ACTI 4850 – Lab 4 – Orchestration Tool

Instructor	Mike Mulder (mmulder10@bcit.ca)
Total Marks	10
Due Dates	Demo Due by End of Class Next Week (Feb. 5th for Monday Set, Feb 6th for the Tuesday Set and Feb. 8th for the Thursday Set)

Applicable Requirements

- **REQ1010** – The Enterprise Development Environment shall be prototyped on Microsoft Azure cloud infrastructure.
- **REQ1130** – The Enterprise Development Environment shall have an Orchestration capability to enable CI/CD Pipelines. The Orchestration tools will be Jenkins.
- **REQ1140** – The Orchestration capability shall integrate with the Source Code Management Capability. Jenkins will pull source code from GitLab.
- **REQ1100** – The Enterprise Development Environment shall provide access to applications on the web through a single web application server acting as a reverse proxy. The reverse proxy will be implemented using Apache2.
- Note: We will add Jenkins to the reverse proxy.
- **SEC1020** – All web applications and API endpoints shall be encrypted (i.e., https endpoints). Note: Self-signed certificates are sufficient for the prototype environment.

Group Work

You will be working on the same Azure cloud environment as the previous lab, shared with your Lab partner. This lab will be done together with your partner.

Update GitLab for Critical Patch

You have been getting a warning in GitLab to upgrade your current version to get a security patch. Please do this now before installing Jenkins.

https://docs.gitlab.com/ee/update/

You want to upgrade based on installation methods, which was Linux Packages (Omnibus) for us.

Make sure to upgrade to the CE version (not EE) - gitlab/gitlab-ce

When logged in to the GitLab, check to make sure you are on the latest version (On the left sidebar, at the bottom, select Help).

Orchestration Tool (Jenkins) Installation and Setup

Step 1 – Create a Virtual Machine in Azure

Create a Linux Virtual Machine (using the Virtual Machines Service) with the following specifications under your Azure for Students subscription:

- Resource group name: Jenkins
- Virtual machine name: Jenkins

- Region: (US) West US (or another region where a B2s is available)
- Image: Ubuntu 20.04 LTS
- Size: B2s
- Authentication Type: SSH public key (create an SSH key if necessary and paste in the public key)
- Inbound Ports: Allow SSH (22) only

Leave everything else at the defaults and Create the Virtual Machine. This may take a few minutes.

Create a DNS Name for your Jenkins Virtual Machine that is named similar to:

jenkins-acit4850-group<group #>.<region>.cloudapp.azure.com

Add an inbound security rule that allows access to your Jenkins Virtual Machine on TCP port 8080.

Step 2 – Install Docker and Jenkins

Login to your Jenkins VM using ssh.

- Install Docker: https://docs.docker.com/engine/install/ubuntu/
- You will need to use sudo for most of the commands as they require root permissions. **Change this** so you can run docker commands as a non-root user with these postinstall steps: https://docs.docker.com/engine/install/linux-postinstall
- Create a docker folder with a dockerfile that contains the following content:

```
FROM jenkins/jenkins:lts-jdk17
LABEL maintainer="mmulder10@bcit.ca"
USER root
RUN mkdir /var/log/jenkins
RUN mkdir /var/cache/jenkins
RUN chown -R jenkins:jenkins /var/log/jenkins
RUN chown -R jenkins:jenkins /var/cache/jenkins
RUN apt-get update
RUN apt-get install -y python3 python3-pip

USER jenkins

ENV JAVA_OPTS="-Xmx4096m"
ENV JENKINS_OPTS="--logfile=/var/log/jenkins/jenkins.log --
webroot=/var/cache/jenkins/war"
```

- Build the docker image (from the docker folder):

```
docker build -t myjenkins .
```

- Start the docker container:

```
docker run -p 8080:8080 -p 50000:50000 --restart always --name=jenkins-master --mount
source=jenkins-log,target=/var/log/jenkins --mount source=jenkins-
data,target=/var/jenkins_home -d myjenkins
```

- You will need to find the initial Jenkins admin password. Get the default Jenkins admin password

```
docker exec -it <container id> /bin/bash
```

Note: You can use the container name (jenkins-master) in place of the container id.

Look at the end of the $JENKINS_HOME/secrets/initialAdminPassword file for the initial admin password. Note the administrator's username is admin.

You should now be able to go to the Jenkins web application at http://<Jenkins DNS>:8080 to complete the installation. Make sure to install the suggested plugins.

Create non-admin users for both yourself and your partner (if applicable).

Below are the key commands you'll need to build, start and stop your Jenkin Docker Image later in the lab (and in future labs):

- Stop

```
docker stop jenkins-master
docker rm jenkins-master
```

- Build the Image (from the docker folder)

```
docker build -t myjenkins .
```

- Start (including restart if the VM is restarted)

```
docker run -p 8080:8080 -p 50000:50000 --restart always --name=jenkins-master --
mount source=jenkins-log,target=/var/log/jenkins --mount source=jenkins-
data,target=/var/jenkins_home -d myjenkins
```

The highlighted part restarts the jenkins-master image automatically if the VM is restarted.

Note that in the above commands, myjenkins is the name of the built Docker image and jenkins-master is the name of the running container based on the myjenkins image (use "docker image ls" to see the list of built docker images on your VM).

You can check if your Jenkins container is running with the following command: `docker ps`

Step 3 – Add Jenkins to the Reverse Proxy

SSH into your Apache VM (containing Apache2 configured as a reverse proxy).

Add the following to your default-ssl.conf file for Jenkins:

```
ProxyPreserveHost On
AllowEncodedSlashes NoDecode

<Proxy *>
        Order deny,allow
        Allow from all
</Proxy>

ProxyPass         /jenkins  http://<Jenkins DNS Name>:8080/jenkins nocanon
ProxyPassReverse  /jenkins  http://<Jenkins DNS Name>:8080/jenkins
ProxyPassReverse  /jenkins  http://<Apache DNS Name>/jenkins
RequestHeader set X-Forwarded-Proto "https"
RequestHeader set X-Forwarded-Port "443"
```

Restart apache2 (i.e., sudo systemctl restart apache2).

Your Jenkins installation must include the /jenkins context path. SSH into your Jenkins VM and update the following in your dockerfile:

```
ENV JENKINS_OPTS="--handlerCountMax=300 --logfile=/var/log/jenkins/jenkins.log --
webroot=/var/cache/jenkins/war --prefix=/jenkins"
```

Run the docker stop, build and start command from Step 3 to re-build and re-start your Jenkins image.

Make sure only the 8080 port is open on your Jenkins VM and it is tied to the IP of the Apache VM (as per Confluence and JIRA).

You should now be able to access Jenkins on: https://<Apache DNS Name>/jenkins

Complete the Jenkins installation process, if you haven't already done so, on your running Jenkins web application. Use the proxy URL (https://<Apache DNS Name>/jenkins) when it asks you set the Jenkins URL. If you already set it to something else, you can update it at: Manage Jenkins -> Configure System -> Jenkins Location -> Jenkins URL. Save it once you've updated it.

Make sure to have a Jenkins user account for both you and your partner (Manage Jenkins -> Manage Users -> Create User).

Step 4 – Create a Jenkins Job and Integration with GitLab

You will need to startup your GitLab VM for this part.

GitLab:

- Login to GitLab with one of your accounts (it should be an Admin account with the ability to create Projects)
- Download the SampleCode.zip from D2L Week 4 (Course Content) and extract it to a folder on your machine.
- Create a new project called **point** in your Prototypes group in GitLab. Make it an internal repository.
- In the Project Overview for the **point** project, follow the "Push an existing folder" instructions to push the sample code to this project.
- Create an Access Token for your account (User Dropdown -> Settings -> Access Tokens)
    - Give it a name of jenkins_svc
    - Check the read_repository and write_repository scopes

    - Create the personal access token and save the token value (you cannot get it again without creating a new acccess token).

Jenkins:

- Login to Jenkins with your account.
- Create a new build job:
    - Select New Item
    - Name it “Point”
    - Select Freestyle Project
    - Press OK
- Configure the build job:
    - In Source Code Management, select Git
    - Repository URL is the https URL of you Point project from GitLab (i.e., clone URL)
        - You should see an error that it cannot connect to the repo
    - Select Add -> Credentials and enter your Access Token name and Password. Give it an ID of jenkins_svc.
        - The error should go away if the Access Token is correct
    - Press Save
    - **Note: The name of the default branch in GitLab *may* have changed from master to main. You will have to change the branch name in the Git configuration in this Jenkins build job to match yours.**
- Run the build job:
    - Select Build Now
    - It should be successful after a few seconds
    - View the Workspace. All the files from the GitLab project should be in the Workspace.
- Add a new build step
    - Select Configure
    - Under Build, select Add Step -> Execute Shell
    - Add the following command:
        - python3 -m unittest test_point_manager
    - Press Save
- Run the build job:
    - Select Build Now
    - It should fail after a few seconds
    - View the console output for the build job
        - It is failing on the unittests because it is missing SQLAlchemy
- Your Jenkins image is missing the SQLAlchemy Python package. SSH into your Jenkins VM and add the following to the your dockerfile under user root **but before the user is changed to jenkins**:

```
USER root
...
RUN apt-get update
RUN apt-get install -y python3 python3-pip
RUN pip3 install SQLAlchemy
RUN pip3 install --upgrade pip
```

- Run the docker stop, build and start command from Step 3 to re-build and re-start your Jenkins image.
- Re-run the build job. It should now be successful.
    - View the Console Output to make sure the unittests have run and all passed.
    - To see the Console Output, select on the build job in the History and the Console Output option should appear on the left side.

<u>Make sure you shutdown any Azure resources (i.e., the VM) to conserve your credits and free tier usage.</u>

Demo, Grading and Submission

A demo of your lab against the applicable requirements which will determine your grade on the lab. All mandatory requirements must be met otherwise you will receive zero on the lab. You can re-demo the lab if you haven't met the mandatory requirements, up to the last class before the midterm week, but you will lose 20% every week late.

Reqt.	**Mandatory**	**Demo**	**Marks**
	Yes	Show that you have upgraded GitLab to the latest version of CE	1
REQ1010	Yes	• Show your Azure dashboard (both students). • Show your running VM for the Jenkins application with DNS name.	1
REQ1100	Yes	• Show your Jenkins instance is accessible through the reverse proxy with SSL enabled.	1
REQ1130	Yes	• Demonstrate that both you and your partner can login to your Jenkins installation with your own accounts.	2
REQ1140	Yes	• Demonstrate that you have put the SampleCode into a GitLab project on your GitLab installation. • Demonstrate that you have a Jenkins job that pulls the SampleCode from GitLab and successfully runs the unit tests.	3
SEC1020	No	• Your reverse proxy is configured to use an SSL self-signed certificate. • Requests to URLs with http on port 80 are Redirected to https on port 443.	1
		Demonstrate that you can run docker commands without sudo (i.e., docker ps showing your running Jenkins container)	1
Total			**10**

Upload a screenshot of your Jenkins Job showing a successful run to the Lab 4 dropbox on D2L.