Migrating TamilPesu to Cloud based Deployment

Authors: Surendhar Ravichandran <surendhar.r@proton.me>, Arunmozhi, T. Shrinivasan <tshrinivasan@gmail.com>, Muthiah Annamalai† ezhillang@gmail.com

Abstract:

Open-Tamil project has expanded to provide its API as a web-service via https://tamilpesu.us website since 2018 [1]. In this article we share the process of migrating the deployment of this API server through cloud based app-platform with a service provider thereby providing significant advantages to users of site like: secure https access, quick time from code commit to deployment, and ease of maintenance for the project developers. We propose these identifications as easier tools for maintenance and growth of Tamil web applications and cause for wider adoption in our community.

1 Introduction

Open-Tamil is a Python library, used to develop Tamil NLP applications in Python. It provides all the basic functionalities to parse the unicode Tamil Text, easily, in Python among other text processing, and simple NLP functionalities [1a,b].

The way python handling the unicode tamil is not readable, by default. It operates on the low level unicode parsing. But, to achieve high level unicode handling parsing, we need an abstraction layer, so that any new developer can handle the tamil text as regular text using open-tamil library [1d].

1.1 TamilPesu.us

TamilPesu.us [1c] is the demonstrative web application for the features of the open-tamil python library. It has a the following features/components; the entire code of this application has been open-sourced for a few years now:

- Spell checkers:
  - GNU ASpell Spell Checker, TamilinayaVaani Spell Checker
  - Sandhi Checker, Tamil Stemmer
- TTS
  - Text to Speech Synthesizer
- NLP Tools
  - Classifier, Tamil Number,
  - Crossword
  - Tamil Multiplication Tables, Word Count, Transliteration, Text summarizer
  - Tamil Morse code, Time and Date (Tamil)

†Corresponding Author

1.2 Architecture

The architecture of Tamilpesu.us web application follows that of all sample Django applications - a model-view-template (MVT) as showin in Fig. 1 (Ref: 2[a])

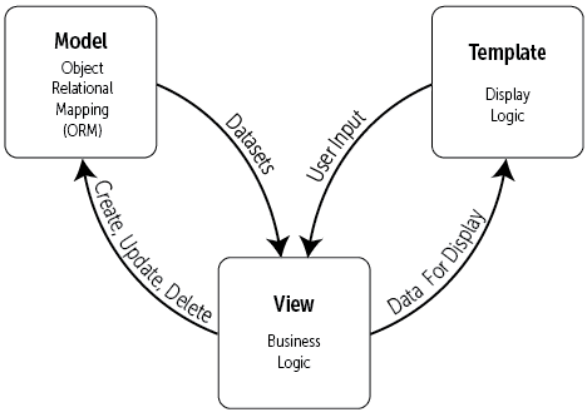

Fig 1.1 Model-view-template of Django (figure courtesy of ref: 2[a])

The busines logic portion will provide the access to the various functionalities specified in sec. 1.1 via calls to the Open-Tamil library, Tamilsandhi library and Tamilinayavaani library.

1.3 API

The API capability of the Tamilpesu.us is useful for 3rd party sites to use the NLP and other functionalities in sec 1.1. Agrisakthi magazine and CloudsIndia [2b] developers use the text summarizer features of Tamilpesu.us.

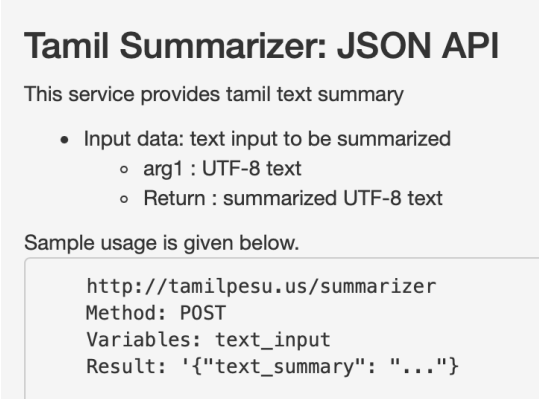

Fig 1.2 API access for Tamil Text Summarizer at https://www.tamilpesu.us/en/summarizer/

2 Deployment Process Legacy Way

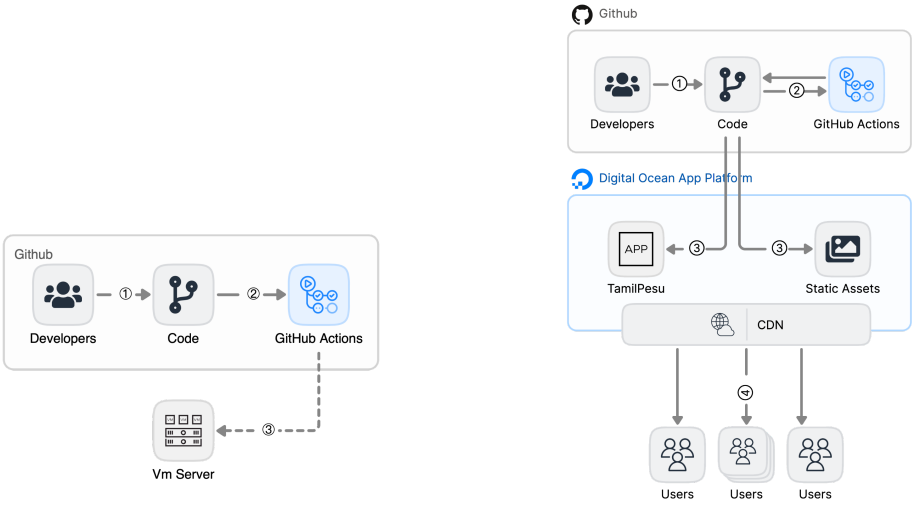

Figure 2(a): Deployment process of the code changes before adopting the app platform; 2(b) Present Deployment

The legacy deployment phase is a fully disconnected process from the development phase. The lifecycle of code change can be summarised into the following three stages.

1. Developers make code changes (Henceforth called simply, ‘Changes’) to fulfil feature additions, bug fixes and configuration changes for the TamilPesu app through a common repository hosted in Github. They propose Changes to the application in the form of Pull Requests.
2. When developers submit Pull Requests, the changes are sanity checked through GitHub Actions. GitHub Actions is a feature that can be used to implement automation tasks such as continuous integration, continuous testing and automated deployments. TamilPesu repository uses GitHub Actions to perform sanity checks against the Changes. For each of such Pull Requests, a sanity check is performed to eliminate errors before the Change is accepted and merged to the TamilPesu code base.
3. The Changes are made available to the application in the form of deployment. In order to perform a deployment, an administrator needs to log in to the Virtual Machine (VM) and obtain the latest code from Github. Once the new code is available, they need to perform certain manual operational tasks including Application server restarts, web server restarts, and database migrations that are required for the application. Administrators perform these deployments typically once a week.

3 Problems with the legacy deployment

Stages 1 and as a result Stage 2 are random events. Developers across the globe introduce Changes whenever they have time to contribute to the TamilPesu App. However, the deployment stage (Stage 3) is a periodic, less frequent event compared to the development events. Over a period of time, these changes accumulate and cause a drift between the server

deployment and the application repository. When an error occurs during the deployment, it is difficult to find the root cause because we deploy multiple code changes simultaneously. Even though we perform sanity checks in the repository, they are lower-level checks for specific functionality. They don't identify the cause of a deployment failure for the application as a whole. When an administrator fixes the deployment, they usually make fixes in the form of code changes, directly in the server. These fixes should be backported to the repository. But since the deployment is manual and the changes must be made twice – once in the server and once in the repository, the backporting often gets neglected. This in turn causes another drift between the code in the server and the repository. Over a long period, the drift makes it impossible for the developers to fix the app and the administrators to do deployments consistently and reliably.

4 Deployment Process - Fully automated on Cloud

In the current architecture, we moved from deploying on top of the infrastructure as a service model to the platform as a service model. We replaced the Virtual machine which acted as a Web server and the application server, with a container-based platform service. There are two components that constitute the TamilPesu deployment. See figure 2.(b).

1. The application server component handles the server-side logic such as computing, API and networking.
2. Static file server which serves static assets such as CSS, javascript and images.

5 Continuous Deployment

From the legacy deployment model, stages 1 and 2 stay as they are. But the key difference is that now the deployment from the code base – stage 3 is automatic. We configured the digital ocean app to look for changes in the production branch of the TamilPesu repository. As soon as the code is merged into the production branch, Digital ocean triggers a deployment.

The manual steps required to deploy the code in the application server are automated using a Dockerfile. Dockerfile is a specification of how to build and run the application from the code. The app platform takes advantage of the existing Dockerfile from the repository and uses it to build and deploy the application.

5.1 Zero Configuration Drifts

As a result of continuous deployment, the drift between the deployment and the repository is fully eradicated. There is a one-to-one relationship between an app deployment and a commit in the production branch of the TamilPesu repository.

5.2 Quick Deployment

Since there are no manual works, the deployment time is reduced significantly. Typically it take about 3 minutes for the deployment to be complete and the new version of website available for public use.

5.3 High Uptime

In case of deployment errors, the previous version of the website is kept functional serving the website traffic. This ensures the application to be available even in case of build failures. The app platform also supports a manual rollback feature to previous versions.

5.4 Monitoring and Alerting on Failures

The app platform provides basic resource usage monitoring such as CPU, Memory and network utilization. In addition, any build or deployment failure can be configured to trigger an alert email or Slack message.

5.5 HTTPS

The app deployment provides a TLS certificates for the domain name and enable them without any additonal configurations or costs.

5.6 CDN

The advantage of adding a separate static component is that these resources are served using a Content delivery network (CDN). A content delivery network is a caching service used to distribute static content across geographical regions and serve them at a high speed for the users local to that region. This significantly reduced the app loading time for users across the globe.

5.7 Scaling

In order to respond to increased app usage, we might need to scale the application horizontally. In app platform, we can increase the application containers to meet the demands of the application usage easily. The scaling completes typically within a minute.

5.8 Other Deployment Options

Digital Ocean's app platfom is one of the platform as a services provider. There are the following alternate services where we can deploy a similar architecture.

1. Kubernetes
2. AWS fargate
3. AWS EKS
4. Heroku
5. pythonhosted

5.9 Applications

In our case we were able to deploy the Open-Tamil code functionality to show Date-time in Tamil words as a Tamilpesu web-app in few hours of coding to enable the change.

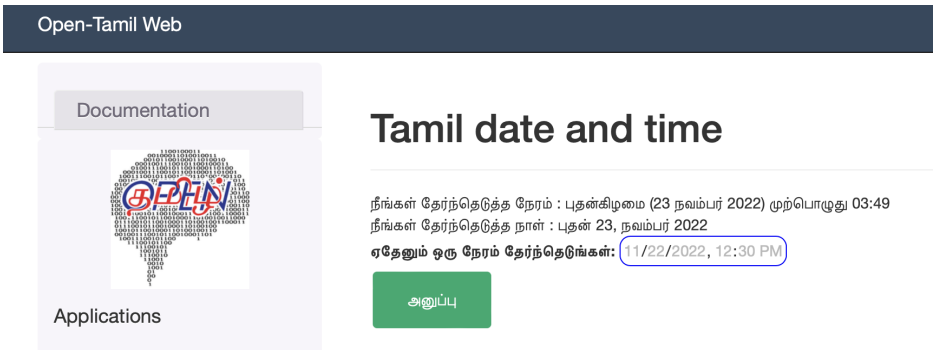

Figure 5: Tamil Date-Time function integrated from open-tamil and published to https://tamilpesu.us using app-platform auto-deploy.

6. Summary and Conclusions

In summary we have migrated Tamilpesu from manual deployment to git-action push-to-deploy methodology using the Digital Ocean app-platform where code changes are seamlessly and effectively deployed to customer. We think this is a good technology suitable for adoption by the wide Tamil developer community.

References:

1. (a) Syed Abuthahir et-al,"Growth and Evolution of Open-Tamil," Tamil Internet Conference (2018).

   (b) Tamilpesu code repository, https://github.com/Ezhil-Language-Foundation/tamilpesu_us (code change May, 30, 2022)

   (c) Tamilpesu site https://www.tamilpesu.us live website (accessed Nov, 2022)

   (d) Open-Tamil code repository, https://github.com/Ezhil-Language-Foundation/open-tamil (accessed Nov, 2022).

2. (a) Mastering Django Structure, https://masteringdjango.com/django-tutorials/mastering-django-structure/ (accessed Nov, 2022)

   (b) Selvamurali, founder https://cloudsindia.in (private communication 2019)

3. Udhayakumar, S. P., and M. Sivasubramanian. "Shift Left: Strengthening the Requirements Elicitation Process for Improving Quality Software in Software Development Projects." (2022).
4. Mulerikkal, Jaison Paul, and Ibrahim Khalil. "An architecture for distributed content delivery network." *2007 15th IEEE International Conference on Networks*. IEEE, 2007.