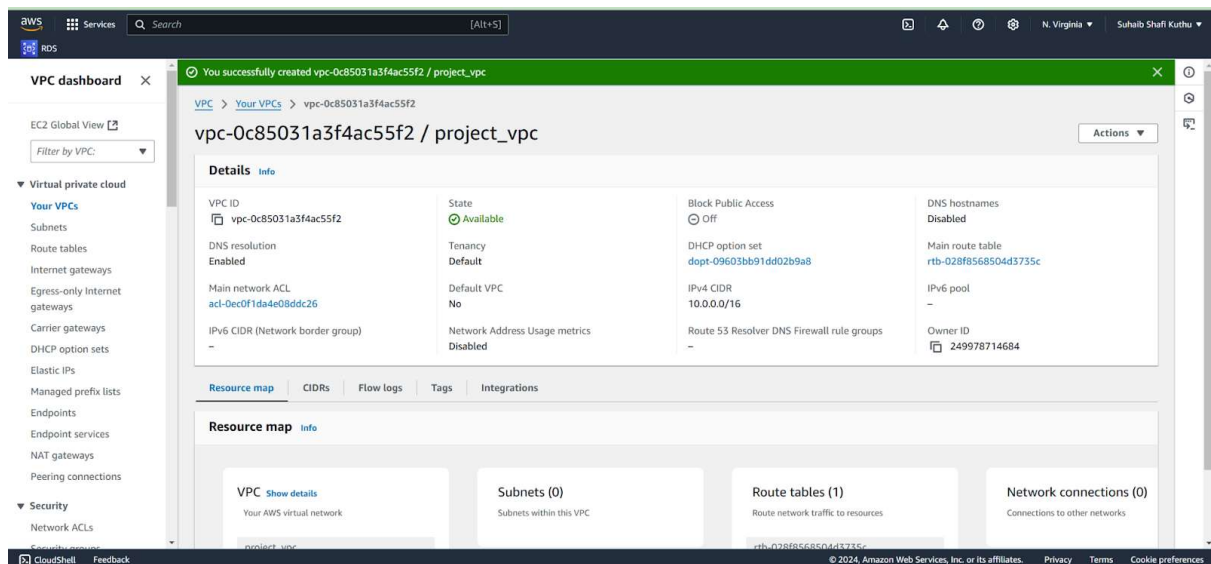


# Project1

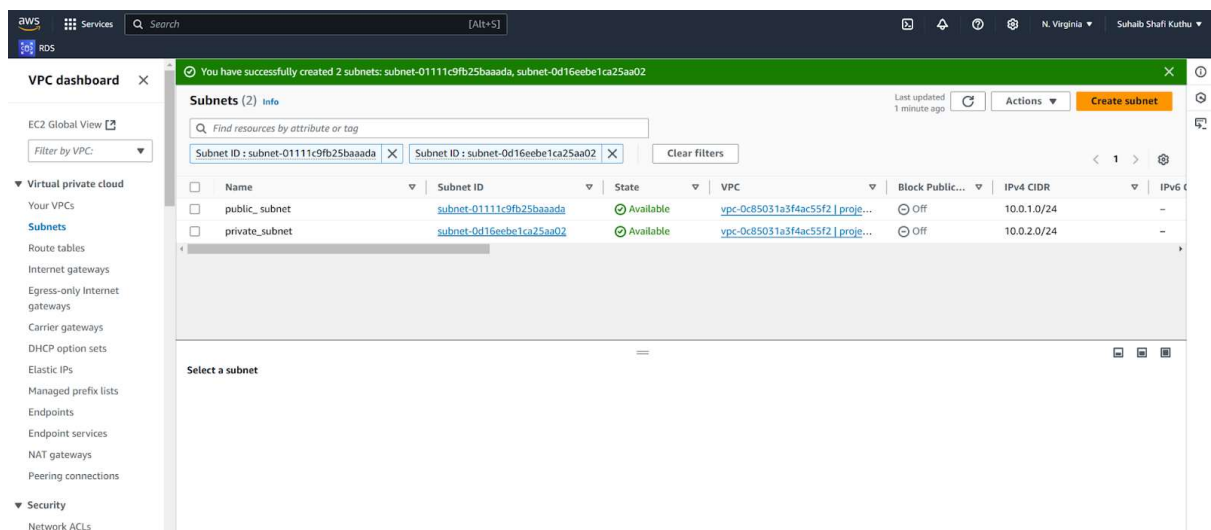
Below are the steps for the procedure to be followed to achieve the required architecture

## Step1:-

Create a vpc where the front end is kept in the public subnet and backend in private subnet.



## Step 2:- create subnets in the vpc with public access and private access



## Step 3 :- create route tables for public and private subnets and associate accordingly

The screenshot shows the AWS Management Console interface for creating a new route table. The breadcrumb navigation indicates the path: VPC > Route tables > Create route table. The page title is 'Create route table' with an 'Info' link. A brief description states: 'A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.'

**Route table settings**

**Name - optional**  
Create a tag with a key of 'Name' and a value that you specify.  
PublicRT

**VPC**  
The VPC to use for this route table.  
vpc-0c85031a3f4ac55f2 (project\_vpc)

**Tags**  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

**Key**  
Name

**Value - optional**  
PublicRT

Buttons: Add new tag, Cancel, Create route table

The screenshot shows the AWS Management Console interface for creating a new route table. The breadcrumb navigation indicates the path: VPC > Route tables > Create route table. The page title is 'Create route table' with an 'Info' link. A brief description states: 'A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.'

**Route table settings**

**Name - optional**  
Create a tag with a key of 'Name' and a value that you specify.  
PrivateRT

**VPC**  
The VPC to use for this route table.  
vpc-0c85031a3f4ac55f2 (project\_vpc)

**Tags**  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

**Key**  
Name

**Value - optional**  
PrivateRT

Buttons: Add new tag, Cancel, Create route table

## Step 4 Subnet associations ,assoociate public subnet with publicRt and private subnet for public rt

VPC > Route tables > rtb-095c6b03b0ed5d4ed > Edit subnet associations

### Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (1/2)

Filter subnet associations

<input type="checkbox"/>	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input checked="" type="checkbox"/>	public_subnet	subnet-01111c9fb25baaada	10.0.1.0/24	-	Main (rtb-028f8568504d3735c)
<input type="checkbox"/>	private_subnet	subnet-0d16eebe1ca25aa02	10.0.2.0/24	-	Main (rtb-028f8568504d3735c)

Selected subnets

subnet-01111c9fb25baaada / public\_subnet X

Cancel Save associations

VPC > Route tables > rtb-09c4bf2287fbbf1de > Edit subnet associations

### Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (1/2)

Filter subnet associations

<input type="checkbox"/>	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input type="checkbox"/>	public_subnet	subnet-01111c9fb25baaada	10.0.1.0/24	-	rtb-095c6b03b0ed5d4ed / PublicRT
<input checked="" type="checkbox"/>	private_subnet	subnet-0d16eebe1ca25aa02	10.0.2.0/24	-	Main (rtb-028f8568504d3735c)

Selected subnets

subnet-0d16eebe1ca25aa02 / private\_subnet X

Cancel Save associations

## Step 5

Create an internet gateway and attach it to vpc to make it available outside as public url.

VPC dashboard X

EC2 Global View

Filter by VPC:

Virtual private cloud

Your VPCs

Subnets

Route tables

Internet gateways

Egress-only Internet gateways

Carrier gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Endpoints

Endpoint services

NAT gateways

Peering connections

Security

The following internet gateway was created: igw-0e1608e6560dc566e - Project\_IG. You can now attach to a VPC to enable the VPC to communicate with the internet. Attach to a VPC X

VPC > Internet gateways > igw-0e1608e6560dc566e

### igw-0e1608e6560dc566e / Project\_IG

Actions

Details Info

Internet gateway ID	State	VPC ID	Owner
igw-0e1608e6560dc566e	Detached	-	249978714684

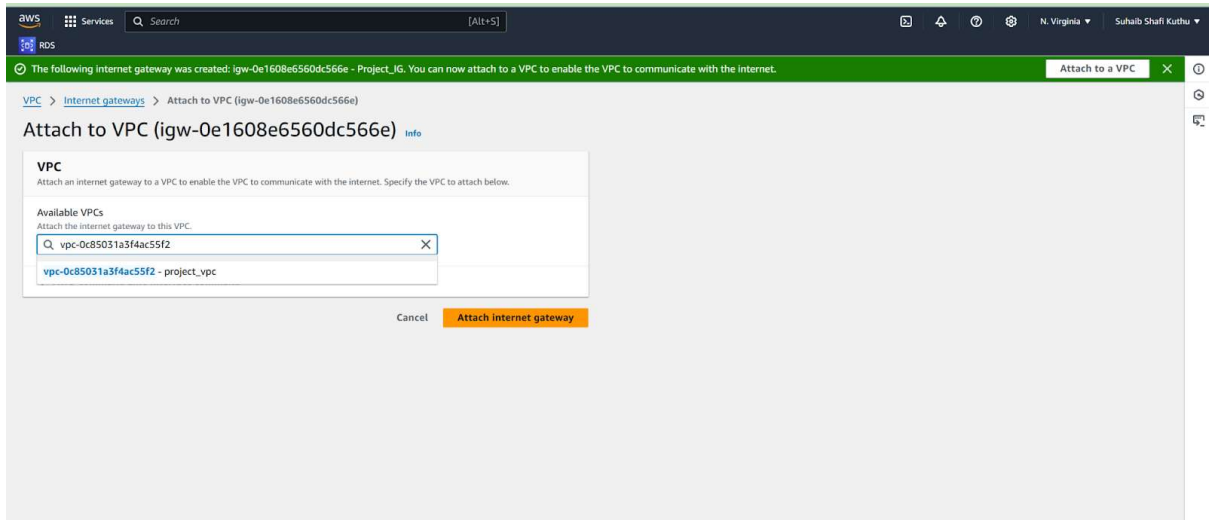
Tags

Search tags

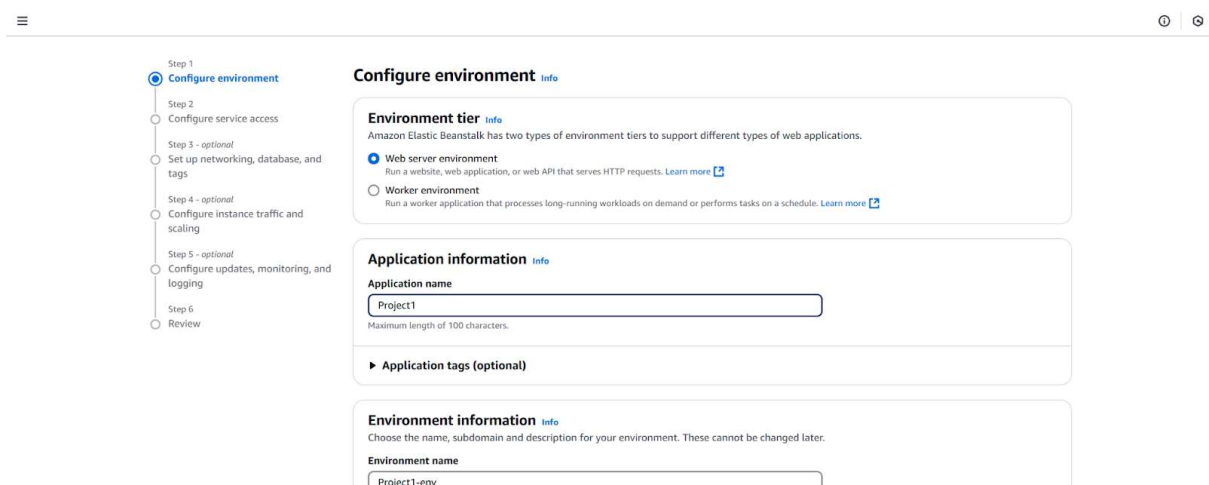
Key	Value
Name	Project_IG

Manage tags

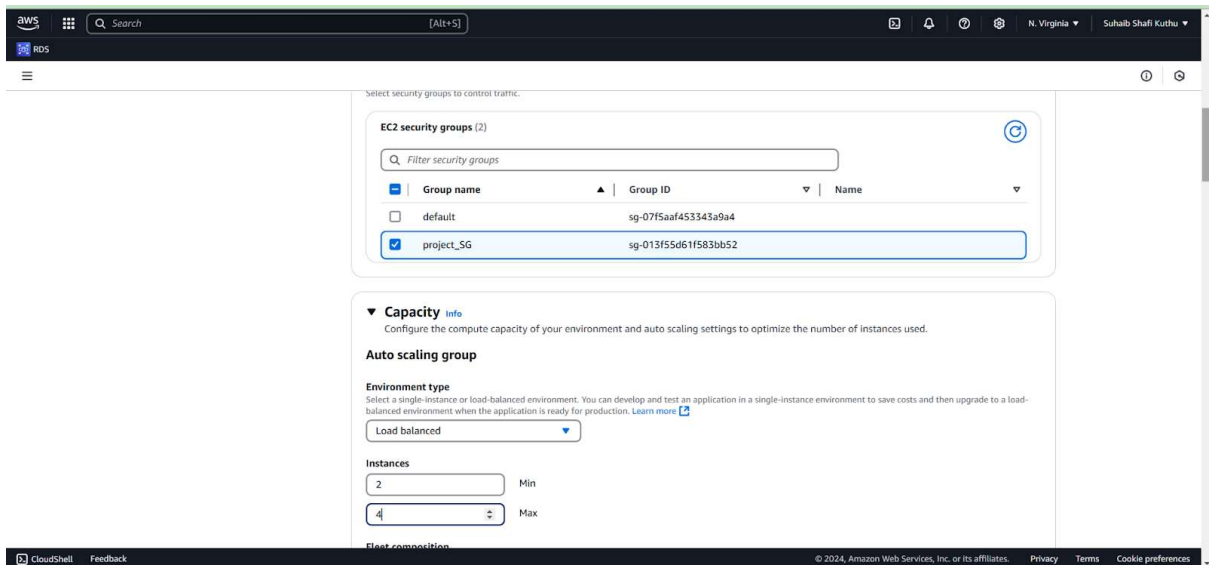
Attach it created vpc



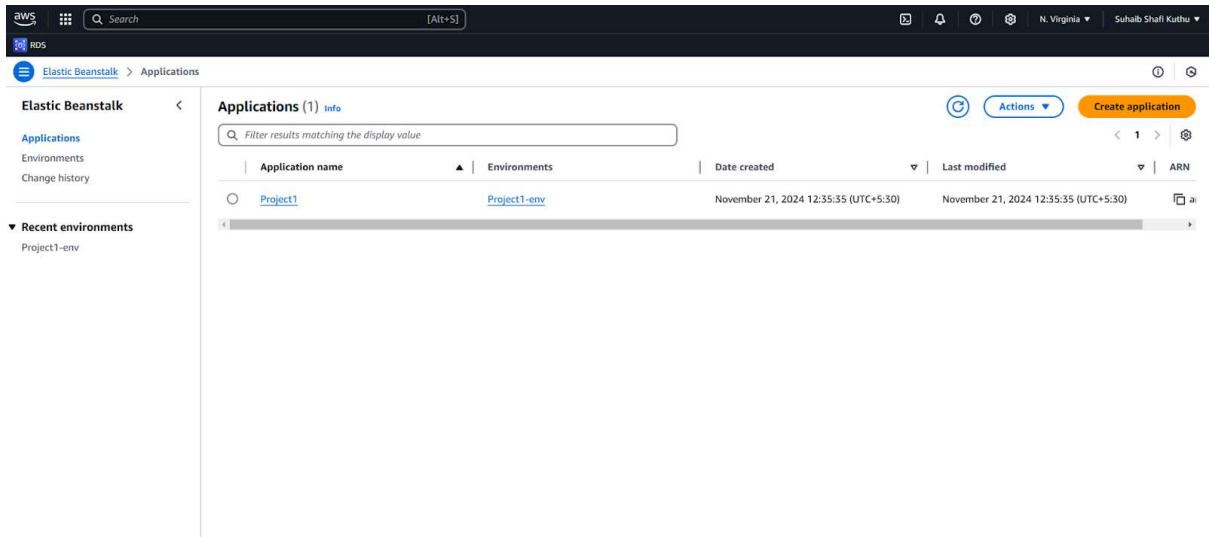
## Step 6 :- go to elastic beanstalk and create an application with node.js environment



## Configure networking and instance and load banacer settings as per the project

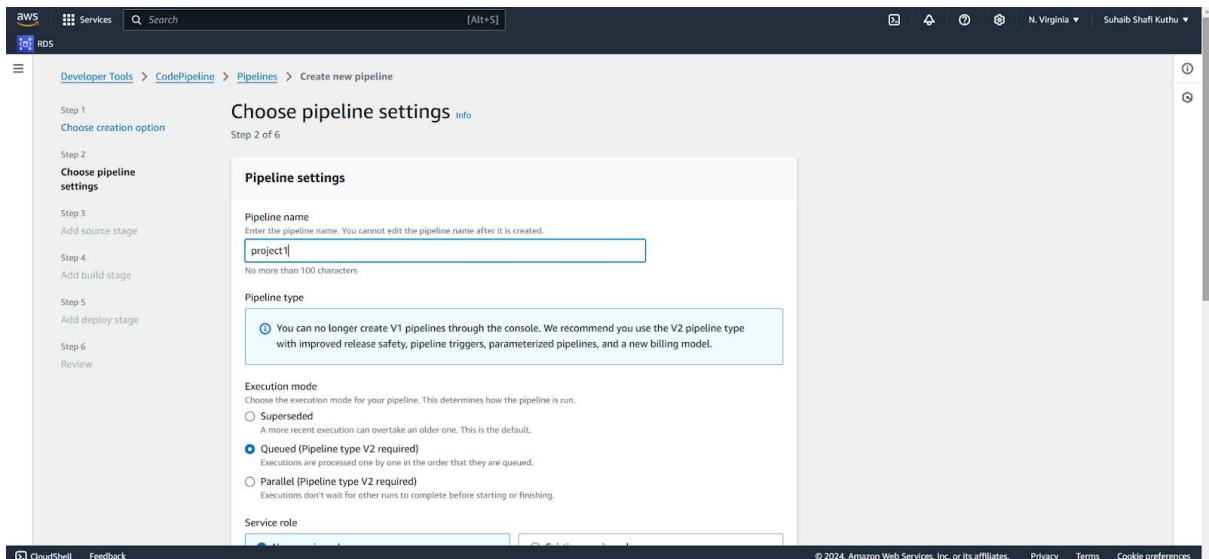


## Project1

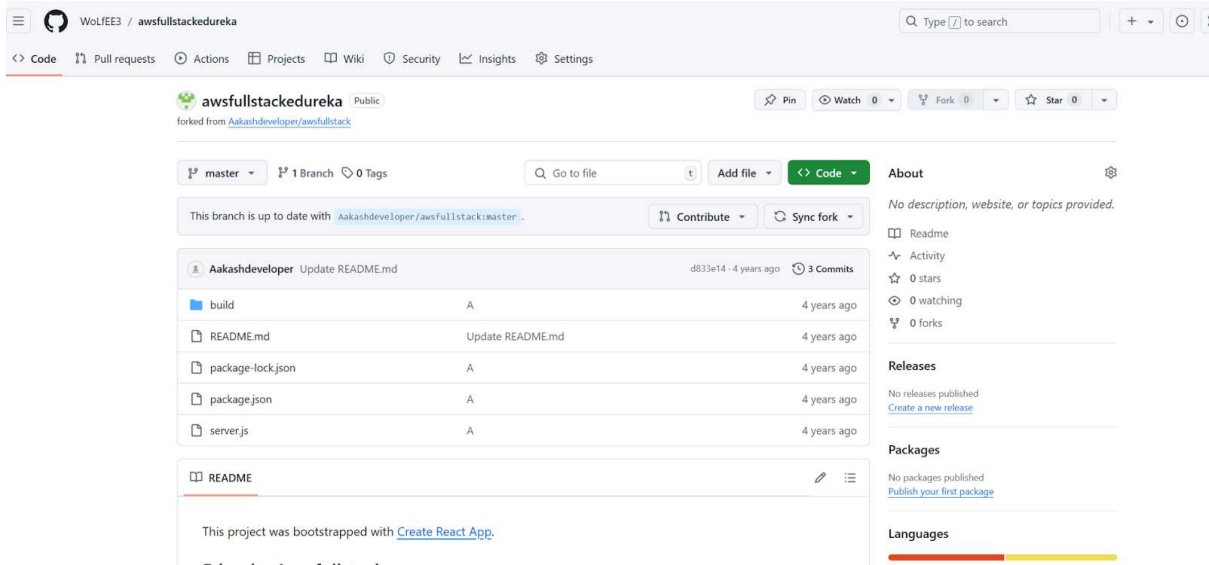


## Step 7

Now for CI/CD go to code pipe line



**Step 8 Now in next step we will select the source from where we have to pick the code and deploy**



## Step 9 Select the repo and branch used

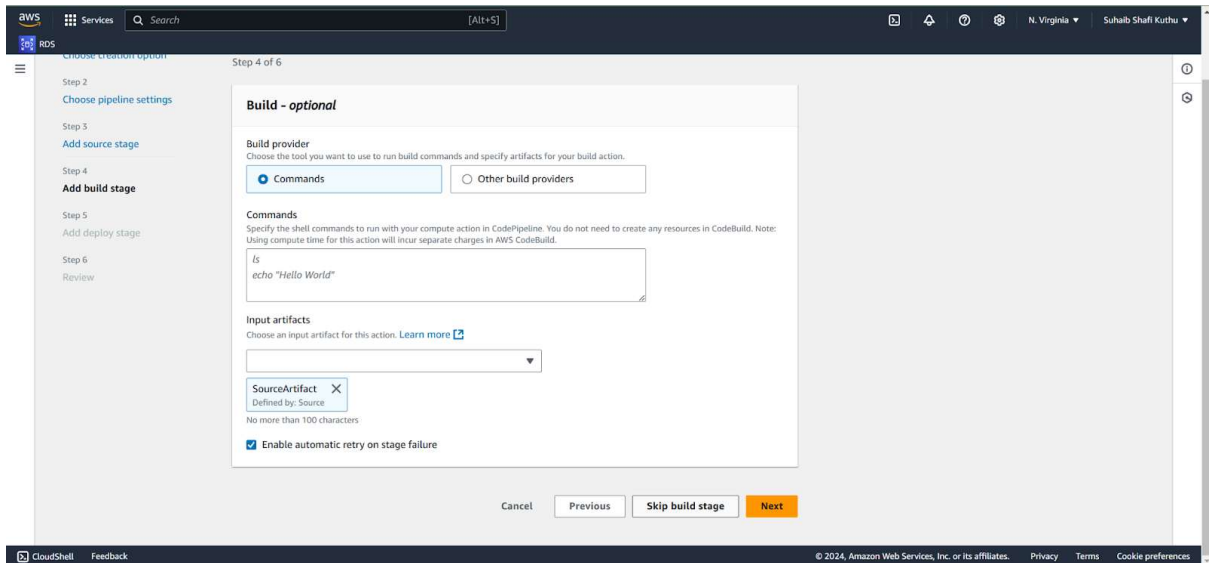
### The github branch

The top screenshot shows the GitHub repository page for `awsfullstackedureka`, which is a public repository forked from `AakashDeveloper/awsfullstack`. The repository is on the `master` branch. The file list includes `build`, `README.md`, `package-lock.json`, `package.json`, and `server.js`. The `README` file is selected, showing the text: "This project was bootstrapped with [Create React App](#)." and "Edureka Aws fullstack app".

The bottom screenshot shows the AWS CodePipeline console. The "Add source stage" step is selected. The "Source provider" is set to "GitHub (via OAuth app)". The "Repository" is `WoLFEE3/awsfullstackedureka` and the "Branch" is `master`. The "Change detection options" are set to "GitHub webhooks (recommended)". A message indicates that the action has been successfully configured with the provider.

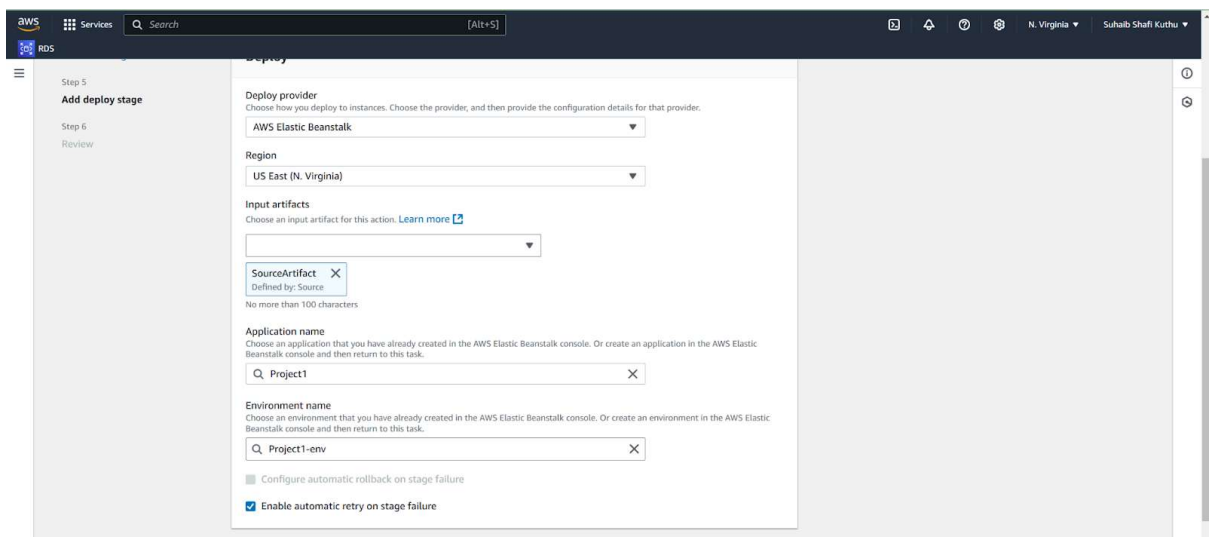
## Step 10

### For this demo we can skip the build stage of the application as it more like core script need to build app



## Step 11

Now select the server where we want to deploy the application and i.e. beanstalk that we have launched

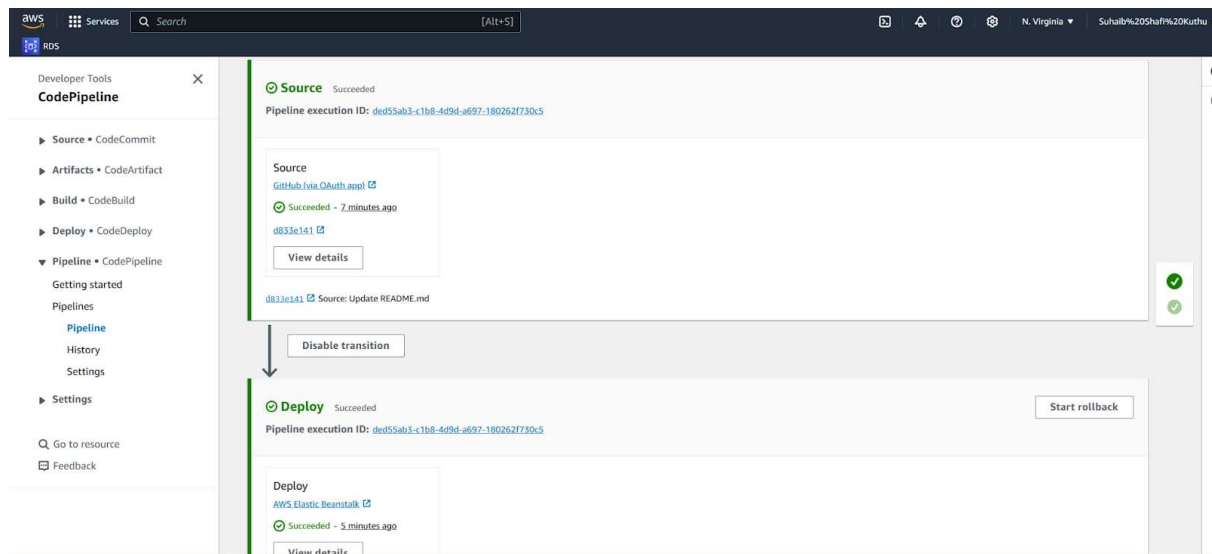


## Step 12

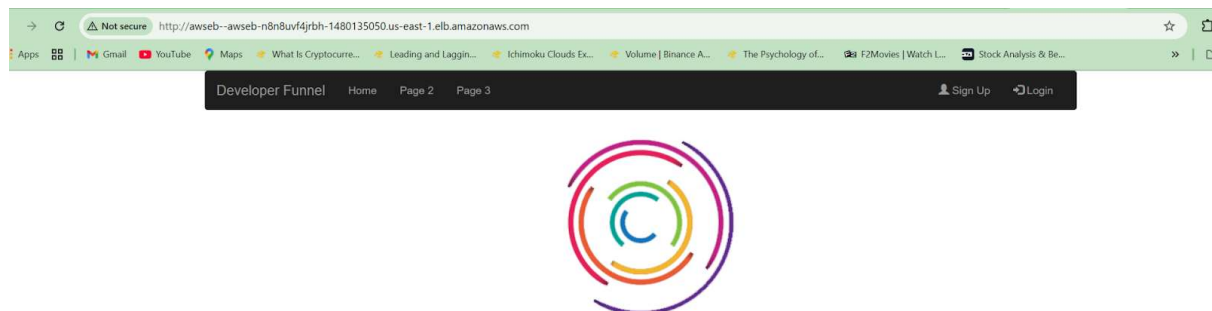
Review the changes



**Step 13 :-As soon as the code commit on the master branch this pipeline will execute and deploy code on the server.**



**As seen the app is deployed successfully**

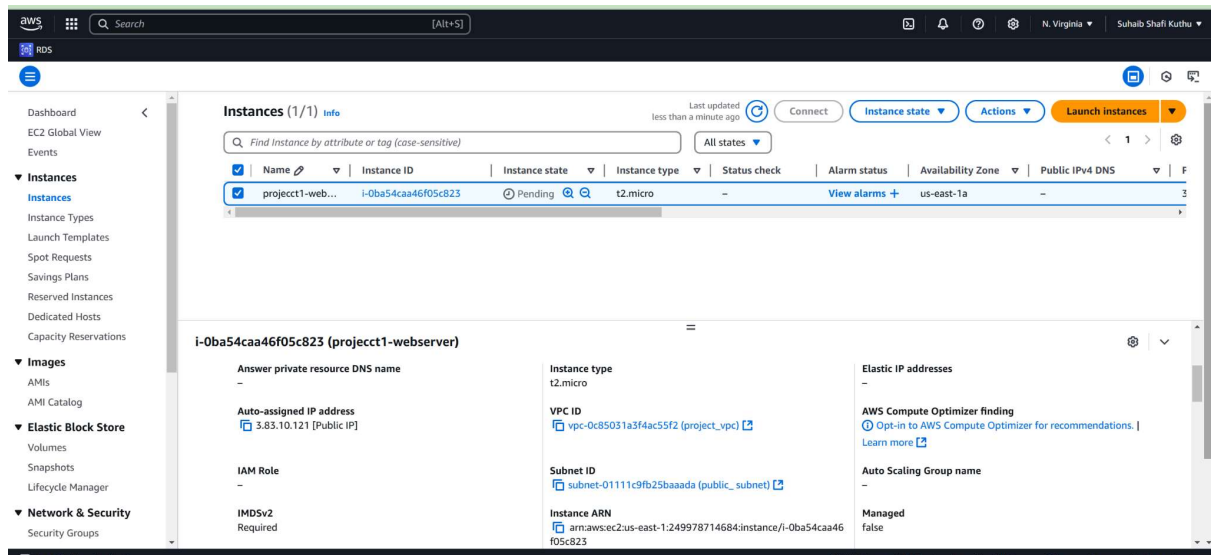


**Now moving on to the backend part**

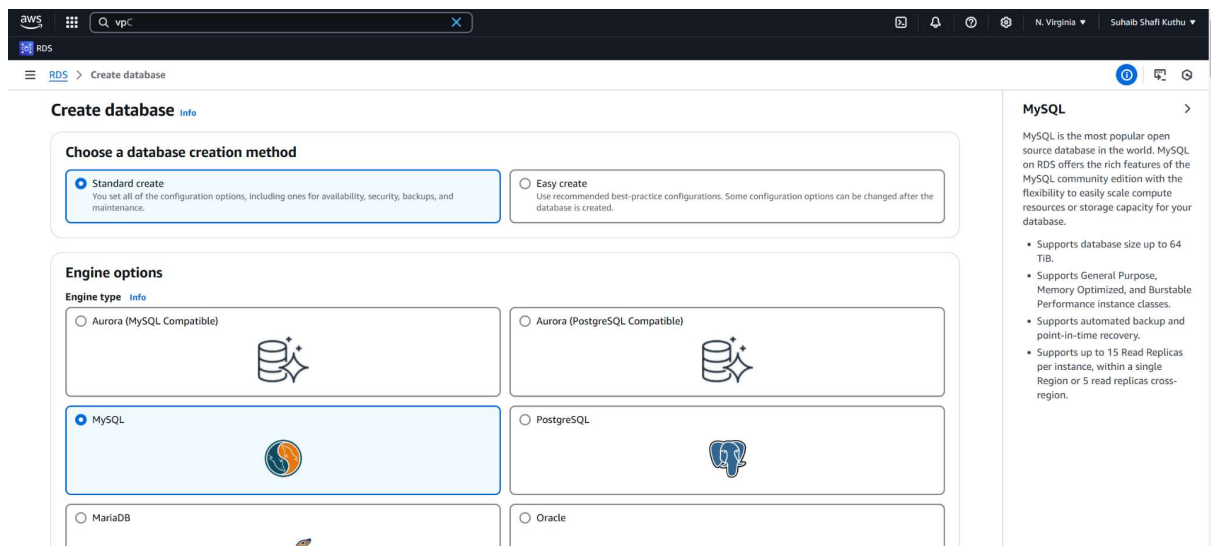
## Step 1

Launch an ec2 instance in the public subnet

And a database in the private subnet so that it can only be accessed by only the instance in public subnet, as they are in the same vpc

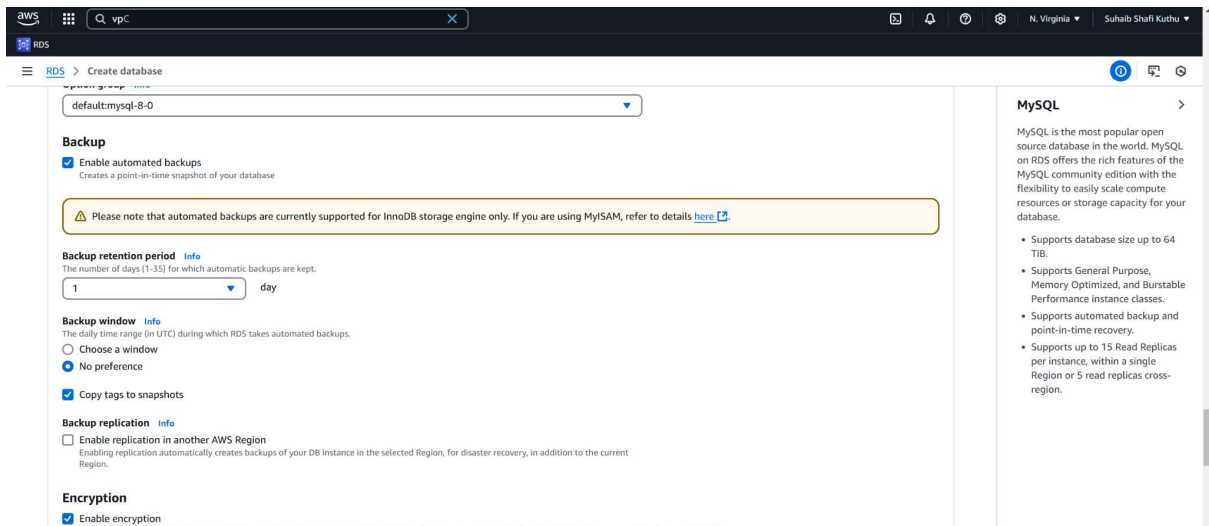


Now for db launch go to RDS and launch a my sql database



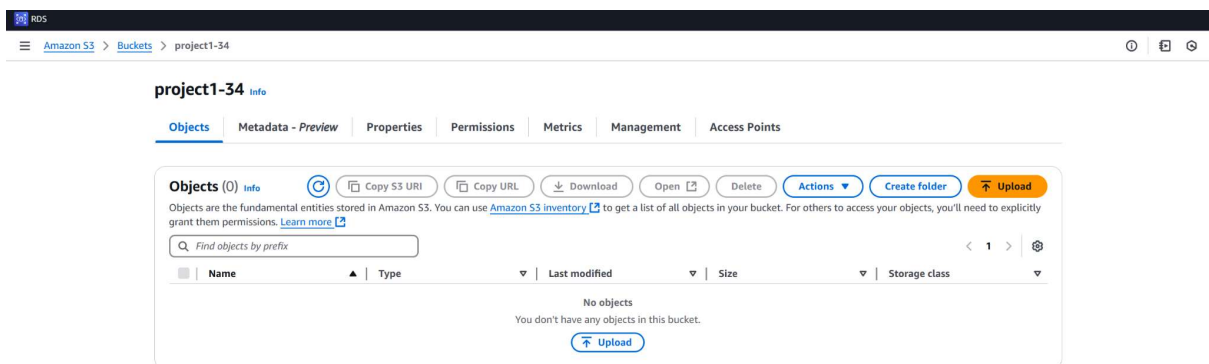
To retain the backup enable backups in database settings

Backups for your Web service may include creating snapshots of the instance or creating an Image of the Instance.

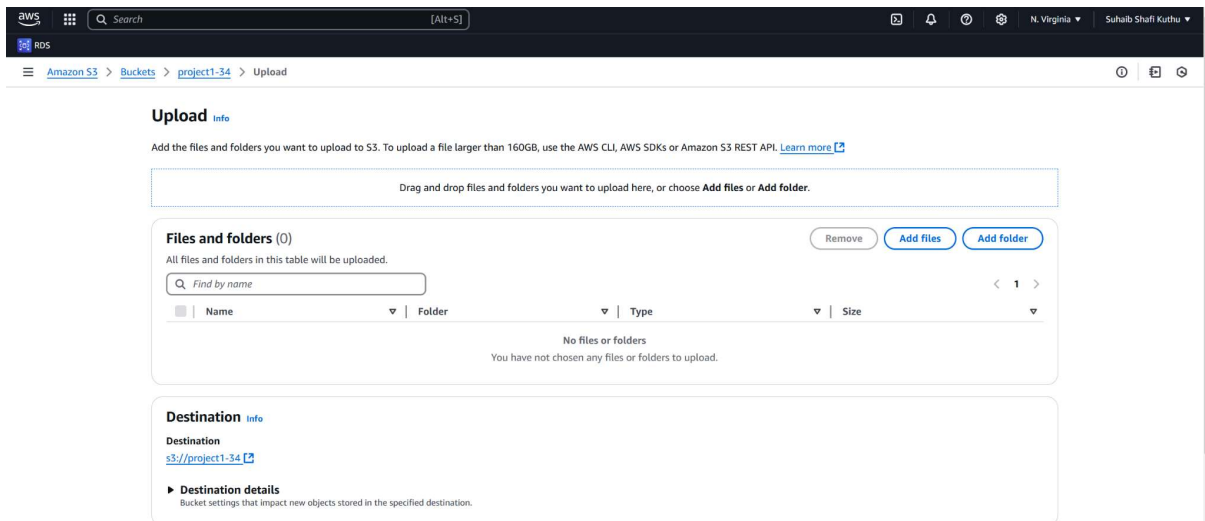


Ensure all the UI images served to the frontend application code are provisioned via a secure storage unit

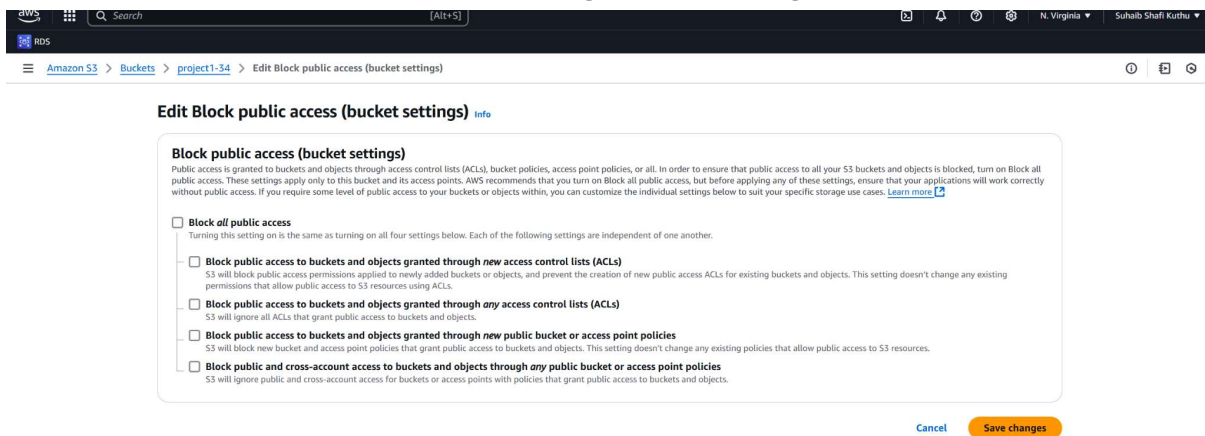
## Create S3 Bucket



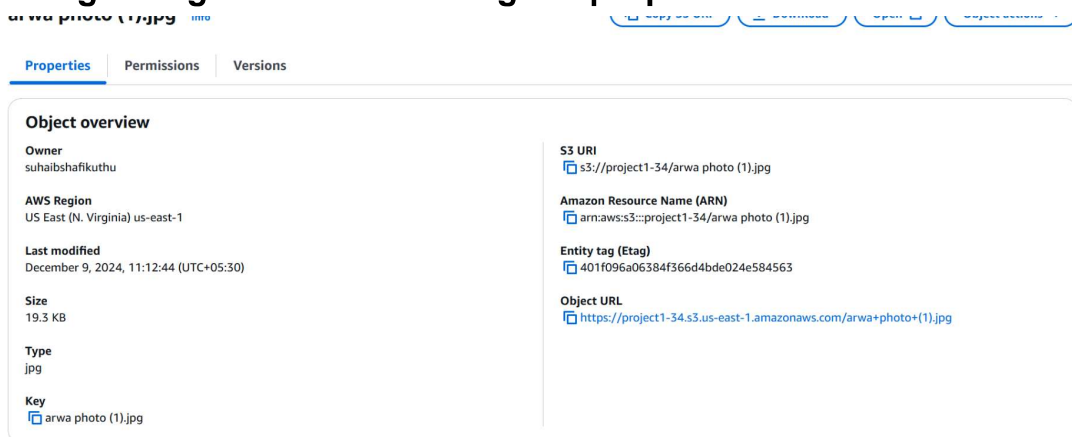
Upload files to s3 bucket



## Make the Bucket Public So we can grab the images

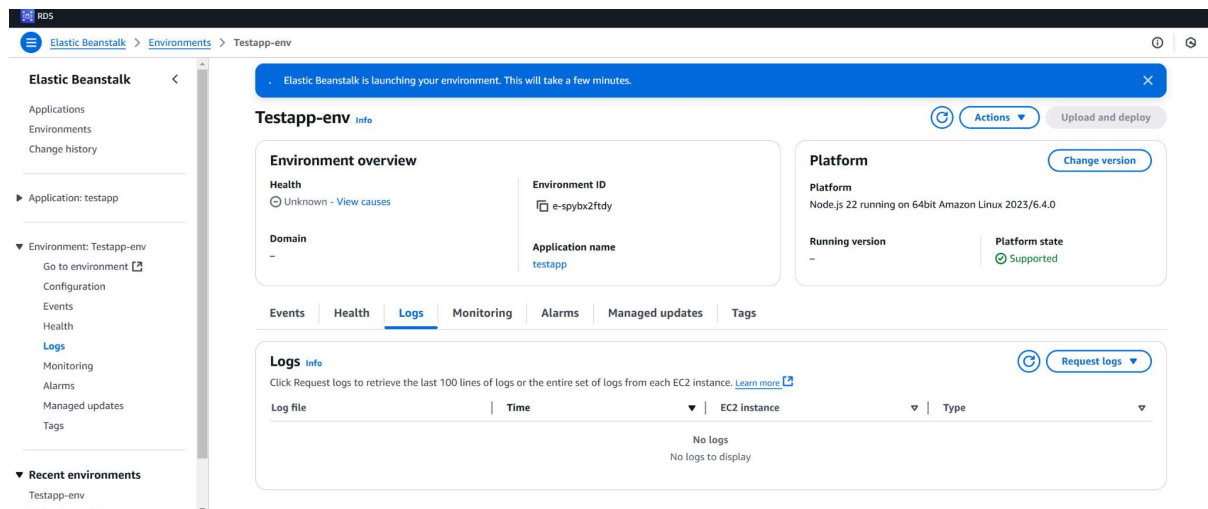


## Find the URLs for your files by clicking on the bucket name, then selecting a single file and showing the properties

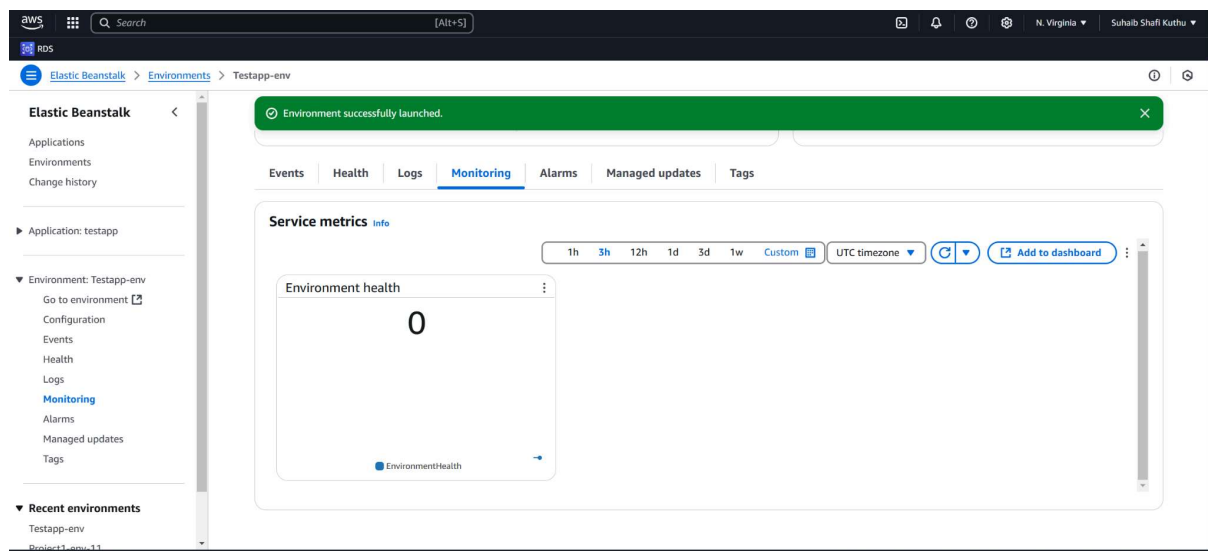


Automate the download all the activity logs into a CSV file,  
create a stream of data,  
analyze it, and display it via a dashboard

In the Elastic BeanStalk console, you will find the option to enable cloud watch, ( Elastic Beanstalk-> Create application->configure more options). Now click on enable monitoring. Then go to Cloudwatch and Create a dashboard to view and analyze the logs .



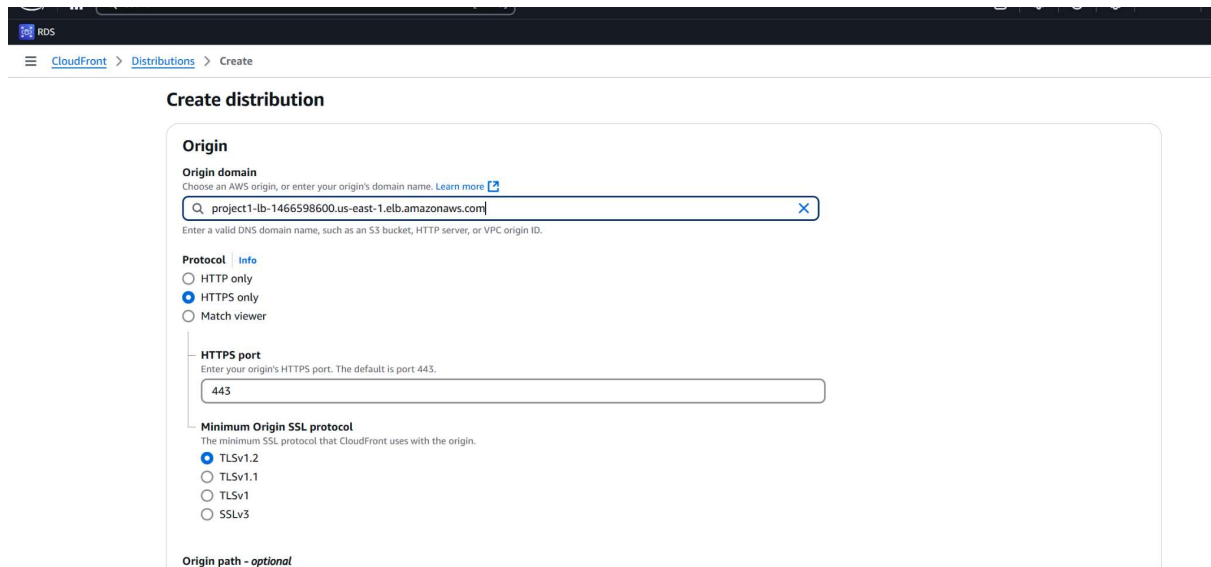
Now to display it on a dashboard  
Go to monitoring and add a dashboard



The Web application should also be cached globally, so users worldwide can access it with low latency

Create a CloudFront distribution to cache our application globally. Navigate to your CloudFront and click on create distribution Under web section click on get started On the Create Distribution page,under

**Origin Settings, choose the ELB that you created earlier Give the original path as project.php**



The screenshot shows the 'Create distribution' page in the AWS Management Console, specifically the 'Origin' tab. The 'Origin domain' field contains 'project1-lb-1466598600.us-east-1.elb.amazonaws.com'. The 'Protocol' section has 'HTTPS only' selected. The 'HTTPS port' field contains '443'. The 'Minimum Origin SSL protocol' section has 'TLSv1.2' selected. The 'Origin path - optional' field is empty.

**Create distribution**

**Origin**

**Origin domain**  
Choose an AWS origin, or enter your origin's domain name. [Learn more](#)

project1-lb-1466598600.us-east-1.elb.amazonaws.com

Enter a valid DNS domain name, such as an S3 bucket, HTTP server, or VPC origin ID.

**Protocol** [Info](#)

☐ HTTP only

☒ HTTPS only

☐ Match viewer

**HTTPS port**  
Enter your origin's HTTPS port. The default is port 443.

443

**Minimum Origin SSL protocol**  
The minimum SSL protocol that CloudFront uses with the origin.

☒ TLSv1.2

☐ TLSv1.1

☐ TLSv1

☐ SSLv3

**Origin path - optional**

**Keep the rest as defaults and create a distribution Once your distribution is created, you will be able to see your website through your domain name of your distribution Check whether your infrastructure is working or not by terminating one of your instances, so that a new instance gets created by your auto scaling group**

**As elastic beanstalk allows us to create the load balancer and autoscaling group in their settings itself no separate creation of LB an Auto-scaling was shown.**

