



Name/id: abdelkareem yousef mamdoh soubar/19110022

Assignment Title: Cloud Computing AWS

Course Tutor: Eng. Moath Malkawi

Submission Date: 26 – Jan - 2022

## Task 1

video : <https://drive.google.com/drive/folders/1Uyd-Lp6cTwdQBNAJczTTTfyrJJMe6vkn?usp=sharing>

part11

in general:

AWS automobile Scaling ceaselessly analyzes your applications and changes capability pro re nata to produce consistent, sure performance at rock bottom possible value. It's straightforward to make up application measurability for varied resources across several services in minutes exploitation AWS automobile Scaling. Your applications can continually have the right resources at the correct time due to AWS automobile Scaling.

in this task the policy for the load blancer I use is that if a cpu uses more than 60% a new instance would show up and have traffic come to it.

An Application Load Balancer routes requests to 1 or additional ports on every instrumentality instance in your cluster, creating routing selections at the applying layer (HTTP/HTTPS). It conjointly supports path-based routing and will route requests to 1 or additional ports on every instrumentality instance in your cluster. Dynamic host port mapping is supported by Application Load Balancers.

In our case:

It also helps in case of failure or false health check that the it would keep the traffic a way from and redirect it to the working server.

As for auto scaling it is place in our case is if any of the devices need extra storage it would provide it and it keeps it eyes on the application in case it needs any more/less and it tries its best to operat under the lowest cost possible .

everything had been placed in side the video but the AWS well architected framework pillars amazon had placed methods ro insure the well architect when it comes to using their cloud.

Pillars are:

1. Operational Excellence
  - Perform operations as code  
we only used bash for code and we did setup sync through it, but we didn't change any of the settings on the cloud.
  - Make frequent, small, reversible changes
  - Refine operations procedures frequently
  - Anticipate failure  
we are using health check if it failed it would direct traffic to other servers
  - Learn from all operational failures
2. Security
  - Implement a strong identity foundation  
we were not allowed to create any IAM roles
  - Enable traceability
  - Apply security at all layers  
we did apply security layers by using vpc /security groups /subnets private and public
  - Automate security best practices
  - Protect data in transit and at rest  
the data in connection is encrypted

- Keep people away from data  
we are using cloudfront that hides the internal connections
  - Prepare for security events  
we did place security groups that control the traffic (I only receive from my ips)
3. Reliability
- Automatically recover from failure  
we have a load balancer in case any failure happens it lunches a new instance plus everything is placed on s3 code bucket that we made.
  - Test recovery procedures
  - Scale horizontally to increase aggregate workload availability  
when more work load occurs the load balances would launch more instances
  - Stop guessing capacity  
most of what we have auto scaling enables so if we need more we will get more and pay for it.
  - Manage change in automation  
we don't have any code related to the service it self but we did do sync in bash.
4. Performance Efficiency
- Democratize advanced technologies
  - Go global in minutes  
yes we are using edge locations through cloudfront
  - Use serverless architectures
  - Experiment more often  
we have a heath check running all the time plus we have placed tones of effort in test our service before the final launch
  - Consider mechanical sympathy
5. Cost Optimization
- Implement cloud financial management  
we tried to use everything that was free for this task
  - Adopt a consumption model
  - Measure overall efficiency  
its externally efficient and it uses about 1.5\$ per day with task4
  - Stop spending money on undifferentiated heavy lifting  
it would use more money in case of having traffic because it opens more instances
  - Analyze and attribute expenditure  
because this is a task there would be no traffic but in real life the first one million requests are free.

terminating 2 instances so the would load balancer would open 2 new one's with latest update on them.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various navigation options like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances, Images, Elastic Block Store, Network & Security, and more. The main area displays a table of instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4 IP, and Elastic IP. There are five entries in the table:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
master	I-00299eedc224f5b35	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-5-84-87-239.com...	3.84.87.239	-
-	I-06b0ef6af6486e6fd	Terminated	t2.micro	-	No alarms	us-east-1b	-	-	-
-	I-0e906e5e6601b954a	Terminated	t2.micro	-	No alarms	us-east-1b	-	-	-
-	I-048ec39e3df480e24	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-44-192-39-21.com...	44.192.39.21	-
-	I-07060ea9cd5b5d288	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-44-200-70-196.co...	44.200.70.196	-

Finished creating new template (sources) while testing

The screenshot shows the AWS EC2 Launch Templates page. At the top, it says "EC2 > Launch templates > Modify template (Create new version)". A success message box states: "Success Successfully modified web-launch-template (lt-0397ac7d87e4c203e). A new version (version 6) has been created." Below this, there's a "Next steps" section with three main items: "Launch an instance", "Create an Auto Scaling group from your template", and "Create Spot Fleet". Each item has a brief description and a "Create" link. At the bottom right of the main content area is a red "View launch templates" button.

here we are testing master and slaves

```
ec2-user@ip-10-0-0-59:~
```

```
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@54.196.179.38 -l labsuser.pem
```
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@3.84.237.239 -l labsuser.pem
```
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@34.207.231.249 -l labsuser.pem
```
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@34.207.231.249 -l labsuser.pem
```
ssh: connect to host 34.207.231.249 port 22: Connection timed out
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@34.207.231.249 -l labsuser.pem
```
ssh: connect to host 34.207.231.249 port 22: Connection timed out
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@34.207.231.249 -l labsuser.pem
```
ssh: connect to host 34.207.231.249 port 22: Connection timed out
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@34.207.231.249 -l labsuser.pem
```
The authenticity of host '34.207.231.249' (34.207.231.249) can't be established.
ED25519 key fingerprint is SHA256:nGA/s0dzeGZKNUj2Q1xJhdHzAVAgSKzGyB8VC.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '34.207.231.249' (ED25519) to the list of known hosts.
Last login: Sat Jan 29 07:13:28 2022 from 188.123.181.43
[|] ( | ) Amazon Linux 2 AMI
[|] (\_|_ |)
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-0-59 ~]$ sudo yum --enablerepo=remi,remi-php72 install php-od
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/cert 211x12
The authenticity of host '35.173.49.184 (35.173.49.184)' can't be established.
ED25519 key fingerprint is SHA256:VWR3JE/xy/5Cn+0qJZKT/hVQFT1Y9WMHTB0dKrrI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '35.173.49.184' (ED25519) to the list of known hosts.
ec2-user@35.173.49.184:~$ curl -s https://keybase.io/~mrk -k -kex_gssapt-with-mic.
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@35.173.49.184 -l labsuser.pem
```
ssh: connect to host 35.173.49.184 port 22: Connection timed out
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@35.173.49.184 -l labsuser.pem
```
ssh: connect to host 35.173.49.184 port 22: Connection timed out
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@35.173.49.184 -l labsuser.pem
```
ssh: connect to host 35.173.49.184 port 22: Connection timed out
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@35.173.49.184 -l labsuser.pem
```
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@35.173.49.184 -l labsuser.pem
[|] ( | ) mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/cert 211x13
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/cert 3
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@44.192.39.21 -l labsuser.pem
```
ssh: connect to host 44.192.39.21 port 22: Connection timed out
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@44.192.39.21 -l labsuser.pem
```
ssh: connect to host 44.192.39.21 port 22: Connection timed out
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@44.192.39.21 -l labsuser.pem
```
ssh: connect to host 44.192.39.21 port 22: Connection timed out
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@44.192.39.21 -l labsuser.pem
```
ssh: connect to host 44.192.39.21 port 22: Connection timed out
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs cd /
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@35.173.49.184 -l labsuser.pem
```
ssh: connect to host 35.173.49.184 port 22: Connection timed out
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ssh ec2-user@35.173.49.184 -l labsuser.pem
```
mrk@mrk-Lenovo-Legion-Y530-15ICH-1060:~/Downloads/certs ]
```

## database creation

Amazon RDS

**database-1-1**

**Summary**

| DB Identifier | CPU              | Status    | Class       |
|---------------|------------------|-----------|-------------|
| database-1-1  | 5.74%            | Available | db.t2.micro |
| Role          | Current activity | Engine    | Region & AZ |
| Instance      | 1 Connections    | MariaDB   | us-east-1a  |

**Connectivity & security**

| Endpoint & port                                                   | Networking                                                                                                             | Security                                                        |
|-------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| Endpoint<br>database-1-1.cdtm1zvdxttc.us-east-1.rds.amazonaws.com | Availability Zone<br>us-east-1a                                                                                        | VPC security groups<br>db sg (sg-0330567b844372063)<br>Active   |
| Port<br>3306                                                      | VPC<br>CT6AWS (vpc-c-011d04c6ae49b91ad)                                                                                | Public accessibility<br>No                                      |
|                                                                   | Subnet group<br>default-vpc-011d04c6ae49b91ad                                                                          | Certificate authority<br>rds-ca-2019                            |
|                                                                   | Subnets<br>subnet-0bb8c12237d346186<br>subnet-0954917ea3223182<br>subnet-06485c017e46d68673<br>subnet-06b28a090eb3f9b5 | Certificate authority date<br>August 22, 2024, 08:08 (UTC±0:08) |

**Security group rules [2]**

Feedback English (US) ▾

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

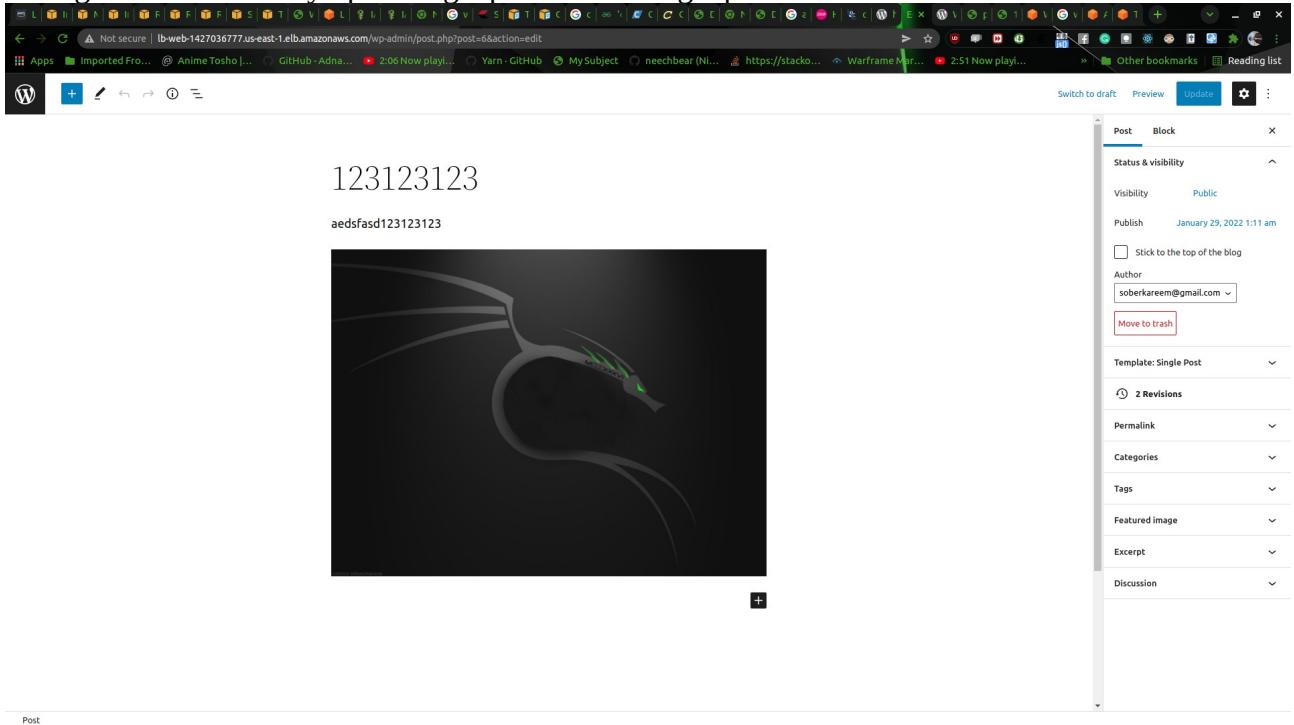
finishing off the creation of the replica

The screenshot shows the AWS RDS (Amazon Relational Database Service) console. On the left, a sidebar lists various RDS management options like Dashboard, Databases, Query Editor, and Snapshots. The main content area is titled "Databases" and shows a table of existing databases. One row is highlighted: "database-1-1" (Primary, MariaDB, us-east-1a, db.t2.micro, Modifying, 5.50% CPU, 1 Connection, none, vpc-011d). Another row, "database-1-2" (Replica, MariaDB, us-east-1b, db.t2.micro, Available), is also listed. A success message at the top states "Successfully created replica database1-2 In US East (N. Virginia)". At the bottom right of the table, there's a "Create database" button.

## subnets

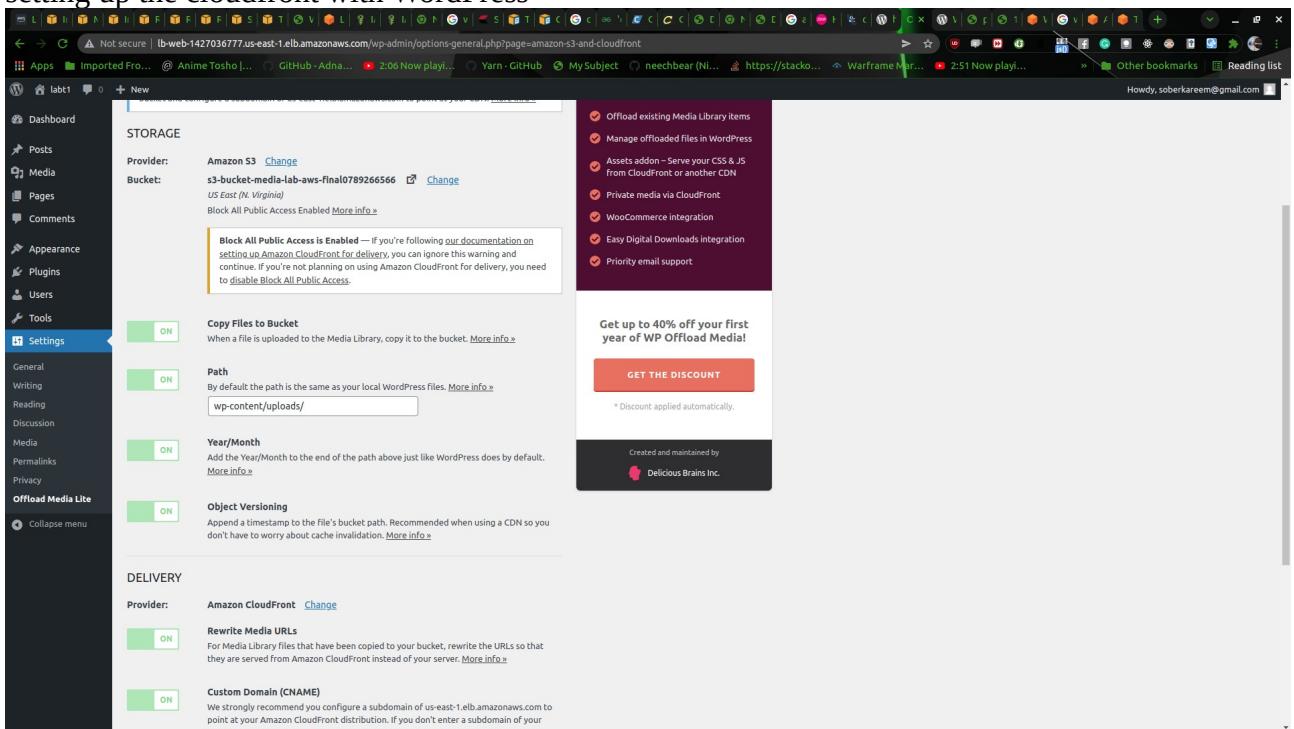
The screenshot shows the AWS VPC (Virtual Private Cloud) console. On the left, a sidebar lists VPC management options such as New VPC Experience, VPC Dashboard, EC2 Global View, Filter by VPC, and Select a VPC. The main content area is titled "Subnets (10)" and shows a table of subnet configurations. A success message at the top says "You have successfully changed subnet settings: Enable auto-assign public IPv4 address". The table includes columns for Name, Subnet ID, State, VPC, IPv4 CIDR, IPv6 CIDR, and Available IPv4 addresses. Several subnets are listed, including "Public subnet a" and "private subnet b". At the bottom, there's a "Select a subnet" section and a "Create subnet" button.

testing the WordPress by uploading a pic and creating a post



Post

setting up the cloudfront with WordPress



Not secure | lb-web-1427036777.us-east-1.elb.amazonaws.com/wp-admin/options-general.php?page=amazon-s3-and-cloudfront

Apps Imported From... GitHub - Adna... 2:06 Now play... Yarn - GitHub My Subject neechbear (N... https://stacko... Warframe Mar... 2:51 Now play... Other bookmarks Reading list

Howdy, soberkareem@gmail.com

Dashboard Posts Media Pages Comments Appearance Plugins Users Tools Settings General Writing Reading Discussion Media Permalinks Privacy Offload Media Lite Collapse menu

By default, the path is the same as your local WordPress files. [More info »](#)

**Year/Month** Add the Year/Month to the end of the path above just like WordPress does by default. [More info »](#)

**Object Versioning** Append a timestamp to the file's bucket path. Recommended when using a CDN so you don't have to worry about cache invalidation. [More info »](#)

**DELIVERY**

**Provider:** Amazon CloudFront [Change](#)

**Rewrite Media URLs** For Media Library files that have been copied to your bucket, rewrite the URLs so that they are served from Amazon CloudFront instead of your server. [More info »](#)

**Custom Domain (CNAME)** We strongly recommend you configure a subdomain of us-east-1.elb.amazonaws.com to point at your Amazon CloudFront distribution. If you don't enter a subdomain of your site's domain in the field below it will negatively impact your site's SEO. [More info »](#)

dn4y8q9t1jt6.cloudfront.net

**Force HTTPS** By default we use HTTPS when the request is HTTPS and regular HTTP when the request is HTTP, but you may want to force the use of HTTPS always, regardless of the request. [More info »](#)

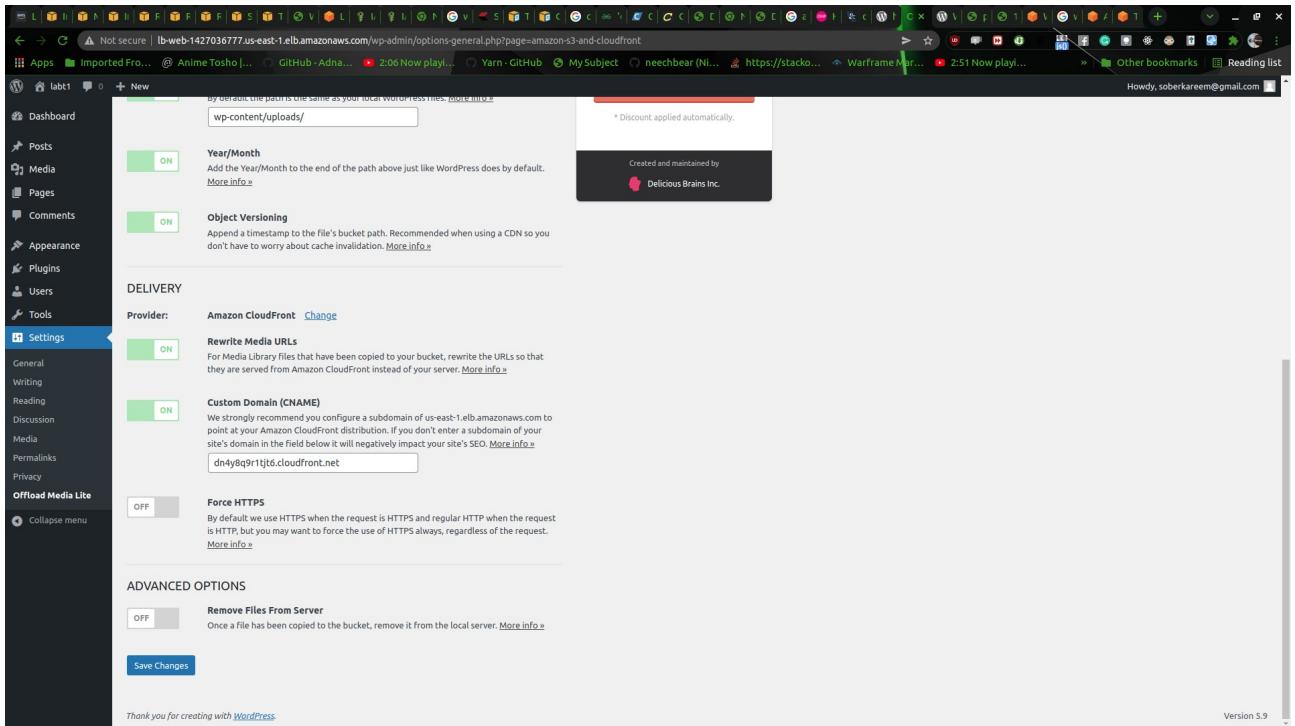
**ADVANCED OPTIONS**

**Remove Files From Server** Once a file has been copied to the bucket, remove it from the local server. [More info »](#)

**Save Changes**

Thank you for creating with [WordPress](#).

Version 5.9



## Task 2

### EC2

| Category                     | Designed for                                                                                                                                                                                                                              |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A1 (Amazon EC2 A1 instances) | It was made for saving cost while scaling out also for Arm based applications such as a web server and in extend to that it also can distribute data stores which are supported by extensive Arm ecosystem.                               |
| M4 instances                 | It's designed for getting a great balance between memory , compute and network resources, and it suites workloads like web server, database...                                                                                            |
| T3 instances                 | T3 instances has a base line for CPU usage that has the ability to burst at any time, it was made for applications that need CPU usage and can get spikes from time to time.                                                              |
| T4g instances                | It was made for running general purpose workloads with low cost in mind, this instance comes with a base line for CPU performance to do common workload but it can go above the baseline when for time when more performance is required. |
| Mac instances                | It gives a stable environment high performance and secured for developers with workload, and it makes the integration with AWS easy for using AWS Lib's and tools plus luching configurations.                                            |

### EC2

| Purpose              | Recommended EC2 Instance type category | Why?                                                   |
|----------------------|----------------------------------------|--------------------------------------------------------|
| Hosting a web server | M5                                     | balances                                               |
| Multiplayer gaming   | c5n                                    | High-performance and low latency                       |
| Data analytics       | R5                                     | It provides high memory plus increased CPU performance |
| Speech recognition   | p3                                     | It can do hardware acceleration with GPU               |

### S3 storage

| Storage Class:         | Designed for:                                                                                                                                     |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| S3 Standard            | Its a storage the provides a low latency, high-performance and very reliable storage its used for web hosting and content distribution and more . |
| S3 Intelligent-Tiering | A cost-benefit analysis is a process used by businesses to evaluate the potential rewards and costs associated with a given decision. The         |

|                         |                                                                                                                                                                                                                                               |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                         | <p>analyst sums up the expected returns and then subtracts the total costs.</p> <p>The ability to store and access knowledge in a variety of ways enables organizations to work seamlessly across different platforms and applications.</p>   |
| S3 Standard-IA          | It was not designed for connect too much to it but on the contrary its used for low usage but when its being used it give high performance and low latency and durability , and it suitable for backups or restoring data in case of disaster |
| S3 One Zone-IA          | Its the same as s3 standers IA but its saved in one zone                                                                                                                                                                                      |
| S3 Glacier              | long-term storage of information that's sometimes accessed and that retrieval latency times of three to five hours ar acceptable                                                                                                              |
| S3 Glacier Deep Archive | Customers who maintain data sets for 7-10 years or longer to meet client needs and regulatory compliance requirements, particularly those in the financial services, healthcare, media and entertainment, and public sector.                  |

| Type of Data                           | Recommended Amazon storage class | Why?                                                                                                                                                               |
|----------------------------------------|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mobile game data                       | S3 Standard                      | it provides low retrieval, low latency and high throughput                                                                                                         |
| Disaster recovery data                 | S3 IA                            | It has a higher data retrieval cost but its also half the s3 Standard cost                                                                                         |
| Data that may or may not be used often | S3 Intelligent-Tiering           | automatically lowers your storage prices on a per-object basis by relocating information to the foremost cost-efficient access tier reckoning on access frequency. |
| Media archives                         | Glacier                          | highest performance, the most retrieval flexibility, the lowest cost archive storage and optimized for different access patterns                                   |

### Task 3

Data availability: refers to the timeliness and consistency with that information is also accessed and used. information accessibility is a component of it.

Data persistence:meaning the data outlasts the process that generated it In AWS, information persistence is done by combining compute and storage services.

Data encryption : Data secret writing may be a security mechanism during which knowledge is encoded and solely someone with the proper encryption key could access or decipher it.

Snapshots square measure progressive backups, which suggests that they solely preserve the blocks on the device that have modified since the last pic.

## EBS

Data availability : The Amazon EbS volumes square measure engineered to be extremely accessible, dependable, and long. Amazon EbS volume knowledge is duplicated across several servers in associate convenience Zone at no extra price to you, preventing knowledge loss because of the failure of any single part.

data encryption : By encrypting your knowledge volumes, boot volumes, and snapshots, EBS coding offers knowledge at rest security. Keys that are handled by Amazon or keys that you create and control using the AWS Key Management Service (KMS).

snapshot : Amazon EbS permits you to save lots of snapshots of your volumes to Amazon S3 at any moment. solely the blocks that have modified since your previous photograph area unit unbroken in Amazon EbS Snapshots, and you're solely charged for the changed blocks.

Data persistence: The Amazon compass point has a higher sturdiness volume (io2), which is meant to generate ninety nine.999 percent sturdiness with a zero.001 percent annual failure rate (AFR), where failure refers to a complete or partial loss of the quantity.

| Volume Type:                   | Description                                                                                                                               | Use cases                                                                                                                     |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| General purpose SSD (gp2)      | gp2 (General Purpose SSD) volumes provide cost-effective storage for a wide range of workloads.                                           | virtual desktops, development and test environments, and low-latency interactive apps                                         |
| Provisioned IOPS SSD (io1)     | (io1 and io2) volumes ar meant for I/O-intensive workloads that ar sensitive to storage performance and consistency, like info workloads. | Mission-critical applications, large database such as MongoDB, Microsoft SQL Server, Cassandra, Oracle, MySQL, and PostgreSQL |
| Throughput optimized HDD (st1) | Low-cost HDD capacity for frequently accessed, high-output tasks.                                                                         | Streaming, big data, data warehouses, log processing                                                                          |
| Cold HDD (sc1)                 | For less-frequently accessible tasks, the cheapest HDD volume is used.                                                                    | large volumes of data that is infrequently accessed                                                                           |

## Task 4

GitHub repo works and its been verified

AWS Amplify

wildrydes-site

Learn how to get the most out of Amplify Hosting

First successful build of your wildrydes-site app  
Nailed it! Your app's first build was successful.

Add a custom domain with a free SSL certificate  
You can connect a custom domain purchased through domain registrars (for example, Amazon Route 53, GoDaddy, and Google Domains) to your app.

Set up a test version of your site by connecting a feature branch  
Amplify Hosting leverages Git branches to create new deployments every time a developer connects a new branch in their repository.

Password-protect your site  
Working on unreleased features? Password protect feature branches that are not ready to be accessed publicly.

Enable pull request previews  
Previews offer a way to view changes from pull requests before they are merged to the target branch.

Create a synthetic canary to proactively monitor your app  
You can use CloudWatch Synthetics for 24/7 monitoring of broken links, heartbeat checks, and testing different user flows in your app.

Hosting environments Backend environments

This tab lists all connected branches; select a branch to view build details.

master Continuous deploys set up (Edit)

https://master.dj4sup7qwf2l.amplifyapp.com

Last deployment 1/29/2022, 4:29:36 PM

Provision Build Deploy Verify

Last commit This is an autogenerated message | Auto-build | GitHub - master

Previews Disabled

Feedback English (US) ▾ © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

the tab title had changed



we created pool for the website

The screenshot shows the AWS Cognito User Pools console. A message at the top right says "Your user pool was created successfully." Below it, the Pool ARN is listed as "arn:aws:cognito-idp:us-east-1:410739854364:userpool/us-east-1\_Juhh96ipq". The "General settings" tab is selected. Other tabs include "Attributes", "Policies", "MFA and verifications", "Advanced security", "Message customizations", "Tags", "Devices", "App clients", "Triggers", "Analytics", "App integration", "Federation", "Identity providers", and "Attribute mapping". The "Identity providers" section is currently active. At the bottom, there are links for "Feedback", "English (US)", and copyright information.

login mail verification was successful

The screenshot shows a web browser window with a verification message. The URL is "https://master.dj45up7qwff2i.amplifyapp.com/verify.html". A modal dialog box displays the message: "master.dj45up7qwff2i.amplifyapp.com says Verification successful. You will now be redirected to the login page." with an "OK" button. Below the modal, the word "VERIFY EMAIL" is visible, along with an email address "soberkareem@gmail.com" and a verification code "773940". A large red "VERIFY" button is centered at the bottom. The background of the page has a colorful, radial gradient pattern.

## Creating a database for the website

The screenshot shows the AWS DynamoDB console. A modal window at the top right says: "The new DynamoDB console is now complete, and becomes your default experience. Following the preview phase in which we analyzed and incorporated your feedback, we have completed the new DynamoDB console, making it even easier for you to manage your data and resources. Let us know what you think. You can still choose to return to the previous console from the navigation pane." Below this, another modal says: "Creating the Rides table. It will be available for use shortly." The main table view shows one table named "Rides" with the status "Active". The table has a single item with the key "RideId" and value "Stri...". The table class is listed as "Dynamo...".

## Attaching policies and roles for the website

The screenshot shows the AWS IAM console. A modal window at the top right says: "New feature to generate a policy based on CloudTrail events. AWS uses your CloudTrail events to identify the services and actions used and generate a least privileged policy that you can attach to this role." The main page shows a "Summary" for a role named "LabRole". The role ARN is "arn:aws:iam::410739854364:role/LabRole". The role description is "Edit". The instance profile ARNs are "arn:aws:iam::410739854364:instance-profile/LabInstanceProfile". The path is "/". The creation time is "2022-01-27 22:51 UTC+0200" and the last activity is "2022-01-29 16:46 UTC+0200 (Today)". The maximum session duration is "1 hour". The "Permissions" tab is selected, showing "Permissions policies (8 policies applied)". Under "Attach policies", there are two entries: "AmazonEKSWorkerNodePolicy" and "AmazonEKSClusterPolicy", both listed as "AWS managed policy". There is also a link to "Show 6 more".

## creating test for the website

The screenshot shows the AWS Lambda console interface. At the top, a green banner says "Successfully created the function RequestUnicorn. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below this, the function name "RequestUnicorn" is displayed with a small icon. A "Layers" section shows "(0)". There are buttons for "+ Add trigger" and "+ Add destination". On the right, there's a "Description" section with a link to "arn:aws:lambda:us-east-1:410739854364:function:RequestUnicorn". Below the main area are tabs for "Code", "Test", "Monitor", "Configuration", "Aliases", and "Versions", with "Code" being the active tab. Under "Code source", there's a code editor window titled "index.js" containing the following code:

```
1 exports.handler = async (event) => {
```

## doing a test for the web site

The screenshot shows the AWS Lambda console with the "Test" tab selected. A modal dialog titled "Configure test event" is open. It contains instructions: "A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events." Below this, there are two radio buttons: "Create new test event" (selected) and "Edit saved test events". The "Event template" dropdown is set to "hello-world". The "Event name" field is "TestRequestEvent". The "Environment" section shows "RequestUnicorn" and "index.js". The "Code" section shows the "index.js" file content again:

```
1 exports.handler = async (event) => {
```

## creating and doing a test for arriving the unicorn

The screenshot shows the AWS Lambda console interface. A green notification bar at the top states: "The test event TestRequestEvent was successfully saved." Below this, the "Code source" tab is selected. The "index.js" file contains the following code:

```
function handler(event, context) {
  const response = {
    statusCode: 201,
    body: JSON.stringify({
      "RideId": "nqLQybsPsBebkSaKgMRVUg",
      "Unicorn": {
        "Name": "Rocinante",
        "Color": "Yellow",
        "Gender": "Female",
        "UnicornName": "Rocinante",
        "Eta": "30 seconds"
      }
    })
  };

  context.done(null, response);
}
```

The "Execution results" section shows a successful execution with the status "Succeeded", a maximum memory used of 78 MB, and a duration of 646.94 ms. The "Function Logs" section displays the log output, which includes the received event, the function's execution details, and the final response sent back to the client.

## authorizing the web site to use the API

The screenshot shows the AWS API Gateway console. On the left, a navigation sidebar lists various API management features like APIs, Stages, Authorizers, and Resource Policy. The "Authorizers" section is currently selected. In the main content area, it says "Authorizers enable you to control access to your APIs using Amazon Cognito User Pools or a Lambda function." A blue button labeled "+ Create New Authorizer" is visible. Below this, a specific authorizer named "WildRydes" is shown, which is associated with the "Cognito User Pool" "WildRydes - Juhh96ipq (us-east-1)". The "Token Source" dropdown is set to "Authorization". At the bottom of the authorizer card, there are "Edit" and "Test" buttons.

## testing the control through API in relation to the website

The screenshot shows the AWS API Gateway console. On the left, a sidebar for the 'WildRydes' API lists various sections like Resources, Stages, and Authorizers. The 'Authorizers' section is currently selected. In the main content area, a modal window titled 'WildRydes - Test Authorizer' displays a JSON response. The response code is 200, latency is 104, and the claims section contains a large JSON object representing an OAuth token. The JSON includes fields such as 'aud', 'auth\_time', 'cognito:username', 'email', 'email\_verified', 'event\_id', 'exp', 'iat', 'iss', 'jti', 'origin\_jti', 'sub', and 'token\_use'. A 'Close' button is at the bottom of the modal.

## setting up API plan

The screenshot shows the AWS API Gateway console. The sidebar on the left is identical to the previous screenshot, showing the 'WildRydes' API with the 'Stages' section selected. In the main content area, a modal window titled 'prod Stage Editor' is open. It shows the 'Settings' tab selected. Under 'Cache Settings', there is an option to 'Enable API cache' with a checkbox. Below that, 'Default Method Throttling' is configured with a rate of 10000 requests per second and a burst of 5000 requests. There is also a note about API Gateway throttling. The 'Web Application Firewall (WAF)' section indicates 'None' is selected. At the bottom, there is a 'Client Certificate' section. A 'Logs/Tracing' tab is also visible in the navigation bar of the modal.

booked my ride :D

