**Experiment 1**

| **1.0** | **Hands on with GitHub & Git** |
| --- | --- |
| 1.1 | Creating new free GitHub account online. |
| 1.2 | Creating new repository and commit the codes to newly created repository |
| 1.3 | Importing the codebase from different Github account to your Github account |
| 1.4 | Understanding different operations on Github |

* 1. Creating new free GitHub account online.

1. Go to **github.com –Signup** provide mail id and create an account.
2. Install git for windows or sudo yum install git(linux)
   1. Creating new repository and commit the codes to newly created repository
3. Sign in github Create a repository (**DemoRepo**) ,provide a valid name and make public or

private.

Eg: **PreethaS25/DemoRepo**

1. Create Program files in local Folder **Devops Lab** (a.html,login.py …)
2. Go inside Local Folder **Devops Lab ,**Right Click open **Git Bash here**
3. Automatically will create a git repository locally

**git init**

1. Configure name and email

**git config - -global user.name “PreethaS25”**

**git config - -global user.email “preethas@nttf.co.in”**

1. Add the files in the staging Area

**git add a**.**html** or **git add .(**all the files**)**

1. Save the changes in the staging Area

**git commit** –m “messages”

1. Connect remote repository into local repository

**git remote add alternativeReponame “Remotrepositorypath”**

Eg: **git remote add Reponew** [**https://github.com/PreethaS25/DemoRepo.git**](https://github.com/PreethaS25/DemoRepo.git)

1. Perform Push

**git push alternativeReponame Branchname**

**Eg: git push Reponew master**

1. **Authorize github Refresh the remote repository(github) ,the files will be added**
   1. Understanding different operations on Github
2. Perform **pull** operations

Add a file in Remote repository Sample.html

1. Goto GitBash –Perform pull

**git pull alternativeReponame Branchname**

**Eg: git pull Reponew master –**Sample.html is pulled in the system

1. List all the branches

**git branch**

1. Create a new branch

**git branch NewBranchname**

**Eg: git branch BranchAttend**

1. Switch to new branch

**git checkout Branchname**

**Eg: git checkout BranchAttend**

1. Add a file to new Branch-Add the file to staging area,commit and push the file to remote repository
2. **Create a file in local folder Example.html**
3. **git add Example.html;**
4. **git commit –m “Example is added in new branch “**
5. **git push Reponew BranchAttend**
6. Refresh the github and see the result

7.Merge the child branches to master branch

1. switch to master branch,merge new branch and push the changes in github

**a. git checkout master**

**b. git merge Branchname**

**Eg: git merge BranchAttend**

**c. git push Repo BranchAttend**

**1.4**Importing the codebase from different Github account to your Github account

1. Go to Github top corner press +sign .Click import Repository.Paste Old Url,

Give repository details,Begin import

**or**

2. **git clone RemoteRepoPath -** obtain a repo from existing URL

**Experiment 2**

| **2.0** | **Using Chef Solo on Linux for AWS** | **4** |
| --- | --- | --- |
| 2.1 | Configure a Chef node in AWS using Chef Solo |  |
| 2.2 | Write Chef cookbooks and recipes |  |
| 2.3 | Use Chef attributes and templates to generalize your cookbooks |  |
| 2.4 | Create Chef roles to define Chef node functions |  |

Chef solo?

Chef-Solo is an open source tool that runs locally and allows to provision guest machines using Chef cookbooks without the complication of any Chef client and server configuration. It helps to execute cookbooks on a self-created server.

**2.1 Configure a Chef node in AWS using Chef Solo**

**1. Launch an Ubuntu Instance in AWS account and access through putty**

2. **Install Chef dk**

**wget**  [https://packages.chef.io/files/stable/chefdk/3.2.30/ubuntu/18.04/chefdk\_3.2.30- 1\_amd64.deb](https://packages.chef.io/files/stable/chefdk/3.2.30/ubuntu/18.04/chefdk_3.2.30-%201_amd64.deb)

**3. Use your distribution’s package manager to install ChefDK**

For ubuntu

sudo dpkg -i chefdk\_3.2.30-1\_amd64.deb

1. **Check the Chef version installed-**  chef - -version
   1. **Write Chef cookbooks and recipes**
2. **Create a directory -** sudo mkdir chef\_Repo
3. **Change the directory** - cd chef\_Repo
4. **Generate a cookbook –** sudo chef generate cookbook cookbookname

**Example –** sudo chef generate cookbook Sample

1. **Change the directories -**cd Sample ,cd recipes

ubuntu@ip-172-31-19-131:~/chef-repo/sample/recipes$

1. **Create a recipes**

Create a ruby file with file extension .rb – since we are recipe on ruby.

vim hello.rb

file 'HelloChef' do

content 'Hello Ruby'

end

Save the hello.rb

1. **Apply the changes**

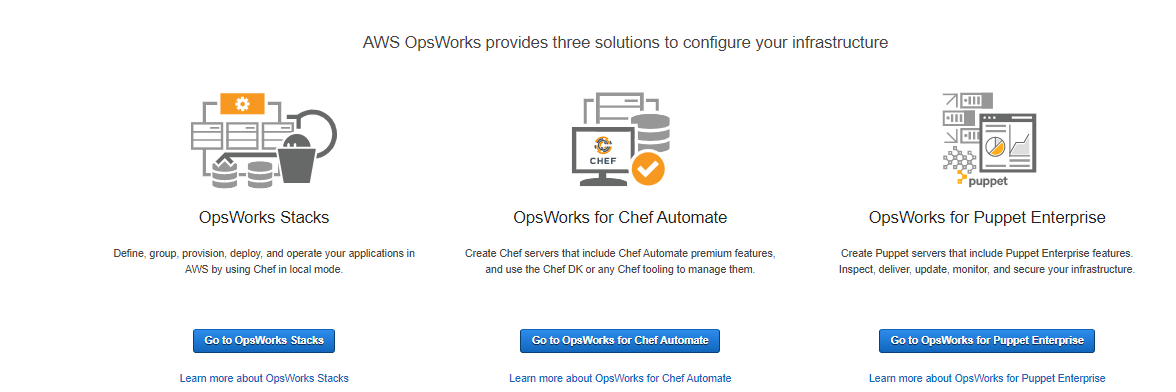
chef-apply hello.rb

Once we run this chef-apply command. We should have a 'HelloChef' file created with content 'Hello Ruby

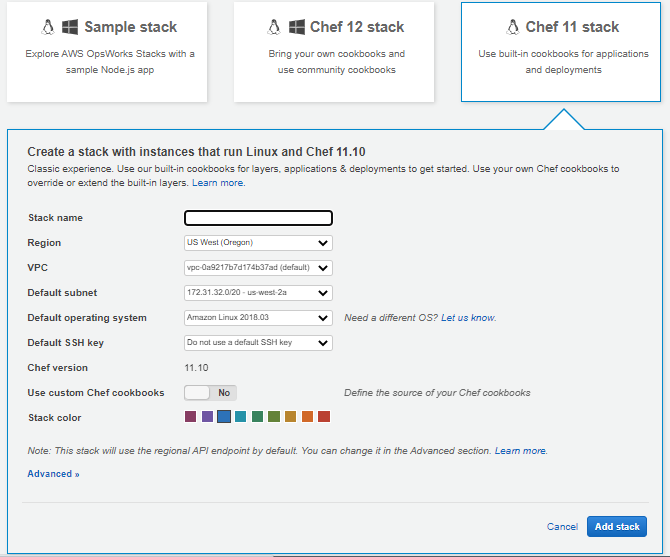
| **3.0** | **Build your first OpsWorks stack** |
| --- | --- |
| 3.1 | Build the infrastructure and automate code deployment for a PHP application with OpsWorks |
| 3.2 | Registering the Instance (Linux) in OpsWorks |
| 3.3 | Deploying Apps- Deploy test PHP application |
| 3.4 | Execute other commands – Undeploy, Rollback, Start Web Server, Stop Web Server, Restart Web Server |

Step1: Go to Services ,Select **OpsWorks**

Step2: Select Opsworks Stacks and click go to opsworks stacks



Step3: Click and Add Your FirstStack and select **Chef11 stack –**provide a Valid name **PHPAppServer**

****

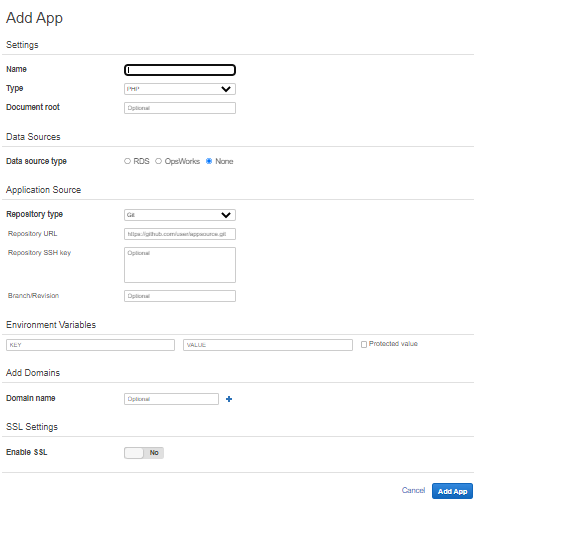
Step4**:** Add a Layer ,choose Opsworks,select Layer Typeas **PHP App Server** and click add layer

Step5: Add a Instance to the layer-choose New instances provide a Hostname php-app1,size t2.micro

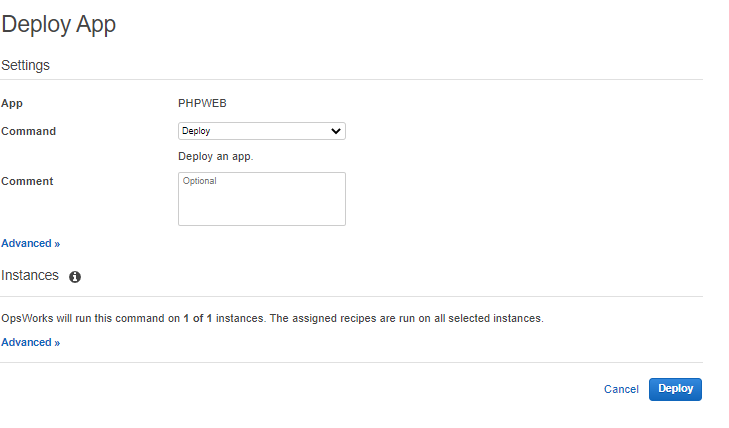
then click Add instance.

Step6 : Click and Start the instance and make it **online**

Step7: Add an application –Click Add app,provide a valid name **PHPWEB**,Resource Type select as Git,provide Git URL[**https://github.com/aws-samples/opsworks-demo-php-simple-app**](https://github.com/aws-samples/opsworks-demo-php-simple-app)**” and Add App**

****

Step8: Perform the action **Deploy ,**In deployment page provide the command as **Deploy** and then click deploy



Step 9: Once deployment completed .Check the ouput. Go to instance,select the ipaddress ,the website is deployed successfully.

**Adding a new Layer**

**Step 1: Create a Loadbalancer –Go to Ec2 ,choose Classic Load balancer,create load balancer,**

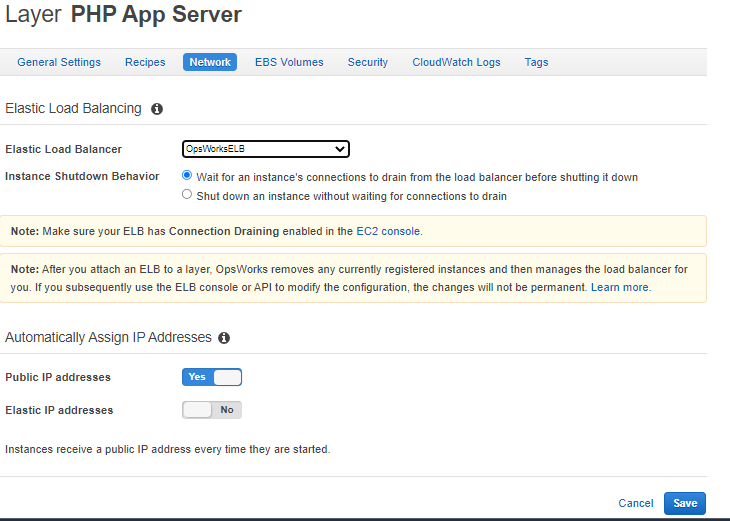
**Provide a Loadbalancer name OpsworksELB.**

Step2 : Next Assign Security group,select existing security group **AWS-OpsWorks-PHP-App-Server**,Click Next and Next and configure Health Check provide ping path **index.php,**Click Next,Next

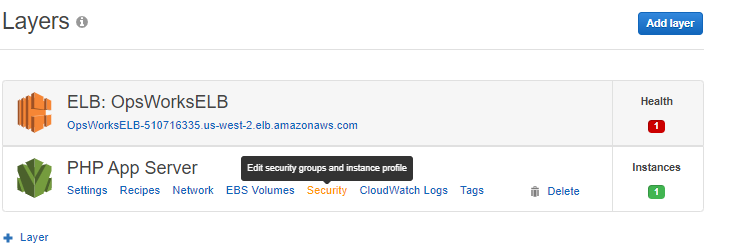
,Review and Create,Create

Step3: Goto Opsworks,Select the stack and layer which u created

Step4: Select Layer PHP App Server,Select Network ,provide Elastic Load Balancer name as **OpsworksELB,**then Click Save



**Step 5:** A Layer is successfully added.Go to Layer and check 2Layers available.namely

****

Step 6: So Loadbalancer will take the load and see output ,Click the Url of Loadbalancer(  
OpsWorksELB-510716335.us-west-2.elb.amazonaws.com) .It will navigate to the application website.

Step7: Add one more instance from the layer ,start the instance,make online and check the application by loadbalancer.Loadbalancer will take care both the application server

Step8: Refresh the web pages request will be taken from php-app1 and php-app2

| **4.0** | **Getting Started with Jenkins on Windows (Hyper or Oracle VM box)** |
| --- | --- |
| 4.1 | Build the server on Hyper or Oracle VM box on on-premise machine |
| 4.2 | Install and configure a Jenkins server |
| 4.3 | Add plugins and create Jenkins jobs |
| 4.4 | Run Jenkins jobs and examine outputs and results |

* 1. Build the server on Hyper or Oracle VM box on on-premise machine

**Step1** : Create an Ubuntu instance in AWS and access through putty

**Step2:** Debian package repository of Jenkins to automate installation and upgrade. To use this repository, first add the key to your system:

**curl -fsSL https://pkg.jenkins.io/debian/jenkins.io.key | sudo tee \**

**/usr/share/keyrings/jenkins-keyring.asc > /dev/null**

When the key is added, the system will return **OK**.

**Step3** : Then add a Jenkins apt repository entry:

**echo deb https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list**

**Step 4**: We should ensure JDK installed, before we install Jenkins on the server.

**Install Java:**

Since Jenkins is a Java application, the first step is to install Java. Update the package index and install the Java 8 OpenJDK package with the following commands

**sudo apt update**

**sudo apt-get install fontconfig openjdk-11-jre**

* 1. Install and configure a Jenkins server

Step 1: Once the Jenkins repository is enabled, update the apt package list and install the latest version of Jenkins by typing:

**sudo apt update**

**sudo apt install Jenkins**

Jenkins service will automatically start after the installation process is complete. You can verify it by printing the service status:

sudo **systemctl status Jenkins**

Step2:

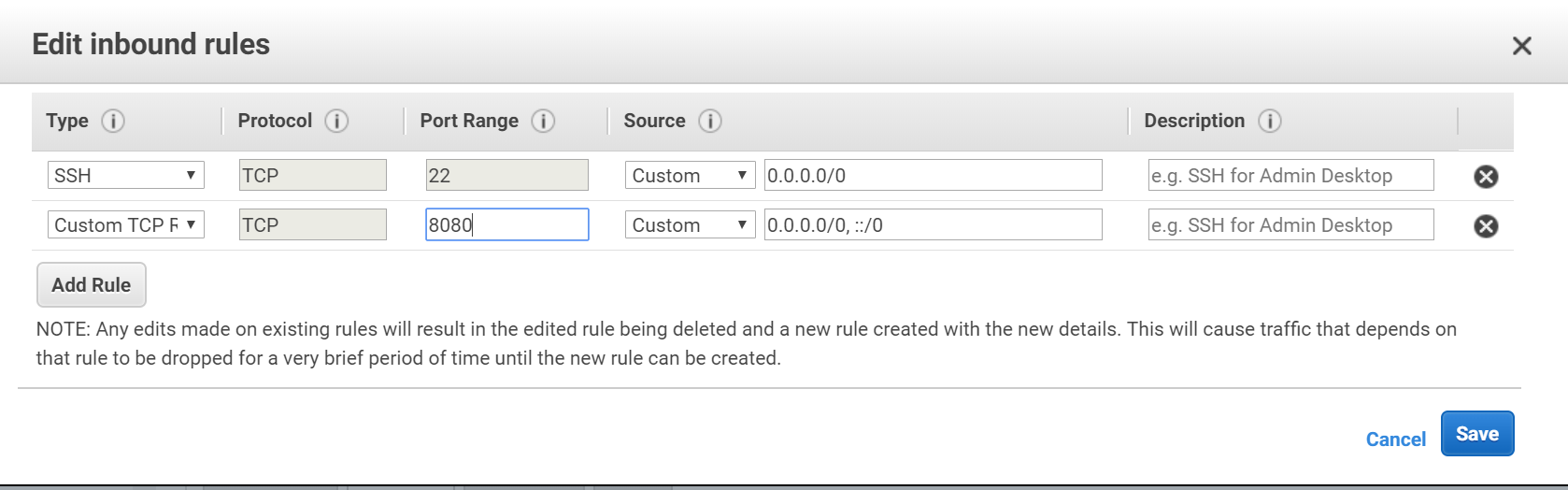
By default, Jenkins listen on port 8080. Access this port with your browser to start configuration.

Allowing 8080 port on Firewall By default, 8080 port is not allowed on firewall.

Click on “Description” on AWS instance (where you have created the instance).

Under Security Groups🡪”launch-wizard-5”

Then click on “Inbound tab” 🡪Edit



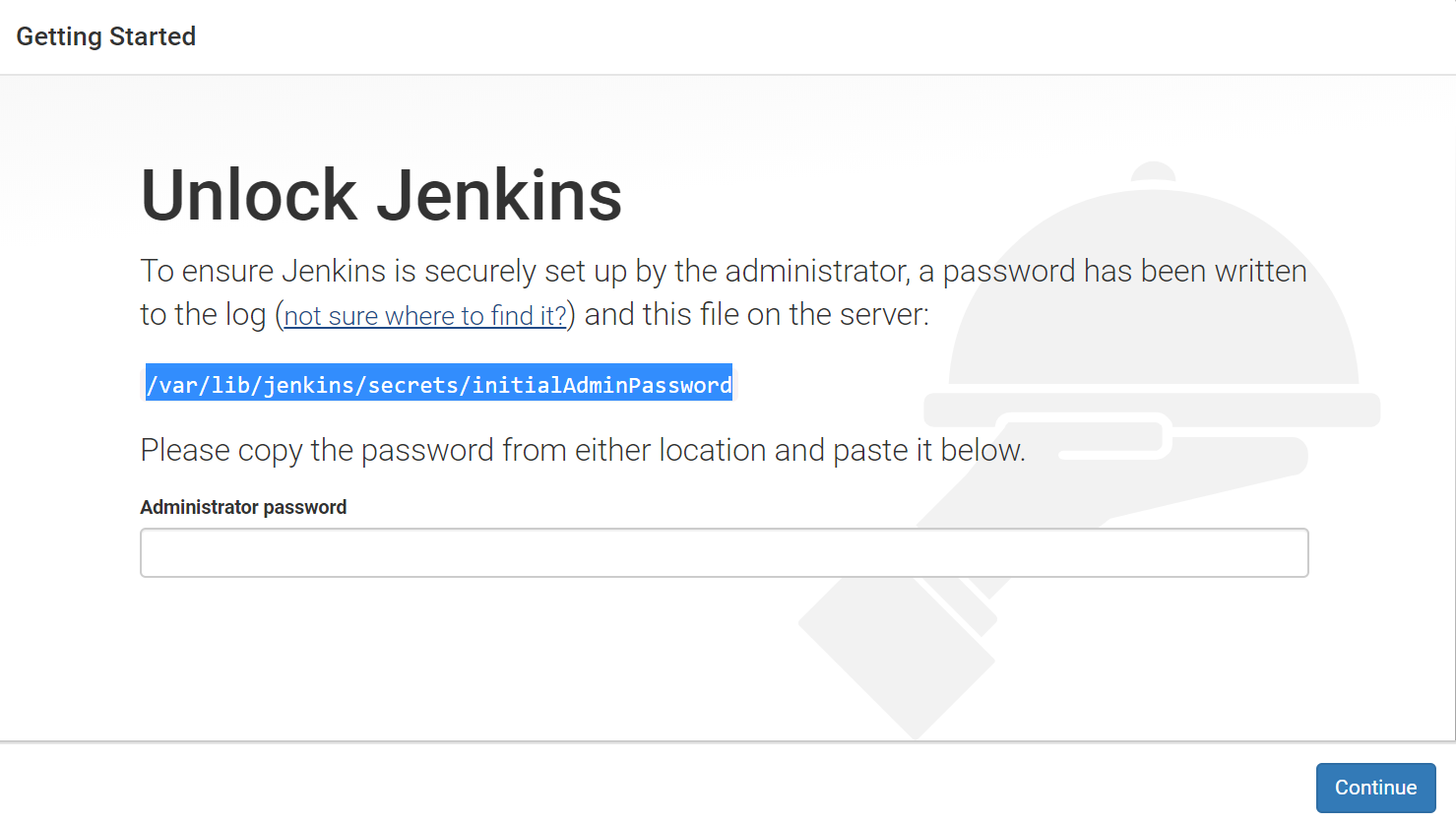
Add a new rule with the “Custom TCP rule” and the range “8080”🡪save

Step3 :

Try to access the public ip with :8080 port on the browser.

**Example**: 35.153.127.253:8000

Step4: To Unlock the jenkins



Make sure you access the “initialAdminPassword “ file with root permission.

root@ip-172-31-35-128:~# sudo su

root@ip-172-31-35-128:~# cd /var/lib/jenkins/secrets/

root@ip-172-31-35-128:/var/lib/jenkins/secrets# nano initialAdminPassword

* 1. Add plugins and create Jenkins jobs

Step1: Install suggested plugins

Step2: Create a Admin user

Step3: Login with Admission permission

Step4: Select Create a Job/New Item

Step5: Enter an Item name Sample, Select freestyle project, Click ok

Step6: Got to General, Give Description of the project

Step7: Trigger the build –select periodically, Example \* \* \* \* \*

Step8: Build Step-Execute Shell –Provide a command ‘date’

Step9: Click on Save

* 1. Run Jenkins jobs and examine outputs and results

Step1: Select the project and Build Now

Step 2:Got Dashboard, Select the project and console Output.

| **5.0** | **Software provisioning and configuration management with Ansible** |
| --- | --- |
| 5.1 | Introduction to Ansible hands-on |
| 5.2 | YAML and Hands-on Exercises |
| 5.3 | Build Ansible Inventory Files and modules |
| 5.4 | Automate provisioning and web server deployment |

* 1. Introduction to Ansible hands-on

Prerequisites

Two or more Ubuntu 18.04 servers. One of these will be used as our Ansible server

While another server will be used as your Ansible hosts.

**Step1** : Set password for root and switch in both the servers.

❖ sudo passwd root

❖ su root

**Step2** : Install Ansible in server

To get the latest version of Ansible for Ubuntu, you can add the project's PPA (personal

package archive) to your system, you should first ensure that you have the software-

properties-common package installed.

❖ sudo apt update

❖ sudo apt install software-properties-common

Then add the Ansible PPA by typing the following command:

❖ sudo apt-add-repository ppa:ansible/ansible

Next, refresh your system's package index once again so that it is aware of the packages available in the PPA:

❖ sudo apt update

Begin with ansible installation

❖ sudo apt install ansible

**Step 3**: Configuring SSH Access to the Ansible Hosts and generate a SSH Key in Server as well as Hosts

❖ ssh-keygen – the keys will be available in cat /root/.ssh/id\_rsa.pub

Above public key ,we have copy into authorized\_keys file

**Step 4**: Copy the rsa key of server and paste in hosts and vice versa.

- cat /root/.ssh/id\_rsa.pub copy from server

- cd /root/.ssh –Switch to the location in hosts and paste

nano authorized\_keys

**Step 5:** Check connection of server and hosts

❖ ssh 172.31.36.39 (IP address of hosts checking in the server)

❖ ssh 172.31.43.84 (IP address of server checking in the hosts)

* 1. Build Ansible Inventory Files and modules

Step 1: Configure and Edit inventory file

nano /etc/ansible/hosts

[test-servers]

172.30.36.39

172.30.36.40

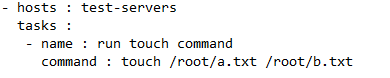
Step 2: Check the status of hosts

ansible –m ping test-servers

5.3 YAML and Hands-on Exercises

Step1 : create a playbook

nano conf.yml



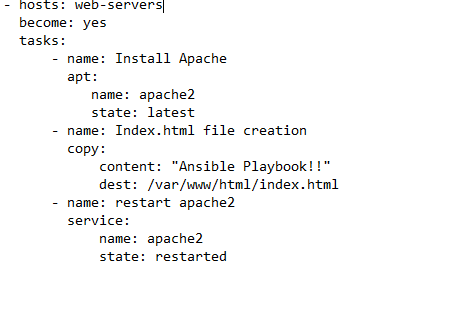
Step 2: Execute the playbook

ansible-playbook conf.yml

step3 : Check the files in the host in the location /root/ two files were created a.txt b.txt

* 1. Automate provisioning and web server deployment

Step1: Create a playbook for apache webserver installation –apache.yml



Step2: Execute Playbook

ansible-playbook apache.yml

1. Check in host apache installed and check version apache2 –v
2. Check Index.html in browser.

**Exp 6**

| **6.0** | **Linux Command Line Byte Session** |
| --- | --- |
| 6.1 | Understand what the Linux command line |
| 6.2 | Directory Manipulations commands |
| 6.3 | Use common commands for working with files and directories |
| 6.4 | Installing Software in Linux CLI |
| 6.5 | Sending E-mails |

**6.1) understand what the linux command line**

**Working commands:-**

**Pwd:- print name of present working directory**

**Who:- used to get information about currently logged in user on the system**

**Ls:- list of information about the files**

**Wc:- print the number of new lines,words,bytes of file**

**6.2) Directory manipulation commands**

**mkdir:- create a directory**

**cd :-switching directory**

**6.3) use common commands for working with file and directories**

**CP**

**Description:- copy file and directories**

**Command:- cp sourcefile destinationfile**

**Example:- cat > file1.txt**

**a**

**b**

**c**

**d**

**ctrl+d**

**cat >file2.txt**

**cp file1.txt file2.txt**

**command :- cp filename directory**

**example:- cp file1.txt d1**

**for viewing : cd d1**

**ls**

**3) MV**

**Description:- move files**

**Cmd:- mv filename directory**

**Example:- mv file2.txt d1**

**For viewing:-**

**Cd d1**

**Ls**

**MOVING DIRECTORY TO DIRECTORY**

**Command:- mv sourcrfile destinationfile**

**Example:- mv d1 d2**

**For viewing:- cd d2**

**Ls**

**4) CHANGING DIRECTORY**

**Description:- change directory**

**Cmd: cd directoryname**

**Example:- mv d1 d2**

**For viewing:- cd d2**

**Ls**

**Rm**

1. **Remove directory**

**Rm directory**

**Non-empty directory**

**Rm -r directoryname**

1. **Ls**

**Description:- list information about the files**

**Touch**

**Ex:- touch a.txt b.txt**

**CAT**

**Ex:- cat>file1.txt**

**Content in file:-**

**A**

**B**

**C**

**D**

**Ctrl+d**

**Ex:- vi file1.txt(for opening a file)**

**6.4 INSTALLING APPLICATION ON LINUX**

**sudo apt install appname**

**Exampe: sudo apt install apache**

**For uninstall the application**

**sudo apt remove appname**

**Example :sudo apt install apache**

**To upgrade installed application**

**sudo apt update**

**sudo apt upgrade**

**Update single app**

**sudo apt upgrade appname**

**Example: sudo apt upgrade apache**

**6.5 SENDING EMAILS**

**1)SET THE PASSWORD FOR ROOT**

**Sudo passwd root**

**Pass:xxxxxxxxxx (retype the password )**

**Then switch root:- su root**

**2. INSTALL SEND MAIL PACKAGE**

**Sudo apt-install sendmail**

**3. install smtp configuration file**

**Sudo apt-install ssmtp**

**4. configure ssmtp,open the file**

**Nano /etc/ssmtp/ssmtp.conf**

**Use STARTTLS = yes**

[**Root=example@gmail.com**](mailto:Root=example@gmail.com)

**Mail hub= ssmtp.gmail.com:587**

**Authuser=** [**example@gmail.com**](mailto:example@gmail.com)

**Authpass:- xxxxxxxx**

**Ctrl+x (for save the file)**

1. **SEND A EMAIL**

**Echo “subject:Hello”/sendmail** [**example@gmail.com**](mailto:example@gmail.com)

| **7.0** | **Create AWS Resources with Terraform** |
| --- | --- |
| 7.1 | Install Terraform on Linux |
| 7.2 | Configure Terraform providers |
| 7.3 | Create AWS resources with Terraform |

**7.1 Install Terraform on Linux**

Step1: Create an EC2 Instance, download the latest version of Terraform source using the following command:

wget https://releases.hashicorp.com/terraform/1.1.2/terraform\_1.1.2\_linux\_amd64.zip

Step2:

Once the download is completed, extract the downloaded file using the following command.

unzip terraform\_1.1.2\_linux\_amd64.zip

Step3:

Copy the Terraform binary from the extracted file to the /usr/bin/ directory:

sudo mv terraform /usr/local/bin/

Step4:

Verify the Terraform version by running the following command:

terraform -v

Copy

You will get the following output:

Terraform v1.1.2

on linux\_amd64

**7.2 Configure Terraform providers**

provider "aws" {

region = "us-west-2"

access\_key = "my-access-key"

secret\_key = "my-secret-key"

}

**Generate an access-key and secret-key.**

Step1: Go to Services –Select IAM, Select User

Step2:Add a User,Set User Details,Name it as **Pol,Select** AWSCredential Type as

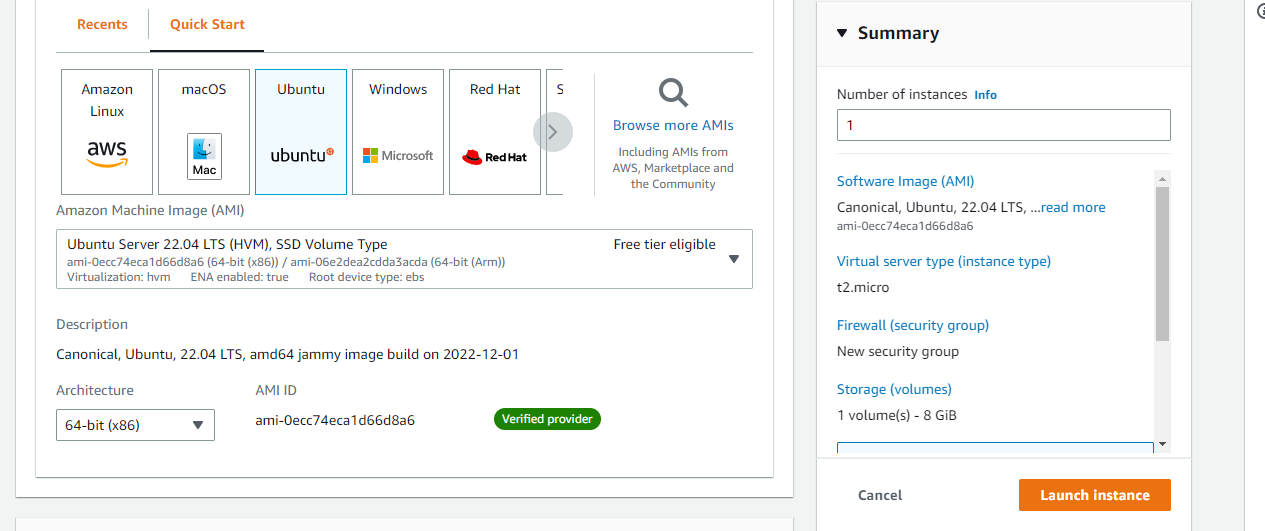
**Access key - Programmatic access,** Click Next

Step3: Set the permissions as Attach Existing policies, Select Administrator Access,Click Next,Add Tags,Review and Create User.

Step4: User created Successfully. Download csv file ,Accesskey and Secret Key available ,Use this in the configuration file **infra.tf**

**Get Amazon Machine Image**

**Create your Own AMI or select from Lauch Instance-ANY Ubuntu AMIID**

****

**7.3 Create AWS resources with Terraform**

Step1:

**Create a Aws resource AWS Instance (Ubuntu)**

Make a folder- **mkdir TerraformDemo**,cd TerraformDemo and save the file

**vi infra.tf**

provider "aws" {

region = "us-west-2"

access\_key = "AKIAUL3OWSZDHBJ6MQEJ"

secret\_key = "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"

}

resource "aws\_instance" "webserver" {

ami = "ami-0b22b0162cd733db8"

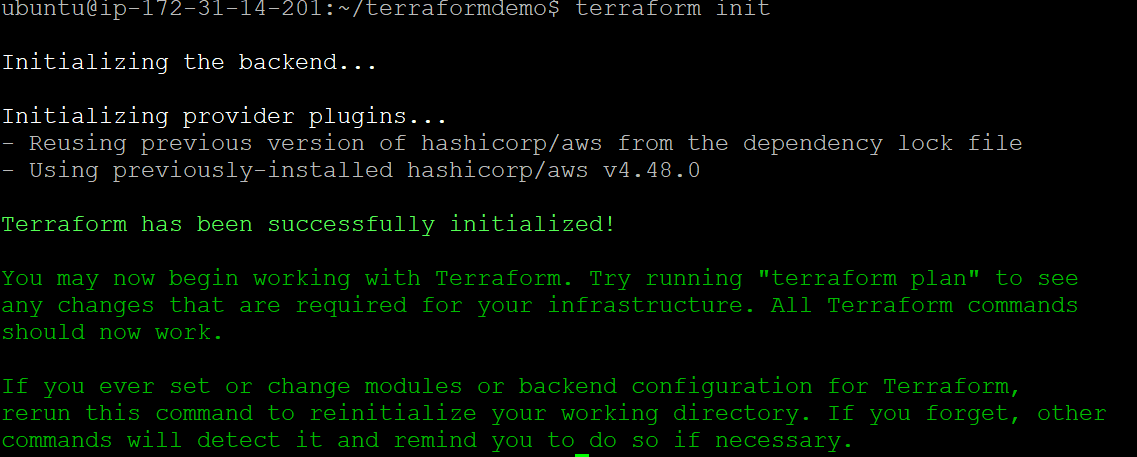
instance\_type = "t2.micro"

}

Step2: Execute the Configuration file - HashiCorp Configuration Language (HCL)

Main commands:

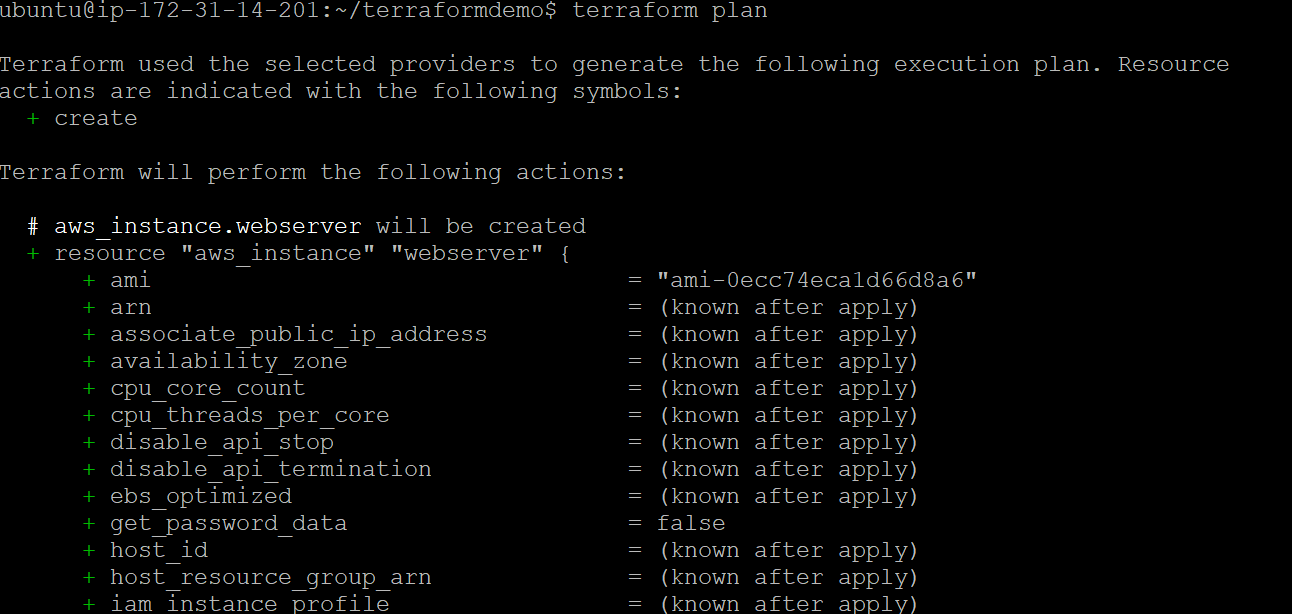
1. **terraform init** Prepare your working directory for other commands



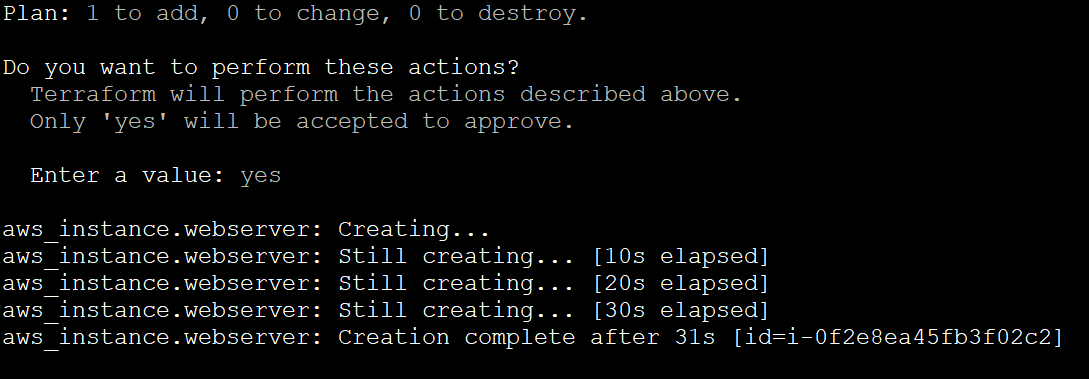
1. **terraform validate** Check whether the configuration is valid



1. **terraform plan** Show changes required by the current configuration

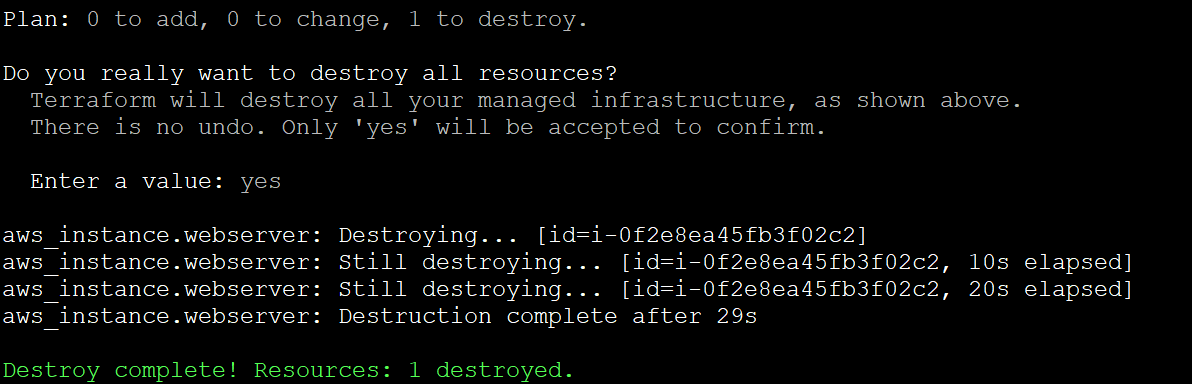


1. terraform apply Create or update infrastructure



After apply the changes ,Check in EC2 an Instance will be created

1. terraform destroy Destroy previously-created infrastructure



Check the Instance is destroyed in AWS.

**EXPERIMENT 8**

| **8.0** | **Deploy Word press using Cloud Formation** |
| --- | --- |
| 8.1 | Create your first cloud infrastructure using Amazon Cloud Formation |
| 8.2 | Use CloudFormation to launch an EC2 (Linux) instance |
| 8.3 | Create the Stack and choose template for launching instance |
| 8.4 | Perform Wordpress full installation in the instance |

**Create your first cloud infrastructure using Amazon Cloud Formation**

Step1: Go to services –Select CloudFormation

Step2: Create Stack, Choose the opion Create Template in Designer

Step3: Click Create Template in Designer, From the Resource Types, Drag and drop a resource

Named as S3 –Bucket

Step4: Go to Top corner,Click  and Create stack.

Step5: Now Template is ready.Click Next button,Specify the stack Details ,specify a Stack name **S3Stack**

Step6:Click Next,Configure stack options,Click Next,Review and Submit (status-Creation in progress).

Step7:Check the status –CREATE\_COMPLETE ,Check in S3 service-A bucket is created.

**Perform Wordpress full installation in the instance**

Step1: Select Cloud Formation from services-Create Stack, Select with new resources

Step2: Choose the option Use a Sample Template

Step3: Select a Sample templates Wordpress Blog ,Click Next

Step4:Specify stack Details-Provide a Stack name WordPress Stack,Specify Parameters

Step5: DBName,DBPassword,DBRootPassword,DBuser,InstanceType-t2.micro,

Keyname(Generate a key and provide),Click Next button

Step6: Configure the stack options,Click Next,Review and Click Submit Button.

Step7:Verify the word press –Goto StackInfo ,Outputs –Click the Wordpress WEB Url

http://ec2-54-92-203-82.compute-1.amazonaws.com/wordpress/wp-admin/index.php

Step8: Provide Site Login informations such as Usename ,Password,mail id …

Step9: Once successfully created ,You can able to access WordPress Content Management Dashboard

EXPERIMENT=9

| **9.0** | **Develop and Deploy an Application** |
| --- | --- |
| 9.1 | Create AWS CodeStar projects |
| 9.2 | Monitor project activity |
| 9.3 | Develop and deploy code using AWS CodeStar |
| 9.4 | Manage teams inside of AWS CodeStar projects |

STEP 1: GO TO SERVICES AND *SELECT CODESTAR*

Graphical user interface, application

Description automatically generated

2: IN TEMPLATE

AWS SERVICE SELECT: *AWS EC2*

APPLICATION TYPE*: WEB APPLICATION*

PROGRAMMING LANGUAGE*: NODE.js*

DOWN TO TWO PROGRAMMING LANGUAGE WILL BE AVAILABLE SELECT THE *NODE.JS.*

Graphical user interface

Description automatically generated

3: CLICK *NEXT*

STEP 2: **IN SET UP YOUR PROJECT**

1. GIVE PROJECT NAME.

2. PROJECT ID WILL GENERATE AUTOMATICALLY.

3. IN PROJECT REPOSITORY SELECT *CODE COMMIT*

4. EC2 CONFIGURATION

INSTANCE TYPE: *t2.micro*

VPC: SELECT THE DEFAULT ONE.

SUBNET: SELECT THE DEFAULT ONE.

KEY PAIR: SELECT THE DEFAULT ONE.

5.SELECT I acknowledge that I have access to the private key file.

Graphical user interface, text, application

Description automatically generated

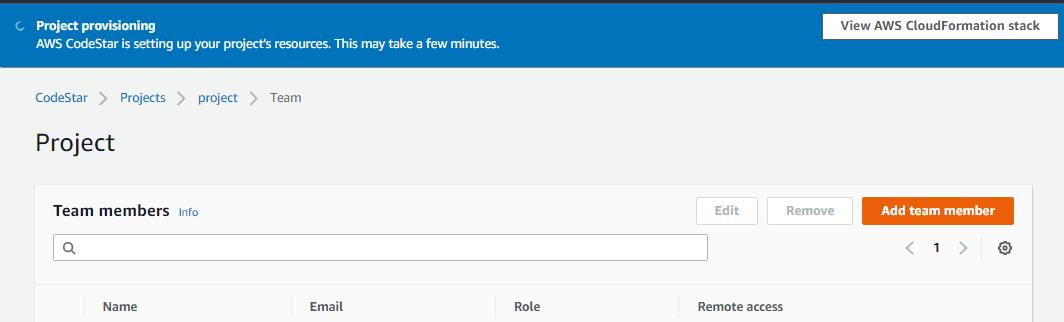
6: CLICK *NEXT.*

STEP 3: **REVIEW**

CLICK CREATE PROJECT.Graphical user interface, application, website

Description automatically generated

IT WILL TAKE TIME TO PROVISINONING THE PROJECT WAIT UNTILL IT COMPLETE.



**ADD TEAM MEMBERS**

STEP 1: CLICK ON TEAM

STEP 2: CLICK ON *ADD TEAM MEMBER*.

Graphical user interface, application, Teams

Description automatically generated

STEP 3: GIVE THE USERNAME.

STEP 4: GIVE DISPLAY NAME.

STEP 5: GIVE EMAIL ADDRESS.

STEP 6: SPECIFY THE PROJECT ROLE.

OWNER

CONTRIBUTOR

VIWER.

SELECT CONTRIBUTOR.

Graphical user interface, application

Description automatically generated

STEP 7: CLICK ADD TEAM MEMBER.

THE MEMBER WILL BE ADDED TO PROJECT.

**DEPLOY**

STEP 1: CLICK ON PROJECT AND SELECT THE PROJECT WHICH HAS BEEN CREATED.

SELECT PIPELINE.

CLICK ON *VIEW APPLICATION*

.

Graphical user interface, text, application

Description automatically generated

YOU CAN SEE THE DEFAULT NODE.js WEB APPLICATION.Graphical user interface, application

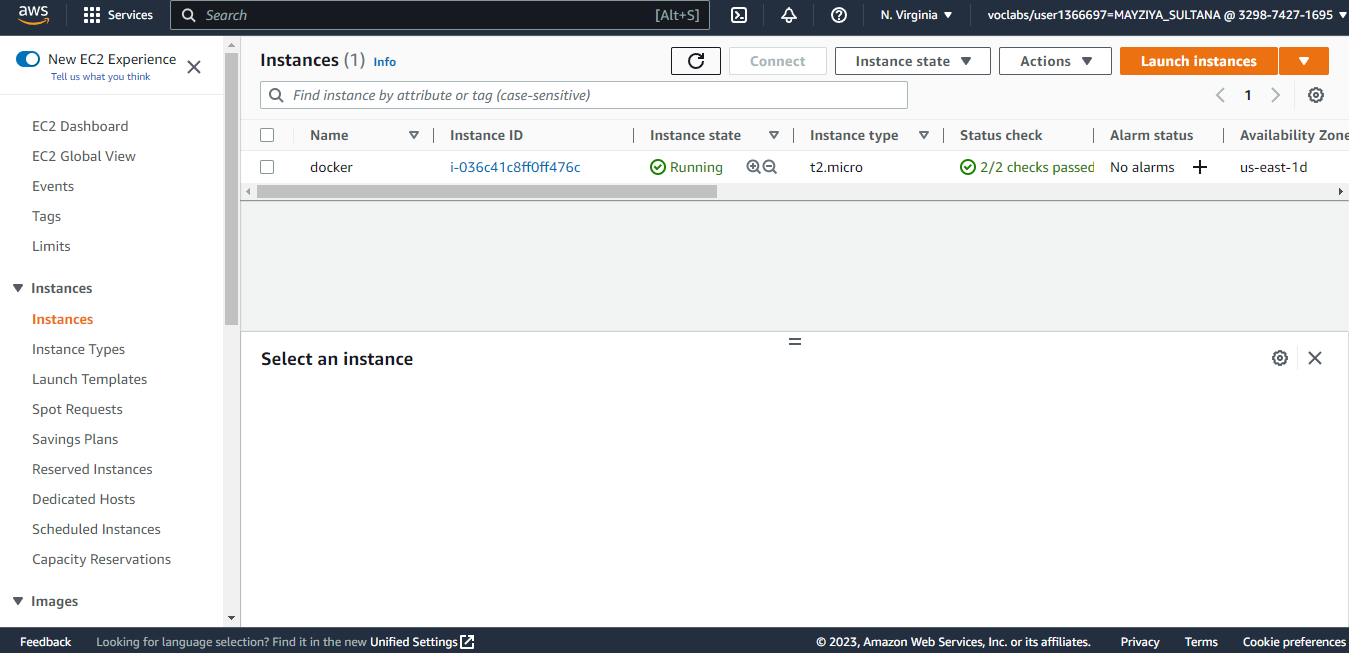
Description automatically generated

Experiment – 10

| **10.0** | **Use AWS Fargate for Serverless Deployment of Container Applications** |
| --- | --- |
| 10.1 | Use application source files to create Docker container images and place them in Amazon ECR |
| 10.2 | Create an Amazon ECS cluster |
| 10.3 | Use the Amazon ECR images to create ECS task definitions and services |
| 10.4 | Implement the services in ECS by using AWS Fargate enjoying the advantages of serverless |
| 10.5 | Register ECS tasks with an Application Load Balancer (ALB) target group |

**Docker Installation**

Step 1 :- Create an ubuntu Instance Switch to root account



2 :- Copy IP address and paste in putty or Connect through Aws console

Terminal.

3 :- Install docker by using shell script

curl -fsSL https://get.docker.com -o get-docker.sh

DRY\_RUN=1 sudo sh ./get-docker.sh

* You will get this message, but don’t worry :- To run Docker as a

non-privileged user.

4 :- Check docker is installed or not

Syntax :- docker -–version

* Latest version will appear :- Docker version 20.10.22,

1. :- check the status of docker image in the system

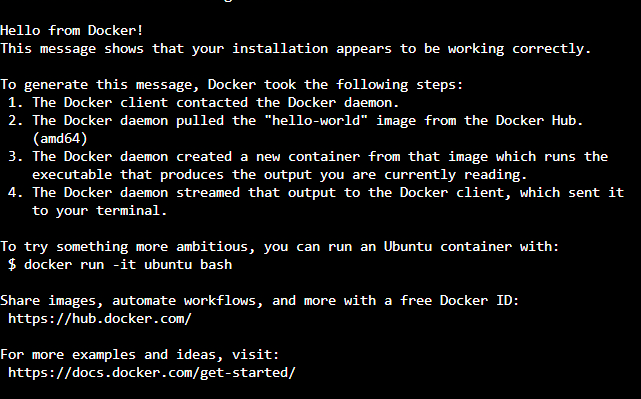
docker images (No Images have created)

* For ex:- REPOSITORY TAG IMAGE ID CREATED SIZE

1. :-Pull and run docker image (hello-world) from **DockerHub**

sudo docker run hello-world

* After executing the command you will get this message



7 :- sudo docker images (Check Image is pulled or not)



* Image (hello-world is created)

8::- Checking container running status on the system

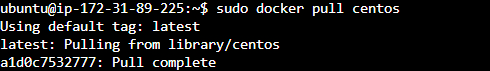
sudo docker ps



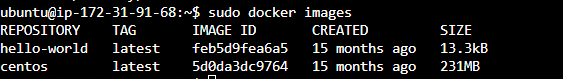
Here no container is running hello-world will exited immediately

**Step 2 :- Create Another Image For ex:- centos**

1 :- sudo docker pull centos (pull the image)



2 :- sudo docker images (check Image Created or Not)

* For ex :- 
* See Now 2 Images have been created

(1 🡪 hello-world , 2 🡪 centos)

1. :- Run and change the terminal

sudo docker run -it centos (it will change from ubuntu to centos(root account))

* For ex :-



Step 3 :- **Open another terminal with same IP address**

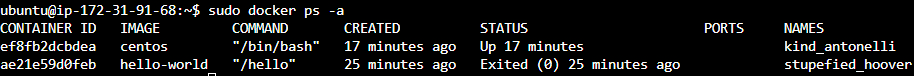
1 :- sudo docker ps(checking process status)

* For ex :-



2 :- sudo docker ps -a(checking process status all)

* For ex :-



**10.1 Use application source files to create Docker container images and place them in Amazon ECR**

**Step1:** Install aws-cli for linux

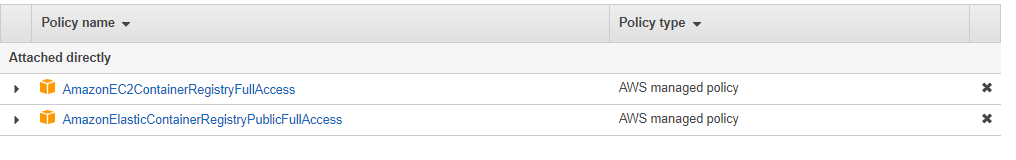
**curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"**

**unzip awscliv2.zip**

**sudo ./aws/install**

**Step2:** Please allow permission to aws-cli to communicate to AWS

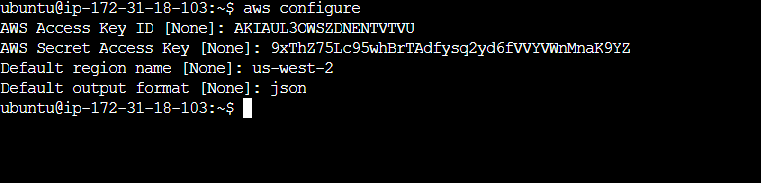
1. Goto IAM create a user with permissions-
2. Create user->Give user details ->Preetha->select credential type as Programmatic access->click next->set permissions-Attach existing policies directly –



Click next->Review and create a user

1. Download .csv file
2. Go to Instance and configure aws configure File and type

aws configure and provide Access key ,Secret key,Region and output format.

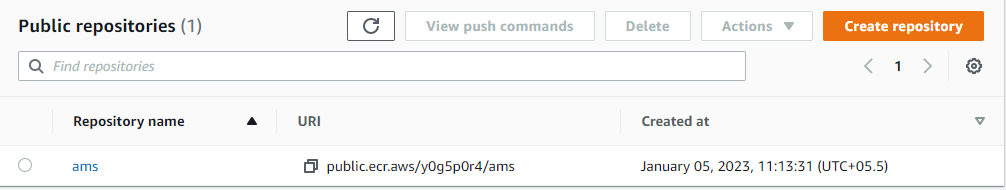


**Step3: Create**  Elastic container Registry(ECR) from AWS

Goto Services,Select ECR

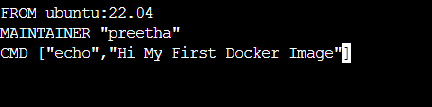
Create a repository –(Public or private)-Enter a repository name –(eg:**ams**)-Click create respository

A repository created.



**Step4**: Create a Docker image(custom) and filename should be dockerfile

**nano dockerfile**

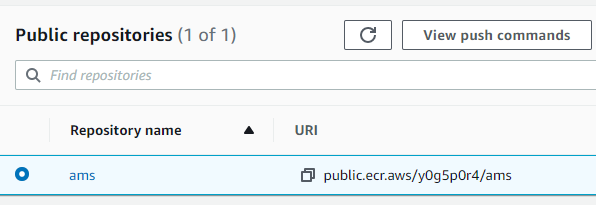
****

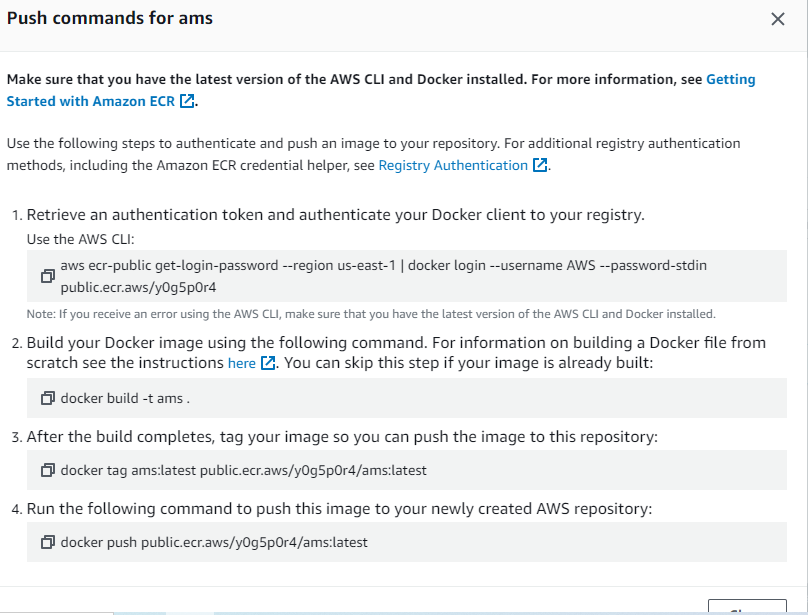
and save

**Step5:** Upload docker images into ECR

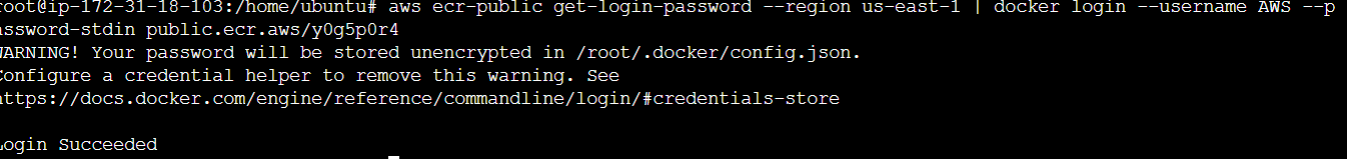
1. Goto ECR –select the repository created ams and Click view push commands

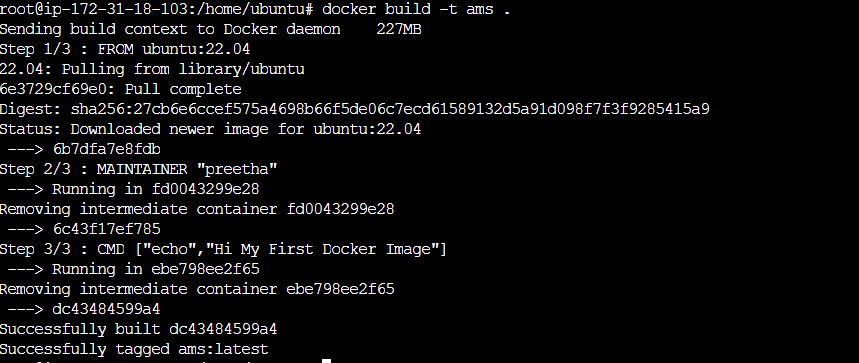
and follow the instructions and do one by in awscli in Ubuntu instance.

****

****

**Outpu t of each**

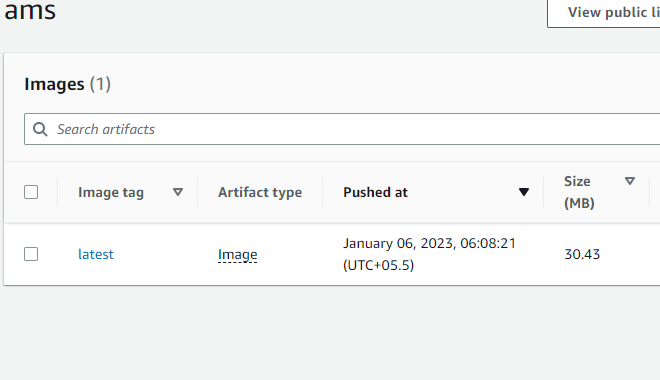
**1. **

**2.** ****

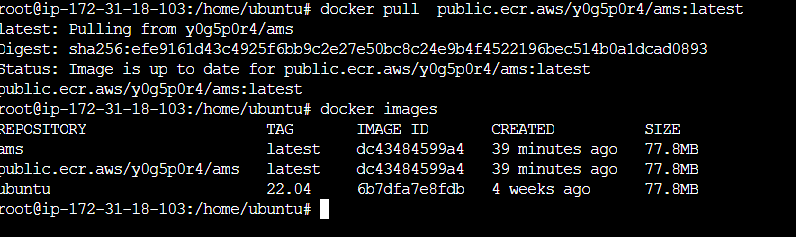
**3.4.**

****

**Docker Images is pushed to ECR.Goto ECR(ams) to check output**

****

**Step6**: Pull the images in the production system and create container

****

Then execute run command

