

Q3,Q4

Create iam user, give custom password, then **create group, give user and group cloud9environment permission. -> attach policy directly -> cloud9environmentmember.**
-> create user.

Open the console log-in link in incognito tab.

Go to cloud9 -> main browser tab -> save a random file. Share -> invite user u made -> now open environment link in incognito tab.

Collab in chat box

Q5, Q6, Q22

S3 bucket -> iam role -> elastic beanstalk -> create pipeline

S3 -> create Bucket -> give name and create bucket -> **create iam role** -> create role and use case ec2 -> next -> give permissions -> AWS elasticbeanstalk webtier -> role name -> create -> **Elastic beanstalk** -> create application -> give name -> choose platform PHP -> use existing role -> choose your role -> key pair (NO CHANGES) -> ec2 instance profile (choose your role) -> Skip to review (skip step 3, 4, 5) -> submit -> AFTER ENVIRONMENT IS LAUNCHED -> **create code pipeline** -> step1 - only give pipeline name -> step2 - add source stage (Github version2) -> connect to github -> authorize -> install a new app -> wait for installing -> connect -> give github repo link -> branch name -> step3 - skip build stage -> step4 deploy stage -> aws elastic beanstalk -> app name and env name -> step5 - create pipeline
Check output -> pipeline deploy -> link -> elastic beanstalk -> domain link -> see output
(If any changes in output. Go to repo and change in index.html)

Q7

Download terraform -> set path -> create iam user -> custom password -> attach policy directly -> administrator access -> next -> create user -> go to user -> click access key (blue color) -> use case (cli) -> give description (any) -> create access key -> same folder create file (.tf extension) -> copy paste code -> change the access key and secret key in code -> open cmd in terraform folder -> "terraform init" -> "terraform plan" -> "terraform apply" -> Check output on ec2 instances -> "terraform destroy"

Q8,9,10

Set path for sonarqube and sonar path "bin->windows->lib" is the path to be given in env variable

Sonarscanner path is given till bin, path to be given in environment var and "PATH" -> new

Go to c->sonarqube->bin->windows->start sonar

Local9000

Create new project -> give name, key is generated and generate token (command is generated with token)

Now go to sonarscanner -> conf -> sonarscanner -> open and give sonar.projectKey=TypeScript

sonar.projectName=TypeScript sonar.projectVersion=1.0

sonar.sources=C:\sonar-scanner-5.0.1.3006-windows\conf

Open sonarscanner -> conf (where the code is) open terminal and paste token command

View op on dashboard

Q11,12,13

-> Search Lambda -> Create Function -> Use a blueprint -> (Blueprint name) Hello world Python 3.10 -> give function name -> create function

Q14, Q15, Q16

(Execution role) -> Create a new role from AWS policy templates (s3 full access, cloudwatch full access, lambda basic execution role) -> role name -> Create bucket -> create function -> author from scratch -> function name -> select python 3.7 -> change default execution role -> use an existing role -> choose your role -> create -> code copy paste -> deploy -> configure test event -> give name -> save and test -> after success go to s3 bucket to check the object added -> download the file and check output in vs code.

Test -> Configure test event -> Event name -> Edit event JSON

```
import json
import boto3
s3=boto3.client('s3')
def lambda_handler(event,context):
    bucket="q14bucket"
    dataToUpload = {}
    dataToUpload['PID'] = '211121'
    dataToUpload['DEPT'] = 'INFT'
    dataToUpload['NAME'] = 'Brijraaj'
    dataToUpload['FILE'] = 'brij'
    fileName = 'brij' + '.json'
    uploadByteStream= bytes(json.dumps(dataToUpload).encode('UTF-8'))
    s3.put_object(Bucket=bucket,Key=fileName,Body=uploadByteStream)
    print('an object has been added')
```

Search IAM -> Roles -> Create role -> (Usecase) Lambda -> Next

Permissions policies

[CloudWatchFullAccess](#)

[AWSLambdaBasicExecutionRole](#)

[AmazonS3FullAccess](#)

Step 1: Select trusted entities Edit

Trust policy

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": "sts:AssumeRole",
7       "Resource": [
8         "arn:aws:iam::111111111111:role/*"
9       ],
10      "Principal": {
11        "Service": "lambda.amazonaws.com"
12      }
13    ]
14  }
15 }

```

Step 2: Add permissions Edit

Permissions policy summary

Policy name	Type	Attached as
AmazonS3FullAccess	AWS managed	Permissions policy
AWSLambdaExecuteRolePolicy	AWS managed	Permissions policy
CloudWatchFullAccess	AWS managed	Permissions policy

->Enter Role Name -> Create Role

->Search S3 -> Create Bucket -> Enter Bucket name -> Create

->Search Lambda -> Create function -> Enter name -> Change default execution role -> Use an existing role -> role1->Create function

Click on Deploy -> click test -> Invoke

Execution results Status: Success

Test Event Name
(Unsaved) test event

Response
null

Function Logs

```

START RequestId: e85c7685-28da-4d74-aaab-b437bc0fd9a9 Version: $LATEST
Object has been uploaded
END RequestId: e85c7685-28da-4d74-aaab-b437bc0fd9a9
REPORT RequestId: e85c7685-28da-4d74-aaab-b437bc0fd9a9 Duration: 239.96 ms Billed Duration: 240 ms Memory Size: 128 MB Max Memory Used: 78 MB Init Duration: 605.14 ms

```

Request ID
e85c7685-28da-4d74-aaab-b437bc0fd9a9

Search S3 ->

Amazon S3 > Buckets > hoelygoatbucket

hoelygoatbucket [info](#)

Objects | Properties | Permissions | Metrics | Management | Access Points

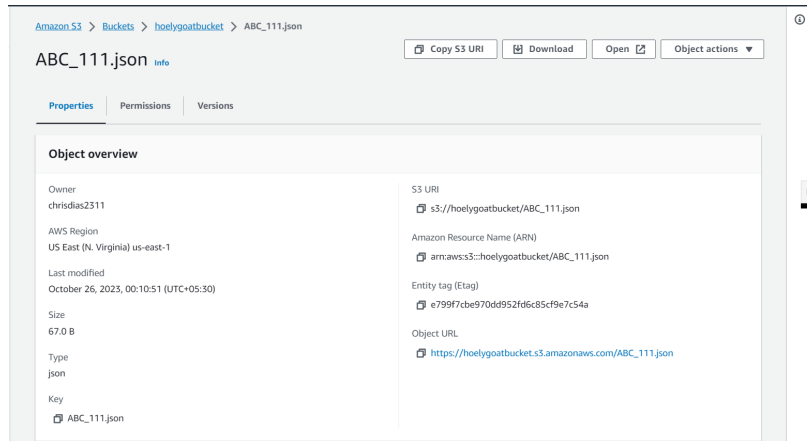
Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	ABC_111.json	json	October 26, 2023, 00:10:51 (UTC+05:30)	67.0 B	Standard



Q15

Search Lambda -> Create Function -> Use a blueprint -> (Blueprint name) Hello world Python 3.7

(Execution role) -> Create a new role from AWS policy templates -> role name -> Create function

Test -> Configure test event -> Event name -> Edit event JSON

Use invoke instead of run/save. Next go to monitor for visualization

Q16

Search Lambda -> Create Function -> Use a blueprint -> (Blueprint name) Hello world Python 3.7

(Execution role) -> Create a new role from AWS policy templates -> role name -> Create function

Test -> Configure test event -> Event name -> Edit event JSON

Index.mjs is the automatic code, change it, invoke it and test.op will be generated in logs

Q17

Create instance -> give name -> ubuntu -> t2 micro -> key pair -> create key pair -> key name -> .ppk -> create -> create -> connect the made instance -> ubuntu terminal will be opened

Display present working directory in cloudshell- pwd

Q18

Q19

Easy

Q20

Same as 14

Q21

Same as q3

Q22