Group Project Abstract

Devops CI/CD Pipeline – SDN Orchestration Automation

Team Members:

1) Sharang Nadkarni
2) Pranith Gadapa
3) Ashish Shahane
4) Jay Kharwadkar

Our pipeline procedure is triggered when python code is committed to a repo hosted in our laptop. Next comes notification to a build system, such as Jenkins. The build system compiles the code and runs unit tests. our project will include a job that pulls the latest code from the repository and builds it.

The purpose of the dev environment is to provide insight into what is currently on the **develop branch**, or whatever branch is intended to be in the next "release" of the code.

The purpose of the QA environment is to provide a more stable and complete version of the code for the purpose of QA testing and perhaps other kinds of approval.

The purpose of the prod environment is to host production-ready code that is currently on the **master branch** (or whatever branch you use for this purpose). This represents what can be made available to users, even if the *actual* production environment is hosted elsewhere. The code in this branch is **only** what has already been approved in the QA environment with **no additional changes**.

This build should:

1. check out the specific commit (or ref)
2. build the code as usual
3. tag the commit in Git
4. push the tag to the origin repo
5. (optional, but likely) deploy it to a server

Jobs are the runnable tasks that are controlled and monitored by Jenkins. Examples of jobs include compiling source code, running tests, provisioning a test environment, deploying, archiving, posting build jobs such as reporting, and executing arbitrary scripts.

In a few steps, I will create a job that builds a Java project using the Maven build automation tool. Maven allows developers to automate the process of creating an initial folder structure for a Java application and performs the compilation, testing, and deployment of the final product.

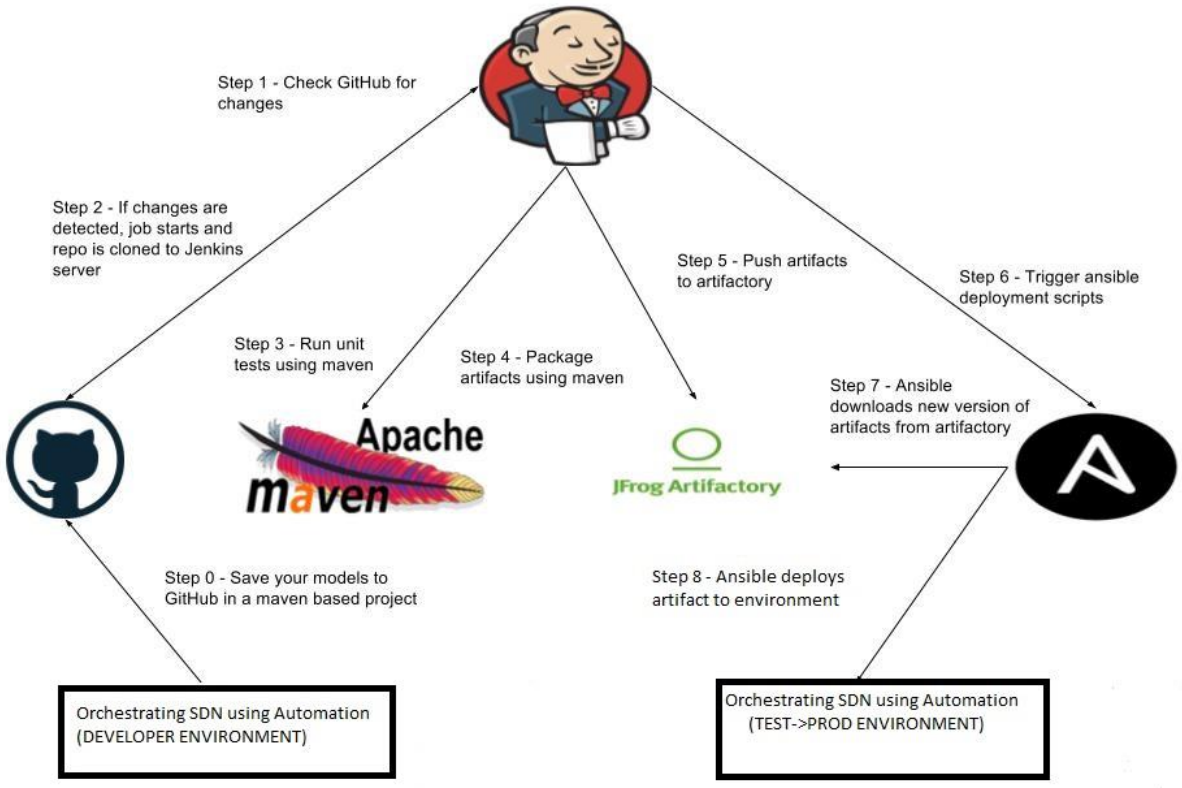
Our specified pipeline will consist of several steps that can be grouped in several stages. For example:

Stage 1	Pull code from repository	Pull Code CheckIn
Stage 2	Build your project and artifacts	Build Project/Artifacts of Project
Stage 3	Deploy your application	Deploy from centralized repo to specified environment
Stage 4	Perform functional tests	QA performance will happen
Stage 5	Perform performance tests	· (UI and performance can test.

flow of the build pipeline for demonstration purpose:

1. **Dev-Deployment**
2. **Dev-Testing**
3. **UAT-Deployment**
4. **UAT-Testing**
5. **Prod-Deployment**
6. **Prod-Testing**

Architecture:



Following will be the applications:

First we will setup environment which required for this project. Basically, first we will decide our tools and targets which is initial step of any development here we are going to use Agile software development rules to implement our project.

Tools:

1.

OpenvSwitch - Openvswitch is a virtual switch which is used in automation of virtual environment in protocol such as OpenFlow.

2.

Wireshark -This useful tool to get deep insight of any packet and working if any protocol. https://www.wireshark.org/

3.

iPerf: Measure Performance (Bandwidth, Delay Jitter, packet loss) - https://sourceforge.net/projects/iperf/

4.

RYU Controller:

Ryu Controller is an open, software - defined networking (SDN) Controller.: This is mastermind of our project and all controlling will happen through this tool .https://ryu.readthedocs.io/en/latest/

5.

MiniNet -MiniNet creates a realistic virtual network, running real kernel,

switch and application code, on a single machine (VM, cloud or native), in

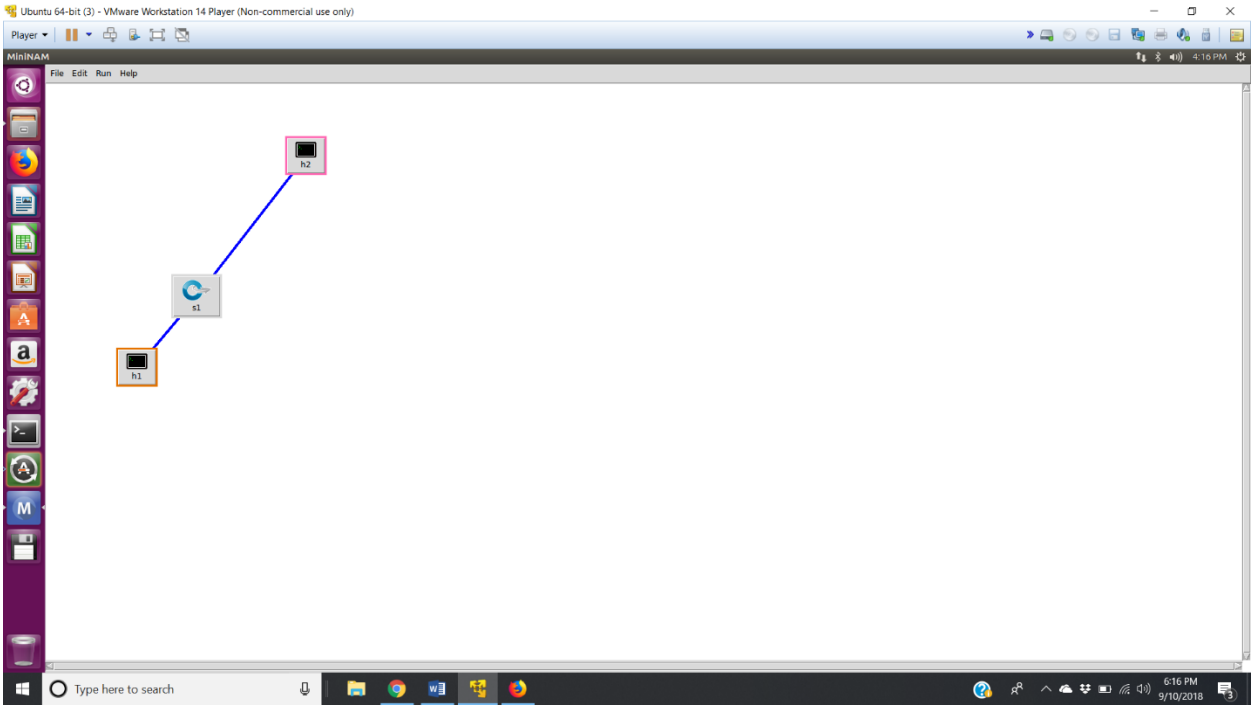
seconds. For this we are using different python codes with different topologies combinations to

observe flow of packets. We are going to use CLI of MiniNet to get operations done.

6.

MiniNAM - A network Animator to visualize realtime packet flows in

MiniNet .



Tasks:

1. **Set up Ubatu Machine:**
    - With help of Version control i.e. Git, we will deploy Vagrant.
    - With help of Vagrant we will deploy ubuntu machine
    - Ubuntu will handle SDN related task

2. **Ubuntu Machine:**
    - Install tools which mentioned in above like RYU controller and Wireshark etc.
    - Perform different operations and flows monitoring using a controller and Open Vswitch.
    - Simulate Python code and observe effect through Mininet
3. **Use of DevOps tools:**
    - After that, we want to make changes in the code and want to add some more nodes in topology using a code and test forwarding of packets. For that we need one more branch-like new release branch separated form origin branch. So here we will perform branching and merging
    - So, in the new branch, we will add new node like router through coding and with help of Mininet CLI and OpenVswitch we will assign flow between nodes. Along with that, we want to check basic networking between nodes like static switching, packet forwarding.
    - We will use Jenkins for CM during this project

Roles:

- Sharang Nadkarni: CI/CD tasks
- Pranith Gadapa: SDN Implementation Tasks
- Ashish Shahane: Testing and Deploying
- Jay Kharwadkar: Monitoring and flow checking in SDN and Agile process.

Conclusion:

In this project, we are going to use DevOps tools to Develop SDN controller and observe its operations. With help of DevOps tools, we are going to dig deep into concepts like merging and branching, CI/CD. Automation through DevOps tools and Source control and version control process.