Academic CI Final Report

6156 Topics in Software Engineering
Larry Zhao lz2479@columbia.edu
Keir Lauritzen kcl2143@columbia.edu
Ricardo Martinez rmm2222@columbia.edu

Deliverables

We are delivering a base application and documentation to walk developers through the full continuous delivery path from code writing to deployment. The base application is a modified version of ImHere[1], an attendance-taking application that was created in Prof. Kaiser's 4156 class (Fall 2016). Because the application is aimed at students, the code base has been kept relatively simple and easy to understand.

Access

The code base:
https://github.com/keirl/coms4156_jumpstart

The documentation:
https://github.com/keirl/coms4156_jumpstart/wiki

Demo website:
https://test-4156.appspot.com/
(It can take a couple of seconds to load if it hasn't been accessed in a while.)

Tech Stack

Original Project - The application is a modified version of the ImHere project. Ricardo had previously worked on this project. We also felt it would be great to further develop an application that could be put to production use in the future. The major refactoring to the original project is the database backend. Switching from Sqlalchemy to Google' NOSQL Datastore required major overhauls to the application.
https://github.com/youngthejames/ImHere

Framework - To keep the codebase easy to understand for students, we utilized Flask, a python microframework for web applications.

[1] https://github.com/youngthejames/ImHere

Database - The official Flask tutorial uses SQLite for the database. However, we have opted for Google's Datastore because of it is available in Google Cloud's always-free tier.
Tutorial used to quickstart Datastore:
https://cloud.google.com/python/getting-started/using-cloud-datastore

Web Hosting - We are also using Google's App Engine for deployment for the same reason. App Engine and Datastore have great integration. Keeping the environment in always-free services makes the project more accessible to students. If they decide to expand their project, App Engine and Datastore can scale to paid tiers with ease.
Tutorial used to quickstart App Engine deployment:
https://cloud.google.com/appengine/docs/standard/python/quickstart

Code Repository - GitHub was chosen because it is free for open source projects. Git is also an industry standard.

Continuous Integration - Travis CI was chosen because of its ease of setup and free tier for open source projects.

Development Environment - We are advising students to use VirtualBox to set up their development environments to ensure compatibility. It is also possible to develop natively on MAC OS and Windows; however, these will not be officially supported. A few Windows laptops have virtualization disabled, so we provided Windows links as a contingency option.

Results and Lessons Learned

Larry worked on refactoring the ImHere code to work with both Google App Engine deployment and Google Datastore. Restructuring the code for deployment on App Engine was actually very straightforward. Google's sample projects for deployment on App Engine are very easy to understand and incorporate into your own code. The conversion from Sqlalchemy to Google's NOSQL Datastore required a much larger effort than expected. Google's documentation for Datastore is not as clear as their App Engine documentation. Although the sample python project Google provides for Datastore aided in developing for Datastore, there were many scenarios that were not covered. Online support for Python Datastore is also lacking. For example, there are significantly fewer Stack Overflow posts on Datastore compared to other systems Larry

has worked with in the past. An example of one particular problem faced was that Datastore was giving a cryptic error message when trying to do an equality filter on a DateTime field. It was finally discovered that Python's default DateTimes are naive to timezones. When a naive DateTime is given to Datastore, Datastore will automatically convert that object to a timezone-aware object. This means Datastore could not compare a naive Datetime to a Datetime object in a Datastore database. SqlAlchemy was also a much larger part of the original Imhere project than initially expected. Almost all files with the exception of a couple html templates were touched. Moving from a relational database to a NOSQL database also required significant changes to many pieces of logic. Because the frontend and backend in Flask are coupled, as opposed to more restful stacks Django and MEAN, this meant that the front end also required changes when the back end was changed. (The http router and back end are tied together. The router lives on the back end and returns static html that is generated from templates. Django and MEAN utilize RESTFUL API calls to the back end, and JS can be used to dynamically generate the html on the client side). The team originally planned to add Javascript to the front end for dynamic user interfaces when the plan was to use Django. However, this feature was dropped when Larry realized that the tight coupling of the front end to back end in Flask makes this less than ideal. It is still possible, but it requires some hack-y techniques that may not follow best practices. This was also Larry's first time implementing oauth2 credentials. It is a much more involved process than it appears, but it is doable.

Keir worked on the development environment, continuous integration, and deployment. This involved setting up Travis and Google Cloud as well as writing the documentation for the process. The development environment chosen was Ubuntu 16.04 on Virtualbox. Set-by-step instructions we provided.

Keir had never encounter any issues with Virtualbox until the demo, when Sony had turned off virtualization support. This was resolved, but a set of Windows links are now provided as backup.

Travis and Google Cloud are both well documented but unforgiving in their configuration. Setup took longer than expected the first time, but is stable once configured. The trickiest aspect to work with was OAuth. It is very important to get the redirect_urls exactly right on the OAuth pages. Prof. Kaiser may want to have the students be prepared on the second class to set up all of the keys in the webapp, or try it before class and dedicate time in class to debugging. Getting all of the keys is

straightforward for someone with experience, but can be confusing for a novice. Having everyone work together to get it working would be valuable.

The main area we had an issue with was testing with Datastore. Google has decent documentation on unittest[2] and pytest[3]. We were not able to work out the intricacies of setting these up with Datastore. There is an emulator available[4], which is useful for debugging. Students can use the emulator locally, but we did not find tools to inspect the data in the emulator. Testing could be done using the "production" database, but that could lead to breaking things, which is probably OK in this setting. Alternatively, two Google Cloud projects could be created and all of the files on GitHub pointing to the "production" project and then all of the files on developers machines pointing to the test environment. The emulator is probably the simplest way to test. We believe it is possible to put use the emulator in Travis.

(Ricardo) I worked on creating a minimal ImHere application by stripping the completed ImHere of cosmetic/extraneous functionalities, and reforming it to be a functional base for users of coms4156_jumpstart. I also worked on some of the documentation (about the app, and enabling some of Google's API), as well as ensuring that the instructions in our project were complete and robust.

What I learned from the project is that tools which are only moderately difficult to use in isolation can become extremely confusing to use when joined together with other tools. Having gone through 4156 myself, I was already familiar with the technology stack that is implemented in this project, but I realize that learning new tools is a challenge in and of itself when developing software; furthermore, using multiple new tools in combination can be troublesome because it is not immediately obvious which tools provide which functionalities, and which tools don't. It is for this reason that I think experience proves most useful, which might not be attainable in only 6 weeks. Nevertheless, we made a great effort to make the development environment as easy as possible to set up and dive into.

---

[2] https://cloud.google.com/appengine/docs/standard/python/tools/localunittesting
[3] https://github.com/keirl/coms4156_jumpstart/tree/datastore/2-structured-data/tests
[4] https://cloud.google.com/datastore/docs/tools/datastore-emulator