Experiment – 2 Cloud9 Python

AWS Cloud9 IDE.

1.Aim: To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE, write and run simple python program in IDE.
1. Login to AWS account.
2. Check EC2 and cloudFormation dashboard. Make sure no instances and stack running for your account.
3. Navigate to Cloud 9 IDE service from Developer tools section
4. Click on Create Environment
5. Provide name for the Environment (WebAppIDE) and click on next.
6. Keep all the Default settings as it is
7. Review the Environment name and Settings and click on Create Environment
8. Go to EC2 dashboard to ensure new instance running.
9. Go to CloudFormation to ensure new stack created. check resources and templates tabs.
10. Launch IDE
11. run some commands on terminal.
12. write simple python program in an IDE
13. save changes to py file , run the code and check the result in terminal.
14. Click on settings option. Change some of the settings.

Experiment – 3 Cloud9 Collaborate

A lambda function using cloud9 IDE and collaborate with
other users over cloud9 IDE.

Aim: To create, deploy and invoke a lambda function using cloud9 IDE and collaborate with other users over cloud9 IDE.
7. Laboratory Exercise
Create, deploy and invoke a lambda function
1. AWS lambda console, check if no function present in youir account.
2. Go to aws explorer of cloud9 IDE
3. create lambda SAM application
4. select python 3.7 version
5. select SAM hello world basic app

6. select work env folder
7. give name to your application
8. from folder open to app.py lambda python file
9. make changes to this file
10. save and deploy
11. select s3 bucket to deploy code
12. create and name your own bucket
13. create cloud formation stack
14. terminal window will show stack updated status.
15. from IDE

16. right click o lambda function
17. select invoke on AWS
18. check s3 for new bucket.

Creating IAM user for collaboration
1. In other tab -Open IAM Identity and Access Management to Add User.
2. Give user console access , Provide user with custom password.
3. Create group from IAM
4. Provide group name and click on create group.
5. Navigate to user Groups from left pane in IAM.
6. click on your group name which you have created and nevigate to permission tab
7. Add permission and select Attach Policy after that search for Cloud9 related policy and select Awscloud9EnviornmentMember policy and add it.
8. Move towards cloud9 IDE Enviornment tab
9. Coud9 IDE has 1. file organiser, 2. coding window and 3. aws integrated CLI with some preinstalled softwares like git, node, python.
10. for command operations: check git, node and python version, iam user details etc.
11. Lets setup collaborative enviornment
12. Click on File - choose from template, select html file to collaborate.
13. Edit html file and save it
14. Share this file to collaborate with other members of the team. Click on Share option on Top Right Pane, write username which you created in IAM , check the accesses given (RW) and send Invite. Click on Done. Click OK for Security warning.
15. Now Open your Browsers Incognito Window and login with IAM user which you configured before.
16. After Successful login with IAM user open Cloud9 service from dashboard services and click on shared with you enviornment to collaborate.
17. Click on Open IDE you will see same interface as your other member have to collaborate in real time, also you all within team can do group chats
18. You can also explore settings where you can update permissions of your teammates as from RW to R only or you can remove user too.

Experiment – 4 App CodePipeline EC2

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

Aim: 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
Step 2: Get the copy of Sample Code
In this step you will retrieve the sample app's code and choose a source to host the code.
Pipeline takes the code and performs actions on it.
You can use three options.
● GitHub Repository
● Amazon S3 Bucket
● AWS CodeCommit Repository
Open Amazon S3 console and Create your S3 Bucket:
Create Bucket
Upload the code to Bucket
Select Bucket and Click on Upload (Right Corner)
Add Files -- upload zip file from downloads of your computer. Click on Upload Button
Step 3: Create 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.
Create new pipeline: Source Provider = Amazon S3, Bucket = VaishaliBucket1, S3 Object Key = Copy S3 Uri from Amazon S3 bucket.
Add Deploy Stage:
Review the settings and create the pipeline.
Add deploy stage and Give details : Deployement provider : AWS Elastic Beanstalk, Region,Application Name and Environment Name...(Auto suggested.)
Pipeline created successfully.
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
In this step, you will revise the sample code and commit the change to your repository.
CodePipeline will detect your updated sample code and then automatically initiate deploying it to
1.Visit your own copy of the repository that you forked in GitHub.
● Open index.html
● Select edit icon
2.Update the webpage by copying and pasting the following text on line 30.
3.Commit the change to your repository.
4.Return to your pipeline in the CodePipeline console. In a few minutes, you should see the Source change to blue, indicating that the pipeline has detected the changes you made to your source repository. Once this occurs, it will automatically move the updated code to Elastic Beanstalk.
● After the pipeline status displays Succeeded, in the status area for the Beta stage, click AWS Elastic Beanstalk.

5. The AWS Elastic Beanstalk console opens with the details of the deployment. Select the environment you created earlier. And click the URL again from EBS environment again.

Step 6: Clean up the resources

To avoid future charges, you will delete all the resources you launched which includes the pipeline, the Elastic Beanstalk application, and the source you set up to host the code.
First, you will delete your pipeline:
● In the pipeline view, click Edit.
● Click Delete.
● Type in the name of your pipeline and click Delete.
Second, delete your Elastic Beanstalk application:
● Visit the Elastic Beanstalk console.
● Click Actions.
● Then click Terminate Environment.
● We have successfully created an automated software release pipeline using AWS CodePipeline! Using CodePipeline, We created a pipeline that uses GitHub, Amazon S3, or AWS CodeCommit as the source location for application code and then deploys the code to an Amazon EC2 instance managed by AWS Elastic Beanstalk. Pipeline will automatically deploy your code every time there is a code change.

Experiment – 7 Terraform

To understand Terraform lifecycle, basic concepts /terminologies and install it on Windows/Linux machine.

Aim: To understand Terraform lifecycle, basic concepts/terminologies and install it on Windows /Linux machine.
Step 1 : Download appropriate terraform package(.zip) from terraform.io/downloads.html for Windows

Step 2: Download Terraform for Windows 64-bit / (32-bit).
Step 3: Create a folder 'terraform' in drive C .
Step 4 : Extract downloaded zip in to this c:/terraform folder.
Step 5: Now we need to set path for terraform. Go to My computer/ This PC, right click, select propoerties, go to advanced system settings.
Step 6: click on environment variable
Step 7: click on New, give variable name = Path, click on browse directory, select c:/terraforms/terra....exe, OK.
Step 8: Cross verify terraform installed properly or not . go to MS Powershell, run as a administrator
Step 9: Type terraform
Step 10: You will find init, validate, plan, apply and destroy options means you have installed terraform succesfully.
Step 11: Create a folder c:\SfitApp.
Step 12 : Open Atom/VS CODE Editor ...Open folder SfitApp
Step 13: write main.tf file with input variables. The input variables, like the one above, use a couple of different types: string, list, map, and Boolean.
Step 14: Check the output on command Prompt...Go to C:\SfitApp, Type Terraform Console
You will get terraform prompt, run the .tf with var.MyVariable1, You will get welcome to SFIT msg.
Step 15: try Boolean variable...Create Boolean.tf

```
Variable "MyVariable1" {
        type = string
        Default = "Hello Lian!"
}
Terraform console
var.MyVariable1
```

variable “Password”{default = false}
var.Password

Experiment – 8 Terraform EC2

To build, apply and destroy AWS Resources using Terraform.

Aim: To build, apply and destroy AWS using Terraform.
Step 1: First we will check that no instance is running on EC2.
Step 2: Create an IAM user with Programmatic Password, Administrator access and download access key and secret key from download.csv
IAM:
USERS:
ADD USER:
Give console access
I want to create user
custom pwd
Attach policy directly : Administrator Access
Review:
Create user
Clk on user name
Create access key
Command line interface
Create access key
Step 3: Now write a Terraform program in vs code, create new file with .tf extension

```
provider "aws"{
        access_key "AKIA5QMWP7VMW4ENZDMW" =
        secret_key = "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"
        region = "us-east-1"
}
resource "aws_instance" "terraform_sfit"{
        ami = "ami-065b889ab5c33720e"
        instance_type "t2.micro" =
}
```

In Launch instance, you will get ami : amazon machine image
For Instance type : t2.micro is freely available

Step 4: Now initialize the terraform ...type c:\SfitApp> terraform init
Terraform has been initialized successfully.
Step 5: c:\sfitApp>terraform plan
Step 6: Check the instance on Ec2 before terraform apply
Instance is not yet created.
Step 7: Terraform apply
Step 8: Check terraform created instance on EC2...we have created 3 instances.
Step 9: Now destroy the instance from command prompt....c:\SfitApp> terraform destroy

Experiment – 10 SonarQube

To test code (python files) with SonarQube and observe, analyse the results.

Aim: To perform analysis of the code to detect bugs, code smells, security vulnerabilities on a Python application.
Step 1: StartSonar server from SonarQube folder
Step 2: : on browser chk https://localhost:9000
Step 3: By default Username and Password is admin. You can change the password here.
Step 4: Login with new password.
Step 5: Click on Manually. Create a Project with name
Step 6: Click on Locally and given token name
Step 7: Give token name and continue
Step 8: Select what type of project you want to test
Step 9: Click on Other and select OS as Windows
Step 10: Create a folder to keep python scripts
Step 11: Start configuration of the scanner. Add following lines to C:/SonarScanner/conf/Sonar-Scanner.properties

```
#No information about specific project should appear here
#----- Default SonarQube server
#sonar.host.url=http://localhost:9000
#----- Default source code encoding
#sonar.sourceEncoding=UTF-8
sonar.projectKey=
sonar.projectName=
sonar.projectVersion=1.0
sonar.sources=
```

Step 12: Open new command prompt to Run and scan python files

```
C:/SFITJobs/PY Scripts >sonar-scanner.bat -D"sonar.projectKey=SFIT_SAST" -D"sonar.sources=."
-D"sonar.host.url=http://localhost:9000" -D"sonar.login=edabde9219cd89080688d93f3ff57ee8ba4caaf4"
```

Step 13: SonarQube command will give output on dashboard. Command Prompt is showing

Experiment – 11 Lamda Function

To understand AWS Lambda, its workflow, and to create the first Lambda function using Python/Java.

Aim: To write first Lambda function using Python/Java/Node.js.
a. Perfoprm following steps (attach screenshots)
- Enter Lambda service from console
- Select a Lambda Blueprint
- Configure and create Lambda function
- Invoke Lambda function and verify results
- AWS Lambda automatically monitors Lambda functions and reports metrics
- Clean up Lambda function

Experiment – 12 An object has been added

To create AWS Lambda function to log “an object has been added” on adding the object to s3 bucket.

Aim: To author AWS Lambda function from scratch to automatically create and upload non empty json file over s3 bucket and print log “an object has been added” on lambda.
a. Perfoprm following steps (attach screenshots)

- Create Execution role using IAM
- policies attached to the created role
- empty s3 bucket
- create Lambda function from console
- create new event for lambda function
- lambda function script
- Invoke Lambda function and verify results
- AWS Lambda automatically monitors Lambda functions and reports metrics
- Clean up resources