or component to update (for example weblate/application), or use --all to update all existing components.

updategit

weblate updategit <project|project/component>
Fetches remote VCS repositories and updates the internal cache.

You can either define which project or component to update (for example weblate/application), or use --all to update all existing components.

Notes: Usually it is better to configure hooks in the repository to trigger updategit, instead of regular polling by updategit.

4.0 releases

The 4.0 release includes a number of improvements:

- Support for new backend PostgreSQL
- Improved performance
- More efficient database indexing

Web interface changes

Manage | Post announcement

309
Translations will be used only if they reach 60%.

Category
Info (light blue)

Category defines color used for the message.

Expiry date
mm/dd/yyyy

The message will be not shown after this date. Use it to announce string freeze and translation deadline for next release.

Notify users
The message is shown for all translations within the project, until its given expiry, or permanently until it is deleted.

Add
Add Announcement

Required fields are marked in bold.

Message: Translations will be used only if they reach 10% | 

- You can use Markdown and mention users by @username.

Project: WeblateOrg

Component: 

Language: 

Category: Info (lightblue)

Category defines color used for the message.

Expiration: Today

The message will be not shown after this date. Use it to announce string freeze and translation deadlines for next release.

Notify users

Save and add another  Save and continue editing  SAVE
1. Define *Automatic component list assignment* with `^ . * $` as regular expression in both the project and the component fields, as shown on this image:
Optional Weblate modules

Several optional modules are available for your setup.

**Git exporter**

Provides you read-only access to the underlying Git repository using HTTP(S).

1. Add `weblate.gitexport` to installed apps in `settings.py`:
   ```python
   INSTALLED_APPS += ("weblate.gitexport",)
   ```

2. Export existing repositories by migrating your database after installation:
   ```bash
   weblate migrate
   ```
Usage

The module automatically hooks into Weblate and sets the exported repository URL in the Component configuration. The repositories are accessible under the /git/ part of the Weblate URL, for example https://example.org/git/weblate/main/.

Repositories for publicly available projects can be cloned without authentication:

```bash
git clone 'https://example.org/git/weblate/main/
```

Access to browse the repositories with restricted access (with Private access control or when REQUIRE_LOGIN is enabled) requires an API token which can be obtained in your user profile:

```bash
git clone 'https://user:KEY@example.org/git/weblate/main/
```

**Tip:** By default members or Users group and anonymous user have access to the repositories for public projects via Access repository and Power user roles.

---

2.4

This is used on Hosted Weblate to define billing plans, track invoices and usage limits.

---

1. Add weblate.billing to installed apps in settings.py:

```python
INSTALLED_APPS += ("weblate.billing",)
```

2. Run the database migration to optionally install additional database structures for the module:

```bash
weblate migrate
```

Usage

After installation you can control billing in the admin interface. Users with billing enabled will get new Billing tab in their.

The billing module additionally allows project admins to create new projects and components without being superusers (see Adding translation projects and components). This is possible when following conditions are met:

- The billing is in its configured limits (any overusage results in blocking of project/component creation) and paid (if its price is non zero)
- The user is admin of existing project with billing or user is owner of billing (the latter is necessary when creating new billing for users to be able to import new projects).
- Upon project creation user is able to choose which billing should be charged for the project in case he has access to more of them.

---

2.15

This is used on Hosted Weblate to provide required legal documents. It comes provided with blank documents, and you are expected to fill out the following templates in the documents:

- Terms of service document
- Privacy policy document
- Short overview of the terms of service and privacy policy

**Tip:** Legal documents for the Hosted Weblate service are available in this Git repository <https://github.com/WeblateOrg/wllegal/tree/main/wllegal/templates/legal/documents>.
Most likely these will not be directly usable to you, but might come in handy as a starting point if adjusted to meet your needs.

1. Add `weblate.legal` to installed apps in `settings.py`:

```python
INSTALLED_APPS += ("weblate.legal",)

# Optional:

# Social auth pipeline to confirm TOS upon registration/subsequent sign in
SOCIAL_AUTH_PIPELINE += ("weblate.legal.pipeline.tos_confirm",)

# Middleware to enforce TOS confirmation of signed in users
MIDDLEWARE += ["weblate.legal.middleware.RequireTOSMiddleware",]
```

2. Run the database migration to optionally install additional database structures for the module:

```bash
weblate migrate
```

3. Edit the legal documents in the `weblate/legal/templates/legal/` folder to match your service.

**Usage**

After installation and editing, the legal documents are shown in the Weblate UI.

**Avatars**

Avatars are downloaded and cached server-side to reduce information leaks to the sites serving them by default. The built-in support for fetching avatars from e-mails addresses configured for it can be turned off using `ENABLE_AVATARS`.

Weblate currently supports:

- Gravatar
- Libravatar

**Spam protection**

You can protect against spamming by users by using the Akismet service.

1. Install the `akismet` Python module (this is already included in the official Docker image).

2. Obtain the Akismet API key.

3. Store it as `AKISMET_API_KEY` or `WEBLATE_AKISMET_API_KEY` in Docker.

Following content is sent to Akismet for checking:

```text
This (among other things) relies on IP address of the client, please see [AKISMET API KEY] for properly configuring that.
```

**Notes:**

- This (among other things) relies on IP address of the client, please see [AKISMET_API_KEY] for properly configuring that.
Signing Git commits with GnuPG

### 3.1 KEY.

All commits can be signed by the GnuPG key of the Weblate instance.

1. Turn on `WEBLATE_GPG_IDENTITY`. (Weblate will generate a GnuPG key when needed and will use it to sign all translation commits.)

This feature needs GnuPG 2.1 or newer installed.

You can find the key in the `DATA_DIR` and the public key is shown on the "About" page:

2. Alternatively you can also import existing keys into Weblate, just set `HOME=$DATA_DIR/home` when invoking gpg:

   ```
   WEBLATE_GPG_IDENTITY
   ```

### 3.2 RATE.

#### 4.6 RPM.

The rate limiting no longer applies to superusers.
The settings can be also applied in the Docker container by adding `WEBLATE_` prefix to the setting name, for example `RATELIMIT_ATTEMPTS` becomes `WEBLATE_RATELIMIT_ATTEMPTS`.

API

Fedora Messaging

Fedora Messaging is AMQP-based publisher for all changes happening in Weblate. You can hook additional services on changes happening in Weblate using this.

The Fedora Messaging integration is available as a separate Python module `weblate-fedora-messaging`. Please see `<https://github.com/WeblateOrg/fedora_messaging/>` for setup instructions.

Customizing Weblate

Extend and customize using Django and Python. Contribute your changes upstream so that everybody can benefit. This reduces your maintenance costs; code in Weblate is taken care of when changing internal interfaces or refactoring the code.

Neither internal interfaces nor templates are considered a stable API. Please review your own customizations for every upgrade, the interfaces or their semantics might change without notice.

Creating a Python module

If you are not familiar with Python, you might want to look into *Python For Beginners*, explaining the basics and pointing to further tutorials.

To write some custom Python code (called a module), a place to store it is needed, either in the system path (usually something like `/usr/lib/python3.7/site-packages/`) or in the Weblate directory, which is also added to the interpreter search path.

Better yet, turn your customization into a proper Python package:
1. Create a folder for your package (we will use `weblate_customization`).
2. Within it, create a `setup.py` file to describe the package:

```python
from setuptools import setup

setup(
    name="weblate_customization",
    version="0.0.1",
    author="Your name",
    author_email="yourname@example.com",
    description="Sample Custom check for Weblate.",
    license="GPLv3+",
    keywords="Weblate check example",
)"
packages=["weblate_customization"],
)

3. Create a folder for the Python module (also called weblate_customization) for the customization code.
4. Within it, create a __init__.py file to make sure Python can import the module.
5. This package can now be installed using pip install -e. More info to be found in “Editable” Installs.
6. Once installed, the module can be used in the Weblate configuration (for example weblate_customization.
   checks.FooCheck).

Your module structure should look like this:

```
weblate_customization
├── setup.py
└── weblate_customization
    ├── __init__.py
    └── addons.py
    └── checks.py
```

You can find an example of customizing Weblate at <https://github.com/WeblateOrg/customize-example>, it covers all the topics described below.

**Changing the logo**

1. **Django**: Creating a Python module

   Branding appears in the following files:
   - icons/weblate.svg Logo shown in the navigation bar.
   - logo-*.png Web icons depending on screen resolution and web-browser.
   - favicon.ico Web icon used by legacy browsers.
   - weblate-*.png Avatars for bots or anonymous users. Some web-browsers use these as shortcut icons.
   - email-logo.png Used in notifications e-mails.

2. Add it to INSTALLED_APPS:

   ```
   INSTALLED_APPS = (
       # Add your customization as first
       "weblate_customization",
       # Weblate apps are here...
   )
   ```

3. Run weblate collectstatic --noinput, to collect static files served to clients.

**Changing the configuration**

1. **Python**: Creating a Python module

   ```
   # Checks
   CHECK_LIST += ("weblate_customization.checks.FooCheck",)
   
   # Autofixes
   AUTOFIX_LIST += ("weblate_customization.autofix.FooFixer",)
   
   # Add-ons
   WEBLATE_ADDONS += ("weblate_customization.addons.ExamplePreAddon",)
   ```
## Site administration

### REPORTS
- Website support status
- Status of repositories
- SSH keys
- Performance report
- Translation-memory

### ACCOUNTS
- Audit log entries
- User profiles
- Verified e-mails

### AUTH TOKEN
- Tokens

### AUTHENTICATION
- Groups
- Roles
- Users

### BILLING
- Billing plans
- Customer billings
- Invoices

### PROJECTS
- First groups
- Projects

### LEGAL
- TOS agreements

### PYTHON SOCIAL AUTH
- Associations
- Neames
- User social audits

### SCREENSHOTS
- Screenshots

### TRANSLATION MEMORY
- Translation-memory entries

### WEBSITE CONFIGURATION
- Settings

### WEBSITE LANGUAGES
- Languages

### WEBSITE TRANSLATORS
- Announcements
- Component bots
- Components
- Contributor agreements
- Projects
Weblate translations

Project configuration

Component configuration

Weblate languages

Weblate administration

Add Project

Required fields are marked in bold.

**Project name:**

**URL slug:**

**Project website:**

**Translation instructions:**

You can use Markdown and mention users by @username.

- Set "Language-Team" header
- Set shared translation memory
- Contribute to shared translation memory

**Access control:**

- Protected

- Enable reviewers
- Enable source reviewers
- Enable hooks

**Language allows:**

321
### Component configuration

<table>
<thead>
<tr>
<th>ID</th>
<th>Component configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>324</td>
<td></td>
</tr>
</tbody>
</table>
Weblate

3.8

Additionally, when Weblate is turned on:
List of public projects (name, URL and website)
Discover Weblate is an opt-in service that makes it easier for users to find Weblate servers and communities. Users can browse registered services on [https://weblate.org/discover/], and find there projects to contribute.

Participating in Discover Weblate makes Weblate submit some information about your server, please see [Weblate](https://weblate.org/subscription/discovery/).

To list your server with an active support subscription (see [Weblate](https://weblate.org/subscription/discovery/)) in Discover Weblate all you need to do is turn this on in the management panel:

List your server without a support subscription in Discover Weblate:
1. [https://weblate.org/user/]
2. Weblate [https://weblate.org/subscription/discovery/]
3. Confirm the service activation in your Weblate and turn on the discovery listing in your Weblate management page using *Enable discovery* button:

You can customize the listing by providing a text and image (570 x 260 pixels) at <https://weblate.org/user/>.

**Legal documents**

**NOTE:** Herein you will find various legal information you might need to operate Weblate in certain legal jurisdictions. It is provided as a means of guidance, without any warranty of accuracy or correctness. It is ultimately your responsibility to ensure that your use of Weblate complies with all applicable laws and regulations.

**ITAR and other export controls**

Weblate can be run within your own datacenter or virtual private cloud. As such, it can be used to store ITAR or other export-controlled information, however, end users are responsible for ensuring such compliance.

The Hosted Weblate service has not been audited for compliance with ITAR or other export controls, and does not currently offer the ability to restrict translations access by country.
US encryption controls

Weblate does not contain any cryptographic code, but might be subject export controls as it uses third party components utilizing cryptography for authentication, data-integrity and confidentiality.

Most likely Weblate would be classified as ECCN 5D002 or 5D992 and, as publicly available libre software, it should not be subject to EAR (see Encryption items NOT Subject to the EAR).

Software components used by Weblate (listing only components related to cryptographic function):

See https://wiki.python.org/moin/PythonSoftwareFoundationLicenseFaq#Is_Python_subject_to_export_laws.3F

Optionally used by Weblate

Used by Python and cURL

The strength of encryption keys depends on the configuration of Weblate and the third party components it interacts with, but in any decent setup it will include all export restricted cryptographic functions:

- In excess of 56 bits for a symmetric algorithm
- Factorisation of integers in excess of 512 bits for an asymmetric algorithm
- Computation of discrete logarithms in a multiplicative group of a finite field of size greater than 512 bits for an asymmetric algorithm
- Discrete logarithms in a group different than above in excess of 112 bits for an asymmetric algorithm

Weblate doesn’t have any cryptographic activation feature, but it can be configured in a way where no cryptography code would be involved. The cryptographic features include:

- Accessing remote servers using secure protocols (HTTPS)
- Generating signatures for code commits (PGP)

Export Controls (EAR) on Open Source Software

Weblate

If you find a possible mistake in the source string, you can mark it with a comment in the Weblate editor. This way, it can be discussed and corrected. If you’re certain, you can also click on the link in the Source string location section and submit a PR with your correction.
You are welcome to improve the documentation page of your choice. Do it easily by clicking the Edit on GitHub button in the top-right corner of the page.

Please respect these guidelines while writing:

1. Don’t remove part of the documentation if it’s valid.
2. Use clear and easily-understandable language. You are writing tech docs, not a poem. Not all docs readers are native speakers, be thoughtful.
3. Don’t be afraid to ask if you are not certain. If you have to ask about some feature while editing, don’t change its docs before you have the answer. This means: You change or ask. Don’t do both at the same time.
4. Verify your changes by performing described actions while following the docs.
5. Send PR with changes in small chunks to make it easier and quicker to review and merge.
6. If you want to rewrite and change the structure of a big article, do it in two steps:
   1. Rewrite
   2. Once the rewrite is reviewed, polished, and merged, change the structure of the paragraphs in another PR.

---

You can translate the docs.

---

**Extending built-in language definitions**

The language definitions are in the weblate-language-data repository.

You are welcome to add missing language definitions to languages.csv, other files are generated from that file.

---

**Weblate**

[issue](https://gitlab.weblate.org/weblate/weblate/issues)   [GitHub](https://github.com/WeblateOrg/weblate)   [discussions](https://gitlab.weblate.org/weblate/weblate/discussions)

**Weblate**

Weblate [donate page](https://weblate.donation/)   Libre [FreedomBox](https://freedombox.org/)   Weblate

**Weblate**

Weblate [donate page](https://weblate.donation/)   Libre [FreedomBox](https://freedombox.org/)   Weblate

---

Weblate: Yashiro Ccs

Cheng-Chia Tseng

Timon Reinhard

Cassidy James

Loic Dachary

Marozed

https://freedombox.org/

GNU Solidario (GNU Health)

BallotReady

Richard Nespithal

MyExpenses.Mobi

---

330
Weblate

1. Weblate:
   ```
git clone https://github.com/WeblateOrg/weblate.git
cd weblate
   ```

2. virtualenv:
   ```
virtualenv .venv
.venv/bin/activate
   ```

3. Weblate:
   ```
    pip install -e .
   ```

4. Weblate:
   ```
    pip install -r requirements-dev.txt
   ```

5. Celery Worker:
   ```
   ./weblate/examples/celery start
   ```

6. Local testing:
   ```
   ./scripts/test-database
   ./manage.py test
   ```

Docker Weblate

The script also accepts some parameters, to execute tests, run it with the test parameter and then specify any test parameters, for example running only tests in the weblate.machine module:

```
./rundev.sh test --failfast weblate.machine
```
./rundev.sh logs

Docker:

PyCharm

PyCharm IDE: Weblate

GitHub

PyCharm virtualenv
Be careful with the property called `No reload`: It prevents the server from being reloaded live if you modify files. This allows the existing debugger breakpoints to persist, when they normally would be discarded upon reloading the server.
import demo createadmin

Weblate

Weblate GitHub

Weblate GitHub Weblate

Weblate

Weblate

PEP-8 black flake 8 .pre-commit-config.yaml

requirements-lint.txt Weblate pre-commit install

pre-commit run --all

Debugging Weblate

Turning on debug mode will make the exceptions show in the browser. This is useful to debug issues in the web interface, but not suitable for production environment as it has performance consequences and might leak private data.

Weblate logs

Weblate can produce detailed logs of what is going on in the background. In the default configuration it uses syslog and that makes the log appear either in /var/log/messages or /var/log/syslog (depending on your syslog daemon configuration).

The Celery process (see Celery) usually produces own logs as well. The example system-wide setups log to several files under /var/log/celery/.

Docker containers log to their output (as usual in the Docker world), so you can look at the logs using docker-compose logs.

Contains LOGGING configuration.
Not processing background tasks

Lot of things happen in background Celery workers. In case things like sending out e-mails or component removal does not work, there might be some issue with it.

Check Celery process is running, see Celery [command]
Check Celery queue status either in [command] or using celery_queues
Look into Celery logs for errors (see Weblate logs)

Not receiving e-mails from Weblate

You can verify whether outgoing e-mail is working correctly by using the sendtestemail management command (see Invoking management commands for instructions on how to invoke it in different environments) or using [command] under the Tools tab.

These send e-mail directly, so this verifies that your SMTP configuration is correct (see [command]). Most of the e-mails from Weblate are however sent in the background and there might be some issues with Celery involved as well, please see Not processing background tasks for debugging that.

Analyzing application crashes

In case the application crashes, it is useful to collect as much info about the crash as possible. The easiest way to achieve this is by using third-party services which can collect such info automatically. You can find info on how to set this up in [command].

Silent failures

Lots of tasks are offloaded to Celery for background processing. Failures are not shown in the user interface, but appear in the Celery logs. Configuring [command] helps you to notice such failures easier.

Performance issues

In case Weblate performs badly in some situation, please collect the relevant logs showing the issue, and anything that might help figuring out where the code might be improved.

In case some requests take too long without any indication, you might want to install dogslove along with [command] and get pinpointed and detailed tracebacks in the error collection tool.

Weblate [command]

[command]: [command] Weblate [command]

Weblate [command] Django [command]

Weblate [command]: [command] Sphinx [command]

Docker [command]: Docker [command] Weblate [command]

*Django*[command] [command] Weblate [command]: Weblate [command]
Weblate

Django

Optional Weblate modules

accounts
addons

Weblate

api

Django REST framework

auth

billing

checks

fonts

formats

translate-toolkit

gitexport

lang

legal

machinery

memory

screenshots

OCR

trans

utils

vcs

wladmin

Django
class weblate.addons.base.BaseAddonDelete

    Weblate
    # Weblate

class weblate.addons.base.BaseAddonDelete

    classmethod can_install(component, user)

    configure (settings)

    daily (component)

    classmethod get_add_form(user, component, **kwargs)

    get_settings_form(user, component, **kwargs)

    post_add(translation)

    post_commit(component)

    post_push(component)

    post_update(component, previous_head: str, skip_push: bool)

    previous_head(str) -- HEAD

    skip_push(bool) -- Whether the add-on operation should skip pushing changes upstream. Usually you can pass
    this to underlying methods as commit_and_push or commit_pending.

    pre_commit(translation, author)

    pre_push(component)

    pre_update(component)

    save_state()

    stay_on_create = False

    Weblate

    store_post_load(translation, store)

    It receives an instance of a file format class as a argument.

    This is useful to modify file format class parameters, for example adjust how the file will be saved.

    unit_pre_create(unit)

    #
    # Copyright © 2012 - 2021 Michal Čihař <michal@cihar.com>
    #
    # This file is part of Weblate <https://weblate.org/>
    #
    # This program is free software: you can redistribute it and/or modify
    # it under the terms of the GNU General Public License as published by
    # the Free Software Foundation, either version 3 of the License, or
    # (at your option) any later version.
    #
    # This program is distributed in the hope that it will be useful,
```python
from django.utils.translation import gettext_lazy as _
from weblate.addons.base import BaseAddon
from weblate.addons.events import EVENT_PRE_COMMIT

class ExampleAddon(BaseAddon):
    # Filter for compatible components, every key is
    # matched against property of component
    compat = {
        "file_format": {"po", "po-mono"}
    }  # List of events add-on should receive
    events = (EVENT_PRE_COMMIT,)
    # Add-on unique identifier
    name = "weblate.example.example"
    # Verbose name shown in the user interface
    verbose = _("Example add-on")
    # Detailed add-on description
    description = _("This add-on does nothing it is just an example.")

    # Callback to implement custom behavior
    def pre_commit(self, translation, author):
        return
```

# Weblate

Weblate supports the latest, stable releases of all major browsers and platforms.
Alternative browsers which use the latest version of WebKit, Blink, or Gecko, whether directly or via the platform’s web view API, are not explicitly supported. However, Weblate should (in most cases) display and function correctly in these browsers as well.

Older browsers might work, but some features might be limited.

The yarn package manager is used to update third party libraries. The configuration lives in `scripts/yarn` and there is a wrapper script `scripts/yarn-update` to upgrade the libraries, build them and copy to correct locations in `weblate/static/vendor`, where all third party frontend code is located. The Weblate specific code should be placed directly in `weblate/static` or feature specific subdirectories (for example `weblate/static/editor`).

Adding new third-party library typically consists of:

```bash
# Add a yarn package
yarn --cwd scripts/yarn add PACKAGE
# Edit the script to copy package to the static folder
edit scripts/yarn-update
# Run the update script
./scripts/yarn-update
# Add files to git
git add .
```
document.write(gettext('this is to be translated'));

```javascript
var object_count = 1 // or 0, or 2, or 3, ...
  s = ngettext('literal for the singular case',
    'literal for the plural case', object_count);

fmts = ngettext('There is %s object. Remaining: %s',
  'There are %s objects. Remaining: %s', object_count);
  s = interpolate(fmts, [11, 20]);
// s is 'There are 11 objects. Remaining: 20'
```

Translation topic in the Django documentation

Weblate depends on third-party components for many things. In case you find a vulnerability affecting one of those components in general, please report it directly to the respective project. Some of these are:

- Django
- Django REST framework
- Python Social Auth
Weblate testsuite and continuous integration

Testsuites exist for most of the current code, increase coverage by adding testcases for any new functionality, and verify that it works.

Continuous integration

Current test results can be found on GitHub Actions and coverage is reported on Codecov.

There are several jobs to verify different aspects:
- Unit tests
- Documentation build and external links
- Migration testing from all supported releases
- Code linting

Setup verification (ensures that generated dist files do not miss anything and can be tested)

The configuration for the CI is in .github/workflows directory. It heavily uses helper scripts stored in ci directory. The scripts can be also executed manually, but they require several environment variables, mostly defining Django settings file to use and database connection. The example definition of that is in scripts/test-database:

```bash
# Simple way to configure test database from environment

export CI_DATABASE=${1:-postgresql}

# Database server configuration
export CI_DB_USER=weblate
export CI_DB_PASSWORD=weblate
export CI_DB_HOST=127.0.0.1

# Django settings module to use
export DJANGO_SETTINGS_MODULE=weblate.settings_test
```

The simple execution can look like:

```bash
./scripts/test-database
./ci/run-migrate
./ci/run-test
./ci/run-docs
```

Local testing

To run a testsuite locally, use:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test
```

**Note:** You will need a database (PostgreSQL) server to be used for tests. By default Django creates separate database to run tests with test_ prefix, so in case your settings is configured to use weblate, the tests will use test_weblate database. See Weblate [Readme](https://github.com/sistecnologia/weblate) for setup instructions.

The weblate/settings_test.py is used in CI environment as well (see Continuous integration) and can be tuned using environment variables:

```bash
# Simple way to configure test database from environment

export CI_DATABASE=${1:-postgresql}

# Database server configuration
export CI_DB_USER=weblate
export CI_DB_PASSWORD=weblate
export CI_DB_HOST=127.0.0.1
```
# Django settings module to use
export DJANGO_SETTINGS_MODULE=weblate.settings_test

Prior to running tests you should collect static files as some tests rely on them being present:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py collectstatic
```

You can also specify individual tests to run:

```
DJANGO_SETTINGS_MODULE=weblate.settings_test ./manage.py test weblate.
```

---

The tests can also be executed inside developer docker container, see [Docker Weblate](https://weblate.org/docs/). See [Django](https://docs.djangoproject.com/) for more info on running and writing tests for Django.

---

## Weblate JSON Schema

### Weblate

https://weblate.org/schemas/weblate-memory.schema.json

```json
{
  "category": 1,
  "origin": "/test.tmx",
  "source": "Hello",
  "target": "Ahoj",
  "source_language": "en",
  "target_language": "cs"
}
```

---

342
Table 7 - cs

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>^\[^ ]+$</code></td>
<td>cs</td>
</tr>
<tr>
<td>language</td>
<td><code>cs</code></td>
</tr>
<tr>
<td>hide_completed</td>
<td><code>False</code></td>
</tr>
</tbody>
</table>

**Weblate**

https://weblate.org/schemas/weblate-userdata.schema.json

```json
{
  "basic": {
    "username": "noreply@example.com",
    "email": "noreply@example.com",
    "date_joined": "2019-11-18T18:53:54.862Z"
  },
  "profile": {
    "language": "cs",
    "suggested": 1,
    "translated": 24,
    "uploaded": 1
  },
  "hide_completed": "False",
  "secondary_in_zen": "True",
  "hide_source_secondary": "True"
}
```

343
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>False</td>
</tr>
<tr>
<td>editor_link</td>
<td>^.*$</td>
</tr>
<tr>
<td>translate_mode</td>
<td>0</td>
</tr>
<tr>
<td>zen_mode</td>
<td>Zen</td>
</tr>
<tr>
<td>special_chars</td>
<td>^.*$</td>
</tr>
<tr>
<td>dashboard_view</td>
<td>1</td>
</tr>
<tr>
<td>dashboard_component_list</td>
<td>null</td>
</tr>
<tr>
<td>languages</td>
<td>cs</td>
</tr>
<tr>
<td>secondary_languages</td>
<td>sk</td>
</tr>
<tr>
<td>watched</td>
<td>weblate</td>
</tr>
<tr>
<td>auditlog</td>
<td>IP</td>
</tr>
<tr>
<td>user_agent</td>
<td>^.*$</td>
</tr>
</tbody>
</table>

Table 8 -
Weblate

GitHub <https://github.com/WeblateOrg/weblate/milestones>

1. ./scripts/list-translated-languages
2. ./scripts/prepare-release
3. make -C docs update-screenshots
4. Merge any possibly pending translations wlc push; git remote update; git merge origin/weblate
5. ./scripts/create-release --tag
6. Docker
7. GitHub
8. Docker
9. Helm
10. .github/workflows/migrations.yml
11. Increase version in the website download links.
12. ./scripts/set-version

345
Tracking dependencies for vulnerabilities

Security issues in our dependencies are monitored using Dependabot. This covers the Python and JavaScript libraries, and the latest stable release has its dependencies updated to avoid vulnerabilities.

There might be vulnerabilities in third-party libraries which do not affect Weblate, so those are not addressed by releasing bugfix versions of Weblate.

Docker container security

The Docker containers are scanned using Anchore and Trivy. This allows us to detect vulnerabilities early and release improvements quickly. You can get the results of these scans at GitHub — they are stored as artifacts on our CI in the SARIF format (Static Analysis Results Interchange Format).

Continuous integration
The project logos and other graphics are available in https://github.com/WeblateOrg/graphics.

Michal Čihař
michal@cihar.com

Weblate 4.9

Weblate 4.8.1

Released on September 10th 2021.
Fixed user removal in Django admin interface.
Document add-on parameters in more detail.
Fixed JavaScript error on glossary.
Add limit to number of matches in consistency check.
Improve placeholders handling in machine translations.
Fixed creating add-ons using API.
Added PRIVACY_URL setting to add privacy policy link to the footer.
Hide member e-mail addresses from project admins.
Improved gettext PO merging in case of conflicts.
Improved glossary highlighting.
Improved safe-html flag behavior with XML checks.
Fixed commit messages on linked components.
All changes in detail.
Weblate 4.8

Released on August 21th 2021.
Added support for Apple stringsdict format.
The exact search operator is now case-sensitive on PostgreSQL.
Fixed saving glossary explanations in some cases.
Documentation improvements.
Performance improvements.
Improved squash add-on compatibility with Gerrit.
Fixed adding strings to monolingual glossary components.
Improved performance in handling variants.
Fixed squash add-on sometimes skipping parsing upstream changes.
Preserve file extension for downloads.
Added support for the Fluent format.
Added support for using tabs to indent JSON formats.
All changes in detail.

Weblate 4.7.2

Released on July 15th 2021.
Support more language aliases to be configured on a project.
Fixed search string validation in API.
Fixed Git exporter URLs after a domain change.
Fixed cleanup add-on for Windows RC files.
Fixed possible crash in XLIFF updating.
All changes in detail.

Weblate 4.7.1

Released on June 30th 2021.
Improved popup for adding terms to glossary.
Added support for LibreTranslate machine translation service.
Added rate limiting on creating new projects.
Improved performance of file updates.
All changes in detail.

Weblate 4.7

Released on June 17th 2021.
gettext PO object-pascal-format
mi18n lang SAML
Fixed Gerrit integration to better handle corner cases.
Weblate now requires Django 3.2.
Fixed inviting users when e-mail authentication is disabled.
Improved language definitions.
Added support for blocking users from contributing to a project.
Fixed automatic creation of glossary languages.
Extended documentation about add-ons.
Performance improvements for components with linked repositories.
Added support for free DeepL API.
The user management no longer needs Django admin interface.

**Weblate 4.6.2**

Released on May 8th 2021.

RTL

Git

Weblate 4.6.1

Released on May 2nd 2021.

Remove obsolete spam protection code.

Update list of user interface languages in Docker.

Pagure

Weblate 4.6

Released on April 19th 2021.

The auto_translate management command has now a parameter for specifying translation mode.

Added date filtering when browsing changes.

Improved activity charts.

Sender for contact form e-mails can now be configured.

API

The rate limiting no longer applies to superusers.

Docker
API for creating components now automatically uses Weblate URL.
Simplified state indication while listing strings.
Renamed Argon2 to clarify the purpose.
Initial support for Scaling horizontally the Docker deployment.

**Weblate 4.5.3**

Released on April 1st 2021.

- Fixed possible loss of newly added strings on replace upload.

**Weblate 4.5.2**

Released on March 26th 2021.

- Lua
- Ignore format strings in the check.
- Allow uploading screenshot from a translate page.
- Added forced file synchronization to the repository maintenance.

**Weblate 4.5.1**

Released on March 5th 2021.

- Fixed editing of glossary flags in some corner cases.
- Extend metrics usage to improve performance of several pages.
- Improved alerts behavior on glossary components.
- Improved bulk edit performance.
- Fixed preserving "Needs editing" and "Approved" states for ODF files.
**Weblate 4.5**

Released on February 19th 2021.

**gettext PO**

Fixed multiple unnamed variables check behavior with multiple format flags.
Dropped mailing list field on the project in favor of generic instructions for translators.

**TermBase eXchange**

Strings can now be added and removed in bilingual formats as well.

**Amazon Translate**

Java **Java MessageFormat**

Glossaries are now stored as regular components.
Dropped specific API for glossaries as component API is used now.
Added simplified interface to toggle some of the flags.

Moved text direction toggle to get more space for the visual keyboard.
Added check whether translation matches the glossary.

**Weblate 4.4.2**

Released on January 14th 2021.
Fixed corruption of one distributed MO file.

**Weblate 4.4.1**

Released on January 13th 2021.
Fixed displaying help for project settings.

**gettext PO**

Fixed cleanup add-on behavior with HTML, ODF, IDML and Windows RC formats.

**CSV**

Use content compression for file downloads.
Improved user experience on importing from ZIP file.

351
Avoid duplicate pull requests on Pagure.

Reimplemented translation editor to use native browser textarea.

Added API for add-ons.

**Weblate 4.4**

Released on December 15th 2020.

Weblate now requires Django 3.1.

Fixed read-only state handling in bulk edit.

**CodeMirror**

Syntax highlighting in translation editor for XML, HTML, Markdown and reStructuredText.

Improved support for non-standard language codes.

The user is now presented with a filtered list of languages when adding a new translation.

Extended search capabilities for changes in history.

Improved billing detail pages and libre hosting workflow.

**API**

Added tasks API.

Improved display of user defined special characters.

Improved naming of ZIP downloads.

**Weblate 4.3.2**

Released on November 4th 2020.

Fixed crash on certain component filemasks.

Pagure **Markdown**

Simplified setup of Git repositories with different default branch than "master".

 Newly created internal repositories now use main as the default branch.

reStructuredText

Fixed CodeMirror display issues in some situations.
Renamed Template group to "Sources" to clarify its meaning.
Fixed GitLab pull requests on repositories with longer paths.

**Weblate 4.3.1**

Released on October 21st 2020.

- Improve hooks compatibility with Bitbucket Server.
- Reduced memory usage.

**Weblate 4.3**

Released on October 15th 2020.

- Include user stats in the API.
- Fixed component ordering on paginated pages.
- Rewritten support for GitHub and GitLab pull requests.
- Fixed configuration of enforced checks.
- Improved documentation about built-in backups.
- Generic placeholders check now supports regular expressions.
- Added support for interacting with multiple GitLab or GitHub instances.
- Extended API to cover project updates, unit updates and removals and glossaries.
- Unit API now properly handles plural strings.
- Component creation can now handle ZIP file or document upload.
- Consolidated API response status codes.
- Improved JSON, YAML and CSV formats compatibility.
- Improved performance of file downloads.
- Improved repository management view.
- Automatically enable java-format for Android.
- Python 3.9

Fixed translating HTML files under certain conditions.
**Weblate 4.2.2**

Released on September 2nd 2020.

- **JSON**
  - Fixed login redirect for some authentication configurations.
  - Fixed LDAP authentication with group sync.

- **Git**
  - Fixed creating local VCS components using API.

**Weblate 4.2.1**

Released on August 21st 2020.

- **Android**
- **XLIFF**
  - Allow setting up localization CDN in Docker image.

**Weblate 4.2**

Released on August 18th 2020.

- **Improved user pages and added listing of users.**
- **Dropped support for migrating from 3.x releases, migrate through 4.1 or 4.0.**
- **Added exports into several monolingual formats.**
- **Improved activity charts.**
- **Number of displayed nearby strings can be configured.**
- **Simplified main navigation (replaced buttons with icons).**
- **Improved language code handling in Google Translate integration.**
  - The Git squash add-on can generate `Co-authored-by:` trailers.
- **Improved query search parser.**
- **Improved user feedback from format strings checks.**
- **Improved performance of bulk state changes.**
- **Added compatibility redirects after project or component renaming.**
- **Added notifications for strings approval, component locking and license change.**
- **Added support for ModernMT.**
- **Allow to avoid overwriting approved translations on file upload.**
- **Dropped support for some compatibility URL redirects.**

**ECMAScript**

- **Removed leading dot from JSON unit keys.**

**Celery**

- **Allow to configure Content-Security-Policy HTTP headers.**
- **Added support for aliasing languages at project level.**

New add-on to help with HTML or JavaScript localization, see `JavaScript CDN`.

The Weblate domain is now configured in the settings, see `SITE_DOMAIN`.
**Weblate 4.1.1**

Released on June 19th 2020.
Fixed changing autofix or add-ons configuration in Docker.
Fixed possible crash in "About" page.
Fixed adding words to glossary.
Fixed keyboard shortcuts for machinery.
Removed debugging output causing discarding log events in some setups.
Fixed lock indication on project listing.
Fixed listing GPG keys in some setups.
Added option for which DeepL API version to use.
Added support for acting as SAML Service Provider, see SAML.  

**Weblate 4.1**

Released on June 15th 2020.
Added support for creating new translations with included country code.
Added support for searching source strings with screenshot.
Extended info available in the stats insights.
Improved search editing on "Translate" pages.
Improved handling of concurrent repository updates.
Include changes count in credits.
Fixed UI language selection in some cases.
Allow to whitelist registration methods with registrations closed.
Improved lookup of related terms in glossary.
Improved translation memory matches.
Group same machinery results.
Add direct link to edit screenshot from translate page.
Improved removal confirmation dialog.
Include templates in ZIP download.
Add support for Markdown and notification configuration in announcements.
Extended details in check listings.
Consistently use dismissed as state of dismissed checks.
Fixed editor keyboard shortcut to dismiss checks.
Improved machine translation of strings with placeholders.
Show ghost translation for user languages to ease starting them.
Improved language code parsing.
Show translations in user language first in the list.
Added new quality checks: go-i18n JSON files, ARB File.  
Reintroduced support for wiping translation memory.
Fixed option to ignore source checks.
Added support for configuring different branch for pushing changes.

API

Added support for Google Translate V3 API (Advanced).
Added ability to restrict access on component level.
Added support for whitespace and other special chars in translation flags, see [API documentation](https://weblate.readthedocs.io/en/latest/api.html).

API now supports filtering of changes.
Added support for sharing glossaries between projects.

**Weblate 4.0.4**

Released on May 7th 2020.
Fixed testsuite execution on some Python 3.8 environments.
Typo fixes in the documentation.
Fixed creating components using API in some cases.
Fixed JavaScript errors breaking mobile navigation.
Fixed crash on displaying some checks.
Fixed screenshots listing.
Fixed monthly digest notifications.
Fixed intermediate translation behavior with units non existing in translation.

**Weblate 4.0.3**

Released on May 2nd 2020.
Fixed possible crash in reports.
User mentions in comments are now case insensitive.
Fixed PostgreSQL migration for non superusers.
Fixed changing the repository URL while creating component.
Fixed crash when upstream repository is gone.

**Weblate 4.0.2**

Released on April 27th 2020.
Improved performance of translation stats.
Improved performance of changing labels.
Improved bulk edit performance.
Improved translation memory performance.
Fixed possible crash on component deletion.
Improved warning about too long celery queue.
Fixed possible false positives in the consistency check.
Fixed deadlock when changing linked component repository.
Included edit distance in changes listing and CSV and reports.
Avoid false positives of punctuation spacing check for Canadian French.
Fixed XLIFF export with placeholders.
Fixed false positive with zero width check.
Improved reporting of configuration errors.
Fixed bilingual source upload.
Automatically detect supported languages for DeepL machine translation.
Fixed progress bar display in some corner cases.
Fixed some checks triggering on non translated strings.

**Weblate 4.0.1**

Released on April 16th 2020.
Fixed package installation from PyPI.

**Weblate 4.0**

Released on April 16th 2020.
Weblate now requires Python 3.6 or newer.
Added management overview of component alerts.
Added component alert for broken repository browser URLs.
Improved sign in and registration pages.
Project access control and workflow configuration integrated to project settings.
Added check and highlighter for i18next interpolation and nesting.
Added check and highlighter for percent placeholders.

Record source string changes in history.
Upgraded Microsoft Translator to version 3 API.
Reimplemented translation memory backend.
Added support for several is: lookups in **[API]**.
Allow to make avoid internal blacklist.
Improved comments extraction from monolingual po files.
Renamed whiteboard messages to announcements.
Fixed occasional problems with registration mails.
Improved LINGUAS update add-on to handle more syntax variants.
Fixed editing monolingual XLIFF source file.
Added support for exact matching in **[API]**.
Add support for source upload on bilingual translations.
Added support for intermediate language from developers.
Added support for source strings review.
Extended download options for platform wide translation memory.

**Weblate 3.x series**

**Weblate 3.11.3**

Released on March 11th 2020.
Fixed searching for fields with certain priority.
Fixed predefined query for recently added strings.
Fixed searching returning duplicate matches.
Gmail **[API]**
Fixed reverting changes from the history.
Added links to events in digest notifications.
Fixed email for account removal confirmation.
Added support for Slack authentication in Docker container.
Avoid sending notifications for not subscribed languages.
Include Celery queues in performance overview.

Raised bleach dependency to address CVE-2020-6802.
Fixed listing project level changes in history.
Fixed stats invalidation in some corner cases.
Fixed searching for certain string states.
Improved format string checks behavior on missing percent.
Fixed authentication using some third party providers.

**Weblate 3.11.2**

Released on February 22nd 2020.

Fixed some strings wrongly reported as having no words.

**Weblate 3.11.1**

Released on February 20th 2020.
Documented Celery setup changes.
Improved filename validation on component creation.
Fixed minimal versions of some dependencies.
Fixed adding groups with certain Django versions.
Fixed manual pushing to upstream repository.
Improved glossary matching.

**Weblate 3.11**

Released on February 17th 2020.
Allow using VCS push URL during component creation via API.
Fixed links in notifications e-mails.
Improved look of plaintext e-mails.
Display ignored checks and allow to make them active again.

Recommend upgrade to new Weblate versions in the system checks.
Provide more detailed analysis for duplicate language alert.
Include more detailed license info on the project pages.
Automatically unshallow local copies if needed.
Fixed download of strings needing action.
New alert to warn about using the same filemask twice.
Improve XML placeables extraction.

The `SINGLE_PROJECT` can now enforce redirection to chosen project.
Added option to resolve comments.
Added bulk editing of flags.
Added support for labels.

Added option for.

Increased default validity of confirmation links.
Improved Matomo integration.
Fixed to correctly handle source string change.
Extended automatic updates configuration by AUTO_UPDATE.

LINGUAS Weblate 3.10.3

Released on January 18th 2020.
Support for translate-toolkit 2.5.0.

Weblate 3.10.2

Released on January 18th 2020.
Add lock indication to projects.
Fixed CSS bug causing flickering in some web browsers.

Improved repository matching for GitHub and Bitbucket hooks.
Fixed data migration on some Python 2.7 installations.
Allow configuration of Git shallow cloning.
Improved background notification processing.
Fixed broken form submission when navigating back in web browser.
New add-on to configure YAML formatting.
Fixed same plurals check to not fire on single plural form languages.
Fixed regex search on some fields.

Weblate 3.10.1

Released on January 9th 2020.
Extended API with translation creation.
Fixed several corner cases in data migrations.
Compatibility with Django 3.0.

Added support for customizable security.txt.
Improved breadcrumbs in changelog.
Improved translations listing on dashboard.
Improved HTTP responses for webhooks.
Added support for GitLab merge requests in Docker container.
**Weblate 3.10**

Released on December 20th 2019.
Improved application user interface.
Added doublespace check.
Fixed creating new languages.
Avoid sending auditlog notifications to deleted e-mails.

Added support for Markdown in comments.
Allow placing translation instruction text in project info.

Improved support for Mercurial.
Improved Git repository fetching performance.
Add search lookup for age of string.

Show context for nearby strings.
Added support for notifications on repository operations.
Improved translation listings.
Extended search capabilities.
Added support for automatic translation strings marked for editing.
Avoid sending duplicate notifications for linked component alerts.

Improve default merge request message.
Better indicate string state in Zen mode.
Added support for more languages in Yandex Translate.
Improved look of notification e-mails.
Provide choice for translation license.

**Weblate 3.9.1**

Released on October 28th 2019.
Remove some unneeded files from backups.
Fixed potential crash in reports.
Fixed cross database migration failure.
Added support for force pushing Git repositories.
Reduced risk of registration token invalidation.

Added search based on priority.
Fixed possible crash on adding strings to JSON file.
Safe HTML check and fixup now honor source string markup.
Avoid sending notifications to invited and deleted users.
Fix SSL connection to redis in Celery in Docker container.
Weblate 3.9

Released on October 15th 2019.
Include Weblate metadata in downloaded files.
Improved UI for failing checks.
Indicate missing strings in format checks.
Separate check for French punctuation spacing.
Add support for fixing some of quality checks errors.
Add separate permission to create new projects.
Extend stats for char counts.
Improve support for Java style language codes.
Added new generic check for placeholders.
Added support for WebExtension JSON placeholders.
Added support for flat XML format.
Extended API with project, component and translation removal and creation.
Added support for Gitea and Gitee webhooks.
Added new custom regex based check.
Allow to configure contributing to shared translation memory.
Added ZIP download for more translation files.
Make XLIFF standard compliant parsing of maxwidth and font.
Added new check and fixer for safe HTML markup for translating web applications.
Add component alert on unsupported configuration.
Extend automatic translation to add suggestions.
Display add-on parameters on overview.
Sentry is now supported through modern Sentry SDK instead of Raven.
Changed example settings to be better fit for production environment.
Added automated backups using BorgBackup.
Split cleanup add-on for RESX to avoid unwanted file updates.
Advanced search capabilities.
Allow users to download their own reports.
Added localization guide to help configuring components.
Added support for GitLab merge requests.
Improved display of repository status.
Perform automated translation in the background.

Weblate 3.8

Released on August 15th 2019.
Added support for simplified creating of similar components.
Added support for parsing translation flags from the XML based file formats.
Log exceptions into Celery log.
Improved look of notification e-mails.
Fixed password reset behavior.
Improved performance on most of translation pages.
Fixed listing of languages not known to Weblate.
Add support for replacing file content with uploaded.
Add support for translating non VCS based content.
Added OpenGraph widget image to use on social networks.
Added support for animated screenshots.
Improved handling of monolingual XLIFF files.
Avoid sending multiple notifications for single event.
Add support for filtering changes.
Extended predefined periods for reporting.
Added webhook support for Azure Repos.
New opt-in notifications on pending suggestions or untranslated strings.
Add one click unsubscribe link to notification e-mails.
Fixed false positives with Has been translated check.
New management interface for admins.
String priority can now be specified using flags.
Added language management views.
Add checks for Qt library and Ruby format strings.
Added configuration to better fit single project installations.
Notify about new string on source string change on monolingual translations.
Added separate view for translation memory with search capability.

**Weblate 3.7.1**

Released on June 28th 2019.
Documentation updates.
Fixed some requirements constraints.
Updated language database.
Localization updates.
Various user interface tweaks.
Improved handling of unsupported but discovered translation files.
More verbosely report missing file format requirements.

**Weblate 3.7**

Released on June 21st 2019.
Added separate Celery queue for notifications.
Use consistent look with application for API browsing.
Include approved stats in the reports.
Report progress when updating translation component.
Allow to abort running background component update.
Extend template language for filename manipulations.
Use templates for editor link and repository browser URL.
Indicate max length and current characters count when editing translation.
Refreshed landing page for new contributors.
msgmerge
Delay opening SMTP connection when sending notifications.
Improved error logging.
Allow custom location in MO generating add-on.
Added add-ons to cleanup old suggestions or comments.
Added option to enable horizontal mode in the Zen editor.
Improved import performance with many linked components.
Fixed examples installation in some cases.

Added new horizontal stats widget.
Improved format strings check on plurals.
Added font management tool.

Added support for subtitle formats.
Include overall completion stats for languages.
Added reporting at project and global scope.
Improved user interface when showing translation status.
New Weblate logo and color scheme.
New look of bitmap badges.

**Weblate 3.6.1**

Released on April 26th 2019.
Improved handling of monolingual XLIFF files.
Fixed digest notifications in some corner cases.
Fixed add-on script error alert.
Fixed generating MO file for monolingual PO files.
Fixed display of uninstalled checks.
Indicate administered projects on project listing.
Allow update to recover from missing VCS repository.

**Weblate 3.6**

Released on April 20th 2019.
Add support for downloading user data.

Improved instructions for resolving merge conflicts.
Cleanup add-on is now compatible with app store metadata translations.
Configurable language code syntax when adding new translations.
Warn about using Python 2 with planned termination of support in April 2020.
Extract special characters from the source string for visual keyboard.
Extended contributor stats to reflect both source and target counts.
Admins and consistency add-ons can now add translations even if disabled for users.
Fixed description of toggle disabling Language-Team header manipulation.
Notify users mentioned in comments.
Removed file format autodetection from component setup.
Fixed generating MO file for monolingual PO files.
Added digest notifications.
Added support for muting component notifications.
Added notifications for new alerts, whiteboard messages or components.
Notifications for administered projects can now be configured.
Improved handling of three letter language codes.

**Weblate 3.5.1**

Released on March 10th 2019.
Fixed Celery systemd unit example.
Fixed notifications from HTTP repositories with login.
Fixed race condition in editing source string for monolingual translations.
Include output of failed add-on execution in the logs.
Improved race condition in editing source string for monolingual translations.
Allow to edit file format in component settings.
Update installation instructions to prefer Python 3.
Performance and consistency improvements for loading translations.
Microsoft Terminology service Zeep
Localization updates.

**Weblate 3.5**

Released on March 3rd 2019.
Improved performance of built-in translation memory.
Added interface to manage global translation memory.
Improved alerting on bad component state.
Added user interface to manage whiteboard messages.
Add-on commit message now can be configured.
Reduce number of commits when updating upstream repository.
Fixed possible metadata loss when moving component between projects.
Improved navigation in the Zen mode.
Added several new quality checks (Markdown related and URL).
Added support for app store metadata files.
Added support for toggling GitHub or Gerrit integration.
Kashida
Added option to squash commits based on authors.
Improved support for XLSX file format.
Compatibility with Tesseract 4.0.
Billing add-on now removes projects for unpaid billings after 45 days.

**Weblate 3.4**

Released on January 22nd 2019.
Added support for XLIFF placeholders.
Celery can now utilize multiple task queues.
Added support for renaming and moving projects and components.
Include characters counts in reports.
Added guided adding of translation components with automatic detection of translation files.
Customizable merge commit messages for Git.
Added visual indication of component alerts in navigation.
Improved performance of loading translation files.
Changed default merge style to rebase and made that configurable.
Better handle private use subtags in language code.
Improved performance of fulltext index updates.
Extended file upload API to support more parameters.

Weblate 3.3

Released on November 30th 2018.
Added support for component and project removal.
Improved performance for some monolingual translations.
Added translation component alerts to highlight problems with a translation.
Expose XLIFF string resname as context when available.
Added support for XLIFF states.
Added check for non writable files in DATA_DIR.
Improved CSV export for changes.

Weblate 3.2.2

Released on October 20th 2018.
Remove no longer needed Babel dependency.
Updated language definitions.
Improve documentation for add-ons, LDAP and Celery.
Fixed enabling new dos-eol and auto-java-messageformat flags.
Fixed running setup.py test from PyPI package.
Improved plurals handling.
Fixed translation upload API failure in some corner cases.
Fixed updating Git configuration in case it was changed manually.

Weblate 3.2.1

Released on October 10th 2018.
Document dependency on backports.csv on Python 2.7.
Fix running tests under root.
Improved error handling in gitexport module.
Fixed progress reporting for newly added languages.
Correctly report Celery worker errors to Sentry.
Fixed creating new translations with Qt Linguist.
Fixed occasional fulltext index update failures.
Improved validation when creating new components.
Added support for cleanup of old suggestions.
Weblate 3.2

Released on October 6th 2018.
Add install_addon management command for automated add-on installation.

- Added support for export and import of Excel files.
- Improve component cleanup in case of multiple component discovery add-ons.
- Microsoft Terminology
- Weblate now uses Celery to offload some processing.
- Improved search capabilities and added regular expression search.
- Added support for Youdao Zhiyun API machine translation.
- Added support for Baidu API machine translation.
- Integrated maintenance and cleanup tasks using Celery.
- Improved performance of loading translations by almost 25%.
- Removed support for merging headers on upload.
- Removed support for custom commit messages.
- Configurable editing mode (zen/full).
- Added support for error reporting to Sentry.
- Added support for automated daily update of repositories.
- Added support for creating projects and components by users.

- Users and projects can import their existing translation memories.
- Better management of related strings for screenshots.
- Added support for checking Java MessageFormat.

See 3.2 milestone on GitHub for detailed list of addressed issues.

Weblate 3.1.1

Released on July 27th 2018.
Fix testsuite failure on some setups.

Weblate 3.1

Released on July 27th 2018.
Upgrades from older version than 3.0.1 are not supported.
Allow to override default commit messages from settings.
Improve webhooks compatibility with self hosted environments.
Added support for Amazon Translate.
Compatibility with Django 2.1.
Django system checks are now used to diagnose problems with installation.
Removed support for soon shutdown libravatar service.

- Add support for jumping to specific location while translating.
- Downloaded translations can now be customized.
- Improved calculation of string similarity in translation memory matches.
- Added support by signing Git commits by GnuPG.
**Weblate 3.0.1**

Released on June 10th 2018.
Fixed possible migration issue from 2.20.
Localization updates.
Removed obsolete hook examples.
Improved caching documentation.
Fixed displaying of admin documentation.
Improved handling of long language names.

**Weblate 3.0**

Released on June 1st 2018.
Rewritten access control.
Several code cleanups that lead to moved and renamed modules.

The import_project management command has now slightly different parameters.
Added basic support for Windows RC files.
New add-on to store contributor names in PO file headers.
The per component hook scripts are removed, use add-ons instead.
Add support for collecting contributor agreements.
Access control changes are now tracked in history.

Support for more variables in commit message templates.
Add support for providing additional textual context.

**Weblate 2.x series**

**Weblate 2.20**

Released on April 4th 2018.
Improved speed of cloning subversion repositories.
Changed repository locking to use third party library.
Added support for downloading only strings needing action.
Added support for searching in several languages at once.
New add-on to configure gettext output wrapping.
New add-on to configure JSON formatting.
Added support for authentication in API using RFC 6750 compatible Bearer authentication.
Added support for automatic translation using machine translation services.
Added support for HTML markup in whiteboard messages.
Added support for mass changing state of strings.
Translate-toolkit at least 2.3.0 is now required, older versions are no longer supported.

Added support for DeepL machine translation service.
Machine translation results are now cached inside Weblate.
Weblate 2.19.1

Released on February 20th 2018.
Fixed migration issue on upgrade from 2.18.
Improved file upload API validation.

Weblate 2.19

Released on February 15th 2018.
Fixed imports across some file formats.
Display human friendly browser information in audit log.
Added TMX exporter for files.
Various performance improvements for loading translation files.
Added option to disable access management in Weblate in favor of Django one.
Improved glossary lookup speed for large strings.
Compatibility with django_auth_ldap 1.3.0.
Configuration errors are now stored and reported persistently.
Honor ignore flags in whitespace autofixer.
Improved compatibility with some Subversion setups.

Added support for SAP Translation Hub service.
Microsoft Terminology service
Removed support for advertisement in notification e-mails.
Improved translation progress reporting at language level.
Improved support for different plural formulas.
Added support for Subversion repositories not using stdlayout.

Weblate 2.18

Released on December 15th 2017.
Extended contributor stats.
Improved configuration of special characters virtual keyboard.
Added support for DTD file format.
Changed keyboard shortcuts to less likely collide with browser/system ones.
Improved support for approved flag in XLIFF files.
Added support for not wrapping long strings in gettext PO files.
Added button to copy permalink for current translation.
Dropped support for Django 1.10 and added support for Django 2.0.
Removed locking of translations while translating.
Added support for adding new strings to monolingual translations.
Added support for translation workflows with dedicated reviewers.
Weblate 2.17.1

Released on October 13th 2017.
Fixed running testsuite in some specific situations.

Weblate 2.17

Released on October 13th 2017.
Weblate by default does shallow Git clones now.
Improved performance when updating large translation files.
Added support for blocking certain e-mails from registration.
Users can now delete their own comments.
Added preview step to search and replace feature.
Client side persistence of settings in search and upload forms.
Extended search capabilities.
More fine grained per project ACL configuration.
Default value of BASE_DIR has been changed.
Added two step account removal to prevent accidental removal.
Project access control settings is now editable.
Added optional spam protection for suggestions using Akismet.

Weblate 2.16

Released on August 11th 2017.
Various performance improvements.
Added support for nested JSON format.
Added support for WebExtension JSON format.
Fixed git exporter authentication.
Improved CSV import in certain situations.
Improved look of Other translations widget.
The max-length checks is now enforcing length of text in form.
Make the commit_pending age configurable per component.
Various user interface cleanups.
Fixed component/project/site wide search for translations.

Weblate 2.15

Released on June 30th 2017.
Show more related translations in other translations.
Add option to see translations of current string to other languages.
Use 4 plural forms for Lithuanian by default.
Fixed upload for monolingual files of different format.
Improved error messages on failed authentication.
Keep page state when removing word from glossary.

Added Perl format quality check.
Added support for rejecting reused passwords.

369
Extended toolbar for editing RTL languages.

**Weblate 2.14.1**

Released on May 24th 2017.
Fixed possible error when paginating search results.
Fixed migrations from older versions in some corner cases.
Fixed possible CSRF on project watch and unwatch.
The password reset no longer authenticates user.
Fixed possible CAPTCHA bypass on forgotten password.

**Weblate 2.14**

Released on May 17th 2017.
Add glossary entries using AJAX.
The logout now uses POST to avoid CSRF.
The API key token reset now uses POST to avoid CSRF.
Weblate sets Content-Security-Policy by default.
The local editor URL is validated to avoid self-XSS.
The password is now validated against common flaws by default.
Notify users about important activity with their account such as password change.
The CSV exports now escape potential formulas.
Various minor improvements in security.

Suggestion content is stored in the history.
Store important account activity in audit log.
Ask for password confirmation when removing account or adding new associations.
Show time when suggestion has been made.
There is new quality check for trailing semicolon.
Ensure that search links can be shared.
Included source string information and screenshots in the API.
Allow to overwrite translations through API upload.

**Weblate 2.13.1**

Released on Apr 12th 2017.
Fixed listing of managed projects in profile.
Fixed migration issue where some permissions were missing.
Fixed listing of current file format in translation download.
Return HTTP 404 when trying to access project where user lacks privileges.
**Weblate 2.13**

Released on Apr 12th 2017.
Fixed quality checks on translation templates.
Added quality check to trigger on losing translation.
Add option to view pending suggestions from user.

Default dashboard for unauthenticated users can be configured.
Add option to browse 25 random strings for review.
History now indicates string change.
Better error reporting when adding new translation.
Added per language search within project.
Group ACLs can now be limited to certain permissions.
The per project ACLs are now implemented using Group ACL.
Added more fine grained privileges control.
Various minor UI improvements.

**Weblate 2.12**

Released on Mar 3rd 2017.
Improved admin interface for groups.
Added support for Yandex Translate API.
Improved speed of site wide search.
Added project and component wide search.
Added project and component wide search and replace.

Added support for opening source files in local editor.
Added support for configuring visual keyboard with special characters.
Improved screenshot management with OCR support for matching source strings.
Default commit message now includes translation information and URL.
Added support for Joomla translation format.
Improved reliability of import across file formats.

**Weblate 2.11**

Released on Jan 31st 2017.
Include language detailed information on language page.
Mercurial backend improvements.
Added option to specify translation component priority.
More consistent usage of Group ACL even with less used permissions.
Added WL_BRANCH variable to hook scripts.
Improved developer documentation.
Better compatibility with various Git versions in Git exporter add-on.
Included per project and component stats.

Added language code mapping for better support of Microsoft Translate API.
Moved fulltext cleanup to background job to make translation removal faster.
Fixed displaying of plural source for languages with single plural form.
Improved error handling in import_project.
Various performance improvements.

**Weblate 2.10.1**

Released on Jan 20th 2017.
Do not leak account existence on password reset form (CVE-2017-5537).

**Weblate 2.10**

Released on Dec 15th 2016.
Added quality check to check whether plurals are translated differently.
Fixed GitHub hooks for repositories with authentication.
Added optional Git exporter module.
Support for Microsoft Cognitive Services Translator API.
Simplified project and component user interface.
Added automatic fix to remove control characters.
Added per language overview to project.
Added support for CSV export.
Added CSV download for stats.
Added matrix view for quick overview of all translations.
Added basic API for changes and strings.
Added support for Apertium APy server for machine translations.

**Weblate 2.9**

Released on Nov 4th 2016.
Extended parameters for createadmin management command.
Extended import_json to be able to handle with existing components.
Added support for YAML files.
Project owners can now configure translation component and project details.
Use "Watched" instead of "Subscribed" projects.
Projects can be watched directly from project page.
Added multi language status widget.
Record suggestion deletion in history.
Improved UX of languages selection in profile.
Fixed showing whiteboard messages for component.
Show source string comment more prominently.
Automatically install Gettext PO merge driver for Git repositories.
Added search and replace feature.
Added support for uploading visual context (screenshots) for translations.
**Weblate 2.8**

Released on Aug 31st 2016.
Documentation improvements.
Translations.
Updated bundled javascript libraries.
Added list_translators management command.
Django 1.8 is no longer supported.
Fixed compatibility with Django 1.10.
Added Subversion support.
Separated XML validity check from XML mismatched tags.
Fixed API to honor HIDE_REPO_CREDENTIALS settings.
Show source change in Zen mode.
Alt+PageUp/PageDown/Home/End now works in Zen mode as well.
Add tooltip showing exact time of changes.
Add option to select filters and search from translation page.
Added UI for translation removal.
Improved behavior when inserting placeables.
Fixed auto locking issues in Zen mode.

**Weblate 2.7**

Released on Jul 10th 2016.
Removed Google web translate machine translation.
Improved commit message when adding translation.
Fixed Google Translate API for Hebrew language.
Compatibility with Mercurial 3.8.
Added import_json management command.
Correct ordering of listed translations.
Show full suggestion text, not only a diff.
Extend API (detailed repository status, statistics, …).
Testsuite no longer requires network access to test repositories.

**Weblate 2.6**

Released on Apr 28th 2016.
Fixed validation of components with language filter.
Improved support for XLIFF files.
Fixed machine translation for non English sources.
Added REST API.
Django 1.10 compatibility.
Added categories to whiteboard messages.
Weblate 2.5

Released on Mar 10th 2016.
Fixed automatic translation for project owners.
Improved performance of commit and push operations.
New management command to add suggestions from command line.
Added support for merging comments on file upload.
Added support for some GNU extensions to C printf format.
Documentation improvements.
Added support for generating translator credits.
Added support for generating contributor stats.
Site wide search can search only in one language.
Improve quality checks for Armenian.
Support for starting translation components without existing translations.
Support for adding new translations in Qt TS.
Improved support for translating PHP files.
Performance improvements for quality checks.
Improved support for XLIFF files.
Extended list of options for import_project.
Improved targeting for whiteboard messages.
Support for automatic translation across projects.
Optimized fulltext search index.
Added management command for auto translation.
Added placeables highlighting.
Improved keyboard shortcuts for placeables, checks and machine translations.
Improved translation locking.
Added quality check for AngularJS interpolation.
Added extensive group based ACLs.
Support for Python 3.
Dropped support for Django 1.7.
Dropped dependency on msginit for creating new gettext PO files.
Added configurable dashboard views.
Improved notifications on parse errors.
Added option to import components with duplicate name to import_project.
Improved support for translating PHP files.
Added XLIFF export for dictionary.
Added XLIFF and gettext PO export for all translations.
Documentation improvements.
Added support for configurable automatic group assignments.
Improved adding of new translations.
**Weblate 2.4**

Released on Sep 20th 2015.
Improved support for PHP files.
Ability to add ACL to anonymous user.
Improved configurability of import_project command.
Added CSV dump of history.
Avoid copy/paste errors with whitespace characters.
Added support for Bitbucket webhooks.
Tighter control on fuzzy strings on translation upload.
Several URLs have changed, you might have to update your bookmarks.
Hook scripts are executed with VCS root as current directory.
Hook scripts are executed with environment variables describing current component.
Add management command to optimize fulltext index.
Added support for error reporting to Rollbar.
Projects now can have multiple owners.
Project owners can manage themselves.
Added support for javascript-format used in gettext PO.
Support for adding new translations in XLIFF.
Improved file format autodetection.
Extended keyboard shortcuts.
Improved dictionary matching for several languages.
Improved layout of most of pages.
Support for adding words to dictionary while translating.
Added support for filtering languages to be managed by Weblate.
Added support for translating and importing CSV files.
Rewritten handling of static files.
Direct login/registration links to third-party service if that’s the only one.
Commit pending changes on account removal.
Add management command to change site name.
Add option to configure default committer.
Add hook after adding new translation.
Add option to specify multiple files to add to commit.

**Weblate 2.3**

Released on May 22nd 2015.
Dropped support for Django 1.6 and South migrations.
Support for adding new translations when using Java Property files.
Allow to accept suggestion without editing.
Improved support for Google OAuth 2.0.
Added support for Microsoft .resx files.
Tuned default robots.txt to disallow big crawling of translations.
Simplified workflow for accepting suggestions.
Added project owners who always receive important notifications.
Allow to disable editing of monolingual template.
More detailed repository status view.
Direct link for editing template when changing translation.
Allow to add more permissions to project owners.

Weblate 2.2

Released on Feb 19th 2015.
Performance improvements.
Fulltext search on location and comments fields.
New SVG/javascript based activity charts.
Support for Django 1.8.
Support for deleting comments.
Added own SVG badge.
Added support for Google Analytics.
Improved handling of translation filenames.
Added support for monolingual JSON translations.
Record component locking in a history.
Support for editing source (template) language for monolingual translations.
Added basic support for Gerrit.

Weblate 2.1

Released on Dec 5th 2014.
Added support for Mercurial repositories.
Replaced Glyphicon font by Awesome.
Added icons for social authentication services.
Better consistency of button colors and icons.
Documentation improvements.
Various bugfixes.
Automatic hiding of columns in translation listing for small screens.
Changed configuration of filesystem paths.
Improved SSH keys handling and storage.
Improved repository locking.
Customizable quality checks per source string.
Allow to hide completed translations from dashboard.

Weblate 2.0

Released on Nov 6th 2014.
New responsive UI using Bootstrap.
Rewritten VCS backend.
Documentation improvements.
Added whiteboard for site wide messages.
Configurable strings priority.
Added support for JSON file format.
Fixed generating mo files in certain cases.
Added support for GitLab notifications.
Added support for disabling translation suggestions.
Django 1.7 support.
ACL projects now have user management.
Extended search possibilities.
Give more hints to translators about plurals.
Fixed Git repository locking.
Compatibility with older Git versions.
Improved ACL support.
Added buttons for per language quotes and other special characters.
Support for exporting stats as JSONP.

**Weblate 1.x series**

**Weblate 1.9**

Released on May 6th 2014.
Django 1.6 compatibility.
No longer maintained compatibility with Django 1.4.
Management commands for locking/unlocking translations.
Improved support for Qt TS files.
Users can now delete their account.
Avatars can be disabled.
Merged first and last name attributes.
Avatars are now fetched and cached server side.
Added support for shields.io badge.

**Weblate 1.8**

Released on November 7th 2013.
Please check manual for upgrade instructions.
Nicer listing of project summary.
Better visible options for sharing.
More control over anonymous users privileges.
Supports login using third party services, check manual for more details.
Users can login by e-mail instead of username.
Documentation improvements.
Improved source strings review.
Searching across all strings.
Better tracking of source strings.
Captcha protection for registration.
Weblate 1.7

Released on October 7th 2013.
Please check manual for upgrade instructions.
Support for checking Python brace format string.
Per component customization of quality checks.
Detailed per translation stats.
Changed way of linking suggestions, checks and comments to strings.
Users can now add text to commit message.
Support for subscribing on new language requests.
Support for adding new translations.
Pango + Cairo Pillow
Add status badge widget.
Changes in dictionary are now logged in history.
Performance improvements for translating view.

Weblate 1.6

Released on July 25th 2013.
Nicer error handling on registration.
Browsing of changes.
Fixed sorting of machine translation suggestions.
Improved support for MyMemory machine translation.
Added support for Amagama machine translation.
Various optimizations on frequently used pages.
Highlights searched phrase in search results.
Support for automatic fixups while saving the message.
Tracking of translation history and option to revert it.
Added support for Google Translate API.
Added support for managing SSH host keys.
Various form validation improvements.
Various quality checks improvements.
Performance improvements for import.
Added support for voting on suggestions.
Cleanup of admin interface.

Weblate 1.5

Released on April 16th 2013.
Please check manual for upgrade instructions.
Added public user pages.
Better naming of plural forms.
Added support for TBX export of glossary.
Added support for Bitbucket notifications.
Activity charts are now available for each translation, language or user.
Extended options of import_project admin command.
Compatible with Django 1.5.
Avatars are now shown using libravatar.

Added possibility to pretty print JSON export.

Various performance improvements.

Indicate failing checks or fuzzy strings in progress bars for projects or languages as well.

Added support for custom pre-commit hooks and committing additional files.

Rewritten search for better performance and user experience.

New interface for machine translations.

Added support for monolingual po files.

Extended amount of cached metadata to improve speed of various searches.

Now shows word counts as well.

**Weblate 1.4**

Released on January 23rd 2013.

Fixed deleting of checks/comments on string deletion.

Added option to disable automatic propagation of translations.

Added option to subscribe for merge failures.

Correctly import on projects which needs custom ttkit loader.

Added sitemaps to allow easier access by crawlers.

Provide direct links to string in notification e-mails or feeds.

Various improvements to admin interface.

Provide hints for production setup in admin interface.

Added per language widgets and engage page.

Improved translation locking handling.

Indicate failing checks or fuzzy strings in progress bars.

More options for formatting commit message.

Fixed error handling with machine translation services.

Improved automatic translation locking behaviour.

Support for showing changes from previous source string.

Added support for substring search.

Various quality checks improvements.

Support for per project ACL.

Basic code coverage by unit tests.

**Weblate 1.3**

Released on November 16th 2012.

Compatibility with PostgreSQL database backend.

Removes languages removed in upstream git repository.

Improved quality checks processing.

Added new checks (BB code, XML markup and newlines).

Support for optional rebasing instead of merge.

Possibility to relocate Weblate (for example to run it under /weblate path).

Support for manually choosing file type in case autodetection fails.

Better support for Android resources.

Support for generating SSH key from web interface.

More visible data exports.
New buttons to enter some special characters.
Support for exporting dictionary.
Support for locking down whole Weblate installation.
Checks for source strings and support for source strings review.
Support for user comments for both translations and source strings.
Better changes log tracking.
Changes can now be monitored using RSS.
Improved support for RTL languages.

**Weblate 1.2**

Released on August 14th 2012.
Weblate now uses South for database migration, please check upgrade instructions if you are upgrading.
Fixed minor issues with linked git repos.
New introduction page for engaging people with translating using Weblate.
Added widgets which can be used for promoting translation projects.
Added option to reset repository to origin (for privileged users).
Project or component can now be locked for translations.
Possibility to disable some translations.
Configurable options for adding new translations.
Configuration of git commits per project.
Simple antispam protection.
Better layout of main page.
Support for automatically pushing changes on every commit.
Support for e-mail notifications of translators.

Improved handling of not known languages when importing project.
Support for locking translation by translator.
Optionally maintain Language-Team header in po file.
Include some statistics in about page.
Supports (and requires) django-registration 0.8.
Checking of requirements during setup.
Documentation improvements.

**Weblate 1.1**

Released on July 4th 2012.
Improved several translations.
Better validation while creating component.
Added support for shared git repositories across components.
Do not necessary commit on every attempt to pull remote repo.
Added support for offloading indexing.
**Weblate 1.0**

Released on May 10th 2012.
Improved validation while adding/saving component.
Experimental support for Android component files (needs patched ttkit).
Updates from hooks are run in background.
Improved installation instructions.
Improved navigation in dictionary.

**Weblate 0.x series**

**Weblate 0.9**

Released on April 18th 2012.
Fixed import of unknown languages.
Improved listing of nearby messages.
Improved several checks.
Documentation updates.
Added definition for several more languages.
Various code cleanups.
Documentation improvements.
Changed file layout.
Update helper scripts to Django 1.4.
Improved navigation while translating.
Better handling of po file renames.
Better validation while creating component.
Integrated full setup into syncdb.
Added list of recent changes to all translation pages.
Check for not translated strings ignores format string only messages.

**Weblate 0.8**

Released on April 3rd 2012.
Replaced own full text search with Whoosh.
Various fixes and improvements to checks.
New command updatechecks.
Lot of translation updates.
Added dictionary for storing most frequently used terms.
Added /admin/report/ for overview of repositories status.
Machine translation services no longer block page loading.
Management interface now contains also useful actions to update data.
Records log of changes made by users.
Ability to postpone commit to Git to generate less commits from single user.
Possibility to browse failing checks.
Automatic translation using already translated strings.
New about page showing used versions.
Django 1.4 compatibility.
Ability to push changes to remote repo from web interface.
Added review of translations done by others.

**Weblate 0.7**

Released on February 16th 2012.
Direct support for GitHub notifications.
Added support for cleaning up orphaned checks and translations.
Displays nearby strings while translating.
Displays similar strings while translating.
Improved searching for string.

**Weblate 0.6**

Released on February 14th 2012.
Added various checks for translated messages.
Tunable access control.
Improved handling of translations with new lines.
Added client side sorting of tables.
Please check upgrading instructions in case you are upgrading.

**Weblate 0.5**

Released on February 12th 2012.

Support for machine translation using following online services:
- Apertium
- Microsoft Translator
- MyMemory

Several new translations.
Improved merging of upstream changes.
Better handle concurrent git pull and translation.
Propagating works for fuzzy changes as well.
Propagating works also for file upload.
Fixed file downloads while using FastCGI (and possibly others).

**Weblate 0.4**

Released on February 8th 2012.
Adder usage guide to documentation.
Fixed API hooks not to require CSRF protection.
**Weblate 0.3**

Released on February 8th 2012.
Better display of source for plural translations.
New documentation in Sphinx format.

Improved error page to give list of existing projects.
New per language stats.

**Weblate 0.2**

Released on February 7th 2012.
Improved validation of several forms.
Warn users on profile upgrade.
Remember URL for login.
Naming of text areas while entering plural forms.
Automatic expanding of translation area.

**Weblate 0.1**

Released on February 6th 2012.
Initial release.
w
wlc.??
wlc.config.??
wlc.main.??
ANY /.*?/api
GET /api/.*?/addons
GET /api/addons/.*?
GET /api/addons/(int:id)/.*?
PUT /api/addons/(int:id)/.*?
DELETE /api/addons/(int:id)/.*?
PATCH /api/addons/(int:id)/.*? /api/changes
GET /api/changes/.*?
GET /api/changes/(int:id)/.*? /api/component-lists
GET /api/component-lists/.*?
GET /api/component-lists/(str:slug)/.*?
POST /api/component-lists/(str:slug)/components/.*?
PUT /api/component-lists/(str:slug)/.*?
DELETE /api/component-lists/(str:slug)/.*?
DELETE /api/component-lists/(str:slug)/components/(str:component_slug)/.*?
PATCH /api/component-lists/(str:slug)/.*? /api/components
GET /api/components/.*?
GET /api/components/(string:project)/(string:component)/.*?
GET /api/components/(string:project)/(string:component)/changes/.*?
GET /api/components/(string:project)/(string:component)/links/.*?
GET /api/components/(string:project)/(string:component)/lock/.*?
GET /api/components/(string:project)/(string:component)/monolingual_base/.*?
GET /api/components/(string:project)/(string:component)/new_template/.*?
GET /api/components/(string:project)/(string:component)/repository/.*?
GET /api/components/(string:project)/(string:component)/screenshots/.*?
GET /api/components/(string:project)/(string:component)/statistics/.*?
GET /api/components/(string:project)/(string:component)/translations/.*?
POST /api/components/(string:project)/(string:component)/addons/.*?
POST /api/components/(string:project)/(string:component)/links/.*?
POST /api/components/(string:project)/(string:component)/lock/.*?
POST /api/components/(string:project)/(string:component)/repository/.*?
POST /api/components/(string:project)/(string:component)/translations/.*?
PUT /api/components/(string:project)/(string:component)/.*?
DELETE /api/components/(string:project)/(string:component)/.*?
DELETE /api/components/(string:project)/(string:component)/links/(string:project_slug)/.*?
PATCH /api/components/(string:project)/(string:component)/.*? /api/groups
GET /api/groups/.*?
GET /api/groups/(int:id)/.*?
POST /api/groups/(int:id)/.*?
POST /api/groups/(int:id)/componentlists/.*?
POST /api/groups/(int:id)/components/.*?
POST /api/groups/(int:id)/languages/.*?
POST /api/groups/(int:id)/projects/.*?
POST /api/groups/(int:id)/roles/.*?
GET /api/tasks/(str:uuid)/, ?? /api/translations
GET /api/translations/, ??
GET /api/translations/(string:project)/(string:component)/(string:language)/, ??
GET /api/translations/(string:project)/(string:component)/(string:language)/changes/, ??
GET /api/translations/(string:project)/(string:component)/(string:language)/file/, ??
GET /api/translations/(string:project)/(string:component)/(string:language)/repository/, ??
GET /api/translations/(string:project)/(string:component)/(string:language)/statistics/, ??
GET /api/translations/(string:project)/(string:component)/(string:language)/units/, ??
POST /api/translations/(string:project)/(string:component)/(string:language)/autotranslate/, ??
POST /api/translations/(string:project)/(string:component)/(string:language)/file/, ??
POST /api/translations/(string:project)/(string:component)/(string:language)/repository/, ??
POST /api/translations/(string:project)/(string:component)/(string:language)/units/, ??
DELETE /api/translations/(string:project)/(string:component)/(string:language)/, ?? /api/units
GET /api/units/, ??
GET /api/units/(int:id)/, ??
PUT /api/units/(int:id)/, ??
DELETE /api/units/(int:id)/, ??
PATCH /api/units/(int:id)/, ?? /api/users
GET /api/users/, ??
GET /api/users/(str:username)/, ??
GET /api/users/(str:username)/notifications/, ??
GET /api/users/(str:username)/notifications/(int:subscription_id)/, ??
GET /api/users/(str:username)/statistics/, ??
POST /api/users/, ??
POST /api/users/(str:username)/groups/, ??
POST /api/users/(str:username)/notifications/, ??
PUT /api/users/(str:username)/, ??
PUT /api/users/(str:username)/notifications/(int:subscription_id)/, ??
DELETE /api/users/(str:username)/, ??
DELETE /api/users/(str:username)/notifications/(int:subscription_id)/, ??
PATCH /api/users/(str:username)/, ??
PATCH /api/users/(str:username)/notifications/(int:subscription_id)/, ?? /exports
GET /exports/rss/, ??
GET /exports/rss/(string:project)/, ??
GET /exports/rss/(string:project)/(string:component)/, ??
GET /exports/rss/(string:project)/(string:component)/(string:language)/, ??
GET /exports/rss/language/(string:language)/, ??
GET /exports/stats/(string:project)/(string:component)/,?? /hooks
GET /hooks/update/(string:project)/,??
GET /hooks/update/(string:project)/(string:component)/,??
POST /hooks/azure/,??
POST /hooks/bitbucket/,??
POST /hooks/gitea/,??
POST /hooks/gitee/,??
POST /hooks/github/,??
POST /hooks/gitlab/,??
POST /hooks/pagure/,??