

AWS CLOUD ASSIGNMENT

Bench Training Module

Sagar Khatri

Associate Software Engineer
sagar.khatri@nagarro.com

Submitted on
7th June '23

Assignment: AWS Cloud Assignment

Step-by-step Procedure

Open the link for AWS Management Console: <https://aws.amazon.com/console/>

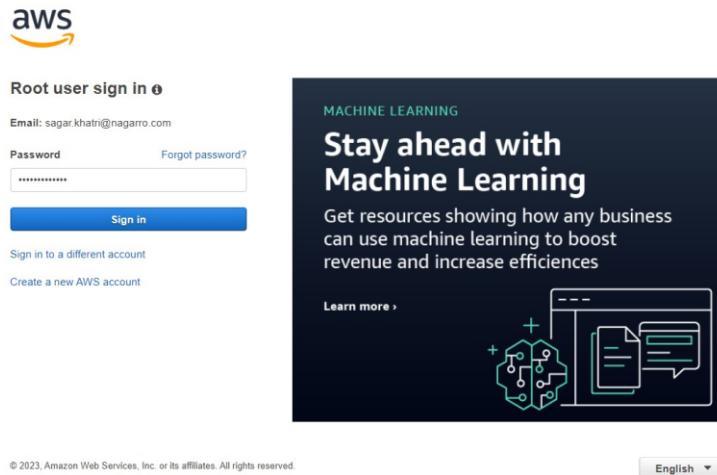
The screenshot shows the AWS Management Console homepage. At the top, there's a navigation bar with links for Products, Solutions, Pricing, Documentation, Learn, Partner Network, AWS Marketplace, Customer Enablement, Events, Explore More, and a search icon. On the far right of the top bar are Contact Us, Support, English, My Account, and a yellow "Sign In to the Console" button. Below the navigation is a banner with the text "AWS Management Console" and "Everything you need to access and manage the AWS Cloud — in one web interface". A large orange "Log back in" button is centered below the banner. The main content area features several cards with icons and descriptions:

- Free AWS Training**: Advance your career with AWS Cloud Practitioner Essentials—a free, six-hour, foundational course.
- AWS Machine Learning Training**: Choose from courses that cover the entire machine learning pipeline.
- AWS Certification**: Propel your career forward with AWS Certification.
- AWS Free Digital Training**: Access 350+ free digital courses with training built by AWS experts.
- Build Mobile and Web Apps Fast**: Add authentication and data syncing with AWS Amplify in just a few lines of code.
- Amazon EC2 Supports macOS Big Sur**: Build and run on-demand Apple workloads on AWS, the only major cloud provider to offer macOS.
- AWS Lambda Container Image Support**: Build and deploy Lambda based applications by using your favorite.
- Introducing Amazon EKS Anywhere**: Create and operate Kubernetes clusters on your on-premises infrastructure.

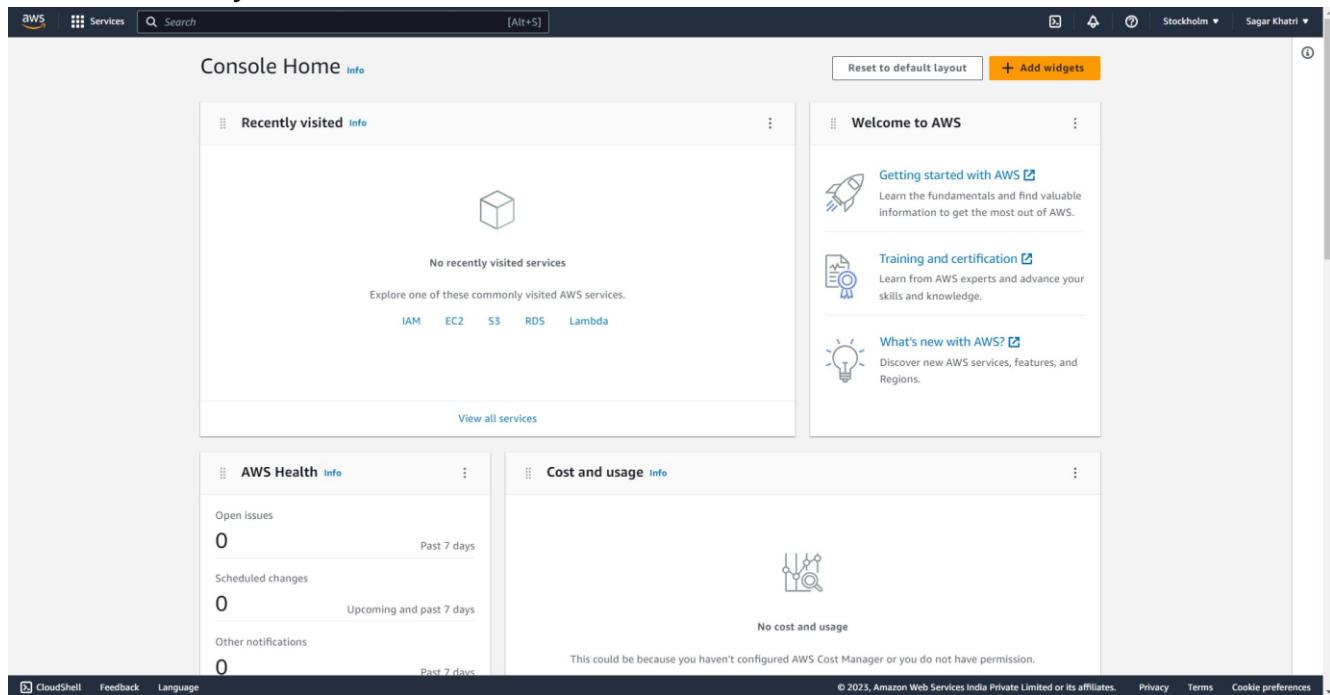
Click on login button which will take you to login page.

The screenshot shows the AWS Sign In page. At the top left is the AWS logo. Below it is a "Sign In" section with two radio button options: "Root user" (selected) and "IAM user". The "Root user" option is described as "Account owner that performs tasks requiring unrestricted access. Learn more". The "IAM user" option is described as "User within an account that performs daily tasks. Learn more". Below these options is a "Root user email address" field containing "sagar.khatni@nagarro.com". A "Next" button is located below the email field. To the right of the sign-in form is a "MACHINE LEARNING" section with the heading "Stay ahead with Machine Learning". It includes the text "Get resources showing how any business can use machine learning to boost revenue and increase efficiencies" and a "Learn more >" link. There's also a small graphic of a brain and a document. At the bottom of the page are links for "New to AWS?", "Create a new AWS account", and copyright information: "© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved." and a language selection dropdown set to "English".

Login as root user, and enter your email address, then enter your password :



This will take you to AWS Console Home



Now we can proceed with the assignment instructions from next page.

Creating a Virtual Network (VPC) with 2 Subnets:

- a) Open the AWS Management Console and navigate to the VPC service.

- b) Click on "Your VPCs" in the left-hand menu.

| VPC ID | Name | State | DNS hostnames | DNS resolution |
|-----------------------|------|-----------|---------------|----------------|
| vpc-0221debd1d328951f | - | Available | Enabled | Enabled |

| Attribute | Value |
|-------------------------------|--|
| Tenancy | Default |
| Default VPC | Yes |
| Network Address Usage metrics | Route 53 Resolver DNS Firewall rule groups |
| IPv4 CIDR | 172.31.0.0/16 |
| IPv6 CIDR | - |
| IPv6 pool | - |
| Owner ID | - |

c) Click on the "Create VPC" button.

VPC settings

- Resources to create:** VPC only (selected)
- Name tag - optional:** my-vpc-01
- IPv4 CIDR block:** 10.0.0.0/24 (selected)
- IPv6 CIDR block:** No IPv6 CIDR block (selected)
- Tenancy:** Default

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.

d) Enter the following details:

- VPC name: my-vpc-01
- IPv4 CIDR block: 10.0.0.0/24

e) Click on the "Create" button to create the VPC.

Details

| | | | |
|---|---|---|---|
| VPC ID: vpc-006411ad37e5c817b | State: Available | DNS hostnames: Disabled | DNS resolution: Enabled |
| Tenancy: Default | DHCP option set: dopt-0f5d0d3182b039ee3 | Main route table: rtb-0ba30ae4f0425db1c | Main network ACL: acl-0a9850ff954869dff |
| 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: 569579439013 | |

Resource map

- VPC: my-vpc-01
- Subnets (0): Subnets within this VPC
- Route tables (1): Route network traffic to resources
- Network connections (0): Connections to other networks

Introducing the VPC resource map

Solid lines represent relationships between resources in your VPC. Dotted lines represent network traffic to network functions.

f) Once the VPC is created, click on "Subnets" in the left-hand menu.

The screenshot shows the AWS VPC Subnets page. On the left, there's a navigation sidebar with sections like Virtual private cloud, Security, DNS firewall, and Network Firewall. The main area displays a table titled 'Subnets (3) Info' with columns: Name, Subnet ID, State, VPC, IPv4 CIDR, IPv6 CIDR, and Available IPv4. The table lists three subnets: subnet-0828c8e758939654f, subnet-0049fdfc4339b56c8, and subnet-01951fcda83784e8, all in the 'Available' state. A 'Create subnet' button is located at the top right of the table.

g) Click on the "Create subnet" button.

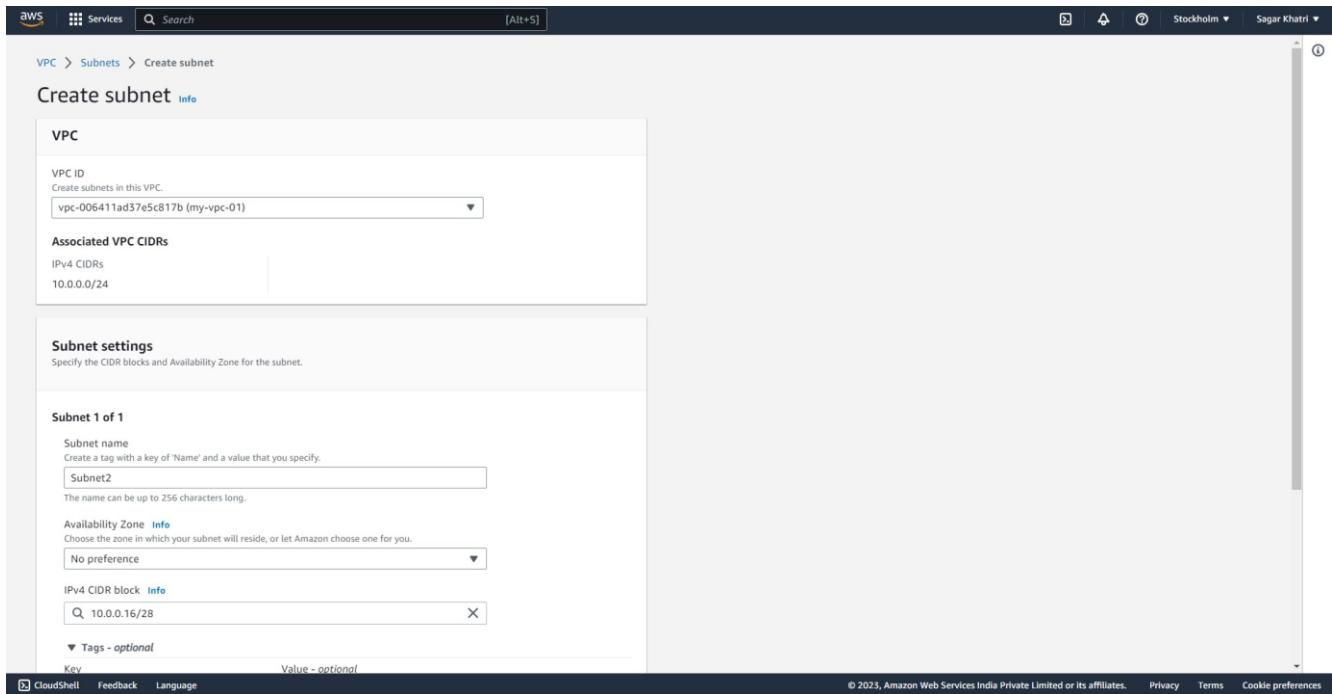
The screenshot shows the 'Create subnet' wizard. Step 1 is 'VPC'. It has a dropdown for 'VPC ID' containing 'my-vpc-01'. Below it, 'Associated VPC CIDRs' shows an IPv4 CIDR of '10.0.0.0/24'. Step 2 is 'Subnet settings', which asks for a 'Subnet name' ('Subnet1') and an 'Availability Zone' ('No preference'). Step 3 is 'IPv4 CIDR block', showing '10.0.0.0/28'. Step 4 is 'Tags - optional', with a key 'Name' and value 'Subnet1'.

h) Enter the following details for the first subnet:

- Name tag: Subnet1
- VPC: my-vpc-01
- Availability Zone: Stockholm
- IPv4 CIDR block: [Enter the CIDR block for the subnet]

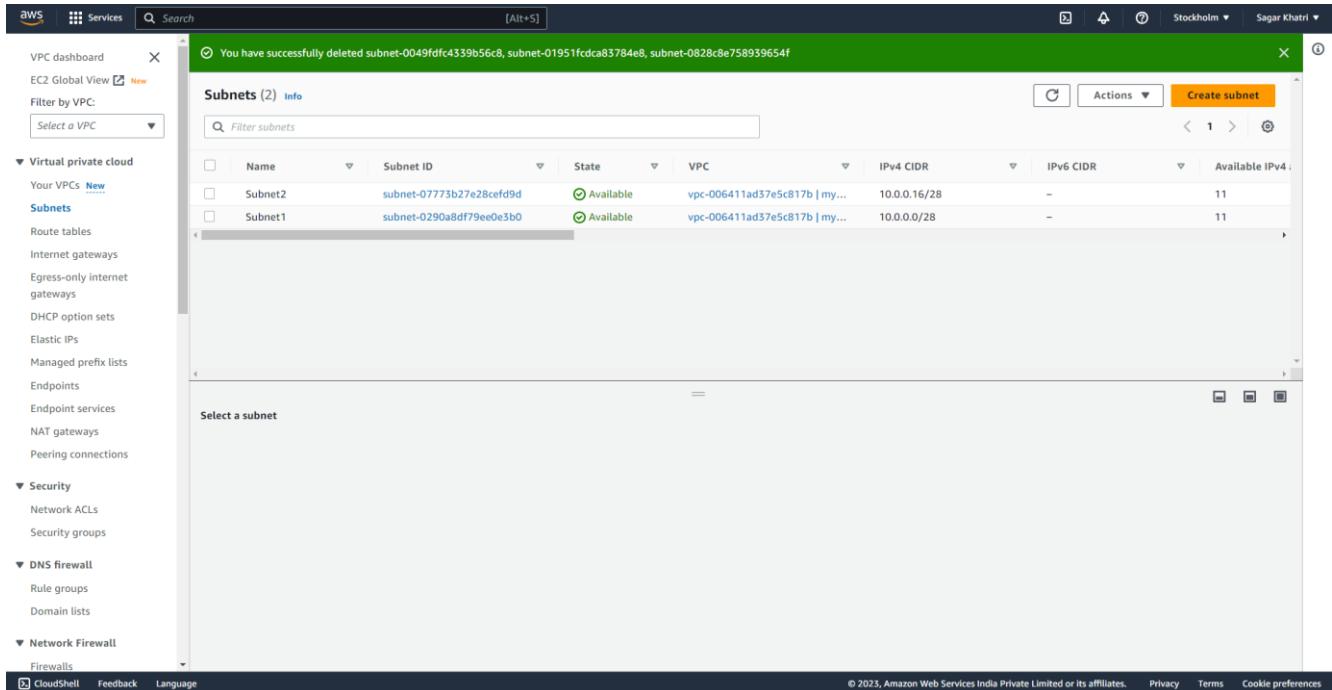
i) Click on the "Create" button to create the first subnet.

- j) Repeat steps h-i to create the second subnet, using a different CIDR block within the VPC.



The screenshot shows the 'Create subnet' page in the AWS VPC console. In the 'VPC' section, the VPC ID is set to 'vpc-006411ad37e5c817b (my-vpc-01)'. Under 'Associated VPC CIDs', the IPv4 CIDR is listed as '10.0.0.0/24'. The 'Subnet settings' section shows 'Subnet 1 of 1' being created. The 'Subnet name' is 'Subnet2', and the 'Availability Zone' is 'No preference'. The 'IPv4 CIDR block' is '10.0.0.16/28'. There are no tags defined for this subnet.

- k) View the created subnets



The screenshot shows the 'Subnets (2)' list page in the AWS VPC console. A green banner at the top indicates that three subnets have been successfully deleted. The table lists two subnets: 'Subnet2' and 'Subnet1'. Both subnets are in the 'Available' state, belong to the VPC 'vpc-006411ad37e5c817b', and have an IPv4 CIDR of '10.0.0.16/28'. The table also shows an 'Actions' dropdown and a 'Create subnet' button.

| Name | Subnet ID | State | VPC | IPv4 CIDR | IPv6 CIDR | Available IPv4 |
|---------|--------------------------|-----------|-------------------------------|--------------|-----------|----------------|
| Subnet2 | subnet-07773b27e28cefd9d | Available | vpc-006411ad37e5c817b my... | 10.0.0.16/28 | - | 11 |
| Subnet1 | subnet-0290a8df79ee0e3b0 | Available | vpc-006411ad37e5c817b my... | 10.0.0.0/28 | - | 11 |

Creating a VM and Deploying Application Code in one of the Subnets:

a) Launch an EC2 Instance in the Desired Subnet:

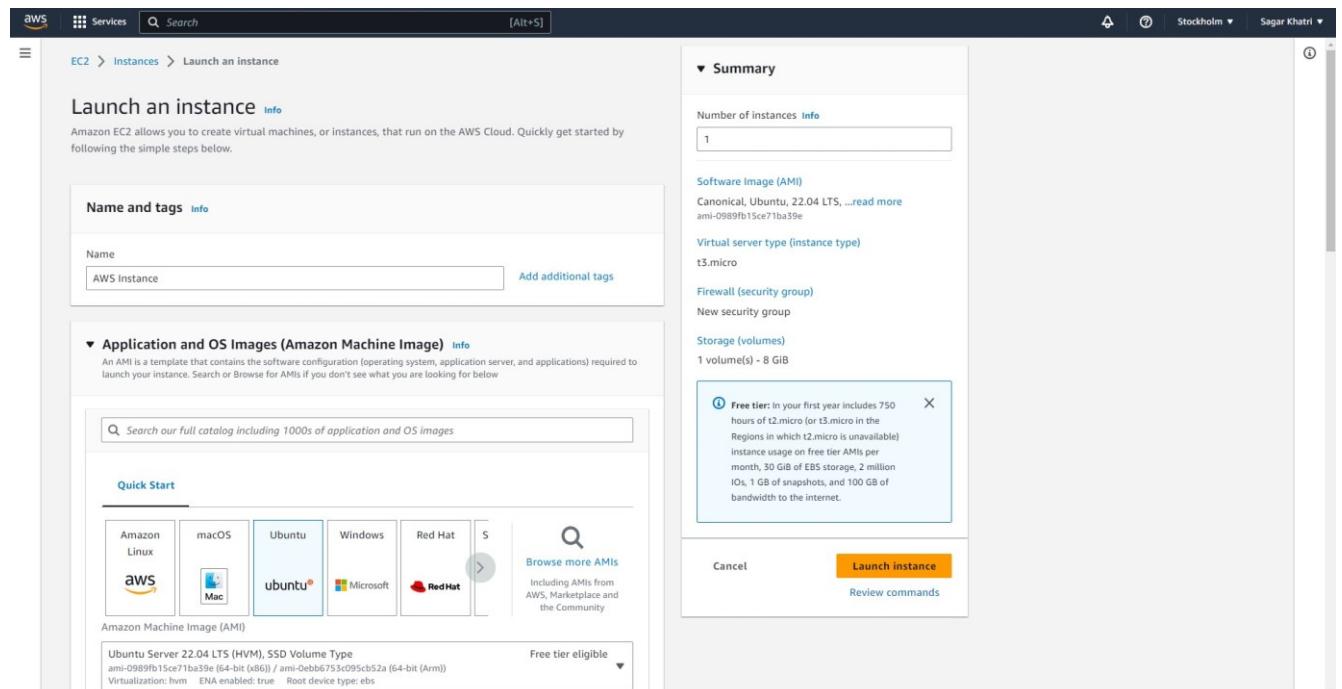
- Open the AWS Management Console and navigate to the EC2 service.

The screenshot shows the AWS Management Console search interface. The search term 'EC2' is entered in the search bar. Below the search bar, there are two sections: 'Services' and 'Features'. The 'Services' section lists EC2, EC2 Image Builder, Amazon Inspector, and AWS Firewall Manager. The 'Features' section lists Dashboard, Limits, and AMIs. On the right side of the screen, there is a sidebar titled 'Welcome to AWS' with links for 'Getting started with AWS', 'Training and certification', and 'What's new with AWS?'. The bottom of the screen shows the AWS footer with copyright information and links for CloudShell, Feedback, Language, Privacy, Terms, and Cookie preferences.

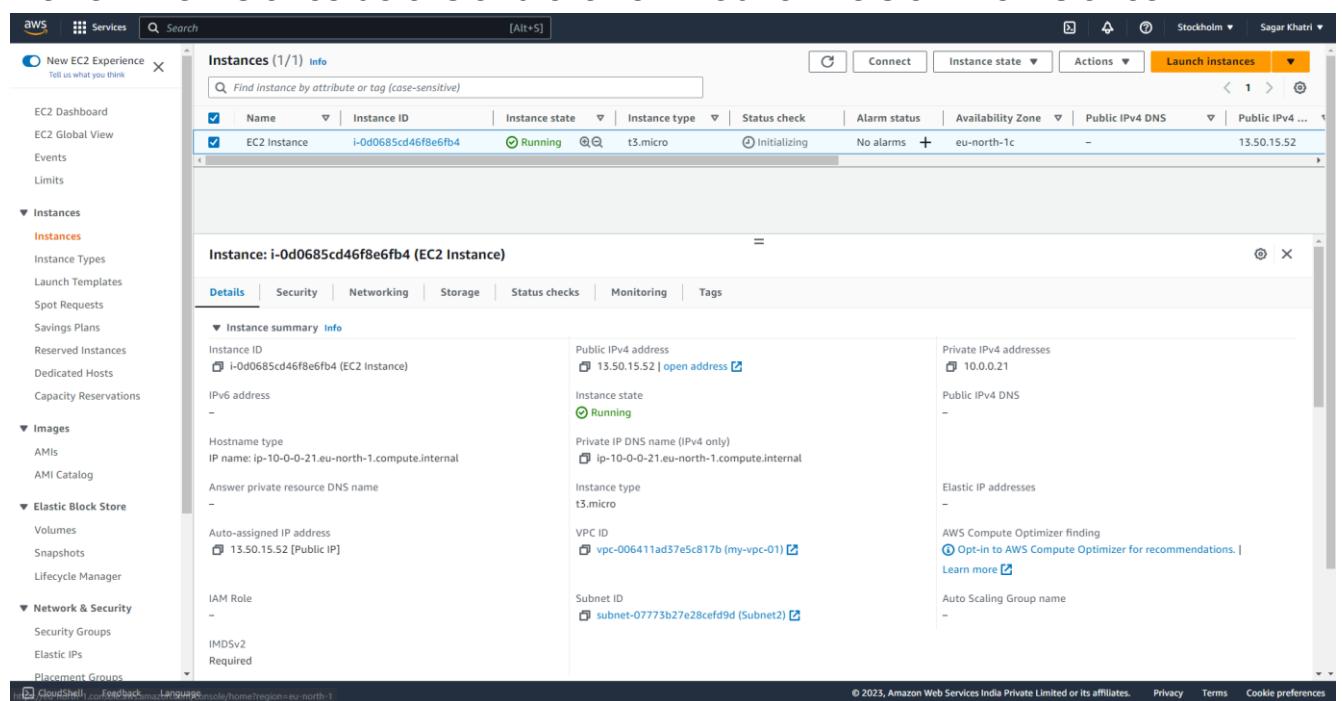
- Click on "Instances" in the left-hand menu.

The screenshot shows the AWS EC2 Dashboard. The left sidebar has a navigation menu with options like EC2 Dashboard, Instances, Images, Elastic Block Store, and Network & Security. The main content area is titled 'Resources' and shows the following statistics for the Europe (Stockholm) Region: Instances (running) 0, Auto Scaling Groups 0, Dedicated Hosts 0, Elastic IPs 0, Instances 0, Key pairs 0, Load balancers 0, Placement groups 0, Security groups 2, Snapshots 0, and Volumes 0. Below this, there is a 'Launch instance' button and a note about launching instances in the Europe (Stockholm) Region. To the right, there are sections for 'Service health' (status: This service is operating normally), 'Zones' (listing eu-north-1a, eu-north-1b, eu-north-1c, Zone ID eun1-az1, eun1-az2, eun1-az3), and 'Explore AWS' (sections for Get Up to 40% Better Price Performance, Enable Best Price-Performance with AWS Graviton, and a note about saving up to 90% on EC2 with Spot Instances). The bottom of the dashboard shows a 'Migrate a server' button.

- Click on the "Launch Instances" button.

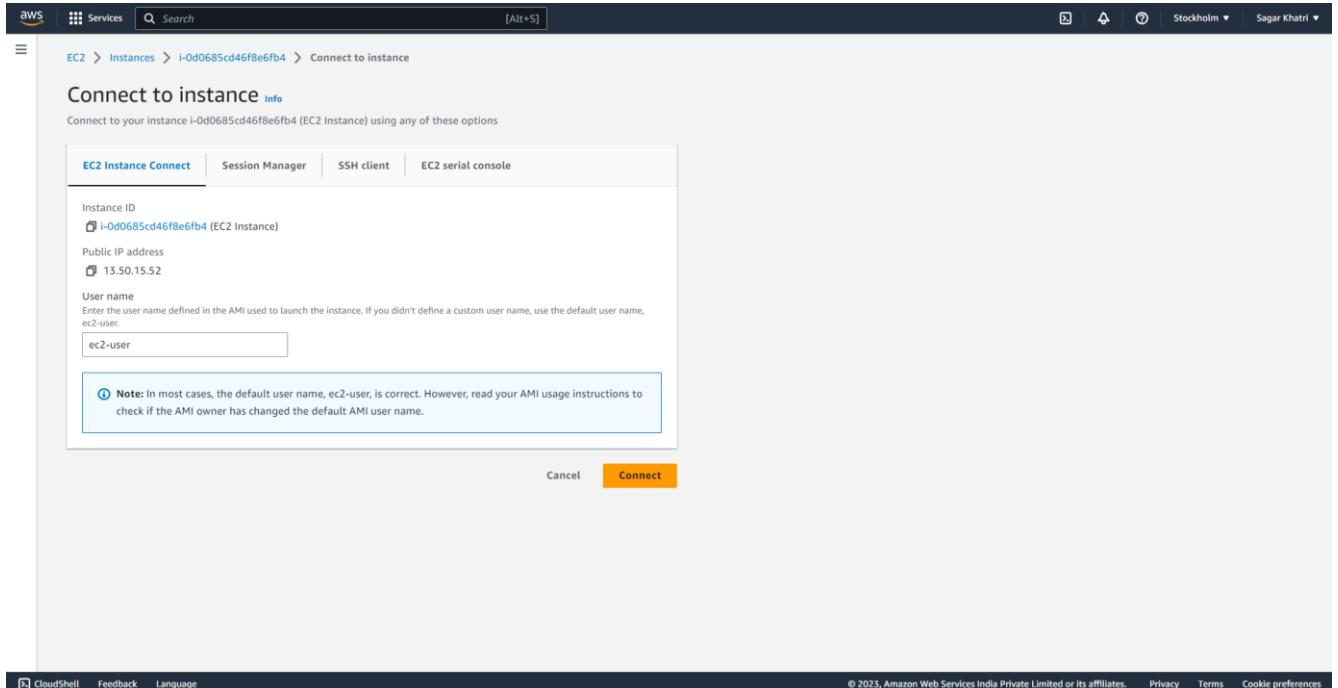


- Select an Amazon Machine Image (AMI) that matches your application requirements.
- Choose an instance type based on your application's resource needs.
- Configure the instance details, such as the subnet, security group, and any additional settings.
- Review the instance details and click on "Launch" to start the instance.



b) Connect to the Instance:

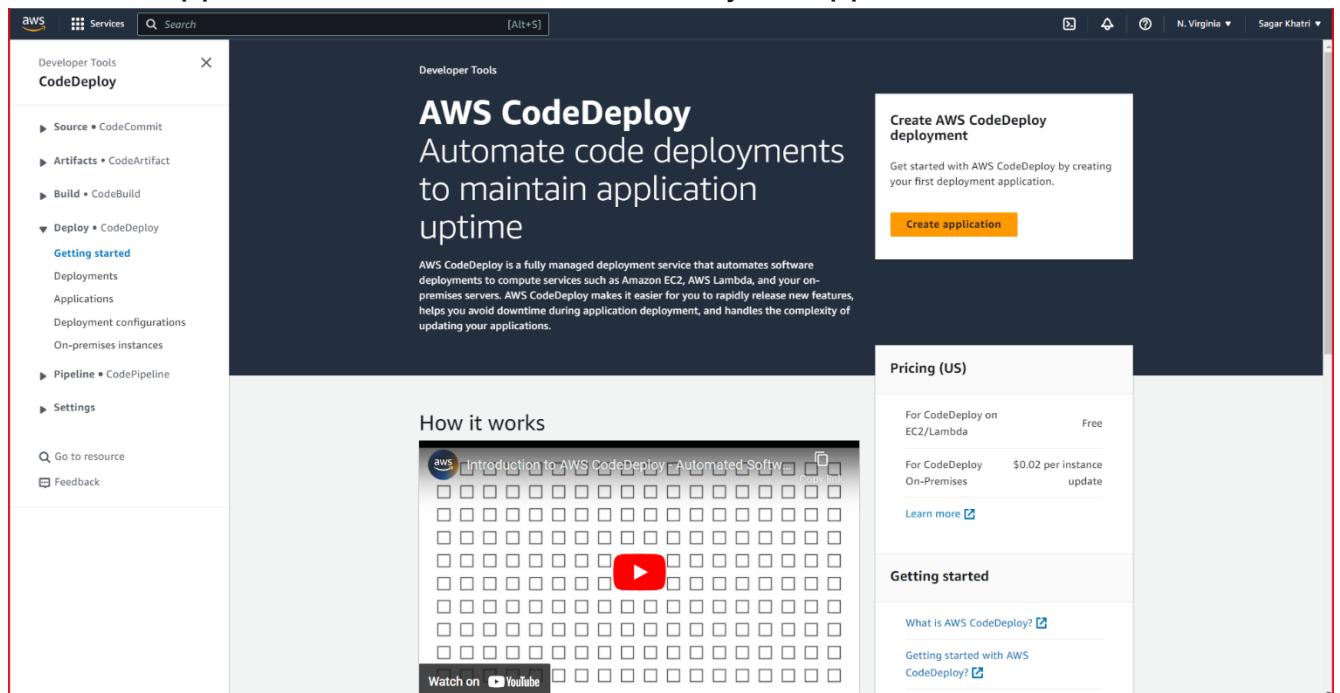
- Once the instance is running, select the instance from the EC2 dashboard.
- Click on the "Connect" button and follow the instructions to connect to the instance using SSH or Remote Desktop, depending on the operating system.
- Use the appropriate credentials to log in to the instance.



b) Deploy the Application Code:

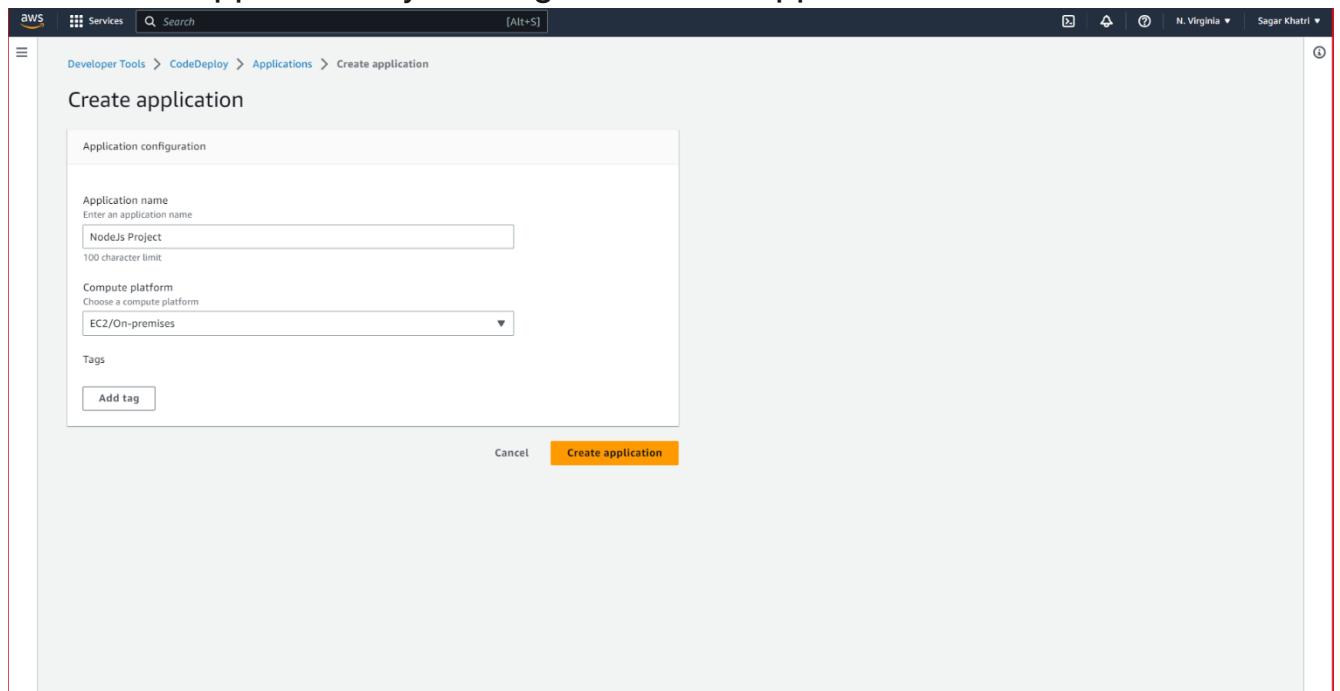
- Transfer the application code to the instance using secure file transfer methods like SCP or SFTP.
- Install any necessary dependencies or libraries required by your application.
- Configure the application environment and any relevant settings.

- Start the application or web server to make your application accessible.



The screenshot shows the AWS CodeDeploy landing page. On the left, there's a sidebar with navigation links for Source, Artifacts, Build, Deploy, Pipeline, and Settings. The main content area features the title "AWS CodeDeploy" and the subtitle "Automate code deployments to maintain application uptime". Below this is a brief description of what AWS CodeDeploy does. To the right, there's a section titled "Create AWS CodeDeploy deployment" with a "Create application" button. At the bottom, there's a "How it works" section with a video thumbnail and a "Watch on YouTube" button.

- Create the application by clicking on “Create Application”.



The screenshot shows the "Create application" configuration page. The URL in the browser is "Developer Tools > CodeDeploy > Applications > Create application". The form has fields for "Application name" (set to "NodeJs Project"), "Compute platform" (set to "EC2/On-premises"), and "Tags" (with a "Add tag" button). At the bottom right of the form is a "Create application" button.

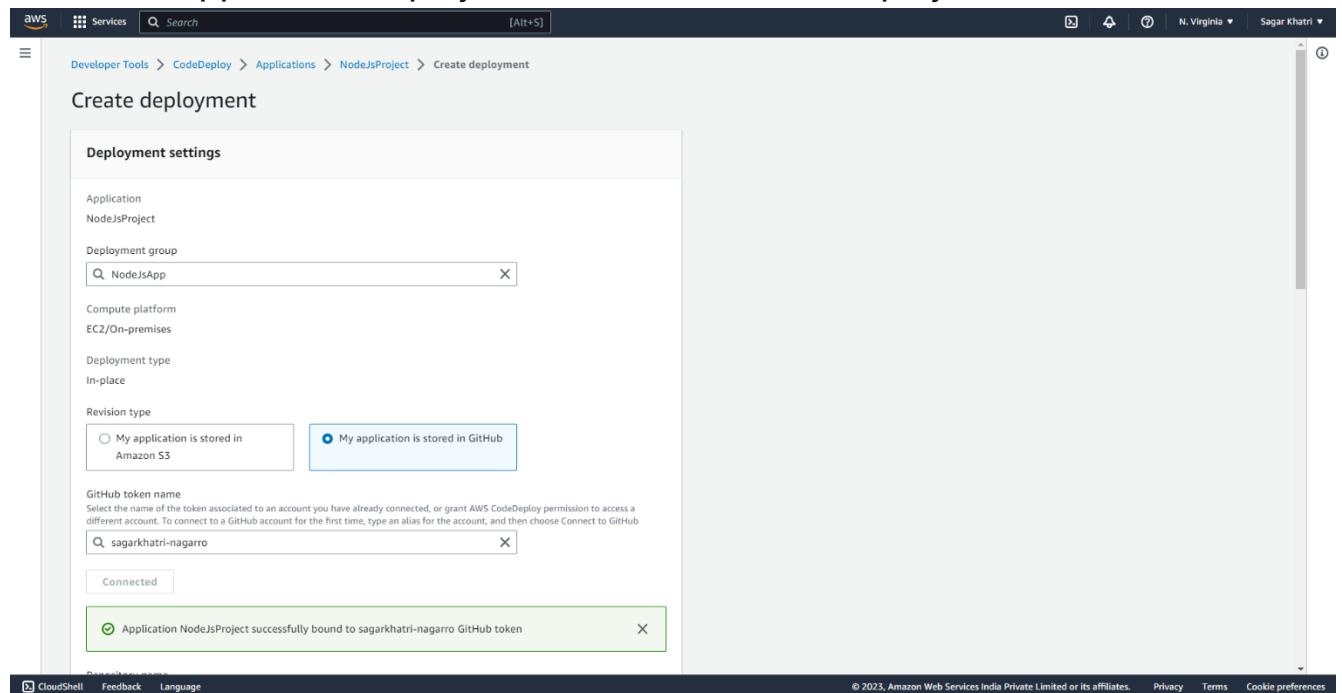
- After creating the application, we next have to create a deployment group for initiating our deployments.

The screenshot shows the 'Create deployment group' wizard in the AWS CodeDeploy console. The 'Deployment group name' field is populated with 'NodeJsDeployment'. The 'Service role' field contains a dropdown menu with 'Q NodeService' selected. The 'Deployment type' field has 'In-place' selected.

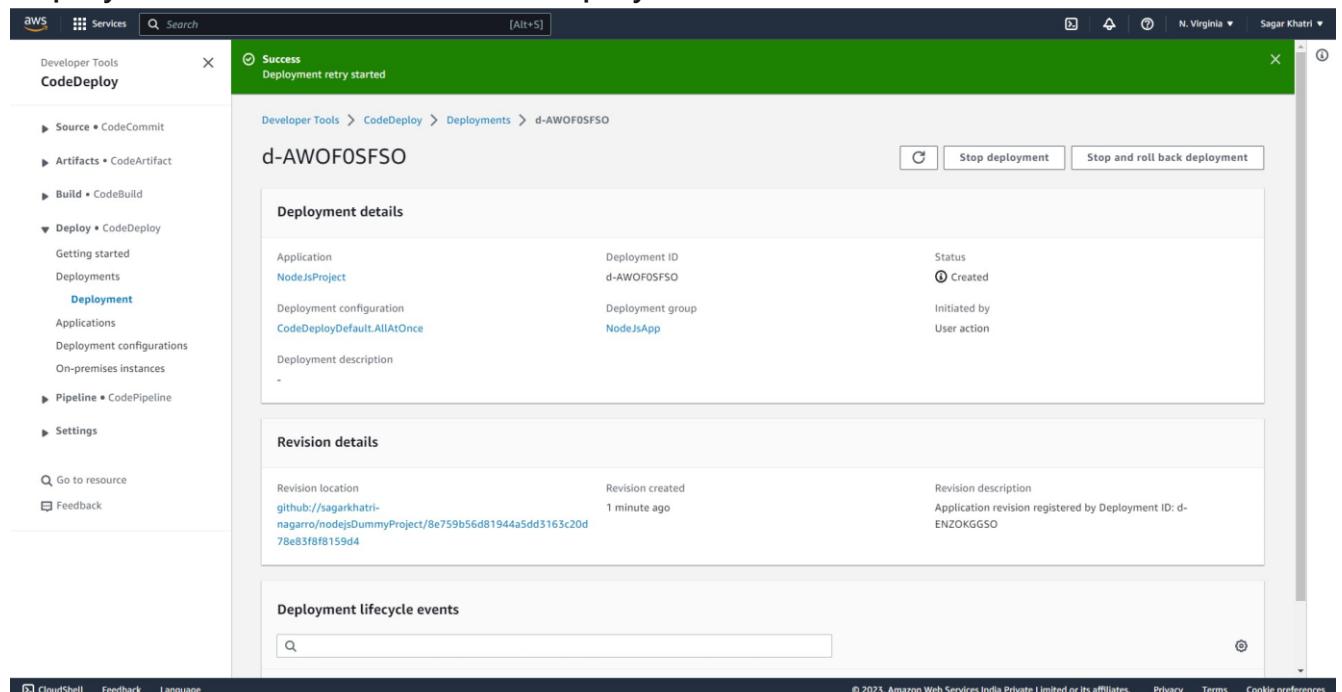
- Once the deployment group is created, we can now see it visible on the screen where we can deploy our app into it.

The screenshot shows the 'NodeJsApp' deployment group details page in the AWS CodeDeploy console. The deployment group name is 'NodeJsApp', the application name is 'NodeJsProject', and the service role ARN is 'arn:aws:iam::569579439013:role/CodeDeployServiceRole'. The deployment type is 'In-place'. The environment configuration section shows an 'Application' key with value 'NodeJsApp'. There are no triggers listed.

- To create application deployment click on “Create deployment”.



- Here, select your application stored in GitHub, where you will be asked your github token name.
- Once you are done with the filling of required information, click on “create deployment” button to finish the deployment.



- It will look like this once the deployment is complete.

Deploy the same application to Elastic beanstalk Service:

- a) Open the AWS Management Console and navigate to the Elastic Beanstalk service.

- b) Click on "Create Application" and provide the necessary details (application name, platform, etc.).
- c) Configure the application settings, such as environment type, instance type, and scaling options.
- d) Upload your application code or connect to a version control system.
- e) From the left panel click environments.

- f) Click on create environment button to create a new beanstalk environment.

Configure environment

Environment tier [Info](#)
Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.

- Web server environment
Run a website, web application, or web API that serves HTTP requests. [Learn more](#)
- Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. [Learn more](#)

Application information [Info](#)

Application name

Maximum length of 100 characters.

▼ Application tags (optional)
Apply up to 50 tags. You can use tags to group and filter your resources. A tag is a key-value pair. The key must be unique within the resource and is case-sensitive. [Learn more](#)

| Key | Value - optional |
|-----------------------------------|--|
| <input type="text" value="Name"/> | <input type="text" value="AppInstance"/> |

[Add new tag](#)

Add your name and respective keys like Name and their values. Fill in the values to configure your environment.

- g) Now proceed to second step of this process and configure the service access by defining the existing service role or creating a new one. Also Choose an EC2 pair key created before.

Configure service access

Service access
IAM roles, assumed by Elastic Beanstalk as a service role, and EC2 instance profiles allow Elastic Beanstalk to create and manage your environment. Both the IAM role and instance profile must be attached to IAM managed policies that contain the required permissions. [Learn more](#)

Service role
 Create and use new service role
 Use an existing service role

Service role name
Enter the name for an IAM role that Elastic Beanstalk will create to assume as a service role. Beanstalk will attach the required managed policies to it.

[View permission details](#)

EC2 key pair
Select an EC2 key pair to securely log in to your EC2 instances. [Learn more](#)

EC2 instance profile
Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations.

[View permission details](#)

h) Now setup networking Database and tags.

i) Configure instance traffic and scaling

j) Setup updates ,monitoring and logging

Configure updates, monitoring, and logging - optional

Monitoring

Health reporting

Enhanced health reporting provides free real-time application and operating system monitoring of the instances and other resources in your environment. The **EnvironmentHealth** custom metric is provided free with enhanced health reporting. Additional charges apply for each custom metric. For more information, see [Amazon CloudWatch Pricing](#).

System

Basic
 Enhanced

CloudWatch Custom Metrics - Instance

CloudWatch Custom Metrics - Environment

Health event streaming to CloudWatch Logs

Configure Elastic Beanstalk to stream environment health events to CloudWatch Logs. You can set the retention up to a maximum of ten years and configure Elastic Beanstalk to delete the logs when you terminate your environment.

Log streaming
 Activated (standard CloudWatch charges apply.)

Retention

k) Now click on the review button to review your selected settings before creating the environment.

Review

Step 1: Configure environment

Environment information

| | |
|--|------------------|
| Environment tier | Application name |
| Web server environment | NodeJsWebApp |
| Environment name | Application code |
| NodeJsWebApp-env | index.js |
| Platform | |
| arn:aws:elasticbeanstalk:us-east-1:platform/Node.js 18 | |
| running on 64bit Amazon Linux 2/5.8.2 | |

Step 2: Configure service access

Service access

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.

| | | |
|---|--------------|-------------------------------|
| Service role | EC2 key pair | EC2 instance profile |
| arn:aws:iam::569579439013:role/service-role | virginiakey | aws-elasticbeanstalk-ec2-role |

i) Review your capacity and load balancer metrics to avoid additional cost.

| Capacity | | |
|--------------------------|--------------------------|----------------------|
| Environment type | Fleet composition | On-demand base |
| Single instance | On-Demand instance | 0 |
| On-demand above base | Capacity rebalancing | Processor type |
| 0 | Deactivated | x86_64 |
| Instance types | AMI ID | Availability Zones |
| t3.micro,t3.small | ami-05ed12a50d2fdb1a0 | Any |
| Metric | Statistic | Unit |
| NetworkOut | Average | Bytes |
| Period | Breach duration | Upper threshold |
| 5 | 5 | 6000000 |
| Scale up increment | Lower threshold | Scale down increment |
| 1 | 2000000 | -1 |
| Load balancer | | |
| Load balancer visibility | Load balancer subnets | |
| public | subnet-09800deee5d07a1e4 | |

m) Now click Submit to create the new environment.

The screenshot shows the AWS Elastic Beanstalk console with the following details:

- Left sidebar:** Shows the navigation menu with "Applications", "Environments", and "Change history". Under "Environments", "NodeJsWebApp" is expanded, showing "Application versions" and "Saved configurations". Below it, "Environment: NodeJsWebApp-env" is expanded, showing "Go to environment" (with a link), "Configuration", "Events", "Health", "Logs", "Monitoring", "Alarms", "Managed updates", and "Tags".
- Header:** Shows "Elastic Beanstalk" in the top-left, a search bar, and a message: "Elastic Beanstalk is launching your environment. This will take a few minutes." The top right includes account information ("N. Virginia" and "Sagar Khatri") and navigation icons.
- Content Area:**
 - Environment Overview:** Displays "Health" (Unknown), "Domain" (empty), "Environment ID" (e-r7ifn2iyi), and "Application name" (NodeJsWebApp).
 - Platform:** Shows "Platform" as "Node.js 18 running on 64bit Amazon Linux 2/5.8.2" and "Running version" as empty.
 - Events Tab:** Active tab, showing 2 events. A search bar at the top says "Filter events by text, property or value". The table below has columns "Time", "Type", and "Details".

you can now deploy your application in the created beanstalk environment now.

Once the deployment is successful, you can access your application using the provided URL or the Elastic Beanstalk environment URL.

Test your application to ensure it is functioning as expected.

Creating a Lambda Function Triggered by S3 Bucket Upload:

- a. Open the AWS Management Console and navigate to the Lambda service.

The screenshot shows the AWS Lambda service homepage. At the top, there's a banner with the text "AWS Lambda lets you run code without thinking about servers." Below the banner, a paragraph explains that users pay only for compute time consumed. To the right, there's a "Get started" section with a "Create a function" button. Below this, the "How it works" section is visible, featuring a "Run" button and a sample Node.js code snippet:

```

1 * exports.handler = async (event) => {
2     console.log(event);
3     return 'Hello from Lambda!';
4 };
5

```

At the bottom of the page, there are links for CloudShell, Feedback, Language, and cookie preferences.

- b. Click on "Create function" and choose the authoring method (e.g., "Author from scratch").

The screenshot shows the "Create function" wizard. In the "Basic information" step, the "Author from scratch" option is selected. Other options shown are "Use a blueprint" and "Container image". The "Function name" field is set to "NodeJsFunction". The "Runtime" dropdown is set to "Node.js 18.x". The "Architecture" dropdown has "x86_64" selected. The "Permissions" section is partially visible at the bottom.

c. Provide the necessary details for the function (function name, runtime, etc.).

The screenshot shows the 'Create function' wizard on the AWS Lambda service. The current step is 'Change default execution role'. It asks for an execution role, with the option 'Create a new role with basic Lambda permissions' selected. A note says 'Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.' Below this, it states that Lambda will create an execution role named NodeJsFunction-role-qyrrz5kjp with permission to upload logs to Amazon CloudWatch Logs.

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).
 Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

ⓘ Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named NodeJsFunction-role-qyrrz5kjp, with permission to upload logs to Amazon CloudWatch Logs.

Advanced settings

- [Enable Code signing](#) Info
Use code signing configurations to ensure that the code has been signed by an approved source and has not been altered since signing.
- [Enable function URL](#) Info
Use function URLs to assign HTTP(S) endpoints to your Lambda function.
- [Enable tags](#) Info
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, track your AWS costs, and enforce attribute-based access control.
- [Enable VPC](#) Info
Connect your function to a VPC to access private resources during invocation.

d. Under "Permissions," choose an existing or create a new execution role with the necessary S3 permissions.

e. Click on the "Create function" button to create the Lambda function.

The screenshot shows the 'NodeJsFunction' overview page. A green banner at the top indicates 'Successfully created the function NodeJsFunction. You can now change its code and configuration. To invoke your function with a test event, choose "Test".' The main area displays the function's name, a placeholder for triggers and destinations, and its ARN. The ARN is arn:aws:lambda:us-east-1:569579439013:function:NodeJsFunction. The 'Code' tab is selected, showing the 'Code source' section with an 'Upload from' button.

Function overview Info

NodeJsFunction

Description
-

Last modified
36 seconds ago

Function ARN
arn:aws:lambda:us-east-1:569579439013:function:NodeJsFunction

Function URL Info
-

Actions

Code source Info

Upload from

f. Scroll to the code source in the lambda function created.

The screenshot shows the AWS Lambda console interface. At the top, there are buttons for '+ Add trigger' and '+ Add destination'. On the right, it displays the last modified time ('7 minutes ago'), Function ARN ('arn:aws:lambda:us-east-1:569579439013:function:No deJsFunction'), and Function URL ('Info'). Below this, tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions' are visible, with 'Code' being the active tab. Under the 'Code source' section, there is a file browser showing a folder 'NodeJsFunction' containing a file 'index.mjs'. The code editor window shows the following JavaScript code:

```

1 exports.handler = async (event) => {
2     // Retrieve the file name from the S3 event
3     const fileName = event.Records[0].s3.object.key;
4
5     // Print the file name
6     console.log('The File uploaded by Sagar Khatri is : ', fileName);
7
8     // Return a response if necessary
9     return {
10         statusCode: 200,
11         body: 'Sagar has uploaded the File successfully',
12     };
13 };
14

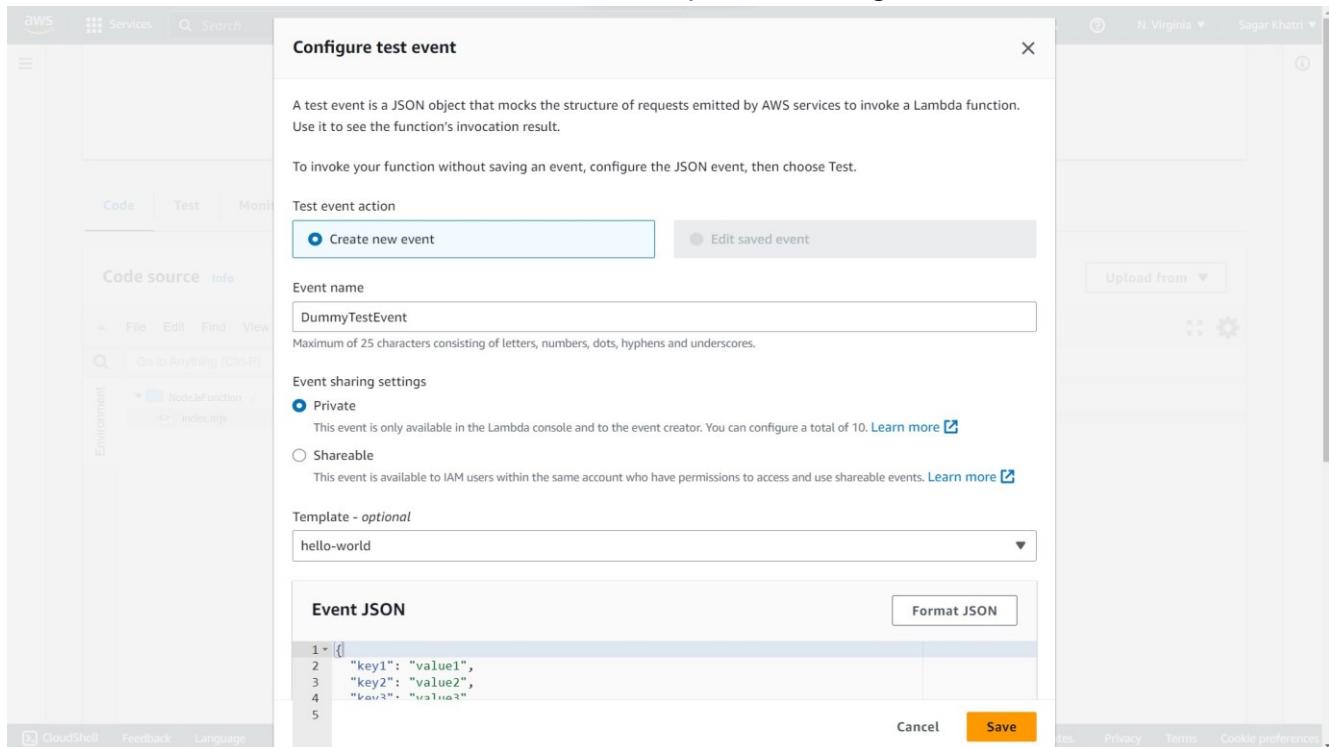
```

g. In the function's code editor, write the code to handle the S3 trigger event and print the uploaded file's name.

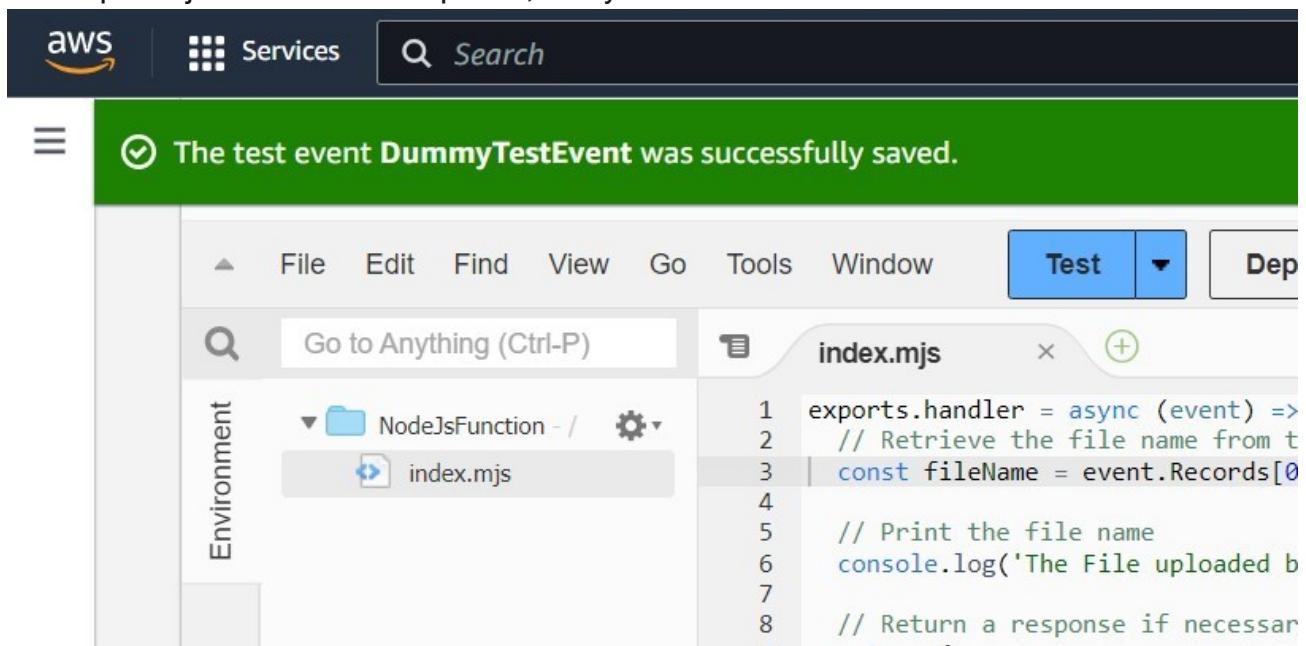
The screenshot shows the AWS Lambda code editor for the 'index.mjs' file. The code is identical to the one shown in the previous screenshot, but the cursor is positioned on the line starting with 'return {'. This indicates that the user is currently editing or preparing to save the function's code.

h. Save the Lambda function.

I. Click on “Test” and then click on configure to open the dialog box



j. Fill the details of the test event to test the code which is already written above and provided the required json data and templates, if any. Click Save.



Now run the test event to check whether the code is working or not. Then click Deploy to deploy the function.

The screenshot shows the AWS Lambda console interface. At the top, there's a green banner stating "The test event DummyTestEvent was successfully saved." Below this, the "Execution results" section displays the following details:

- Test Event Name:** DummyTestEvent
- Status:** Succeeded | Max memory used: 68 MB | Time: 110.22 ms
- Response:**

```
{
  "statusCode": 200,
  "body": "Sagar has uploaded the File successfully"
}
```
- Function Logs:**

```
START RequestId: 634df51b-d216-4cb6-9c64-ba437ccf56ff Version: $LATEST
END RequestId: 634df51b-d216-4cb6-9c64-ba437ccf56ff
REPORT RequestId: 634df51b-d216-4cb6-9c64-ba437ccf56ff Duration: 110.22 ms Billed Duration: 111 ms Memory Size: 128 MB Max Memory Used: 68 MB
```
- Request ID:** 634df51b-d216-4cb6-9c64-ba437ccf56ff

At the bottom, there's a "Code properties" section and a standard AWS footer with links for CloudShell, Feedback, Language, Privacy, Terms, and Cookie preferences.

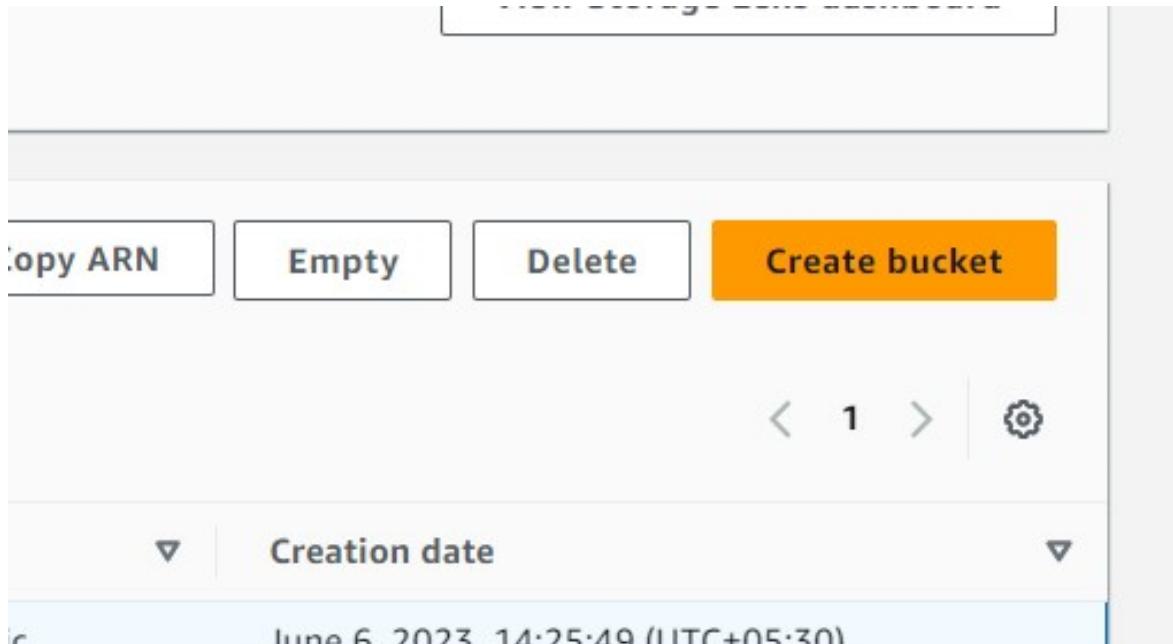
Now move to the search bar to search for S3 bucket. Click on S3 in the search suggestions.

The screenshot shows the AWS S3 console interface. On the left, there's a navigation sidebar with options like Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Metrics, IAM Access Analyzer for S3, Storage Lens, Dashboards, AWS Organizations settings, Feature spotlight, and AWS Marketplace for S3. The main area is titled "Amazon S3" and shows the following details:

- Account snapshot:** Storage lens provides visibility into storage usage and activity trends. [Learn more](#)
- Buckets (1) [Info](#)**: Buckets are containers for data stored in S3. [Learn more](#)
 - Name:** elasticbeanstalk-us-east-1-1-569579439013
 - AWS Region:** US East (N. Virginia) us-east-1
 - Access:** Objects can be public
 - Creation date:** June 6, 2023, 14:25:49 (UTC+05:30)

At the bottom, there's a standard AWS footer with links for CloudShell, Feedback, Language, Privacy, Terms, and Cookie preferences.

Click on the “Create bucket” button to start creating a new bucket.



Enter the details of the S3 bucket you want to create, like its name and region. Also select the ownership of objects written in this bucket.

The screenshot shows the 'Create bucket' configuration page. The 'General configuration' section includes fields for 'Bucket name' (set to 'myAwsBucket') and 'AWS Region' (set to 'US East (N. Virginia) us-east-1'). The 'Object Ownership' section contains two options: 'ACLs disabled (recommended)' (selected) and 'ACLs enabled'. The 'Object Ownership' field is currently set to 'Bucket owner enforced'.

click Finish to finish the creation of bucket. You will be able to see the newly created bucket in the dashboard.

Now we should create an intelligent tiering archive config as bucketConfig.

fill the details to create and view the bucket config

The screenshot shows the AWS S3 Intelligent-Tiering Archive configurations page. At the top, a green banner indicates "Successfully created Intelligent-Tiering archive configuration 'bucketConfig'". Below the banner, the breadcrumb navigation shows: Amazon S3 > Buckets > sdffslkdksdz > Intelligent-Tiering Archive configurations. A table lists the configuration with one row: Name (bucketConfig), Status (Enabled), Scope (Prefix: prefix, Tags: [Name: bucketConfig]), Days until transition to Archive Access tier (91), and Days until transition to Deep Archive Access tier (-). There are buttons for View details, Edit, Delete, and Create configuration.

In the bucket configuration, create an event config by scrolling down to event config

The screenshot shows the AWS S3 Bucket Configuration page. The "Event notifications" section is visible, containing a table with columns: Name, Event types, Filters, Destination type, and Destination. A button labeled "Create event notification" is present. Other sections shown include "Access" (No data events), "Amazon EventBridge" (Send notifications to Amazon EventBridge for all events in this bucket, currently Off), "Transfer acceleration" (Transfer acceleration Disabled), and "Object Lock" (Store objects using a write-once-read-many (WORM) model to help prevent deletion or overwritten for a fixed amount of time or indefinitely). A "Configure in CloudTrail" button is also visible.

Click create a new config button to start.

The screenshot shows the AWS S3 console with the 'Intelligent-Tiering' configuration page. In the 'Trigger' section, a new rule is being created:

- Event:** Delete marker added by Lifecycle for a versioned object (s3:LifecycleExpirationDeleteMarkerCreated)
- Destination:** Lambda function (NodeJSFunction)
- Save changes** button is visible at the bottom.

choose the information like destination and the trigger like “put” triggers to trigger the event config. Choose Lambda function in the destination section and then click “Save changes”

The screenshot shows the AWS S3 console with a green success message: "Successfully created event notification "eventconfig". Operation successfully completed."

The event notifications list shows one entry:

| Name | Event types | Filters | Destination type | Destination |
|-------------|--------------------------|---------|------------------|----------------|
| eventconfig | All object create events | - | Lambda function | NodeJSFunction |

On successful creation you have now created a Lambda that should trigger as soon as you upload a file in the S3 bucket.