EXP1:- To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

Steps:

1. Login with your AWS account.

2. Navigate to Cloud 9 service from Developer tools section as below:

3. Click on Create Environment :

4. Provide name for the Enviornment (WebAppIDE) and click on next.

5. Keep all the Default seetings as shown in below:

6. Review the Enviornment name and Settings and click on Create Enviornment:
7. Till that time open IAM Identity and Access Management in order to Add user In other tab.

8. Add user provide manual password if you want and click on Next permission tab.

9. Click on Create group

10. Provide group name and click on create group.

11. After that group is created click on next if u want to provide tag else click on Review for user settings and click on create user as shown in fig.

12. Now close that window and Navigate to user Groups from left pane in IAM.

13. click on your group name which you have created and nevigate to permission tab

14. Now click on Add permission and select Attach Policy after that search for Cloud9 related policy and select Awscloud9EnviornmentMember policy and add it.
15. now we move towards our cloud9 IDE Enviornment tab it shows as shown :

16. If you check at bottom side Cloud9 IDE also giving you and aws CLI for command operations: as we here checked git version, iam user details and so on...

17. Now we will setup collaborative enviornment Click on File you can create new file or choose from template, here m opting html file to collaborate.

18. Edit html file and save it

19. now in order to share this file to collaborate with other members of your team click on Share option on Roght Pane and username which you created in IAM before into Invite members and enable persmission as RW (Read and Write) and click on Done. Click OK for Security warning.

20. Now Open your Browsers Incognito Window and login with IAM user which you configured before.

21. After Successful login with IAM user open Cloud9 service from dashboard services and click on shared with you enviornment to collaborate.

22. Click on Open IDE you will same interface as your other member have to collaborate in real time, also you all within team can do group chats as shown below:

23. you can also explore settings where you can update permissions of your temmates as from RW to R only or you can remove user too.

EXP2 :- To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Step1: Create a deployment environment

Step2: Get a copy of the sample code

In this step, you will retrieve a copy of the sample app's code and choose a source to host the code.

The pipeline takes code from the source and then performs actions on it.

You can use one of three options as your source: a GitHub repository, an Amazon S3 bucket, or an AWS CodeCommit repository. Select your preference and follow the steps below:

a. If you plan to use Amazon S3 as your source, you will retrieve the sample code from the AWS GitHub repository, save it to your computer, and upload it to an Amazon S3 bucket.

- Visit our GitHub repository containing the sample code at https://github.com/imoisharma/aws-codepipeline-s3-codedeploy-linux-2.0
- Click the dist folder.

b. Save the source files to your computer:

- Click the file named aws-codepipeline-s3-aws-codedeploy_linux.zip
- Click View Raw.
- Save the sample file to your local computer.

c. open the Amazon S3 console and create your Amazon S3 bucket:

- Click Create Bucket
- Bucket Name: type a unique name for your bucket, such as awscodepipeline-demobucket- variables. All bucket names in Amazon S3 must be unique, so use one of your own, not one with the name shown in the example.
- Region: In the drop-down, select the region where you will create your pipeline, such as ap- South-1
- Click Create.

d.The console displays the newly created bucket, which is empty.

- Click Properties.

- Expand Versioning and select Enable Versioning. When versioning is enabled, Amazon S3 saves every version of every object in the bucket.

e. You will now upload the sample code to the Amazon S3 bucket:

- Click Upload.
- Follow the on-screen directions to upload the .zip file containing the sample code you downloaded from GitHub.

- **you can upload directly zip file here from https://github.com/imoisharma/aws-codepipeline- s3-codedeploy-linux-2.0**
- ## Step3: Create your Pipeline
-
- In this step, you will create and configure a simple pipeline with two actions: source and deploy. You will provide CodePipeline with the locations of your source repository and deployment environment.
- A true continuous deployment pipeline requires a build stage, where code is compiled and unit tested. CodePipeline lets you plug your preferred build provider into your pipeline. However, in this we will skip the build stage.

-
- Goto Pipeline again and create it

In Step 4: Deploy Stage:

  - Deployment provider: Click AWS Elastic Beanstalk.
  - Application name: MYEBS.
  - Environment name: Click Myebs-env.
  - Click Next step.
- After your pipeline is created, the pipeline status page appears and the pipeline automatically starts to run. You can view progress as well as success and failure messages as the pipeline perform each action.
-
- To verify your pipeline ran successfully, monitor the progress of the pipeline as it moves through each stage. The status of each stage will change from No executions yet to In Progress, and then to either Succeeded or Failed. The pipeline should complete the first run within a few minutes.
- Now go to your EBS environment and click on the URL to view the sample website you deployed.

Step 5: Commit a change and then update your app

Step 6: Clean up your resources

To avoid future charges, you will delete all the resources you launched throughout this tutorial, which includes the pipeline, the Elastic Beanstalk application, and the source you set up to host the code.

1. First, you will delete your pipeline:

b. Second, delete your Elastic Beanstalk application:

EXP3 :- **To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.**

Step:1
In Order to configure Kubernetes Cluster we require 2 systems we can consider it as one master and one slave.
I have launch two Virtual machines master as kmaster and slave as knode as shown in Fig.
Configure static IP as:

Master: 192.168.0.101 (kmaster)
Slave: 192.168.0.102 (knode)

1. ## Step 2: Install OpenSSH-Server (In Master and Slave)

Now we have to install openshh-server. Run the following command:

```
#sudo apt-get install openssh-server
```

1. ## Step:3 Install Docker (In Master and Slave)

Now we have to install Docker because Docker images will be used for managing the

containers in the cluster. Run the following commands:

```
# sudo su
# apt-get update
# apt-get install -y docker.io
```

Step: 4 Next we have to install these 3 essential components for setting up Kubernetes environment: kubeadm, kubectl, and kubelet. (In Master and Slave)

Run the following commands before installing the Kubernetes environment.

```
root@kmaster-VirtualBox:~# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
root@kmaster-VirtualBox:~# cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
>deb http://apt.kubernetes.io/ kubernetes-xenial main
>EOF
root@kmaster-VirtualBox:~# apt-get update
```

Step 5: Install kubeadm, Kubelet And Kubectl (In Master and Slave)

Now its time to install the 3 essential components.Kubelet is the lowest level component in Kubernetes. It’s responsible for what’s running on an individual machine. Kuebadm is used for administrating the Kubernetes cluster. Kubectl is used for controlling the configurations on various nodes inside the cluster.

```
root@kmaster-VirtualBox:~# apt-get install -y kubelet kubeadm kubectl
```

```
root@knode-VirtualBox:~# apt-get install -y kubelet kubeadm kubectl
```

Step 6: Updating Kubernetes Configuration (In Master and Slave)

Next, we will change the configuration file of Kubernetes. Run the following command:

```
root@knode-VirtualBox:~# nano /etc/systemd/system/kubelet.service.d/10-
kubeadm.conf
```

This will open a text editor, enter the following line after the last “Environment Variable”:

```
Environment=”cgroup-driver=systemd/cgroup-driver=cgroupfs”
```

Step 7: we will individually set the configurations in both machines. (In Master and Slave) Comment on swap entry last line.

```
root@kmaster-VirtualBox:~# swapoff -a
```

```
root@kmaster-VirtualBox:~# nano /etc/fstab
```

1. Steps For Only Kubernetes Master VM (kmaster)

Step 1: We will now start our Kubernetes cluster from the master’s machine. Run the following command:

```
root@kmaster-VirtualBox:~# kubeadm init –apiserver-advertise-
address=192.168.0.101--pod-network-cidr=192.168.0.0/16
```

```
root@kmaster-VirtualBox:~# kubeadm init --apiserver-advertise-
address=192.168.0.101 --pod-network-cidr=192.168.0.0/16 --ignore-preflight-
errors=NumCPU
```

1. The commands marked as (1), execute them as a non-root user. This will enable you to use kubectl from the CLI

2. The command marked as (2) should also be saved for future. This will be used to join nodes to your cluster

Step 2: As mentioned before, run the commands from the above output as a non-root user

To verify, if kubectl is working or not, run the following command:

$ kubectl get pods -o wide --all-namespaces

Step 3: You will notice from the previous command, that all the pods are running except one: 'kube-dns'. For resolving this we will install a pod network. To install the CALICO pod network, run the following command:

kmaster@kmaster-VirtualBox:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

Step 4: Next, we will install the dashboard. To install the Dashboard, run the following command:

kmaster@kmaster-VirtualBox:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml

Step 5: Your dashboard is now ready with it's the pod in the running state.

Step 6: By default dashboard will not be visible on the Master VM. Run the following

command in the command line:

kmaster@kmaster-VirtualBox:~$ kubectl proxy

Then you will get something like this:

In Master Browser put this URL:

http://127.0.0.1:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/login

You will then be prompted with this page, to enter the credentials:

Step 7: In this step, we will create the service account for the dashboard and get it's credentials.

Note: Run all these commands in a new terminal, or your kubectl proxy command will stop.

Run the following commands:

1. This command will create a service account for dashboard in the default namespace

2. Copy this token and paste it in Dashboard Login Page, by selecting token option

3. You have successfully logged into your dashboard!

1. Steps For Only Kubernetes Node VM (knode)

2. **Note:** Run this command with "sudo".
3.

```
root@kmaster-VirtualBox:~# sudo kubeadm join 192.168.0.101:6443 --token
l6dcje.2ye0st1v7osokmab \ --discovery-token-ca-cert-hash
sha256:5c6a9f740a86a5a3c6a6bdf4fc37336e15a037e715c2402c7ab3675916b0e651
```

EXP4 :- **To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.**

In Order to configure Kubernetes Cluster we require 2 systems we can consider it as one master and one slave.

I have launch two Virtual machines master as kmaster and slave as knode as shown in Fig. Configure static IP as:

Master: 192.168.0.101 (kmaster)
Slave: 192.168.0.102 (knode)

Step 1: First create a folder inside which you will create your deployment and service. After that, use an editor and open a Deployment file.

Step 2: Once you open the deployment file, mention all the specifications for the application you want to deploy. Here I am trying to deploy an httpd application.

Relace httpd to nginx

Step 3: After you write your deployment file, apply the deployment using the following command.

Here -f is a flag name used for the file name.

Step 4: Now, once the deployment is applied, get the list of pods running.

Here, -o wide are used to know on which node is the deployment running.
Step 5: After you have created a deployment, now you have to create a service. For that again use an editor and open a blank service.yaml file.

```
nano service.yaml
```

Step 6: Once you open a service file, mention all the specifications for the service.

Relace httpd to nginx

Step 7: After you write your service file, apply the service file using the following command.

```
kubectl apply -f service.yaml
```

Step 8: Now, once your service is applied to check whether the service is running or not use the following command.

```
kubectl get svc
```

Step 9: Now, to see the specifications of service, and check which Endpoint it is binded to, use the following command.

```
kubectl describe svc <name of the service>
```

Step10: Check status in dashboard for Pods, Services, Replicas, etc.

EXP 5 :- **To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.**

Terraform Installation Steps on Ubunu18.04

Step: 1 Terraform uses HashiCorp Configuration Language (HCL) to manage environments of Operators and Infrastructure teams. To download go to site https://www.terraform.io/downloads.html

Select the appropriate package for your operating system and architecture.

Step:2 unzip the archive by using below command

Step 3: Change the directory to unzipped folder

and Move the terraform binary to a directory included in your system's PATH in my case *usr/*local/bin/

Step 4: To check whether Terraform is installed, run:

EXP 6 :- **To Build, change, and destroy AWS infrastructure Using Terraform.**

1. Install the AWS CLI version 2 on Linux

Follow these steps from the command line to install the AWS CLI on Linux.
Install curl on linux

```
vishal@apsit:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```
vishal@apsit:~$ sudo apt install unzip
```

```
vishal@apsit:~$ sudo unzip awscliv2.zip
```

```
vishal@apsit:~$ sudo ./aws/install
```

```
vishal@apsit:~$ aws --version
```

it should display the below outout.

```
aws-cli/2.1.29 Python/3.8.8 Linux/5.4.0-1038-aws exe/x86_64.ubuntu.18 prompt/off
```

2. Create a new access key if you don't have one. Make sure you download the keys in your local machine.

Login to AWS console, click on username and go to My security credentials

Continue on security credentials, click on access keys
Perform below commands in Linux where you have installed Terraform

First setup your access keys, secret keys and region code locally.

```
vishal@apsit:~$aws configure
```

You can check region as shown in below image :
Create one Directory for Terraform project in which all files of terraform we can save

```
vishal@apsit:~$ cd ~
vishal@apsit:~$ mkdir project-terraform
```

vishal@apsit:~$ cd project-terraform
Create Terraform Files

vishal@apsit:~$ sudo nano variables.tf
Give name to key pair file as **terraform**

Use your Region and Key name in variable.tf as shown and provide instance type which you want to create.

After creating variable terraform file note down the AMI ID of instance which u want to create which we will use to configure our instance in main.tf file.

Now create main.tf file:

```
provider "aws" {
  region = var.aws_region
}

#Create security group with firewall rules
resource "aws_security_group" "security_jenkins_port" {
  name        = "security_jenkins_port"
  description = "security group for jenkins"

  ingress {
   from_port   = 8080
    to_port     = 8080
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

 ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
# outbound from jenkis server
  egress {
    from_port   = 0
    to_port     = 65535
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags= {
    Name = "security_jenkins_port"
  }
}

resource "aws_instance" "myFirstInstance" {
  ami           = "ami-0b9064170e32bde34"
  key_name = var.key_name
  instance_type = var.instance_type
  security_groups= [ "security_jenkins_port"]
  tags= {
    Name = "jenkins_instance"
  }
}

# Create Elastic IP address
resource "aws_eip" "myFirstInstance" {
  vpc      = true
  instance = aws_instance.myFirstInstance.id
tags= {
    Name = "jenkins_elstic_ip"
  }
}
```

Put AMI-ID in above highlighted space and Now execute the below command:

you should see like below screenshot.

Execute the below command

```
terraform plan
terraform apply
```

Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

Now login to EC2 console, to see the new instances up and running, you can see Jenkins_instance is up and running which we deploy from terraform.

You can also check the security group resource details which you created from terraform :

Terraform destroy

you can also destroy or delete your instance by using terraform destroy command :

Now you can see instance which you created by using terraform is deleted successfully from aws console also you can check it will removed successfully:

EXP 7 :- **To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.**

1) Install and configure a Jenkins and SonarQube CICD environment using Docker containers.

```
manjusha@apsit:~$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key

add -
```

When the key is added, the system will return OK. Next, append the Debian package repository address to the server's sources.list:

```
manjusha@apsit:~$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
```

When both of these are in place, run update so that aptwill use the new repository:

```
manjusha@apsit:~$ sudo apt update
```

Finally, install Jenkins and its dependencies:

```
manjusha@apsit:~$sudo apt install jenkins
```

Let's start Jenkins using systemctl:

```
manjusha@apsit:~$sudo systemctl start jenkins
```

Since systemctl doesn’t display output, you can use its status command to verify that Jenkins started successfully:

```
manjusha@apsit:~$sudo systemctl status jenkins
```

If everything went well, the beginning of the output should show that the service is active and configured to start at boot:

Now that Jenkins is running, let’s adjust our firewall rules so that we can reach it from a web browser to complete the initial setup.

Opening the Firewall

By default, Jenkins runs on port 8080, so let’s open that port using ufw:

```
manjusha@apsit:~$sudo ufw allow 8080
```

Setting Up Jenkins

To set up your installation, visit Jenkins on its default port, 8080, using your server domain name or IP address: **http://your_server_ip_or_domain:8080**

You should see the Unlock Jenkins screen, which displays the location of the initial password:

In the terminal window, use the cat command to display the password:

```
manjusha@apsit:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Copy the 32-character alphanumeric password from the terminal and paste it into the Administrator password field, then click Continue.

The next screen presents the option of installing suggested plugins or selecting specific plugins:

We’ll click the Install suggested plugins option, which will immediately begin the installation

process:

When the installation is complete, you will be prompted to set up the first administrative user. It’s possible to skip this step and continue as admin using the initial password we used above, but we’ll take a moment to create the user.

After confirming the appropriate information, click Save and Finish. You will see a confirmation page confirming that “Jenkins is Ready!”:
Click Start using Jenkins to visit the main Jenkins dashboard:

SonarQube Setup

Before proceeding with the integration, we will setup SonarQube Instance. we are using SonarQube Docker Container.

manjusha@apsit:~$docker run -d -p 9000:9000 sonarqube

In the above command, we are forwarding port 9000 of the container to the port 9000 of the host machine as SonarQube is will run on port 9000. Then, from the browser, enter http://localhost:9000. After That, you will see the SonarQube is running. Then, login using default credentials (admin:admin).

1. Generate User Token

Now, we need to get the SonarQube user token to make connection between Jenkins and SonarQube. For the same, go to **Administration> User > My Account > Security** and then, from the bottom of the page you can create new tokens by clicking the Generate Button. Copy the Token and keep it safe.
C96798e9bd081e117189b516c868ddb7d87ee785   SonarQube

2) Configure Jenkins with the SonarQube Scanner plugin for automated static code analysis.

1. Jenkins Setup for SonarQube

Before all, we need to install the SonarQube Scanner plugin in Jenkins. For the same, go to **Manage Jenkins > Plugin Manager > Available.** From here, type SonarQube Scanner then select and install.

1. Tool Configuration SonarQube Scanner

Now, we need to configure the Jenkins plugin for SonarQube Scanner to make a connection with the SonarQube Instance. For that, got to **Manage Jenkins > Configure System > SonarQube Server.** Then, Add SonarQube. In this, give the Installation Name, Server URL then Add the Authentication token in the Jenkins Credential Manager and select the same in the configuration.

Then, we need to set-up the SonarQube Scanner to scan the source code in the various stage. For the same, go to **Manage Jenkins > Global Tool Configuration > SonarQube Scanner**. Then, Click **Add SonarQube Scanner Button**. From there, give some name of the scanner type and **Add Installer** of your choice. In this case, I have selected SonarQube Scanner from Maven Central.

SonarQube Scanner in Jenkins Pipeline

Now, It’s time to integrate the SonarQube Scanner in the Jenkins Pipeline. For the same, we are going to add one more stage in the Jenkinsfile called SonarQube and inside that, I am adding the following settings and code.

EXP 8 :- **Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Java application.**

Steps:

1) Install and configure a Jenkins and SonarQube CICD environment using Docker containers.

2) Configure Jenkins with the SonarQube Scanner plugin for automated static code analysis.

3) Create and set up a Jenkins build pipeline using a Jenkinsfile stored within a GitHub repo.

4) Use the SonarQube web application to examine and review the generated static analysis report.

5) Use the Blue Ocean Plugin to review Pipeline Steps.

Note: From Step 1 and 2 we have already done in Expt. 7 as a Pre-requiste required for Integration settings of Jenkins SAST with SonarQube so in this Experiment we will continue from 3rd Step.

Check the contents of jenkins-sonarqube repository which we are using for Pipeline Project.

Check path for the Source and Test Java Programs from repository

Provide sonar host as http://127.0.0.1:9000 in POM.xml which is available in Project on Github.

To integrate the SonarQube Scanner in the Jenkins Pipeline. For the same, we are going to add one more stage in the Jenkinsfile called SonarQube and inside that, I am adding the following settings and code.

Github Repository Configuration in Jenkins Pipeline Project

Pipeline Script where stages are written along with scanner tool, repository path for source and test Java sample program, SonarQube Credential for integration, Application name on sonarqube, code language,etc.

After creating a Pipeline Script Build it in Jenkins , Click on save and then Click on Build Now

Click on Console Output to check output whether build is successful or not.

Click on Pipeline Steps to check Sequence of events during building of pipeline.

Also you can use Blue Ocean to check Pipeline execution stage by stage and log of the pipeline too.

If you login to the SonarQube and visit the Dashboard, you will see the Analysis of the project there.

For Detailed Report for code analysis you can go to application overview and check for all Bugs, Vulnerabilities, code smells and all parameters as shown in below image.

Since we have both Jenkins and SonarQube in the Enterprise standard, we have a lot of features including the alert system. Where we can configure the Email, or Instance message Notification system for the findings in the SonarQube or Jenkins. In the best case, we can auto convert certain bugs or findings as ticket and assign to the respective developer as a one option.

EXP 9 :- **To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.**

1 - Pre-requisite

First requirement is to install Apache and PHP first. Use the following commands to complete it. And use commands to install required packages for Nagios.

```
manjusha@apsit:~$ sudo apt-get update
manjusha@apsit:~$ sudo apt-get install wget build-essential unzip openssl libssl-dev
manjusha@apsit:~$ sudo apt-get install apache2 php libapache2-mod-php php-gd libgd-dev
```

1. 2 – Create Nagios User

Create a new user account for Nagios in your system and assign a password.

```
manjusha@apsit:~$ sudo adduser nagios
```

Now create a group for Nagios setup “nagcmd” and add nagios user to this group. Also, add nagios user in the Apache group.

```
manjusha@apsit:~$sudo groupadd nagcmd
manjusha@apsit:~$sudo usermod -a -G nagcmd nagios
manjusha@apsit:~$sudo usermod -a -G nagcmd www-data
```

2. Step 3 – Install Nagios Core Service

After installing required dependencies and adding user accounts and Nagios core installation. Download latest Nagios core service from the official site.

```
manjusha@apsit:~$cd /opt/

manjusha@apsit:~$sudo wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.3.tar.gz
manjusha@apsit:~$sudo tar xzf nagios-4.4.3.tar.gz
```

After extracting naviate to nagios source directory and install using make command.

```
manjusha@apsit:~$cd nagios-4.4.3

manjusha@apsit:~$sudo ./configure --with-command-group=nagcmd
manjusha@apsit:~$sudo make all
manjusha@apsit:~$sudo make install


manjusha@apsit:~$sudo make install-init
manjusha@apsit:~$sudo make install-daemoninit
manjusha@apsit:~$sudo make install-config
manjusha@apsit:~$sudo make install-commandmode
manjusha@apsit:~$sudo make install-exfoliation
```

Now copy event handlers scripts under libexec directory. These binaries provides multiple events triggers for your Nagios web interface.

```
manjusha@apsit:~$sudo cp -R contrib/eventhandlers/ /usr/local/nagios/libexec/

manjusha@apsit:~$sudo chown -R nagios:nagios /usr/local/nagios/libexec/eventhandlers
```

3. Step 4 – Setup Apache with Authentication

Now create an Apache configuration file for your Nagios server as below:

```
manjusha@apsit:~$sudo nano /etc/apache2/conf-available/nagios.conf
```

Add below lines to nagios.conf file.

```
ScriptAlias /nagios/cgi-bin "/usr/local/nagios/sbin"

<Directory "/usr/local/nagios/sbin">

   Options ExecCGI
   AllowOverride None
   Order allow,deny
   Allow from all
   AuthName "Restricted Area"
   AuthType Basic
   AuthUserFile /usr/local/nagios/etc/htpasswd.users
   Require valid-user
</Directory>


Alias /nagios "/usr/local/nagios/share"
```

```
<Directory "/usr/local/nagios/share">
   Options None
   AllowOverride None
   Order allow,deny
   Allow from all
   AuthName "Restricted Area"
   AuthType Basic
   AuthUserFile /usr/local/nagios/etc/htpasswd.users
   Require valid-user
</Directory>
```

To setup apache authentication for user **nagiosadmin**

```
manjusha@apsit:~$sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
```

Enable Apache configuration and restart Apache service to make the new settings take effect.cd

```
manjusha@apsit:~$sudo a2enconf nagios
manjusha@apsit:~$sudo a2enmod cgi rewrite
manjusha@apsit:~$sudo service apache2 restart
```

4. Step 5 – Installing Nagios Plugins

After installing and configuring Nagios core service, Download latest nagios-plugins source and install using follocdwing commands.

```
manjusha@apsit:~$cd /opt
manjusha@apsit:~$sudo wget http://www.nagios-plugins.org/download/nagios-plugins-2.2.1.tar.gz
manjusha@apsit:~$sudo tar xzf nagios-plugins-2.2.1.tar.gznagios
manjusha@apsit:~$cd nagios-plugins-2.2.1
```

Now compile and install Nagios plugins

```
manjusha@apsit:~$sudo ./configure --with-nagios-user=nagios --with-nagios-group=nagios --with-openssl
manjusha@apsit:~$sudo make
manjusha@apsit:~$sudo make install
```

5. Step 6 – Verify Settings

Use the Nagios commands to verify the Nagios installation and configuration file. After successfully verify start the Nagios core service.

```
manjusha@apsit:~$/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
manjusha@apsit:~$ sudo service nagios start
```

Also configure Nagios to auto start on system boot.

6. Step 7 – Access Nagios Web Interface

Access your nagios setup by access nagios server using hostname or ip address followed by /nagios.

http://127.0.0.1/nagios/

Prompting for Apache Authentication Password –

username: nagiosadmin
Password : 123456 (which you enter while configuration)

We have successfully installed and configured Nagios Monitoring Server core service in our system now we need to install NRPE on all remote Linux systems to monitor with Nagios.

EXP 10 :- **To perform Port, Service monitoring, Linux server monitoring using Nagios.**

1. Step 1 – Configure NRPE on Linux Host

Follow the below steps to install and configure NRPE on client machine and check connectivity with Nagios server.

1. Step 1.1 – Install NRPE

```
manjusha@apsit:~$  sudo apt-get install nagios-nrpe-server nagios-plugins
```

2. Step 1.2 – Configure NRPE

After successfully installing NRPE service, Edit nrpe configuration file /etc/nagios/nrpe.cfg in your favorite editor and add your nagios service ip in allowed hosts.

```
manjusha@apsit:~$ sudo nano /etc/nagios/nrpe.cfg
```

```
allowed_hosts=127.0.0.1, 192.168.64.3, 192.168.1.100
```

Where **192.168.1.100** is your Nagios server ip address.

After making above changes in nrpe configuration file, Lets restart NRPE service as per your system

```
manjusha@apsit:~$ sudo /etc/init.d/nagios-nrpe-server restart
```

ng

3. Step 1.3 – Verify Connectivity from Nagios

Now run the below command from Nagios server to make sure your nagios is able to connect nrpe client on remote Linux system. Here **192.168.64.3** is your remote Linux system ip.

```
manjusha@apsit:~$  /usr/local/nagios/libexec/check_nrpe -H 192.168.64.3
NRPE v2.15
```

2.

3. Step 2 – Add Linux Host in Nagios

First create a configuration file using below values. for example you Linux hosts ip is . We also need to define a service with host. So add a ping check service, which will continuously check that host is up or not.

```
manjusha@apsit:~$  sudo nano /usr/local/nagios/etc/servers/MyLinuxHost001.cfg
```

```
define host {
        use                     linux-server
        host_name                   Linux_Host_001
        alias                   Linux Host 001
        address                   192.168.64.3
        register                  1
}
define service{
      host_name                     Linux_Host_001
      service_description            PING
      check_command                   check_ping!100.0,20%!500.0,60%
      max_check_attempts              2
      check_interval                2
      retry_interval               2
      check_period                 24x7
      check_freshness               1
      contact_groups               admins
      notification_interval          2
      notification_period            24x7
      notifications_enabled           1
      register                    1
}
```

Now verify configuration files using following command. If there are no errors found in configuration, restart nagios service.

```
manjusha@apsit:~$ sudo  nagios -v /usr/local/nagios/etc/nagios.cfg
manjusha@apsit:~$ sudo service nagios restart
```

4. Step 3 – Check Host in Nagios Web Interface

Open your Nagios web interface and check for new Linux hosts added in Nagios core service.

EXP 11:- **To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.**

Steps: First Lambda functions using Python

1. Open Aws Console and search for Lambda Service and open home screen of Lambda.

2. Choose region in which you need to create Lambda function as it is region specific.

3. Create sum as a Lambda Function in Python Language so select latest version of Python and choose role with basic Lambda Permission to allow cloudwatch for monitoring.
4. Lambda sum function is created successfully

5. Write a sample python code for sum of two numbers:
6. Configure Test Event in Json Format

Write a sample Second sample python Code:

Configure Test Event

If condition met returns a value as apsit

EXP 12 ;- To create a Lambda function which will log "An Image has been added" once you add an object to a specific bucket in S3

Theory:

1.
1. Creating S3 Bucket

Let us start first by creating a s3 bucket in AWS console using the steps given below −

Step 1

Go to Amazon services and click **S3** in storage section as highlighted in the image given below −

Step 2

Click **S3** storage and **Create bucket** which will store the files uploaded.

Step 3

Once you click Create bucket button, you can see a screen as follows −

Step 4

Enter the details Bucket name, Select the Region and click Create button at the bottom left side. Thus, we have created bucket with name :

Step 5

Now, click the bucket name and it will ask you to upload files as shown below −

Thus, we are done with bucket creation in S3.

Create Role that Works with S3 and Lambda

To create role that works with S3 and Lambda, please follow the Steps given below –

Step 1

Go to AWS services and select IAM as shown below −

Step 2

Now, click **IAM -> Roles** as shown below −

Step 3

Now, click **Create role** and choose the services that will use this role. Select Lambda and click **Permission** button.

Step 4

Add the permission from below and click Review.

AmazonS3FullAccess, AWSLambdaFullAccess and CloudWatchFullAccess.

Step 5

Observe that we have chosen the following permissions −

Observe that the Policies that we have selected are **AmazonS3FullAccess, AWSLambdaFullAccess and CloudWatchFullAccess.**

Step 6

Now, enter the Role name, Role description and click Create Role button at the bottom

Thus, our role named lambdawiths3service is created.

Create Lambda function and Add S3 Trigger

In this section, let us see how to create a Lambda function and add a S3 trigger to it. For this purpose, you will have to follow th Steps given below −

Step 1

Go to AWS Services and select Lambda as shown below −

Step 2

Click **Lambda** and follow the process for adding **Name**. Choose the **Runtime, Role** etc. and create the function. The Lambda function that we have created is shown in the screenshot below −

Step 3

Now let us add the S3 trigger.

Step 4

Choose the trigger from above and add the details as shown below −

You can add Prefix and File pattern which are used to filter the files added. For Example, to trigger lambda only for .jpg images. as we need to trigger Lambda for all jpg image files uploaded. Click Add button to add the trigger.

Step 5

You can find the the trigger display for the Lambda function as shown below −

Step 6

Let's add the details for the aws lambda function. Here, we will use the online editor to add our code and use nodejs as the runtime environment.

To trigger S3 with AWS Lambda, we will have to use S3 event in the code as shown below −

Step 7:

let us save the changes and test the lambda function with S3upload.

Step 8:

Now, save the Lambda function. Open S3 from Amazon services and open the bucket we created earlier namely lambdawiths3.

Upload the image in it as shown below −

Click **Add files** to add files. You can also drag and drop the files. Now, click **Upload** button.

Thus, we have uploaded one image in our S3 bucket.

Step 9

To see the trigger details, go to AWS service and select CloudWatch. Open the logs for the Lambda

AWS Lambda function gets triggered when file is uploaded in S3 bucket and the details are logged in Cloudwatch as shown below −

An image has been Added -> apsit_logo.jpg you can see in cloudwatch logs.

EXP 13 :- To demonstrate working of cloud launcher to launch the web application and Manage and Monitor the Application.

To demonstrate the working of **Google Cloud Launcher** (**Google Cloud Marketplace**) to launch a web application, The purpose of this lab is to deploy a web application using a pre-configured environment from the Google Cloud Marketplace and

Prerequisites

- A **Google Cloud Platform (GCP)** account.
- **Billing enabled** on your GCP account (to avoid deployment restrictions).
- Basic knowledge of **web applications** (e.g., web servers, databases).
- Familiarity with the **Google Cloud Console**.

Step 1: Setting Up Your GCP Account

1. **Login to Google Cloud Console**:
    - Visit the Google Cloud Console.
    - Sign in using your Google account credentials.

Create a New Project:

- Click on the **project drop-down** on the top-left corner of the console.

Select **New Project**.

- Name the project (e.g., "Cloud Launcher Demo") and select your billing account.
- Click **Create**

Step 2: Navigating Google Cloud Marketplace

1. **Open Google Cloud Marketplace**:
    - In the **left-hand menu**, navigate to **Marketplace**.
    - This section allows you to find pre-configured solutions such as WordPress, LAMP stack, or Django, as well as other web application frameworks and services.
2. **Search for a Web Application**:
    - Use the **search bar** in the Marketplace to find a web application you want to deploy. For this demo, let’s deploy a simple **WordPress** instance.
    - Type **"WordPress"** and select the official package from the search results.

Step 3: Deploying the Web Application

1. **Launch the WordPress App**:

- On the WordPress product page, click **Launch on Compute Engine**.
- You'll be taken to a page to configure the deployment.

2. **Configure Deployment Settings**:
   - Choose the **zone** where you want to deploy the application (e.g., `us-central1`).
   - Select the **machine type** (e.g., `e2-small`) which will allocate the appropriate CPU and memory resources.
   - **Disk type**: Choose either the default **Persistent Disk** or **SSD** (for faster performance).
   - **Networking**: Leave the default networking settings, or create a new Virtual Private Cloud (VPC) if necessary.
3. **Click Deploy**:
   - After selecting your settings, click **Deploy** to start the process. GCP will automatically create the necessary infrastructure, including a virtual machine (VM), storage, and network configuration.

Step 4-Accessing the Web Application

1. **Monitor Deployment Progress**:

See the progress of the deployment in the Google Cloud Console. This includes setting up the virtual machine, installing WordPress, and configuring the environment.

2. **Get the External IP Address**:

   Once deployment is complete, go to the **VM instances** page from the left-hand menu under **Compute Engine**.

   Locate the deployed WordPress instance, and note the **External IP Address** assigned to the VM.

3. **Access the Web Application**:
   - Open a new browser tab and navigate to the External IP address you noted.
   - You should see the WordPress installation page, where you can set up the site (e.g., language, admin credential

Step 5 -Configuring the Web Application

1. **Set Up WordPress**:
   - Follow the on-screen instructions to complete the WordPress setup.
   - Choose your site's title, create an admin username, and password.
   - After setup, log in to the WordPress admin panel to customize your site.

2. **Explore the Admin Dashboard**:
   - Once logged in, explore the admin dashboard where you can add new content, change the theme, or install plugins.

Step 6: Manage and Monitor the Application

1. **View Logs and Performance**:
   - Go to the **Operations** menu in the Cloud Console and select **Logs Explorer**.
   - This allows you to view real-time logs from the web server, which can be useful for troubleshooting and monitoring application performance.

1. **Scaling and Auto-healing**:
   - Discuss how you can modify the configuration to enable auto-scaling and high availability.
   - For advanced users, demonstrate how to enable **Cloud Monitoring** and **Alerting** to get notified if the web server is under heavy load or goes down.

Step 7 : Clean Up Resources

1. **Delete Resources**:
   - To avoid unnecessary billing charges, delete the resources you created after the lab.
   - Go to **Compute Engine > VM Instances**, and select the instance you deployed (WordPress).
   - Click **Delete** to remove the virtual machine and associated resources.
2. **Verify Billing**:
   - Check your **billing dashboard** to ensure there are no active resources still incurring charges.