**DevOps CICD using Jenkins, Ansible & Docker**

Rishi Ravi

[**https://freelabs.nuvepro.io**](https://freelabs.nuvepro.io)

[**https://nuvepro.com**](https://nuvepro.com)

**freelabs@nuvepro.com**

Contents

[Overview 2](#_Toc75593323)

[Know your lab 2](#_Toc75593324)

[Lab Software 4](#_Toc75593325)

[Why Docker? 4](#_Toc75593326)

[Docker Image & Container 5](#_Toc75593327)

[About Demo Projects 5](#_Toc75593328)

[Manual Build & Deploy 6](#_Toc75593329)

[Configure Jenkins 6](#_Toc75593330)

[Manual Build & Deploy using Jenkins 7](#_Toc75593331)

[Continuous Integration & Continuous Deployment 7](#_Toc75593332)

[CI using Jenkins & GIT 7](#_Toc75593333)

[Ansible 8](#_Toc75593334)

[Continuous Deployment using Ansible 8](#_Toc75593335)

[CICD & CICD Pipeline 9](#_Toc75593336)

[CICD 9](#_Toc75593337)

[Building CICD Pipeline 9](#_Toc75593338)

[Miscellaneous 10](#_Toc75593339)

[Additional Software Installation 10](#_Toc75593340)

[Reference 10](#_Toc75593341)

# 

# Overview

The DevOps CICD Lab is a multi VM lab framework having two machines – a Jenkins Server and an Application Node. Both the machines are preloaded with all required software for you to build your Jenkins CICD pipeline.

In this session we will go hands-on!! Integrating the code from source repo, building it each time for every commit / per the schedule, deploying the application on one or more nodes and run automation script to test the application.

# Know your lab

Once you start the lab, you can choose one of the lab names from the drop-down in the left corner of the lab frame, which loads the lab. At a time only the chosen lab will load in the same page.

Graphical user interface, application

Description automatically generated

There are a few other options to Expand / Collapse lab window, Refresh, Maximize, Pop-out and Stop the lab, in the right corner of the lab frame.

A screenshot of a computer

Description automatically generated with medium confidence

You can further maximize that lab. You can also pop the lab, by choosing one by one lab from the drop-down, out to a new tab and work on both the labs.

Graphical user interface, application

Description automatically generated

## Lab Software

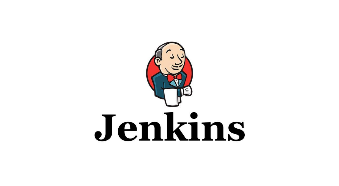
A picture containing tool

Description automatically generatedLogo, company name

Description automatically generatedLogo, company name

Description automatically generated

Logo, company name

Description automatically generated

Logo, company name

Description automatically generatedLogo, company name

Description automatically generatedA picture containing diagram

Description automatically generated

# Why Docker?

Graphical user interface, application

Description automatically generated

* Solve the “*it works on my machine*” headache.
* Became an Industry Standard.
* Isolate app from its environment.
* De facto standard to build and share containerized apps.

# Docker Image & Container

* Create, deploy, and run applications by creating docker image layers, finally to create a custom docker image of our app.
* We will use Dockerfile to create our custom docker image.

Logo

Description automatically generated

* By running the instance of the docker image, we create a docker container to make our app live.
* We will make the docker container up and running using one of these options.
  + docker container run
  + docker-compose

# About Demo Projects

* We have two demo projects – one that can be deployed on Apache Tomcat image and another one using an Alpine image, and a test automation project to test the deployed applications.
  + <https://github.com/nuvepro-hands-on-labs/DevOps-CICD-Demo-WebAppJAR.git>
  + <https://github.com/nuvepro-hands-on-labs/DevOps-CICD-Demo-WebAppWAR.git>
  + <https://github.com/nuvepro-hands-on-labs/Test-Automation-CICD-Demo.git>
  + [https://github.com/nuvepro-hands-on-labs/DevOpsCICD-Demo-Commons.git](https://github.com/nuvepro-hands-on-labs/DevOps-CICD-Demo-WebAppJAR.git)
* The deployment configurations are also included in the project folder.



* All projects are in the public space of nuvepro-hands-on-labs, which you can clone and make use of.
  + git clone <https://github.com/nuvepro-hands-on-labs/DevOps-CICD-Demo-WebAppJAR.git>
  + git clone <https://github.com/nuvepro-hands-on-labs/DevOps-CICD-Demo-WebAppWAR.git>

# Manual Build & Deploy

* Build the application jar using Command Prompt / Eclipse IDE.
  + mvn clean install
  + java -jar demo-0.0.1-SNAPSHOT.jar
  + mkdir -p ./deploymentconfig/target && cp ./target/demo-0.0.1-SNAPSHOT.jar ./deploymentconfig/target
* Build the Application Docker Image using the Dockerfile.
  + sudo docker image build -t webappjar **.**
* Create the Application Docker Container for each.
  + General command: sudo docker run --publish hostport:containerport --name container\_name --detach image\_name
  + sudo docker container run --publish 8083:8083 --name webappjar --detach webappjarAccess localhost link and see the application.
  + <http://localhost:8083>
* Install docker-compose.
  + sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  + sudo chmod +x /usr/local/bin/docker-compose
  + sudo docker-compose -version
* Create docker network
  + sudo docker network create webnet
* Use docker compose to run the container.
  + sudo docker-compose -f webappjar-docker-compose.yml --env-file webappjar.settings.qa.env up -d
* Use docker compose to stop the container.
  + sudo docker-compose -f webappjar-docker-compose.yml --env-file webappjar.settings.qa.env down

# Configure Jenkins

* Configure initial plugins and Admin user
* Go to – Manage Jenkins – Global Tool Configuration
  + Configure Java
    - echo $JAVA\_HOME
    - /usr/lib/jvm/java-8-oracle
  + Configure Maven
    - echo $MAVEN\_HOME
    - /opt/maven
  + Configure GIT
    - which git
    - /usr/bin/git
* Go to – Manage Jenkins – Manage Plugins
  + Install the plugin, Publish Over SSH.
    - After plugin installation, go to – Manage Jenkins – Configure System
    - Configure the Application Node details in Jenkins to Publish over SSH.
      * Find SSH Servers – Click Add
      * Set the following
        + SSH Server Name – friendly name to identify.
        + Hostname
        + Username
        + Remote Directory
        + Click Advanced and set use password authentication.

Enter only the password and test configuration.

* + - * Save the changes.
  + Install other plugins
    - GitHub
    - Build Pipeline Plugin
    - TestNG

# Manual Build & Deploy using Jenkins

* Create a new item in Jenkins and configure it with the project in GIT.
* Add build action – Maven command.
  + clean install
* Add post build action to deploy the application to the Application Node.
  + Send artifacts over SSH
  + Choose the Web Server
  + Add required commands to deploy application.
    - cd /home/ubuntu/deploy-webjar/deploymentconfig
    - sudo docker-compose -f ./webappjar-docker-compose.yml --env-file ./webappjar.settings.qa.env down
    - sudo docker rmi webappjar
    - sudo docker network rm webnet
    - sudo docker image build -t webappjar .
    - sudo docker network create webnet
    - sudo docker-compose -f ./webappjar-docker-compose.yml --env-file ./webappjar.settings.qa.env up -d
* Try building the project manually from Jenkins.
* Note: Replace these commands with single Ansible command later to build docker images and deploy into single or more than one node.

# Continuous Integration & Continuous Deployment

## CI using Jenkins & GIT

* Configure Webhook settings in Jenkins & GIT.
  + Run ngrok kept in /home/ubuntu using the command – /home/ubuntu/ngrok http 8080, to expose Jenkins localhost 8080.
  + Or cd /home/ubuntu and run ./ngrok http 8080
  + Running NGROK expose the local Jenkins server (http://localhost:8080/) to the internet, so that the http / https URL that the ngrok gives can be configured in GIT code repository webhook settings.
  + Override the default localhost hook URL using the NGROK URL in Jenkins.
  + Go to project repository – Settings – Webhook (Settings left pane) – Manage Webhook – Payload URL. Replace the payload URL.
  + ngrok session will end after 2 hours, post which we must relaunch and reconfigure the webhook settings in both Jenkins and Git.

## Ansible

* Simple, agentless IT automation engine, which helps in automating execution of any commands in multiple nodes.
* Ansible connects to the nodes and push out small programs, called Ansible Modules to them.
* Ansible does the job with the help of playbook yml files and the inventory host, which has all the node details.
* The machine having Ansible installed is called the Controller node.
* By default, it uses SSH to connect other nodes.

## Continuous Deployment using Ansible

* Jenkins server is also called a Controller server since it has the Ansible.
* The playbook and host files are also available in the deployment configuration folder of each project.
* Add user ‘ubuntu’ to sudoers list for sudo access
  + echo "ubuntu ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
* Configure Ansible execution command in Jenkins.
  + sudo ansible-playbook -i ./qahost-single ./webappjar-deploy-qa-playbook.yaml
  + sudo ansible-playbook -i ./qahosts-multiple --private-key=/home/ubuntu/.ssh/id\_rsa ./webappjar-deploy-qa-playbook.yaml --limit [ubuntu@172.31.25.94](mailto:ubuntu@172.31.25.94)
  + sudo ansible-playbook -i ./deploy-webjar/deploymentconfig/qahosts-multiple --private-key=/home/ubuntu/.ssh/id\_rsa /home/ubuntu/deploy/deploymentconfig/webappjar-deploy-qa-playbook.yaml -u ubuntu
* Setting seamless SSH connectivity: SSH private key
  + ssh-keygen
  + ssh-copy-id [ubuntu@172.31.25.94](mailto:ubuntu@172.31.25.94)

Diagram

Description automatically generated

# CICD & CICD Pipeline

## CICD

* Make changes to the code and commit the changes.
* Check Jenkins to see the Build & Deployment of application.
* Docker compose commands for scaling docker chrome container up / down.
  + docker-compose up -d
  + docker-compose scale chrome=3
  + docker container ls | grep selenium/node-chrome:3.141.59-20200525 | wc -l
  + docker-compose down
* Access application and see the changes.

## Building CICD Pipeline

A picture containing diagram

Description automatically generated

* Build a pipeline to trigger one by one projects building, deploying, and running automation scripts to test the applications.
* Whenever the commit happens see how different projects in the pipeline is getting build, the applications getting installed and running automation scripts to test the deployed application.

# Miscellaneous

## Additional Software Installation

* **docker-compose**
  + sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  + sudo chmod +x /usr/local/bin/docker-compose
  + sudo docker-compose -version
* Re. **Configure Jenkins** for details of Jenkins related plugins installation.

## Reference

* **Jenkins**
  + Installation: <https://www.jenkins.io/doc/book/installing/linux/>
  + Java app with Maven: <https://www.jenkins.io/doc/tutorials/build-a-java-app-with-maven/>
  + Jenkins Pipeline: <https://www.jenkins.io/doc/pipeline/tour/hello-world/>
* **Docker**
  + Installation – <https://docs.docker.com/get-docker/>
  + Ubuntu: <https://docs.docker.com/engine/install/ubuntu/>
  + Dockerfile: <https://docs.docker.com/engine/reference/builder/>
  + Docker-Compose: <https://docs.docker.com/compose/>
  + Docker-Compose Environment: <https://docs.docker.com/compose/environment-variables/>
  + Docker Container Run: <https://docs.docker.com/engine/reference/run/>
* **Ansible**
  + Installation: <https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html/>
  + Overview: <https://www.ansible.com/overview/how-ansible-works/>
  + Documents: <https://docs.ansible.com/?extIdCarryOver=true&sc_cid=701f2000001OH7YAAW/>
  + Playbook: <https://docs.ansible.com/ansible/latest/user_guide/playbooks.html/>