***ALL TASKS***

EFS TASKS

\*\*\*\*\*\*\*\*

1. Setup and configure EFS

2. Launch EC2 instance and mount EFS using the below commands

yum install -y nfs-utils

mkdir efs

mount -t nfs4 xxxxxxx.efs.ap-south-1.amazonaws.com:/ efs/

cd efs

3. Create some sample files using touch command and create some dirs

4. Launch another EC2 instance and mount EFS using the above commands

5. verify the data between the EC2 instances.

6. Terminate EC2 instances and delete EFS

\*\*\*\*\*\*\*\*\*\*\*\*\*

RDS TASKS

\*\*\*\*\*\*\*\*\*\*\*\*

1. Launch PostGres or Mysql DB Instance in free tier

2. Modify Security Group: add postgres and mysql ports

3. Download PgAdmin or Sqlectron

4. Connect to the RDS Db Instance using endpoint, username and Pwd(try to create a small table or a database)

5. Take a manual snapshot, copy snapshot, try encrypted snapshot

6. Create a Read Replica in another ireland(modify Ireland SG(add port numbers), connect to it and try to create a DB or table)

7. Restore a DB from the snapshot

Check the logs in cloudwatch logs

8. Delete the RDS instances in Mumbai and RR in ireland

9. Delete the manual snapshot.

10. Promote RR to standalone

Sample application with a registration form and update details to RDS Database\*\*\*\*\*\*\*\*\*

DynamoDB

\*\*\*\*\*\*\*\*

1. Create a table

2. Add items to the tables

3. Work on Primary key and Sort Key Concepts

4. Delete the tables

5. If you are intrested, load the data to the table using programing like Python

8. Simulate CORS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Create 2 buckets name bkt1 and bkt2 --> public

upload index.html and load.html on bkt1 --> enable website hosting

Access the bkt1 website endpoint, load.html will load because load.html is in same bucket

upload load.html only on bkt2, make it public --> enable website hosting

take the load.html url/Endpoint from bkt2

update the URL on index.html on bkt1 (load load.html from bkt2 on bkt1)

Access the bkt1 website endpoint, load.html will NOT load, because load.html is in another bucket

enable cors on bkt2

Now, access the bkt1 index.html, load.html will work now, because CORS is enabled

https://docs.aws.amazon.com/AmazonS3/latest/dev/cors.html ---> get json from here

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

9. Setup Life cycle rules

11. CRR implementation

bkt1 --> bkt2

bkt2 ---> bkt3

bkt1 --> bkt3

12. SRR implementation

13. From your account --> Create 2 IAM users called user1 and user2

Create 2 buckets in S3, user1bkt and user2bkt

either create IAM policy or bucket policy

logout from your account

login as user1 and access user1bkt (should be access) , access user2bkt(should not access)

logout user1

login as user2 and access user2bkt (should be access) , access user1bkt(should not access)

14. Download S3 browser --> Add account --> Provide KEYS --> Start accessing S3 or put the data in buckets

S3: Simple Storage Service

\*\*\*\*\*\*\*\*\*

TASKS

1. Create private bucket and upload the objects

2. Create a public bucket and upload the object and access the object using endpoint

--> Make sure you make the object public "Make Public"

3. Create a bucket (public or private) with versioning enabled

upload the objects and check versioning

4. Suspend the versioning and perform few tasks on the objects

upload the objects and check versioning

5. You should receive the notification if anyone upload a object inside the bucket (you should use S3 EVENT)

6. Enable Server access logs on S3 Bucket

7. Setup static website hosting on S3.

--> create a public bucket

--> upload all your images, html files

--> select all objects and make public

--> go to bucket properties --> enable static webite hosting --> index.html and error.html

8. Delete all buckets

\*\*\*\*

BeanStalk

\*\*\*\*

--> Create Application with single instance preset

--> Create beanstalk env with tomcat --> Access the website using Env-URL

--> Create beanstalk env with Python--> Access the website using Env-URL

--> GO to application level options Swap the URL and then access the env-URL (Blue Green Deployment)

--> Create a new Application and environment with HA and check in EC2 console for ASG, ELB etc

--> Clone env, restart env, rebuild env ---> See what is happening

--> do Blue green deployment on HA

--> Terminate the environment and restore it

--> Download the logs and Go through it once

--> Delete the Application

Upload you own application and redeploy with latest application

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

CloudWatch

\*\*\*\*\*\*\*\*

--> Stop or Terminate the EC2 instance by creating Alarm (if the CPU is less than 10%)

--> Create a sample Dashboard and add widgets

--> Create Alarm from CloudWatch Console

--> Create a billing alarm

--> Create an Event and get the Email notification if any one launch the EC2 instance

--> Try Schedular in CloudWatch with Lambda

--> Setup CloudWatch Logs on Linux

--> Clean up Everything(Terminate EC2, delete dashboard,events,Logs)

--> Setup CloudWatch Logs on Windows(explore)

\*\*\*\*\*\*\*\*\*

Auto-Scaling

\*\*\*\*\*\*\*\*\*

1. Create ELB with Empty Target Group

1.1: Keep HTTP in your SG, and make sure you have default rule ALL TRAFFIC with self

2. Create Launch Configurations(select AMI ID, provide user data script)

3. Create ASG from the created Launch configuration (min, DC, max)

3.1: Check your targets in target groups, it should be healthy

4. Access your ELB DNS Name.

--> try to do manual scaling

--> try to do schedule scaling

--> try to do instance refresh

--> try to terminate ec2 instances manually and see what happens

5. Delete ASG

6. Delete launch Config

7. Delete ELB and TG

Load Balancer:

\*\*\*\*\*

1. Launch 2 Ec2 instance with the user data(server1 userdata put it as "My first webiste on server1 and vice versa on server2")

2. Modify SG by adding HTTP protocal with myip and dont modify default rule (all traffic)

3. Access the 2 EC2 instances Public IP's on the browser and verify the website is working or not

4. Create Load Balancer (in health check ensure to have /index.html) --> create target group and register targets in the TG

5. Access ELB DNS Name.

6. Check the targets in target groups, it should show healthy

7. Explore path based routing (create 4 instances, 2 with /index(TG1) and 2 with /admin(TG2) and add rules in target grp)

7. Terminate EC2 instance, ELB and target groups.

TASKS

\*\*\*\*

1. CROSS TASK

2. Increase the root volume size

3. Add addtional volume and make it available to the user.

4. Play with attach and detach volume for Root and Addtional volumes

5. Extend the existing volume on Linux

6. New addtional volume for Linux

7. Life Cycle Manager policy

8. Terminate EC2 instances

9. Delete addtional volumes

\*\*\*\*\*\*\*\*

IMAGES TASKS

\*\*\*\*\*\*\*\*

Launch a windows instance -->login to that instance --> create some files or install a software --> Create image --> launch instance from image --> login to the newly created instance(password?) --> Check the files or software are availble

Copy the image to another region --> once verified, delete the image in ireland region

Encrypt the image --> use copy option --> check encryption option --> create image

Copy the encrypted image to same region and another region (?)

Copy the Snapshot to another region --> once verified ,delete the snapshot

Try share the image and snapshot to another account

Create a image from the snapshot

First try deleting the snapshot and then AMI --> see if this works

First try deleting the AMI and then snapshot--> see if this works

Terminate EC2 instances, Delete images, Delete snapshots

\*\*\*\*\*\*\*\*\*\*

EC2

\*\*\*\*\*\*\*\*\*\*\*

TASKS:

1. Launch Windows 2019 EC2 Instance(step 1 to step7(create a key pair and save it safe))

Add a new rule RDP protocal to the Security Group (dont do any changes to the default rule)

use remote desktop protocal to connect to windows machine(username: administrator, pwd: key-pair)

2. Launch Linux redhat EC2 Instance and try to connect it

Download Putty

Download PuttyGenerator

convert PEM to PPK using PuttyGen

--> Open PuttyGen

--> Load your Pem file (Load --> Choose Pem file --> save private key as xxxx.ppk)

Add a new rule SSH protocal to the Security Group(dont do any changes to the default rule)

Use Putty to connect to Linux machine(hostname: publicIP --> expand SSH --> select Auth --> browse ppk file)

3. Try to stop and start EC2 instance --> public IP changes

4. Assign EIP to the machine and try to stop and start --> EIP will not change

5. Disssociate EIP from the Instance and release it to AWS.

**\*\*IAM user creation\*\***

Login to the ROOT account --> Navigate to IAM service

1. Customize IAM user sign-in Link and bookmark it.

2. Create IAM user with admin policy attached

3. Setup MFA for the IAM user that you have created(yourownname)

4. Setup MFA for Root account

\*\* Signout from root account and login as IAM user that you have just created. From now onwards

always login with IAM user

5. Create another sample IAM user(test)

6. Attach and detach policies for sample IAM user test(not your main IAM user)

7. Create 2 groups and attach users to the groups.

8. Try to disable console to the test IAM user and try to login with that disabled IAM user

9. Reset the IAM user password and try to login with the reseted password

10. Create Access key and Secret Keys for the test IAM user

11. Make the KEys inactive.

12. Delete the KEYS.