Transtats Documentation

Release 0.1.5

Sundeep Anand

Jun 22, 2018

Table of Contents

In a software release cycle, localisation steps like extracting or updating language resource, pushing them to translation platform, pulling and packaging translations, quality checks etc. seem necessary but sometimes they lack attention which results in delay. Transtats is an attempt to tie up loose ends and then, may be automate some of the steps. Concept is based on syncing with Upstream Repository, Translation Platform and Build System for statistics and translation resources, comparing stats based on mapping, calculating translation differences, keep upstream updated and create notifications.

CHAPTER 1

Stack

Transtats Server is a simple django application with PostgreSQL backend. Has one CLI and some ansible playbooks for deployment.

1.1 Transtats Server

1.1.1 Summary and Details

1. **Translation Status - Releases** Translation workload estimation for a release branch across packages. It has three views: combined, detailed and language-wise. Combined will sum up translated and untranslated for all languages for a package. Detailed contains language wise %age representation. Language-wise shows translated and untranslated stats of a package for each language.
2. **Translation Status - Packages** Translation progress of a package for sync'd branches in all enabled languages. Package needs to get sync'd with Upstream Repository, Translation Platform and Build System to fill statistics. Stats are represented in tabular and graph forms, per language also. This shows translation treads of last few releases. Summary highlights out-of-sync packages.
3. **Translation Coverage** Coverage of a group of packages for a specific release in associated or selected languages. These are based on graph rules. Packages must have branch mapping and respective sync done before they can participate in graph rule.

1.1.2 Configuration

1. **Inventory** Languages & their sets, translation platforms and release streams are grouped as inventory. One release stream can have multiple release branches. Like, Fedora is a release stream and Fedora 28, Fedora 29 are release branches. Inventory could be managed through admin panel. Demo data contains some of them. Inventory are basis to release branches, packages, jobs and graph rules. And hence probably the first thing to look at.
2. **Release Branch** A particular release which has a schedule and information regarding *in how many languages it will be available*. Release branch are primary to branch mapping. A package associated with any of the

release streams, automatically being tracked for all release branches underneath. One release branch can be associated with one language set.

3. **Packages** Translation progress would be tracked for added packages. They should have upstream repository URL and translation platform project URL. A package can be linked with multiple release streams and should have a branch mapping. Once all versions / tags are sync'd with respective sources, differences could be generated. Packages should be sync'd at intervals.

1.1.3 Jobs

1. **Sync Jobs**

   Functions which are planned to be automated like

   - sync with translation platform for translation stats
   - sync with upstream repositories for translation resources
   - sync with release schedule to update calendar
   - **sync with build system for build tags** *required for branch mapping (this is one of the first steps)*

   Logs are kept.

2. **YAML Based Jobs** Currently, this is to talk with build system. The YAML template can be used as-is.

   This requires three values:

   - Package Name (*derived stats can be saved only if this is added already*)
   - Build System (*build system associated with added release stream*)
   - Build Tag (*To fill dropdown with build tags, run sync job for build tags*)

   Once started it locates latest build, get SRPM and extract, apply patches, filter translation resources and calculates stats. Job could be run for any tag. Dry runs are also supported.

1.2 Transtats APIs

1. **Ping Server** : `<transtats_server>/api/ping`

   Returns server version.

   ```
   GET /api/ping HTTP/1.1
   ```

2. **Package Status** : `<transtats_server>/api/package/<package_name>`

   Returns translation stats of package for enabled languages, for example `abrt`.

   ```
   GET /api/package/abrt HTTP/1.1
   ```

3. **Graph Rule Coverage** : `<transtats_server>/api/coverage/<graph_rule_name>`

   Returns translation coverage according to graph rule, for example `rhinstaller`.

   ```
   GET /api/coverage/rhinstaller HTTP/1.1
   ```

4. **Release Status** : `<transtats_server>/api/release/<release_branch_name>`

   Returns translation stats of packages which are being tracked for a given release, for example `fedora-27`.

```
GET /api/release/fedora-29 HTTP/1.1
```

(a) **Release Status Detail** : `<transtats_server>/api/release/<release_branch_name>/detail`

Returns per language translation stats of packages for a release.

```
GET /api/release/fedora-29/detail HTTP/1.1
```

(b) **Release Status Locale** : `<transtats_server>/api/release/<release_branch_name>/locale/<locale>`

Returns translation stats of packages for a release of a single language.

```
GET /api/release/fedora-29/locale/ja_JP HTTP/1.1
```

1.3 Transtats CLI

transtats-cli is a command line interface to query transtats server.

- **Usage**

```
$ transtats [OPTIONS] COMMAND [ARGS]...
```

- **Options**

--help Show help message and exit.

- **Commands**

1. **coverage** Translation coverage as per graph rule.
2. **package** Translation status of a package.
3. **version** Display the current version.
4. **release** Translation status of a release.

1.4 Getting Started

1.4.1 Check Inventory

Graphs will be generated for enabled languages, aliases are used while syncing. One can create a language set, which can be associated with a release branch. Multiple instances of a translation platform can be added. A release branch (for example - Fedora 28) should have a language set and a schedule. Transtats jobs sync with upstream repository, translation platform and build system to keep stats, build tags and schedule latest.

1.4.2 Add and Configure Packages

While adding a package, upstream URL is required. And package name is verified with selected translation platform. Translation of a package can be tracked for multiple release streams. Package should be sync'd with translation platform and build system. Once sync'd, branch mapping can be created. It maps Transtats release branches with most suitable project versions available at translation platform and with appropriate build tags. Package can be sync'd with upstream repo as well.

1.4.3 Generate Diff

Once we have all versions / tags mentioned in branch mapping of a package sync'd with either translation platform or build system, differences can be created and observed. This answers - for a package - latest translations are packaged or not? If not, is some patch applied at the last moment? Which languages need attention?

1.4.4 Add Graph Rule

Translation coverages (translation status of a group of packages in a set of languages to a given release) are based on rules. Slug form of rule name would be saved. This should be specific for a release branch. Packages having branch mapping created can only be included here. Languages could be picked from language set associated with the release branch or from enabled ones. Somehow if a package is not tracked for a release stream and selected for inclusion Transtats would show an error.

1.4.5 Translation Status

Summary and Details Transtats has two traversal options: releases and packages. One can see high level summary and can pick any one release or a package for details. Summary can tell you - packages which are out of sync, translation workload estimate for each language per release, and much more. Detailed views contain language wise stats, translation position etc. Another form is coverage, which depends on graph rules and branch mapping.

1.5 Get Involved

1.5.1 Try and test

- **Docker**

  Get docker daemon running. Build or pull transtats image and get started.

  - Build the image *(optional)*

    * **Clone the repo and build the image**

      ```
      $ git clone https://github.com/transtats/transtats.git
      $ cd transtats
      $ sudo docker build -t transtats/transtats deploy/docker
      ```

  - **Pull the image** ***(No need to pull, if you have built the image)***

    ```
    $ sudo docker pull docker.io/transtats/transtats
    ```

  - **Run the image**

    ```
    $ sudo docker run -d --name container_name -p 8080:8015 transtats/
    ↪transtats
    ```

    or you can specify custom database credentials using environment variables

    ```
    $ sudo docker run -d --name container_name -p 8080:8015 -e DATABASE_
    ↪NAME=db_name \
           -e DATABASE_USER=db_user -e DATABASE_PASSWD=db_passwd transtats/
    ↪transtats
    ```

  - Application should be available at `localhost:8080` with `transtats | transtats` as login credentials.

- **docker-compose**
  - Install docker-compose
  - **This will clone the repo and start *transtats* server**

```
$ git clone https://github.com/transtats/transtats.git
$ cd transtats/deploy/docker-compose
$ sudo docker-compose up
```

  - Application should be available at `localhost:8080`.

1.5.2 Hack and Develop

- Install and run Ansible, Docker and Vagrant.
- **This will setup devel environment and run container plus, *ssh* into it**

```
$ sudo vagrant plugin install vagrant-hostmanager
$ git clone https://github.com/transtats/transtats.git
$ cd transtats
$ sudo vagrant up
$ sudo vagrant ssh
```

- **Run application**

```
$ cd /workspace
$ make run
```

- Hit `localhost:8080` in browser
- Create migrations `make migrations`
- Run tests `make lint test`
- Create cache `make cache`
- Generate docs `make docs`

1.5.3 Contribution

- The *devel* branch is the release actively under development.
- The *master* branch corresponds to the latest stable release.
- If you find any bug/issue or got an idea, open a GitHub issue.
- Feel free to submit feature requests and/or bug fixes on *devel* branch.
- Transtats uses CircleCI for tests.

1.6 Roadmap

As the project evolves, a roadmap will be published for each major release. Comments, suggestions, and requests to the current roadmap are welcome. Our goal in publishing a roadmap is transparency and community inclusion. A roadmap is the team's best guess based on experience, community requests, and feedback.

1.6.1 Transtats 0.1.6

We are currently working on `0.1.6` release.

Target delivery: In mid of August 2018

For features list please look here.

1.6.2 To Do

- **Translation Status of Packages at all 3 places:**
  - Translation Platform (Done)
  - Upstream Repository (Done)
  - Build System (Done)
- **Transtats Jobs**
  - YML Based Jobs - Parser & ActionMapper (Done)
  - Streamline Jobs and String Breakage (In Progress)
  - Scheduling of Jobs as per Release Schedule
- **Transtats New Dashboard**
  - Integration of new PatternFly UI
- **Transtats Notifications**
  - Emails about push/pull or translation status/diff

1.7 Releases

Please look github releases for release details.

See changelog here.

1.8 License and Credits

Transtats is licensed under Apache License, Version 2.0

Contributors

- transtats
- transtats-cli