CORK INSTITUTE OF TECHNOLOGY

Module Title / Code: Agile Processes / COMP7039

Title: Continuous Integration Project

Group Name: CSQM

Group Logo:

Date: 30th November 2018

Version number: 1.0

NAME	STUDENT ID
RAJA NOOR HAFAWATI BINTI RAJA AZMAN SHAH	R00182082
CONOR MORGAN	R00139200
SHANE FENTON	R00150873
MOHAMMED ALOM	R00144214

Table of Contents

Declaration of originality:

1.0 Introduction

Continuous integration is process of automating the build and testing of code every time a team member commits changes to version control. The process of committing code changes and automating a build system as soon as the commit process happens help to build, test and validate the latest changes on the code to prevent further merge conflicts. There are a few numbers of continuous integration service that can be used to be linked with the code repository to do this task.

GitHub is used as a location of the code repository as it can be accessed online by numbers of people across devices. Besides that, any changes inside GitHub can easily be committed before integrated with the connected continuous integration service. The codebase that has been used is in Python programming language.

Therefore, a total of three continuous integration service named Travis, Jenkins and Concourse has been selected. The GitHub code repository will be setup with each continuous integration service to configure a continuous deployment pipeline that will automatically build and deploy the script whenever there are any changes committed in the code repository.

2.0 Travis CI

2.1 Setup and configuration process

Travis continuous integration (CI) is a hosted, distributed continuous integration service used to build and test software projects at GitHub. If users decide to make their project open source, they can use Travis CI for free. However, if the project is private, there is fee need to be paid by users to get the project tested at Travis.

To setup Travis with GitHub code repository, we had to make an account at Travis CI website (https://travis-ci.org/). The registering account process is simple and straightforward as there are steps provided inside the website, helping to guide users. We have the option to sign up into Travis's website with their GitHub account.

As soon as we are done with the previous setup, we had to accept the 'Authorization of Travis'. We were directed to GitHub and we had to click the green 'Activate' button and select the repositories we want to use with Travis. To finish off setup and configuration process, we had to create a travis.yml file inside the chosen GitHub repository and commit it. This is where it gets a little bit tricky.

Inside travis.yml file, we have to write instructions on what should Travis do if there is any changes committed inside the repository. We searched for various examples of travis.yml file online to get the idea on what need to be written. In conclusion, travis.yml file will contain vital information such as what language that we used for the program committed, the versions of the language and what tests that Travis should run once there is any changes committed to repository.

Once we have written travis.yml file and committed it to the code repository, Travis will start to build the integration. To know whether the integration is successful or not, we will have to go Travis website and check the 'My Repositories' tab. The integration is successful if it's shown as below:

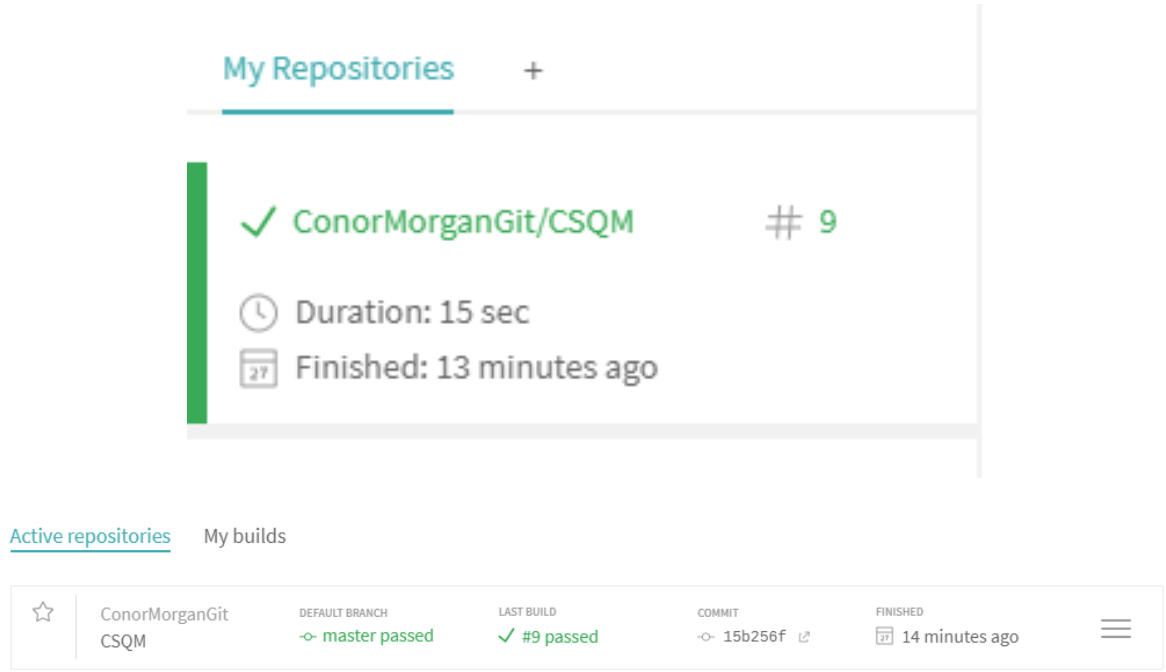


Figure 2.1: Successful Travis integration with GitHub repository

These are the files available inside our GitHub repository once the integration with Travis is successful.

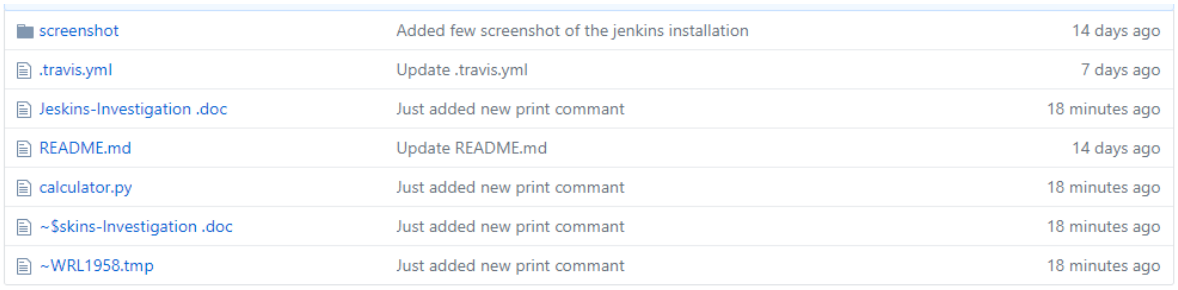


Figure 2.2: GitHub repository's files

2.2 Evaluation of Travis setup and configuration process

Setting up repository with Travis was quick and easy due to the fact that there is no downloading of any programs need to be done. Registering a Travis account was made easy as there are steps provided at the website to guide users.

Connecting Travis with GitHub account wasn't that hard either as we just have to find out how to do this by following the steps explained and as soon as we understand the guide, we can easily sync Travis with GitHub repository that we chose.

The most difficult step in this task was to create travis.yml file as we had to do some searching online to know what need to be written inside the file. After finding a good explanation about it, we went to look for appropriate travis.yml examples that is relevant with our GitHub repository. We tried a few different versions of these examples with our own travis.yml file and while doing so, we encountered errors for the first few numbers of commits. Upon further investigation and a series of trial and error, we successfully came up with a right solution.

In conclusion, the setup and configuration process of Travis and GitHub repository was quick and mostly easy to complete.

3.0 Concourse

3.1 Setup and configuration process

Concourse is an open source continuous thing-doer. It helps teams that practice agile development due its continuous integration and continuous delivery system. As Concourse is built on simple mechanics of resources, tasks and jobs, the central concept of Concourse is to run tasks.

Concourse can be used when user need to automate test driven development, maintain compatibility between multiple build versions, target multiple platforms and configurations such as different clouds and deliver frequently (weekly, daily or even, multiple times a day). In Concourse, a pipeline exists as the result of configuring jobs and resources together. Pipelines are configured as declarative YAML files. YAML file is a human-readable data serialization language. Each entry under resources is a dependency, and each entry under jobs describes a plan to run when the job is triggered.

To setup and configure Concourse with code repository, first, we downloaded Docker for Mac. Docker provides a way to run applications securely isolated in a container. It

allows us to build and share containers and automate development of pipeline from a single environment.

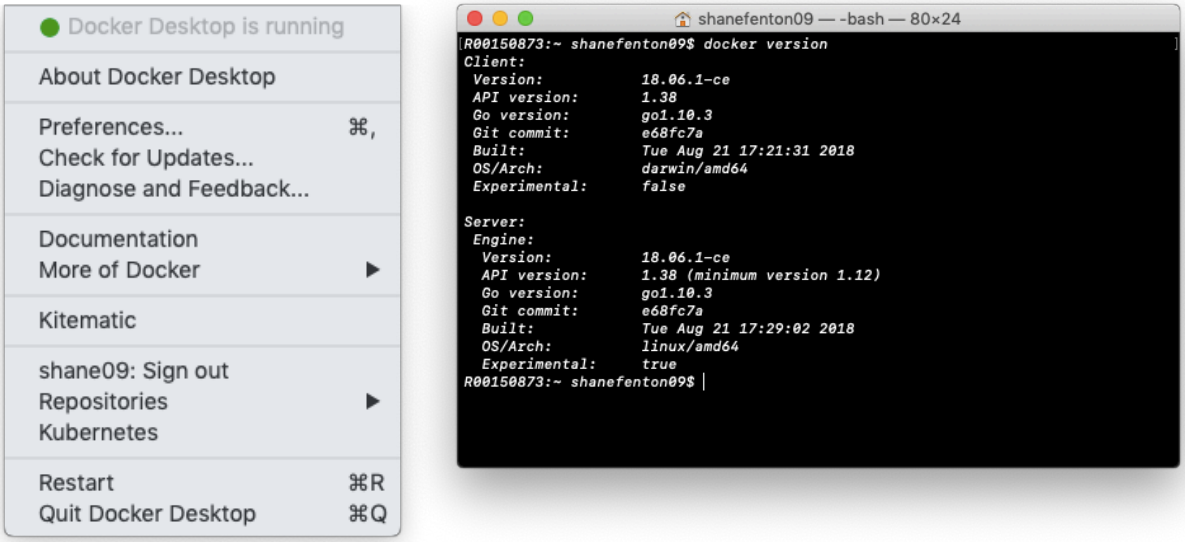


Figure 3.1: Successful download of Docker for Mac

Now, to setup Concourse, we downloaded the quick start Docker Compose file provided by Concourse. We used the 'wget' command to download this file.

Figure 3.2: Downloading Docker Compose file

We ran 'docker ps' to show that docker has setup two Docker containers: one for Concourse and another one is for Postgres. All the ports, permissions and credentials have been taken care of as shown below.

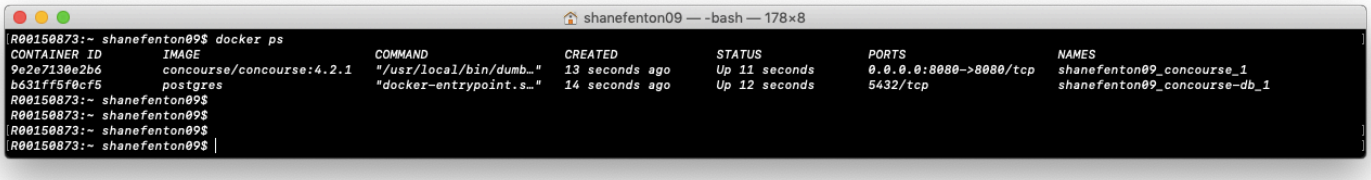


Figure 3.3: Docker containers and ports, permissions and credentials.

To access Concourse, we used http://127.0.0.01:8080/ as the host address.

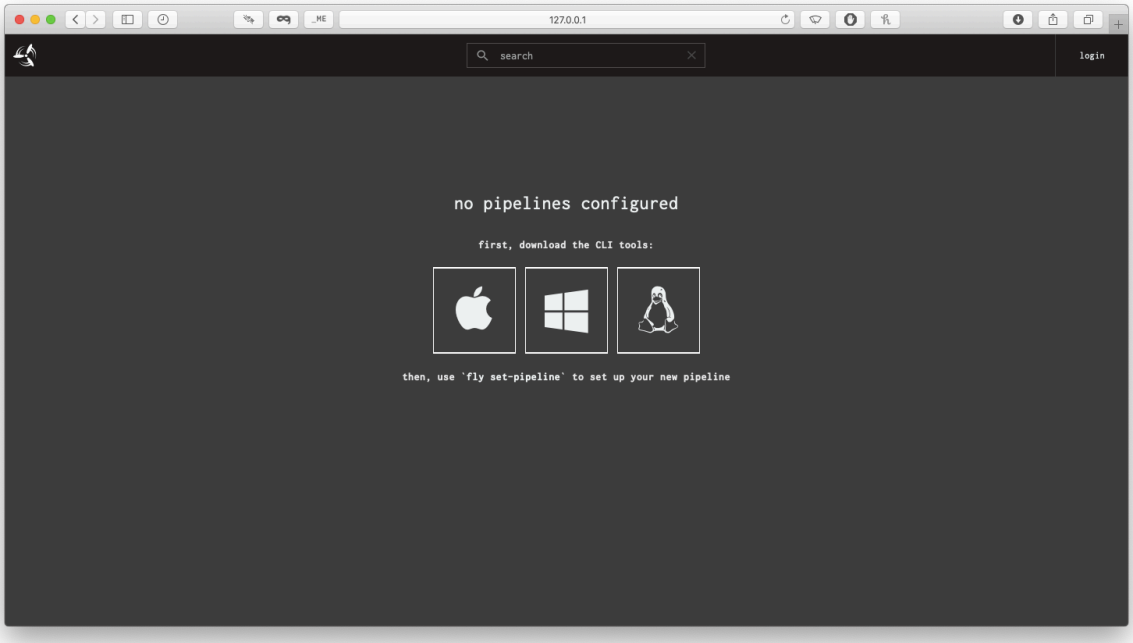


Figure 3.4: View of http://127.0.0.01:8080/

To install Concourse CLI(fly) inside our system, we clicked on the Operating System we are currently working on and download the file. As we use Apple device to run Concourse here, we clicked on Apple's icon. We then change our terminal to view inside the Download folder and install fly.

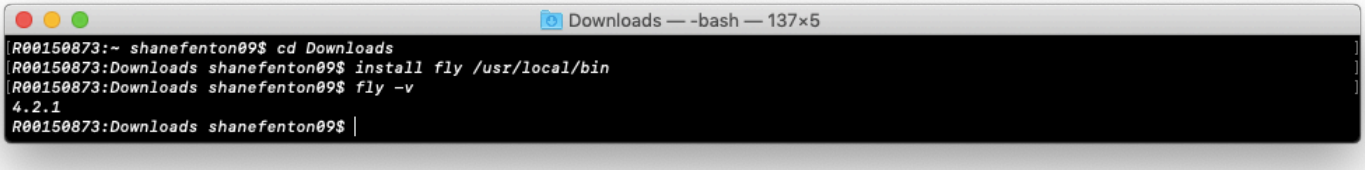


Figure 3.5: Installing fly

To setup a pipeline, we have to login using the fly login command.

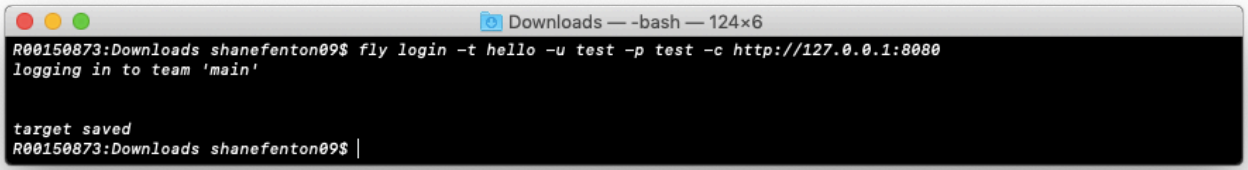


Figure 3.6: Log in using fly login command

From here on, we need to setup a pipeline to use Concourse with our repository. To make this work, we get a YAML file from an associated GitHub in the form of a raw file. We used the 'wget' command to the .yml file.

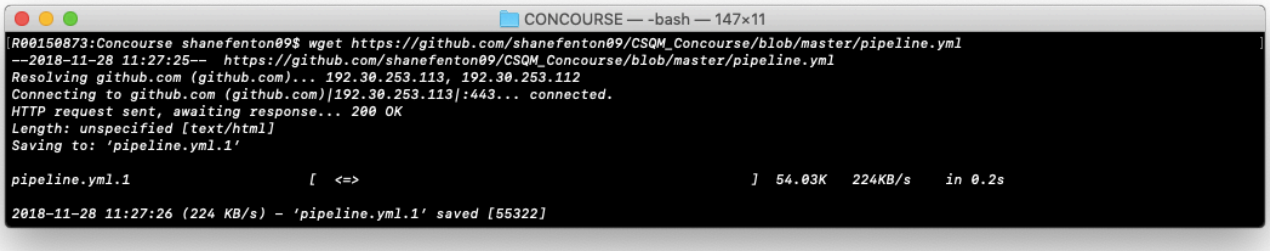


Figure 3.7: Getting YAML file from associated GitHub using ‘wget’ command.

Then, we used fly command to set the pipeline. The pipeline was un-paused afterward and trigger was set.

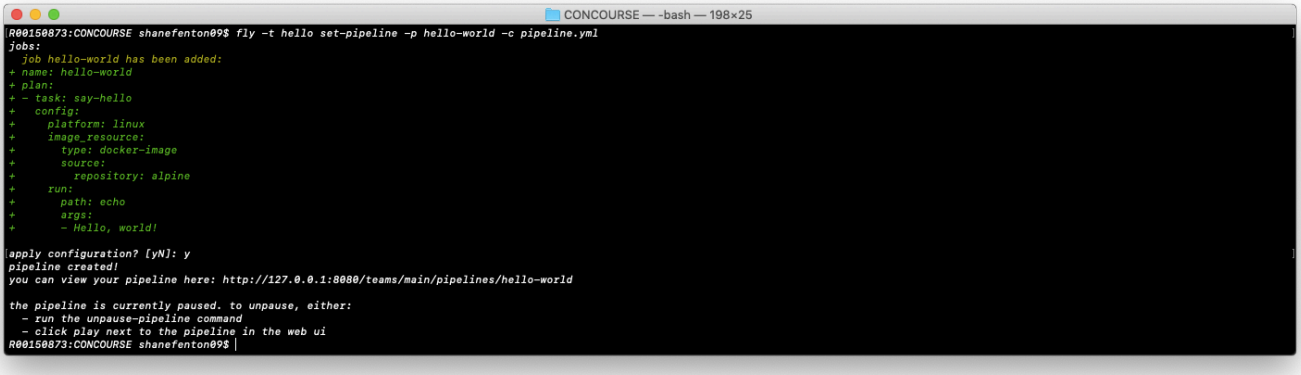


Figure 3.8: Using fly command to setup pipeline

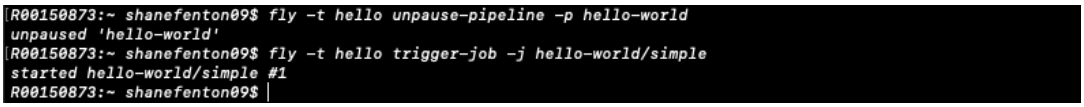
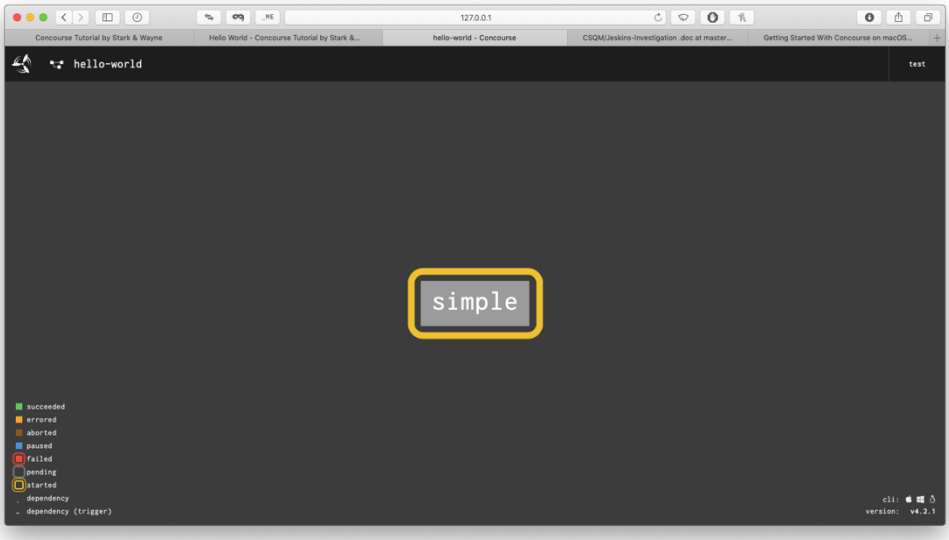


Figure 3.9: Un-pausing pipeline and set the trigger

Below are the outcomes for setting up and configuration with Concourse. The different colors are corresponding with the status.



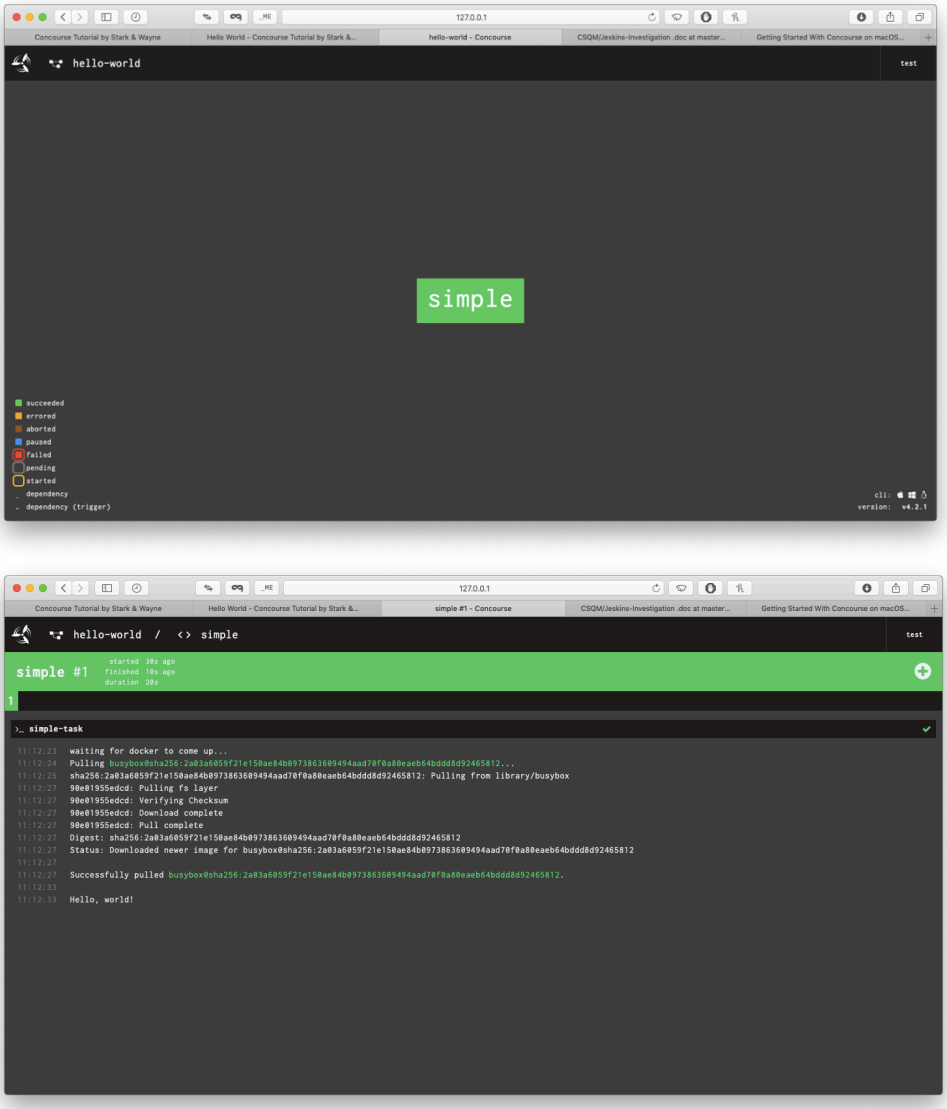


Figure 3.10: Result of setup and configuration with Concourse

3.2 Evaluation of Concourse setup and configuration process

Installing Concourse on Mac with Docker was not difficult as it is mainly straightforward but however, we had trouble with upgrading fly to the correct version due to the lack of tutorials online and forums. Fly is the Concourse command line interface (CLI).

The lack of tutorials and forums on sites such as Stack Overflow making it hard for us to find guidance and resources, hence, it slowed the installation process down until we could figure out the problem. We were installing fly in a directory stated by the Concourse documentation but however, this documentation is outdated. The specific directory was removed in latest MAC update.

After Concourse was installed and we have parallel versions of both Concourse and Fly, we started to create a pipeline. Concourse use YAML format to configure pipelines. Tasks are written as Linux scripts, and this is where we face a difficulty. No one in our group is familiar with Linux scripting. This makes it difficult to create a script that will be integrated with our GitHub code repository that has Python program.

We did find a sample of YML files online through the Concourse documentation. We tested these samples to see and understand how Concourse works. Concourse has a web interface (Graphical User Interface) that could be shown on a screen, where it's very easy to see the current state of a job, as it colors coded to green and red. This is a good advantage as it could quickly alert a member of the team, if something has failed.

In conclusion, the setup of Concourse was kind of easy, but creating pipelines with specific jobs and tasks was difficult due to a lack of experience with Linux scripting.

4.0 Jenkins

4.1 Setup and configuration process

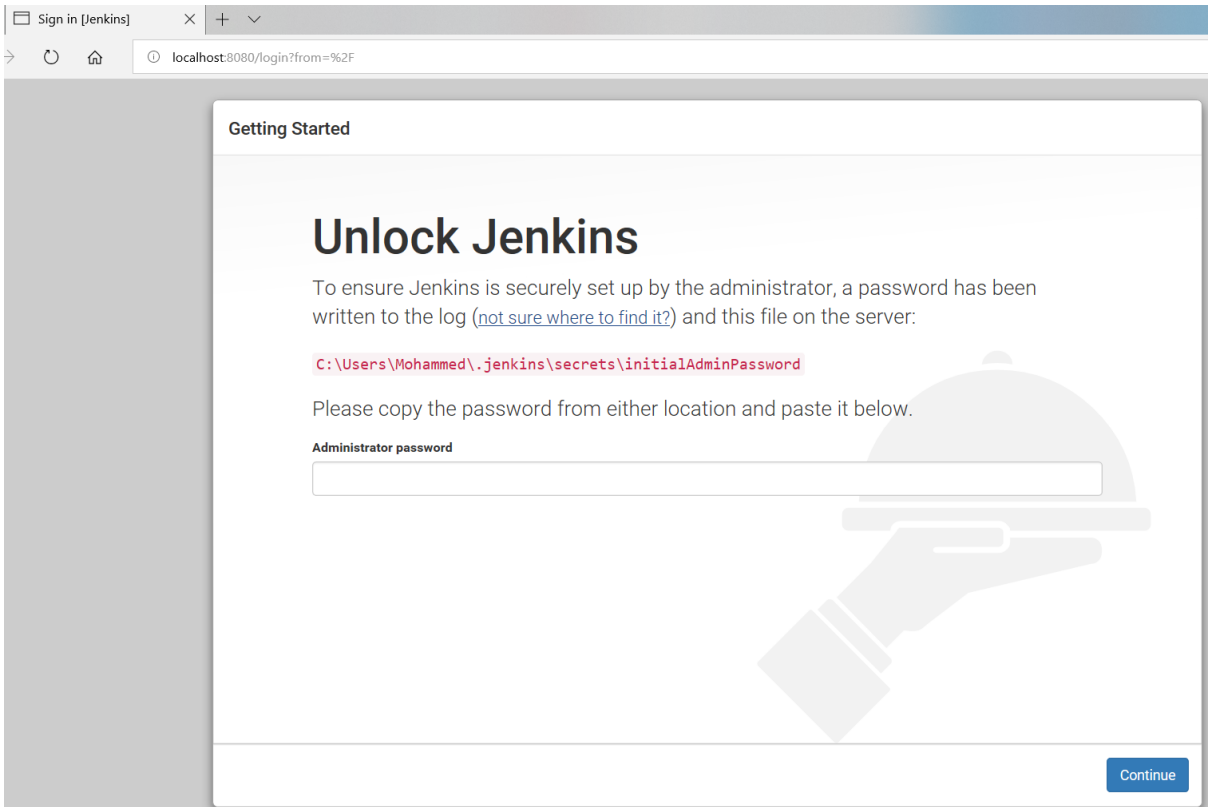
Jenkins is a self-contained, open source automation server which can be used to automate all sort of tasks related to building, testing and delivering or deploying a software. Build will be triggered each time a change is made and committed inside a repository. Once the code is built, Jenkins will deploy it on test server for testing task. This helps for continuous integration of software and continuous delivery. Jenkins can be installed through native system packages, Docker or even, run standalone by any machine that has Java Runtime Environment (JRE) installed.

To install and setup Jenkins in a local machine, we make sure that our machine meets the pre-requirements that has been set. Jenkins typically runs as a standalone application in its own with built in Java servlet container or application server (Jetty). The requirements to install Jenkins are:

- Minimum hardware requirement:
  - 256MB of RAM.
  - 1GB of Drive space (10GB is recommended for running Jenkins as a Docker container).

- Recommended hardware configuration for small team:
  - 1GB+ of RAM.
  - 50GB+ of drive space.

- Software requirements (Modern Jenkins version have the following Java requirements):
    - Java 8 is the only supported runtime environment (both 32-bit and 64-bit).
    - Older versions of Java and Java 9 are not supported.
    - Java 10 and 11 preview supports are available.



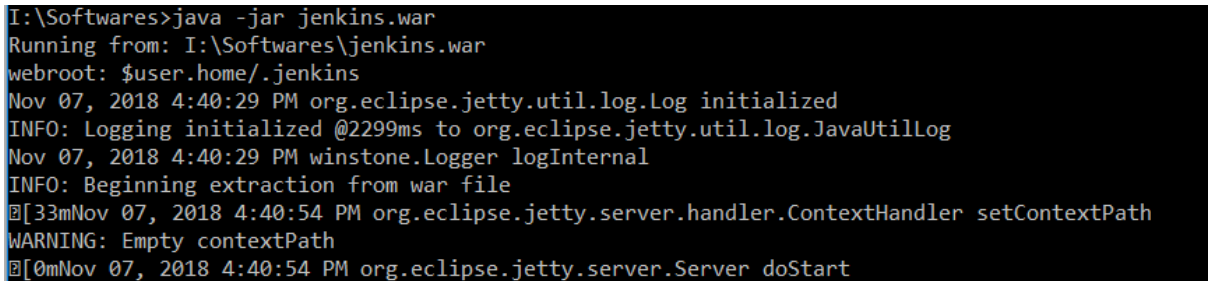


Figure 4.1: Post installation setup process

Once we have installed Jenkins inside our machine, we had to go to localhost:8080 to do setup process. The admin password that was generated during the installation process will be inserted when local host page prompted for it. If we want to login admin panel with password, we have to make sure that Jenkins.war file is running.

We can change the port by specifying –httpPort option when we run the 'java -jar jenkins.war' command. For example, to make Jenkins is accessible through port 9090, we could specify it by using this command: 'java -jar Jenkins.war –httpPort=9090'.

After unlocking Jenkins, 'Customize Jenkins' page appeared. In here, we can install any number of useful plugins as part of our initial setup. We can click on one of the options shown inside the page which are:

- Install suggested plugins: to install the recommended set of plugins, which are based on most common use cases.
- Select plugins to install: to choose which set of plugins to initially install.

When we first accessed the plugin selection page, the suggested plugins were selected by default. After customizing Jenkins with plugins, Jenkins asked us to create our first administrator user. To do this, the steps below were done:

1. When 'Create First Admin User' page appeared, the details of administrator user are specified inside the respective field.
2. The 'Save and Finish' button is clicked.
3. When 'Jenkins is ready' page appeared, we clicked on 'Start using Jenkins' button.
   3.1 There are cases where the page might indicate 'Jenkins is almost ready!' instead and if this happen, just click 'Restart'.
   3.2 If the page doesn't automatically refresh in a minute, use our web browser to refresh the page manually.
4. If required, login to Jenkins with the credentials of user that we specified in step 1 and we were ready to start using Jenkins.

To configure Jenkins with GitHub project, we followed the steps specified as below:

1. Create a Java program.
2. Compile from command line javac 'name of .java class' java 'name of class without extension .java'.
3. Create a Jenkins job to run the program.
4. Add the project to Git and GitHub.
5. Configure Jenkins job to trigger build on Git commit.

When creating new item in Jenkins, we can choose the jdk in General tab (execute multiple builds if necessary option), either we click on System or the jdk that we have installed on that moment (for example, jdk_1.8.0_181).

With the help of Git plugin, Jenkins easily pull source code from any git repository that Jenkins build node can access. Using 'GitHub Authentication' plugin, it is possible to use GitHub's own authentication scheme to implement authentication in our Jenkins's instance.

Figures below shows a successful creation build version between Jenkins and our GitHub repository named CSQM.

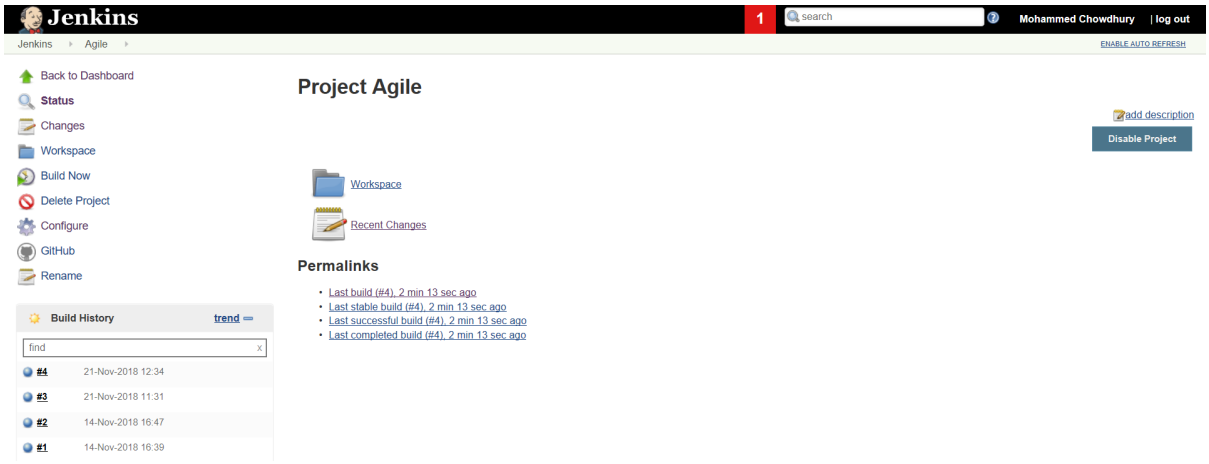


Figure 4.2: Jenkins's dashboard

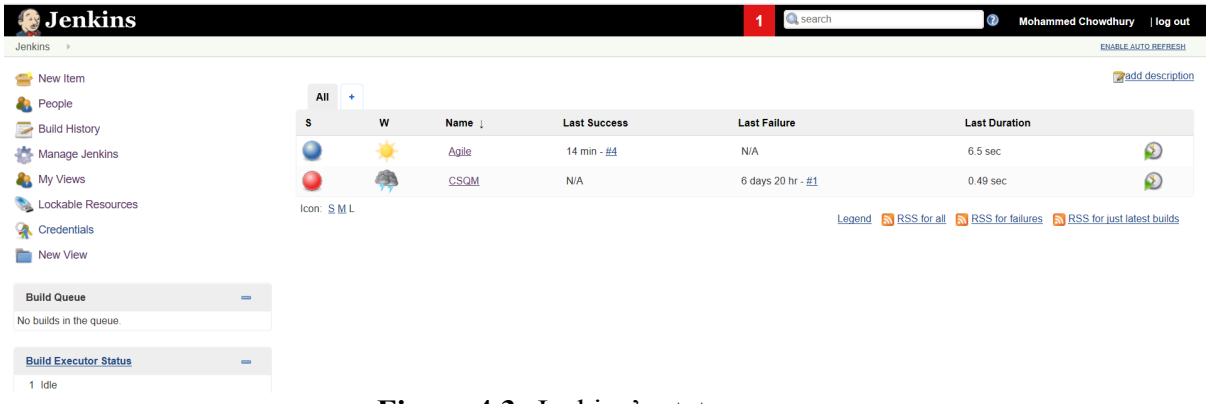


Figure 4.3: Jenkins's status

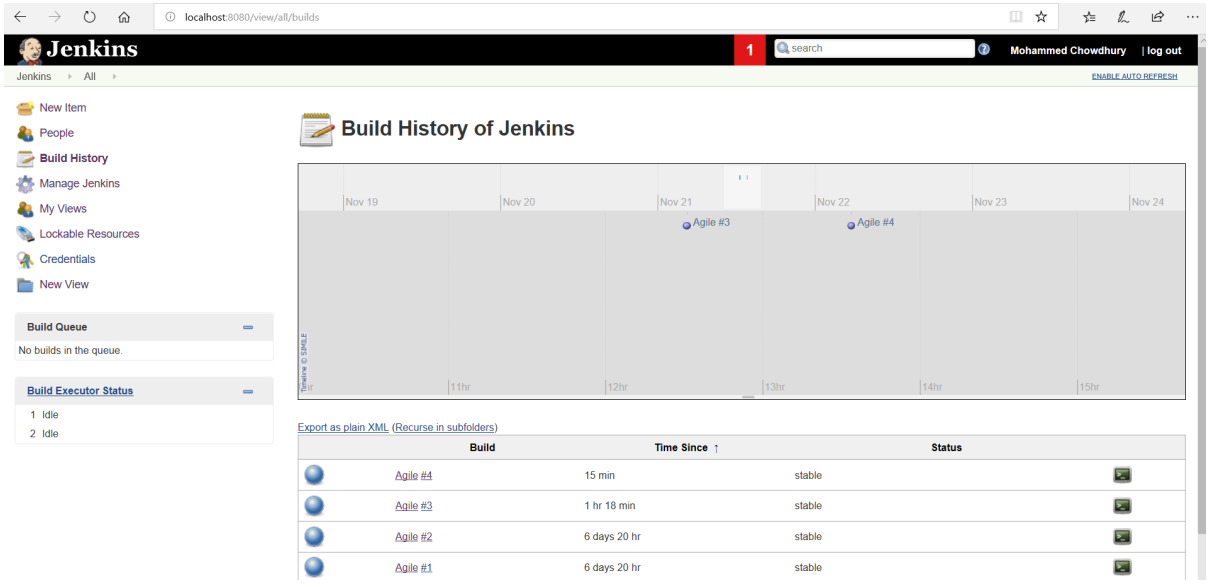


Figure 4.4: Jenkins's build history

Other than running Jenkins standalone, we can also run Jenkins on Tomcat by making sure that we meet the pre-requisite specified:

- Tomcat: 5 or above.
- Java (7 or above) should above.

The steps to run Jenkins on Tomcat are:

1. Download Tomcat.
2. Unzip and place Tomcat folder at any location.
3. Copy Jenkins.war file inside tomcat/webapps folder.
4. Go to command prompt (cmd) on Windows / Terminal on mac and write:

   Go to tomcat/bin directory –
   Make all files executable: chmod +x *.sh

   This step might not be needed if we are on Windows as sometimes, the command will not work. Else, we can right click on file/folder goto security tab and change the permissions.
5. Start Tomcat: ./startup.sh (to shutdown tomcat: ./shutdown.sh).
6. To verify if tomcat has started: type 'http://localhost:8080' in browser.
7. To verify if Jenkins is running on tomcat: type 'http://localhost:8080/jenkins'.

Other than that, if we want to change the home directory for Jenkins (this directory contains all configurations, plugins, job details and logs), we can follow the steps below:

1. Check our current home directory.
2. Create a new folder (which will be new home dir).
3. Copy all data from old dir to new dir.
4. Change 'env variable – JENKINS_HOME' and set to 'new dir Windows – change env variable'. For Mac, go to terminal export JENKINS_HOME=/Users/raghav/Desktop/Tools/Jenkins/JenkinsHome/.
5. Restart Jenkins.

Jenkins also has its own command line interface and to use it:

1. Start Jenkins.
2. Go to Manage Jenkins – Configure Global Security – Enable Security.
3. Go to 'http://localhost:8080/cli/'
4. Download jenkins-cli.jar. Place this file at any location.
5. Test the Jenkins command line is working.

To do automated deployments with Jenkins:

1. Start Jenkins.
2. Install plugin: https://wiki.jenkins.io/display/JENKINS/Deploy+Plugin. Or, we can also go to our Jenkins on browser and click on 'Manage Jenkins – Plugins Manager – Click on Available tab' and search for deploy to container 'Add to Jenkins'.
3. Create a Build job in Jenkins.
4. Add Post-build Action Deploy war/ear to container (add values to fields). Sample war file: https://tomcat.apache.org/tomcat-6.0-doc/appdev/sample/.

5. In tomcat-users.xml, add user for Deployment user username=”deployer” password=”deployer” roles=”manager-script”.
6. Run and validate.

Jenkins also has a pipeline which is a workflow with group of events or jobs that are chained and integrated with each other in sequence. Every job in a pipeline has some dependency on one or more other jobs. There are two pipelines in Jenkins which are Delivery and Build. Below are the steps to setup Delivery and Build pipelines in Jenkins.

Setup Delivery pipeline in Jenkins	**Setup Build pipeline in Jenkins**
Step 1: Chain required jobs in sequence. Add upstream/downstream jobs.	Step 1: Chain required jobs in sequence. Add upstream/downstream jobs.
Step 2: Install Delivery Pipeline plugin.	Step 2: Install Build Pipeline plugin.
Step 3: Add Delivery Pipeline View to configure the view.	Step 3: Add Build Pipeline View to configure the view.
Step 4: Run and validate.	Step 4: Run and validate.

Table 4.1: Steps by steps comparison to setup Build and Delivery pipeline

There are a few numbers of difference between Build and Delivery pipeline. At a high level, we can say that Build pipeline has a smaller scope in terms of the entire process of software development and delivery whereas Delivery pipeline has a much broader scope. When we say delivery, Build pipeline will be one of its components. Build pipeline provides a view of the upstream and downstream jobs setup for the build process whereas Delivery pipeline gives the visualization of the complete delivery process that may include build, deploy and test. In Delivery pipeline view, we can see each of these actions in a separate box.

After doing few numbers of research and going through Travis CI, Concourse and Jenkins setup and configuration process, we have decided that there is one continuous integration service that is better than the other two.

4.2 Evaluation of Jenkins setup and configuration process

Jenkins is widely known in the DevOps world as the leading open source automation server for continuous integration and continuous delivery. Jenkins enables developers to find and solve defects in a code base rapidly and automate testing of their builds.

After doing thorough reading and performing CI with Jenkins, what we found out is that Jenkins is an open source CI, a user friendly one as well. It is rather easy to install Jenkins and it does not require additional installations or components. It is also free of cost. Besides that, Jenkins can be easily modified and extended. It deploys code instantly and generates test reports too.

Jenkins can be configured according to the requirements for continuous integrations and continuous delivery. This CI service is available for all platforms and cater to various different operating systems too such as OS X, Windows and Linux.

The extensive pool of plugins makes Jenkins flexible and allows building, deploying and automating across various platforms. Easy support is available because as Jenkins is open source and widely used, there is no shortage of support from large online communities of agile teams. Issues are detected and resolved almost right away which keeps the software in a state where it can be released at any time safely. Most of the integration work is automated. Hence, fewer integration issues. This saves both time and money over the lifespan of a project.

However, we also managed to figure out a few cons of Jenkins which are:

- Jenkins has too many plugins.
- Jenkins does not support microservice well.
- Continuous integration doesn't equal to continuous delivery.
- Jenkins must be run on a server (this requires cost) so it often needs the attention of someone with system administration skills (takes time). We can't expect Jenkins to be set up and run by itself as the system requires frequent update and maintenance.
- The main barrier to entry for most teams is initial setup, procrastination or failed previous attempt to set it up.

Last but not least, Jenkins is very versatile. It's free and easy to set up and configure. Furthermore, it's not that difficult to use Jenkins and we can nearly do everything with it as it has tremendous number of plugins available.

5.0 Continuous Integration (CI) service recommendation

After doing a thorough research and going through Travis CI, Concourse and Jenkins setup and configuration process, we have decided that there is one continuous integration service that is better than the other two. Travis CI is highly recommended due to few numbers of reasons.

First, the setup and configuration process of Travis CI with GitHub is quick and easy. There is no need to download any program or run any command from command line interface as everything could be done easily at Travis CI website and GitHub website. There are a good number of tutorials posted online to guide user and help them with Travis CI configuration process in case we face any difficulty too.

The build status interface is also simple enough: user can easily understand if the build is a success or a fail by looking at the color (green: success and red: fail). This continuous integration service can also be configured to run tests on a range of different machines with

different software installed as well. Travis CI supports building software in numerous programming languages such as C, C++, Python, PHP, Ruby and many more.

Once a build is deployed, the result can be notified to user via email, IRC, Campfire and Slack. Travis CI also doesn't require any integrated development environment (IDE). The fact that it integrates well with GitHub repository also one of the deciding factor why we recommend this service. The fact that it is free to use (if your project is open source) is also a plus.

All in all, we recommend Travis CI for anyone who wants to start using continuous integration service as it is clean, smooth and easy to understand.

6.0 References

To setup and configure Travis CI, we refer to these websites for guidance:

- https://travis-ci.org/
- https://travis-ci.com/

To setup and configure Jenkins, we refer to these websites for guidance:

- www.jenkins.io/doc
- www.github.com/jenkinssci/jenkins
- www.tutorialspoint.com/jenkins

To setup and configure Concourse, we refer to these websites for guidance:

- https://concourse-ci.org/tutorials.html
- https://medium.com/concourse-ci/getting-started-with-concourse-ci-on-macos-fb3a49a8e6b4
- https://gist.github.com/kevin-smets/f20afd45a24ab3f88d01b2049ce7744f