



kura labs

COHORT 3

Deployment #3

Kerri Smith

18th October 2022

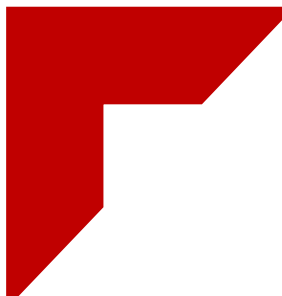


Table of Contents

Overview 1

Pipeline 3

VPC..... 4

Local VMs..... 5

STACK 6

Screen Capture 7

Proposed Improvements 9

Overview

This deployment exercise demonstrated the steps for setting up a basic CI/CD pipeline deployment to a custom AWS VPC.

Pipeline deployment

The software application used in this case was a Flask web application called “url shortener”

GitHub was used to manage the code and Jenkins was used to automate the following stages:

1. Build
2. Test
3. Clean
4. Deploy

Issues:

There were issues encountered in the initial deployment due to:

1. The configuration of the nginx server “/etc/nginx/sites-enabled/default” file.

initial instructions:

```
server {  
    listen 5000
```

The errors encountered included requesting browser not being able to access dependencies in subfolder (e.g. css files and js files)

correction:

```
server {  
    listen 5000 default_server;  
    listen [::]:5000 default_server;
```

(See Deployment_3-Assignment page 11)

2. The script in the Jenkins file did not successfully allow the application to continue running at the end of a “successful” deployment.

Correction: download the Jenkins plugin “Pipeline Keep Running Step”

Use revised script in Jenkinsfile: (See Deployment_3-Assignment page 12)

VPC

The deployment environment used two Amazon Virtual Private Clouds (Amazon VPCs):

1. The default VPC created with the AWS account
2. A custom VPC called “Kura-VPC”. This VPC consisted of
 - a. Two availability zones
 - b. Two private subnets and one public subnet
 - c. An internet gateway
 - d. One EC2 which was used to run the Jenkins agent and deploy the application

Issues:

Extended time was used running EC2 to trouble shoot initial issues with deployment. This resulted in the AWS free tier being exceeded. The revised deployment instructions were simulated using Virtual Machines and Virtual Network on a local computer. Since both VMs were on the same LAN the connection between the Jenkins Server and the Jenkins agent used “Username with password” instead of “SSH Username with private key”.

The image displays two screenshots of the Jenkins configuration page for an agent named 'awsDeploy'. The top screenshot shows the 'Configure' tab with fields for Name, Description, Number of executors, Remote root directory, Labels, Usage, Launch method, and Host. The bottom screenshot shows the 'Advanced' configuration options, including Host, Credentials, Host Key Verification Strategy, and Availability.

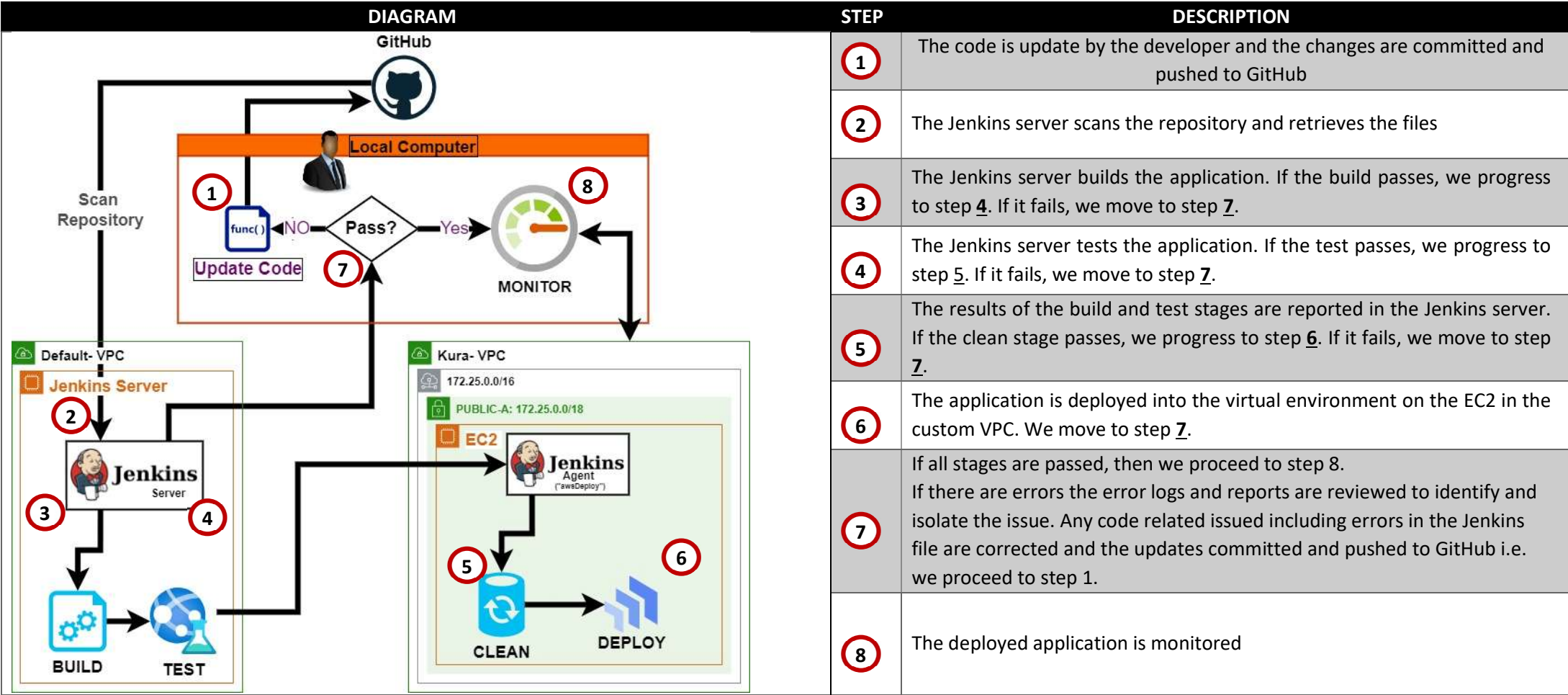
Top Screenshot (Configure Tab):

- Name: awsDeploy
- Description: Deployment server
- Number of executors: 1
- Remote root directory: /home/kerri/agent
- Labels: awsDeploy
- Usage: Only build jobs with label expressions matching this node
- Launch method: Launch agents via SSH
- Host: (empty)

Bottom Screenshot (Advanced Tab):

- Labels: awsDeploy
- Usage: Only build jobs with label expressions matching this node
- Launch method: Launch agents via SSH
- Host: 192.168.1.201
- Credentials: kerri/*****
- Host Key Verification Strategy: Non verifying Verification Strategy
- Availability: Keep this agent online as much as possible

Pipeline



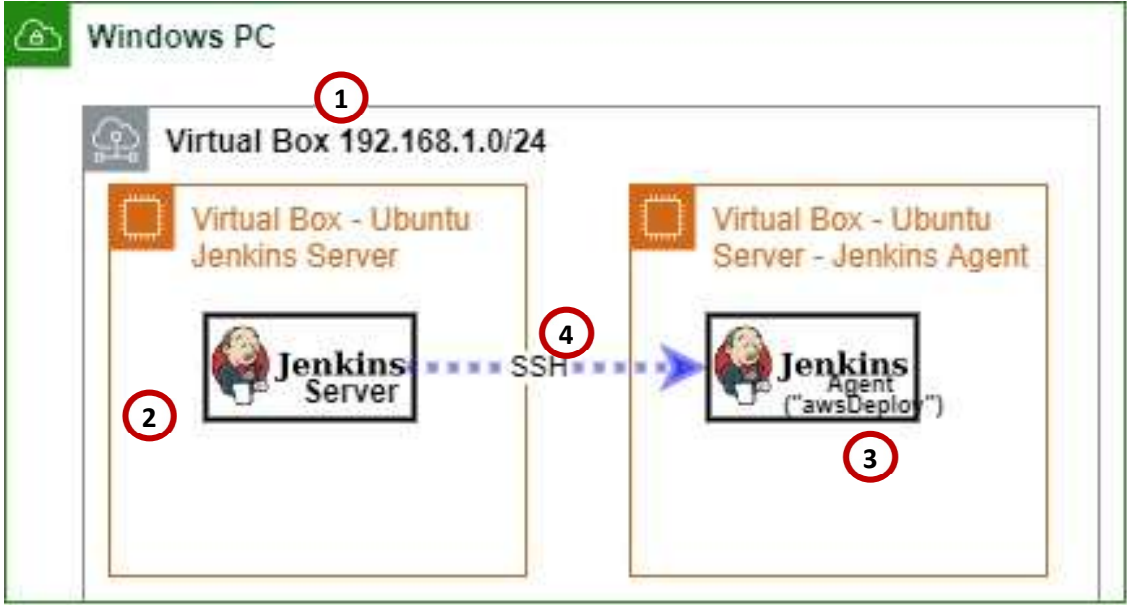
VPC

The deployment environment used two Amazon Virtual Private Clouds (Amazon VPCs): The default VPC created with the AWS account and a custom VPC called “Kura-VPC”

DIAGRAM		ITEM	DESCRIPTION
<p>The diagram illustrates the network architecture. On the left, the 'Default-VPC' in the 'us-east-1a' availability zone contains a 'Jenkins Server' in the '172.31.0.0/16' subnet, connected to an 'INTGW' (Internet Gateway). On the right, the 'Kura-VPC' also in 'us-east-1a' contains three subnets: 'PUBLIC-A: 172.25.0.0/18' with an 'EC2 Jenkins Agent', 'PUBLIC-B: 172.25.128.0/18', and 'PRIVATE-A: 172.25.64.0/18'. It features a 'Kura-INTGW' and a 'Private-KURA-RT' (Private Route Table). SSH connections are indicated: from the Internet to both INTGWs, from the Jenkins Server to the Jenkins Agent, and from the Jenkins Agent to the Kura-INTGW. The Jenkins Agent is also connected to the Private-KURA-RT.</p>		1	Default- VPC: The default VPC contained an EC2 with a public IP address running the Jenkins server
		2	Kura-VPC The Kura-VPC contained three subnets. Two public subnets (PUBLIC-A, PUBLIC-B) in two availability zones and one private subnet. For this deployment only one EC2 with a public IP address in public Subnet PUBLIC-A was used
		3	Jenkins Agent The Jenkins agent was deployed on the EC2 in the Kura-VPC
		4	SSH connection was used by the Jenkins server to set up the Jenkins Agent on the EC2 in Kura-VPC

Local VMs

The revised deployment instruction were simulated using Virtual Machines and Virtual Network on a local computer

DIAGRAM	ITEM	DESCRIPTION
	1	A windows 10 PC running Virtual Box was used to run two VMs for the simulation using the 192.168.1.0/24 network. Each VM was created with 4GB Ram 12GM HDD and used Ubuntu server as the operating system.
	2	Jenkins Server was installed one of the VMs
	3	Jenkins Agent The Jenkins agent was deployed on the other VM
	4	SSH connection was used by the Jenkins server to set up the Jenkins Agent on the other VM

STACK

A software stack consists of independent software components that work together to support the execution of an application. These components can include operating system(s), runtime environments, databases, etc. These components tend to function hierarchically with lower-level components enabling or supporting the functionality of higher-level components. The following diagram describes the software stack used for this deployment

DIAGRAM	ITEM	DESCRIPTION
	1	This is the base operating systems that provides access to processing and memory resources to all of the other elements of the stack
	2	The PIP virtual environment provides another layer of abstraction from the operating system for the remaining levels of the stack
	3	Flask provided the web framework. This means flask provides the tools and libraries that allow the web application to function.
	4	Gunicorn is the web application server that is used to allow the web application to run
	5	This the code that is running to serve the url shortener web application
	6	Nginx provided a reverse proxy. It shields or masks the internal web application server from interacting directly with the user. Users of the web applications send their requests to the nginx ip address and port, nginx then communicates internally with the web application server and relays the response to the end user.

Screen Capture

VPC

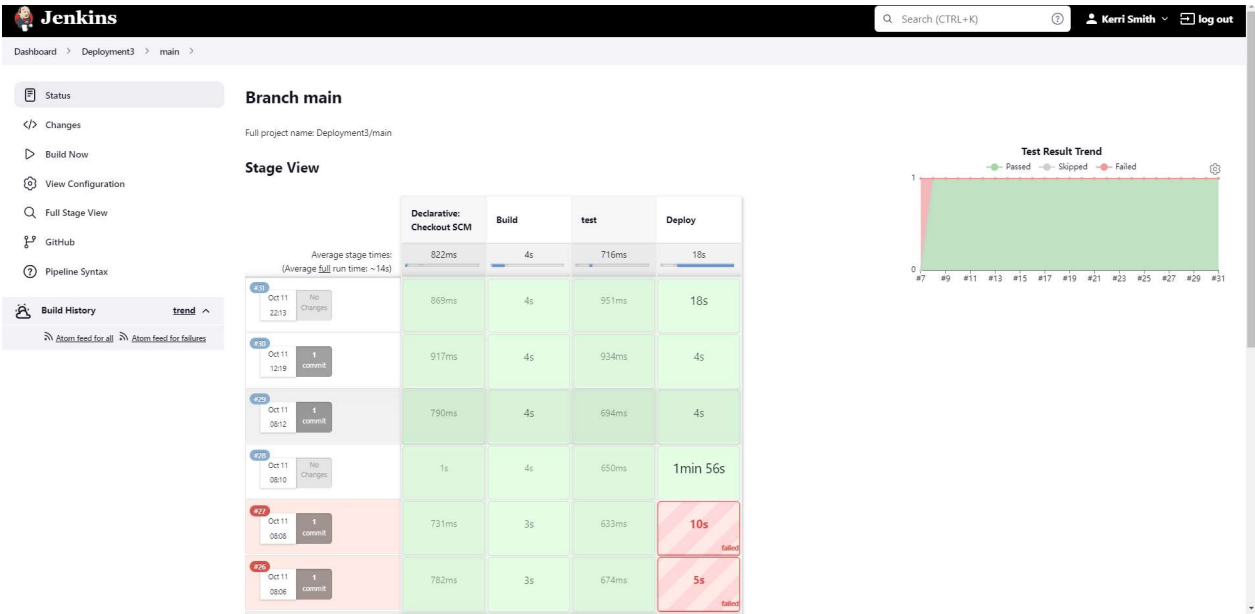


Figure 1 - Jenkins Build History – VPC

Although the Jenkins deployment was “successful” the application failed to work.



Figure 2 - NGINX failure

Local VMs

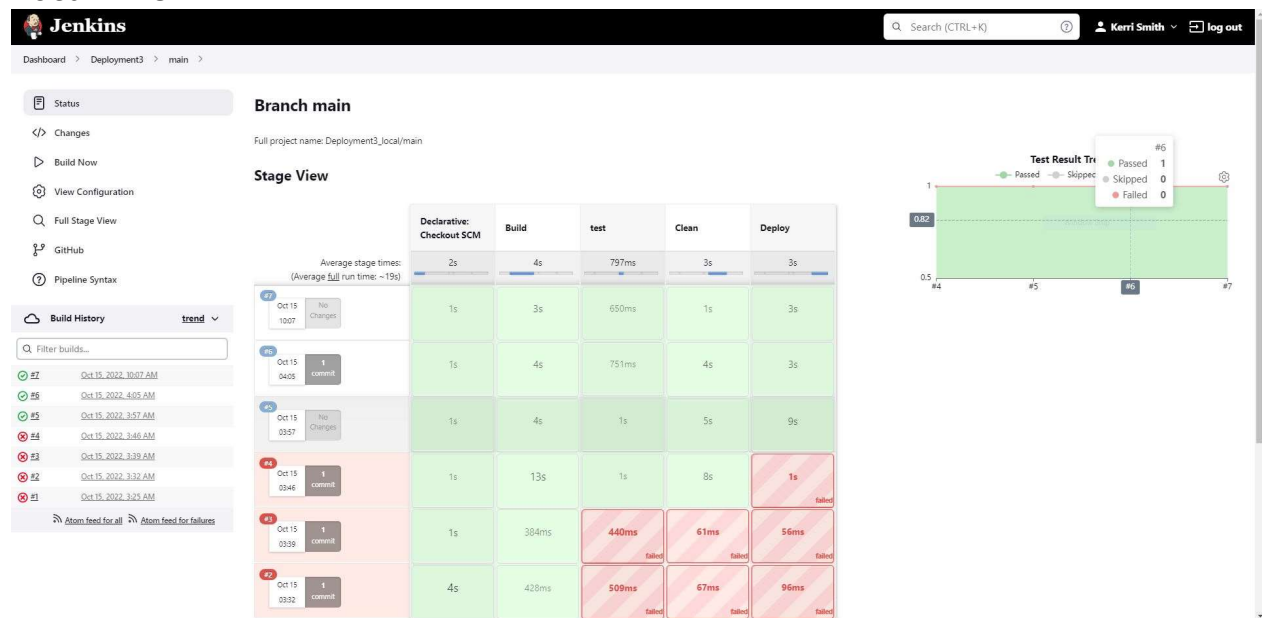


Figure 3 Jenkins Build History- Local VM

The screenshot shows the 'URL Shortener' web application. It has two main sections: 'Website' and 'File'. Each section has a 'Short Name' input field, a 'Website URL' input field, and a 'Shorten' button. The 'File' section also includes a 'Choose File' button and a 'No file chosen' message.

Figure 4 - Successful launch of the web application

Modification

The heading of the templates/base.html was changed header to "URL Shortener Dep3_2022"

The screenshot shows the 'URL Shortener Dep3_2022' web application. It has two main sections: 'Website' and 'File'. Each section has a 'Short Name' input field, a 'Website URL' input field, and a 'Shorten' button. The 'File' section also includes a 'Choose File' button and a 'No file chosen' message.

Proposed Improvements

The pipeline could be improved in the following ways:

1. Include automated monitoring and reporting after deployment
2. Include webhooks to automate deployment of updated code