COMPX341-20A Assignment 4

Instructor: Panos Patros

Due Date: 6pm Friday, Jun 5, 2020 (*The deadline has been moved due to COVID19*)

Individual Assignment—Software will track similarities!

Weight: 20% (*The weight has been modified by +5%*)

Electronic Submission on Moodle—Late submissions will not be accepted!

DevOps and Continuous Deployment on the Cloud

You've been hired by a startup that aims in rapidly creating node.js apps for clients via building a small agile team. Your employer decided that all activities will be run on the cloud to minimize managing hardware/software resources as well as facilitate Continuous Delivery through cloud-based DevOps and Continuous Deployment tools.

You've been assigned to evaluate the efficacy of this approach by creating, maintaining and testing a simple application. Since no credit-card information is needed to start experimenting, IBM Bluemix has been chosen as the cloud provider: you've been requested to base your solution on the following tutorial: https://www.ibm.com/cloud/garage/tutorials/use-develop-test-cloud-foundry-app-toolchain

As the tutorial instructs you, you must already have (or create) a GitHub account. You obviously have one since your employer obviously hired you after checking your GitHub account first!

Ensure you **select Dallas (US-South)** as your location as some other locations do not provide all services.

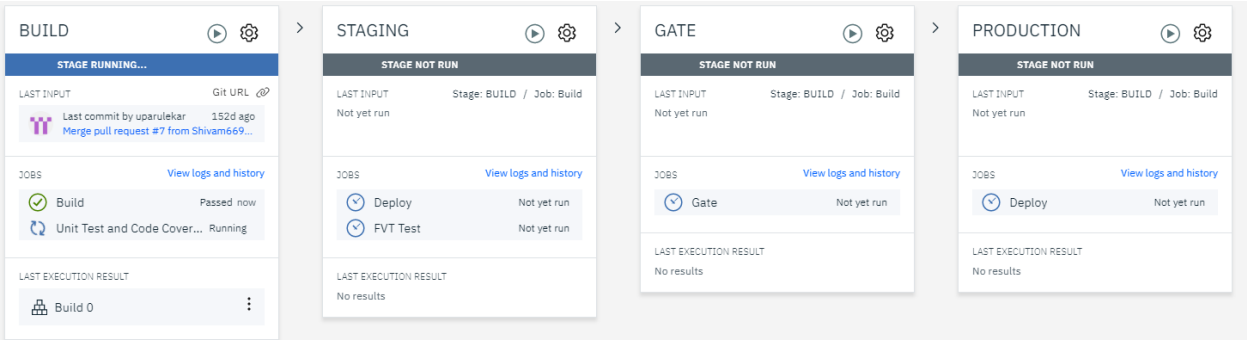

Figure 1

As displayed in Figure 1, toolchains take some time to start. This is because they automatically go through various Software Engineering processes, such as automated testing. You might need to be a bit patient every time you commit a change to see it actually working. Also, take some time to familiarize yourself with the toolchain and read back frequently in the tutorial.

When your toolchain has completed, it will show a green checkmark, as displayed in Figure 2.

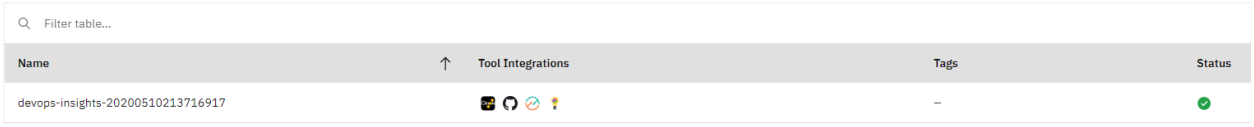

Figure 2

You will be able to test the web app produced by following its link from the Connections tab. The base app looks as in Figure 2.

Figure 3

To maintain the code, you can use the embedded Eclipse Web Orion IDE and make your changes as in Figure 4. After you're done with all the changes in a version, press the git button on the left (under the pencil), write up the commit text, click commit and then sync.

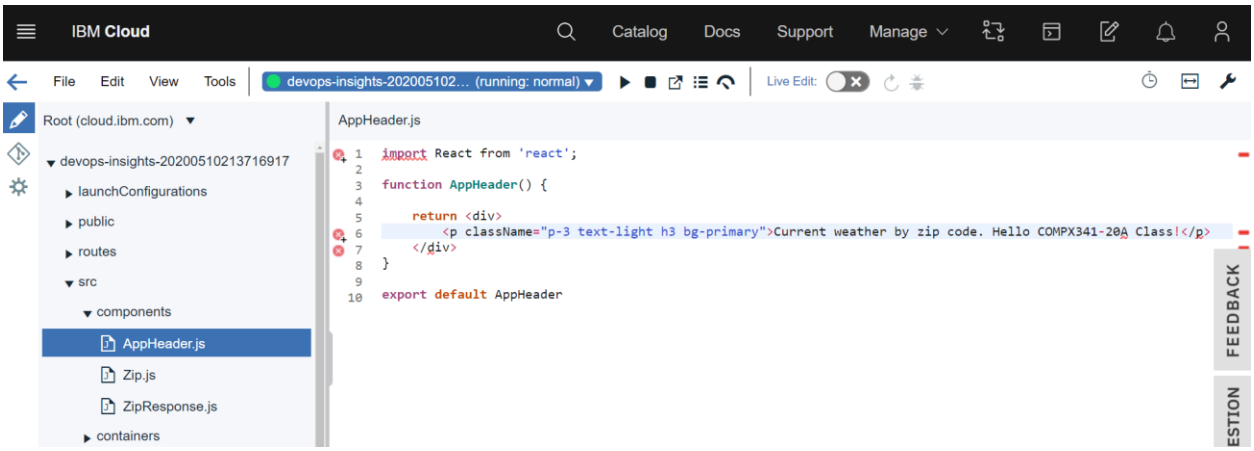

Figure 4

Pushing a new commit, takes it through the whole toochain, which seems to take 20 minutes or so. Check Figure 5 for an example.

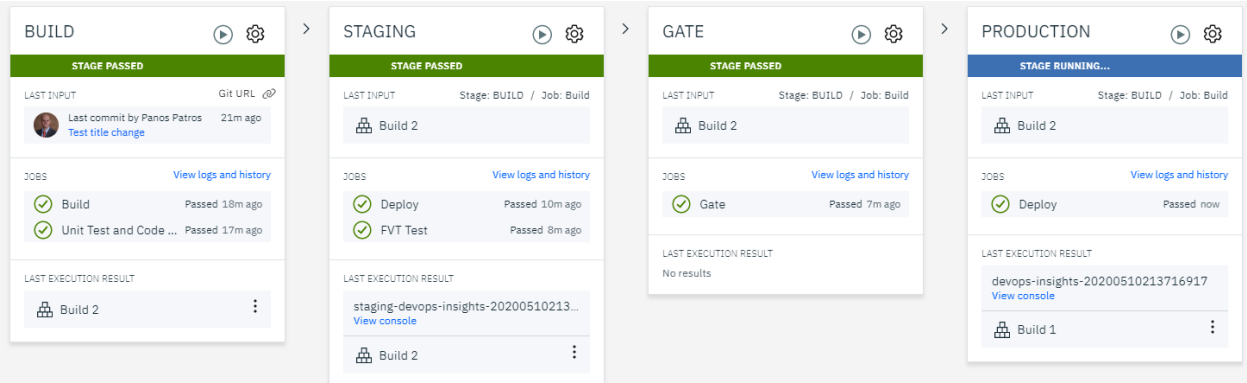

Figure 5

Finally, you can see the effect of your commit live in production and/or staging by clicking the link at the connections tab (Figure 6).

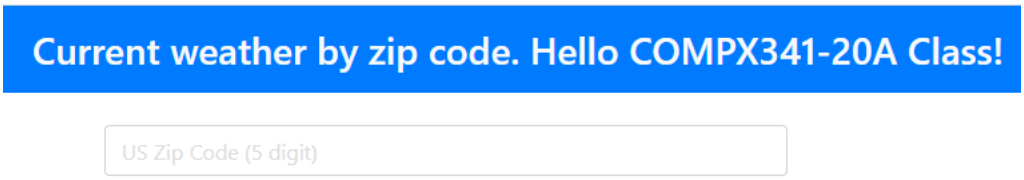

Figure 6

Required Modifications

After ensuring the tutorial app works properly, maintain it by making the following three versions that build on top of each other. For each version, you also need to design and implement new automated tests that will be automatically running in the delivery pipeline. In your commits, clearly name the final production-ready commit for each version as “COMPX341-20 A4 VERSION 1”, “COMPX341-20 A4 VERSION 2” and “COMPX341-20 A4 VERSION 3” respectively.

1) **[Value 40%]** The system currently works with US zip codes. Update it such that it works for New Zealand cities by their name and uses metric rather than imperial. The sample app leverages the openweathermap API; read more on how to inquire weather data per city name here: https://openweathermap.org/current#name
2) **[Value 40%]** Add a front-end Google-maps functionality (https://developers.google.com/maps/documentation/javascript/tutorial) that a) puts a pin on the map for each NZ city on the list and b) allows users to click on the NZ map to select a city (location) for displaying its weather.
3) **[Value 20%]** Create and connect a persistence service of your choice to your system. IBM Cloud offers various storage, database, etc. options (e.g., https://cloud.ibm.com/docs/services/Db2onCloud?topic=Db2onCloud-getting_started_db2oncloud#getting_started_db2oncloud) . Update the application such that it stores the locations displayed in the persistence service. When the webpage loads, it prefills the form with all cities stored. No user management is required; all requests will be viewing/updating the same collection of locations.

As a Software Engineer, you are expected to always learn new languages and technologies expanding on your current knowledge. A good place to learn some of the languages needed for maintaining this software can be found here: https://www.w3schools.com/

Deliverables

Write a report where you explain the modifications (sufficiently technical language and/or appropriate usage of design diagrams is expected; keep it succinct) and tests you performed for each version (justification is expected on the test cases selected) as well as the GitHub link to the commit of each version. Additionally, for each version, include representative manual-testing screenshots and the code changes you've made. Finally, provide a link to your production website, which should be running the latest version you achieved.

Quality Assessment

Per modification, the following will be expected for completeness:

- Good presentation, English, sections, content table, etc.
- Functional solution
- Representative screenshots
- Design discussion and/or diagrams
- Justified test cases
- Automated testing in the toolchain

End of Assignment 3
