EXP!:- 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.

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 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 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.