The Goals of Today Lessons – Advanced Group Study

First Phase – Automation with terraform

- Setup an AWS EC2 Instance
- Connect to EC2 Instance
- Install JDK on AWS EC2 Instance
- Install and Setup Jenkins
- Install Docker
- Install and Setup AWS CLI

Second Phase - Manually – Step by Step

- Update visudo and assign administrative privileges to Jenkins user
- Install and Setup Kubectl
- Install and Setup eksctl
- Create eks cluster using eksctl

Third Phase – Manually – Step by Step

- Add Docker and GitHub Credentials into Jenkins
- Add jenkins stages
- Build, deploy and test CI CD pipeline

First Phase

Open the link below, go to your GitHub account, and fork – This is the updated git repository.

Link to the repo

https://github.com/joshking1/Updated_Terraform-for-Advanced-Class

- ❖ Configure your region
- ❖ Configure your key name
- ❖ Configure your AMI

- ❖ Configure your AWS CLI and input the Access ID and Secret Access Key
- ❖ Go to VSC and run the terraform life cycle.
- ❖ Copy the output IP address and paste it on the web browser
- ❖ Add the port :8080
- ❖ SSH to the Jenkins EC2
- ❖ Obtain the administration password and set up your Jenkins console for management purposes.
- ❖ # sudo docker exec 838 cat /var/jenkins_home/secrets/initialAdminPassword
- ❖ Check the following
- ❖ Java version #java -version
- ❖ Docker version #docker -v
- ❖ Check aws cli version - # aws –version
- ❖ Check ansible version - # ansible –version
- ❖ Check whether git is installed
- ❖ Check the python version # python –version
- ❖ Check the boto version # pip show boto3 | grep version

<u>Second Phase - Manually – Step by Step - very Important</u>

- **Update visudo and assign administrative privileges to Jenkins user** – this is vital because you cannot set up the kubectl, the command line interphase for the cluster as the root. Help you to run commands like kubectl create namespace etc

  Now we have installed the Jenkins on the EC2 instance. To interact with the Kubernetes cluster Jenkins will be executing the **shell script with the Jenkins user, so the Jenkins user should have an administration(superuser) role assigned forehand.**

  Let's **add jenkins user** as an **administrator** and ass **NOPASSWD** so that during the pipeline run it will not ask **for root password**.

Very Important

DO NOT RUN THE COMMAND **# SUDO SU**

Creating a Visudo user called Jenkins

sudo useradd Jenkins
sudo vi /etc/sudoers this is the sudoers configuration file

Add the following line at the end of the file

jenkins ALL=(ALL) NOPASSWD: ALL *BASH*

After adding the line save and quit the file.
Now we can use Jenkins as root user and for that run the following command -

sudo su - jenkins *BASH*

This what you should see

```
[jenkins@ip-10-0-1-213 ~]$ sudo history
sudo: history: command not found
[jenkins@ip-10-0-1-213 ~]$ sudo cat history
cat: history: No such file or directory
[jenkins@ip-10-0-1-213 ~]$
```

Check Docker installation

docker -v

Or

sudo apt install docker.io

Already installed for you by terraform

Add jenkins user to Docker group

Jenkins will be accessing the Docker for building the application Docker images, so we need to add the Jenkins user to the docker group.

sudo usermod -aG docker Jenkins

Install and Setup AWS CLI

Okay so now we have our EC2 machine and Jenkins installed. Now we need to set up the AWS CLI on the EC2 machine so that we can use ***eksctl*** in the later stages

Let us get the installation done for ***AWS CLI***

sudo aws –version

Or

sudo apt install awscli

Already installed by terraform

Configure AWS CLI

Okay now after installing the AWS CLI, let's configure the ***AWS CLI*** so that it can authenticate and communicate with the AWS environment.

To configure the AWS the first command we are going to run is

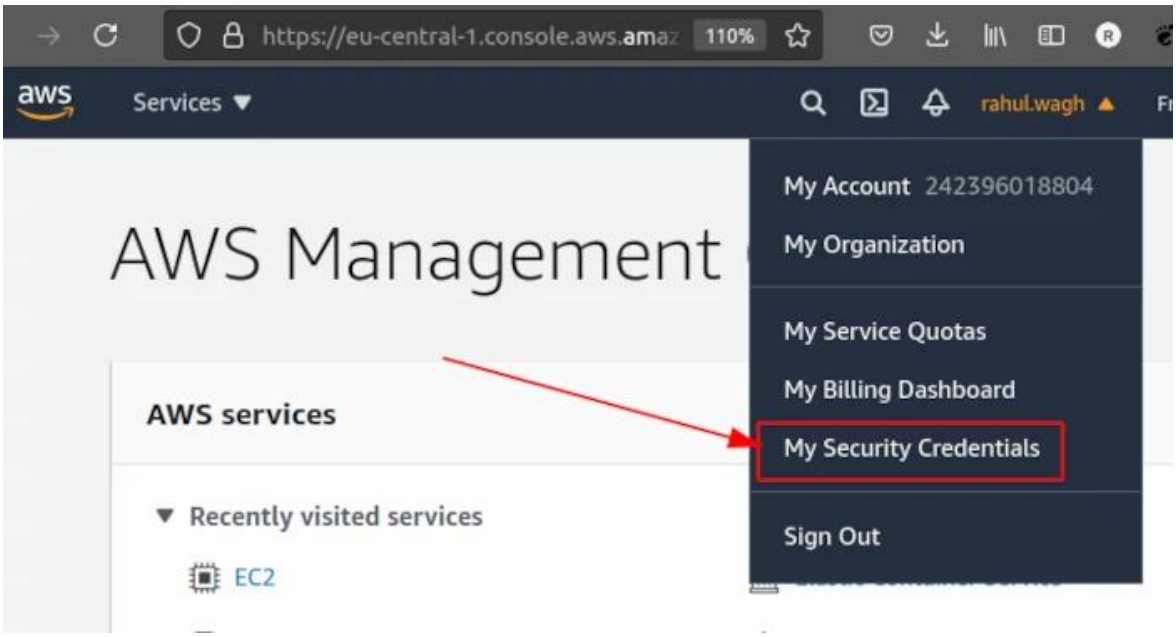
aws configure

Once you execute the above command it will ask for the following information -

1. AWS Access Key ID [None]:
2. AWS Secret Access Key [None]:
3. Default region name [None]:

4. Default output format [None]:

You can find this information by going into AWS -> My Security Credentials



Then navigate to Access Keys (access key ID and secret access key)

You can click on the Create New Access Key and it will let you generate - AWS Access Key ID, AWS Secret Access Key.

Install and Setup Kubectl

If you are root user and not Jenkins's root user, the commands that follow will not work for you. They do not provision when executed by a root user.

Moving forward now we need to set up the kubectl also onto the EC2 instance where we set up the Jenkins in the previous steps.

Here is the command for installing kubectl

curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin

Verify the kubectl installation
Verify the kubectl installation by running the command kubectl version and you should see the following output

kubectl version

Install and Setup eksctl

The next thing which we are gonna do is to install the eksctl, which we will be using to create AWS EKS Clusters.

Okay, the first command which we are gonna run to install the eksctl

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

Create eks cluster using eksctl

In all the previous 9 steps we were preparing our AWS environment. Now in this step, we are going to create EKS cluster using eksctl

You need the following in order to run the eksctl command

1. **Name of the cluster :** –name simo-cluster
2. **Version of Kubernetes :** –version 1.21
3. **Region :** -us-east-2
4. **Nodegroup name/worker nodes :** worker-nodes
5. **Node Type :** t2.micro
6. **Number of nodes:** -nodes 2

Command

eksctl create cluster --name simo-cluster --version 1.21—us-east-2 --nodegroup-name worker-nodes --node-type t2.micro --nodes 2

Breaktime – Cluster need a minimum of 15 -20 minutes to provision everything

Verify the EKS kubernetes cluster from AWS

You can go back to your AWS dashboard and look for Elastic Kubernetes Service -> Clusters

kubectl get nodes

kubectl create namespace

kubectl get services

<u>Third Phase – Manually – Step by Step</u>

- Add Docker and GitHub Credentials into Jenkins
- Add jenkins stages
- Build, deploy and test CI CD pipeline

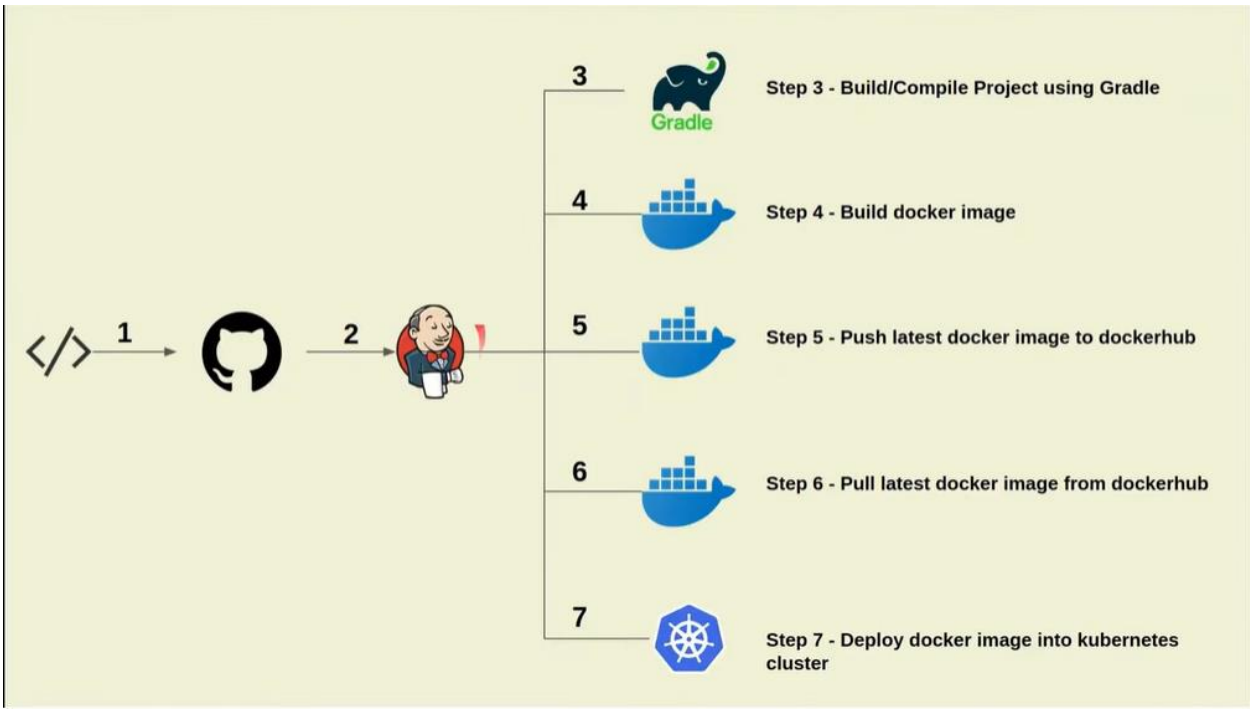
Add Docker and GitHub Credentials into Jenkins

As we know Kubernetes is a container orchestration tool and container management, we are using docker.

(*In case if you haven't set up Docker Hub Account then please create a DockerHub Account because we are gonna need it.*)

Alright so if you are reading this line then I am assuming you have a DockerHub Account and GitHub Account.

Here is the link of GitHub Repository for this project



Setup Docker Hub Secret Text in Jenkins

You can set the docker credentials by going into -

Go to -> Jenkins -> Manage Jenkins -> Manage Credentials -> Stored scoped to jenkins -> global -> Add Credentials