AWS cloud assignment

• Create a virtual network with 2 subnets. Each subnet should have 16 Ips only.

• Inside one of the subnets, create a VM and deploy an application code inside it (any existing application created by you before). Make sure to use appropriate NACLs and SGs.

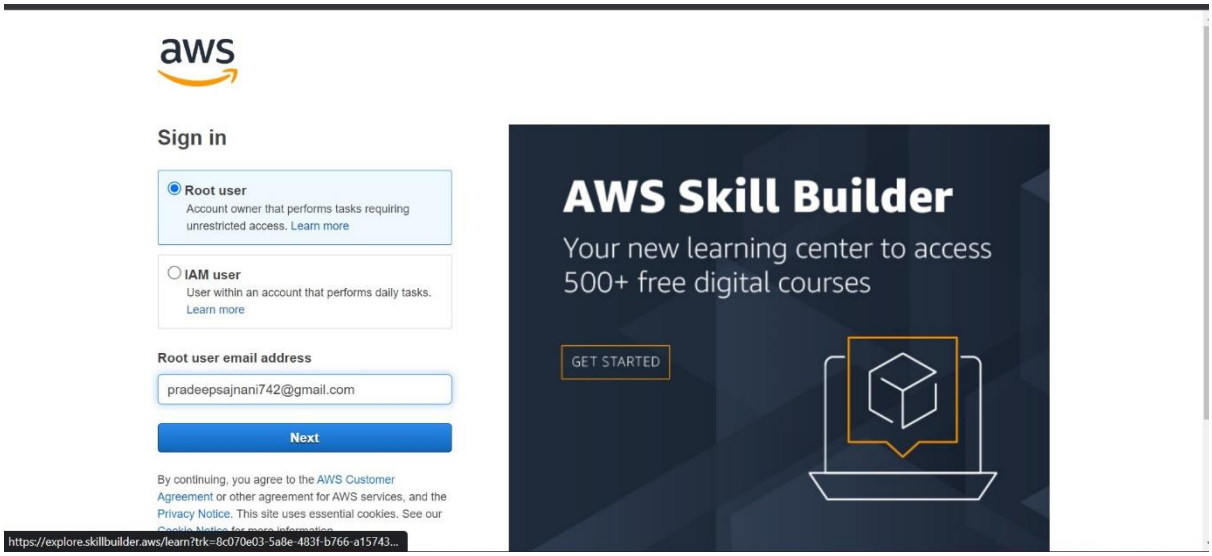
• Deploy the same application to Elastic beanstalk Service.

• Create a Lambda that should trigger as soon as you upload a file in the S3 bucket. Function should be able to print the name of the file uploaded in the function.

Note: • Delete the resources after taking the screenshots

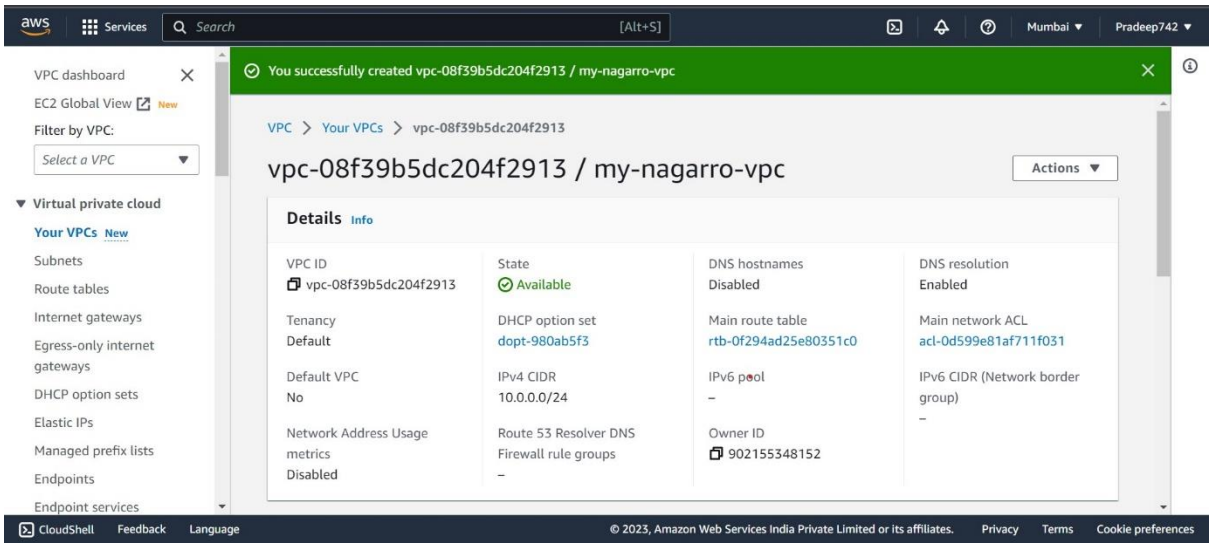
Q1. **Create a virtual network with 2 subnets. Each subnet should have 16 Ips only.**

Step 1. Log in to the AWS Management Console at https://console.aws.amazon.com

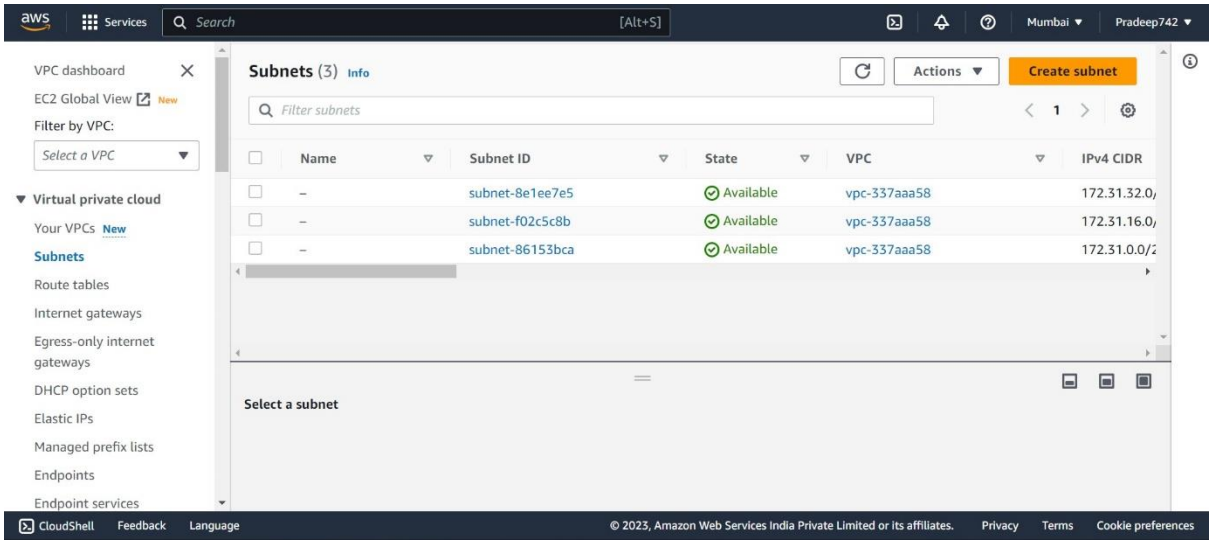


Step 2. Navigate to the Amazon **VPC** service by searching for "**VPC**" in the search bar or by selecting it from the services menu.

- Click on "Your VPCs" from the left navigation pane.
- Click on the "Create VPC" button.
- In the **"Create VPC"** wizard, enter a name for your VPC in the "Name tag" field.
- Enter an IPv4 CIDR block for your VPC. Since you need each subnet to have 16 IP addresses, you can use a /28 CIDR block, which provides 16 addresses. For example, you can enter "**10.0.0.0/24"** as the CIDR block. Note that the CIDR block you choose must not overlap with any existing networks.
- Leave the "IPv6 CIDR block" field empty unless you specifically need IPv6 addresses.
- Click on the "Create" button to create the VPC.



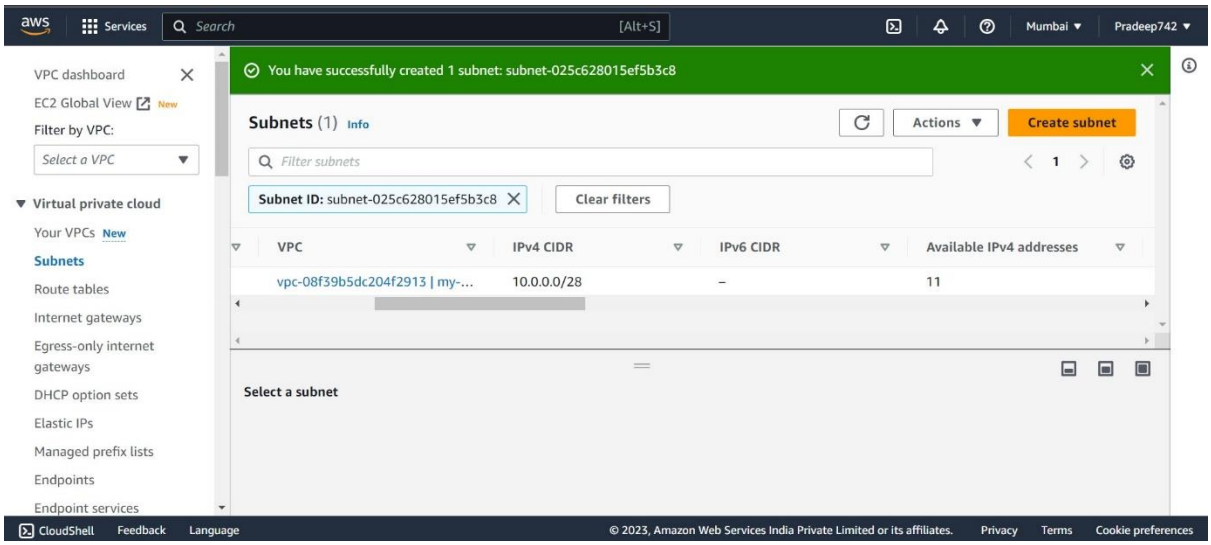
Step 3. Now that you have created the VPC, you can proceed with creating the subnets:



Now, let's proceed with creating the subnets:

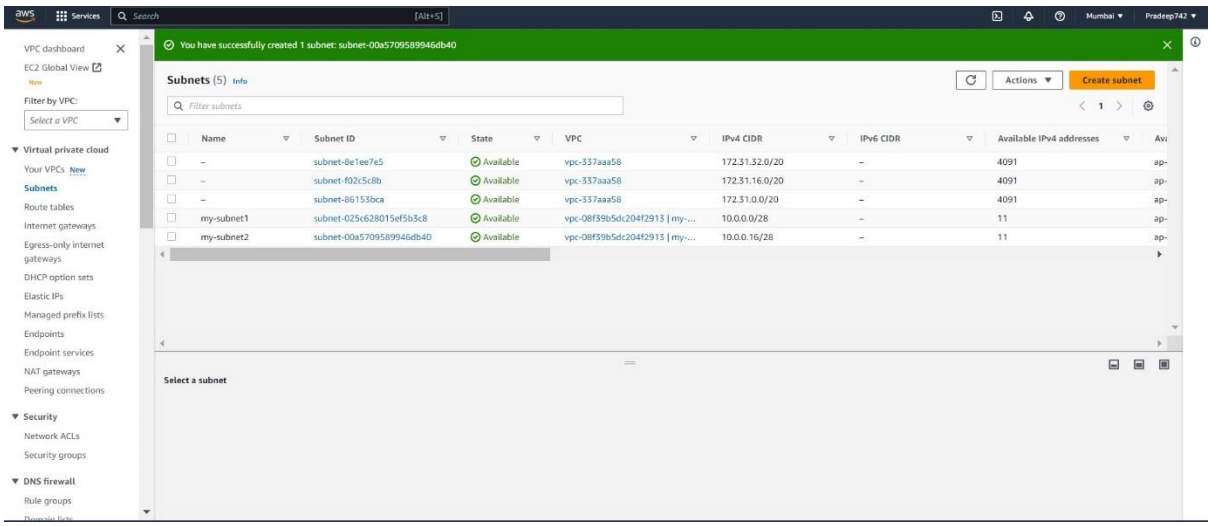
- In the Amazon VPC console, click on "Subnets" from the left navigation pane.
- Click on the "**Create subnet**" button.
- In the "Create subnet" wizard, select the VPC you created earlier from the "VPC" dropdown menu.
- Provide a name for your subnet in the "Na**me tag**" field.
- Select the availability zone for your subnet from the "**Availability Zone**" dropdown menu.

- Enter an IPv4 CIDR block for your subnet within the range of the VPC CIDR block. Since you need 16 IP addresses, you can use a /28 CIDR block. For example, you can start with "**10.0.0.0/28**" for the first subnet.
- Click on the "Create" button to create the subnet.



Repeat the above steps to create the second subnet:

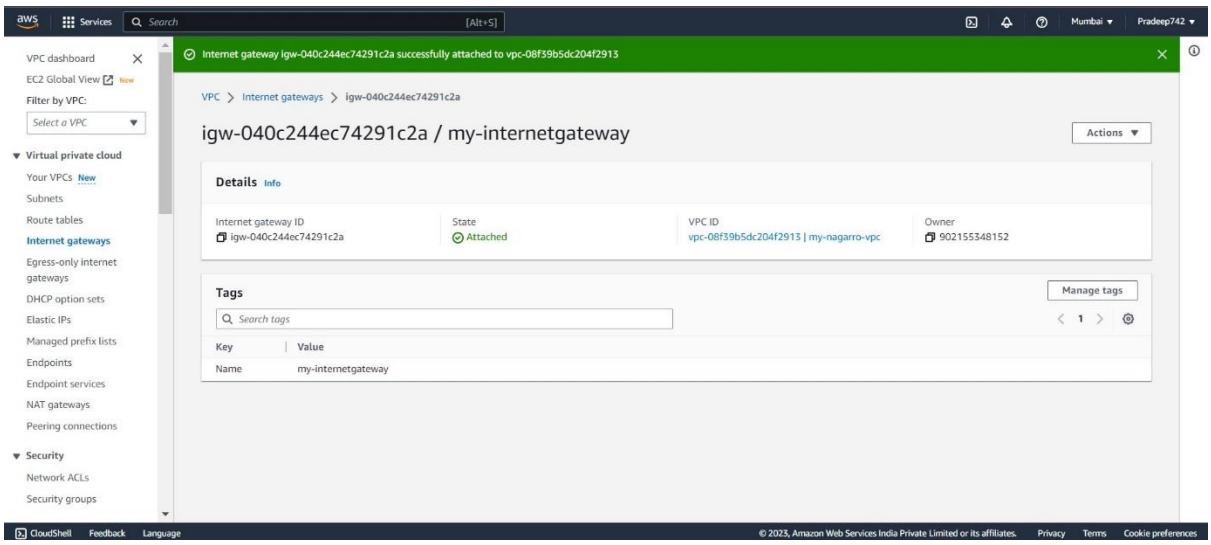
- Click on the "Create subnet" button again.
- Select the same VPC as before from the "VPC" dropdown menu.
- Provide a name for your second subnet.
- Choose a different availability zone from the "Availability Zone" dropdown menu.
- Enter a different CIDR block for your second subnet within the VPC CIDR block. For example, you can use "**10.0.0.16/28**" for the second subnet.
- Click on the "Create" button to create the second subnet.
- Now you should have two subnets within the VPC, each with 16 available IP addresses. Remember to adjust the CIDR blocks based on your requirements and the size of the VPC CIDR block.



Now We create **Internet Gateway**
An internet gateway is a crucial component in Amazon Web Services (AWS) that enables communication between resources within your Virtual Private Cloud (VPC) and the internet. It acts as a gateway between your VPC and the public internet, allowing traffic to flow in and out.

To create an internet gateway in AWS, you can follow these steps:

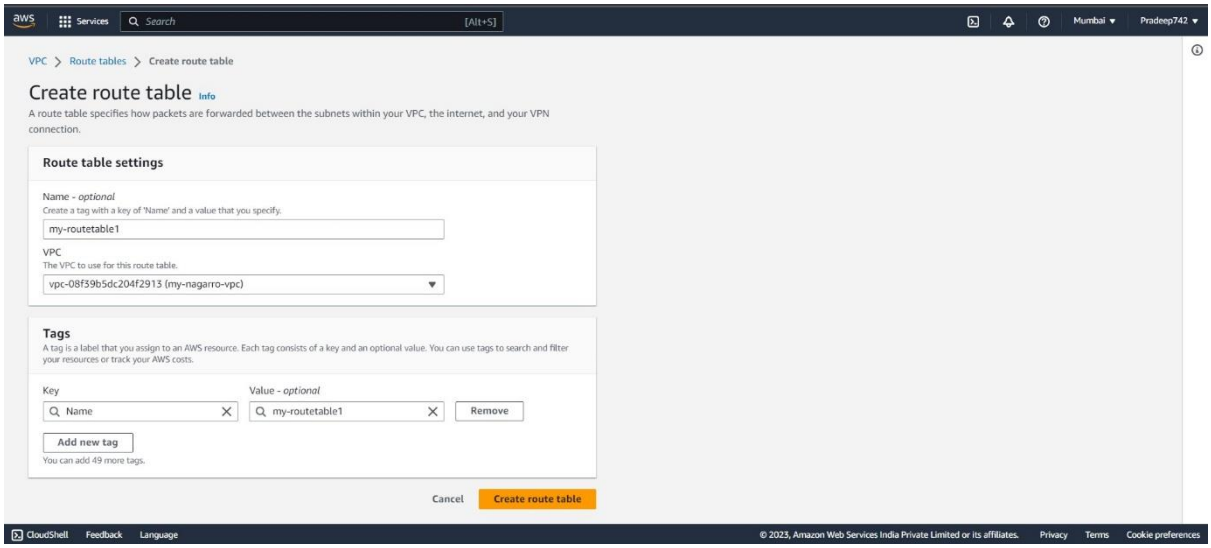
- Go to the Amazon VPC service in the AWS Management Console.
- Click on "Internet Gateways" from the left navigation pane.
- Click on the "Create internet gateway" button.
- Provide a name for your internet gateway (optional).
- Click on the "Create" button to create the internet gateway.
- Once created, select the newly created internet gateway and click on the "Actions" button.
- Choose "Attach to VPC" from the dropdown menu.
- Select the VPC you want to associate the internet gateway with and click on the "Attach" button.

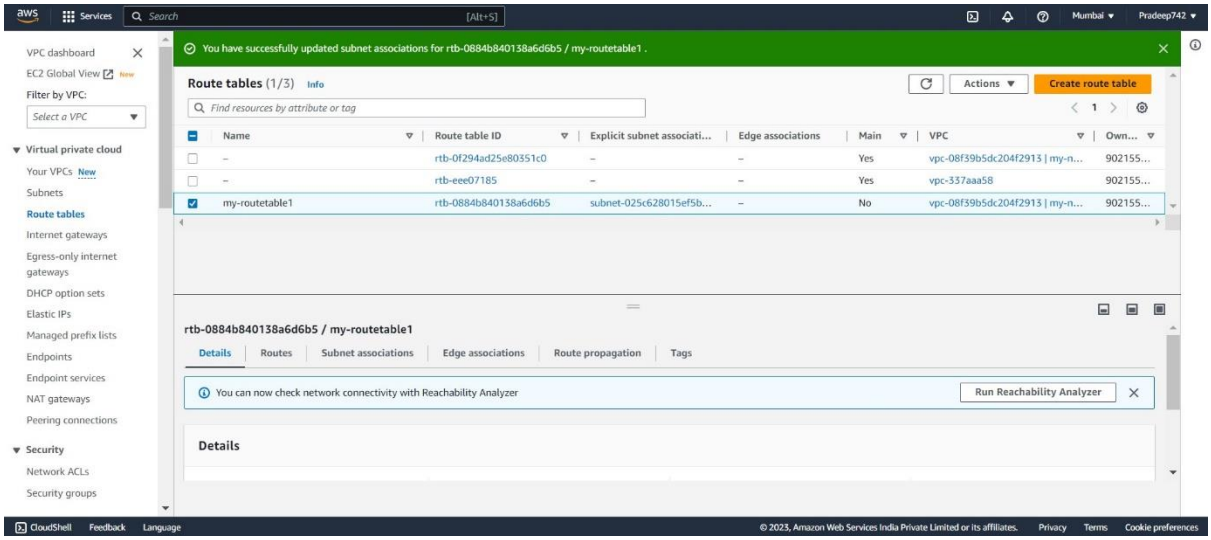


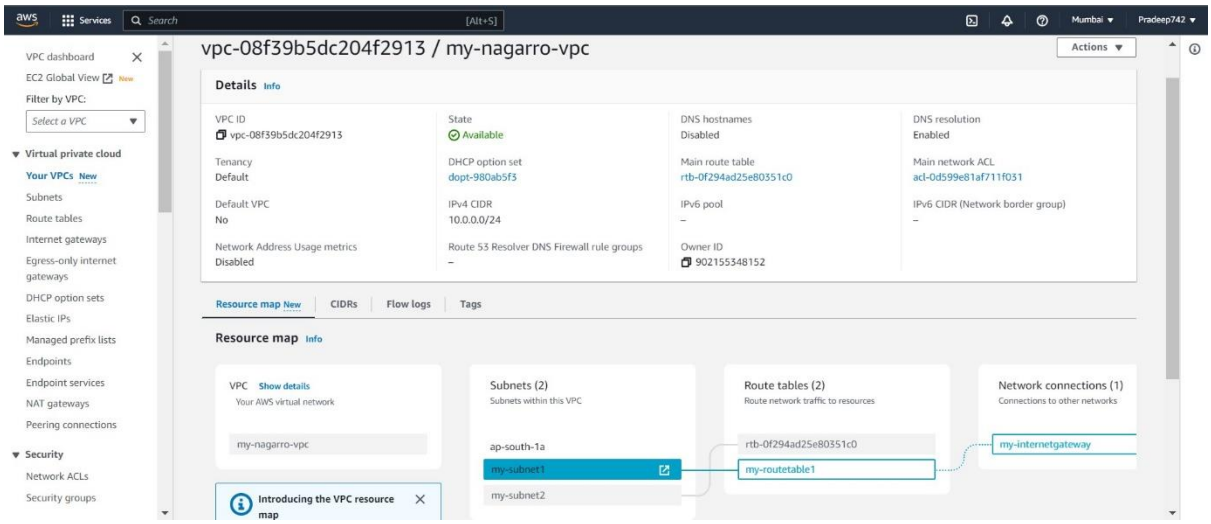
After attaching the internet gateway to your VPC, you will need to configure route tables to direct traffic to and from the internet gateway. This involves adding a route in the VPC's route table with a destination of **"0.0.0.0/0"** (all IPv4 addresses) and the target set as the internet gateway.

Remember to also configure security groups and network ACLs to control the inbound and outbound traffic to your resources within the VPC.

By creating an internet gateway and configuring routing, you can enable internet connectivity for your resources in the AWS VPC, allowing them to communicate with the internet and external services as required.


Services
Search
[Alt+S]
Mumbai
Pradeep742
VPC > Route tables > Create route table
Create route table Info
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.
my-routetable1
VPC
The VPC to use for this route table.
vpc-08f39b5dc204f2913 (my-nagarro-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
Value - optional
Name
my-routetable1
Remove
Add new tag
You can add 49 more tags.
Cancel
Create route table
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Alt+S]
Mumbai
Pradeep742
VPC dashboard
EC2 Global View
Filter by VPC:
Select a VPC
Virtual private cloud
Your VPCs New
Subnets
Route tables
Internet gateways
Egress-only internet gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Security
Network ACLs
Security groups
You have successfully updated subnet associations for rtb-0884b840138a6d6b5 / my-routetable1 .
Route tables (1/3) Info
Actions
Create route table
Find resources by attribute or tag
Name
Route table ID
Explicit subnet associati...
Edge associations
Main
VPC
Own...
rtb-0f294ad25e80351c0
rtb-eee07185
my-routetable1
rtb-0884b840138a6d6b5
subnet-025c628015ef5b...
No
vpc-08f39b5dc204f2913 | my-n...
vpc-337aaa58
902155...
rtb-0884b840138a6d6b5 / my-routetable1
Details
Routes
Subnet associations
Edge associations
Route propagation
Tags
You can now check network connectivity with Reachability Analyzer
Run Reachability Analyzer
Details
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

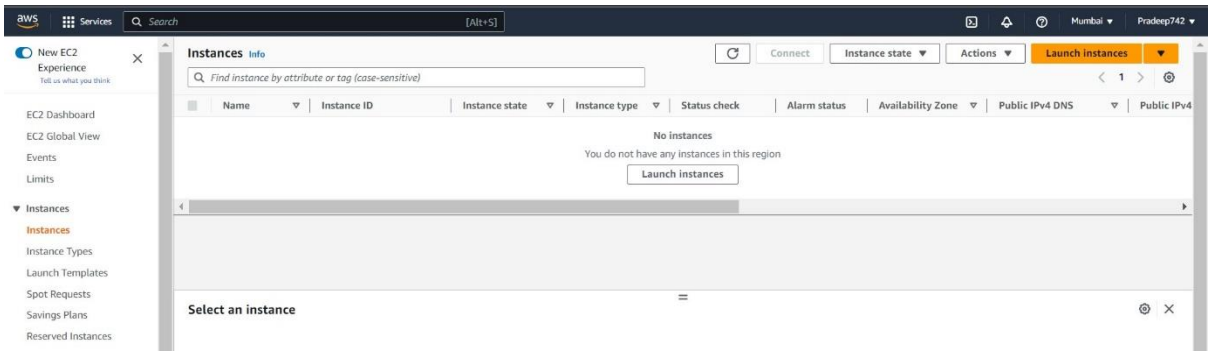

Services
Search
[Alt+S]
Mumbai
Pradeep742
VPC dashboard
EC2 Global View
Filter by VPC:
Select a VPC
Virtual private cloud
Your VPCs New
Subnets
Route tables
Internet gateways
Egress-only internet gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Security
Network ACLs
Security groups
vpc-08f39b5dc204f2913 / my-nagarro-vpc
Actions
Details Info
VPC ID
vpc-08f39b5dc204f2913
State
Available
DNS hostnames
Disabled
DNS resolution
Enabled
Tenancy
Default
DHCP option set
dopt-980ab5f3
Main route table
rtb-0f294ad25e80351c0
Main network ACL
acl-0d599e81af711f031
Default VPC
No
IPv4 CIDR
10.0.0.0/24
IPv6 pool
–
IPv6 CIDR (Network border group)
–
Network Address Usage metrics
Disabled
Route 53 Resolver DNS Firewall rule groups
–
Owner ID
902155548152
Resource map New
CIDRs
Flow logs
Tags
Resource map Info
VPC Show details
Your AWS virtual network
my-nagarro-vpc
Subnets (2)
Subnets within this VPC
ap-south-1a
my-subnet1
my-subnet2
Route tables (2)
Route network traffic to resources
rtb-0f294ad25e80351c0
my-routetable1
Network connections (1)
Connections to other networks
my-internetgateway
Introducing the VPC resource map

Q2. Inside one of the subnets, create a VM and deploy an application code inside it (any existing application created by you before). Make sure to use appropriate NACLs and SGs.

Step 1. Log in to the AWS Management Console at https://console.aws.amazon.com/.

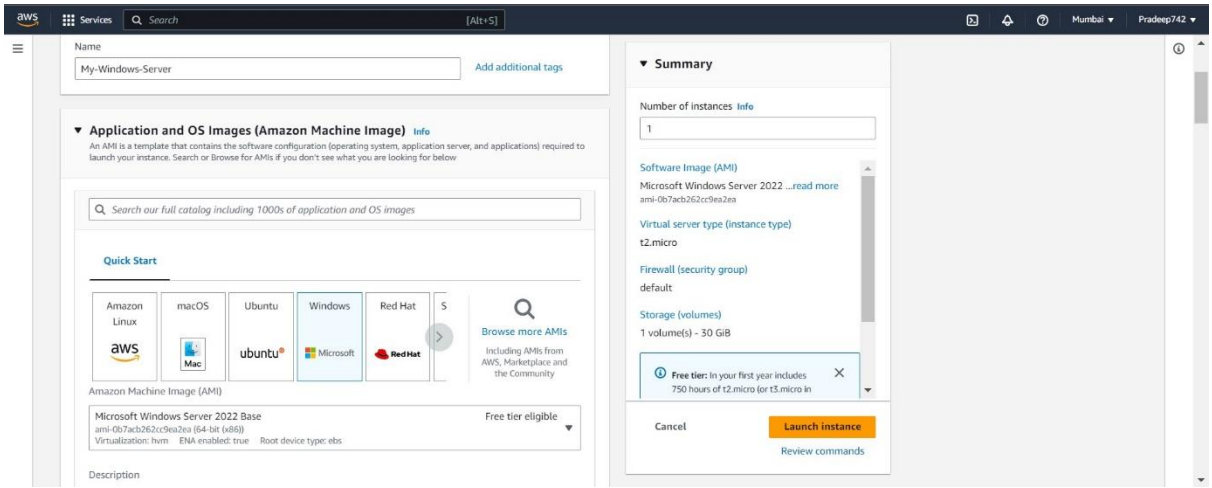
Navigate to the EC2 service by searching for "EC2" in the search bar or by selecting it from the services menu.

In the EC2 Dashboard, click on "Instances" from the left navigation pane.

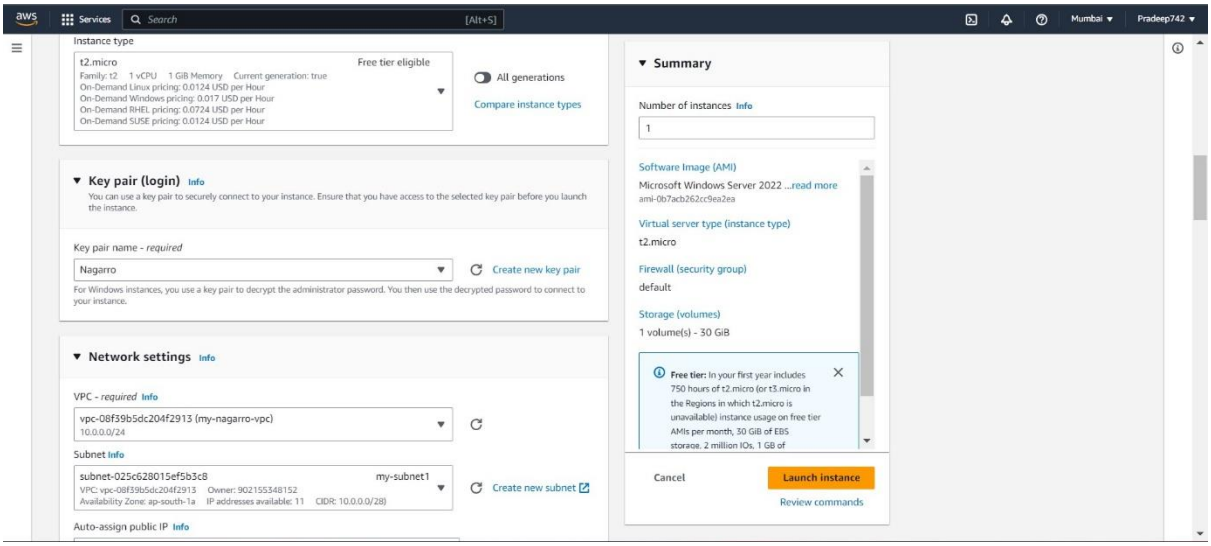


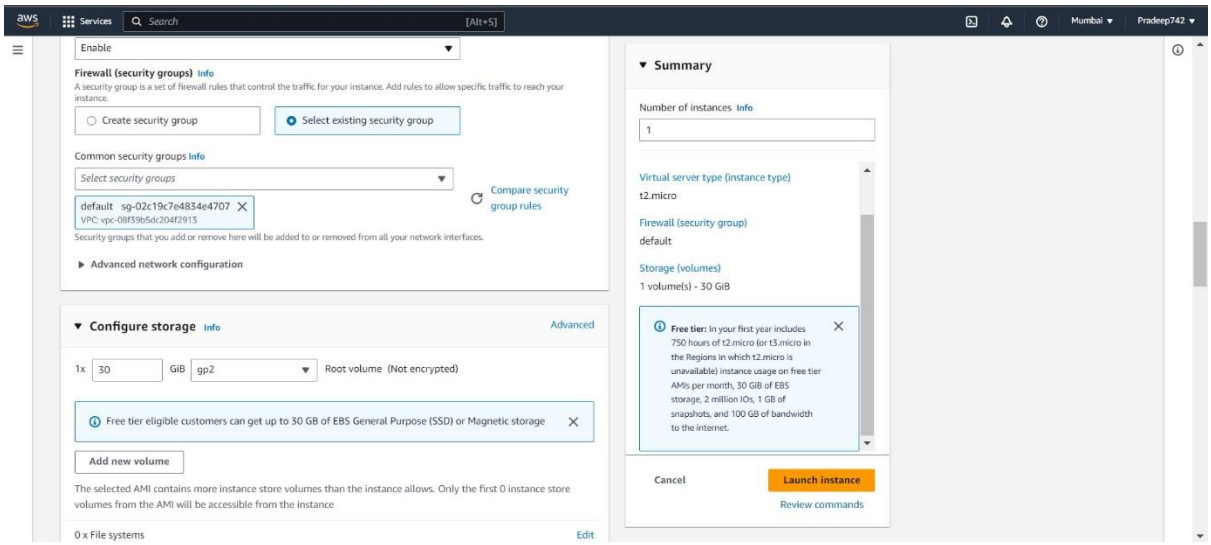
Now Click on Launch Instance then configure the setting according to your requirements.

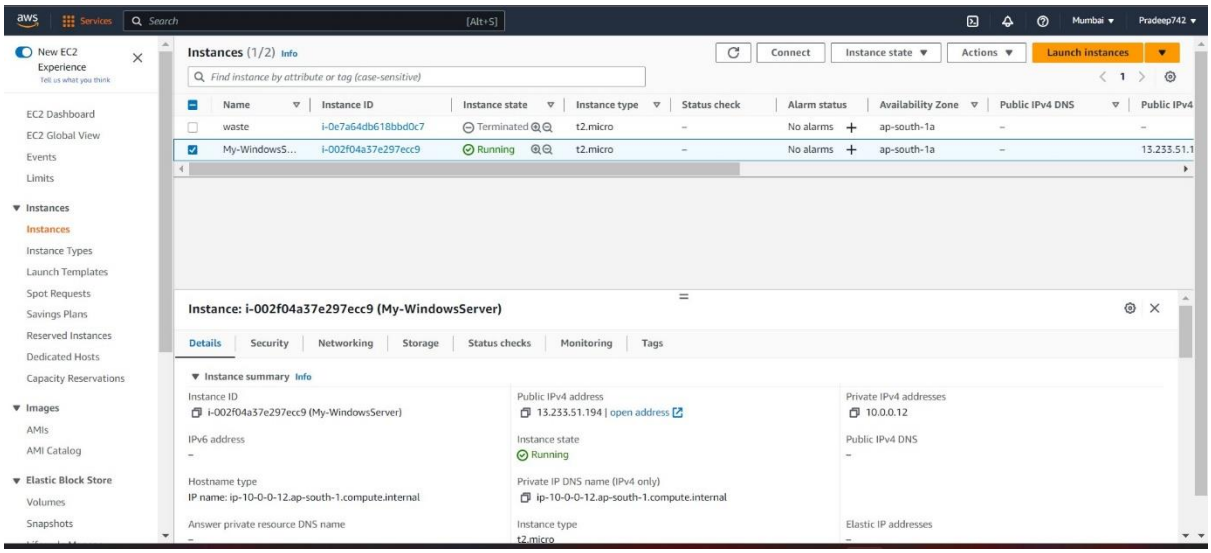
- Choose AMI Amazon Machine Image
- Then select compute like ram storage.
- Then create new key pair or use existing one,
- After that select your VPC and subnet (**my-subnet1)** in network section,
- Configure all settings and then click Launch instance!



I have Created a Windows Server 2022 VM for further implementation.

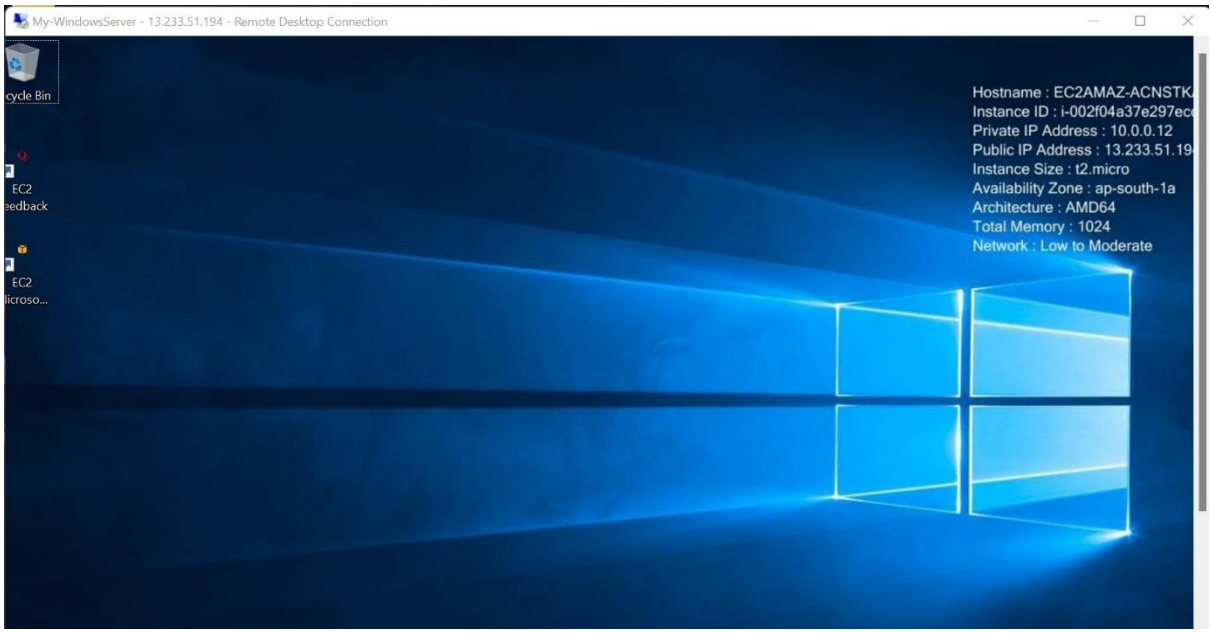

Services
Search
[Alt+S]
Mumbai
Pradeep742
Instance type
t2.micro
Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Linux pricing: 0.0124 USD per Hour
On-Demand Windows pricing: 0.017 USD per Hour
On-Demand RHEL pricing: 0.0724 USD per Hour
On-Demand SUSE pricing: 0.0124 USD per Hour
All generations
Compare instance types
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
Nagarro
Create new key pair
For Windows instances, you use a key pair to decrypt the administrator password. You then use the decrypted password to connect to your instance.
Network settings Info
VPC - required Info
vpc-08f39b5dc204f2913 (my-nagarro-vpc)
10.0.0.0/24
Subnet Info
subnet-025c628015ef5b3c8
my-subnet1
VPC: vpc-08f39b5dc204f2913 Owner: 902155348152
Availability Zone: ap-south-1a IP addresses available: 11 CIDR: 10.0.0.0/28)
Create new subnet
Auto-assign public IP Info
Summary
Number of instances Info
1
Software Image (AMI)
Microsoft Windows Server 2022 ...read more
ami-0b7acb262cc9ea2ea
Virtual server type (instance type)
t2.micro
Firewall (security group)
default
Storage (volumes)
1 volume(s) - 30 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of
Cancel
Launch instance
Review commands


Services
Search
[Alt+S]
Mumbai
Pradeep742
Enable
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
Common security groups Info
Select security groups
default sg-02c19c7e4834e4707
VPC: vpc-08f39b5dc204f2913
Compare security group rules
Security groups that you add or remove here will be added to or removed from all your network interfaces.
Advanced network configuration
Configure storage Info
Advanced
1x 30 GiB gp2 Root volume (Not encrypted)
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage
Add new volume
The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance
0 x File systems
Edit
Summary
Number of instances Info
1
Virtual server type (instance type)
t2.micro
Firewall (security group)
default
Storage (volumes)
1 volume(s) - 30 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.
Cancel
Launch instance
Review commands


Services
Search
[Alt+S]
Mumbai
Pradeep742
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Limits
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Instances (1/2) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Public IPv4
waste
i-0e7a64db618bbd0c7
Terminated
t2.micro
No alarms
ap-south-1a
My-WindowsS...
i-002f04a37e297ecc9
Running
t2.micro
No alarms
ap-south-1a
13.233.51.1
Instance: i-002f04a37e297ecc9 (My-WindowsServer)
Details
Security
Networking
Storage
Status checks
Monitoring
Tags
Instance summary Info
Instance ID
i-002f04a37e297ecc9 (My-WindowsServer)
Public IPv4 address
13.233.51.194 | open address
Private IPv4 addresses
10.0.0.12
IPv6 address
Instance state
Running
Public IPv4 DNS
Hostname type
IP name: ip-10-0-0-12.ap-south-1.compute.internal
Private IP DNS name (IPv4 only)
ip-10-0-0-12.ap-south-1.compute.internal
Answer private resource DNS name
Instance type
t2.micro
Elastic IP addresses

Once The Instance is ready to use you can connect to it
Connect to your Windows Server EC2 instance using a remote desktop client:

- Retrieve the public DNS or public IP address of your EC2 instance from the AWS Management Console.
- Open the Remote Desktop client on your local machine.
- Enter the public DNS or IP address of your EC2 instance and connect using the appropriate credentials.

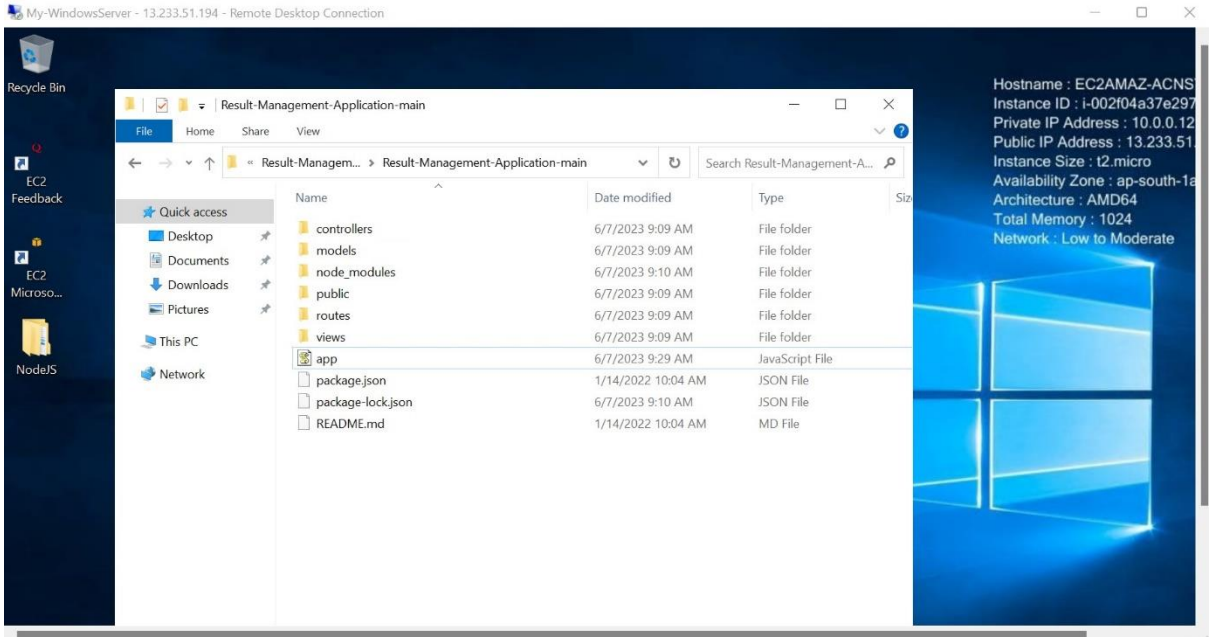


Set up Node.js on your Windows Server:

- Open a web browser on your EC2 instance and go to the official Node.js website: https://nodejs.org/.
- Download the latest LTS version of Node.js for Windows by clicking on the "LTS" button.
- Run the installer once it's downloaded.
- Follow the installation wizard, accepting the default settings. Node.js will be installed on your EC2 instance.

Transfer your Node.js application files to the EC2 instance:

- Choose a method to transfer files to your EC2 instance, such as using an FTP client or the Remote Desktop client's file transfer feature.
- Open the chosen method and connect to your EC2 instance.
- Navigate to a directory on your EC2 instance where you want to store your Node.js application files.
- Transfer the necessary files from your local machine to the EC2 instance.

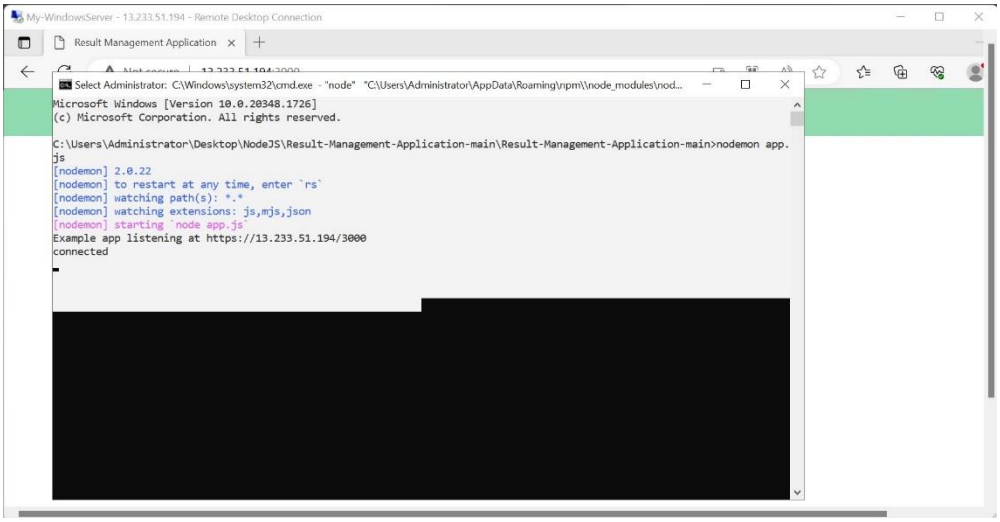


Install dependencies and run your Node.js application:

- Open the Remote Desktop client and connect to your EC2 instance.
- Open a command prompt or PowerShell window on your EC2 instance.
- Navigate to the directory where you transferred your Node.js application files.
- Run the following command to install the required dependencies:
- ***npm install***

Once the dependencies are installed, you can start your Node.js application by running the command:

- ***node app.js***
- Monitor the output and logs:
- As your Node.js application starts, you should see relevant output and any error messages in the command prompt or PowerShell window.
- You can use this information to troubleshoot any issues that may arise.
- Access your Node.js application:
- Open a web browser on your local machine.
- Enter the public DNS or public IP address of your EC2 instance, followed by the appropriate port and route to access your Node.js application.



In the EC2 Dashboard, click on "**Security Groups**" from the left navigation pane.

- Locate the security group associated with your EC2 instance. You can identify it by checking the "Security Group Name" and "Description" columns.

- Select the desired security group by clicking on its name.

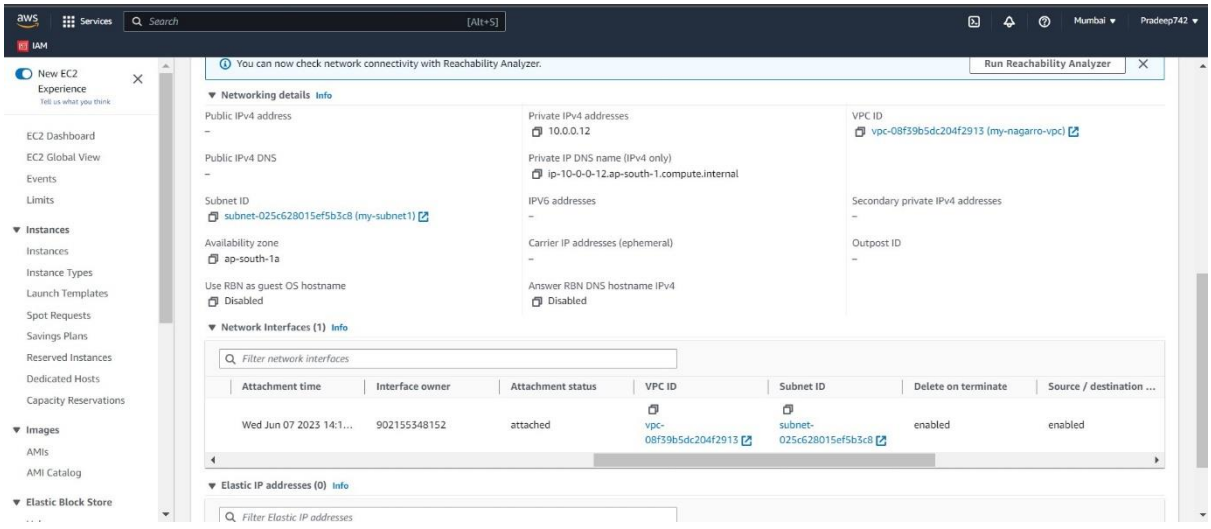
In the "Inbound Rules" tab, you will see the existing inbound rules for the selected security group.

- Click on the "Edit inbound rules" button.

In the "Edit inbound rules" dialog box, click on the "Add rule" button.

- Configure the inbound rule for the new port:
- Enter the port number or port range in the "Port range" field.
- Specify the source of the inbound traffic in the "Source" field. You can use various options such as "Custom," "My IP," or specific IP ranges.
- Provide a description for the rule in the "Description" field (optional).
- Click on the "Save rules" button to apply the changes.

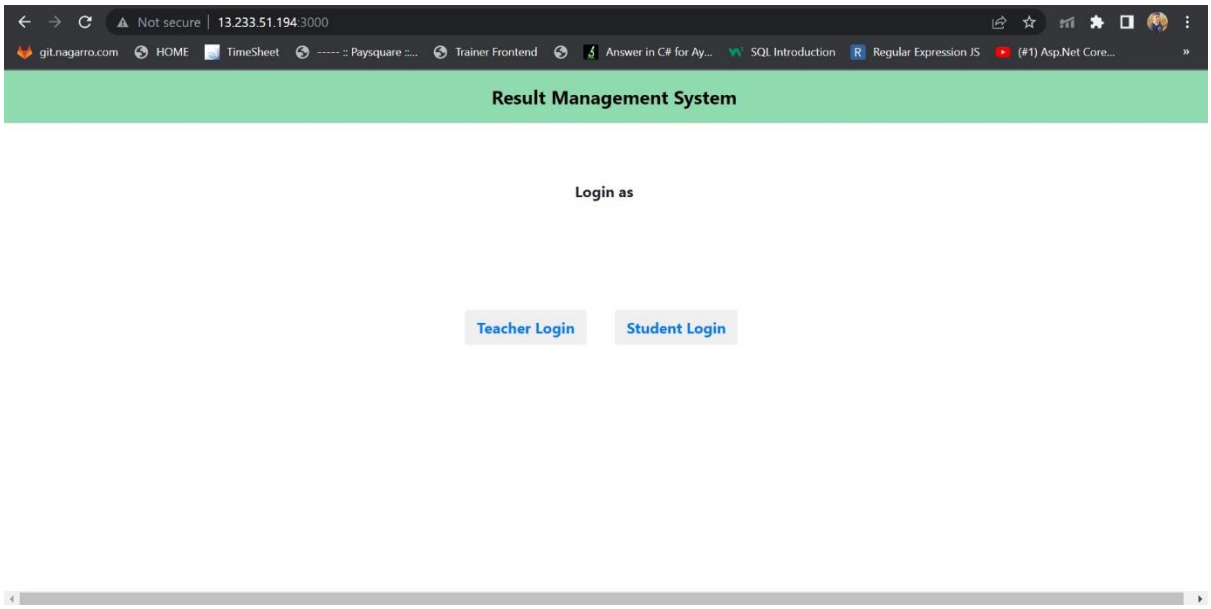
The new inbound rule with the specified port will be added to the security group. This allows incoming traffic on the specified port to reach your EC2 instance. Make sure to configure the rule with the appropriate protocol, port number, and source to meet your application's requirements and security considerations.

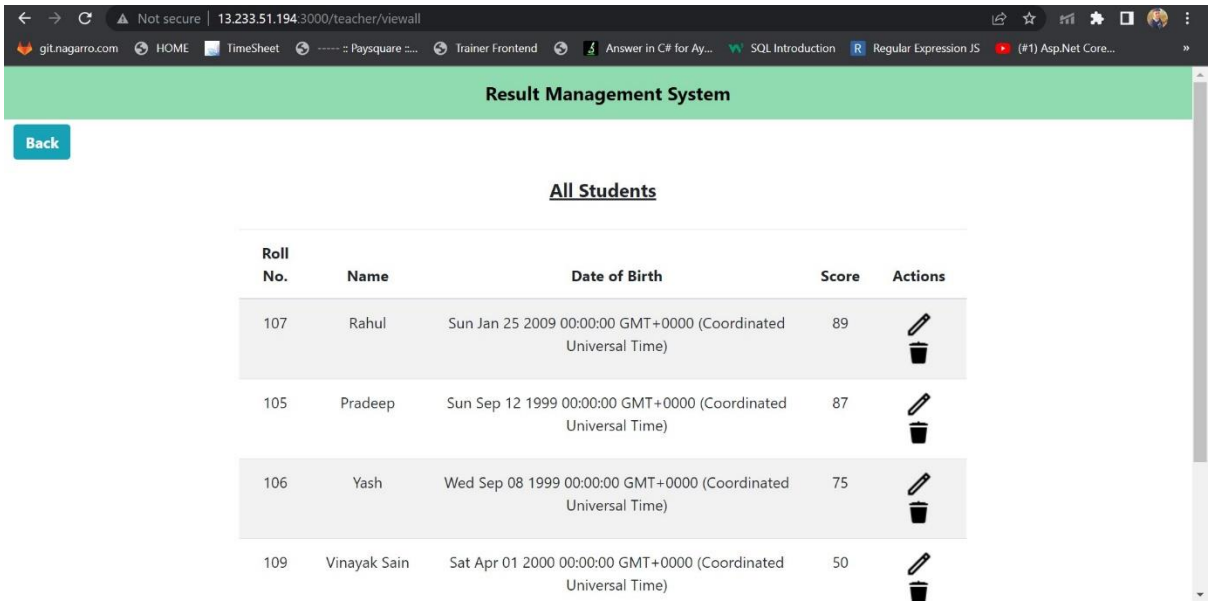


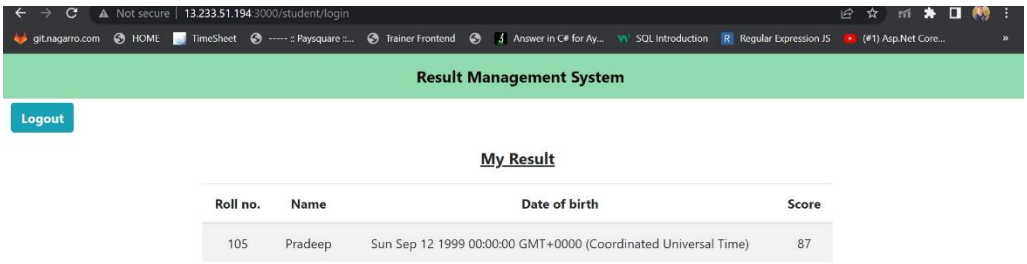
Ensure that your Node.js application is binding to the correct IP and port:

- Open your Node.js application code and check that it explicitly binds to the correct IP address and port.
- If your application is using Express.js, ensure that the server is listening on the desired IP and port using the **app.listen**() method.

- Open a web browser on your local machine.
- Enter the public IP address of your EC2 instance, followed by the port number and any applicable routes or paths, to access your Node.js application.
- For example, if your EC2 instance's public IP address is **13.233.51.194** and your Node.js application is listening on port **3000**, enter http:// 13.233.51.194:3000 in the web browser.





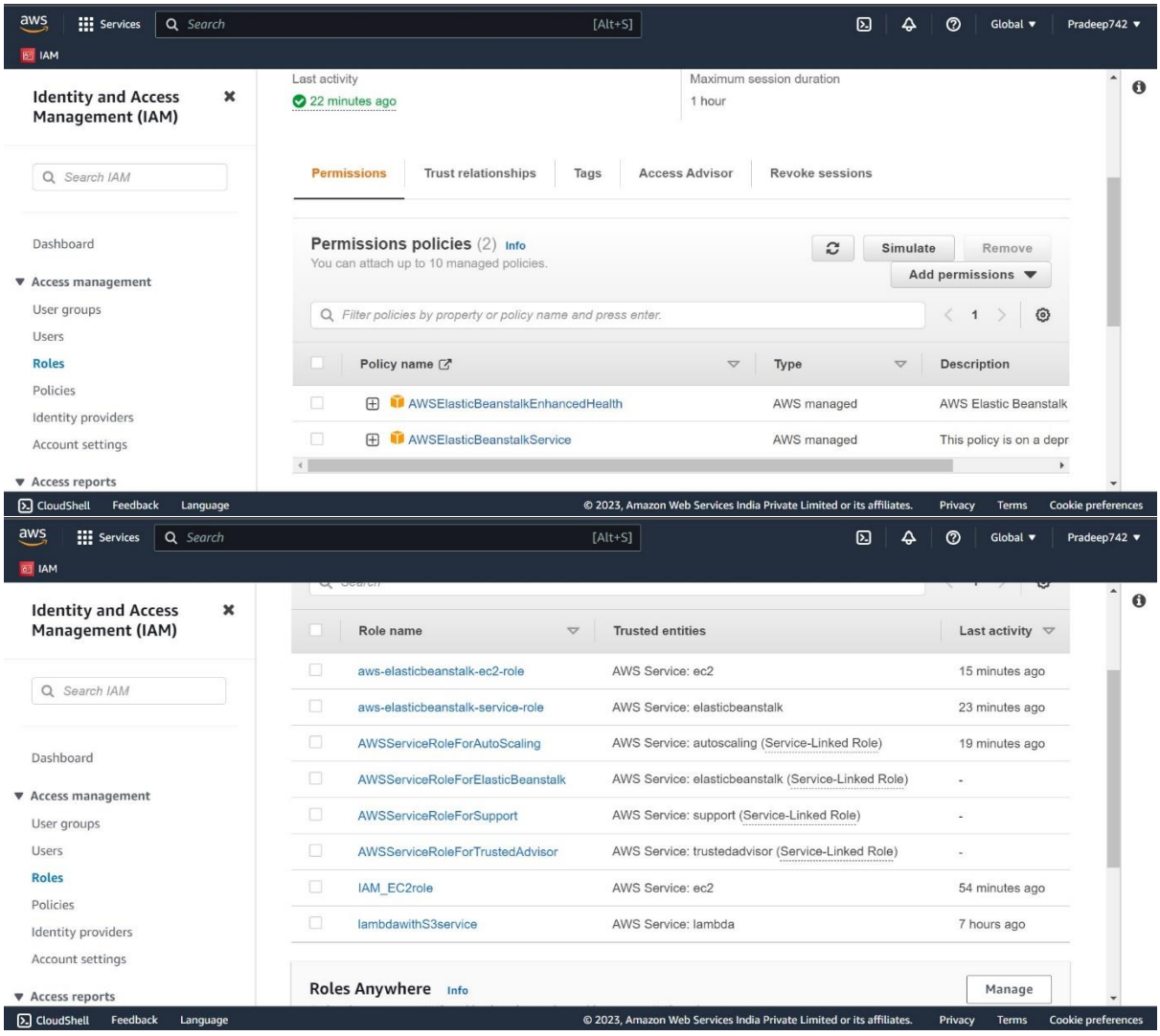


Q3. Deploy the same application to Elastic beanstalk Service.

Elastic Beanstalk is a fully managed service provided by AWS that makes it easier to deploy, run, and scale web applications and services. It abstracts the underlying infrastructure and handles the deployment and management tasks, allowing developers to focus on their application code.

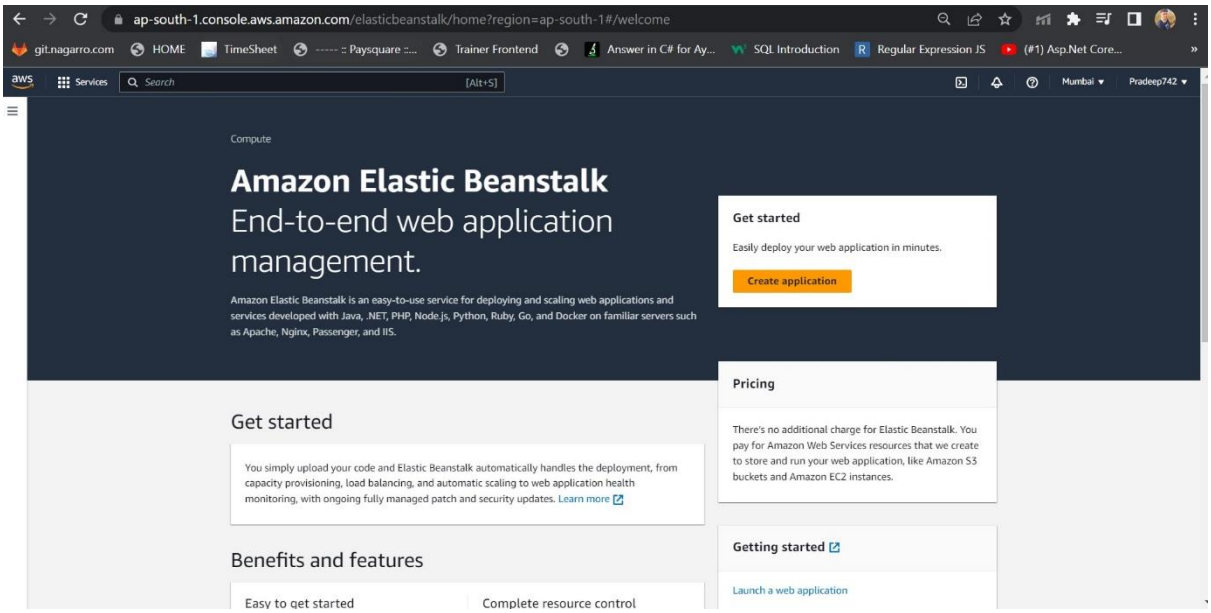
Using Elastic Beanstalk, you can quickly deploy and manage your web applications without worrying about the underlying infrastructure. It provides a simplified and streamlined deployment experience, enabling developers to focus on writing code and delivering their applications faster.

If we have already created the required roles and services like go to IAM and then select roles and create roles for Elastic Beanstalk service and give necessary permissions
eg: create a role called aws-elasticbeanstalk-ec2-role with the following policies:
AWSElasticBeanstalkWorkerTier and AWSElasticBeanstalkMulticontainerDocker

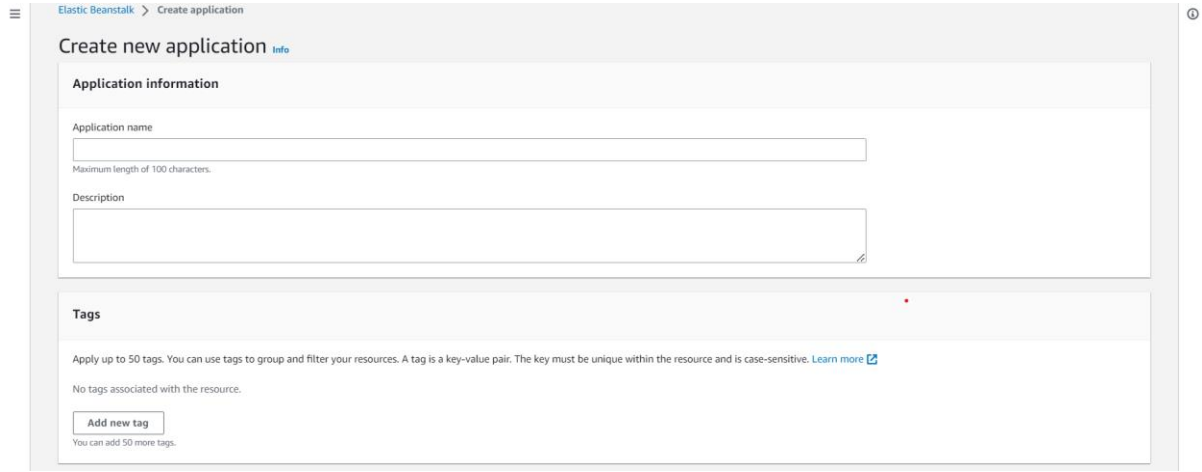


Step 1: Create an Elastic Beanstalk Environment

Go to the AWS Management Console and navigate to the Elastic Beanstalk service.



- Click on "Create Application



Fillip the required details give your application name select tags if required….

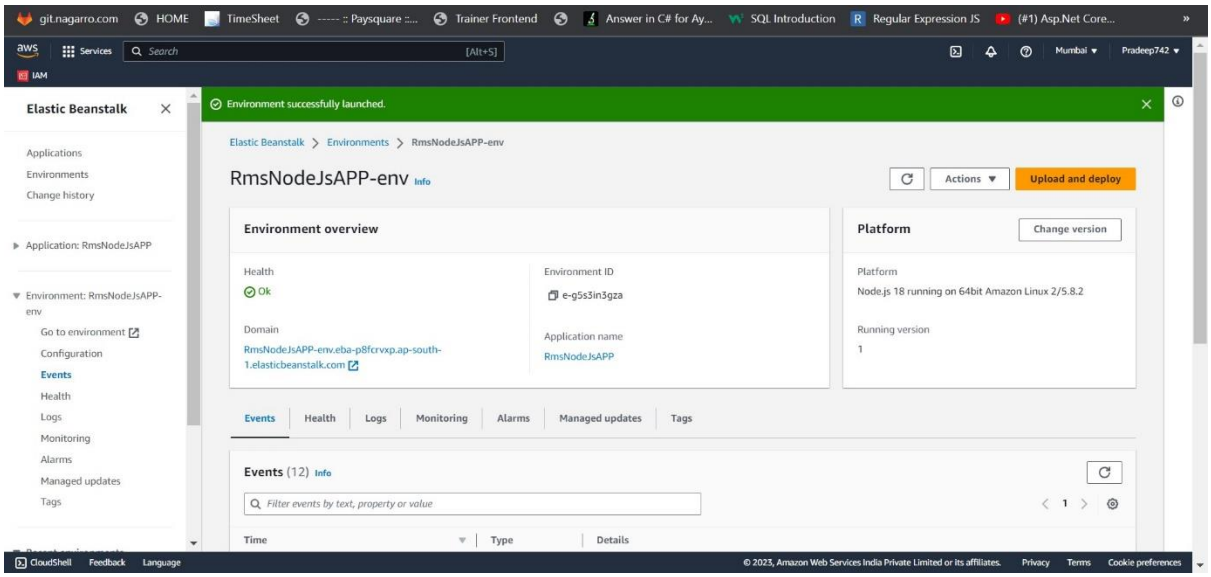
- Click on "Create environment."

- Select "Web server environment" as the environment type.

- Choose a name for your environment and provide a brief description.
- Under "Platform," select "Node.js" as the platform. Choose the desired application stack version.

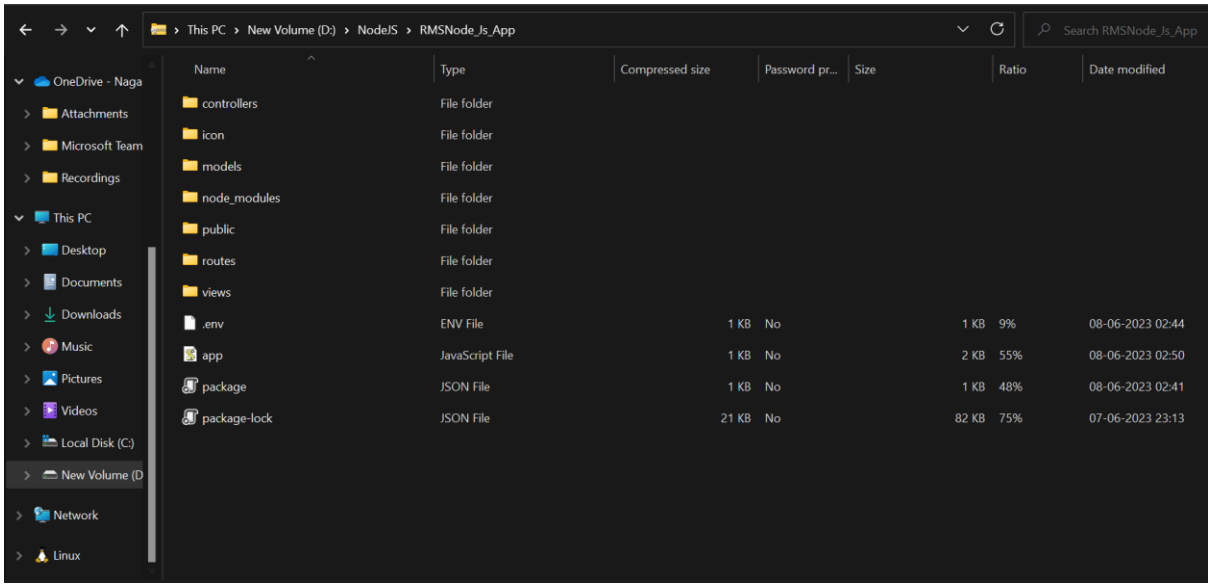
- Click on "Configure more options" if you want to customize your environment further. You can modify settings like instance type, scaling options, security groups, etc.

- Click on "Create environment" to create your environment. Elastic Beanstalk will now start provisioning the necessary resources.



Step 2: Prepare your Node.js Application

Assuming you have a Node.js application codebase ready, follow these steps to prepare it for deployment:



- Open a terminal or command prompt and navigate to the root directory of your Node.js application.

- Make sure your application has a valid package.json file. If it doesn't exist, create one using the **npm init** command and provide the necessary details.

- Install all the required dependencies for your application using the npm install command. This will populate the **node modules** directory with the necessary packages.

- Exclude any unnecessary files or directories from your application codebase. For example, you can exclude development-specific files, test files, or any other files not needed for the production environment.

- Create a ZIP archive of your application code, including the package.json file and the **node modules** directory. Make sure the root of the ZIP archive contains all the necessary files and folders.

Step 3: Deploy your Application.

- Once you have prepared your Node.js application code, follow these steps to deploy it to Elastic Beanstalk:

- In the Elastic Beanstalk console, select your environment from the list of environments.

- Under the "Application versions" section, click on "Upload and deploy."

- Click on "Choose file" and select the ZIP archive of your application code that you created in the previous step.

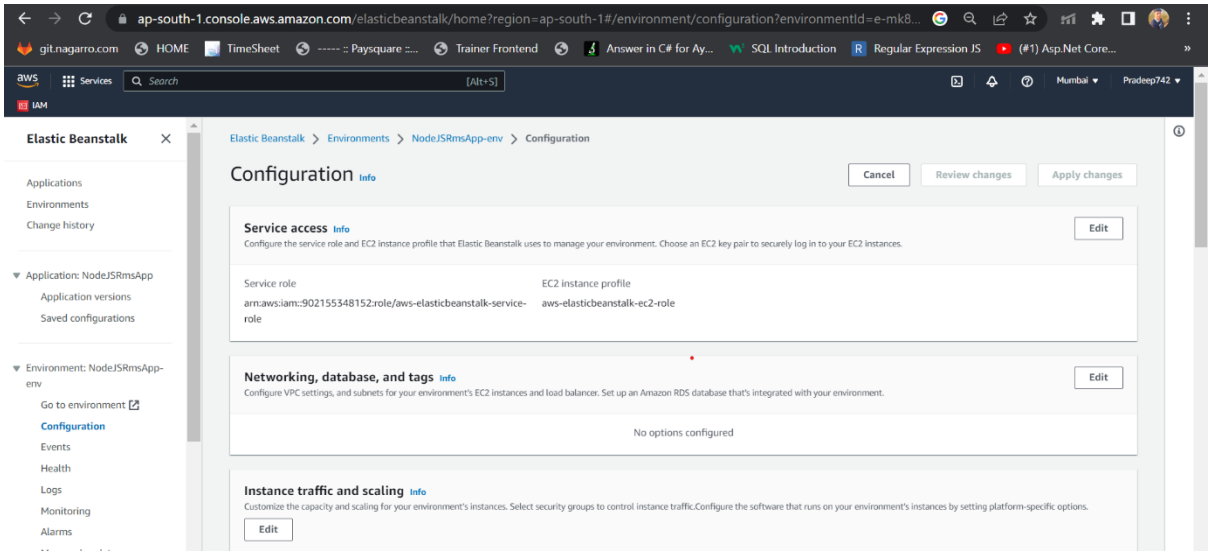
- Once the upload is complete, click on "Deploy" to deploy your application to the environment.

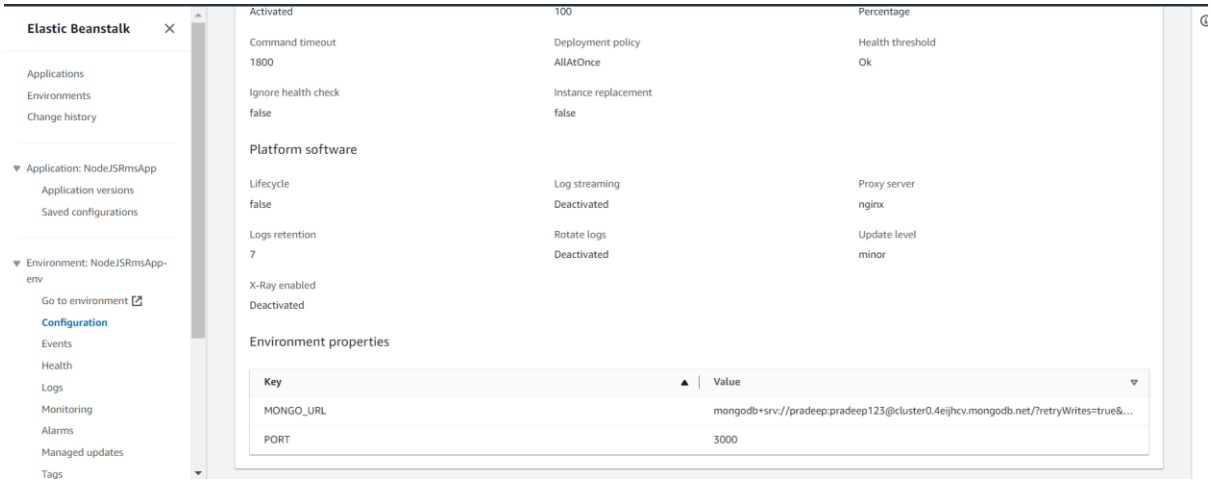
- Elastic Beanstalk will start the deployment process, which may take a few minutes. You can monitor the progress in the Elastic Beanstalk console.

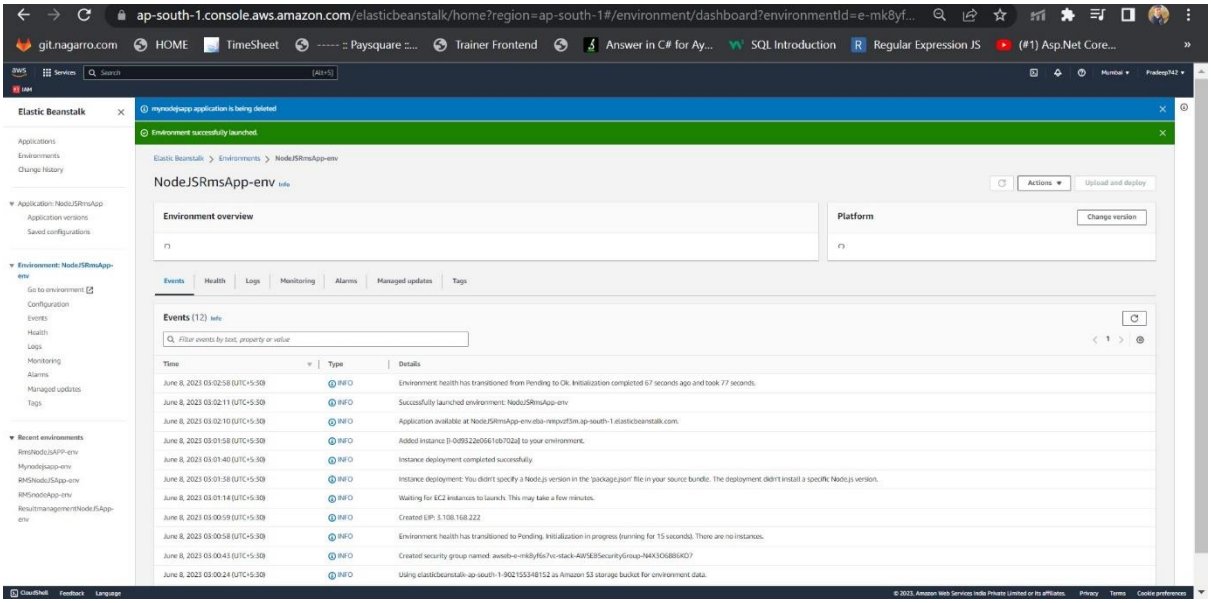
Step 4: Configure Environment Variables

- If your application requires environment variables, you can set them in Elastic Beanstalk. Follow these steps to configure environment variables:
- In the Elastic Beanstalk console, select your environment.
- Click on "**Configuration**" in the left-hand menu.
- Under the "Software" section, click on "Edit" next to "**Environment properties**."
- Add the necessary environment variables for your application. For example, you can add a variable named **DB_HOST** with the value of your database host.

Click on "Apply" to save the changes.


ap-south-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-south-1#/environment/configuration?environmentId=e-mk8...
Elastic Beanstalk
Applications
Environments
Change history
Application: NodeJSRmsApp
Application versions
Saved configurations
Environment: NodeJSRmsApp-env
Go to environment
Configuration
Events
Health
Logs
Monitoring
Alarms
Elastic Beanstalk > Environments > NodeJSRmsApp-env > Configuration
Configuration Info
Cancel
Review changes
Apply changes
Service access Info
Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.
Edit
Service role
arn:aws:iam::902155348152:role/aws-elasticbeanstalk-service-role
EC2 instance profile
aws-elasticbeanstalk-ec2-role
Networking, database, and tags Info
Configure VPC settings, and subnets for your environment's EC2 instances and load balancer. Set up an Amazon RDS database that's integrated with your environment.
Edit
No options configured
Instance traffic and scaling Info
Customize the capacity and scaling for your environment's instances. Select security groups to control instance traffic.Configure the software that runs on your environment's instances by setting platform-specific options.
Edit

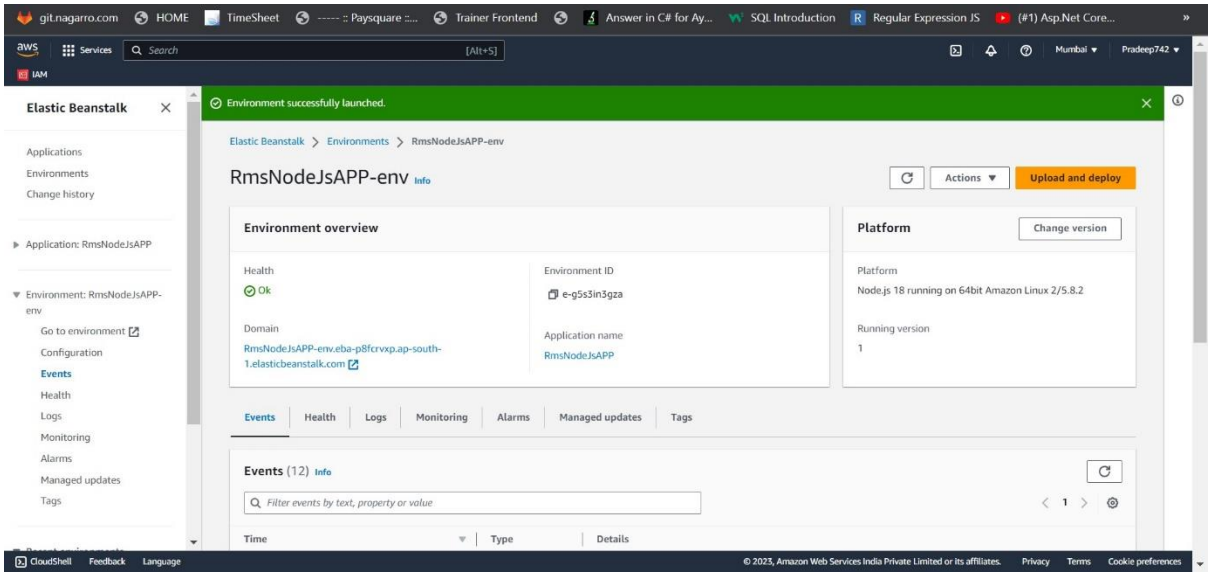

Elastic Beanstalk
Activated
100
Percentage
Command timeout
1800
Deployment policy
AllAtOnce
Health threshold
Ok
Ignore health check
false
Instance replacement
false
Platform software
Lifecycle
false
Log streaming
Deactivated
Proxy server
nginx
Logs retention
7
Rotate logs
Deactivated
Update level
minor
X-Ray enabled
Deactivated
Environment properties
Key
Value
MONGO_URL
mongodb+srv://pradeep:pradeep123@cluster0.4eijhcv.mongodb.net/?retryWrites=true&...
PORT
3000

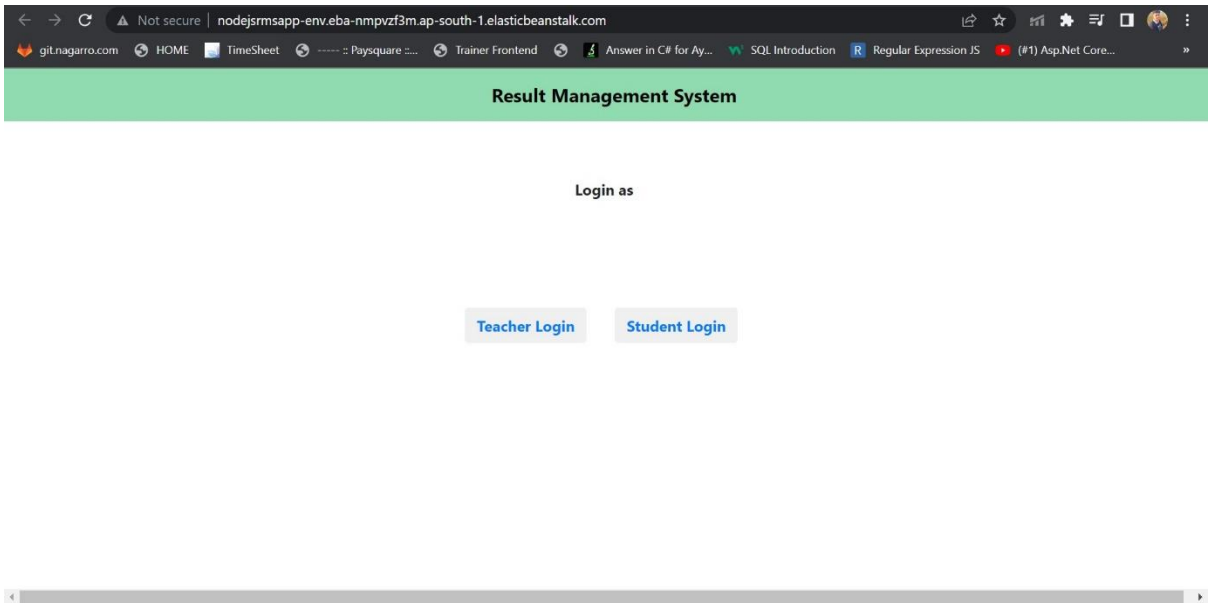

ap-south-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-south-1#/environment/dashboard?environmentId=e-mk8yf...
Elastic Beanstalk
mynodejsapp application is being deleted
Environment successfully launched.
NodeJSRmsApp-env
Actions
Environment overview
Platform
Change version
Events
Health
Logs
Monitoring
Alarms
Managed updates
Tags
Events (12)
Time
Type
Details

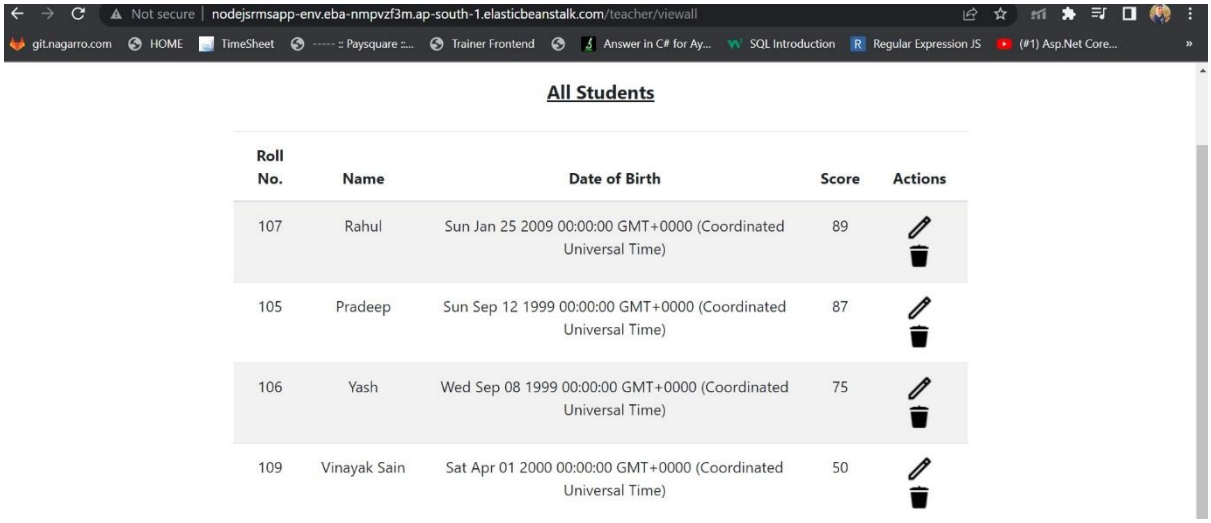
Step 5: Monitor and Access your Application

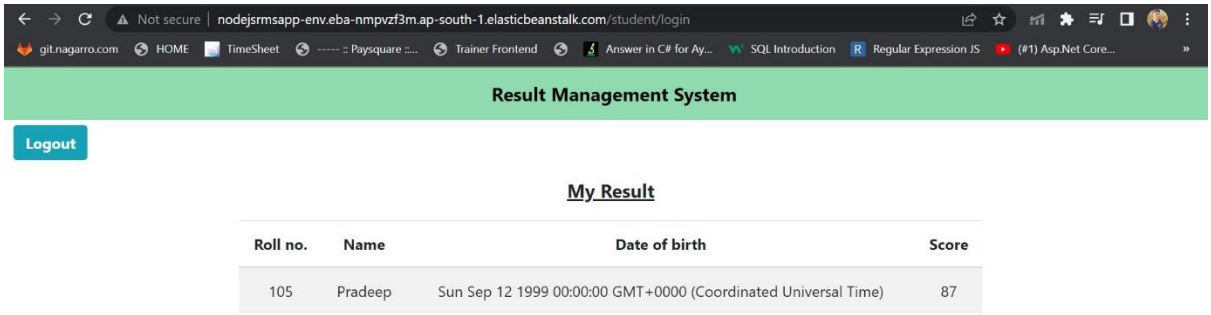
Once the deployment is complete, you can monitor the deployment progress in the Elastic Beanstalk console. After the environment is ready, you can access your Node.js application using the provided URL.

For example, if your environment URL ***http://nodejsrmsapp-env.eba-nmpvzf3m.ap-south-1.elasticbeanstalk.com/***open this URL in a web browser to access your application.









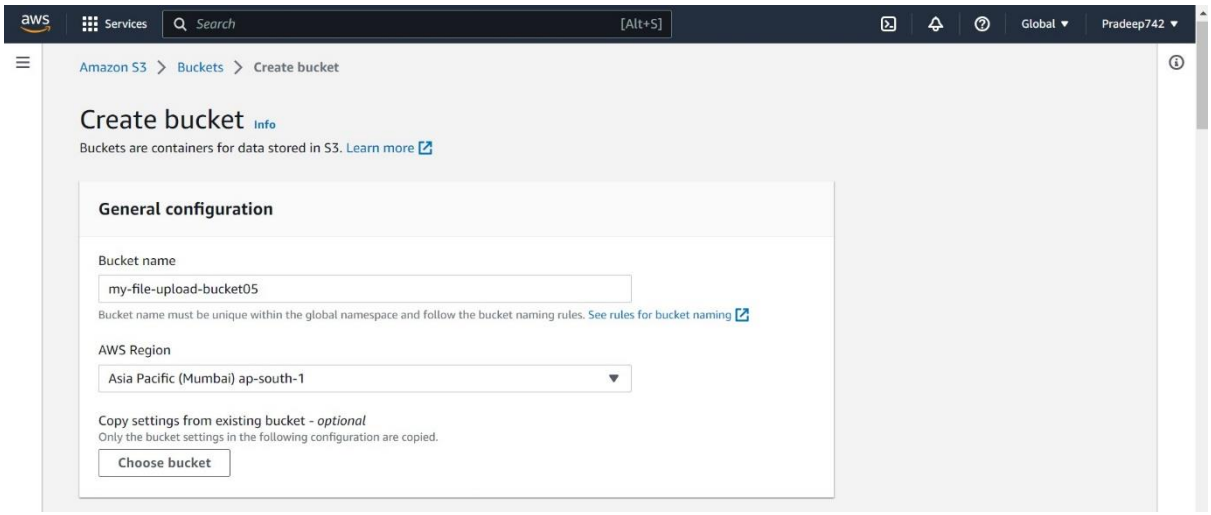
That's it! Your Node.js application is now deployed and running on Elastic Beanstalk in the AWS cloud. Elastic Beanstalk handles the underlying infrastructure, such as EC2 instances and load balancing, allowing you to focus.

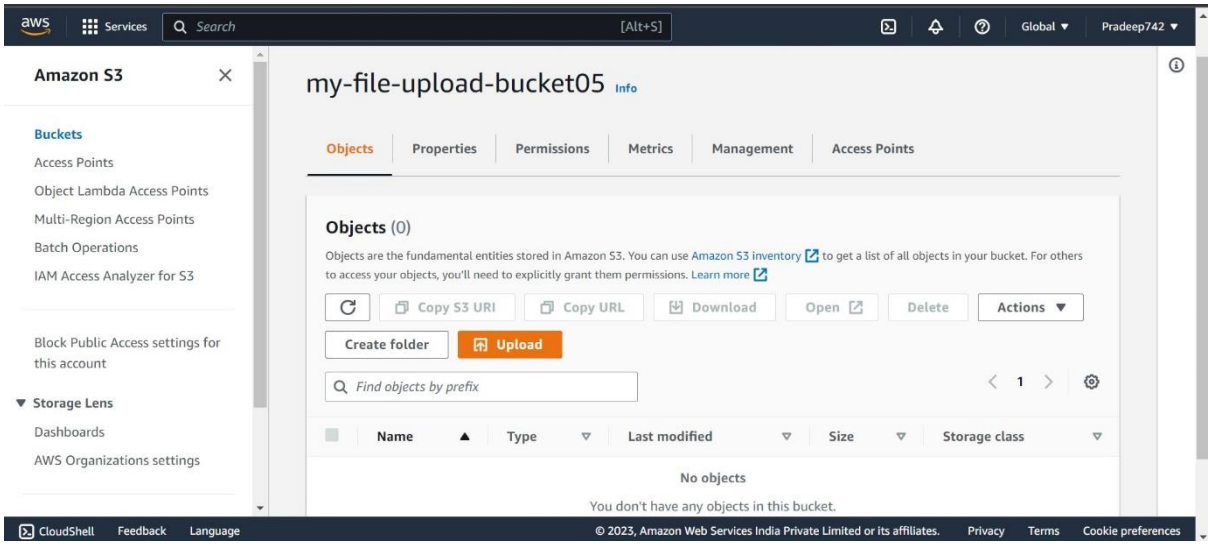
Q4 Create a Lambda that should trigger as soon as you upload a file in the S3 bucket. Function should be able to print the name of the file uploaded in the function.

If you haven't created an S3 bucket yet, here's an additional step to create an S3 bucket before proceeding with the Lambda function setup:

- Create an S3 bucket:
- Log in to the AWS Management Console and navigate to the S3 service.
- Click on the "Create bucket" button.

- Provide a unique name for your bucket, e.g., "**my-file-upload-bucket05**".
- Choose the region where you want to create the bucket.
- Leave the other settings as default or configure them according to your requirements.
- Click on the "Create" button to create the bucket.

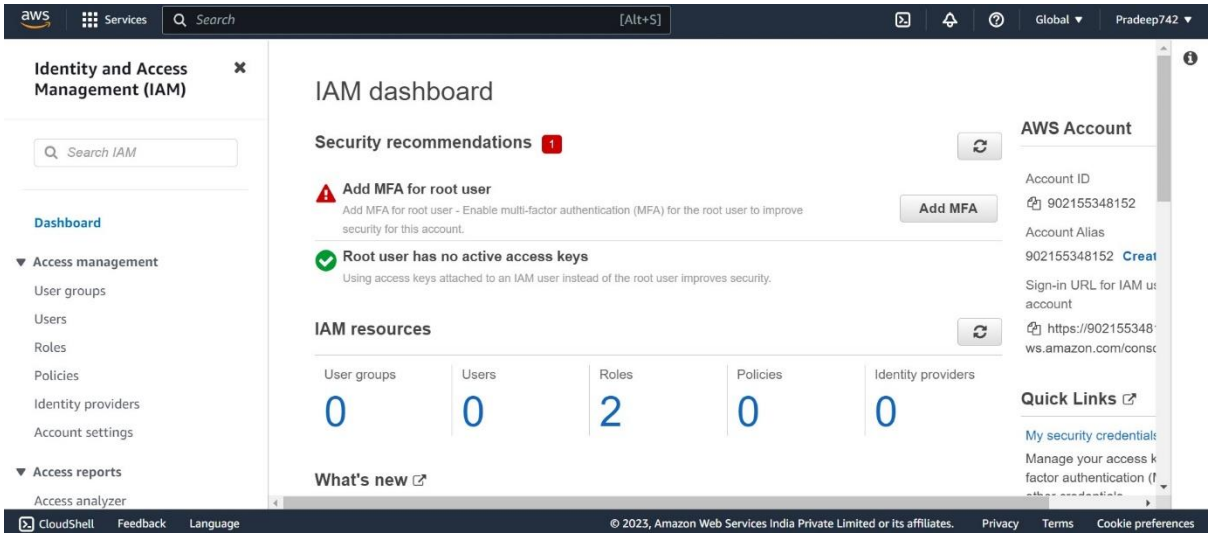


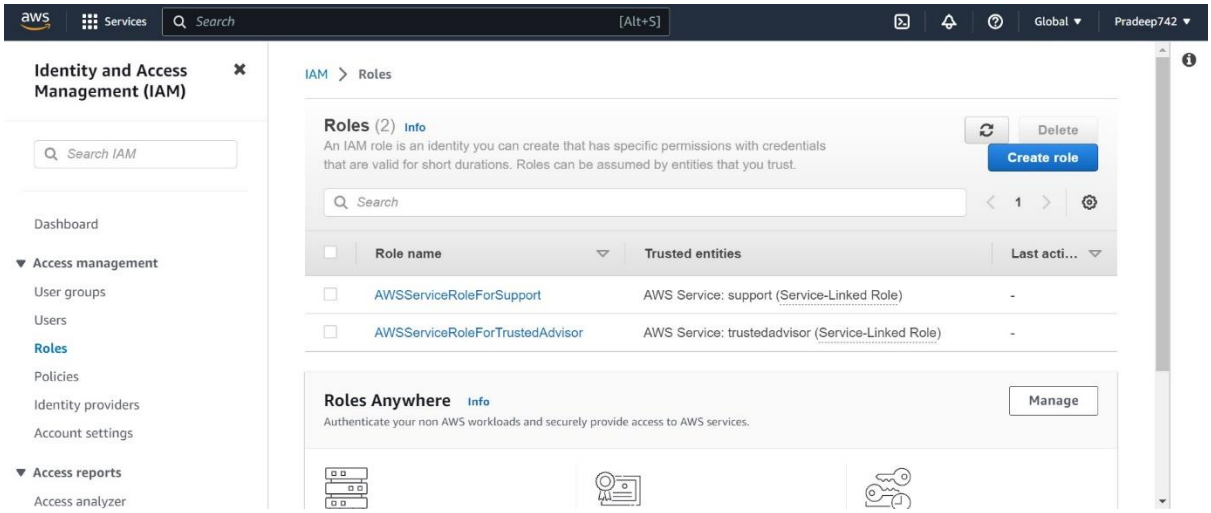


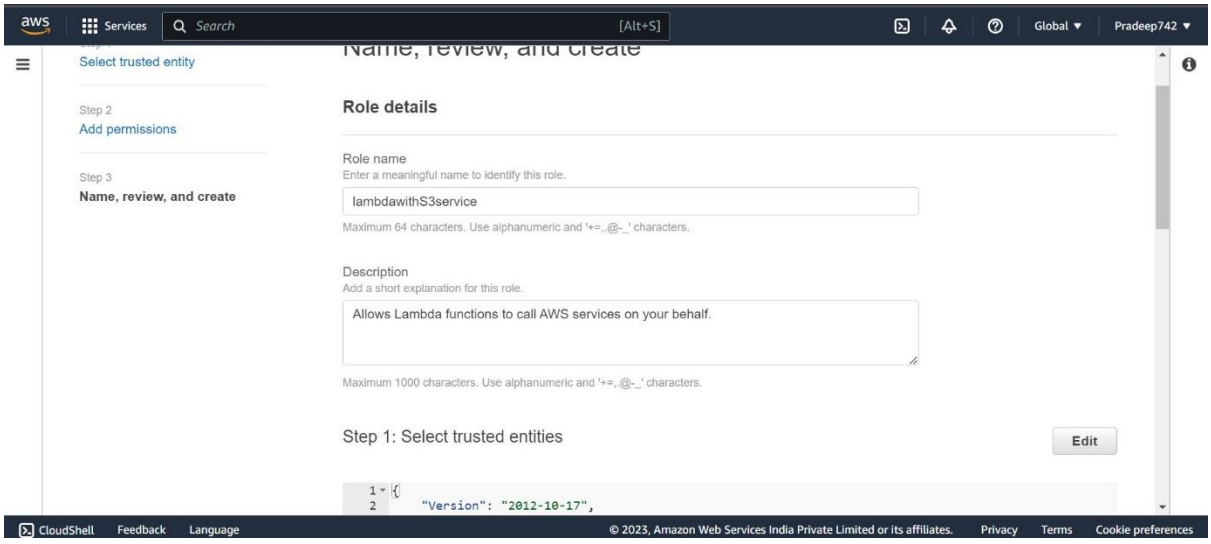
Once you have created the S3 bucket, you can proceed with the steps mentioned earlier to create the Lambda function that triggers when a file is uploaded to the S3 bucket and prints the name of the uploaded file.

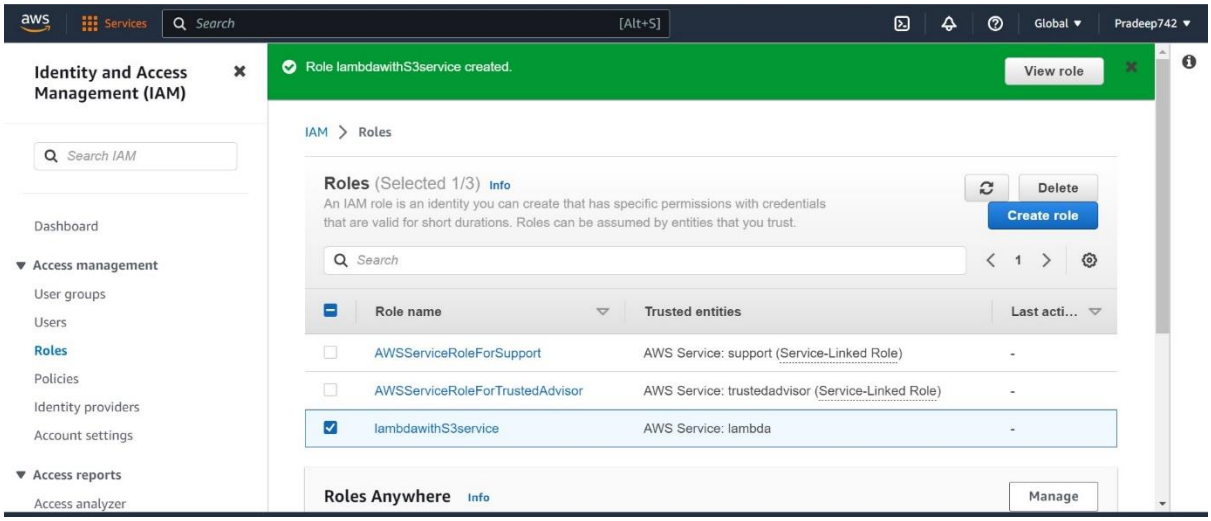
Then Create a IAM role to which allows Lambda function to call AWS Service

Go to IAM dashboard in AWS services!


Services
Search
[Alt+S]
Global
Pradeep742
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access analyzer
IAM dashboard
Security recommendations 1
Add MFA for root user
Add MFA for root user - Enable multi-factor authentication (MFA) for the root user to improve security for this account.
Add MFA
Root user has no active access keys
Using access keys attached to an IAM user instead of the root user improves security.
IAM resources
User groups 0
Users 0
Roles 2
Policies 0
Identity providers 0
What's new
AWS Account
Account ID
902155348152
Account Alias
902155348152
Quick Links
My security credentials
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search
[Alt+S]
Global
Pradeep742
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access analyzer
IAM > Roles
Roles (2) Info
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.
Delete
Create role
Search
Role name
Trusted entities
Last acti...
AWSServiceRoleForSupport
AWS Service: support (Service-Linked Role)
AWSServiceRoleForTrustedAdvisor
AWS Service: trustedadvisor (Service-Linked Role)
Roles Anywhere Info
Authenticate your non AWS workloads and securely provide access to AWS services.
Manage


Services
Search
[Alt+S]
Global
Pradeep742
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
lambdawithS3service
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this role.
Allows Lambda functions to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Step 1: Select trusted entities
Edit
{
"Version": "2012-10-17",
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

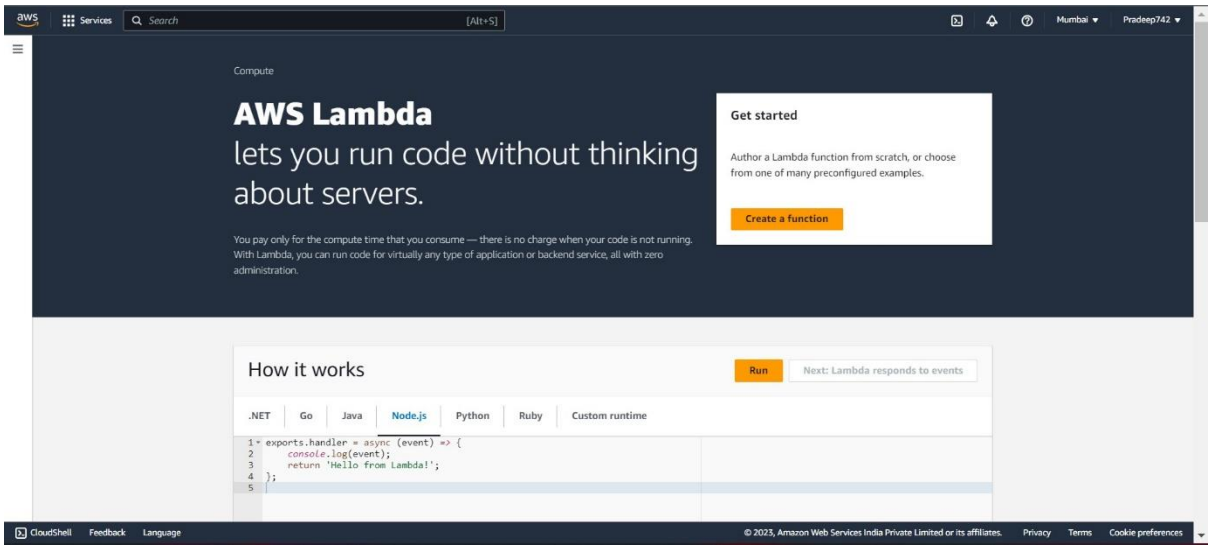


NOW

To create a Lambda function that triggers when a file is uploaded to an S3 bucket and prints the name of the uploaded file:

Step 1: Log in to the AWS Management Console and navigate to the AWS Lambda service.

Step 2: Click on the "Create function" button.

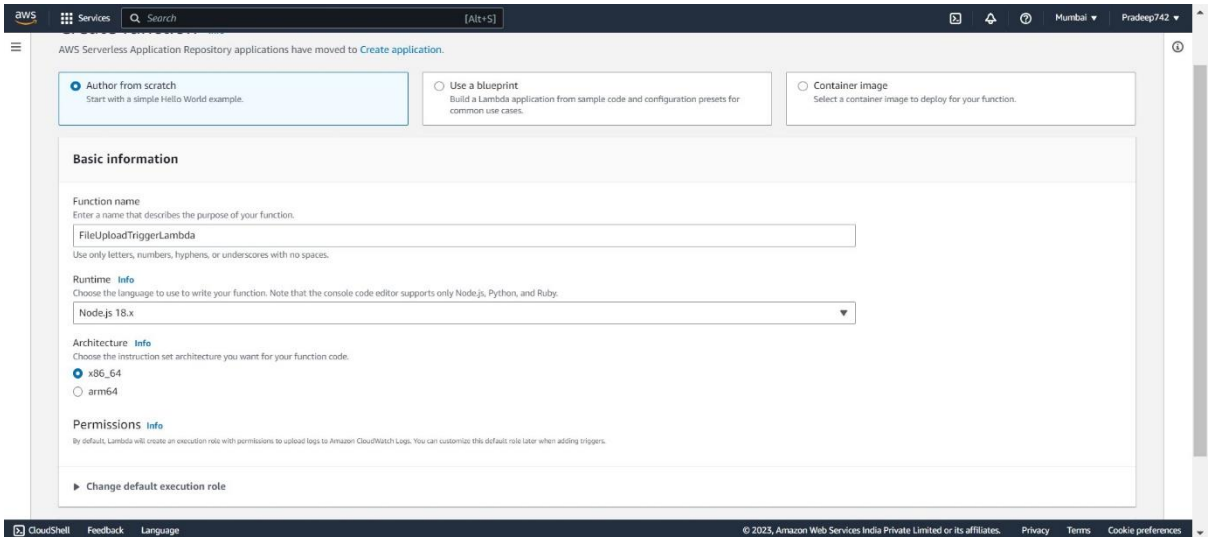


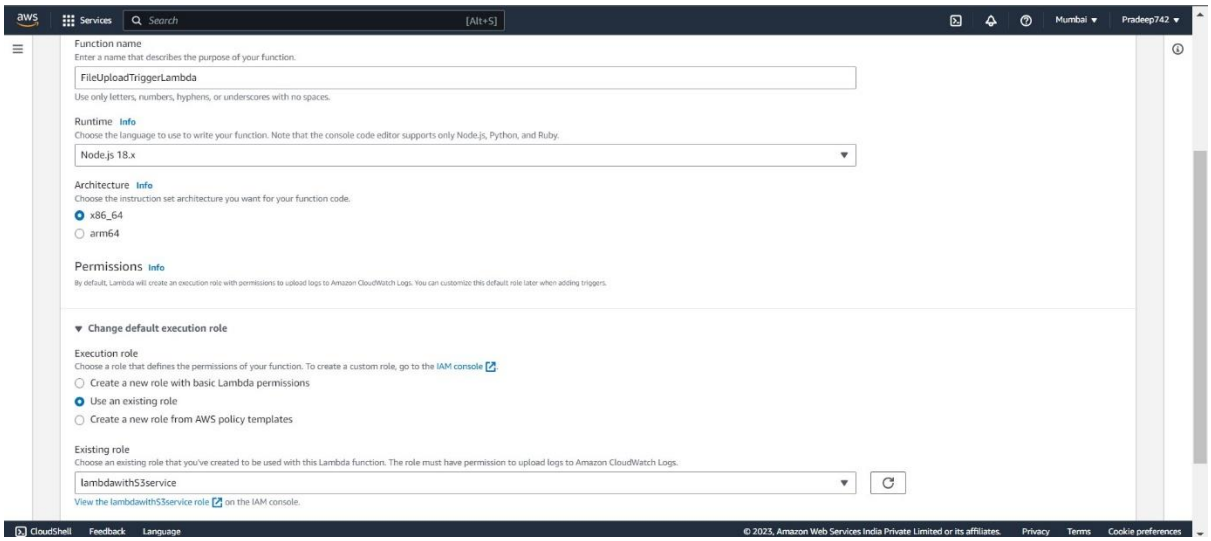
Step 3: Choose a blueprint (optional):

In this example, let's skip choosing a blueprint and create a function from scratch.
Step 4: Configure the function:

- Function name: Enter a name for your Lambda function, e.g., "**FileUploadTriggerLambda**".
- Runtime: Select "**Node.js 14.x**" from the available runtime options.

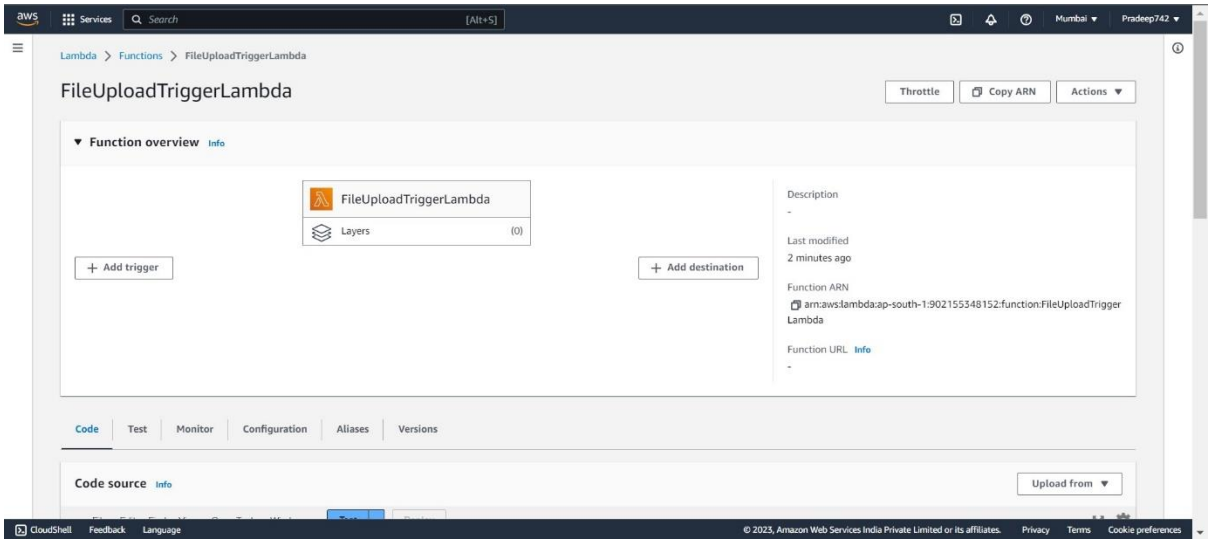
- Permissions: Choose an existing or create a new execution role that has the necessary permissions to access S3. Let's assume you create a new role called "**S3FileUploadRole**" with the required permissions.





Step 5: Configure the trigger:

- Click on "**Add trigger**".
- Service: Select "S3" from the list of services.
- Bucket: Choose the S3 bucket that you want to monitor for file uploads, e.g., "**my-file-upload-bucket**".
- Event type: Select "All object create events" to trigger the Lambda function whenever a file is uploaded to the bucket.
- Prefix and Suffix (optional): You can leave these fields empty for this example.
- Enable trigger: Leave this option enabled.
- Click on "**Add**" to add the trigger.



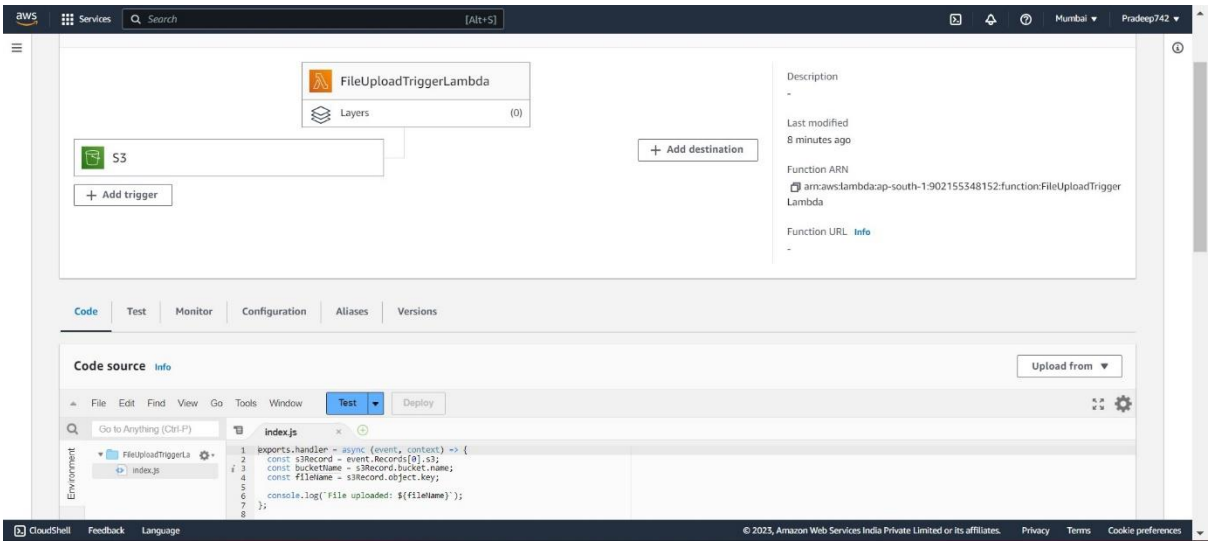
Step 6: Write the Lambda function code:

In the Function code section, replace the existing code with the following Node.js code:

```javascript
exports.handler = async (event, context) => {
  const s3Record = event.Records[0].s3;
  const bucketName = s3Record.bucket.name;
  const fileName = s3Record.object.key;

  console.log(`File uploaded: ${fileName}`);
};
```

- This code uses the AWS Lambda handler function and extracts the bucket name and file name from the event triggered by the S3 upload. It then prints the file name to the Lambda function's logs using **console.log ()**



Step 7: Configure the function settings:

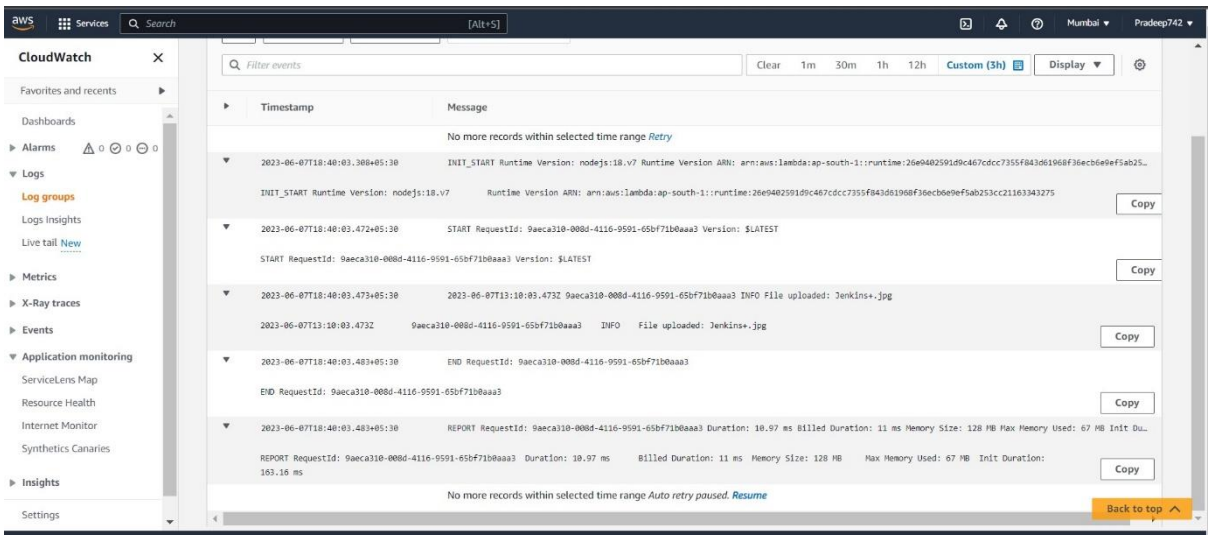
- You can keep the default values for the remaining configuration options or modify them as per your requirements.

Step 8: Save the Lambda function:

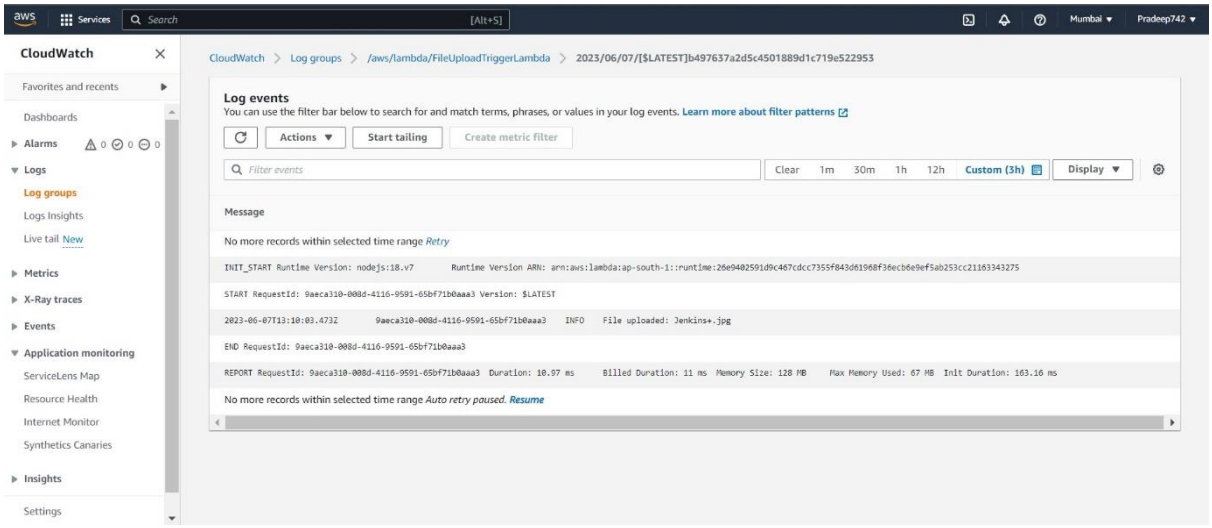
- Click on the "**Save**" button in the upper-right corner of the page.

Step 9: Test the Lambda function:

- Go to the S3 service in the AWS Management Console and select the bucket you configured for the trigger, e.g., "**my-file-upload-bucket**".
- Upload a file to the bucket, either by using the AWS Management Console or programmatically.
- Once the file upload is complete, go to the Lambda function's monitoring tab or logs to view the log output.
- You should see a log message similar to: File uploaded: my-uploaded-file.txt indicating the name of the uploaded file.



That's it! You have now created a Lambda function that triggers when a file is uploaded to the specified S3 bucket and prints the name of the uploaded file.



All the Mention AWS Queries is Implement & performed by me;
If you have any confusion or doubts regarding AWS queries, feel free to contact me at pradeepsajnani742@gmail.com. I am dedicated to providing reliable support and ensuring your AWS implementations are successful. Thank you.

----------**Thank You**----------