

Exp 4 (Collaboration)

Steps

- i) Go to IAM module
- ii) On Dashboard click on Users → Add Users
- iii) Enter username for 1st then add another user and enter uname
- iv) Give access type → Password → Custom password
Enter the custom password
- v) Don't allow them to change password
- vi) Next → Add users to grp
- vii) Create group → Give grp name
- viii) Then give rights for access Cloud9 i.e. AWSCloud9Users
→ Create group
- ix) Next → Tags → Next → Create Click on Create Users
- x) Uid will be generated copy and paste if somehow
- xi) Click on Uid
- xii) Sign with 1st user → Go to Cloud9 → Create environment
→ Give the name of environment, short description
- xiii) Environment settings: Environment → Keep everything default
platform → Ubuntu 30 minutes time then → Next
Create environment
- Create a fab in incognito-mode (or other browser)
- Copy and paste the link → sign with 2nd user
- Cloud9
- Go to user 1 → Share option → invite member using uname
Done
- Go to user 2 → dashboard of Cloud9 → shared with u → open IDE

CICD

Exp 2

- 1-) Go to amazon elastic beanstalk. Create application.
- 2-) Give name, choose platform → tomcat, sample application everything else default then create app
- 3-) Go to github → repositories → consist of your file
- 4-) Create Codepipeline
→ create pipeline → (name) everything default → next
source → Github (version 2).
Connection → connect to github → only select repo
(watch record) → connect
choose repos name, branch name else default → next
- 5-) Build provider → AWS CodeBuild
Create project → name, OS → Ubuntu, runtime → standard
image → any
Buildspec → use a buildspec file
else default → continue to codepipeline
- 6-) Singe build → next
- 7-) Deploy stage → elastic beanstalk → next
- 8-) Review create pipeline

After pipeline is created successfully

Go to Deploy → elastic beanstalk → test

→ Edit code in github and commit

Exp 3

1) Go to E2 → Ubuntu 20.04 → 2 instance

Storage - Default → Security group

Security group add with default

i) HTTP → port 80 source → Anywhere

ii) HTTPS → 443 " " " "

iii) Custom TCP → 8888 6443 " " " "

→ Launch instances, Name them :- k8s Master K8s Worker

→ Now, select master and fire connect

→ same with worker

→ clear master and worker, sudo su

→ Now change host name of master :-

→ hostnamectl set-hostname master-node

→ ~~same with~~ then exit

→ sudo su

→ Same with worker node

To change color (optional)

→ master → ~~tput setaf~~ tput setaf 02

worker → tput setaf 01

Now commands

→ apt-get update (both)

→ apt-get install docker.io -y (both)

→ docker --version (both) (to check version)

- systemctl enable docker } (both)
- systemctl status docker } (both)
- systemctl start docker }

- Add Kubernetes signing key - (both)


```
curl -s https://packages.cloud.google.com/opt/c/apt/keys.gpg | apt-key add
```

- Add Software Repositories (both)


```
apt-get update
apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
```

- apt-get install kubeadm kubeadm kubectl -y (both)
- apt-mark hold kubeadm kubeadm kubectl (both)
- kubeadm version (both)

- swapoff --a (both)

- ```
kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all
```

Master
  
- Now copy the output & get bind paste it somewhere (Your Kubernetes service )
  
- then ~~if~~ enter the 3 commands (master)

→ Then Deploy Pod Network to Cluster (master)

```
kubectl apply -f https://raw.githubusercontent.com/
coreos/flannel/master/Documentation/kube-flanneld.yaml
```

→ kubectl get pods --all-namespaces (master)

→ Now copy paste from word document of (workers)  
 kubeadm join ... (command)

→ kubectl get nodes (master)

3rd for exp done

## Exp 4

S

- 1) Create deployment named nginx
- kubectl create deployment nginx --image=nginx
- kubectl expose deploy nginx --port 80 --target-port 80  
-- type NodePort
- kubectl get services
- \* Here we will get port number in ports column greater than 30000
- \* Now go to instances running, select workers node instance then open public ip address
- \* Then in browser to put : port number we get <sup>preferably</sup>  
of any (master or workers)
- \* This will not work, go to security group and add another group : custom TCP with some port number and source Anywhere  $\rightarrow$  Save rule.
- \* Now again open the browser from workers public ip address and further put : port number. Also remove https  $\rightarrow$  as the site is not secured

## Scale up replicas

- `kubectl scale --current-replicas=1 --replicas=2 deployment/nginx`
- Output : you will see 2/2
- `kubectl describe deployment/nginx` (gives detail of service deployed)
- `kubectl delete service nginx`
- Now refresh that page in web-browser u won't be able to access
- To check if it has been deleted
  - `kubectl get services`

## Exp 5 .

- First launch instance keeping everything default
- then connect
- sudo su, clear
- wget (then link address terraform site)

→ Then we need to un-zip

- ~~apt~~ apt install unzip
- then unzip the zip file using "unzip <filename>"

→ ~~ls~~ → we get list of files

→ Setting path → mv terraform /usr/local/bin/

→ terraform

## Exp 11-12

### AWS Lambda Theory :

→ Code run on AWS Lambda is called Lambda Function.

AWS supports Serverless

:- we don't need to manage servers. AWS does it.

Step 1 :- Create IAM role.

→ Go to IAM → roles → Create role

type → AWS service → Lambda → next

Permissions policies

→ Lambda, full key words search → tick on AWSLambdaFullAccess

→ s3, full access tick

→ CloudWatchLogs, full access

→ next,

Role details :

→ Give Role name :- awsdemo → Create role

→ Now role is created

IAM is over

→ Now go to Lambda

Now go to S3

Step 2 : Create a bucket in AWS S3 to upload image.

Ensure region of bucket is same as that of Lambda function

→ Go to S3 → Create bucket

→ Uncheck block all public access (We don't want to block public access)

→ Tick I acknowledge

→ Then click on Create bucket

Step 3 : Create a Lambda function

→ Go to Lambda → Create Function

→ Use a Blueprint

→ Filter blueprints → Blueprint attribute = S3

→ Now go with S3 object (above's) → Then click on config

Basic Info

→ function name → Use an existing role → select the role created just now

&

S3 triggers

→ Bucket select bucket created if, keep everything default, tick I acknowledge then create function

→ Now go to configuration and increase timeout to 20 sec

- Now go to Test
  - add event name
- Now go to S3 and upload an image
- Now again go to Lambda check monitor → view dogs in cloudwatch
- Now when we click on log we will see what all happened

~~Done~~ \* \* \*

If want customization (Upload image through code)

- Go to the test → Create event → enter event name
- Now we need to change code so go to S3.
- Select S3 bucket and click on copy ARN now paste this ARN in code
- Now in code also replace the bucket name with yours
- Now in code → object → key → change to image/bg2.png
- Go to the bucket where image is uploaded and then go to Settings and enable e-Tag
- Now copy past the e-Tag in code
- Save and Test
- Execution result: succeeded (Logs) → Click on logs
- You will be redirect to cloudwatch
- Go to Code in Lambda → index.js → In try block add console.log ('An image has been added')
- Now deploy & test
- Take S3 if image has been added

resource "aws\_instance" "terraform-VIT" {  
 ami = "ami-000000000000"  
 instance\_type = "t2.micro" → variable  
 } → t2.micro

tab ↴

ami = "ami-000000000000"  
 instance\_type = "t2.micro"

3

Now get EC2 Launch instance and in Ubuntu 20.04  
 Look for ami id copy that in

- Now Ctrl-S to save and the Ctrl-X
- terraform init
- terraform plan -lock=false
- terraform apply -lock=false  
 (New instance will be created check in EC2)

Change

- nano main.tf
- Now below instance\_type add flags = {}

Name = "Terraform-DemoInst"

- terraform Now save & Ctrl-X
- terraform init
- terraform plan -lock=false
- terraform apply -lock=false
- terraform

Destroy

- terraform destroy

## Exp 6

- 1.) Launch instance (Ubuntu 20.04)
- 2.) Click on instance-id, rename → then connect
- 3.) sudo su → clear
- 4.) wget (~~link~~ google drive 386 - ~~CVE~~ link address)
- 5.) apt install uzip then uzip filername(.zip)
- 6.) ls
- 7.) mv terraform /usr/local/bin
- 8.) terraform
- 9.) Create a tf file → touch main.tf
- 10.) Edit file → nano main.tf  
We need to write something in file for that :-
  - i.) Search ~~terraform~~ on aws in google → registry.terraform  
Select usage well past copy paste.
- 11.) Change region to aws (us-east-1)

Now for access key & secret key :-

- Steps :- We have to create instance using IAM user
- Goto IAM → Users → add user
  - Give username → Tick on access key → next
  - Attach existing policies → AdminstratorAccess → next
  - Add tags (skip) → Create user
  - Now we get access and secret key copy it in the file.

Now second part of script

Now type,