Network Management and Automation

Lab 9

CI/CD with Jenkins

University of Colorado Boulder

Network Engineering Program

Professor Levi Perigo, Ph.D.

# Summary

As the network scale increases, so does the complexity of the code that manages, monitors, and configures the network. The CI/CD tools make it easier to continuously integrate and deploy Infrastructure-as-Code (IaC) in a cost-efficient manner, with higher product quality, and shorter release cycles. As the name suggests, one of the benefits of CI/CD tools is the Continuous Deployment of networks. The purpose of this lab is to understand the DevOps cycle by learning Jenkins, integrating it with GitHub, and configuring Jenkins Pipeline to continuously deploy networks via code. The other purpose of this lab is to use unit tests to make sure your code is meeting the pre-defined requirements, which in turn, indicates your code is ready for production. The objectives of this lab are to be used as guidelines, and additional exploration by the student is strongly encouraged.

# Objectives

1. Learn about Jenkins functionalities
2. Learn how to use version control (GitHub)
3. Learn how Jenkins and GitHub work together
4. Learn how to write Jenkins Pipeline
5. Learn how to write a clean and readable code that follows the PEP8 style
6. Learn how to write Unit Tests
7. Learn how to run Jenkins jobs (both push and pull methods)

# Objective 1 – Getting started

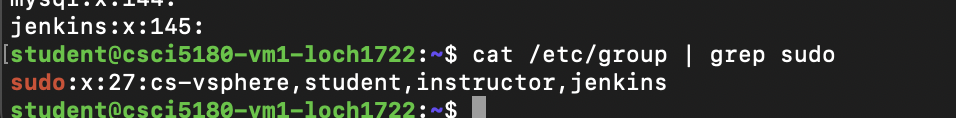
1. Install Jenkins on the NetMan VM or on your local system. Just make sure you have connectivity to GNS3 devices. Provide relevant screenshots of Jenkins status and Jenkins console. **[10 points]**

A computer screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Provide Jenkins super-user permission to avoid using ‘sudo’ for the upcoming objectives. Provide a screenshot. **[5 points]**
2. Create a GNS3 topology provided for objective 2 in the previous Netconf lab. Configure Management IP addresses and NETCONF over SSH on all routers for the upcoming objectives. Paste relevant screenshots **[10 points]**

Here is the general config for the devices, Ips and hostnames are changed accordingly:

hostname R1

ip domain-name R1

crypto key generate rsa

ip ssh version 2

line vty 0 4

transport input ssh

login local

exit

netconf ssh

netconf lock-time 60

netconf max-sessions 5

netconf max-message 99999

int fa 0/0

ip address 192.168.122.11 255.255.255.0

no sh

Topology:

A diagram of a cloud computing network

Description automatically generated

1. You will be using the script ‘netman\_netconf\_obj2.py’ and ‘info.csv’ provided to you (they are present in this directory) for the upcoming objective 2.2. (This script achieves objectives mentioned in objective 2 of the previous Netconf lab)

# Objective 2 – Creating Jenkinsfile for the pipeline

We will use Jenkinsfile to build our pipeline. The requirements of Jenkinsfile are:

1. It should trigger the Jenkins jobs whenever we push code to the repository. **[10 points]**

The way I configured this is in objective 3

1. It should have 4 stages (You can add additional stages without compromising the requirements):

My Jenkinsfile and the other .py files used are being submitted in the assignment. Each stage is labled in the Jenkinsfile. Each .py file mentioned in the Jenkinsfile as well as the .csv is provided.

1. **Stage 1: Update/Install packages in the NetMan VM/laptop** (These libraries are used in the script ‘netman\_netconf\_obj2.py’)

Install the below libraries if they are not present already.

nccclient, pandas, ipaddress, netaddr, prettytable

**[10 points]**

1. **Stage 2: Checking and fixing violations** (Checking if application code complies with PEP8 code style)

It should report the check-style violations using [pylint](https://www.pylint.org/) for the script ‘netman\_netconf\_obj2.py’ and in case of violations, the pipeline should fail. Set the quality gate to 5. On failure, violations should be fixed before proceeding further. **[25 points]**

1. **Stage 3: Running the application** (Executing the application code to be tested in the next objective for functionality)

It should run the application code ‘netman\_netconf\_obj2.py’. **[5 points]**

1. **Stage 4: Unit test** (Testing if the application is meeting the requirements)

Write unit tests using Python’s [unittest framework](https://docs.python.org/3/library/unittest.html) to test the following scenarios:

1. If the IP address of loopback 99 on router 3 is 10.1.3.1/24
2. If R1 is configured only for a single area
3. If a ping from router 2’s loopback to router 5’s loopback is successful

**[30 points]**

1. The post action should send an email from Jenkin’s admin to you informing the build success/failure. This is explained in detail in objective 4 (below) **[5 points]**

See objective 4.

1. Create a private repository in GitHub with the Jenkinsfile, the Unit test script, and the provided script along with the CSV file. Provide a relevant screenshot. **[5 points]**

Github link: <https://github.com/Logan-Chayet/Lab9.git>

A screenshot of a computer

Description automatically generated

# Objective 3 – Adding GitHub project to the pipeline and configuring GitHub webhook

1. Install GitHub plugin in Jenkins. Provide a screenshot showing that it is installed. **[3 points]**

A screenshot of a computer

Description automatically generated

1. Install Git in the NetMan VM/laptop if it is not installed already and set the path to executable in ‘Manage Jenkins’-> ‘Global Tool Configuration’->’Git’. Provide a screenshot. **[5 points]**

A white paper with blue lines

Description automatically generated with medium confidence

1. Install ‘Warnings Next Generation’ plugin in Jenkins. Provide a screenshot showing that it is installed. **[2 points]**

A screenshot of a computer

Description automatically generated

1. Install pylint in NetMan VM/laptop using pip3 and set the path to executable in ‘Manage Jenkins’ -> ‘Configure System’ -> ‘Warnings Next Generation Plugin Global Settings’. Provide a screenshot. **[5 points]**

I did not find this setting in Jenkins but I did install pylint and everything worked great and worked in my Jenkinsfile

1. Create a pipeline project from Jenkins Dashboard. Provide a screenshot. **[5 points]**

A screenshot of a computer

Description automatically generated

1. Go to your pipeline project. Under ‘General’ tab, enter your project URL.
2. In ‘Build Triggers’ tab, enable ‘Github hook trigger for GITScm polling’.
3. In ‘Pipeline’ tab:
   1. Select ‘Definition’ as Pipeline script from SCM
   2. Select ‘SCM’ as Git
   3. Provide repository URL along with a method of credentials (of your choice).
   4. Enter the branch name in ‘Branches to Build’

Save the changes and provide relevant screenshots. **[10 points]**

A screenshot of a computer

Description automatically generated

1. Configure GitHub Webhook:

GitHub integration with Jenkins needs a public URL for the webhook. Use [ngrok](https://ngrok.com/download) to create a third-party tunnel for the same.

Go to your GitHub repository settings and in ‘webhook’ tab, provide:

1. Newly generated URL in ‘Payload URL’ field
2. ‘Content Type’ as application/x-www-form-urlencoded.
3. Enable it for ‘push’ event.

Provide a screenshot. **[10 points]**

**A screenshot of a computer

Description automatically generated**

**Note:** Do not use this URL to access Jenkins (it does not work properly for this purpose), use this just to integrate GitHub with Jenkins

# Objective 4 – Configure E-mail notifications

1. Go to manage Jenkins -> Configure system -> Configure E-mail notification and extended E-mail notifications. Provide a screenshot of configurations **[10 points]**

A screenshot of a login page

Description automatically generated

A white lined paper with black text

Description automatically generated

1. Hit ‘Test configuration by sending test e-mail’ in E-mail notifications and provide screenshot of receiving an email**. [2 points]**

A screenshot of a computer

Description automatically generated

# Objective 5 – Running Jenkins Job

1. Go to Jenkins and hit ‘build now’. GitHub webhook needs one build to be run manually first to trigger the builds later automatically. Provide a screenshot of console output and a SUCCESSFUL build. If your build fails, work to fix it. **[10 points]**

A screenshot of a computer

Description automatically generated

1. Now, make a small change in any file. Commit it to GitHub. The job should start automatically. Show the console output of your next Jenkins job succeeding. The console output should show ‘started by GitHub push…’ **[10 points]**

A screenshot of a computer

Description automatically generated

1. Provide a screenshot of receiving an email about a SUCCESSFUL build. **[10 points]**

A screenshot of a computer

Description automatically generated

# Objective 6 – Schedule Nightly Jobs

Configure Jenkins to run GitHub jobs every night.

1. Provide a relevant screenshot. **[10 points]**

A screenshot of a computer

Description automatically generated

1. Briefly explain the significance of this. **[5 points]**
   1. It is important to do this to make sure the code is updated, and everything is running well. This is like a health check to make sure things are running how they are supposed to.

# Report Questions [15 points]

1. Do you think Jenkins is actively used in the industry? Why?
   1. I think Jenkins is actively used in the industry because it is a great way to manage and verify code that could potentially change high level networks in production. Combining Jenkins with GitHub is perfect to introduce change management as well.
2. What is another CI/CD tool you would like to learn and why?
   1. I would like to learn GitHub Actions because in my Computer Science capstone group, we use GitHub actions to make sure our backend is running and executing as we want it. I am not directly involved in the work for the GitHub Actions set up, but I would love to learn more about it because I have only heard good things from it.
3. Where else could you run your Jenkins jobs?
   1. For example if I was running a website I could set up jobs to check that my website is up and traffic can get to the destination.
4. Explain the use of Unit Testing in Jenkins.
   1. The use of Unit Testing in Jenkins is to check that your code is doing what it is supposed to. It can check for edge cases that makes the code more robust and less likely to fail in production.
5. What other tests can be carried out before an application deployment?
   1. A test could be done to check that code deployed in a Lab environment mirrors that in a real world lab environment before going to production.
   2. If the network is set up for redundancy, Jenkins can test for when a device goes offline or changes some sort of metric and guarantee network connectivity.

# Total Points \_\_\_\_\_\_\_ / 226