Advanced DevOps Practical

1. Launch an EC2 instance :

Login into Aws -> Search EC2 -> Create without key pairs -> Launch Instance -> Connect to the instance -> Run basic Linux commands -> Go to Ec2 dashboard -> Right Click and Stop instance -> Right Click and terminate instance -> Also delete one security group -> You're done!!

2. Setup AWS Cloud9 IDE :

Search Cloud9 -> Open Cloud9 -> Create Environment -> Enter Name -> Create -> Make a new file -> Ctrl+S and name it -> Run it -> Close tab -> Click on the running Environment -> Delete -> You are done!!!

3. Create S3 bucket and upload a file on it :

Search S3 bucket -> Create bucket -> Name it -> Create -> Click on bucket name -> Create a factorial.py file on desktop -> Click on bucket name and select upload -> Drag and drop your file -> Upload succeeded then close ->Click on bucket name -> select the file -> download -> Now select the bucket again -> Click on Empty -> then select it again and click on delete -> Done!!

4. Install Terraform :

Search on google terraform download -> Click on first website(HashiCorp) -> Go to windows and download 386 -> As it is downloading make a new folder named Advdevops inside the C drive-> after the zip file is downloaded extract it into the new

folder that we just created, else just extract normally and then move it there -> Go to system properties -> Environment variables -> system variables -> path -> new and paste the path of the terraform application which was inside c drive -> Click ok and exit ->Open command prompt -> type cd .. and enter until the path is only C:\ -> cd Advdevops (the folder path where the terraform app lies) -> terraform -> it will give you a list of information -> You are done!!

5. Use terraform and create an instance :

Search IAM -> On the left sidebar go to Users -> Enter name -> next -> Create a group -> Tick the AdminstratorAccess -> Give a name to this group ->Next -> Create User -> Click on the user -> click on Create acesss key -> select CLI -> download csv -> Search EC2 -> Launch Instance -> Select Ubuntu OS -> Copy the ami id -> Launch instance -> stop instance -> make a .tf file ->

```
provider "aws"{
   access_key = "AKIAUGOLJUSRVCOPGNOA"
   secret_key                                                    =
"mqOqU1fDklmCTRtEwX9c3SHQhcKvh0y4IcoSkp8u"
   region = "ap-south-1"
}

resource "aws_instance" "Ubuntu"{
   ami = "ami-0287a05f0ef0e9d9a"
   instance_type = "t2.micro"
}
```

-> open cmd and go to the folder where this file is saved and type terraform init, plan, apply, destroy -> Done!!

6. Assignment 6 :

Install SonarQube Community Version -> Extract zip -> add bin to environment variables System -> cd sonarqube\bin\windows-x86-64 -> StartSonar.bat -> Download sonar scanner -> Extract -> \bin copy and cd in new cmd -> sonar-scanner -> Go to localhost:9000 -> user and pass both admin -> Create manually -> locally -> generate -> Continue -> Others -> windows -> add path -> make a folder and make python program -> open cmd there -> copy paste -> done

7. Lambda for s3 bucket :

Create a s3 bucket -> go to Iam dashboard -> Policies -> Create a policy -> Json : {

```
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "logs:PutLogEvents",
            "logs:CreateLogGroup",
            "logs:CreateLogStream"
        ],
        "Resource": [
            "arn:aws:logs:*:*:*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject"
        ],
        "Resource": "arn:aws:s3:::*/*"
    }
]
}
```

-> name the policy -> Go to roles -> Create a role -> select the policy you just created -> upload something to the s3 bucket lets say an

image -> Search lambda and create a function -> use blueprint of hello world python -> Create function -> click on test -> create event -> replace value of key1 to hello world -> Save and run -> Done!!

8. Content-type in s3 bucket :

Make an s3 bucket -> Create new Policy from Iam -> Create new role -> select policy -> upload image in s3 bucket ->

9. S3 bucket Terraform :

exp 5 -> paste this code in .tf file -> provider "aws"{

```
access_key = "AKIAUGOLJUSR6KEXZK5W"
secret_key = "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"
region = "ap-south-1"
}

resource "aws_s3_bucket" "mybucket"{
bucket = "kedar64"
}
```

init, plan, apply, destroy

10. Pipeline

create s3 bucket -> Acl enabled -> Object writer -> Uncheck block all public acess -> Create bucket -> Search codepipeline -> Create -> New service role -> anyrolename -> Source provider github version 1 -> Connect to github -> repository -> branch master -> Change detection pipeline -> Github webhooks -> Aws codebuild -> Create project -> managed Image -> os Ubuntu -> Image- aws code standard 7.0 -> New role -> Use a buildspec file -> Cloudwatch logs untick -? Continue to codepipeline -> Single build build type -> Deploy provider amamzon s3 -> object key -> Extract file -> Deploy click amazon s3 -> permissions -> bucket policy -> paste -> Edit acl -> Tick all -> Enable static web hosting in properties -> index doc -> index.html ->

11. Dynamo db :

Search dynamo -> Create table -> partition key is primary key ->kedar id N and kedar name S-> Create table ->Explore table itrems -> Create Item _> Search lambda -> create function -> use blueprint -> dynamo python3.10 -> role -> policy -> Iam -> role -> use case lambda -> add permission full acess dynamo -> remove api trigger -> copy paste code -> import boto3

```
import json

import boto3


dynamodb_client = boto3.resource('dynamodb')

table = dynamodb_client.Table('kedar-table')


def lambda_handler(event, context):

    try:

        response = table.put_item(Item=event)

        return table.scan()

    except Exception as e:

        raise e -> config test event -> delete json and put db items in json format -> Deploy
```

json -> {

```
    "id": 101,

    "name": "Kedar"
```

}

12. Image upload :

create bucket -> create lambda function -> blueprint -> s3 get -> create a role -> s3 read only permission -> buxcket name in trigger -> event all -> create function -> testevent -> s3put -> upload image -> cloudwatch logs