



DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

LABORATORY MANUAL

III Semester

Batch:2024-26

Name: Mayur Maruti Hambar

USN: 1MS24MC055

Course: Cloud Computing

Course code: 24MCASS3

Course Credits: 0 : 1: 2

RAMAIAH INSTITUTE OF TECHNOLOGY

(Autonomous Institute, Affiliated to VTU)

Accredited by National Board of Accreditation & NAAC with 'A+' Grade, MSR

Nagar, MSRIT Post, Bangalore-560054

www.msrit.edu

Table of Lab Programs

| Sl. No. | Programs/Exercise/Topic. |
|--------------------|--|
| 1. | AWS Account Setup and Configuration. AWS Console Overview. Enable MFA. Create AWS budget alert. |
| 2. | AWS Identity Access Management (IAM) User and Group creation. Enable AWS IAM MFA. Create an AWS Account Alias (for Alternate Sign-in URL) |
| 3. | Amazon S3 – Introduction, Bucket Creation and upload objects (files). |
| 4. | Amazon S3 – Static Website Hosting (Multi-Page website), Versioning, Cross-Region Replication rule. |
| 5. | Overview of EC2. To Launch a Windows EC2 Instance and Connect via RDP Client. |
| 6. | Launch a Linux EC2 Instance and Connect using SSH through PowerShell and PuTTY on Windows machine. Launch a Linux EC2 Instance and Connect using SSH through Linux terminal on Linux machine. Launch a Linux EC2 Instance and Connect through EC2 connect option on the console. |
| 7. | Hosting a static website on EC2 instance: - Manual Installation of Apache (httpd) Web Server on EC2 for Static Website Hosting. - Launching an EC2 Instance with User Data Script to Automatically Install Apache and Host a Static Website. |
| 8. | Custom AMI - Create a Custom AMI from a Working EC2 Instance. - Launch an EC2 Instance using a Custom AMI. - Delete the custom AMI [Deregister and delete snapshot] |
| 9. | Mini-Project: Hosting a Multi-Page Website using EC2 and S3 Submission: 11-11-2025, Marks: 5 |
| 10. | Creation and Configuration of a Custom AWS Virtual Private Cloud (VPC). [Including Public and Private Subnets, Internet Gateway, NAT Gateway, Route Tables] |
| 11. | Launching and Configuring EC2 Instances for Web Server in Public Subnet and Database Server in Private Subnet Using NAT Gateway for Outbound Access. |
| 12. | Deploying and Testing a Load-Balanced Web Application By Creating a Web Server, Building a Custom AMI, Configuring a Target Group, Launching an Application Load Balancer (ALB), Registering the Instance in the Target Group, and Testing the ALB DNS. |
| 13. | Stress Testing a Linux EC2 Instance (CPU Load Test) |
| 14. | Mini Project – Deploying a Load-Balanced Web Application using Application Load Balancer (ALB), Auto Scaling Group (ASG), Custom AMI, and Target Group on AWS with CloudWatch Alarms & Stress Testing for Auto Scaling Validation. |
| 15. | Hosting a WordPress Content Management Website on Amazon Lightsail . (only demo) |

| | |
|-----|--|
| 16. | Building a Basic Python Flask Web Application — Fundamentals of Web Requests & Form Handling (Pre-Requisite for AWS RDS Connectivity Lab) |
| 17. | Flask Web App with Registration & Login Using AWS RDS (MySQL) (Full Deployment on AWS EC2 + AWS RDS) |
| 18. | Creating and Operating a NoSQL Key-Value Database using Amazon DynamoDB |
| 19. | ElastiCache (Redis) as an In-Memory Cache |
| 20. | Deploy a simple Flask application on AWS Elastic Beanstalk and verify it using the public URL. |
| 21. | Deploy a Flask App on AWS Elastic Beanstalk and Perform CRUD on DynamoDB (Using AWS SDK/boto3) |
| 22. | CloudFormation – Launch EC2 (Amazon Linux 2023) + Install Apache using UserData |
| 23. | Static Content Pulled from S3 using CloudFormation [EC2 + S3] |
| 24. | AWS Lambda : Input Processing, Business Logic Execution, and CloudWatch Logging. |
| 25. | Event-Driven Notification System using AWS Lambda and Amazon SNS |
| 26. | Analyze data in a CSV File stored in S3 Using Amazon Athena (No Server) |
| 27. | Smart Sensor Monitoring System using AWS IoT Core . Cloud-Based IoT Data Ingestion and Processing using AWS |
| 28. | Accelerate an S3 Static Website Using Amazon CloudFront (CDN) |
| | Mini Project – Global Static Website Delivery using S3 + CloudFront with Cache Update |
| | Mini Project – Deploying a Containerized App on AWS using Amazon EKS |

Date: 8-10-2025

Exercise: AWS Account Setup and Configuration. AWS Console Overview. Enable MFA. Create AWS budget alert.

AWS Console overview / AWS Home page/ AWS Dash board

Widgets – default view/ Add or remove widgets - small panels on the dashboard showing metrics or shortcuts; users can add or remove them as needed.

Region – Specifies the geographical data center location where your AWS resources are deployed.

Services – A categorized list of all AWS offerings such as Compute, Storage, Database, etc.

Search bar – A quick-access bar to search and pin frequently used services for faster access.

pin the most used services to console by clicking on star next to the service name.

Enable MFA

Notes

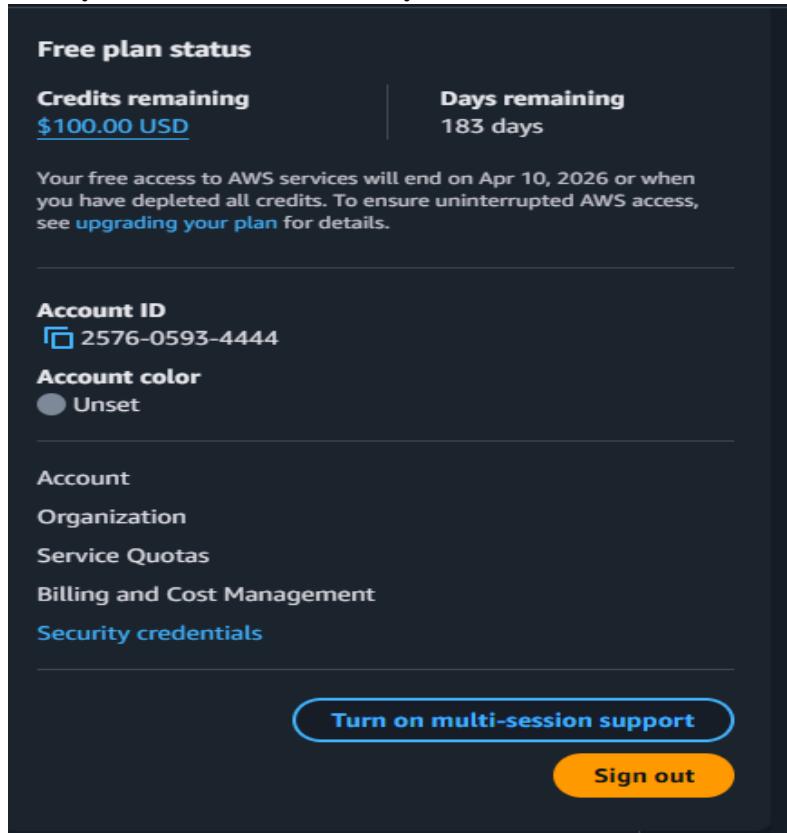
- Make sure your phone is unlocked, Bluetooth is on, and it uses a screen lock (fingerprint/PIN).

Option 1: Add a Passkey for Easier Login

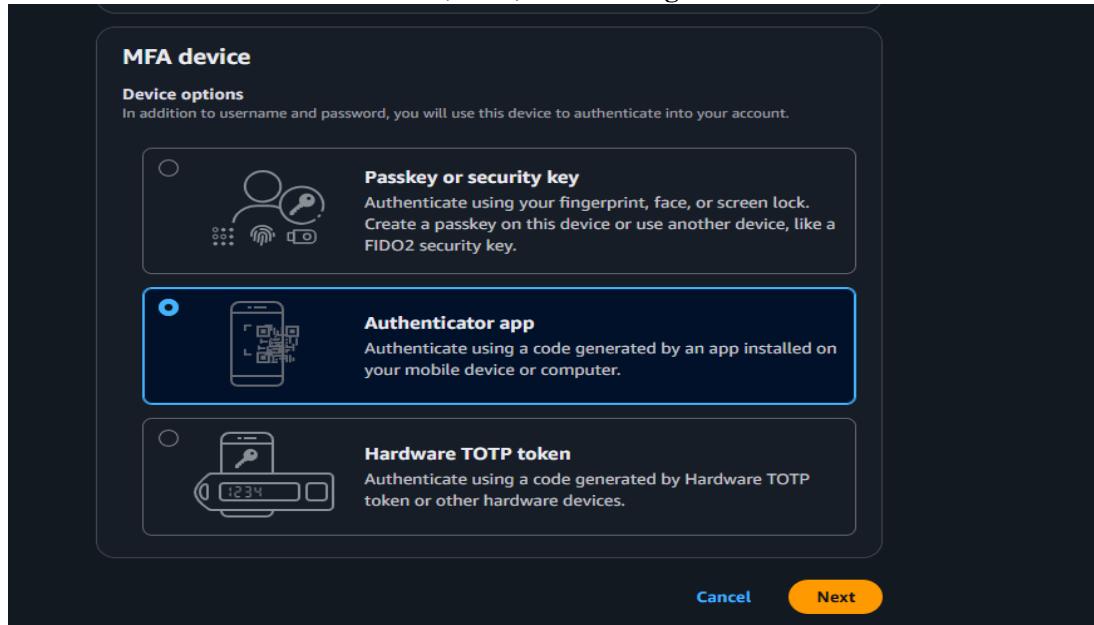
Step1: Go to Security Credentials

- Sign in to AWS console.

- Go to your username → Security credentials.



- Under Multi-factor authentication (MFA) click “Assign MFA device.”



- Choose “Passkeys and security keys” → Next.
- On the next screen choose “Phone or tablet”.

- AWS will show a **browser pop-up** asking to use a device.
 1. Select **your phone** (or “Use another device” if it prompts).
- Look at your phone — you should get a **“Use passkey”** or **biometric prompt**.
- Approve using **fingerprint or phone PIN**.
- Back in AWS, click **Finish**. The passkey is now your MFA method.

Next time you sign in, just choose **“Sign in with a passkey”** → **approve on phone**.

Step 2: Test the Login

- Sign out of AWS.
- Go to the login page.
- Choose “Sign in with a passkey” → Select your phone.
- Approve the prompt on your phone — you should be signed in without any MFA codes.

Option 2: Authenticator App

This uses a 6-digit code from Google Authenticator / Authy / Microsoft Authenticator.

1. In **Security credentials**, click **“Assign MFA device.”**
2. Select **“Authenticator app”** → **Next**.
3. A QR code appears.
4. Open your authenticator app on your phone → **Add account** → **Scan QR code**.
5. The app shows a 6-digit code.
6. Enter that code back in AWS → **Assign MFA**. You’ll use the 6-digit code from the app each time you log in.

The screenshot shows the AWS IAM Security credentials page. A green notification bar at the top states: "MFA device assigned. You can register up to 8 MFA devices of any combination of the currently supported MFA types with your AWS account root and IAM user. With multiple MFA devices, you only need one MFA device to sign in to the AWS console or create a session through the AWS CLI with that user." Below this, the "Multi-factor authentication (MFA) (1)" section lists one entry:

| Type | Identifier | Certifications | Created on |
|---------|------------------------------------|----------------|-----------------|
| Virtual | arn:aws:iam::501168458285:mfa/Amit | Not Applicable | Sat Oct 11 2025 |

Below this, the "Access keys (0)" section indicates "No access keys". It includes a note: "As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials." A "Create access key" button is available.

At the bottom of the page, there are links for "cloudShell", "Feedback", and copyright information: "© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

Create AWS budget alert

Allows to create a simple budget and to send alarms to registered email.

Example: if you are close to or exceeding your designated budget.

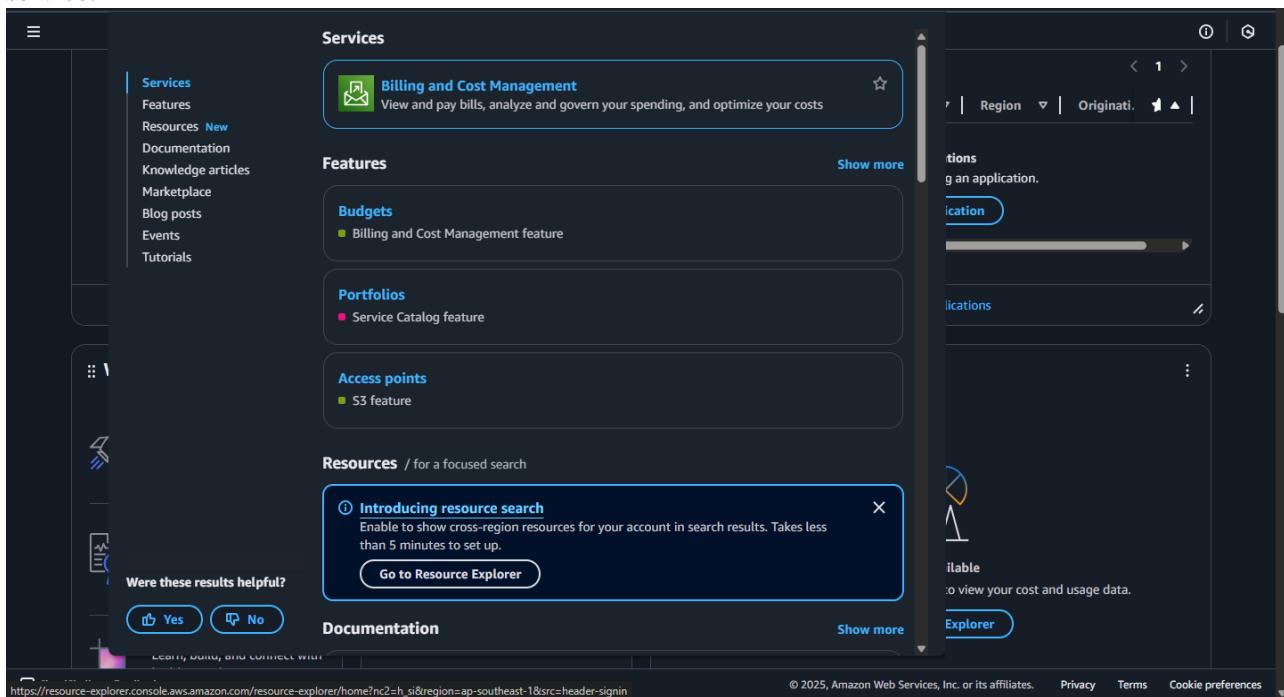
By setting a budget you can monitor budget threshold from the start.

Creating a budget

Step 1:

In the search bar type budgets and under the search results:

Select ‘Budgets’ from the Features group, which is essentially a feature of Billing and Cost Management service.



Step 2: After the ‘Budgets’ page loads

Click on **Create Budget** button

Under Budget setup select ‘Customize’ option

Under Budget types select ‘Cost budget’ option

Click on **Next** button.

Billing and Cost Management > Budgets

AWS Budgets is your hub for creating, tracking, and inspecting your budgets.

Create a budget

Pricing (US)
There is no additional charge for using AWS Budgets. You only pay for configured actions that go beyond the free tier offer of 62 action-enabled budget days.

[View pricing details](#)

Getting started

- What is AWS Budgets?
- Getting started
- Setting up AWS Budgets

More resources

- Documentation
- FAQs

How it works

Benefits and features

- Create and manage budgets**
Set custom cost and usage budgets to easily manage your AWS spend. Monitor your budget status from the AWS Budgets overview.
- Add notifications to your budget**
Get notified by email or publish updates to a Slack channel, Amazon Chime room, or Amazon SNS topic of your choice.
- Refine your budget using filters**
Track your cost or usage across
- Attach actions**
Define actions to reduce or prevent unbudgeted cost before it occurs.

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Billing and Cost Management > Budgets > Create budget

Billing and Cost Management

Choose billing view [New](#)
Primary View

Home
Getting Started
Dashboards [New](#)
Billing and Payments
Bills
Payments
Credits
Purchase Orders
Cost and Usage Analysis
Cost Explorer
Cost Explorer Saved Reports
Cost Anomaly Detection
Free Tier
Data Exports
Customer Carbon Footprint Tool
Cost Organization

Step 1 Choose budget type

- Step 1 Choose budget type
- Step 2 Set your budget
- Step 3 Configure alerts
- Step 4 - Optional Attach actions
- Step 5 Review

Choose budget type

Budget setup

- Use a template (simplified)
Use the recommended configurations. You can change some configuration options after the budget is created.
- Customize (advanced)
Customize a budget to set parameters specific to your use case. You can customize the time period, the start month, and specific accounts.

Billing View
Based on selected billing view.

Budget types

- Cost budget - Recommended**
Monitor your costs against a specified dollar amount and receive alerts when your user-defined thresholds are met. Using cost budgets, the budgeted amount you set represents your expected cloud spend. For example, you can set a cost budget for a business unit and then add additional parameters such as the associated member accounts.
- Usage budget**
Monitor your usage of one or more specified usage types or usage type groups and receive alerts when your user-defined thresholds are met. Using usage budgets, the budgeted amount represents your expected usage. For example, you can use a usage budget to monitor the usage of certain services such as Amazon EC2 and Amazon S3.
- Savings Plans budget**
Track the utilization or coverage associated with your Savings Plans and receive alerts when your percentage drops below a threshold you define. Setting a coverage target lets you see how much of your instance usage is covered by Savings Plans, while setting a utilization target lets you see if your Savings Plans are unused or underutilized.

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 3: Set your budget page loads and fill in the details: MyBudget, Monthly, Recurring budget, set Month and Year, select Fixed – Budgeting method - Enter your budgeted amount – 20.00, select ‘All AWS services’ Scope options. Advanced options – leave it on default.

Click on **Next** button.

Details

Budget name
Provide a descriptive name for this budget.
MyBudget
Names must be between 1-100 characters.

Set budget amount

Period **Monthly**

Budget renewal type
 Recurring budget
 Recurring budgets renew on the first day of every monthly billing period.
 Expiring budget
 Expiring monthly budgets stop renewing at the end of the selected expiration month.

Start month
Oct 2025

Budgeting method **Fixed**
Create a budget that tracks against a single monthly budgeted amount.

Enter your budgeted amount (\$)
Last month's cost: \$0.00
20.00

Budget scope **Info**
Add filtering and use advanced options to narrow the set of cost information tracked as part of this budget

Scope options
 All AWS services (Recommended)
 Track any cost incurred from any service for this account as part of the budget scope
 Filter specific AWS cost dimensions
 Select specific dimensions to budget against. For example, you can select the specific service "EC2" to budget against.

Alerts
No alerts configured.

Billing and Cost Management

Choose billing view **Primary View**

Budgets > Create budget

Enter your budgeted amount (\$)
Last month's cost: \$0.00
20.00

Budget scope **Info**
Add filtering and use advanced options to narrow the set of cost information tracked as part of this budget

Scope options
 All AWS services (Recommended)
 Track any cost incurred from any service for this account as part of the budget scope
 Filter specific AWS cost dimensions
 Select specific dimensions to budget against. For example, you can select the specific service "EC2" to budget against.

Advanced options
 Charge types are now located under Scope options with other filter dimensions. To configure charge types, choose Filter specific AWS cost dimensions, then select Charge type from the Dimension dropdown list.

Aggregate costs by
Unblended costs

Tags (optional)
A tag is a label you assign to an AWS resource. Each tag consists of a key and value. You can use tags to control access to this budget with IAM policies.
No tags associated with the resource.
Add new tag
You can add up to 50 tags.

Budget preview
Cost Data

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 4: Configure alerts

Click on **Add an alert threshold**

Under Set alert threshold

Set Threshold: 80 and Trigger: Actual

Email recipients: enter your email id.

Click on **Next** button.

Billing and Cost Management > Budgets > Create budget

Alert #1

Set alert threshold

Threshold: 80 % of budgeted ... Trigger: Actual

Notification preferences

Email recipients: joshuamnz2.7@gmail.com, joshuamnz221@gmail.com

+ Add alert threshold

Cancel Previous Next

Budget preview

Cost Data

Unable to load chart. An error has occurred with ce:GetCostAndUsage: DataUnavailableException - Cost Explorer was just enabled - it will take some time to prepare your cost and usage data. Please check back in 24 hours.

Alerts

No alerts configured.

Step 5: Under Attach actions – leave it on default

Click on **Next** button.

Billing and Cost Management > Budgets > Create budget

Step 1 Choose budget type
Step 2 Set your budget
Step 3 Configure alerts
Step 4 - Optional Attach actions
Step 5 Review

Attach actions - Optional

Using budgets actions

What is a budget action?

A budget action allows you to define and trigger cost saving responses to reinforce a cost-conscious culture. You have the option to attach actions that run whenever your alert threshold has been exceeded, such as stopping an EC2 instance from incurring any further costs. You can select the alerts to which you would like to attach actions, then define these actions.

How to get started?

To create a budget action, you will first need an alert threshold created from step 2. If you have already created an alert threshold select the type of action you want.

Alert #1 (0 actions attached)

Threshold: 80% Threshold measured against: Actual Costs

Email recipients: joshuamnz2.7@gmail.com, joshuamnz221@gmail.com
Amazon SNS: Not configured

Add action

Cancel Previous Next

Budget preview

Cost Data

Unable to load chart. An error has occurred with ce:GetCostAndUsage: DataUnavailableException - Cost Explorer was just enabled - it will take some time to prepare your cost and usage data. Please check back in 24 hours.

Alerts

Actual cost > 80% | No actions

Step 6: In review page – check and review all the options are set to what is desired.

Click on **Create budget** button.

Now, the budget has been created.

The screenshot shows two consecutive pages from the AWS Billing and Cost Management service.

Top Page (Create budget - Review Step):

- Left Sidebar:** Shows navigation links for Billing and Cost Management, including Home, Getting Started, Dashboards, Bills, Payments, Credits, Purchase Orders, Cost and Usage Analysis, Cost Explorer, Cost Explorer Saved Reports, Cost Anomaly Detection, Free Tier, Data Exports, Customer Carbon Footprint Tool, Cost Organization, Cost Categories, Cost Allocation Tags, Billing Conductor, Budgets and Planning, Savings Plans, and Cost Optimization Hub.
- Central Content:** A five-step wizard:
 - Step 1: Choose budget type**: Set to "Cost budget". Description: "Monitor your costs against a specified dollar amount and receive alerts when your user-defined thresholds are met."
 - Step 2: Set up your budget**:
 - Budget details**: Name: MyBudget, Period: Monthly, Start date: Oct 2025, End date: -, Budget amount: \$20.00.
 - Additional budget parameters**: Tags (Key: Value).
 - Step 3: Configure alerts**:
 - Alerts**: Alert #1: Threshold: 80% of budgeted amount, Threshold measured against: Actual costs.
 - Step 4: Attach actions - optional**: Actions (Edit).
- Bottom Right:** © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

Bottom Page (Overview):

- Left Sidebar:** Same as the top page.
- Central Content:** A summary message: "Your budget MyBudget has been created successfully. After creating a budget, it can take up to 24 hours to populate all of your spend data." Below this is a table titled "Budgets (1) Info":

| Name | Thresholds | Health status | Billing View | Budget | Amount... |
|----------|------------|---------------|--------------|---------|-----------|
| MyBudget | OK | Healthy | Primary View | \$20.00 | - |
- Bottom Right:** © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

Date: 9-10-2025

Exercise: AWS Identity Access Management (IAM) User and Group creation. Enable AWS IAM MFA. Create an AWS Account Alias (for Alternate Sign-in URL)

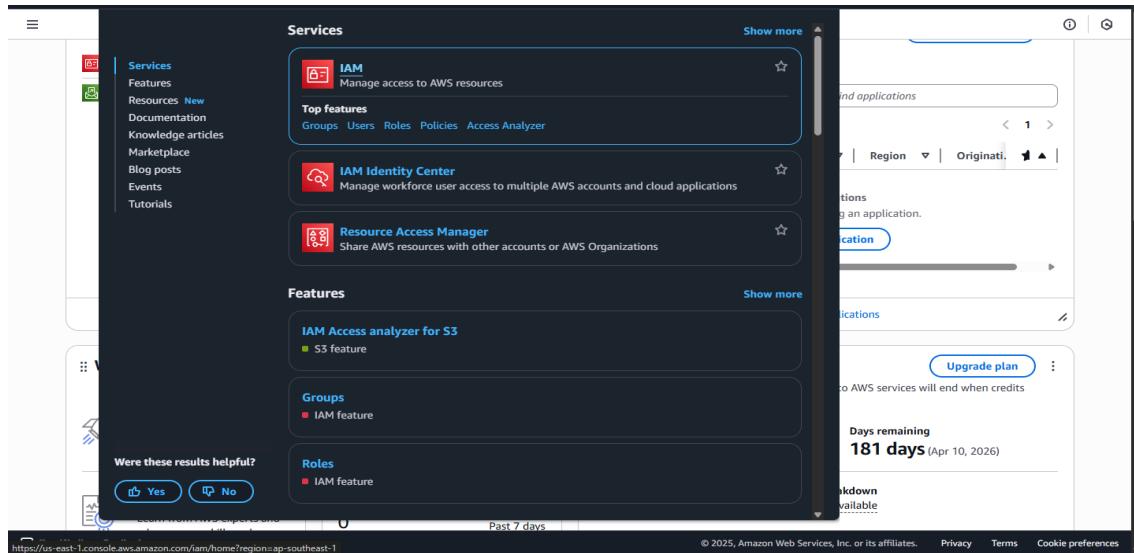
AWS Identity and Access Management (IAM) is a security service that helps you control who can access your AWS resources and what actions they can perform. It is a global AWS service. It allows you to securely manage users, groups, roles, and permissions in your AWS account.

| Concept | Description |
|-----------|--|
| Root User | The account owner who created the AWS account. Has full access and should be used only for account setup. |
| Group | A collection of IAM users that share the same permissions. For example, a “Developers” group or “Students” group. |
| User | A person or application that interacts with AWS (e.g., student1, admin, developer). Each user has its own username, password, and access keys. |
| Policy | A JSON document that defines what actions are allowed or denied (e.g., “allow S3 read access”). |
| Role | A set of permissions that can be temporarily assumed by a user, service, or application — often used by EC2 instances or Lambda functions. |

STEPS for AWS IAM User Group Creation

Step 1: Sign in to AWS Console

- Log in to your AWS Management Console using an administrator account.
- From the Services menu, search for IAM and open it.



The screenshot shows the AWS IAM Dashboard. On the left sidebar, under 'Access management', the 'User groups' link is highlighted with a red box. The main content area displays security recommendations, IAM resources (with counts of 0 for User groups, 0 for Users, 2 for Roles, 0 for Policies, and 0 for Identity providers), and a 'What's new' section. To the right, there are panels for 'AWS Account' (Account ID: 257605934444, Account Alias: Create, Sign-in URL: https://257605934444.signin.aws.amazon.com/console), 'Quick Links' (My security credentials), and 'Tools'.

Step 2: Open Groups Section

- In the IAM dashboard, look at the left sidebar.
- Click on “User groups” → then click “Create group”.

This screenshot is identical to the one above, but the 'User groups' link in the left sidebar is explicitly highlighted with a red box to indicate the step being described in the accompanying text.

The screenshot shows the AWS Identity and Access Management (IAM) console. On the left, there's a navigation sidebar with 'Identity and Access Management (IAM)' selected. Under 'Access management', 'User groups' is also selected. The main area is titled 'User groups (0) Info' and contains a brief description: 'A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.' Below this is a search bar and a table header with columns: Group name, Users, Permissions, and Creation time. A red box highlights the 'Create group' button at the top right of the table area.

Step 3: Name the Group

- Enter a Group name (example: Developers, Admins, Students, etc.).
- Group names must be unique within your account.

This screenshot shows the 'Create user group' wizard. The first step, 'Set group name', is displayed. It has a 'User group name' field containing 'Admins'. Below it is a note: 'Enter a meaningful name to identify this group.' and a character limit note: 'Maximum 128 characters. Use alphanumeric and '+-=_,@-' characters.' The navigation bar at the top shows 'IAM > User groups > Create user group'.

Add users to the group - Optional (0) Info
An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

Attach permissions policies - Optional (4/1076) Info
You can attach up to 10 policies to this user group. All the users in this group will have permissions that are defined in the selected policies.

Filter by Type: All types

| Policy name | Type | Used as | Description |
|---|----------------------------|---------|--|
| <input checked="" type="checkbox"/> AdministratorAccess | AWS managed - job function | None | Provides full access to AWS services an... |
| <input checked="" type="checkbox"/> AdministratorAcces... | AWS managed | None | Grants account administrative permis... |
| <input checked="" type="checkbox"/> AdministratorAcces... | AWS managed | None | Grants account administrative permis... |

Step 4: Attach Permissions Policies

- You can attach IAM policies to define what members of the group can do.
Select the following:
 - AdministratorAccess → Full access to all services.
 - IAMUserChangePassword

The screenshot shows the 'Attach permissions policies' dialog in the AWS IAM console. The policy list includes:

| Policy name | Type | Used as | Description |
|-----------------------|----------------------------|---------|--|
| AdministratorAccess | AWS managed - job function | None | Provides full access to AWS services an... |
| AdministratorAccess | AWS managed | None | Grants account administrative permis... |
| AdministratorAccess | AWS managed | None | Grants account administrative permis... |
| AIOpsAssistantInc... | AWS managed | None | Provides permissions required by the A... |
| AIOpsAssistantPolicy | AWS managed | None | Provides ReadOnly permissions requir... |
| AIOpsConsoleAdmi... | AWS managed | None | Grants full access to Amazon AI Opera... |
| AIOpsOperatorAccess | AWS managed | None | Grants access to the Amazon AI Opera... |
| AIOpsReadOnlyAcc... | AWS managed | None | Grants ReadOnly permissions to the A... |
| AlexaForBusinessD... | AWS managed | None | Provide device setup access to AlexaFo... |
| AlexaForBusinessF... | AWS managed | None | Grants full access to AlexaForBusiness ... |
| AlexaForBusinessG... | AWS managed | None | Provide gateway execution access to A... |
| AlexaForBusinessLi... | AWS managed | None | Provide access to Lifesize AVS devices |
| AlexaForBusinessP... | AWS managed | None | Provide access to Poly AVS devices |

The screenshot shows the 'Attach permissions policies' dialog in the AWS IAM console. The policy list includes:

| Policy name | Type | Used as | Description |
|-----------------------|-------------|---------|--|
| GlobalAcceleratorR... | AWS managed | None | Allow GlobalAccelerator Users Access t... |
| GreengrassOTAUpd... | AWS managed | None | Provides read access to the Greengrass... |
| IAMAccessAdvisorR... | AWS managed | None | This policy grants access to read all acc... |
| IAMAccessAnalyzer... | AWS managed | None | Provides full access to IAM Access Anal... |
| IAMAccessAnalyzer... | AWS managed | None | Provides read only access to IAM Acces... |
| IAMAuditRootUser... | AWS managed | None | Provides access required to check the ... |
| IAMCreateRootUser... | AWS managed | None | Provides access required to create a ro... |
| IAMDeleteRootUser... | AWS managed | None | Provides access required to delete all r... |
| IAMFullAccess | AWS managed | None | Provides full access to IAM via the AW... |
| IAMReadOnlyAccess | AWS managed | None | Provides read only access to IAM via th... |
| IAMSelfManageSer... | AWS managed | None | Allows an IAM user to manage their o... |
| IAMUserChangePas... | AWS managed | None | Provides the ability for an IAM user to ... |

Step 5: Review and Create

- Review all details (group name + permissions).
- Click “Create group” to finalize.

Group Successfully Created

- The new group now appears in the IAM dashboard.
- Any user added to this group automatically inherits all permissions attached to the group.

The screenshot shows the AWS IAM User groups page. At the top, a green banner displays the message "Admins user group created." Below the banner, the heading "User groups (1) Info" is shown. A sub-header explains that a user group is a collection of IAM users. The main table lists one group named "Admins". The table columns are "Group name", "Users", "Permissions", and "Creation time". The "Group name" column shows "Admins", the "Users" column shows "0", the "Permissions" column shows "Defined", and the "Creation time" column shows "1 minute ago". On the left sidebar, under "Access management", the "User groups" section is expanded, showing options like "Users", "Roles", "Policies", "Identity providers", "Account settings", and "Root access management". Other sections like "Access reports" are also listed. At the bottom of the page, there are links for "CloudShell", "Feedback", and copyright information: "© 2025, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".

STEPS for AWS IAM User Creation

Step 1: Sign in to AWS Management Console

- Login to the AWS Management Console using your root user credentials.
- In the search bar, type IAM and open IAM service.

Step 2: Navigate to Users Section

- In the IAM dashboard's left panel (info panel), click on Users.
- Then click on the "Add users" button.

Step 3: Set User Details

- Enter a user name.
- Checkbox – 'Provide user access to the AWS Management Console'.
- Select 'I want to create an IAM user' option

Screenshot of the AWS IAM 'Create user' wizard Step 4: Set permissions.

The page shows the following steps:

- Step 2: Set permissions
- Step 3: Review and create
- Step 4: Retrieve password

User details

User name: John_Doe

Provide user access to the AWS Management Console - optional

If you're providing console access to a person, it's a best practice [to manage their access in IAM Identity Center](#).

Are you providing console access to a person?

User type:

- Specify a user in Identity Center - Recommended
- I want to create an IAM user

We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

Console password

Autogenerated password

You can view the password after you create the user.

Custom password

Enter a custom password for the user.

Show password

Users must create a new password at next sign-in - Recommended

Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 4: Set Permissions

You can grant permissions to the new user in three ways:

1. **Add user to group** – Assign predefined permission groups.
2. Copy permissions from an existing user.
3. Attach existing policies directly.

Recommended: Use IAM **groups** to manage permissions easily.

The screenshot shows the 'Set permissions' step of the IAM user creation wizard. The left sidebar shows steps 1 through 4. Step 2 ('Specify user details') is completed. Step 3 ('Review and create') is selected. Step 4 ('Retrieve password') is not yet started. The main area is titled 'Set permissions' with a sub-section 'Permissions options'. It contains three options: 'Add user to group' (selected), 'Copy permissions', and 'Attach policies directly'. Below this is a table titled 'User groups (1)' showing one group named 'Admins' created 13 minutes ago with policies like 'AdministratorAccess' and 'IAMUserChange...'. A 'Create group' button is available. At the bottom are 'Cancel', 'Previous', and 'Next' buttons, with 'Next' highlighted in orange.

Step 5: Set User Details and Tags (Optional)

- Add tags like Department: MCA or Role: Faculty/Student for easy identification.
- Click Next to review.

The screenshot shows the 'Review and create' step of the IAM user creation wizard. The left sidebar shows steps 1 through 4. Step 1 ('Specify user details') is completed. Step 2 ('Set permissions') is selected. Step 3 ('Review and create') is selected. Step 4 ('Retrieve password') is not yet started. The main area has three sections: 'User details' (User name: John_Doe, Console password type: Custom password, Require password reset: Yes), 'Permissions summary' (IAMUserChangePassword policy, AWS managed, Permissions policy), and 'Tags - optional' (Key: MCA, Value: AWS). A 'Create user' button is at the bottom.

Step 6: Review and Create User

- Review all settings (user name, permissions, tags).

- Click Create user.

Step 7: Save Credentials

- After creation, AWS displays:
 - Console sign-in URL
 - Username
 - Password

User created successfully

You can view and download the user's password and email instructions for signing in to the AWS Management Console.

[View user](#)

Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Step 4
Retrieve password

Retrieve password

You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

Console sign-in details

Console sign-in URL
<https://257605934444.sigin.aws.amazon.com/console>

User name
[John_Doe](#)

Console password
[*****](#) [Show](#)

[Email sign-in instructions](#)

[Cancel](#) [Download .csv file](#) [Return to users list](#)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

NOTE: Download or copy these credentials immediately — they cannot be retrieved later.

Step 8: Test the User Login

- Visit the IAM login URL provided (unique for your AWS account).
- Log in with the newly created username and password.
- Verify access and permissions.

The screenshot shows the AWS Console Home page. On the left, there's a 'Recently visited' section with a placeholder icon and a message 'No recently visited services'. Below it, a 'Explore one of these commonly visited AWS services.' section lists EC2, S3, Aurora and RDS, and Lambda. A 'View all services' link is at the bottom. To the right, there's a 'Applications' section with a 'Create application' button, a 'Select Region' dropdown set to 'eu-north-1 (Current Region)', and a search bar. A red box highlights an error message: 'Access denied to servicecatalog>ListApplications' with a 'Diagnose with Amazon Q' button. Below this is a 'Cost and usage' section with 'Current month' and 'Forecasted month end' status, both showing 'Access denied'. At the bottom, there are links for CloudShell, Feedback, and a footer with copyright information.

Enable AWS IAM MFA

Step 1: Sign in to AWS Console as root user

Step 2: Open IAM Dashboard

- In the search bar, type IAM and select IAM (Identity and Access Management).
- From the left navigation pane, select Users.

Step 3: Select a User

- Click the user name for whom you want to enable MFA.
- This opens the User Summary page.

The screenshot shows the AWS IAM User Details page for a user named 'John_Doe'. The left sidebar contains navigation links for Identity and Access Management (IAM), Access management, Access reports, and IAM Identity Center. The main content area displays the user's ARN, creation date, console access status (Enabled without MFA), and access key information. Below this, the 'Permissions' tab is selected, showing one attached policy: 'IAMUserChangePassword' (AWS managed). Other tabs include Groups, Tags (1), Security credentials, and Last Accessed.

Step 4: Go to Security Credentials Tab

- Click the Security credentials tab.
- Scroll down to the section “Multi-factor authentication (MFA)”.

The screenshot shows the 'Security credentials' tab for the 'John_Doe' user. It includes sections for 'Console sign-in' (with a 'Manage console access' button) and 'Multi-factor authentication (MFA)' (0). The MFA section notes that no devices are assigned and provides a link to 'Assign MFA device'.

Step 5: Assign MFA Device

- Click “Assign MFA device”.
- Choose the MFA type:
 - Virtual MFA device (e.g., Google Authenticator, Authy — most common)
 - Security key (hardware-based, e.g., YubiKey)
 - Authenticator app on phone

Step 6: Configure Virtual MFA

- If you select Virtual MFA device:
 1. Open your Google Authenticator or Authy app on your phone.
 2. Scan the QR code shown on the AWS screen.
 3. The app starts generating 6-digit codes.

Step 7: Verify MFA

- Enter two consecutive codes from your app in the verification fields.
- Click “Assign MFA”.

Step 8: Confirm Setup

- You will see a green checkmark confirming MFA is successfully assigned.
- The user now requires MFA each time they sign in.

The screenshot shows the AWS IAM User Details page for a user named 'John_Doe'. On the left, there's a navigation pane with 'Identity and Access Management (IAM)' selected. The main area displays the user's ARN (arn:aws:iam::257605934444:user/John_Doe), creation date (October 13, 2025, 11:57 (UTC+05:30)), and console access status (Enabled with MFA). A green success message at the top states: 'MFA device assigned. You can register up to 8 MFA devices of any combination of the currently supported MFA types with your AWS account root and IAM user. With multiple MFA devices, you only need one MFA device to sign in to the AWS console or create a session through the AWS CLI with that user.' Below the summary, there are tabs for 'Permissions', 'Groups', 'Tags (1)', 'Security credentials', and 'Last Accessed'. The 'Security credentials' tab is active, showing 'Console sign-in' details (Console sign-in link: https://257605934444.signin.aws.amazon.com/console) and 'Multi-factor authentication (MFA)' details (Console password updated 7 minutes ago). At the bottom, there are buttons for 'Remove', 'Resync', and 'Assign MFA device'.

Create an AWS Account Alias (for Alternate Sign-in URL)

As an IAM user you can sign in using the default URL or create an account alias for it.

An Account Alias gives your AWS account a name instead of using the long numeric Account ID in your sign-in URL.

This makes it easier for IAM users to remember and log in.

Step 1:

- Sign in to the AWS Management Console using root user or an IAM user with administrative privileges.
- In the search bar, type IAM, and open the IAM service.

Step 2:

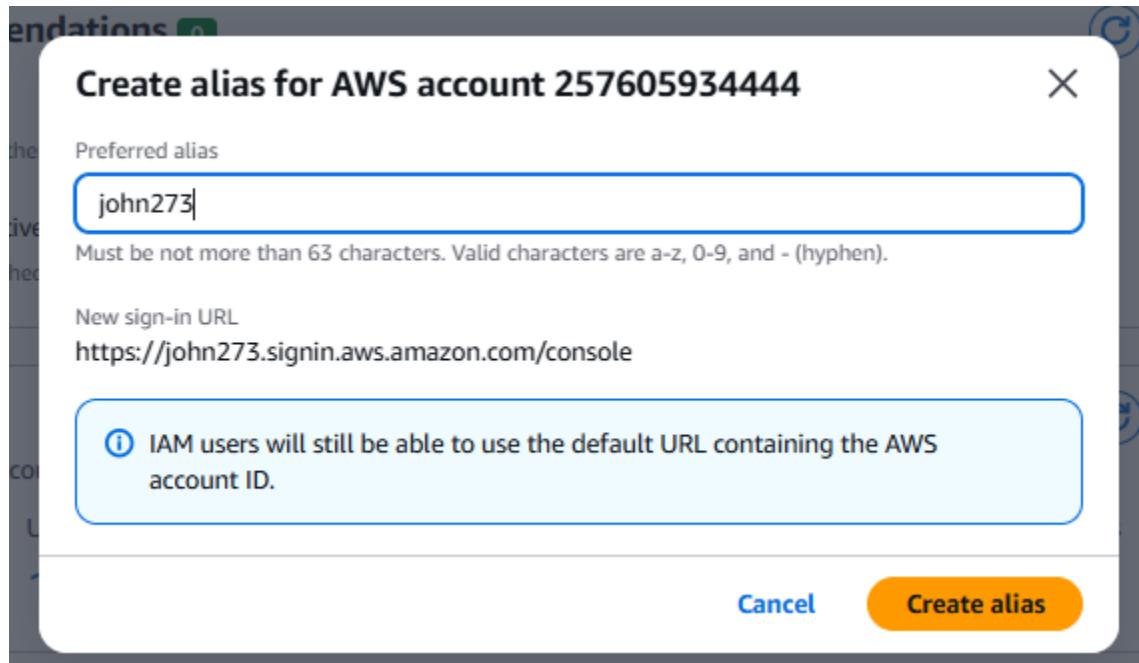
- In the left navigation pane, scroll down and select Dashboard.

Step 3:

- Under the “AWS Account” section, find “Account Alias”.
- Click on “Create” (or “Edit” if one already exists).

Step 4:

- In the pop-up box, enter your preferred alias name.
- Click “Create alias”.



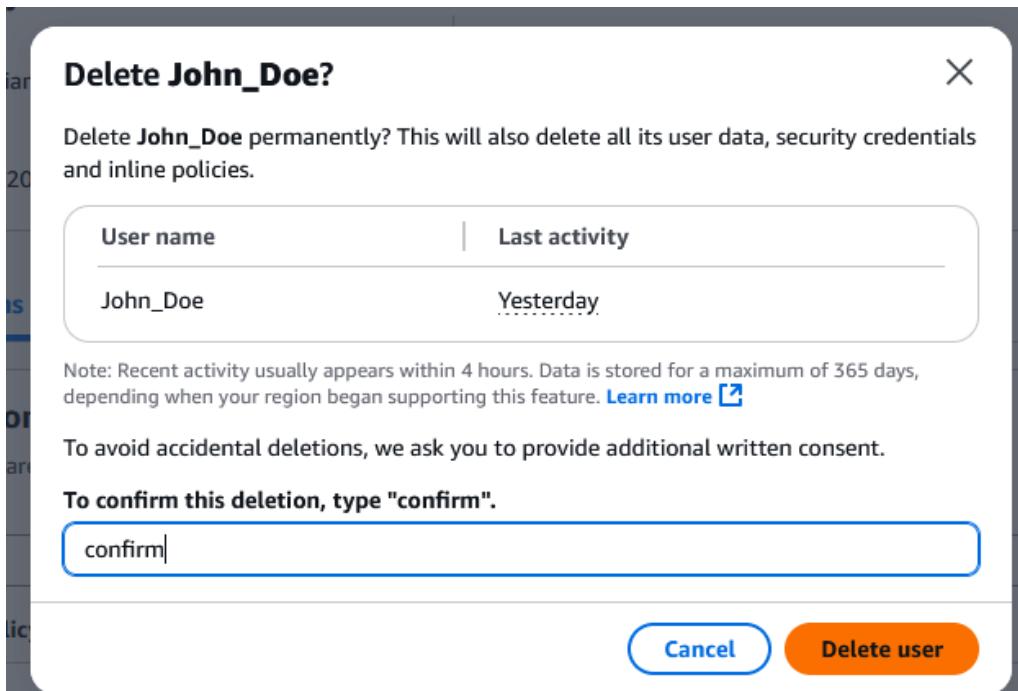
Step 5:

- Once created, you'll see a new Sign-in URL displayed.

The screenshot shows the "AWS Account" settings page. It displays the "Account ID" as "257605934444" and the "Account Alias" as "john273". Below these, under "Sign-in URL for IAM users in this account", it shows the URL "https://john273.signin.aws.amazon.com/console".

To Delete the resources:

The screenshot shows the AWS IAM User Details page for a user named "John_Doe". The top navigation bar includes the AWS logo, search bar, and account information (Account ID: 2576-0593-4444, Global, Joshua). The left sidebar has sections for Identity and Access Management (IAM), Access management (with sub-options like User groups, Users, Roles, Policies, Identity providers, Account settings, Root access management), and Access reports (with sub-options like Access Analyzer, Resource analysis, Unused access, Analyzer settings, Credential report, Organization activity, Service control policies, Resource control policies). The main content area is titled "John_Doe Info" and contains a "Summary" section with details like ARN, Console access (Enabled with MFA), Created date (October 13, 2025, 11:57 (UTC+05:30)), Last console sign-in (Yesterday), and Access key 1 (with a "Create access key" link). Below the summary is a tab-based navigation: Permissions (selected), Groups, Tags (1), Security credentials, and Last Accessed. The "Permissions" tab shows "Permissions policies (1)" attached to the user. A table lists the policy: Policy name: IAMUserChangePassword, Type: AWS managed, Attached via: Directly. There are buttons for "Remove" and "Add permissions". Below the table are sections for "Permissions boundary (not set)" and "Generate policy based on CloudTrail events".



Date: 14-10-2025

Exercise-3: Amazon S3 – Introduction, Bucket Creation and upload objects (files).

Amazon S3 (Simple Storage Service) is a fully managed, object-based storage service offered by AWS. It allows you to store and retrieve unlimited amounts of data from anywhere on the web.

Unlike traditional file systems that store data in folders, S3 stores data as objects inside buckets.

Each object has:

- Data (the actual file),
- Metadata (information about the file),
- Unique key (its name within the bucket).

You can store unlimited data in Amazon S3.

However, each AWS account can create up to 100 buckets by default (can be increased by request).

Each object (file) can be up to 5 TB (terabytes) in size.

Single upload limit: 5 GB (may vary), but larger files can be uploaded using Multipart Upload.

Amazon S3 is designed for:

- Durability: 99.99999999% (11 nines) – this means your data is extremely safe.
- Availability: 99.99% uptime – your data is almost always accessible.
- S3 automatically stores multiple copies of your data across multiple Availability Zones in a region.

Storage classes: S3 offers different storage classes depending on cost and frequency of access.

| Storage Class | Description |
|-------------------------|--|
| S3 Standard | For frequently accessed data (default). |
| S3 Intelligent-Tiering | Automatically moves data between frequent - infrequent access tiers. |
| S3 Standard-IA | For infrequently accessed data but still quickly available. |
| S3 Glacier | For archival storage (low cost, slower retrieval). |
| S3 Glacier Deep Archive | For long-term archival (lowest cost). |

Each object can be accessed through a **unique URL**.

S3 is commonly used for:

- Backup and archival
- Static website hosting
- Application data storage
- Media content delivery

| Feature | Description |
|------------------------|--|
| Buckets | Top-level containers for storing objects. |
| Objects | Actual files stored in S3 (e.g., images, HTML, PDFs). |
| Region | Each bucket is created in a specific AWS region. |
| Versioning | Maintains multiple versions of the same object for recovery. |
| Static Website Hosting | Allows you to host HTML pages directly from an S3 bucket. |

Steps to Create an S3 Bucket and Upload an Image

Step 1: Open the S3 Service

- Sign in to your AWS Management Console.
- In the search bar, type S3 and click Amazon S3.

The screenshot shows the Amazon S3 landing page. At the top, there's a dark header with the AWS logo and the text "Storage". Below it, the main title is "Amazon S3" with the subtitle "Store and retrieve any amount of data from anywhere". A descriptive paragraph states: "Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance." To the right, there's a call-to-action box titled "Create a bucket" containing the text: "Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored." Below this is a large orange "Create bucket" button. Further down, there are sections for "How it works" (with a video thumbnail), "Pricing" (mentioning no minimum fees), and "Resources" (linking to the User guide). The bottom of the page includes standard AWS footer links like CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Step 2: Create a New Bucket

- Click “Create bucket”.
- Bucket type: General purpose
- Bucket name: Enter a name (Bucket names must be globally unique and lowercase.)
- AWS Region: Choose the region nearest to you.
- Scroll down and uncheck:
“Block all public access” → Uncheck this (for viewing file publicly).
Confirm the warning checkbox.
- Keep all other settings as default.
- Click Create bucket.

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region: Asia Pacific (Mumbai) ap-south-1

Bucket type: General purpose
General purpose buckets are for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Bucket name: Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn more](#)

Copy settings from existing bucket - optional
Choose the settings from the following configuration are copied:

Format: [s3://bucketname](#)

Object Ownership Info
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Object Ownership
 ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
 ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced

Block Public Access settings for this bucket
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through access control lists (ACLs)**
AWS recommends that you turn on Block all public access, unless public access is required for specific and verified use cases such as static website hosting.
- Block public access to buckets and objects granted through bucket policies**
Bucket policies are used to deny public access to buckets and prevent them from being publicly accessible.
- Block public access to buckets and objects granted through access point policies**
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Bucket Versioning
Versioning allows keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning
 Disable
 Enable

Tags - optional (0)
You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

You can add up to 50 tags.

Default encryption Info
Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type: Info
By default, S3 uses separate layers of encryption. For details on pricing, see [DSE-KMS pricing](#) on the Storage tab of the [Amazon S3 pricing page](#).

Server-side encryption with Amazon S3 managed keys (SSE-S3)
 Server-side encryption with AWS Key Management Service keys (SSE-KMS)
 Dual-layer server-side encryption with AWS Key Management Service keys (DSE-KMS)

Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSE-KMS. [Learn more](#)

Disable
 Enable

Advanced settings

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Step 3: Upload an Image File

1. Click on the newly created bucket name.
2. Click Upload → Add files.
3. Choose any image file from your computer.
4. Scroll down and click Upload.

Amazon S3 > Buckets > josh-bucket-1ms24mc040

josh-bucket-1ms24mc040 [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

Objects (0)

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

[Upload](#)

| Name | Type | Last modified | Size | Storage class |
|--|------|---------------|------|---------------|
| No objects | | | | |
| You don't have any objects in this bucket. | | | | |
| Upload | | | | |

Amazon S3 > Buckets > josh-bucket-1ms24mc040 > Upload

Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDKs or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose Add files or Add folder.

Files and folders (1 total, 69.6 KB)

All files and folders in this table will be uploaded.

| Name | Folder | Type | Size |
|----------------------------------|--------|-----------|---------|
| Screenshot 2024-04-08 122709.png | - | image/png | 69.6 KB |

[Remove](#) [Add files](#) [Add folder](#)

Destination [Info](#)

Destination
[s3://josh-bucket-1ms24mc040](#)

Destination details
Bucket settings that impact new objects stored in the specified destination.

Permissions
Grant public access and access to other AWS accounts.

Properties
Specify storage class, encryption settings, tags, and more.

[Cancel](#) [Upload](#)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Upload succeeded
For more information, see the [Files and folders](#) table.

Upload: status

After you navigate away from this page, the following information is no longer available.

Summary

| Destination | Succeeded | Failed |
|---|---------------------------|-------------------|
| s3://josh-bucket-1ms24mc040 | 1 file, 69.6 KB (100.00%) | 0 files, 0 B (0%) |

[Close](#)

Files and folders [Configuration](#)

Files and folders (1 total, 69.6 KB)

| Name | Folder | Type | Size | Status | Error |
|----------------------------------|--------|-----------|---------|-----------|-------|
| Screenshot 2024-04-08 122709.png | - | image/png | 69.6 KB | Succeeded | - |

Step 4: Make the Object Public

By default, S3 objects are private. To view them in a browser, make the file public.

1. After upload, go to the Objects tab inside your bucket.
2. Select the uploaded file.
3. Click Actions → Make public using ACL (or Object actions → Make public, depending on console version).
4. Confirm.

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, search bar, and account information (Account ID: 2576-0593-4444, Region: Asia Pacific (Mumbai), User: Joshua). Below the navigation is a breadcrumb trail: Amazon S3 > Buckets > josh-bucket-1ms24mc040. The main area is titled "josh-bucket-1ms24mc040" with a "Info" link. There are tabs for Objects, Properties, Permissions, Metrics, Management, and Access Points. The "Objects" tab is active, showing a list of one object: "Screenshot 2024-04-08 122709.png" (png type, 15.3 KB, last modified October 15, 2025, 15:12:33 (UTC+05:30)). To the right of the list is a "Actions" button, which has a dropdown menu open. The menu includes options like Download as, Share with a presigned URL, Calculate total size, Copy, Move, Initiate restore, Query with S3 Select, Edit actions, Rename object, Edit storage class, Edit server-side encryption, Edit tags, and "Make public using ACL". The "Make public using ACL" option is highlighted with a red rectangle. At the bottom of the page, there are links for CloudShell, Feedback, and a footer with copyright information (© 2025, Amazon Web Services, Inc. or its affiliates) and links for Privacy, Terms, and Cookie preferences.

The screenshot shows the 'Make public' dialog box. At the top, there's a note: 'When public read access is enabled and not blocked by Block Public Access settings, anyone in the world can access the specified objects.' Below this is a table titled 'Specified objects' with one item: 'Screenshot 2024-04-08 122709.png'. The table includes columns for Name, Type, Last modified, Size, and Error. The 'Type' column shows 'png', 'Last modified' shows 'October 15, 2025, 15:12:33 (UTC+05:30)', and 'Size' shows '69.6 KB'. At the bottom right are 'Cancel' and 'Make public' buttons. A green success message box at the top says 'Successfully edited public access'.

Step 5: Copy the Object URL

1. Open the object again by clicking its name.
2. Scroll down to the **Object URL** section.
3. Copy this URL and open it in a new browser tab.

The image is displayed directly from the S3 bucket — meaning AWS S3 is serving that object over HTTP.

The screenshot shows the AWS S3 object details page for 'bp-lot-2-rottweiler-couleur-portrait.png'. The left sidebar shows the bucket structure. The main panel has tabs for 'Properties', 'Permissions', and 'Versions'. Under 'Properties', the 'Object overview' section displays the following information:

- Owner:** eec451edb9f4bc8c8f5dd7f40f93da0af279e09b93bbfc30fce035531706da5
- AWS Region:** Asia Pacific (Mumbai) ap-south-1
- Last modified:** October 15, 2025, 15:25:59 (UTC+05:30)
- Size:** 8.7 MB
- Type:** png
- Key:** bp-lot-2-rottweiler-couleur-portrait.png

On the right side, there are several links:

- S3 URI:** <https://s3.josh-bucket-1ms24mc040.amazonaws.com/bp-lot-2-rottweiler-couleur-portrait.png>
- Amazon Resource Name (ARN):** [arn:aws:s3:::josh-bucket-1ms24mc040/bp-lot-2-rottweiler-couleur-portrait.png](#)
- Entity tag (Etag):** [671d6683e29b92ee1b70484313b9b6c6](#)
- Object URL:** <https://josh-bucket-1ms24mc040.s3.ap-south-1.amazonaws.com/bp-lot-2-rottweiler-couleur-portrait.png>



Date: 15-10-2025

Exercise-4: Amazon S3 – Static Website Hosting (Multi-Page website), Versioning, Cross-Region Replication rule.

Amazon S3 Static Website Hosting

Amazon S3 can host a static website – [a website consisting of only HTML, CSS, JavaScript, images, etc. – no server-side scripting like PHP or Python].

When you enable “Static Website Hosting,” your S3 bucket acts like a web server, and AWS provides a public website URL to access it.

You can create a multi-page static website (e.g., index.html, about.html, contact.html) and upload it to S3. Links within these pages allow users to navigate between them just like a normal website.

Steps to Create a Multi-Page Static Website on S3

Step 1: Create an S3 Bucket

- Open the AWS Management Console → Navigate to S3.
- Click Create bucket.
- Select Bucket type: General purpose
- Enter a unique bucket name (e.g., my-static-web-demo).
- Uncheck “Block all public access.”
- Click Create bucket.

Step 2: Prepare Website Files

Before uploading, organize your files in a folder structure as follows:

```
my-website/
|
├── index.html
├── about.html
├── contact.html
├── error.html
└── images/
    └── banner.jpg
```

Each HTML file should include navigation links.

Step 3: Upload Website Files

- Open your S3 bucket → Click Upload.
- Add all files and folders (HTML, CSS, JS, images).
- Click Upload to store them in S3.

Upload succeeded
For more information, see the [Files and folders table](#).

After you navigate away from this page, the following information is no longer available.

| Summary | |
|-----------------------------|--|
| Destination | Status |
| s3://josh-bucket-1ms24mc040 | Succeeded 7 files, 38.8 KB (100.00%) |
| | Failed 0 files, 0 B (0%) |

Files and folders Configuration

Files and folders (7 total, 38.8 KB)

| Name | Folder | Type | Size | Status | Error |
|--------------|--------------------|------------------|---------|-----------|-------|
| about.html | my-website/ | text/html | 1.1 KB | SUCCEEDED | - |
| contact.html | my-website/ | text/html | 1.9 KB | SUCCEEDED | - |
| error.html | my-website/ | text/html | 976.0 B | SUCCEEDED | - |
| banner.jpg | my-website/images/ | image/jpeg | 32.7 KB | SUCCEEDED | - |
| index.html | my-website/ | text/html | 1.2 KB | SUCCEEDED | - |
| policy.json | my-website/ | application/json | 228.0 B | SUCCEEDED | - |
| README.txt | my-website/ | text/plain | 799.0 B | SUCCEEDED | - |

Step 4: Enable Static Website Hosting

- Go to the Properties tab of the bucket.
- Scroll down to Static website hosting → Click Edit.
- Choose Enable and select ‘Host a static website’.
- Set:
 - Index document: index.html
 - Error document: error.html
- Click Save changes.

[Amazon S3](#) > [Buckets](#) > [josh-bucket-1ms24mc040](#)

josh-bucket-1ms24mc040 [Info](#)

Properties Objects Permissions Metrics Management Access Points

Bucket overview

AWS Region: Asia Pacific (Mumbai) ap-south-1

Amazon Resource Name (ARN): arn:aws:s3:::josh-bucket-1ms24mc040

Creation date: October 15, 2025, 15:04:04 (UTC+05:30)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning: Disabled

Multi-factor authentication (MFA) delete

An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. [Learn more](#)

Disabled

Static website hosting

Edit

Use this bucket to host a website or redirect requests. [Learn more](#)

 We recommend using AWS Amplify Hosting for static website hosting

Deploy a fast, secure, and reliable website quickly with AWS Amplify Hosting. Learn more about [Amplify Hosting](#) or [View your existing Amplify apps](#)

[Create Amplify app](#)

S3 static website hosting

Disabled

 CloudShell [Feedback](#)

[Privacy](#) [Terms](#) [Cookie preferences](#)

© 2025, Amazon Web Services, Inc. or its affiliates.

Edit static website hosting Info

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting

- Disable
 Enable

Hosting type

- Host a static website

Use the bucket endpoint as the web address. [Learn more](#)

- Redirect requests for an object

Redirect requests to another bucket or domain. [Learn more](#)

For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#)

Index document

Specify the home or default page of the website.

index.html

Error document - *optional*

This is returned when an error occurs.

error.html

Step 5: Make Files Public (Bucket Policy)

By default, your files are private. To make them public:

- Go to the Permissions tab → Bucket Policy → Edit.
- Paste the following policy (replace bucket name):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-static-web-demo/*"
    }
  ]
}
```

- Save the changes.

Amazon S3 > Buckets > josh-bucket-1ms24mc040

josh-bucket-1ms24mc040 [Info](#)

Objects Properties **Permissions** Metrics Management Access

Permissions overview

Access finding
Access findings are provided by IAM external access analyzers. Learn more about [How IAM analyzer findings work](#). [View analyzer for ap-south-1](#)

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
⚠ Off
▶ Individual Block Public Access settings for this bucket

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

No policy to display. [Edit](#) [Delete](#) [Copy](#)

CloudShell Feedback Privacy Terms Cookie preferences © 2025, Amazon Web Services, Inc. or its affiliates.

The screenshot shows the 'Edit bucket policy' page in the AWS S3 console. The policy document is as follows:

```

1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Principal": "*",
7        "Action": "s3:GetObject",
8        "Resource": "arn:aws:s3:::josh-bucket-1ms24mc040/*"
9      }
10     ]
11   }
12 }
13

```

On the right side of the editor, there are sections for 'Edit statement' (with a 'Remove' button), 'Add actions' (with a 'Choose a service' dropdown containing 'S3'), 'Available' services (including AI Operations, AMP, API Gateway, API Gateway V2, ARC Region switch, ARC Zonal Shift, ASC), and buttons for 'Add a resource' and 'Add a condition (optional)'. At the bottom, there are 'Cancel' and 'Save changes' buttons.

Step 6: Access Your Website

- Go to the Properties tab → Scroll to Static website hosting.
- Copy the Bucket Website Endpoint URL.
- Paste it into your browser — your homepage (index.html) should appear.
- Use the header links to navigate between pages (About, Contact, etc.).

When Will the Error Page Be Shown?

If a user enters a wrong URL or tries to access a file that doesn't exist (e.g., /abc.html), Amazon S3 automatically displays the file you set as the **Error document** (error.html).

Amazon S3 > Buckets > josh-bucket-1ms24mc040

josh-bucket-1ms24mc040 [Info](#)

Objects [Properties](#) Permissions Metrics Management Access Points

Bucket overview

| | | |
|--|---|---|
| AWS Region Asia Pacific (Mumbai) ap-south-1 | Amazon Resource Name (ARN) arn:aws:s3:::josh-bucket-1ms24mc040 | Creation date October 15, 2025, 15:04:04 (UTC+05:30) |
|--|---|---|

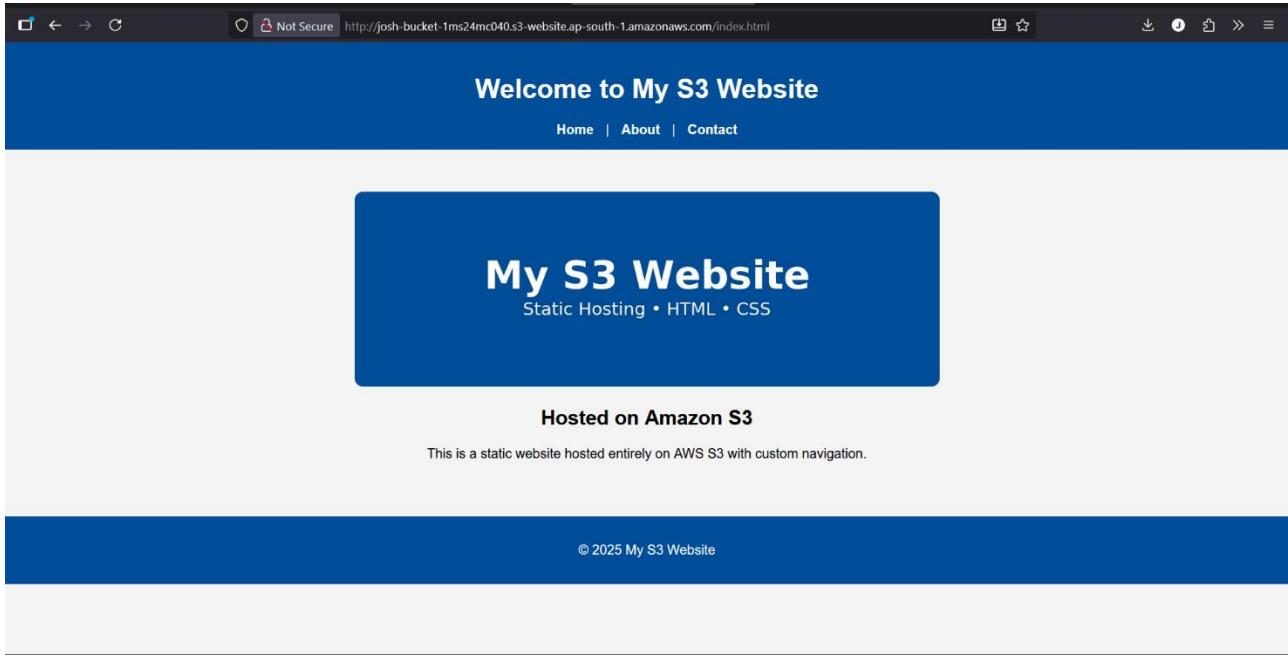
Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning
Disabled

Multi-factor authentication (MFA) delete
An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. [Learn more](#)

Bucket website endpoint
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)
<http://josh-bucket-1ms24mc040.s3-website.ap-south-1.amazonaws.com>



Amazon S3 Versioning

Versioning allows you to keep multiple versions of an object in a bucket. If a file is accidentally deleted or overwritten, you can recover the previous version. Each version gets a unique version ID.

Steps to Enable Versioning

- Go to your S3 bucket
- Open the Properties tab
- Scroll to Bucket Versioning
- Click Edit → Enable
- Click Save changes

Now whenever you upload a file with the same name, S3 will keep both versions.

Amazon S3 > Buckets > josh-bucket-1ms24mc040

josh-bucket-1ms24mc040 Info

Objects Properties Permissions Metrics Management Access Points

Bucket overview

AWS Region: Asia Pacific (Mumbai) ap-south-1 Amazon Resource Name (ARN): arn:aws:s3::josh-bucket-1ms24mc040 Creation date: October 15, 2025, 15:04:04 (UTC+05:30)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning: Disabled

Multi-factor authentication (MFA) delete

An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. [Learn more](#)

Tags (0)

You can use bucket tags to track storage costs and manage buckets. [Learn more](#)

Amazon S3 > Buckets > josh-bucket-1ms24mc040 > Edit Bucket Versioning

Edit Bucket Versioning Info

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

Suspend
This suspends the creation of object versions for all operations but preserves any existing object versions.

Enable

After enabling Bucket Versioning, you might need to update your lifecycle rules to manage previous versions of objects.

Multi-factor authentication (MFA) delete

An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. [Learn more](#)

Disabled

Cancel **Save changes**

Amazon S3

General purpose buckets

- Directory buckets
- Table buckets
- Vector buckets
- Access Grants
- Access Points (General Purpose Buckets, FSx file systems)
- Access Points (Directory Buckets)
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

josh-bucket-1ms24mc040 Info

Properties

Bucket overview

AWS Region
Asia Pacific (Mumbai) ap-south-1

Amazon Resource Name (ARN)
 arn:aws:s3:::josh-bucket-1ms24mc040
0

Creation date
October 15, 2025, 15:04:04 (UTC+05:30)

Note: You can view versions by clicking “List versions” in the bucket objects page.

To Restore or Delete a Specific Version

- Click the object name → Versions
- Select the desired version → Download / Delete
(Deleting only adds a delete marker — older versions are still stored.)

Cross-Region Replication (CRR)

CRR automatically copies objects from one S3 bucket (source) to another (destination) in a different AWS Region.

Used for disaster recovery, compliance, or low-latency access in another region.
Requires Versioning to be enabled on both buckets.

Steps to Set Up CRR

NOTE: Enable Versioning on both:

Source bucket
Destination bucket

Step 1: Choose a different region before you create Destination bucket
Create Destination Bucket first.

Step 2: Create Source Bucket
Give replication permission:
Source bucket → Management tab → Replication rules → Create rule

Step 3: Create Replication Rule page

Enter a Replication rule name
Status: Enabled

Source bucket section:

Choose a rule scope: select “Apply to all objects in the bucket”

Destination:

Select “Choose a bucket in this account”

Bucket name: Select the destination bucket

IAM role:

Select “Create new role”

[An IAM role gives Amazon S3 permission to replicate objects from your source bucket to your destination bucket.

S3 replication won’t work unless you assign (or create) a role with the right permissions.

IAM Role: Either create a new role automatically or choose an existing one]

The screenshot shows the 'Create replication rule' configuration page. It includes sections for 'Replication rule configuration' (with fields for 'Replication rule name' and 'Status'), 'Source bucket' (with 'Source bucket name' set to 'am-call-bucket' and 'Source Region' set to 'Europe (Stockholm) eu-north-1'), and 'Choose a rule scope' (with 'Apply to all objects in the bucket' selected). The URL in the browser is [Amazon S3 > Buckets > am-call-bucket > Replication rules > Create replication rule](#).

The screenshot continues the 'Create replication rule' configuration. It shows the 'Destination' section (with 'Choose a bucket in this account' selected), 'Bucket name' ('lost-mitt-bucket'), 'Destination Region' ('Asia Pacific (Singapore) ap-southeast-1'), 'IAM role' ('Create new role'), 'Encryption' (unchecked), 'Destination storage class' (unchecked), and 'Additional replication options'. The URL in the browser is [Amazon S3 > Buckets > am-call-bucket > Replication rules > Create replication rule](#).

Step 4: Save

Any new objects uploaded to the source bucket will automatically replicate to the destination region.

Note: Replication is not retroactive — only new uploads after enabling CRR are copied.

Reminder – Resource cleanup – to release/delete/terminate the resources created.

Date: 29-10-2025

Exercise-5: Overview of EC2, To Launch a Windows EC2 Instance and Connect via RDP Client

Amazon Elastic Compute Cloud (EC2) is a core AWS Compute service that lets you run virtual servers (instances) in the cloud.

Amazon EC2 is an Infrastructure as a Service (IaaS) offering from AWS.

It allows you to launch virtual machines to host applications and manage them remotely – wherever you are in the world.

Key concepts:

Instance - A virtual machine running in the AWS cloud.

AMI (Amazon Machine Image) - A pre-configured template that includes: OS (Linux, Windows, etc.), Application software, other configurations.

Instance Type - Defines hardware power:

| Family | Example | Use Case |
|-------------------|-------------|----------------------------|
| General Purpose | t2.micro | Basic web apps |
| Compute Optimized | c5.large | High-performance computing |
| Memory Optimized | r5.large | Databases, analytics |
| Storage Optimized | i3.large | Data warehousing |
| GPU Instances | g4dn.xlarge | ML/AI, graphics |

EBS (Elastic Block Store) - Persistent storage for your EC2 instance. Acts like a hard drive — data remains even after instance stops. Types: SSD, HDD, etc.

Security Groups - Virtual firewalls controlling inbound and outbound traffic.

Example: Allow HTTP (port 80), SSH (port 22), HTTPS (port 443)

Key Pair - Used for secure login (SSH for Linux, RDP for Windows). Consists of a public key (stored in AWS) and private key (.pem) that you download.

Common Ways to Access EC2

- SSH (Linux instances)
- RDP (Windows instances) - Use Remote Desktop with Administrator password.
- User Data Script: Run automation commands during instance launch.

EC2 Use Cases

- Hosting static or dynamic websites
- Deploying web servers (Apache/Nginx)
- Running applications, APIs, or databases
- Machine Learning model hosting

- Batch processing jobs

Pricing Models

- On-Demand: Pay per hour/second; flexible.
- Reserved Instances: 1–3 year commitment; cheaper.
- Spot Instances: Unused capacity; up to 90% cheaper.
- Free Tier: t2.micro or t3.micro free for 6 months.

Lifecycle of an Instance

| Step | Description |
|-----------|---|
| Launch | Choose AMI, type, key, security group |
| Running | Accessible and operational |
| Stop | Instance paused, EBS persists |
| Start | Boot again from same EBS |
| Terminate | Deleted permanently, data lost unless backed up |

Two types of IPv4 addresses

When you launch an EC2 instance, AWS automatically assigns two types of IPv4 addresses depending on your network settings (VPC, subnet, etc.):

1. Private IPv4 Address

- Purpose: Used for internal communication within the same VPC (Virtual Private Cloud).
- Assigned Automatically: Yes, by AWS from the subnet's private IP range.
- Visibility: Not accessible from the Internet.
- Use Case:
 - Instance-to-instance communication inside AWS (e.g., web server ↔ database server).
 - Internal services that do not need public internet access.
- Persistence: The private IP remains attached to the instance until it is terminated.

2. Public IPv4 Address

- Purpose: Used for communication over the Internet.
- Assigned Automatically:
 - Yes, if your subnet is public (i.e., auto-assign public IP enabled).
 - No, if it's a private subnet.
- Visibility: Accessible from the Internet.
- Use Case:
 - Accessing the instance via SSH or HTTP from your local system.
 - Hosting web applications publicly.
- Persistence:
 - The public IP changes each time you stop/start the instance.
 - To make it permanent, you can assign an Elastic IP (static public IP).

Remote Desktop Protocol [RDP], is a secure communication protocol developed by Microsoft that allows a user to connect to and control another computer remotely. It establishes an encrypted channel to transmit keyboard and mouse inputs from the client to the remote computer and send screen information back to the client, providing a graphical user interface for remote access.

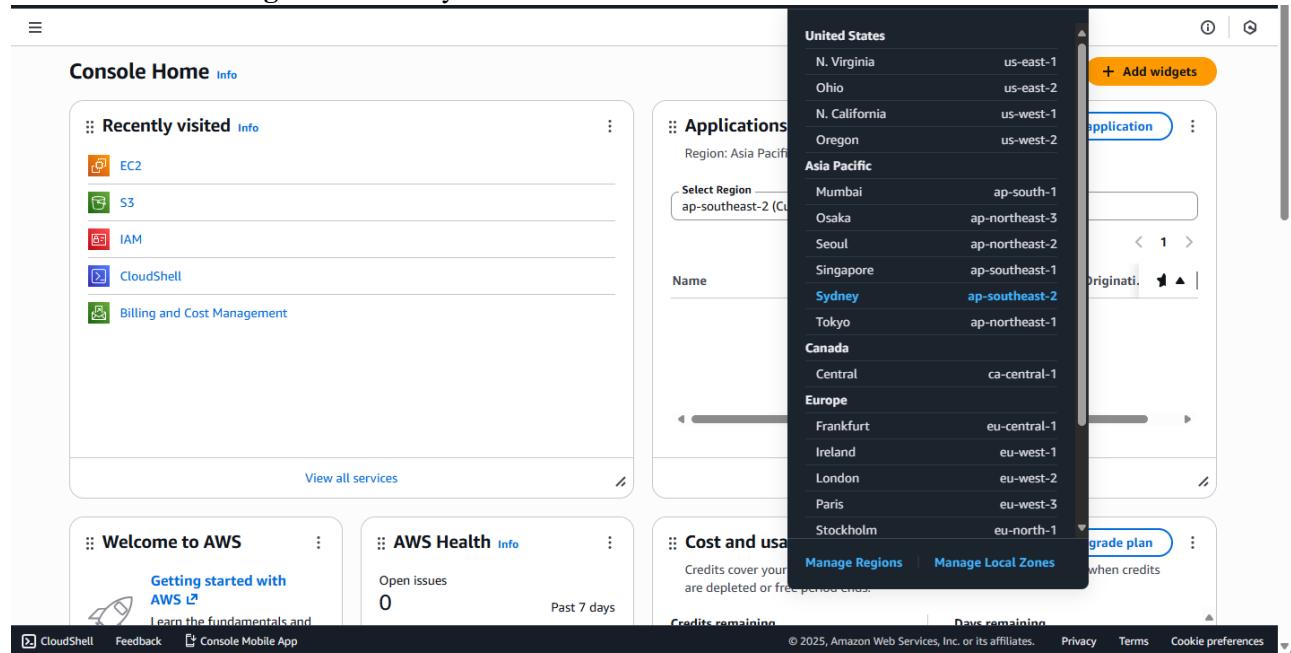
An **RDP client** is the software or app that you use to make this remote connection.

It connects to a remote Windows server or Windows EC2 instance that is running an RDP server (which listens on port **3389**). It is pre-installed in Windows.

Steps to Launch a Windows EC2 Instance and Connect via RDP Client

Step 1: Sign in to AWS Management Console

Select the **Region** closest to you.



The screenshot shows the AWS Management Console home page. On the left, there's a sidebar with links to EC2, S3, IAM, CloudShell, and Billing and Cost Management. Below that is a "Recently visited" section with links to EC2, S3, IAM, CloudShell, and Billing and Cost Management. In the center, there are three cards: "Welcome to AWS", "AWS Health", and "Cost and usage". The "Cost and usage" card has a "Manage Regions" button highlighted with a blue border. A dropdown menu for selecting a region is open, listing regions grouped by continent: United States (N. Virginia, Ohio, N. California, Oregon), Asia Pacific (Mumbai, Osaka, Seoul, Singapore, Sydney, Tokyo), Canada (Central), Europe (Frankfurt, Ireland, London, Paris, Stockholm), and South America (Sao Paulo). The "Select Region" dropdown shows "ap-southeast-2 (C...)".

Step 2: Open the EC2 Dashboard

In the AWS Console, search for **EC2** in the search bar.

Click on **EC2** → Instances → Launch Instance.

Under Name and Tags, give your instance a name.

The screenshot shows the AWS EC2 landing page. The left sidebar contains a navigation menu with sections like Dashboard, AWS Global View, Events, Instances, Images, and Elastic Block Store. The main content area features a large title "Amazon Elastic Compute Cloud (EC2)" and a subtitle "Create, manage, and monitor virtual servers in the cloud." Below this is a brief description of EC2's benefits and a call-to-action button "Launch instance". To the right, there are sections for "Additional actions" (View running instances, Migrate a server) and "Pricing (US)". The bottom of the page includes standard AWS footer links.

Step 3: Choose an Amazon Machine Image (AMI)

Under Application and OS Images (Amazon Machine Image) → click Browse more AMIs or choose: Microsoft Windows Server 2022 Base (Free tier eligible).

This screenshot shows the "Launch an instance" wizard, step 3: Choose an Amazon Machine Image (AMI). The left panel displays a "Name and tags" section where "Vm1" is entered. Below it is a "Application and OS Images (Amazon Machine Image)" section with a search bar and a grid of quick start options including macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian. The right panel is titled "Summary" and includes fields for "Number of instances" (set to 1), "Software Image (AMI)" (Microsoft Windows Server 2022), "Virtual server type (instance type)" (t3.micro), "Firewall (security group)" (New security group), and "Storage (volumes)" (1 volume(s) - 30 GB). It also features "Cancel", "Launch instance", and "Preview code" buttons.

Step 4: Choose Instance Type

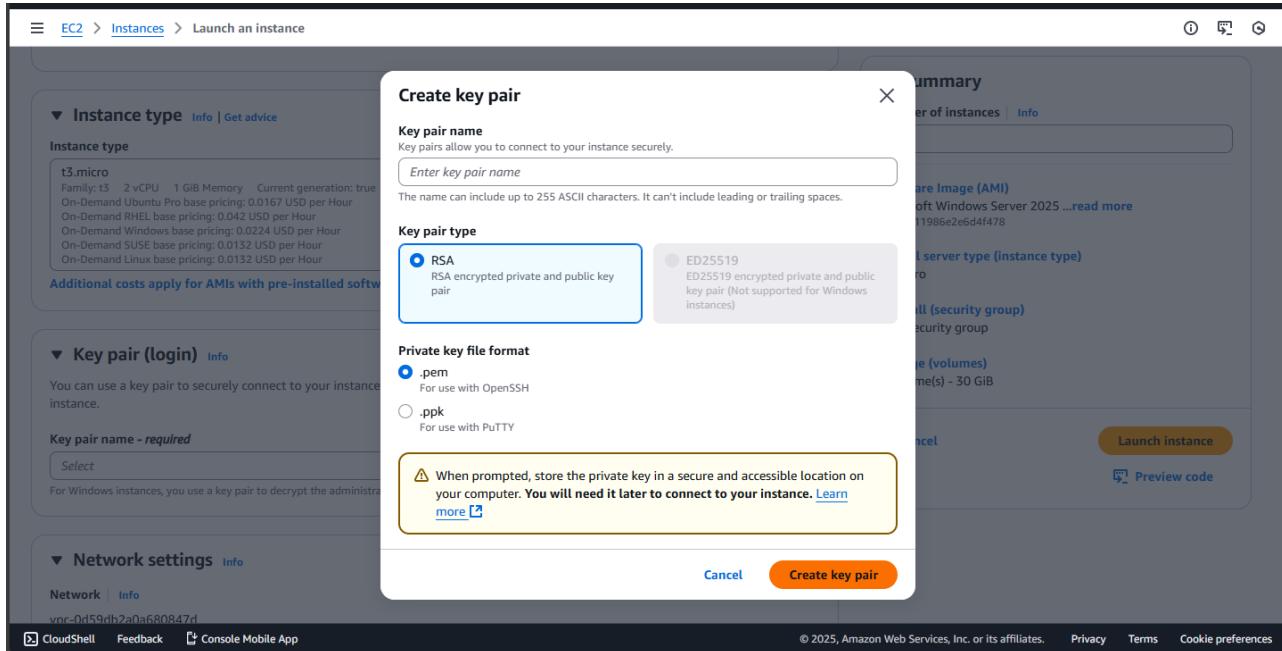
Choose t3.micro (Free-tier eligible).

Step 5: Configure Key Pair

Under Key pair (login), choose an existing key pair or create a new one.

If creating a new key pair:

- Choose type: RSA
- Format: .pem
- Download and save it safely — it's required to decrypt your Windows password later.

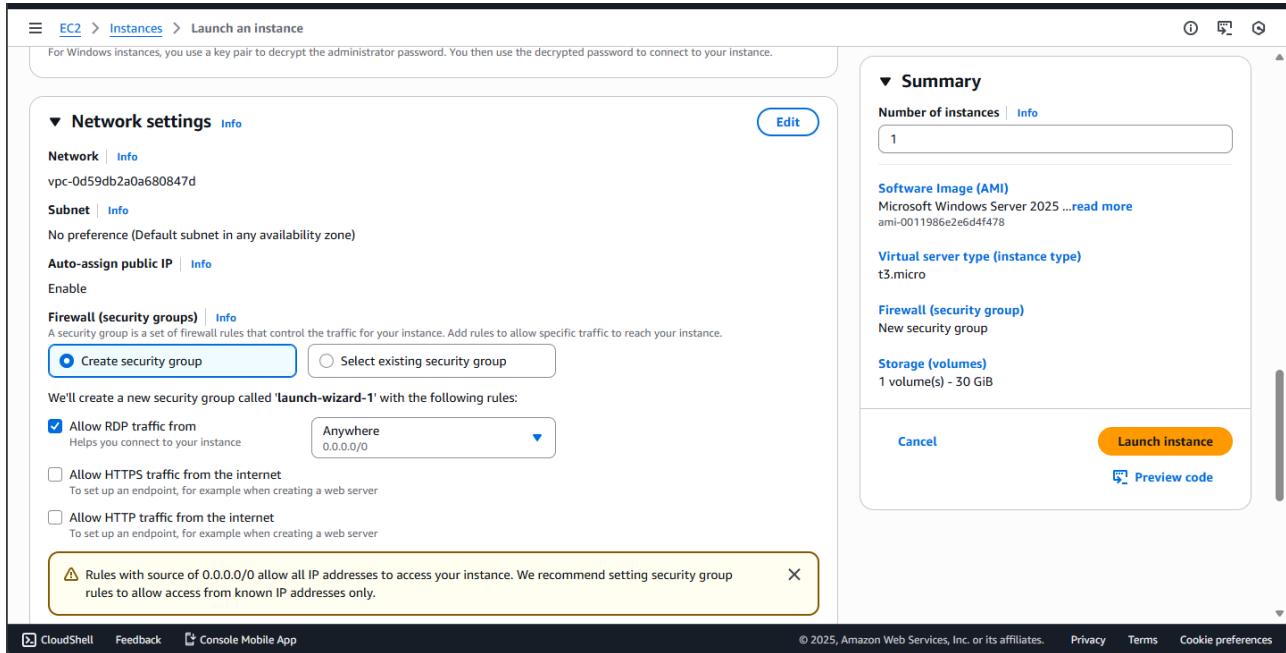


Step 6: Configure Network Settings

Leave default VPC and Subnet settings.

Under Firewall (security group) →

- Select Create security group.
- Allow RDP (port 3389) access from My IP (for better security) or anywhere (0.0.0.0/0).

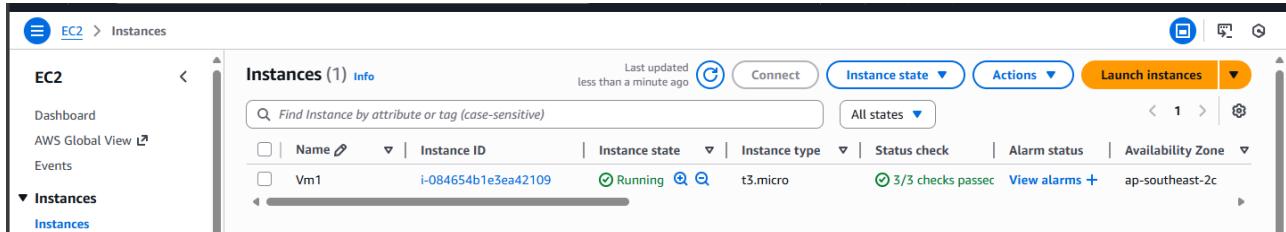


Step 7: Launch the Instance

Review all configurations.

Click Launch Instance.

Wait until the Instance state changes to Running and Status check = 3/3 passed.



Note:

Wait approximately 5 minutes after instance launch to allow AWS to fully initialize the instance and make the Administrator password available.

When a Windows EC2 instance is first launched, AWS needs a few minutes to:

Initialize the instance, Attach the root volume, Generate the Windows Administrator password (encrypted using your key pair).

Step 8: Get the Administrator Password

Select your running instance → click Connect → choose RDP Client tab.

Click Get Password (you must wait about 4 minutes after launch).

Upload your .pem key file and click Decrypt Password.

Copy the Public IPv4 address and Administrator Password shown.

EC2 > Instances > i-084654b1e3ea42109 > Connect to instance

Connect Info

Connect to an instance using the browser-based client.

Session Manager RDP client EC2 serial console

Record RDP connections You can now record RDP connections using AWS Systems Manager just-in-time node access. Learn more Try for free

Instance ID i-084654b1e3ea42109 (Vm1)

Connection Type

Connect using RDP client Download a file to use with your RDP client and retrieve your password.

Connect using Fleet Manager To connect to the instance using Fleet Manager Remote Desktop, the SSM Agent must be installed and running on the instance. For more information, see Working with SSM Agent

You can connect to your Windows instance using a remote desktop client of your choice, and by downloading and running the RDP shortcut file below:

[Download remote desktop file](#)

When prompted, connect to your instance using the following username and password:

Public DNS ec2-13-211-255-78.ap-southeast-2.compute.amazonaws.com

Username Info Administrator

Password Get password

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 > Instances > i-084654b1e3ea42109 > Get Windows password

Get Windows password Info

Use your private key to retrieve and decrypt the initial Windows administrator password for this instance.

Instance ID i-084654b1e3ea42109 (Vm1)

Key pair associated with this instance vm

Private key Either upload your private key file or copy and paste its contents into the field below.

[Upload private key file](#)

vm.pem 1.678KB

Private key contents - optional

```
-----BEGIN RSA PRIVATE KEY-----MIIEpIBAAKCAQEA1aObIm85UDxw4qtTV15ejnR39ccSUEy4oLw0ywSsNxyvYPnpBj/DAGUx8zpgVzSm/Yg18s2TqBhUisegLit+htAaneoC024mg+jUgx/VJSq12tQNRPc1Y+ox3akd+I2cx6of/EsdMhd9wBunU9dgx2Q30coeMH/o8AO60hJ+zh7C4rXODe7xRzLmmnh8QXfoxWerAUyn3c1WN+A/gZrrKbq4UF4JNkfs6lMEoL2DEOzpIOPpcytunpJNSFHCPKTb10TWKvc9e/R+TWtYaslyGw1B7scOBdc9db3Jbh6qrYYAEwdn/uzgokf1L4GzHQBNI LDnD0j/AmwUOwiDAQABoIaFV1Jnxl8eNEo/DCYojFKcCd3QqC08gGyVpsTw9Xce1jyHckffasBl/9qiz1S4iM/bbs8+dKvzz0ONw
```

Cancel Decrypt password

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

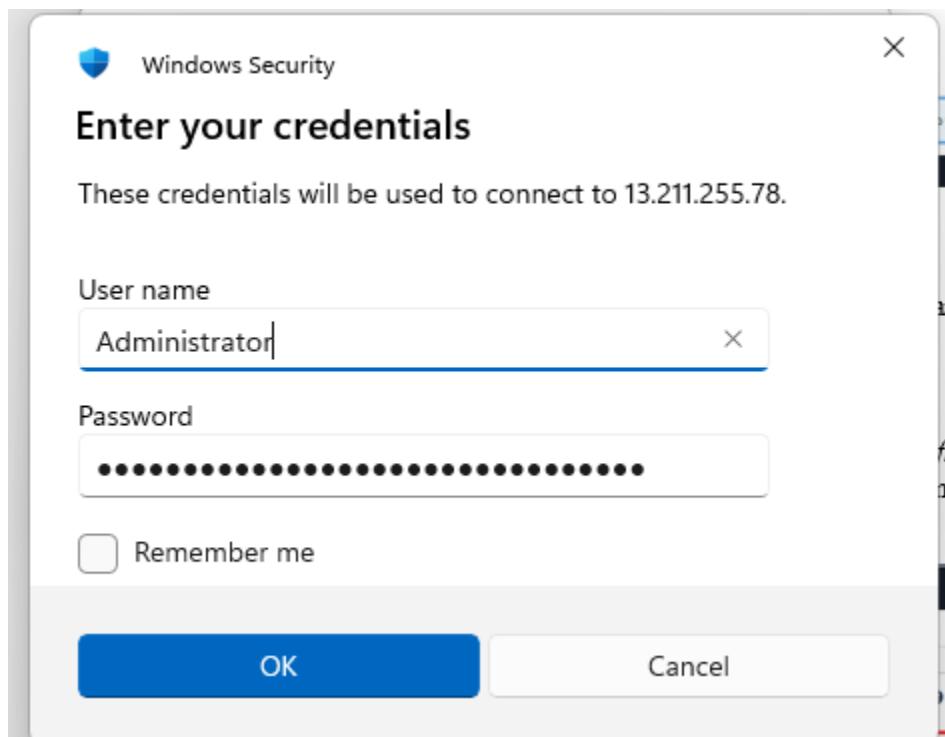
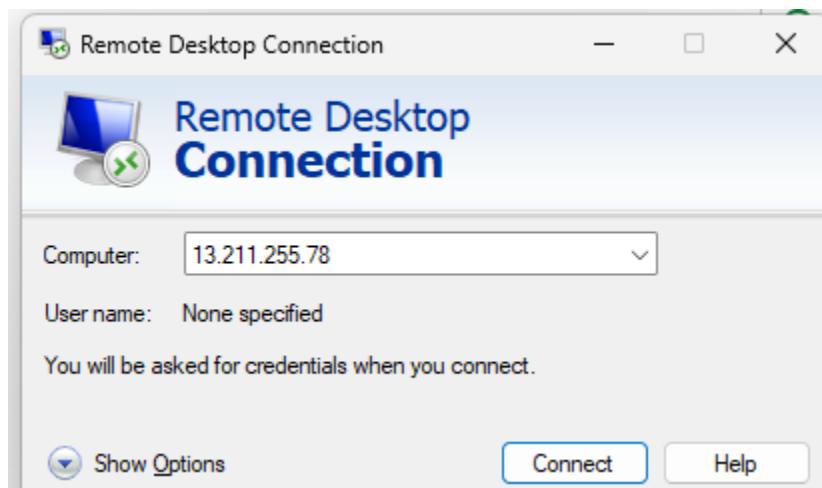
The screenshot shows the AWS EC2 Instances page for an instance named i-084654b1e3ea42109. The 'RDP client' tab is selected. It displays the instance ID (i-084654b1e3ea42109) and connection type options. The 'Connect using RDP client' option is selected, with a note that it requires an RDP client and password. An alternative 'Connect using Fleet Manager' option is also shown. A download link for the remote desktop file is provided, along with the public DNS and password fields. A note indicates that directory credentials can be used if joined to a domain.

Step 9: Connect Using RDP Client

On Windows system:

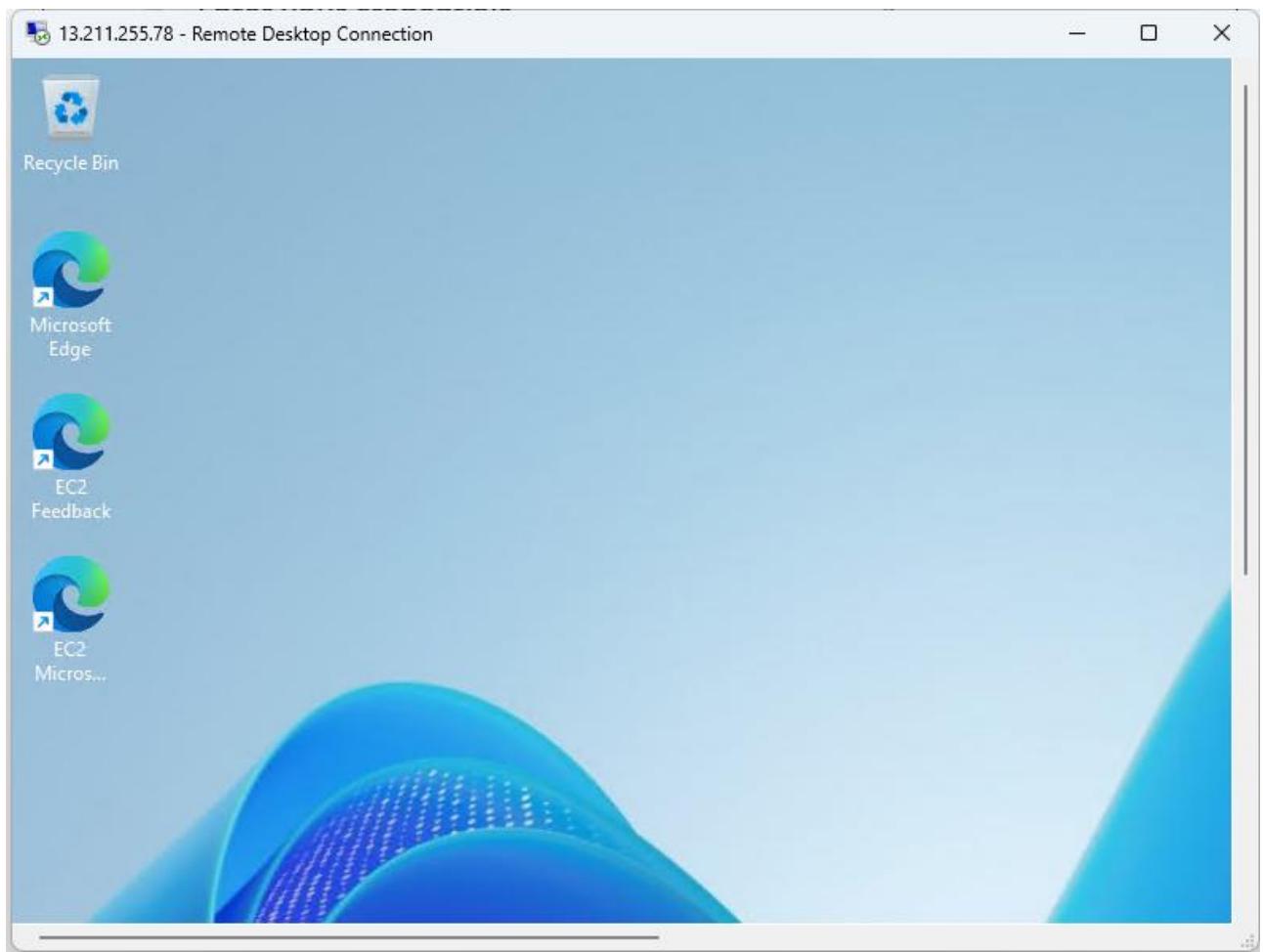
1. Open Remote Desktop Connection (from Start Menu).
2. Enter your instance's Public IPv4 address.
3. Click Connect → Enter:
 - o Username: Administrator
 - o Password: (*the decrypted password from AWS*)
4. Click OK → accept the certificate → the remote Windows desktop opens!

The screenshot shows the AWS EC2 Instances page for the same instance. The 'Instances' section is expanded, showing the instance summary. The 'Public IPv4 address' field (13.211.255.78) is highlighted with a red box. Other details shown include the instance ID (i-084654b1e3ea42109), Instance state (Running), and Public DNS (ec2-13-211-255-78.ap-southeast-2.compute.amazonaws.com).



Step 10: Verify Connection

- You should now see a Windows Server desktop running inside your local window.
- You can open File Explorer, browse settings, or install software (within the free-tier limits).



Note:

- Always **Stop** (not Terminate) the instance when not in use to avoid charges.
- RDP uses port 3389, so ensure it's open in the security group.
- Avoid sharing your decrypted password or key pair.

Date: 30-10-2025

Exercise-6: Launch a Linux EC2 Instance and Connect using SSH through PowerShell/Linux Terminal and PuTTY on Windows.

Note: Choosing the Correct Key Pair Format

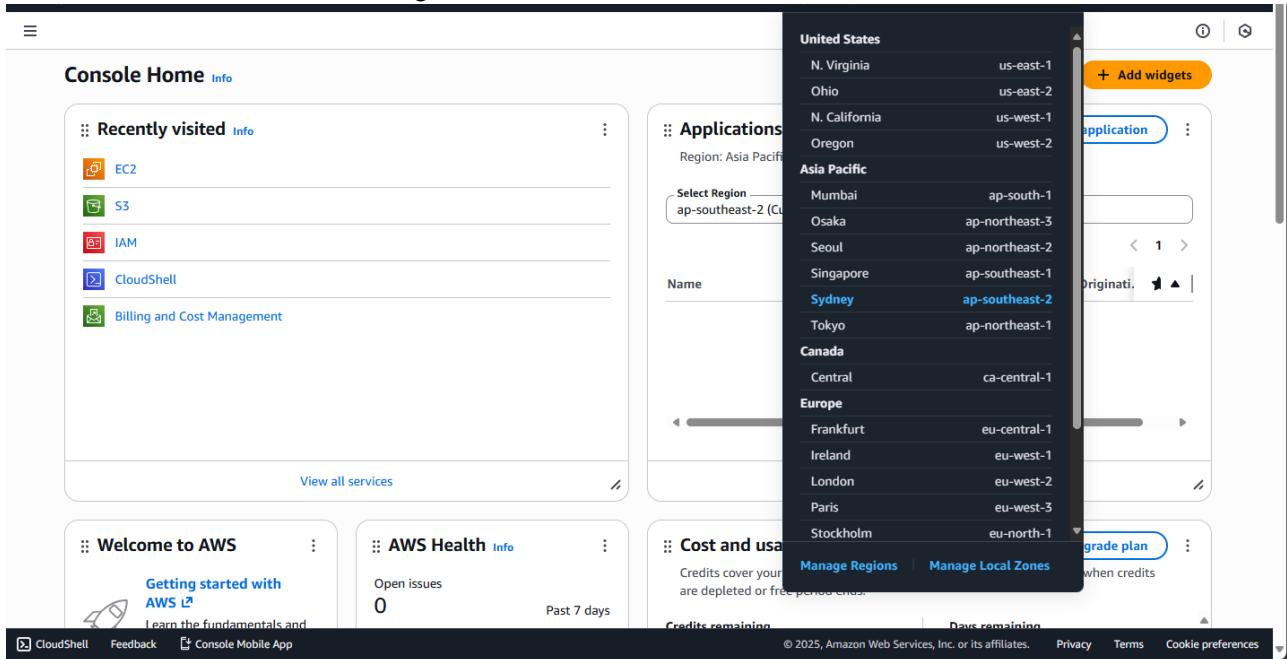
While creating a Key Pair, you are asked to select the Private Key File Format — either .pem or .ppk. The correct choice depends on the operating system and the method you will use to connect to your EC2 instance.

| Scenario | Key File Format | Explanation |
|--|-----------------|---|
| Using PuTTY on Windows | .ppk | The .ppk file is specific to the GUI based PuTTY application, a popular SSH client for Windows system. |
| Using PowerShell on Windows, Linux terminal on Linux | .pem | The .pem file is the default AWS key format used by the OpenSSH client available in PowerShell (Windows), Linux, and macOS terminals. Used for CLI. |

Steps to Launch a Linux EC2 Instance and Connect using SSH through PowerShell/Linux

Step 1: Sign in to AWS Management Console

select the nearest AWS Region.



Step 2: Open EC2 Service

In the search bar, type EC2 and click EC2 Dashboard.

Select Instances → Launch Instance.

EC2

Compute

Amazon Elastic Compute Cloud (EC2)

Create, manage, and monitor virtual servers in the cloud.

Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest compute platform, with over 600 instance types and a choice of the latest processors, storage, networking, operating systems, and purchase models to help you best match the needs of your workload.

Benefits and features

EC2 offers ultimate scalability and control

Fully resizable compute capacity to support virtually any workload. This service is best if you want:

- Highest level of control of the entire technology stack, allowing full integration with all AWS services
- Wide variety of server size options
- Wide availability of operating systems to choose from including Linux, Windows, and

Additional actions

- View running instances
- Migrate a server

Pricing (US)

Step 3: Configure Instance Details

Under Name and Tags, give your instance a name, e.g., *Linux-SSH-Demo*.

Under Application and OS Images (AMI) → select Amazon Linux 2 AMI (Free tier eligible).

Under Instance type, choose t3.micro (Free-tier eligible).

EC2 > Instances > Launch an instance

Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: vm3 | Add additional tags

Application and OS Images (Amazon Machine Image)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recent AMIs: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, Debian

Browse more AMIs

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.9.2...read more

Virtual server type (instance type): t3.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Cancel | Launch instance | Preview code

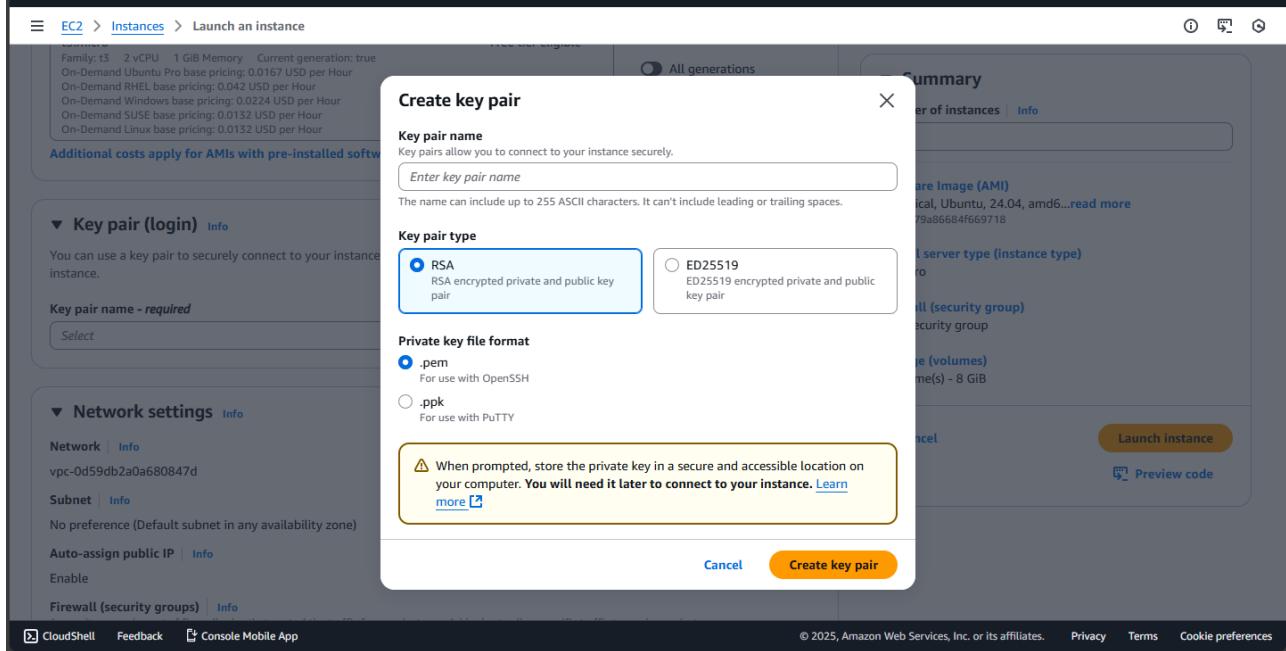
Step 4: Create or Select a Key Pair

Under Key pair (login) → choose Create new key pair.

Choose:

- Key pair type: RSA
- Private key file format: .pem (for SSH via PowerShell/Linux)

Click Create key pair → the .pem file will automatically download.
Save it securely on your local machine (you'll need it for SSH login).



Step 5: Configure Network Settings

Under Network settings, leave the defaults.

In Firewall (security group) → select Create security group.

Ensure SSH (port 22) is allowed:

- Type: SSH
- Protocol: TCP
- Port Range: 22
- Source: My IP (recommended for security) or Anywhere (0.0.0.0/0).

Step 6: Launch the Instance

Review the configuration → click Launch Instance.

Wait for the Instance State to show Running and Status Checks: 3/3 passed.

Step 7: Get the Public IP Address

Select your instance → In the summary section →

Copy the Public IPv4 address — you'll use this to connect.

Step 8: Connect using SSH

(A) On Windows using PowerShell

- Open **PowerShell** (search “PowerShell” from the Start menu).
- Navigate to the folder where your .pem file is saved:
cd "C:\Users\<YourName>\Downloads"

- Connect using the SSH command:
ssh -i "keyfile.pem" ec2-user@<Public-IP-address>
- When prompted, type **yes** to continue connecting.
- You'll now be logged into your EC2 Linux terminal.

```
PS C:\Users\MCA-B1\Downloads> ssh -i "vm3.pem" ec2-user@13.54.149.84
  _#
 /_###_
~~ \####\ Amazon Linux 2023
~~ \###|
~~ \#/ ___ https://aws.amazon.com/linux/amazon-linux-2023
~~ \~'`-->
~~~ /
~~~ .-`-/
~~~ /-`-
~/`-/
/m/`-/
[ec2-user@ip-172-31-4-166 ~]$ ls -a
. .. .bash_logout .bash_profile .bashrc .ssh
[ec2-user@ip-172-31-4-166 ~]$ cd ..
[ec2-user@ip-172-31-4-166 home]$ cd ..
[ec2-user@ip-172-31-4-166 ~]$ ls
bin boot dev etc home lib lib64 local media mnt opt proc root run sbin srv sys tmp usr var
[ec2-user@ip-172-31-4-166 ~]$ cd var/
[ec2-user@ip-172-31-4-166 var]$ ^C
[ec2-user@ip-172-31-4-166 var]$ ls
account adm cache db empty ftp games kerberos lib local lock log mail nis opt preserve run spool tmp yp
```

(B) On Linux Terminal (Ubuntu / macOS)

- Open **Terminal**.
- Navigate to the directory where your .pem file is stored.
- Set proper permission for the key file:
chmod 400 keyfile.pem
- Connect to the instance:
ssh -i keyfile.pem ec2-user@<Public-IP-address>
- Type **yes** when prompted.
- You will be connected to your EC2 instance remotely.

Step 9: Verify Connection

- Once connected, the command prompt will appear as:
[ec2-user@ip-172-31-xx-xx ~]\$
- Try a few commands:
uname -a and sudo yum update -y to confirm access.

Step 10: Stop Instance after Use

Go to the EC2 console.

Select your instance → Instance State → Stop Instance (to avoid charges).

Steps to Install PuTTY on Windows

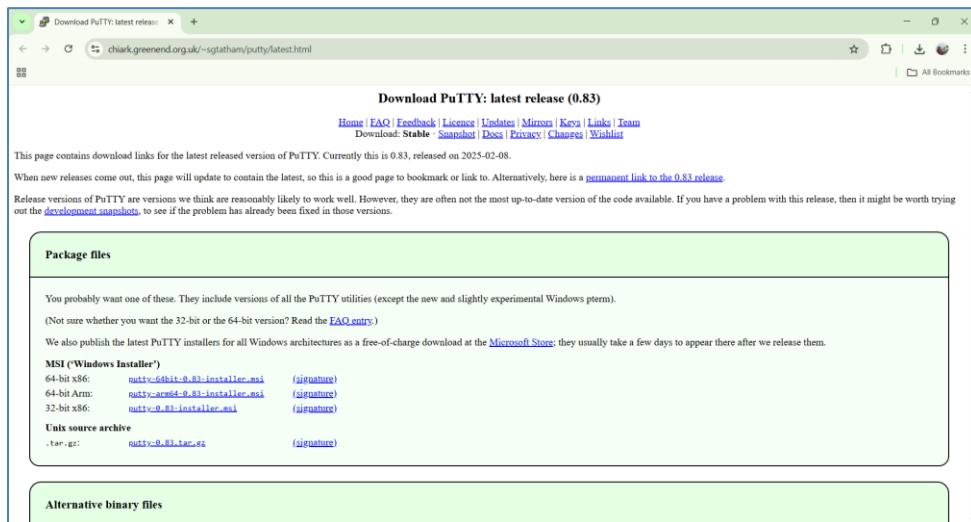
PuTTY is a client program for the SSH, Telnet and Rlogin network protocols. These protocols allow you to interact with a remote server as if you were sitting right in front of it.

It is primarily used in the Windows platform.

In an era where cloud computing and remote servers are the norm, being able to securely connect and interact with these servers is vital. It provides a secure and reliable way to connect to these remote systems. It supports a range of network protocols including the secure ones like SSH.

Use the correct, safe download link: Official download page: This is the only genuine PuTTY source.

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>



Under the **Package files** section, look for:

MSI ('Windows Installer')

64-bit x86: putty-64bit-0.83-installer.msi

Click the link: **putty-64bit-0.83-installer.msi**



Once it downloads, open the file and follow: Next → Next → Install → Finish

After installation, you will have:

- PuTTY – to connect to your EC2 instance via SSH
- PuTTYgen – to convert .pem to .ppk (if needed)
- Pageant – optional key manager

You can open PuTTY by typing **PuTTY** in the Windows search bar/start menu.

Steps to Launch a Linux EC2 Instance and Connect using PuTTY on Windows

Step 1: Sign in to AWS Management Console

Select your nearest AWS Region.

Step 2: Open the EC2 Service

In the AWS Console search bar, type EC2 and select EC2 Dashboard.

Click Instances → Launch Instance.

Step 3: Configure Instance Details

Under Name and Tags, enter an instance name, e.g., *Linux-PuTTY-Demo*.

Under Application and OS Images (AMI) → choose Amazon Linux 2 AMI (Free tier eligible).

Under Instance Type, select t2.micro (Free tier eligible).

The screenshot shows the 'Launch an instance' wizard in the AWS Management Console. The top navigation bar includes 'EC2 > Instances > Launch an instance'. A blue header bar contains 'Take a walkthrough' and 'Do not show me this message again.' buttons. The main area is titled 'Launch an instance' with an 'Info' link. It explains that Amazon EC2 allows creating virtual machines. The 'Name and tags' section has a 'Name' field containing 'Linux-SSH-Demo' and a 'Add additional tags' button. The 'Application and OS Images (Amazon Machine Image)' section has a 'Search our full catalog including 1000s of application and OS images' input field. Below it, under 'Recent', are icons for Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian. Under 'Quick Start', there are icons for CloudShell, Feedback, and Console Mobile App. On the right, a 'Summary' panel shows 'Number of instances' set to 1, 'Software Image (AMI)' as Amazon Linux 2023 AMI 2023.9..., 'Virtual server type (instance type)' as t3.micro, 'Firewall (security group)' as New security group, and 'Storage (volumes)' as 1 volume(s) - 8 GiB. At the bottom are 'Cancel', 'Launch instance', and 'Preview code' buttons.

Step 4: Create or Select a Key Pair

Under Key pair (login) → select Create new key pair.

Choose the following:

- Key pair type: RSA
- Private key file format: .ppk (*for PuTTY on Windows*)

Click Create key pair → a .ppk file will download automatically.

Save it securely — this file is required to connect later.

Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA
RSA encrypted private and public key pair

ED25519
ED25519 encrypted private and public key pair

Private key file format

.pem
For use with OpenSSH

.ppk
For use with PuTTY

⚠️ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)

Cancel **Create key pair**

Step 5: Configure Network Settings

Under Network settings, leave default VPC/Subnet settings.

Under Firewall (security group):

- Select Create security group.
- Ensure SSH (port 22) is allowed:
 - Type: SSH
 - Protocol: TCP
 - Port Range: 22
 - Source: anywhere.

Step 6: Launch the Instance

Review all configurations.

Click Launch Instance.

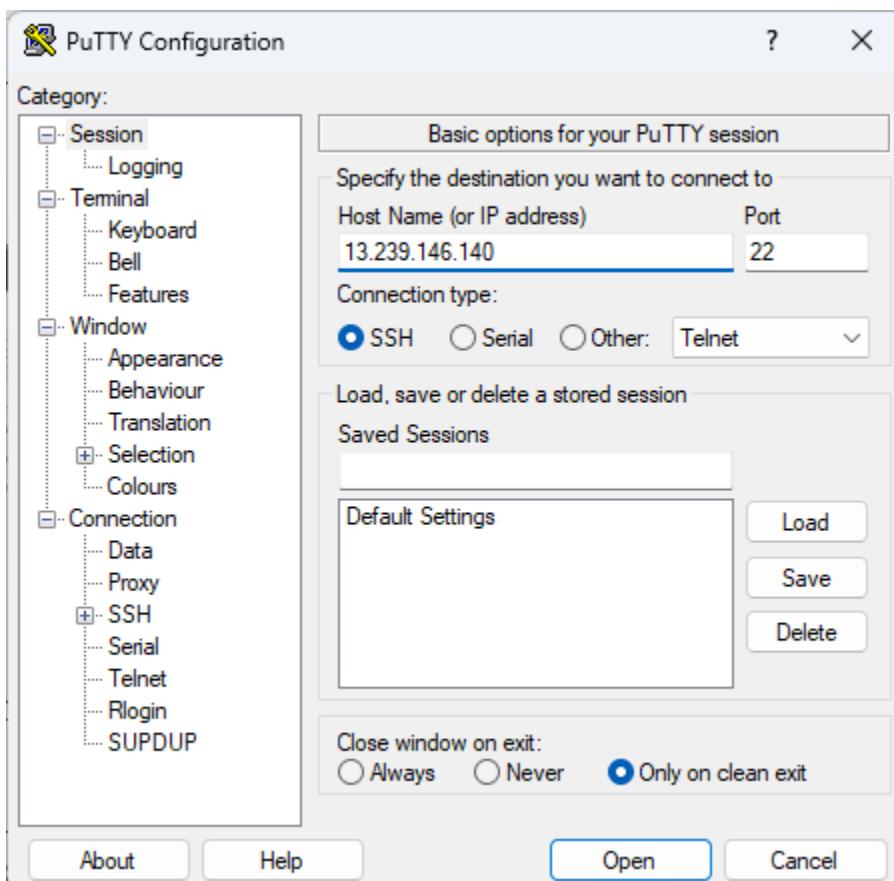
Wait until the Instance State changes to Running and Status Check = 3/3 passed.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with 'EC2' selected under 'Instances'. The main area displays a table of instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Two instances are listed: 'Vm1' (terminated) and 'Linux-SSH-De...' (running). The 'Actions' and 'Launch instances' buttons are visible at the top right.

Step 7: Obtain the Public IP

Select your instance → In the summary section →

Copy the Public IPv4 address or Public DNS (IPv4) — you'll use this to connect from PuTTY.



Step 8: Connect using PuTTY

Open PuTTY on your Windows system.

In the **Host Name (or IP address)** field, enter: `ec2-user@<Public-IP-address>`

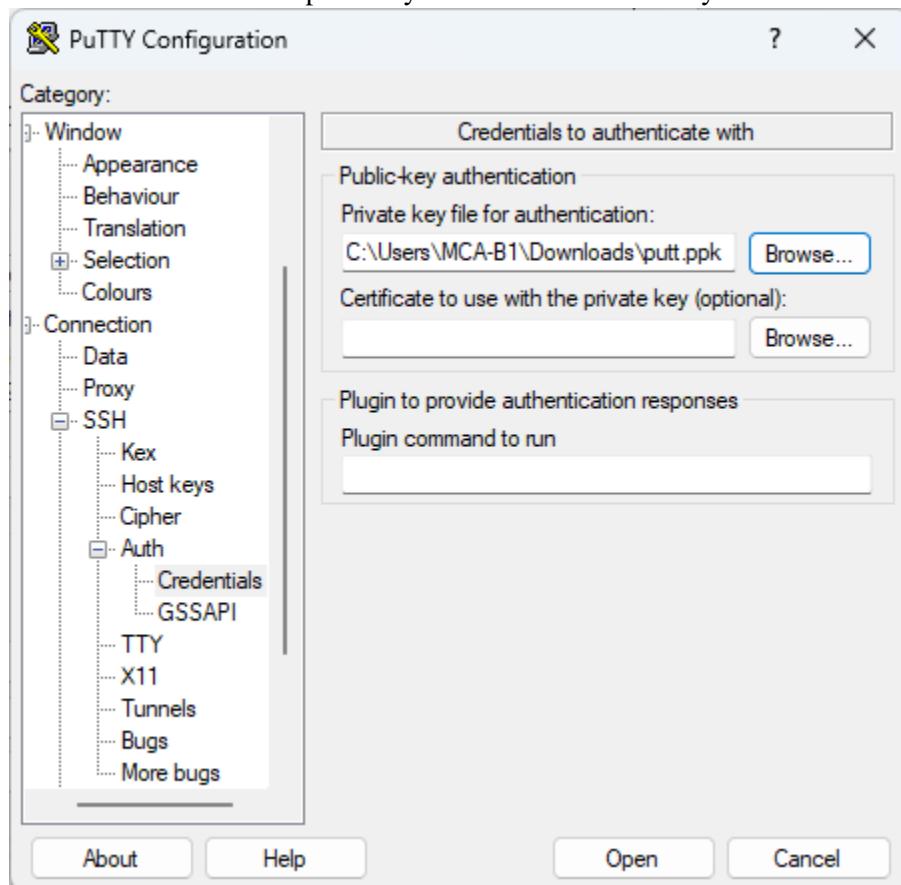
In the **Category** list on the left, expand: Connection → SSH → Auth → Credentials

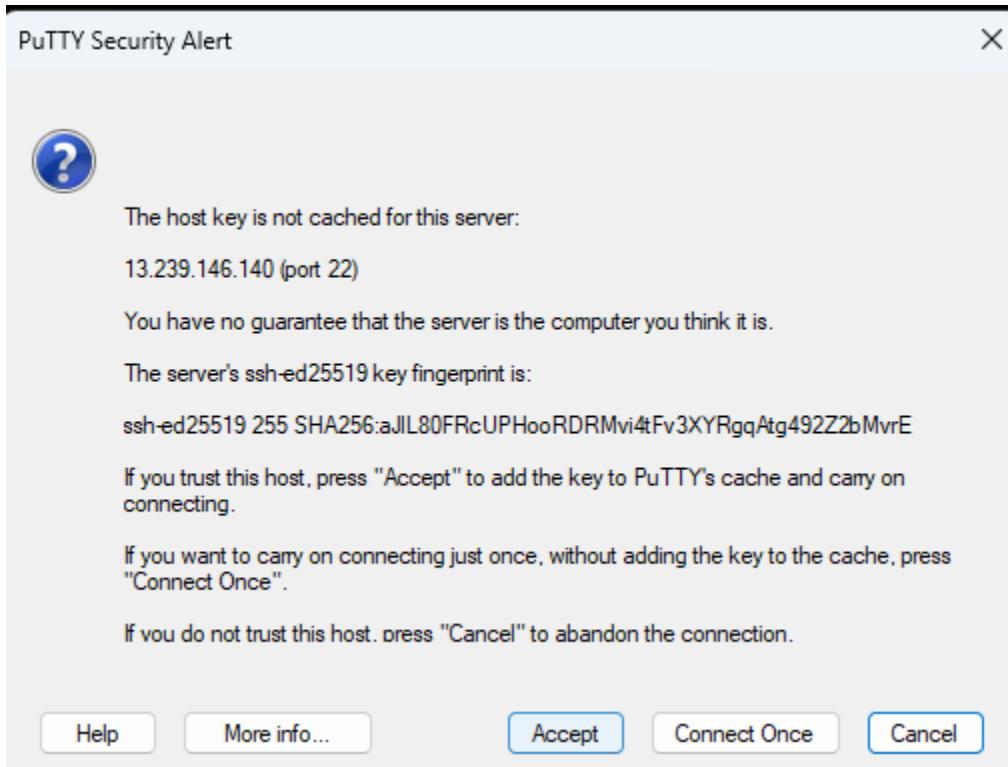
Click **Browse** → locate and select your .ppk key file downloaded earlier.

Click **Open**.

When prompted, click **Accept** to trust the host.

A terminal window opens — you're now connected to your EC2 Linux instance!





```
ec2-user@login:~$ login as: ec2-user
ec2-user@login:~$ Authenticating with public key "putt"
      _#
     ~\ _###_          Amazon Linux 2023
     ~~ \####\ \
     ~~   \|##|
     ~~     \|/
     ~~       V~' '-->
     ~~~           /
     ~~~ . .
     /m/ \
[ec2-user@login:~]$
```

Step 9: Verify Connection

Once connected, your prompt should look like: [ec2-user@ip-172-31-xx-xx ~]\$
Try verifying: uname -a or update packages: sudo yum update -y

Step 10: Stop the Instance

Return to the EC2 Dashboard.

Select your instance → click Instance State → Stop Instance.

This prevents charges when you're not using it.

Note

- Use .ppk format key when connecting with **PuTTY (Windows)**.
- Use .pem format key when connecting with **PowerShell / Linux / macOS terminal**.
- Both keys serve the same purpose — secure authentication to your EC2 instance.

Date: 5-11-2025

Exercise-7: Hosting a static website on EC2 instance.

- Manual Installation of Apache (httpd) Web Server on EC2 for Static Website Hosting.
- Launching an EC2 Instance with User Data Script to Automatically Install Apache and Host a Static Website.

When you create an EC2 instance, it's just a **bare machine** — It does not have the software to host a website yet. To host a website:

- You need a web server software → Apache (httpd)
- You put your website files (HTML, CSS, etc.) in a special folder (usually /var/www/html)
- Apache listens on port 80 (HTTP) or port 443 (HTTPS) for incoming connections

Apache HTTP Server (often called Apache or httpd) is a web server software.

It runs on a computer (like your EC2 instance) and delivers web pages (HTML, images, CSS, etc.) to users over the HTTP/HTTPS protocol.

So, whenever someone types your website's URL (like <http://your-ec2-ip/>), Apache receives that request and serves your webpage files from your server to the browser.

httpd stands for **HTTP Daemon.**

A *daemon* in Linux is a background service that keeps running to handle requests.

So, httpd is the background process that runs the Apache web server.

When you install Apache, you're really installing the **httpd service**.

File Locations (default) – Website files go into: /var/www/html

Steps for Manual Installation of Apache (httpd) Web Server on EC2 for Static Website Hosting.

Part 1 – Creating an EC2 Instance

Step 1: Sign in to AWS Management Console

In the search bar, type EC2 and open the EC2 Dashboard.

Step 2: Launch a New Instance

Click "Launch Instance."

Give a name. (e.g., MyApacheServer)

Step 3: Choose an Amazon Machine Image (AMI)

Select Amazon Linux 2 AMI (Free tier eligible)

The screenshot shows the 'Launch an instance' wizard. At the top, the navigation bar shows 'EC2 > Instances > Launch an instance'. The main section is titled 'Launch an instance' with a 'Name and tags' sub-section. A 'Name' field contains 'MyApacheServer' and a 'Add additional tags' button is visible. Below this, a 'Application and OS Images (Amazon Machine Image)' section is expanded, showing a search bar and a 'Quick Start' tab selected. Under 'Quick Start', there are cards for 'Amazon Linux' (selected), 'macOS', 'Ubuntu', 'Windows', 'Red Hat', and 'SUSE Linux'. A 'Browse more AMIs' link is also present.

Step 4: Choose Instance Type

Choose t3.micro (Free tier eligible).

Step 5: Create / Select Key Pair

Under Key pair (login), choose:

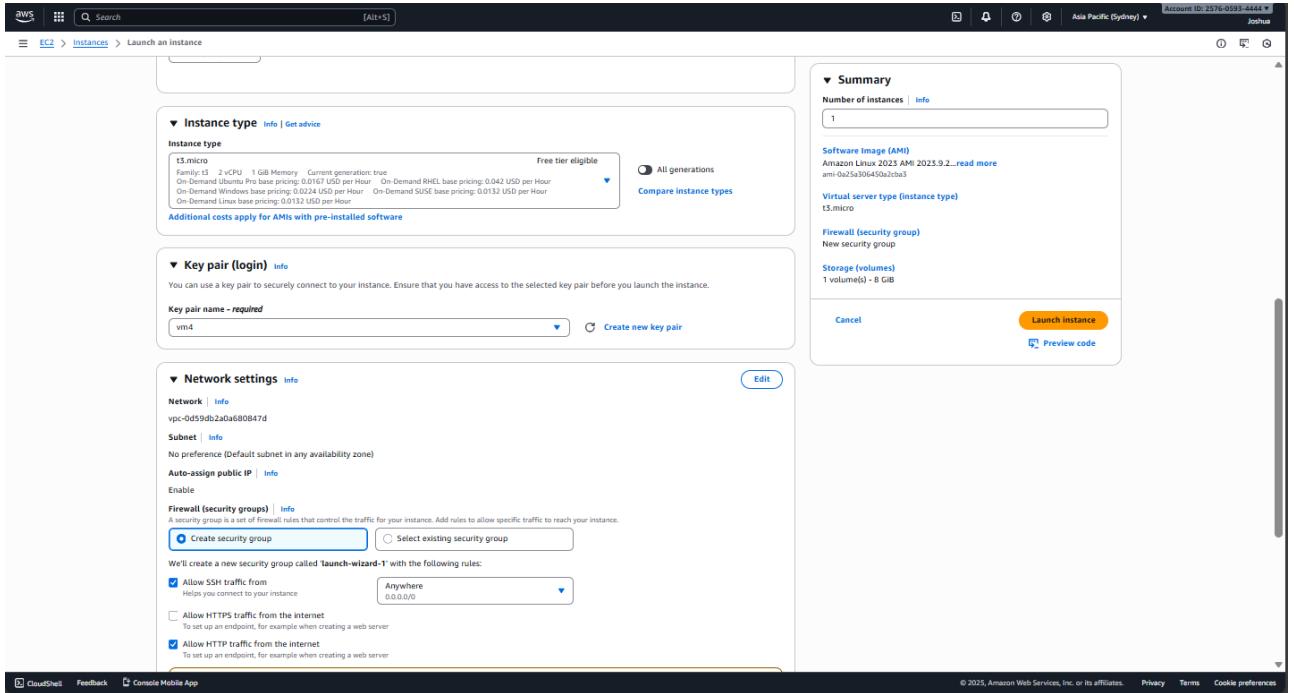
Create new key pair → Give a name → File format = .pem

Download the key file → Keep it safe.

Step 6: Configure Network Settings

Click → Allow SSH traffic

→ Allow HTTP traffic from the internet. (This automatically opens port 80 for web access).



Step 7: Launch Instance

Review → Click **Launch Instance**.

Wait for the instance to start.

Step 8: Connect to the Instance

Using powershell type the following commands:

- Open PowerShell (search “PowerShell” from the Start menu).
- Navigate to the folder where your .pem file is saved:
cd "C:\Users\<YourName>\Downloads"
- Connect using the SSH command:
ssh -i "keyfile.pem" ec2-user@<Public-IP-address>
- When prompted, type yes to continue connecting.

- You'll now be logged into your EC2 Linux terminal.

```
PS C:\Users\MCA-B1\Downloads> ssh -i "vm4.pem" ec2-user@ec2-13-239-169-152.compute.amazonaws.com
  _#_
 /_###_      Amazon Linux 2023
 /_###\_
 \##|
 #/ __ https://aws.amazon.com/linux/amazon-linux-2023
 \~' '-->
 /_/
 /_/
 /m/
Last login: Wed Nov  5 06:17:13 2025 from 13.239.158.5
[ec2-user@ip-172-31-25-83 ~]$ |
```

Part 2 – Manual Installation of Apache (httpd) Web Server

Step 1: Update the Packages

sudo yum update -y

```
[ec2-user@ip-172-31-25-83 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-25-83 ~]$ |
```

Step 2: Install Apache (httpd)

sudo yum install httpd -y

(This installs the Apache Web Server package.)

```
[ec2-user@ip-172-31-25-83 ~]$ sudo yum install httpd -y
Last metadata expiration check: 0:00:35 ago on Wed Nov  5 06:24:44 2025.
Dependencies resolved.
=====
Package           Architecture   Version        Repository      Size
=====
Installing:
httpd            x86_64        2.4.65-1.amzn2023.0.2  amazonlinux    47 k
=====
Installing dependencies:
apr              x86_64        1.7.5-1.amzn2023.0.4  amazonlinux    129 k
apr-util          x86_64        1.6.3-1.amzn2023.0.2  amazonlinux    97 k
apr-util-lmdb     x86_64        1.6.3-1.amzn2023.0.2  amazonlinux    13 k
generic-logos-httpd  noarch      18.0.0-12.amzn2023.0.3  amazonlinux    19 k
httpd-core        x86_64        2.4.65-1.amzn2023.0.2  amazonlinux    1.4 M
httpd-filesystem  noarch      2.4.65-1.amzn2023.0.2  amazonlinux    13 k
httpd-tools       x86_64        2.4.65-1.amzn2023.0.2  amazonlinux    81 k
libbrotli         x86_64        1.0.9-4.amzn2023.0.2  amazonlinux    315 k
mailcap           noarch      2.1.49-3.amzn2023.0.3  amazonlinux    33 k
=====
Installing weak dependencies:
apr-util-openssl  x86_64        1.6.3-1.amzn2023.0.2  amazonlinux    15 k
mod_http2         x86_64        2.0.27-1.amzn2023.0.3  amazonlinux    166 k
mod_lua           x86_64        2.4.65-1.amzn2023.0.2  amazonlinux    60 k
=====
Transaction Summary
=====
Install 13 Packages

Total download size: 2.4 M
Installed size: 6.9 M
Downloading Packages:
(1/13): apr-util-lmdb-1.6.3-1.amzn2023.0.2.x86_64.rpm          345 kB/s | 13 kB  00:00
(2/13): apr-1.7.5-1.amzn2023.0.4.x86_64.rpm                  2.7 MB/s | 129 kB  00:00
(3/13): apr-util-1.6.3-1.amzn2023.0.2.x86_64.rpm          1.9 MB/s | 97 kB  00:00
(4/13): apr-util-openssl-1.6.3-1.amzn2023.0.2.x86_64.rpm    706 kB/s | 15 kB  00:00
(5/13): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch.rpm 878 kB/s | 19 kB  00:00
(6/13): httpd-2.4.65-1.amzn2023.0.2.x86_64.rpm          1.9 MB/s | 47 kB  00:00
(7/13): httpd-filesystem-2.4.65-1.amzn2023.0.2.noarch.rpm  562 kB/s | 13 kB  00:00
(8/13): httpd-tools-2.4.65-1.amzn2023.0.2.x86_64.rpm       3.2 MB/s | 81 kB  00:00
(9/13): httpd-core-2.4.65-1.amzn2023.0.2.x86_64.rpm        23 kB/s | 1.4 MB  00:00
(10/13): mailcap-2.1.49-3.amzn2023.0.3.noarch.rpm        1.3 MB/s | 33 kB  00:00
(11/13): libbrotli-1.0.9-4.amzn2023.0.2.x86_64.rpm        8.4 MB/s | 315 kB 00:00
```

Step 3: Start the Apache Service

```
sudo systemctl start httpd
```

Step 4: Enable Apache to Start on Boot

```
sudo systemctl enable httpd
```

Step 5: Check Apache Status

```
sudo systemctl status httpd [It should show active (running). Press q to exit status view.]
```

Step 7: Test Apache Server

Copy your Public IPv4 address from the EC2 dashboard.

Paste it into a browser: http://

You should see the Apache Test Page



Part 3 – Host a Static Website on Apache

Step 1: Move to Web Root Folder

```
cd /var/www/html
```

Step 2: Create your HTML file

```
sudo nano index.html
```

Step 3: Add HTML content

Paste the following code:

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to My Website</title>
</head>
<body style="text-align:center; background-color: #f0f0f0;">
<h1>Hello from Apache on AWS EC2!</h1>
<p>This is a static website hosted on Apache web server.</p>
</body>
</html>
```

Press Ctrl + O, Enter, then Ctrl + X to save and exit.

After Pressing Ctrl + O You'll see:

File Name to Write: index.html [Just press Enter to confirm saving with that name].

To Exit Nano: press Ctrl + X [This closes the Nano editor and will be back to the Linux prompt]

```
[ec2-user@ip-172-31-25-83 html]$ cat index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to My Website</title>
</head>
<body style="text-align:center; background-color: #f0f0f0;">
<h1>Hello from Apache on AWS EC2!</h1>
<p>This is a static website hosted on Apache web server.</p>
</body>
</html>
[ec2-user@ip-172-31-25-83 html]$ |
```

Step 4: Restart Apache

sudo systemctl restart httpd

Step 5: View Website

- In browser: http://<your-ec2-public-ip>
- You'll see your custom HTML page.



Optional: Add Multiple Pages

- Upload other pages like about.html, contact.html in the same directory.

Steps for launching an EC2 Instance with User Data Script to Automatically Install Apache and Host a Static Website.

In this method, you don't manually install Apache or upload files after connecting.

Instead, you write a **shell script** in the **User Data** section (during instance creation).

When the EC2 instance starts for the first time, it automatically:

1. Installs Apache (httpd)
2. Starts the service
3. Creates an index.html webpage
4. Hosts your website immediately after launch

Step 1: Sign in to AWS Management Console

Go to EC2 Dashboard → Click Launch Instance

Step 2: Name and OS

Name: AutoApacheWebServer

AMI: Choose Amazon Linux 2 AMI (Free tier eligible)
Instance Type: t3.micro

Step 3: Key Pair

Select existing key pair (or create new) → Format = .pem

Step 4: Network Settings

Allow HTTP traffic from the Internet (port 80)
Allow SSH traffic (port 22)

Step 5: Add User Data Script

Scroll down to Advanced Details → User data box.

This script:

- Updates all packages
- Installs and starts Apache
- Creates a sample index.html webpage in /var/www/html

Paste the following shell script:

```
#!/bin/bash
# Update packages
yum update -y

# Install Apache Web Server
yum install -y httpd

# Start Apache
systemctl start httpd
systemctl enable httpd

# Create website content
echo "<html>
<head><title>Welcome to My Auto Web Server</title></head>
<body style='text-align:center; background-color:#e9f5ff;'>
<h1>Welcome to EC2 Instance</h1>
<h2>Apache Installed Automatically via User Data Script</h2>
<p>This is a static website hosted on EC2 using User Data automation.</p>
</body>
</html>" > /var/www/html/index.html
```

Step 6: Launch the Instance

Click Launch Instance
Wait for the status → Running

Step 7: Test Your Web Server

Copy the Public IPv4 address of your instance.

Paste it in your browser: <http://<your-public-ip>>

You should immediately see your webpage without logging into EC2!

Date: 7-11-2025

Exercise-8: Create a Custom AMI from a Working EC2 Instance.

Launch an EC2 Instance using a Custom AMI

Delete the custom AMI

AMI (Amazon Machine Image) is a pre-configured template that contains only Operating System (e.g., Amazon Linux, Ubuntu, Windows) needed to launch an EC2 instance.

When you launch a new EC2 instance, you select an AMI as the base image.

A **Custom AMI** is created from your *own running EC2 instance* after you've installed software, uploaded website files, or applied settings.

Benefits:

1. **Reusability** – Launch multiple identical servers quickly.
2. **Backup** – Acts as a snapshot of your configured instance.
3. **Auto Scaling** – Used by Auto Scaling Groups to create identical instances automatically.
4. **Disaster Recovery** – You can recreate your setup if the original instance fails.
5. **Time-saving** – No need to reinstall Apache or re-upload files each time.

A Custom AMI is like a master copy of your EC2 setup. It ensures your website or app can be duplicated instantly and consistently.

Steps to Create a Custom AMI from a Working EC2 Instance

To capture the current configuration — installed packages, website files, and settings — into a reusable Amazon Machine Image (AMI).

Step 1: Select the running instance

Go to EC2 → Instances.

Select the instance that already hosts your website.

Or create an instance (add user data for web hosting)

EC2 > Instances > Launch an instance

Name and tags Info

Name Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recents Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian [Browse more AMIs](#)

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI
ami-0715b650e07377859 (64-bit (x86), uefi-preferred) / ami-014d443f4927eb52 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Summary

Number of instances Info
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2... [read more](#)
ami-0715b650e07377859

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Cancel [Launch instance](#) [Preview code](#)

EC2 > Instances > Launch an instance

Instance type Info | [Get advice](#)

Instance type
t3.micro Free tier eligible

Family: t3 2 vCPU 1 GiB Memory Current generation: true
On-Demand Ubuntu Pro base pricing: 0.0167 USD per Hour
On-Demand RHEL base pricing: 0.042 USD per Hour
On-Demand Windows base pricing: 0.0224 USD per Hour
On-Demand SUSE base pricing: 0.0132 USD per Hour
On-Demand Linux base pricing: 0.0132 USD per Hour

All generations [Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

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
 [Create new key pair](#)

Network settings Info

Network Info
vpc-0d59db2a0a680847d [Edit](#)

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Summary

Number of instances Info
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2... [read more](#)
ami-0715b650e07377859

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Cancel [Launch instance](#) [Preview code](#)

EC2 > Instances > Launch an instance

Advanced details

[Info](#)

Domain join directory | [Info](#)

IAM instance profile | [Info](#)

Hostname type | [Info](#)

DNS Hostname | [Info](#)
 Enable IP name IPv4 (A record) DNS requests
 Enable resource-based IPv4 (A record) DNS requests
 Enable resource-based IPv6 (AAAA record) DNS requests

Instance auto-recovery | [Info](#)

Shutdown behavior | [Info](#)

Stop - Hibernate behavior | [Info](#)

Termination protection | [Info](#)

Summary

Number of instances | [Info](#)

Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2... [read more](#)
ami-0715b650e07377859

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

CloudShell Feedback [Console Mobile App](#) © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 > Instances > Launch an instance

2

Allow tags in metadata | [Info](#)

User data - optional | [Info](#)
Upload a file with your user data or enter it in the field.

```
#!/bin/bash
# User data code
# Proceed to install httpd - (Amazon Linux 2 version)
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
```

User data has already been base64 encoded

Summary

Number of instances | [Info](#)

Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2... [read more](#)
ami-0715b650e07377859

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

CloudShell Feedback [Console Mobile App](#) © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 2: Create Image

From the Actions → Image and templates → Create image.

The screenshot shows the AWS EC2 Instances page. A green banner at the top indicates "Successfully initiated termination (deletion) of i-0d59c46244c7b505b, i-0b78f2a4d6dce0a75". Below this, the "Instances (1/3) Info" section shows a table with one row. The first two rows are terminated instances, while the third row, "Am", is a running instance with ID i-09bbacf26a1c98d89. To the right of the table, a context menu is open for the running instance, with options like "Create image", "Create template from instance", and "Launch more like this". The left sidebar contains navigation links for EC2 services like Dashboard, AWS Global View, Events, Instances, Images, and Elastic Block Store.

Step 3: Enter details

Image name: MyWebsiteAMI

Description: AMI created from configured Apache website instance

Leave “No reboot” unchecked (so the filesystem is consistent).

The screenshot shows the "Create image" wizard. Step 1: Image details. It shows the selected instance ID: i-09bbacf26a1c98d89 (Am). The image name is set to "new im". The "Reboot instance" checkbox is unchecked. The "Instance volumes" section is partially visible below.

Step 4: Storage volumes

The root volume will appear automatically; keep defaults unless you need more space.

Step 5: Create image

Click **Create image**.

A confirmation message appears; note the **Image ID**.

Step 6: Verify creation

In left panel → **AMIs** → refresh until **Status = Available**.

Your custom AMI is now saved in that region and can be used to launch identical webserver instances.

Steps to Launch an EC2 Instance using a Custom AMI

To launch a new EC2 instance from a previously created Custom Amazon Machine Image (AMI) — containing your configured website and software.

Step 1: Go to AMIs

Open the EC2 Service dashboard.

In the left navigation pane, click AMIs (under “Images”).

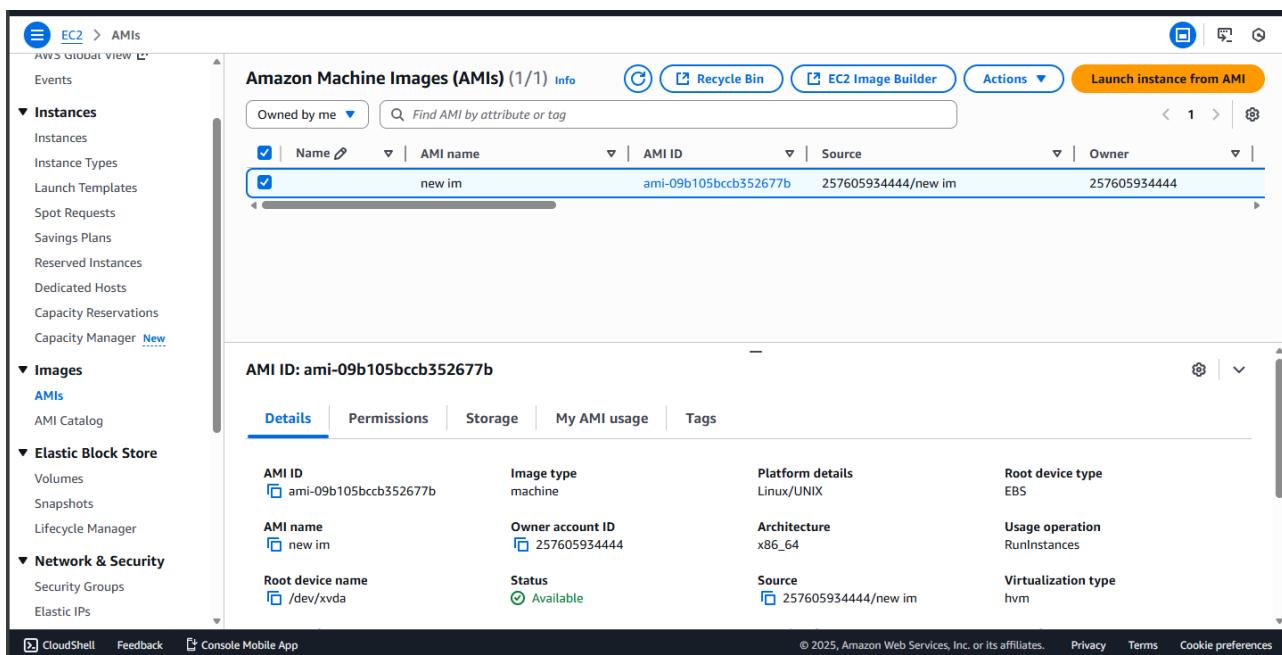
Step 2: Select Your Custom AMI

Locate the AMI you created earlier (e.g., MyWebsiteAMI).

Ensure Status = Available.

Step 3: Launch Instance from AMI

Select the AMI → click Launch instance from image.



The screenshot shows the AWS EC2 service dashboard with the 'AMIs' section selected. The left sidebar includes links for Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, and Capacity Manager. The main content area displays a table titled 'Amazon Machine Images (AMIs) (1/1)'. The table has columns for Name, AMI ID, Source, and Owner. A single row is selected, showing 'new im' as the name, 'ami-09b105bccb352677b' as the AMI ID, '257605934444/new im' as the Source, and '257605934444' as the Owner. Below the table, a detailed view for the selected AMI is shown, with tabs for Details, Permissions, Storage, My AMI usage, and Tags. The 'Details' tab is active, displaying information such as AMI ID, Image type, Platform details, Root device type, AMI name, Owner account ID, Architecture, Usage operation, Root device name, Status, Source, and Virtualization type.

Step 4: Configure Instance Details

Name: WebServer-from-Custom-AMI

Instance type: t3.micro (Free Tier eligible)

Key pair: Choose an existing .pem key for SSH access.

Network settings:

VPC: Default

Subnet: Public

Security Group: Allow HTTP (80) and SSH (22)

The screenshot shows the 'Launch an instance' wizard. In the 'Name and tags' section, the name 'Clon' is entered. In the 'Amazon Machine Image (AMI)' section, the AMI 'new im ami-09b105bccb352677b' is selected. The 'Virtual server type (instance type)' is set to 't3.micro'. The 'Firewall (security group)' dropdown shows 'New security group'. The 'Storage (volumes)' section indicates 1 volume(s) - 8 GiB. The right side displays a summary with 1 instance, the selected AMI, instance type, and a 'Launch instance' button.

Step 5: Storage (EBS Volume)

Keep default root volume (e.g., 8 GB gp3).

The screenshot shows the 'Launch an instance' wizard. In the 'Firewall (security groups)' section, the 'Create security group' option is selected. Under 'Allow traffic from', 'Allow SSH traffic from' is checked, and 'Anywhere' is selected. A warning message states: 'Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.' In the 'Configure storage' section, the root volume is set to 8 GiB gp3. The right side displays a summary with 1 instance, the selected AMI, instance type, and a 'Launch instance' button.

Step 6: Review and Launch

Click Launch instance.

Wait until the instance state becomes Running.

Step 7: Access the Website

Copy the Public IPv4 address from the instance details.

Open in a browser → <http://<Public-IP>>

You should see your same website, confirming the custom AMI works.

A new EC2 instance is successfully launched using the Custom AMI, automatically containing the OS, Apache, configurations, and website files — no manual setup required.

The top screenshot shows the AWS EC2 Instances page. It displays an instance summary for 'i-083b6f2906075d9ca (Clone)'. The instance is running and has a public IPv4 address of 52.62.253.205. The bottom screenshot shows a browser window with the URL 52.62.253.205, which displays the text 'It works!'. This confirms that the custom AMI was successful.

Steps to Delete the custom AMI

Deleting a **custom AMI** involves **deregistering** the image and then **deleting its associated EBS snapshot**.

When you deregister an AMI, it is removed from your account and can no longer be used to launch new instances.

However, the **snapshot** that was created along with the AMI still remains in your storage and **continues to incur charges, so you must delete it separately** to free up space and stop costs.

This ensures your AWS environment remains clean and cost-efficient.

Step 1 – Open EC2 Dashboard

Sign in to the AWS Management Console.

Navigate to EC2 service.

In the left navigation pane, scroll down to Images → AMIs.

Step 2 – Locate Your Custom AMI

In the Owned by me tab, you will see all AMIs you created (custom AMIs).

Select the AMI you want to delete.

You can identify it by Name or AMI ID.

Step 3 – Deregister the AMI

Select the AMI → Click on Actions dropdown.

Choose Deregister AMI.

Confirm by clicking Deregister in the pop-up.

This removes the AMI record, but not the underlying snapshot(s).

The screenshot shows the AWS EC2 console with the 'AMIs' section selected. A single AMI named 'new im' is listed. A context menu is open over this AMI, with the 'Deregister AMI' option highlighted. Below the main list, a detailed view of the AMI 'new im' is shown with its ID, name, owner account ID, architecture, source, and other details. At the bottom, a 'Deregister AMI' dialog box is displayed, containing the following text and options:

Deregister AMI

After you deregister an AMI, you can't use it to launch new instances.

Are you sure that you want to deregister these AMIs?

ami-09b105bccb352677b

Delete associated snapshots

▶ Associated snapshots (1)

Cancel **Deregister AMI**

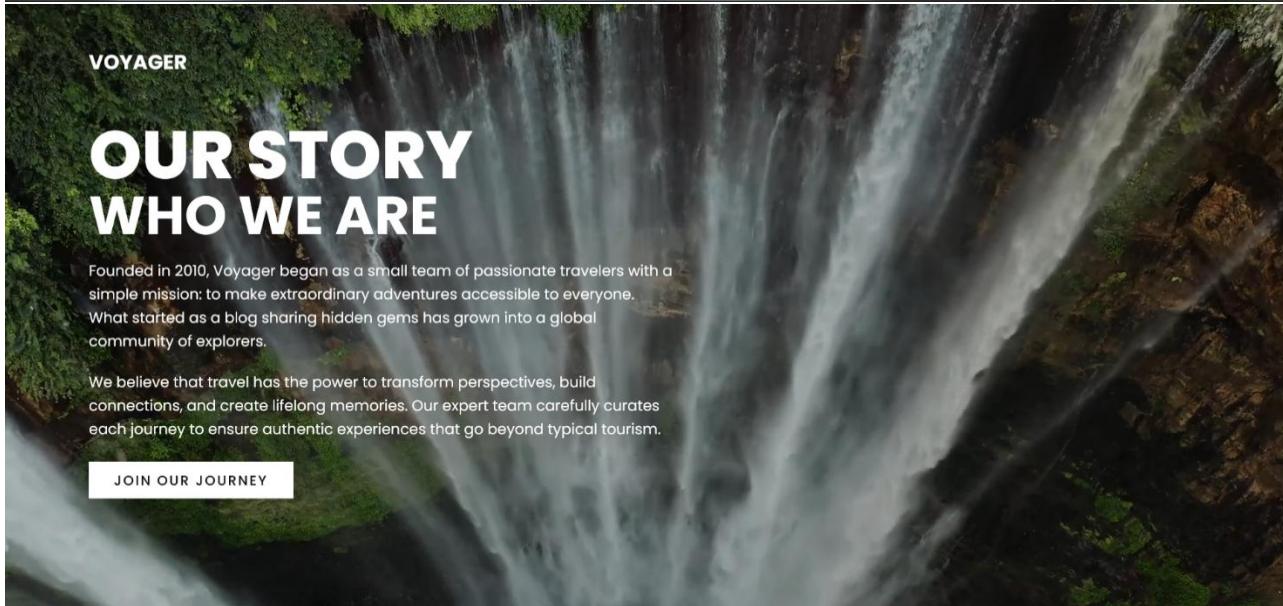
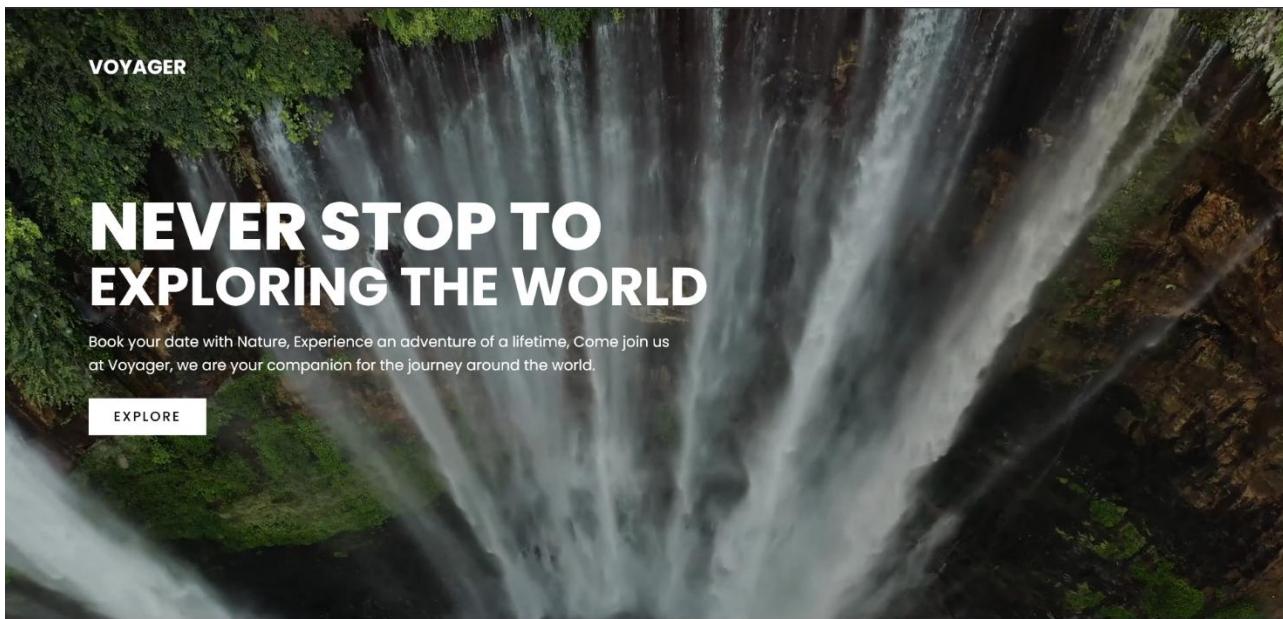
Step 4 – Delete the Associated Snapshot

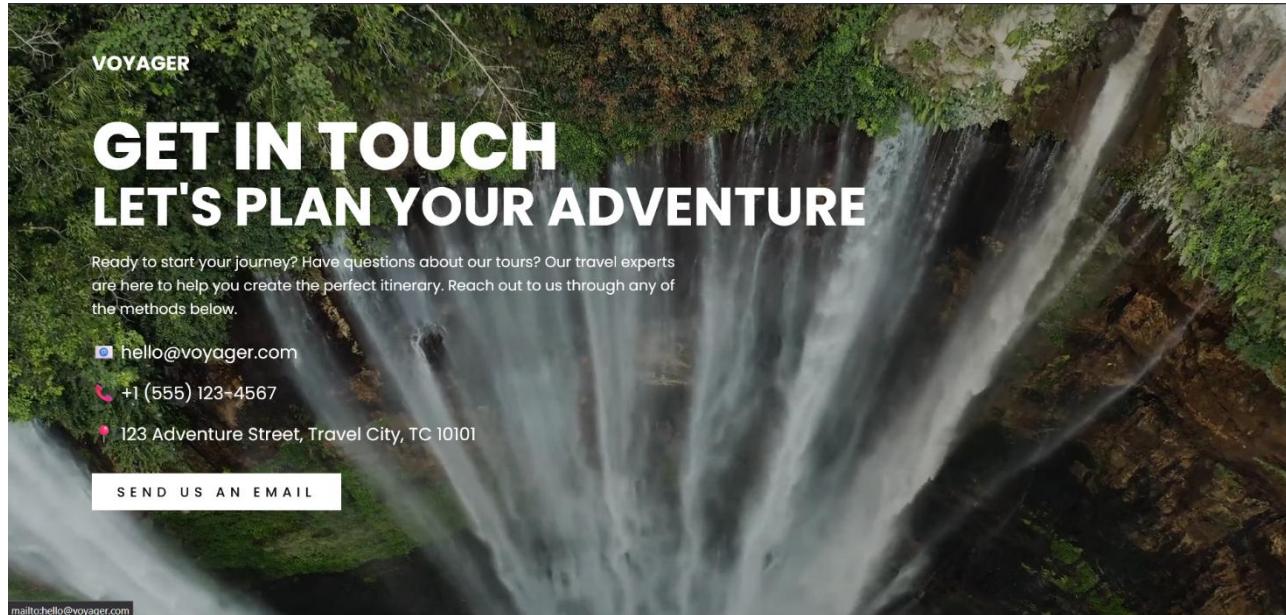
To fully free up storage space (and avoid charges):

- The AMI is now deregistered (unavailable for future instance launches).
- The snapshot is also deleted, freeing up EBS storage and costs.

Date: 11-11-2025

Mini project assessment





VOYAGER

GET IN TOUCH LET'S PLAN YOUR ADVENTURE

Ready to start your journey? Have questions about our tours? Our travel experts are here to help you create the perfect itinerary. Reach out to us through any of the methods below.

hello@voyager.com

+1 (555) 123-4567

123 Adventure Street, Travel City, TC 10101

SEND US AN EMAIL

mailto:hello@voyager.com

aws | Search [Alt+S] Account ID: 2576-0593-4444 ▾ Joshua Asia Pacific (Sydney) ▾

EC2 > Instances > i-071d445f3365b8828

EC2

- Dashboard
- AWS Global View
- Events
- Instances
 - Instances
 - Instance Types
 - Launch Templates
 - Spot Requests
 - Savings Plans
 - Reserved Instances
 - Dedicated Hosts
 - Capacity Reservations
 - Capacity Manager [New](#)
- Images
 - AMIs
 - AMI Catalog
- Elastic Block Store
 - Volumes
 - Cross-Region

Instance summary for i-071d445f3365b8828 (My W Server) [Info](#)

Updated less than a minute ago

| Attribute | Value |
|----------------------------------|---|
| Instance ID | i-071d445f3365b8828 |
| IPv6 address | - |
| Hostname type | IP name: ip-172-31-26-55.ap-southeast-2.compute.internal |
| Answer private resource DNS name | IPv4 (A) |
| Auto-assigned IP address | 3.106.254.214 [Public IP] |
| IAM Role | - |
| IMDSv2 | Required |
| Public IPv4 address | 3.106.254.214 open address |
| Instance state | Running |
| Private IP DNS name (IPv4 only) | ip-172-31-26-55.ap-southeast-2.compute.internal |
| Instance type | t3.micro |
| VPC ID | vpc-0d59db2a0a680847d |
| Subnet ID | subnet-08ffeff18234bba5 |
| Instance ARN | arn:aws:ec2:ap-southeast-2:257605934444:instance/i-071d445f3365b8828 |
| Private IPv4 addresses | 172.31.26.55 |
| Public DNS | ec2-3-106-254-214.ap-southeast-2.compute.amazonaws.com open address |
| Elastic IP addresses | - |
| AWS Compute Optimizer finding | Opt-in to AWS Compute Optimizer for recommendations. Learn more |
| Auto Scaling Group name | - |
| Managed | false |

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows two pages from the Amazon S3 console:

- General purpose buckets (1) Info**: This page lists a single bucket named "amzn-pro-bucket". The bucket was created on November 11, 2025, at 18:02:58 (UTC+05:30) in the Asia Pacific (Sydney) region.
- amzn-pro-bucket Info**: This page displays the contents of the "amzn-pro-bucket". It contains three objects:

| Name | Type | Last modified | Size | Storage class |
|-----------------------|------|---|---------|---------------|
| land.css | css | November 11, 2025, 18:03:57 (UTC+05:30) | 3.5 KB | Standard |
| land.js | js | November 11, 2025, 18:03:57 (UTC+05:30) | 242.0 B | Standard |
| Waterfall - 37088.mp4 | mp4 | November 11, 2025, 18:04:06 (UTC+05:30) | 13.4 MB | Standard |

Amazon S3 sidebar:

- General purpose buckets
- Directory buckets
- Table buckets
- Vector buckets
- Access Grants
- Access Points (General Purpose Buckets, FSx file systems)
- Access Points (Directory Buckets)
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3
- Block Public Access settings for this account
- Storage Lens**
 - Dashboards
 - Storage Lens groups
 - AWS Organizations settings

Footer:

- CloudShell
- Feedback
- Console Mobile App
- © 2025, Amazon Web Services, Inc. or its affiliates.
- Privacy
- Terms
- Cookie preferences

S3 Bucket is used to store the files such as .css .js .jpeg. These are made public so that the link can be used and copy pasted into the html files using href and src attributes.

Date: 12-11-2025

Exercise-10: Creation and Configuration of a Custom AWS Virtual Private Cloud (VPC)

[Including Public and Private Subnets, Internet Gateway, NAT Gateway, Route Tables]

Virtual Private Cloud [VPC]

It is a logically isolated section of the AWS Cloud where you can launch your AWS resources (like EC2 instances, databases, etc.) in a customized, secure network environment — similar to having your own private data center inside AWS.

It is a virtual network dedicated to your AWS account.

It gives you complete control over your networking environment, including IP address ranges, subnets, route tables, and network gateways.

Components of a VPC

| Component | Description |
|-------------------------------------|---|
| CIDR Block (IP Range) | The range of IP addresses for your VPC (e.g., 10.0.0.0/16). |
| Subnets | Smaller divisions inside your VPC; can be Public (accessible from internet) or Private (internal only). |
| Internet Gateway (IGW) | Allows internet access for resources in public subnets. |
| Route Tables | Define how traffic is directed between subnets and gateways. |
| NAT Gateway / NAT Instance | Enables instances in private subnets to connect to the internet without being exposed. |
| Security Groups | Virtual firewalls that control inbound/outbound traffic at the instance level. |
| Network ACLs (Access Control Lists) | Additional firewall at the subnet level. |
| VPC Peering | Connects two VPCs so they can communicate privately. |

Default VPC and Custom VPC

Characteristics of Default VPC

When you first create an AWS account, AWS automatically creates a default VPC for you in each region.

| Feature | Description |
|-----------------------|---|
| Best For | Beginners, quick testing, learning environments, or temporary setups. |
| Created Automatically | One Default VPC per AWS Region (created by AWS). |
| Ready to Use | You can launch EC2 instances immediately — no setup needed. |
| CIDR Block | Always uses 172.31.0.0/16. |

| | |
|-------------------------|---|
| Subnets | One default subnet in each Availability Zone within the region. |
| Internet Connectivity | Each default subnet is a public subnet (has a route to Internet Gateway). |
| Route Table | Already configured to connect to the Internet Gateway. |
| Security Groups & NACLs | Default ones are automatically created and allow basic communication. |

Characteristics of Custom VPC

A Custom VPC is created manually by the user to have complete control over the network configuration.

| Feature | Description |
|-----------------------|--|
| Best For | Production environments, enterprise setups, or multi-tier architectures. |
| Created Manually | You define the VPC and its settings yourself. |
| CIDR Block | You can choose your own IP range (e.g., 10.0.0.0/16). |
| Subnets | You decide how many subnets, and whether they are public or private. |
| Internet Connectivity | You attach your own Internet Gateway. |
| Route Tables | Must be created and configured manually. |
| Security | You can create custom Security Groups and NACLs as needed. |

Subnets

- Subdivisions inside a VPC, used to organize resources.
- Each subnet belongs to one Availability Zone.
- Two types:
 - **Public Subnet** → Connected to Internet Gateway; for web servers.
 - **Private Subnet** → No direct internet access; for databases or internal apps.
- CIDR examples:
 - Public: 10.0.1.0/24
 - Private: 10.0.2.0/24

Internet Gateway (IGW)

- A gateway that connects your VPC to the Internet.
- Required for instances in a public subnet to receive internet traffic.

- Must be attached to the VPC and referenced in the route table.
- Supports bi-directional communication (inbound and outbound).

NAT Gateway

- Enables outbound internet access for private subnet instances.
- Allows downloads and updates (e.g., OS patches) without exposing private IPs.
- Deployed inside a public subnet.
- Needs an Elastic IP for internet access.
- Traffic is one-way: private → internet only (not vice versa).

Route Tables

- Define rules (routes) that determine where network traffic goes.
- Each subnet must be associated with one route table.
- Common routes:
 - For public subnet → 0.0.0.0/0 → Internet Gateway
 - For private subnet → 0.0.0.0/0 → NAT Gateway
- Ensures proper separation between public and private networks.

Security Groups

- **Instance-level firewalls** controlling inbound and outbound traffic.
- Stateful: if inbound traffic is allowed, corresponding outbound is automatically allowed.
- Rules are based on protocol, port number, and source/destination.
- Example:
 - WebServer-SG: allows HTTP(80), HTTPS(443), SSH(22)
 - Database-SG: allows MySQL(3306) from WebServer-SG only

Network ACL (Access Control List)

- **Subnet-level firewall**, acts as an additional layer of security.
- **Stateless**: inbound and outbound rules must be defined separately.
- Default NACL allows all traffic; custom NACLs can be restrictive.
- Used for fine-grained control or compliance environments.

EC2 Instances

- Virtual machines running inside your subnets.
- Public subnet → hosts Web Server (Apache).
- Private subnet → hosts Database Server (MySQL).
- Public EC2s get a public IP; private ones use private IPs only.
- Controlled by their Security Groups and Route Tables.

Elastic IP (EIP)

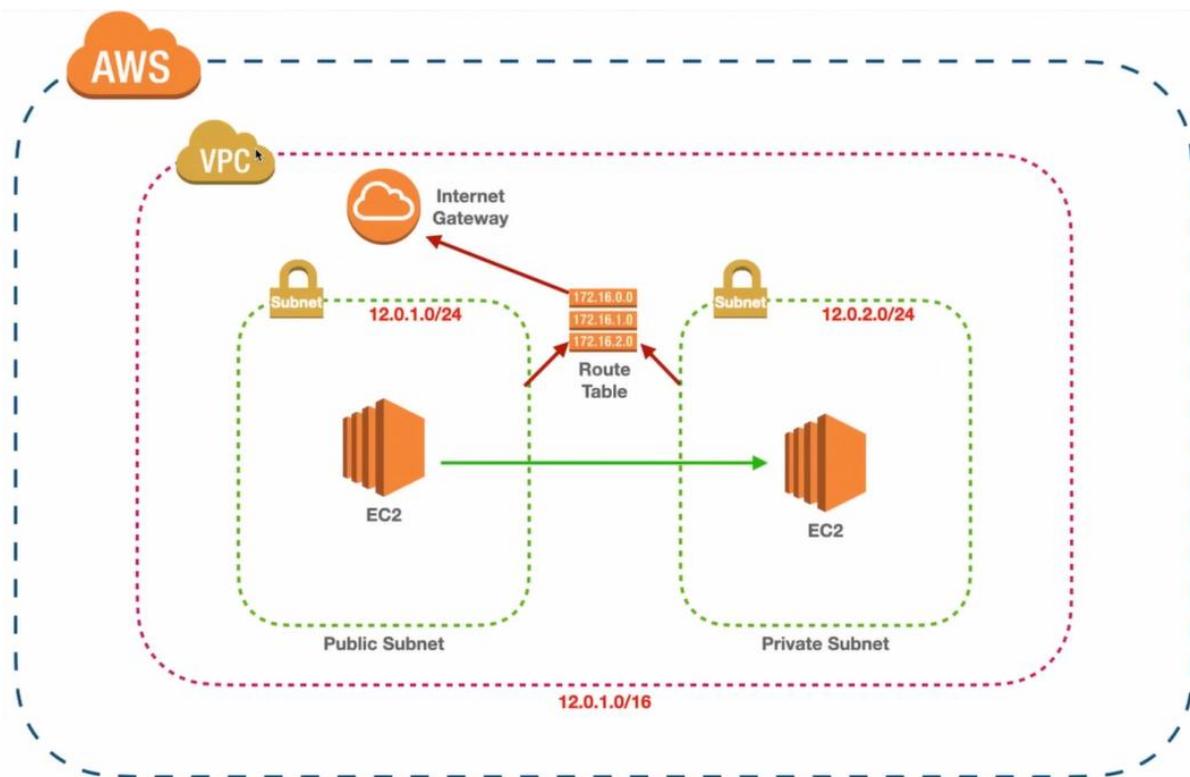
- A **static, public IPv4 address** that can be attached to an EC2 or NAT Gateway.
- Remains constant even if the instance is stopped or restarted.
- Useful for **NAT Gateways**, load balancers, or fixed server access.

Example: Two-Tier Architecture

| Tier | Subnet | Function | Internet Access |
|---------------|----------------|----------------------------|---------------------------|
| Web Tier | Public Subnet | Hosts front-end web server | Yes (via IGW) |
| Database Tier | Private Subnet | Hosts back-end database | Outbound only (via NATGW) |

Create a new VPC for a web application:

- One **Public Subnet** for web servers (via Internet Gateway)
- One **Private Subnet** for databases (via NAT Gateway)
- Custom route tables and tighter security rules.



Objective

To design and configure a **Virtual Private Cloud (VPC)** in AWS with:

- One **Public Subnet** for web servers (via **Internet Gateway**)
- One **Private Subnet** for databases (via **NAT Gateway**)
- Separate **Route Tables** for public and private subnets
- Custom **Security Groups** to enforce network isolation

Step 1 – Create a New VPC

- Open AWS Management Console → VPC.
- Click Create VPC.
- Choose VPC only option.
- Enter:
 - Name → MyWebAppVPC
 - IPv4 CIDR block → 10.0.0.0/16
 - Tenancy → Default
- Click Create VPC.

This allocates a private IP range for your virtual network.

The screenshot shows the AWS VPC Dashboard. On the left, there's a sidebar with navigation links for VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, and Route servers. Below that is a section for Security with Network ACLs and Security groups. At the bottom of the sidebar are links for CloudShell, Feedback, and Console Mobile App.

The main area displays "Resources by Region" for the N. Virginia region. It shows counts for various resources: VPCs (1), NAT Gateways (0), Subnets (6), VPC Peering Connections (0), Route Tables (1), Network ACLs (1), Internet Gateways (1), Security Groups (1), Egress-only Internet Gateways (0), and Customer Gateways (0). Each resource type has a link to "See all regions".

On the right side, there are several boxes: "Service Health" (View complete service health details), "Settings" (Block Public Access, Zones, Console Experiments), "Additional Information" (VPC Documentation, All VPC Resources, Forums, Report an Issue), and "AWS Network Manager" (AWS Network Manager provides). At the bottom of the dashboard, there are links for CloudShell, Feedback, and Console Mobile App, along with copyright information: © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

The screenshot shows the 'Create VPC' configuration page in the AWS Management Console. At the top, the navigation path is 'VPC > Your VPCs > Create VPC'. Below the title 'Create VPC' is a descriptive text: 'A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.' The main section is titled 'VPC settings'. Under 'Resources to create', the 'VPC only' option is selected. In the 'Name tag - optional' field, 'MyWebAppVPC' is entered. The 'IPv4 CIDR block' section shows '10.0.0.0/16' as the input, with a note that the size must be between /16 and /28. The 'IPv6 CIDR block' section has 'No IPv6 CIDR block' selected. At the bottom, there are 'Next Step' and 'Cancel' buttons.

VPC > Your VPCs > Create VPC

Create VPC [Info](#)

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.

VPC settings

Resources to create [Info](#)

Create only the VPC resource or the VPC and other networking resources.

VPC only VPC and more

Name tag - optional

Creates a tag with a key of 'Name' and a value that you specify.

MyWebAppVPC

IPv4 CIDR block [Info](#)

IPv4 CIDR manual input IPAM-allocated IPv4 CIDR block

IPv4 CIDR

10.0.0.0/16

CIDR block size must be between /16 and /28.

IPv6 CIDR block [Info](#)

No IPv6 CIDR block IPAM-allocated IPv6 CIDR block Amazon-provided IPv6 CIDR block IPv6 CIDR owned by me

X [Cancel](#) [Next Step](#)

Step 2 – Create Subnets

We'll create two subnets inside the VPC.

The screenshot shows the AWS VPC Subnets page. The left sidebar has sections for Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, Route servers), Security (Network ACLs, Security groups), and a CloudShell link. The main content area shows a table of 6 subnets with columns for Name and Subnet ID. The subnets listed are: - (subnet-05d7f1b9eaaf25b1d), - (subnet-06e14b5948a561fa6), - (subnet-0b614af667cff5875), - (subnet-03135310e24071c58), - (subnet-08d67912f4a2cf627). A search bar at the top right says "Find subnets by attribute or tag". A "Create subnet" button is at the top right. Below the table is a section titled "Select a subnet".

| Name | Subnet ID |
|------|--|
| - | subnet-05d7f1b9eaaf25b1d |
| - | subnet-06e14b5948a561fa6 |
| - | subnet-0b614af667cff5875 |
| - | subnet-03135310e24071c58 |
| - | subnet-08d67912f4a2cf627 |

(a) Public Subnet

- Go to Subnets → Create subnet.
- Select VPC: MyWebAppVPC.
- Choose Availability Zone A (e.g., ap-south-1a).
- Enter:
 - Name → PublicSubnet
 - IPv4 CIDR block → 10.0.1.0/24
- Click Create subnet.

VPC > Subnets > Create subnet

Create subnet Info

VPC

VPC ID
Create subnets in this VPC.
vpc-03bbe6bae2aaaf3f47 (MyWebAppVPC)

Associated VPC CIDRs

IPv4 CIDRs
10.0.0.0/16

Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1

Subnet name
Create a tag with a key of 'Name' and a value that you specify.
PublicSubnet
The name can be up to 256 characters long.

Availability Zone Info
Choose the zone in which your subnet will reside, or let Amazon choose one for you.
United States (N. Virginia) / us-east-1a
CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

IPv4 VPC CIDR block Info
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.
10.0.0.0/16

IPv4 subnet CIDR block
10.0.1.0/24 256 IPs
< > ^ v

Tags - optional

| Key | Value - optional |
|------|------------------|
| Name | PublicSubnet |

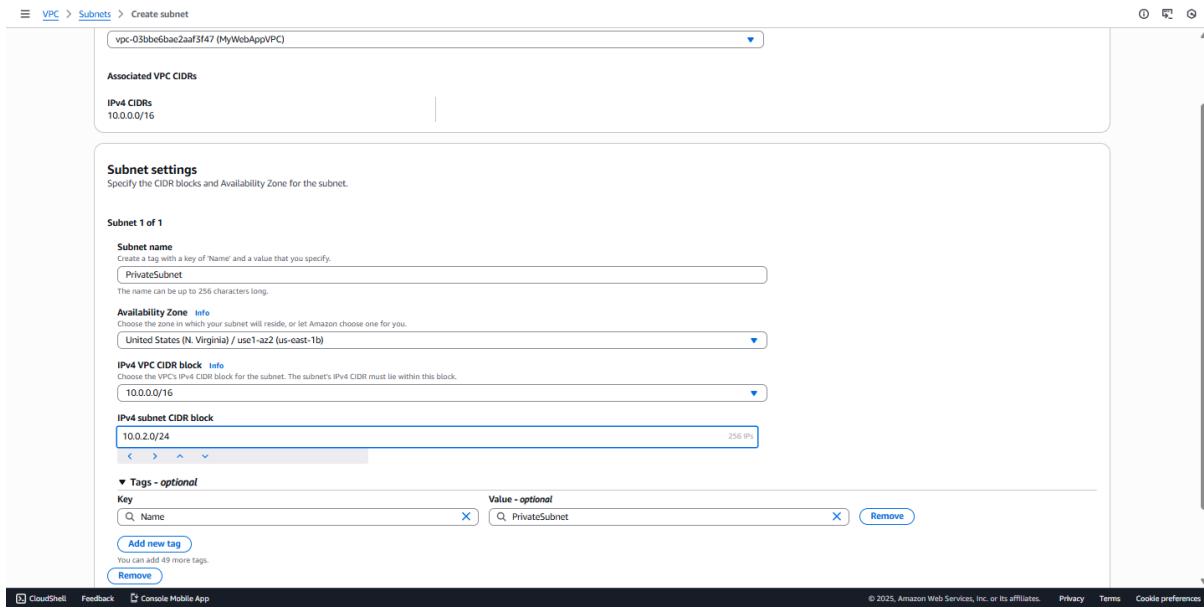
Add new tag
You can add 49 more tags.
Remove
Add new subnet

Cancel **Create subnet**

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

(b) Private Subnet

- Click Create subnet.
- Select same VPC.
- Choose Availability Zone B.
- Enter:
 - Name → PrivateSubnet
 - IPv4 CIDR block → 10.0.2.0/24
- Click Create subnet.



Step 3 – Create and Attach an Internet Gateway

- In the left navigation pane, scroll down and click on “Internet Gateways”.
- Internet Gateways → Create Internet Gateway.
 - Name: MyWebApp-IGW
- Click Create Internet Gateway

At this point, the IGW exists but is not yet connected to your VPC.

The screenshot shows the AWS VPC Internet Gateways page. On the left, there's a navigation sidebar with sections like VPC dashboard, Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, Route servers), Security (Network ACLs, Security groups), and PrivateLink and Lattice (Getting started, Endpoints, Endpoint services, Service networks, Lattice services, Resource configurations, Resource gateways, Target groups). The main area displays a table titled "Internet gateways (1) Info". The table has columns: Name, Internet gateway ID, State, and VPC ID. One row is listed: Name is empty, Internet gateway ID is "igw-0efcdffd17288714d", State is "Attached", and VPC ID is "vpc-0e06f4". Below the table, a message says "Select an internet gateway above". At the bottom of the page, there are links for CloudShell, Feedback, and Console Mobile App, along with copyright information: © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

The screenshot shows the "Create internet gateway" settings page. The title is "Create internet gateway" with an "Info" link. A sub-instruction says "An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below." The "Internet gateway settings" section contains a "Name tag" input field where "MyWebApp-IGW" is typed. The "Tags - optional" section explains what tags are and provides a form to add them. A table shows a single tag: Key "Name" and Value "MyWebApp-IGW". There's a "Remove" button next to the value. Below the table, it says "You can add 49 more tags." At the bottom right, there are "Cancel" and "Create internet gateway" buttons.

- **Attach the Internet Gateway to Your VPC**

- Select the IGW you just created (checkbox).
- Click on the “Actions” drop-down.
- Choose “Attach to VPC.”
- From the list, select your **VPC name**, e.g. MyWebAppVPC.
- Click “Attach Internet Gateway.”

Now the IGW is linked to your VPC and the public subnet can reach the internet.

The screenshot shows the AWS VPC console. On the left, there's a sidebar with 'VPC dashboard' and a 'Virtual private cloud' section containing 'Your VPCs', 'Subnets', 'Route tables', 'Internet gateways' (which is selected), and 'Egress-only internet gateways'. The main area is titled 'Internet gateways (1/2)'. It lists one gateway: 'MyWebApp-IGW' (ID: igw-04223e748cee73ba0). A context menu is open over this entry, with 'Attach to VPC' highlighted. Below this, the 'Actions' dropdown shows options like 'View details', 'Detach from VPC', 'Manage tags', and 'Delete internet gateway'. At the bottom of the main area, it says 'Attach to VPC (igw-04223e748cee73ba0)'. A sub-menu is open here, showing a search bar with 'vpc-03bbe6bae2aaaf3f47' and a 'Cancel' button. There's also an 'AWS Command Line Interface command' link. At the very bottom right of the sub-menu is a large orange 'Attach internet gateway' button.

Step 4 – Create a NAT Gateway

- Go to NAT Gateways → Create NAT Gateway.
- Choose:
 - Subnet → PublicSubnet
 - Elastic IP Allocation ID → Click Allocate new Elastic IP
- Name → MyWebApp-NATGW.
- Click Create NAT Gateway.

This allows instances in private subnet to access the internet (e.g., for updates) without being exposed.

Note: NAT Gateway is a paid resource.

VPC > NAT gateways > Create NAT gateway

Elastic IP address 3.213.231.249 (eipalloc-02c0b83c38d3a4988) allocated.

Create NAT gateway Info

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the internet.

NAT gateway settings

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

 The name can be up to 256 characters long.

Subnet
 Select a subnet in which to create the NAT gateway.

Connectivity type
 Select a connectivity type for the NAT gateway.
 Public
 Private

Elastic IP allocation ID Info
 Assign an Elastic IP address to the NAT gateway.

Additional settings Info

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 |
|--|--|
| <input type="text" value="Name"/> <input type="button" value="X"/> | <input type="text" value="MyWebApp-NATGW"/> <input type="button" value="X"/> <input type="button" value="Remove"/> |

You can add 49 more tags.

Step 5 – Configure Route Tables

(a) Public Route Table

- Go to Route Tables → Create route table.
 - Name → PublicRT
 - VPC → MyWebAppVPC
- Under Routes → Edit routes → Add route
 - Destination: 0.0.0.0/0
 - Target: Internet Gateway (MyWebApp-IGW)
- Under Subnet Associations → Edit subnet associations → Select PublicSubnet → Save.

Now the PublicSubnet has internet access.

VPC > Route tables > rtb-0d1616c48a04efa8c

rtb-0d1616c48a04efa8c / PublicRT

Details Info

| | | | |
|---|-------------------------------------|---|------------------------|
| Route table ID rtb-0d1616c48a04efa8c | Main <input type="checkbox"/> No | Explicit subnet associations subnet-03fab818deb823f46 / PublicSubnet | Edge associations – |
| VPC vpc-03bbe6bae2aaaf3f47 MyWebAppVPC | Owner ID 257605934444 | | |

Routes Subnet associations Edge associations Route propagation Tags

Routes (2)

| Destination | Target | Status | Propagated | Route Origin |
|-------------|-------------------|--------|------------|--------------------|
| 0.0.0.0/0 | igw-04223e748c... | Active | No | Create Route |
| 10.0.0.0/16 | local | Active | No | Create Route Table |

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

VPC > Route tables > rtb-0d1616c48a04efa8c > Edit subnet associations

Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (1/2)

| Name | Subnet ID | IPv4 CIDR | IPv6 CIDR | Route table ID |
|---------------|--------------------------|-------------|-----------|----------------------------------|
| PublicSubnet | subnet-03fab818deb823f46 | 10.0.1.0/24 | – | rtb-0d1616c48a04efa8c / PublicRT |
| PrivateSubnet | subnet-0ccc95561054a7... | 10.0.2.0/24 | – | Main (rtb-010770c542d927db3) |

Selected subnets

subnet-03fab818deb823f46 / PublicSubnet

(b) Private Route Table

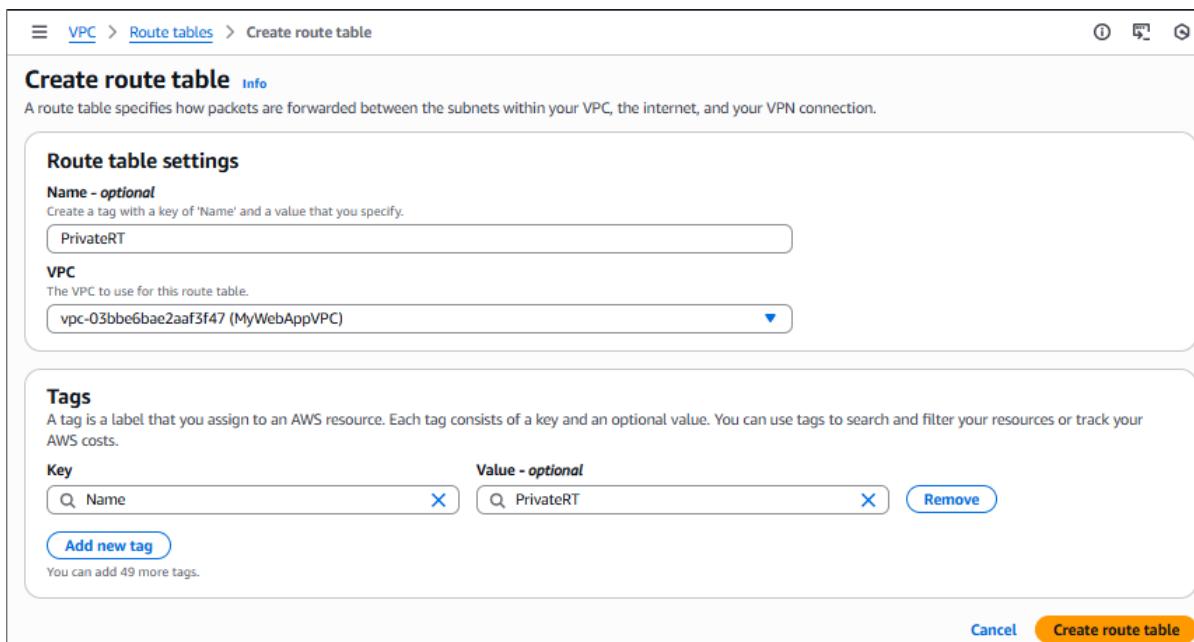
- Create another Route Table → Name PrivateRT.
- Under Routes → Edit routes → Add route
 - Destination: 0.0.0.0/0
 - Target: NAT Gateway (MyWebApp-NATGW)
- Under Subnet Associations → Select PrivateSubnet → Save.

PrivateSubnet traffic goes through the NAT Gateway.

Note:

Each subnet in a VPC can be associated with **only one route table**.

If a subnet is not explicitly associated with a custom table, it will automatically use the **Main Route Table** created by default with the VPC.



The screenshot shows the 'Create route table' wizard in the AWS VPC console. The top navigation bar includes 'VPC', 'Route tables', and 'Create route table'. The main section is titled 'Create route table' with an 'Info' link. A descriptive text states: 'A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.' Below this is the 'Route table settings' section, which includes fields for 'Name - optional' (containing 'PrivateRT') and 'VPC' (containing 'vpc-03bbe6bae2aaaf3f47 (MyWebAppVPC)'). The 'Tags' section allows adding key-value pairs; one tag ('Name' key, 'PrivateRT value') is already present. A note says 'You can add 49 more tags.' At the bottom right are 'Cancel' and 'Create route table' buttons.

VPC > Route tables > rtb-0afc55038afabebc4 > Edit routes

Edit routes

| | | |
|--|----------------------------------|---------------|
| Route 1 | Target | Status |
| Destination 10.0.0.0/16 | local | Active |
| Propagated No | Route Origin CreateRouteTable | |
| Route 2 | Target | Status |
| Destination 0.0.0.0/0 | NAT Gateway | - |
| Propagated No | Route Origin CreateRoute | |
| Add route Remove | | |

[Cancel](#) [Preview](#) [Save changes](#)

VPC > Route tables > rtb-0afc55038afabebc4 > Edit subnet associations

Edit subnet associations

Change which subnets are associated with this route table.

| Available subnets (1/2) | | | | | |
|---|--------------------------|-------------|-----------|--------------------------------|-------------------------------------|
| <input type="text"/> Filter subnet associations | | | | | |
| Name | Subnet ID | IPv4 CIDR | IPv6 CIDR | Route table ID | |
| PublicSubnet | subnet-03fab818deb823... | 10.0.1.0/24 | - | rtb-0d1616c48a04efa8c / Public | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> PrivateSubnet | subnet-0ccc95561054a7... | 10.0.2.0/24 | - | Main (rtb-010770c542d927db3) | <input checked="" type="checkbox"/> |

Selected subnets

subnet-0ccc95561054a73c0 / PrivateSubnet [X](#)

[Cancel](#) [Save associations](#)

Step 6 – Create Security Groups

(a) WebServer-SG

1. Go to Security Groups → Create security group.
 - o Name → WebServer-SG
 - o Description → Allow HTTP, HTTPS, SSH
 - o VPC → MyWebAppVPC

The screenshot shows the AWS VPC Security Groups page. On the left, there's a navigation sidebar with options like VPC dashboard, AWS 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, NAT gateways, Peering connections, Route servers), and Security (Network ACLs, Security groups). The main area displays a table of security groups:

| Name | Security group ID | Security group name | VPC ID |
|------|--------------------------------------|---------------------|-----------|
| - | sg-083284a83f850421e | default | vpc-0e... |
| - | sg-0f49113c54f74e79c | default | vpc-0e... |

2. Inbound Rules:

| Type | Protocol | Port Range | Source |
|-------|----------|------------|-----------|
| SSH | TCP | 22 | My IP |
| HTTP | TCP | 80 | 0.0.0.0/0 |
| HTTPS | TCP | 443 | 0.0.0.0/0 |

The screenshot shows the 'Create security group' wizard. It has two main sections: 'Basic details' and 'Inbound rules'.

Basic details:

- Security group name: WebServer-SG
- Description: Allow HTTP, HTTPS, SSH
- VPC: vpc-03bbe6bac2aa3f3f47 (MyWebAppVPC)

Inbound rules:

| Type | Protocol | Port range | Source | Description - optional |
|-------|----------|------------|----------|------------------------|
| SSH | TCP | 22 | My IP | 49.206.243.162/32 |
| HTTP | TCP | 80 | Anywhere | 0.0.0.0/0 |
| HTTPS | TCP | 443 | Anywhere | 0.0.0.0/0 |

A note at the bottom states: "⚠️ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only."

(b) Database-SG

- Create another → Database-SG.
 - Allow MySQL/Aurora (TCP 3306) only from WebServer-SG.
 - No public inbound access.

Enforces least-privilege network communication.

Security Note:

Security Groups are **stateful** — if you allow incoming traffic, the return traffic is automatically permitted. Network ACLs, however, are **stateless** — inbound and outbound rules must be configured separately.

DATE: 14-11-2025

Exercise-11

Launching and Configuring EC2 Instances for Web Server in Public Subnet and Database Server in Private Subnet Using NAT Gateway for Outbound Access

Step 1 — Launch Windows instance in Public Subnet

- Go to EC2 → Launch Instance.
- Choose:
 - AMI → Windows Server (Free tier eligible)
 - Instance type → t2.micro / t3.micro
- Select:
 - VPC → MyWebAppVPC
 - Subnet → PublicSubnet
 - Auto assign Public IP → Enable
- Security Group:
 - Create WebInstance-SG
 - Allow:
 - RDP (3389) → Anywhere (for practice)
- Launch instance using key pair (.pem).

EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

WebServe

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

Search our full catalog including 1000s of application and OS images

Recents Quick Start

| | | | | | |
|--------------|-------|--------|---------|---------|------------|
| Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Linux |
| | | | | | |

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

CloudShell Feedback Console Mobile App Privacy Terms Cookie preferences

© 2025, Amazon Web Services, Inc. or its affiliates.

EC2 > Instances > Launch an instance

Network settings [Info](#)

VPC - required [Info](#)

vpc-0503646573ad5bb82 (Mywebappvpc)
10.0.0.0/16

Subnet [Info](#)

subnet-00529d6a3ae0b109c Public Subnet
VPC: vpc-0503646573ad5bb82 Owner: 257605934444
Availability Zone: us-east-1a (use1-az1) Zone type: Availability Zone
IP addresses available: 251 CIDR: 10.0.1.0/24

Create new subnet [Create new subnet](#)

Auto-assign public IP [Info](#)

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

Security group name - required

launch-wizard-2

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#@[]+=;&;!\$*

Description - required [Info](#)

launch-wizard-2 created 2025-11-25T06:54:58.899Z

Inbound Security Group Rules

[CloudShell](#) [Feedback](#) [Console Mobile App](#) [Privacy](#) [Terms](#) [Cookie preferences](#)

© 2025, Amazon Web Services, Inc. or its affiliates.

The screenshot shows the AWS EC2 Instances Launch an instance wizard. The current step is "Configure security group". The "Inbound Security Group Rules" section contains one rule:

- Type: rdp
- Protocol: TCP
- Port range: 3389
- Source type: Anywhere
- Source: 0.0.0.0/0
- Description: e.g. SSH for admin desktop

A warning message is displayed: "⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only." with a close button.

Buttons available include "Add security group rule", "Advanced network configuration", and "Configure storage".

At the bottom, there are links for CloudShell, Feedback, Console Mobile App, Privacy, Terms, and Cookie preferences, along with a copyright notice: "© 2025, Amazon Web Services, Inc. or its affiliates."

Step 2 — Launch Windows instance in Private Subnet

- Click Launch Instance.
- Choose:
 - Windows Server AMI
- Select:
 - VPC → MyWebAppVPC
 - Subnet → PrivateSubnet
 - Auto assign Public IP → Disable
- Security Group:
 - Create DBInstance-SG

- Allow RDP only from WebInstance-SG

- Launch instance.

The DBInstance is now **fully isolated** and cannot be accessed directly from the internet.

Name and tags [Info](#)

Name
PrivateWebS [Add additional tags](#)

Application and OS Images (Amazon Machine Image) [Info](#)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recents [Quick Start](#)

Amazon Linux macOS Ubuntu Windows Microsoft Red Hat SUSE Linux Debian [Browse more AMIs](#) Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Microsoft Windows Server 2025 Base [Free tier eligible](#)
ami-0b4bc1e90f30ca1ec (64-bit (x86))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

[CloudShell](#) [Feedback](#) [Console Mobile App](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Network settings [Info](#)

VPC - required [Info](#)
vpc-0503646573ad5bb82 (Mywebappvpc) 10.0.0.0/16

Subnet [Info](#)
subnet-0fc06cecd99cd1b5 Private Subnet
VPC: vpc-0503646573ad5bb82 Owner: 257605934444 Availability Zone: us-east-1b (use1-sz2) Zone type: Availability Zone IP addresses available: 251 CIDR: 10.0.2.0/24

Auto-assign public IP [Info](#)
Disable

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

Security group name - required
DBInstanceSG

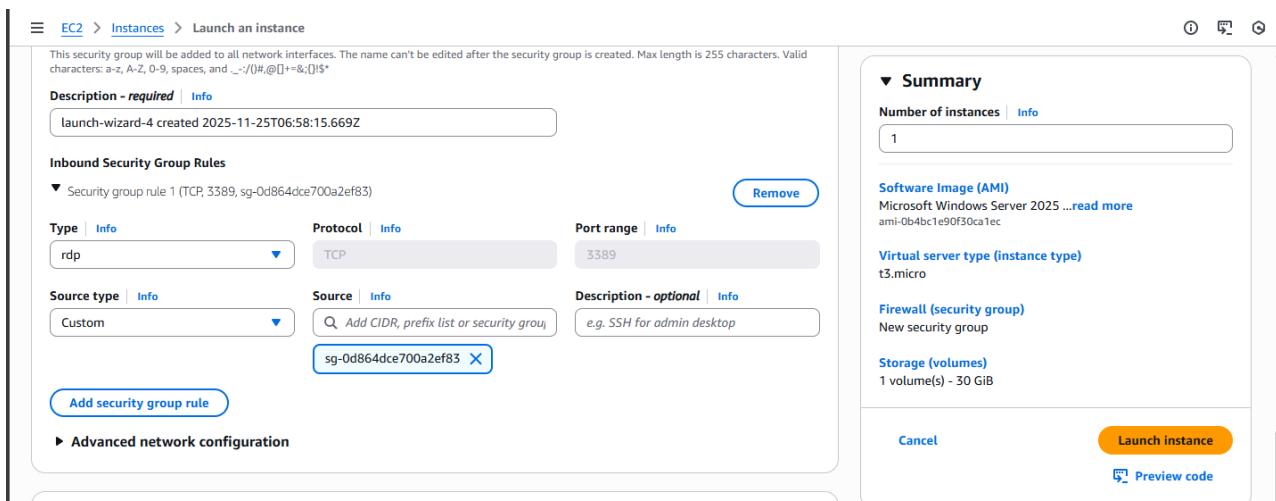
This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _-./@#=;&{}\$

Description - required [Info](#)
launch-wizard-4 created 2025-11-25T06:58:15.669Z

Inbound Security Group Rules

[CloudShell](#) [Feedback](#) [Console Mobile App](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)



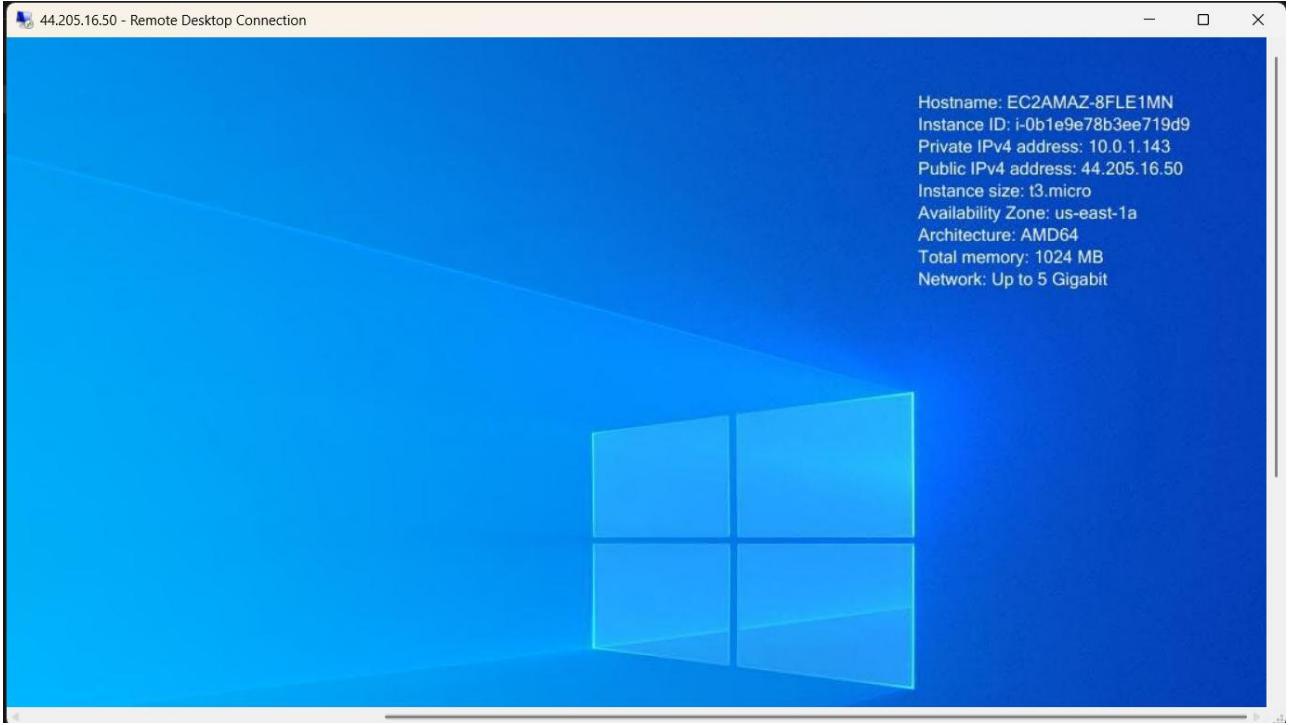
Step 3 — Connect to Public Windows Instance

1. Select WebInstance → Click Connect.
2. Choose RDP Client tab.
3. Click Get Password.
4. Upload your .pem key pair.
5. It shows the decrypted Windows Administrator password.
6. Copy the Public IP into a document for further use.

Connect using RDP

- Open Remote Desktop Connection (mstsc).
- In the dialog:
 - Computer → Public IP of WebInstance
- Click Connect.
- Credentials dialog:
- Username → Administrator
- Password → Paste decrypted password
- Click OK.

You are now inside the public Windows instance.



Step 4 — connect to private windows instance

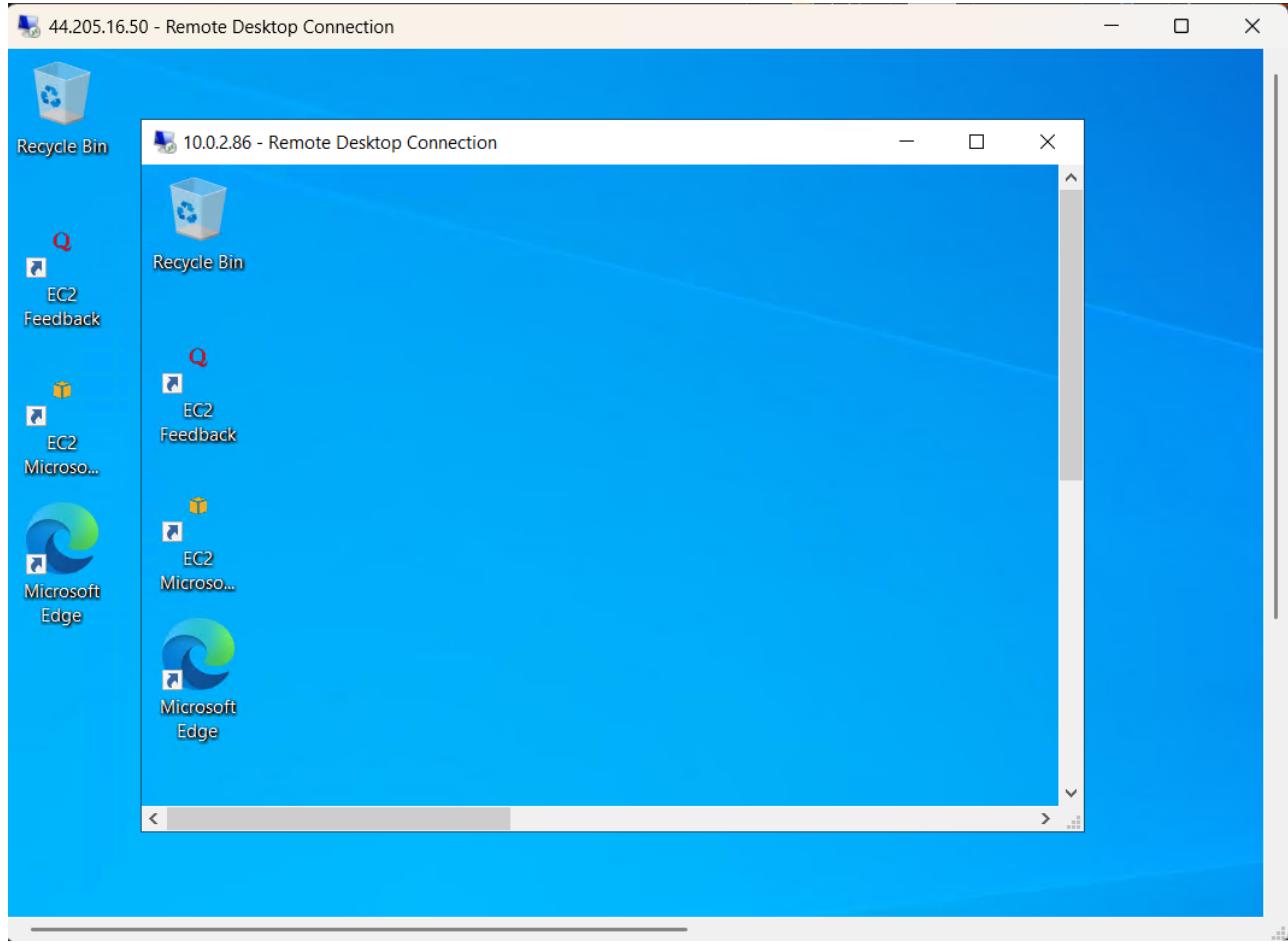
We cannot connect directly from your laptop → No Public IP.

We must connect **from within WebInstance** (jump server / bastion host).

Remote Login from WebInstance → DBInstance

- While logged into WebInstance, open Remote Desktop Connection. [**mstsc**]
- Enter:
 - Computer → Private IP of DBInstance
- Click Connect.
- Credentials:
 - Username → Administrator
 - Password → Paste the decrypted password (same key pair)
- Click OK.

You are now inside the private Windows instance.



Step 5 — testing NAT gateway connectivity (Verify Internet Access)

From WebInstance (Public Subnet)

- Open a browser → Internet should work (through IGW).

From DBInstance (Private Subnet)

- Open browser → Internet should also work (through NAT Gateway).
- Internet icon will show "Internet Access".

Security Check

- DBInstance cannot receive inbound traffic from internet.
- Only outbound is allowed via NAT (safe for DB servers).

Test: From DBInstance (Private Subnet) → ping Google (Outbound Allowed)

Steps

Connect to DBInstance (using RDP from WebInstance).

Open Command Prompt inside DBInstance.

Type: ping google.com

Expected Result

- It will successfully ping google.com
- You will see replies like:
Reply from 142.250.xxx.xxx: bytes=32 time=20ms TTL=115

Why does this work?

- Outbound traffic is allowed from Private Subnet → NAT Gateway → Internet.
- NAT Gateway acts as a proxy for the private instance.
- So the private instance can reach internet,
but internet cannot reach the private instance.

What will NOT work?

From the Private DBInstance: ping <Your Own Public IP>
or anyone trying to ping: ping <DBInstance Private IP>

Both will fail because:

- Inbound traffic is blocked
- DBInstance has no Public IP
- SG allows inbound only from WebInstance-SG

Private instance → internet = YES (via NAT Gateway)

Internet → private instance = NO (fully blocked)

Because NAT Gateway is **one-way** only.

Elastic IP address

- A static IP address is a permanent address that doesn't change. You can manually configure a device to have a static IP address.
- An Elastic IP address is static and has to be used in a specific Region, it cannot be moved to a different Region.
- An Elastic IP address comes from Amazon's pool of IPv4 addresses.
- To use an Elastic IP address, you first allocate one to your account, and then associate it with your instance or a network interface.
- When you associate an Elastic IP address with an instance or its primary network interface, if the instance already has a public IPv4 address associated with it, that public IPv4 address is released

back into Amazon's pool of public IPv4 addresses and the Elastic IP address is associated with the instance instead.

- You can disassociate an Elastic IP address from a resource, and then associate it with a different resource.
- A disassociated Elastic IP address remains allocated to your account until you explicitly release it.
- You are charged for all Elastic IP addresses in your account, regardless of whether they are associated or disassociated with an instance.
- Static IP addresses are especially important in cases where a device has to be quickly found over the internet on a permanent basis.
- Web Servers: A website must have one or more static IP addresses to be assigned to the domain always point to the correct server.

DATE: 19-11-25

Exercise–12 Deploying and Testing a Load-Balanced Web Application

By Creating a Web Server, Building a Custom AMI, Configuring a Target Group, Launching an Application Load Balancer (ALB), Registering the Instance in the Target Group, and Testing the ALB DNS

Objective

To deploy a scalable and highly available web application on AWS by configuring:

- EC2 web server
- Custom AMI
- Target Group
- Application Load Balancer (ALB)
- Testing the ALB DNS

Description for Each Component

EC2 Web Server

What: A virtual Linux machine that hosts your web application files.

Why: It serves the actual website content that users access through the ALB.

Custom AMI

What: A snapshot/template of your configured EC2 instance (with Apache + website files).

Why: Ensures every new instance launched by the Auto Scaling Group has the same setup automatically.

Target Group

What: A collection of EC2 instances that the ALB sends traffic to.

Why: It allows the load balancer to forward requests only to healthy instances in the Auto Scaling Group.

Application Load Balancer (ALB)

What: A managed service that distributes incoming HTTP traffic across multiple EC2 instances.

Why: Ensures high availability and balanced traffic distribution for the web application.

Security Group

What: A virtual firewall controlling inbound/outbound traffic to EC2 instances and ALB.

Why: Ensures only required traffic (HTTP/SSH) is allowed for the application components.

VPC + Subnets

What: A private network environment where all resources run.

Why: Provides isolation, routing, and a structured network setup for public & private components.

Launch Template

What: A reusable blueprint containing AMI, instance type, key pair, and security group settings.

Why: The ASG uses this template to launch identical EC2 instances.

STEP 1 — Launch Base EC2 Instance

- Open AWS Console → EC2 → Launch Instance
- Name: Base-WebServer
- AMI: Amazon Linux 2
- Instance type: t3.micro
- Key Pair: your .pem file
- Security Group:
 - HTTP (80) → Anywhere
 - SSH (22) → My IP
- Launch the instance

STEP 2 — Install Apache and Create Web Page

- SSH into the instance.
- Install Apache:
 - sudo yum install httpd -y
 - sudo systemctl start httpd
 - sudo systemctl enable httpd
- Move to Web Root Folder - cd /var/www/html
- Create a webpage with hostname:
 - echo "<h1>Hello from Instance 1 — \$(hostname)</h1>" | sudo tee /var/www/html/index.html

Test in browser using the instance's public IP.

EC2 Instances

Instances (1/2) Info

Find Instance by attribute or tag (case-sensitive)

Instance state = running | Clear filters | All states ▾

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone |
|----------------|---------------------|----------------|---------------|-------------------|---------------|-------------------|
| Base_web | i-0c27452d42aa52de6 | Running | t3.micro | 3/3 checks passed | View alarms + | us-east-1f |
| Base_webserver | i-0430be589fd04e9f4 | Running | t3.micro | 3/3 checks passed | View alarms + | us-east-1f |

i-0c27452d42aa52de6 (Base_web)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary

| | | |
|--|---|---|
| Instance ID i-0c27452d42aa52de6 | Public IPv4 address 3.238.24.214 open address ↗ | Private IPv4 addresses 172.31.71.20 |
| IPv6 address - | Instance state Running | Public DNS ec2-3-238-24-214.compute-1.amazonaws.com open address ↗ |
| Hostname type IP name: ip-172-31-71-20.ec2.internal | Private IP DNS name (IPv4 only) ip-172-31-71-20.ec2.internal | Elastic IP addresses |
| Amazon private route 56 DNS name | Instance type | |

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

```

Installed:
apr-1.7.5-1.amzn2023.0.4.x86_64
apr-util-openssl-1.6.3-1.amzn2023.0.2.x86_64
httpd-core-2.4.65-1.amzn2023.0.2.x86_64
libbrotli-1.0.9-4.amzn2023.0.2.x86_64
mod_lua-2.4.65-1.amzn2023.0.2.x86_64

apr-util-1.6.3-1.amzn2023.0.2.x86_64
generic-logos-https-18.0.0-12.amzn2023.0.3.noarch
httpd-filesystem-2.4.65-1.amzn2023.0.2.noarch
mailcap-2.1.49-3.amzn2023.0.3.noarch

apr-util-lmdb-1.6.3-1.amzn2023.0.2.x86_64
httpd-2.4.65-1.amzn2023.0.2.x86_64
httpd-tools-2.4.65-1.amzn2023.0.2.x86_64
mod_http2-2.0.27-1.amzn2023.0.3.x86_64

Complete!
[ec2-user@ip-172-31-71-20 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-71-20 ~]$ sudo systemctl enable httpd
-bash: o: command not found
[ec2-user@ip-172-31-71-20 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-71-20 ~]$ cd /var/www/html
[ec2-user@ip-172-31-71-20 html]$ echo "<h1>Hello from Instance 1 - $(hostname)</h1>" | sudo tee /var/www/html/index.html
<h1>Hello from Instance 1 - ip-172-31-71-20.ec2.internal</h1>
[ec2-user@ip-172-31-71-20 html]$ 

```

Not secure 3.238.24.214 InPrivate

Hello from Instance 1 — ip-172-31-71-20.ec2.internal

STEP 3 — Create Custom AMI

- Go to EC2 → Instances
- Select Base-WebServer
- Actions → Image and Templates → Create Image
- Name: WebServer-AMI
- Create
- Wait until AMI status = Available

This AMI now contains Apache + your index.html.

EC2 Instances

Instances (1/1) info

Find Instance by attribute or tag (case-sensitive)

Instance state = running

Clear filters

Name: Base_web | Instance ID: i-0c27452d42aa52de6 | Instance state: Running | Instance type: t2.micro | Status check: Create image

Actions: Connect, Instance state, Actions, Launch instances

Launch diagnostics, Instance settings, Networking, Security, Zone, Image and templates, Storage, Monitor and troubleshoot

i-0c27452d42aa52de6 (Base_web)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary

Instance ID: i-0c27452d42aa52de6 | Public IPv4 address: | Private IPv4 addresses:

CloudShell Feedback Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Create image

WebServer-AMI

Maximum 127 characters. Can't be modified after creation.

Image description - optional

Image description

Maximum 255 characters.

Reboot instance

When selected, Amazon EC2 reboots the instance so that data is at rest when snapshots of the attached volumes are taken. This ensures data consistency.

Instance volumes

| Storage type | Device | Snapshot | Size | Volume type | IOPS | Throughput | Delete on termination | Encrypted |
|--------------|-----------|-------------------------------|------|-------------------------------|------|------------|-----------------------|-----------|
| EBS | /dev/x... | Create new snapshot from v... | 8 | EBS General Purpose SSD - ... | 3000 | Enable | Enable | |

Add volume

During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.

Tags - optional

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.

Tag image and snapshots together

Tag the image and the snapshots with the same tag.

Tag image and snapshots separately

Tag the image and the snapshots with different tags.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel Create image

CloudShell Feedback Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 AMIs

Amazon Machine Images (AMIs) (1) Info

Owned by me

Find AMI by attribute or tag

Recycle Bin, EC2 Image Builder, Actions, Launch instance from AMI

AMIs (1)

| AMID | Source | Owner | Visibility | Status | Created |
|-----------------------|----------------------------|--------------|------------|-----------|---------|
| ami-0920f39e3335fd68e | 257605934444/WebServer-AMI | 257605934444 | Private | Available | 2025/ |

STEP 4 — Create Target Group

1. EC2 → Target Groups → Create Target Group
2. Target type: Instances
3. Name: WebApp-TG
4. Protocol: HTTP
5. Port: 80
6. Health Check Path: /
7. Create (do not register instances manually)

Purpose: Target Group holds the list of EC2 instances behind the Load Balancer.

The screenshot shows the AWS EC2 Target Groups page. The left sidebar navigation includes: AMIs, AMI Catalog, Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), Load Balancing (Load Balancers, Target Groups, Trust Stores), Auto Scaling (Auto Scaling Groups, Settings). The main content area is titled "Target groups" and displays a message: "No target groups" and "You don't have any target groups in us-east-1". It features a "Create target group" button. The bottom of the page includes links for CloudShell, Feedback, and Console Mobile App, along with copyright information: © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

EC2 > Target groups > Create target group
Print | Refresh | Help

Indicate what resource type you want to target. Only the selected resource type can be registered to this target group.

Instances
 Supports load balancing to instances in a VPC. Integrate with Auto Scaling Groups or ECS services for automatic management.
 Suitable for: ALB NLB GWLB

IP addresses
 Supports load balancing to VPC and on-premises resources. Facilitates routing to IP addresses and network interfaces on the same instance. Supports IPv6 targets.
 Suitable for: ALB NLB GWLB

Lambda function
 Supports load balancing to a single Lambda function. ALB required as traffic source.
 Suitable for: ALB

Application Load Balancer
 Allows use of static IP addresses and PrivateLink with an Application Load Balancer. NLB required as traffic source.
 Suitable for: NLB

Target group name
Name must be unique per Region per AWS account.

WebApp-tg

Accepts: a-z, A-Z, 0-9, and hyphen (-). Can't begin or end with hyphen. 1-32 total characters; Count: 9/32

Protocol
Protocol for communication between the load balancer and targets.

Port
Port number where targets receive traffic. Can be overridden for individual targets during registration.

HTTP

80

1-65535

IP address type
Only targets with the indicated IP address type can be registered to this target group.

IPv4
Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.

CloudShell Feedback Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy Terms Cookie preferences

vpc-0e06f443318ef0d95 (default) 172.31.0.0/16 Create VPC

Protocol version

HTTP1
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.

Health checks
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol

HTTP

Health check path
Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.

/

Up to 1024 characters allowed.

▶ **Advanced health check settings**

CloudShell Feedback Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy Terms Cookie preferences

EC2 > Target groups > Create target group

| | | | | | | | |
|------|-----------------------|-----------------|----------|-----------------|----------|------------------|-------|
| Name | WebApp-tg | Target type | Instance | Protocol : Port | HTTP: 80 | Protocol version | HTTP1 |
| VPC | vpc-0e06f443318ef0d95 | IP address type | IPv4 | | | | |

Health check details

| | | | | | | | |
|-----------------------|-----------|-------------------|---|---------------------|--------------|---------------|------------|
| Health check protocol | HTTP | Health check path | / | Health check port | traffic-port | Interval | 30 seconds |
| Timeout | 5 seconds | Healthy threshold | 5 | Unhealthy threshold | 2 | Success codes | 200 |

Step 2: Register targets

Targets (0)

| Instance ID | Name | Port | Zone |
|------------------|------|------|------|
| No targets added | | | |

Buttons: Cancel, Previous, Create target group

STEP 5 — Create Application Load Balancer (ALB)

1. EC2 → Load Balancers → Create Load Balancer
2. Choose Application Load Balancer
3. Name: WebApp-ALB
4. Scheme: Internet-facing
5. Listeners: HTTP (80)
6. Select two public subnets
7. Security Group: allow HTTP (port 80)
8. Forward to Target Group → WebApp-TG
9. Create

Copy the ALB DNS name for testing later.

EC2 > Load balancers

Load balancers What's new?

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Actions **Create load balancer**

Filter load balancers

| Name | State | Type | Scheme | IP address type | VPC ID | Ava |
|---------------------------|-------|------|--------|-----------------|--------|-----|
| 0 load balancers selected | | | | | | |

Select a load balancer above.

EC2 > Load balancers > Create Application Load Balancer

▶ How Application Load Balancers work

Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme | **Info**
Scheme can't be changed after the load balancer is created.

Internet-facing
• Serves internet-facing traffic.
• Has public IP addresses.
• DNS name resolves to public IPs.
• Requires a public subnet.

Internal
• Serves internal traffic.
• Has private IP addresses.
• DNS name resolves to private IPs.
• Compatible with the IPv4 and Dualstack IP address types.

Load balancer IP address type | **Info**
Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.

IPv4
Includes only IPv4 addresses.

Dualstack
Includes IPv4 and IPv6 addresses.

Dualstack without public IPv4
Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with internet-facing load balancers only.

EC2 > Load balancers > Create Application Load Balancer

Network mapping

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC | **Info**
The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC for your targets, view target groups.

vpc-0e6f443318ef0d95
172.31.0.0/16 (default) Create VPC

IP pools | **Info**
You can optionally choose to configure an IPAM pool as the preferred source for your load balancer's IP addresses. Create or view Pools in the Amazon VPC IP Address Manager console.

Use IPAM pool for public IPv4 addresses
The IPAM pool you choose will be the preferred source of public IPv4 addresses. If the pool is depleted, IPv4 addresses will be assigned by AWS.

Availability Zones and subnets | **Info**
Select at least two Availability Zones and a subnet for each zone. A load balancer node will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.

us-east-1a (use1-az1)
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-05d7f1b9eaaf25b1d
IPv4 subnet CIDR: 172.31.0.0/20

us-east-1b (use1-az2)
Subnet
Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.
subnet-0b614af667cff5875
IPv4 subnet CIDR: 172.31.80.0/20

us-east-1c (use1-az4)

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the 'Listeners and routing' section of the ALB configuration. A single listener is defined for port 80 (HTTP). The 'Default action' is set to 'Forward to target groups', which is selected. A target group named 'Webapp-tg' is associated with this listener, with a weight of 1 and a percent of 100%. There is also an option to 'Add target group'. The bottom of the screen shows standard AWS navigation and footer links.

STEP 6 — Test the Load Balancer Using ALB DNS

- Go to AWS Console → EC2 → Load Balancers
 - Select your ALB: ALB-WebServer
- Copy the ALB DNS Name
 - You will see something like:
 - WebApp-ALB-123456789.ap-south-1.elb.amazonaws.com
- Open a Browser and Paste the DNS Name
 - Enter: http://<your-alb-dns-name>
 - Example: http://WebApp-ALB-123456789.ap-south-1.elb.amazonaws.com
- View the Output
 - You should see the webpage you created earlier:
 - Hello from Instance 1 — ip-xx-xx-xx-xx
 - This confirms that: The ALB is working
 - The Target Group is routing traffic
 - The instance created from your custom AMI is serving the web page
- Optional – Refresh Multiple Times
 - If you later use Auto Scaling with multiple instances:
 - Clicking Refresh will show different hostnames
 - This confirms load balancing across multiple servers



Hello from Instance 1 — ip-172-31-69-236.ec2.internal

DATE: 20-11-25

Exercise–13 Stress Testing a Linux EC2 Instance (CPU Load Test)

Objective:

To generate high CPU load on an Amazon EC2 Linux instance using the stress-ng tool and observe CPU utilization in CloudWatch.

Step 1 — Launch a Linux EC2 Instance

1. Log in to the AWS Console.
2. Go to **EC2 → Instances → Launch Instance**.
3. Name: **StressTest-EC2**
4. AMI: **Amazon Linux 2023 (Free Tier eligible)**
5. Instance type: **t2.micro / t3.micro**
6. Key pair: Select or create a .pem key.
7. Security Group:
 - Allow **SSH (22)** from My IP.
 - Allow **HTTP (80)** from anywhere (optional).
8. Launch the instance.

Step 2 — Connect to the EC2 Instance

Use Windows PowerShell

Navigate to the folder where pem file is located and type the following command:

```
ssh -i "your-key.pem" ec2-user@<public-ip>
```

Step 3 — Install Apache (Optional, to verify the instance is working)

```
sudo yum install httpd -y
```

```
sudo systemctl start httpd
```

```
sudo systemctl enable httpd
```

Create a test web page:

```
echo "<h1>EC2 Stress Test Demo — $(hostname)</h1>" | sudo tee /var/www/html/index.html
```

STEP 4 — Install and Run the Stress Testing Tool (stress-ng)

(For Amazon Linux 2023 EC2 Instances)

Move to the ec2-user home directory

(It is recommended to run stress-ng from the home folder)

```
cd ~
```

Install stress-ng

Amazon Linux 2023 includes stress-ng directly through dnf, so install it using:

```
sudo dnf install stress-ng -y
```

Run a CPU Stress Test

Generate high CPU load using 4 CPU workers for 2 minutes:

```
stress-ng --cpu 4 --timeout 120
```

Expected Result

- Terminal becomes busy during the 120-second load test
- After completion, you will see:

successful run completed in 120.02s

- CPU usage will spike to 90–100% (you can check using top)

Step 5 — Run Stress Test (CPU Load Generation)

Move to home directory to avoid permission issues:

```
cd ~
```

Run CPU stress for 2 minutes using 4 CPU workers:

```
stress-ng --cpu 4 --timeout 120
```

You should see messages like:

```
stress-ng: info: dispatching hogs: 4 cpu
```

```
stress-ng: info: successful run completed in 120.02s
```

This will increase CPU usage to ~100% for the duration.

Step 6 — Observe CPU Utilization in CloudWatch

Go to CloudWatch → Metrics → EC2 → Per-Instance Metrics

Select:

CPUUtilization → InstanceId → your instance

View graph in 1-minute or 5-minute intervals.

You should see a sharp spike during the 2-minute stress period.

Step 7 — Stop or Terminate the Instance

To avoid charges:

Option A — Stop the instance (safe):

Actions → Instance State → Stop

Option B — Terminate (permanently delete):

Actions → Instance State → Terminate

Expected Output

- CPU Utilization in CloudWatch should reach 90–100%.
- Stress test completes without errors.
- Students understand how CPU load affects EC2 metrics.

STEP 6 — Create Auto Scaling Group (ASG)

6A. Create Launch Template

1. EC2 → Launch Templates → Create
2. Name: WebServer-LT
3. AMI: WebServer-AMI
4. Instance type: t3.micro
5. Security Group: allow HTTP + SSH
6. Create

The screenshot shows the 'Create launch template' wizard on the AWS EC2 console. The current step is 'Launch template name and description'. The 'Software Image (AMI)' is set to 'WebServer-AMI' (ami-0920f39e3335fd68e). The 'Virtual server type (instance type)' is 't3.micro'. The 'Firewall (security group)' is 'launch-wizard-3'. The 'Storage (volumes)' section shows 1 volume(s) - 8 GiB. The 'Summary' panel on the right lists these configurations. At the bottom right of the main form is a 'Create launch template' button.

Create launch template

Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.

Launch template name and description

Launch template name - required

WebServer-LT

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.

Template version description

A prod webserver for MyApp

Max 255 chars

Auto Scaling guidance | Info

Select this if you intend to use this template with EC2 Auto Scaling

Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

► Template tags

► Source template

Summary

Software Image (AMI)
WebServer-AMI
ami-0920f39e3335fd68e

Virtual server type (instance type)
t3.micro

Firewall (security group)
launch-wizard-3

Storage (volumes)
1 volume(s) - 8 GiB

Create launch template

The screenshot shows the 'Create launch template' wizard on the AWS EC2 console. The current step is 'Instance type'. The 'Software Image (AMI)' is set to 'WebServer-AMI' (ami-0920f39e3335fd68e). The 'Virtual server type (instance type)' is 't3.micro'. The 'Firewall (security group)' is 'launch-wizard-3'. The 'Storage (volumes)' section shows 1 volume(s) - 8 GiB. The 'Summary' panel on the right lists these configurations. At the bottom right of the main form is a 'Create launch template' button.

Instance type Info | Get advice

Instance type

t3.micro

Family: t3 2 vCPU 1 GiB Memory Current generation: true

Free tier eligible

On-Demand Ubuntu Pro base pricing: 0.0139 USD per Hour

On-Demand SUSE base pricing: 0.0104 USD per Hour

On-Demand Linux base pricing: 0.0104 USD per Hour

On-Demand RHEL base pricing: 0.0392 USD per Hour

On-Demand Windows base pricing: 0.0196 USD per Hour

Advanced

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

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

Don't include in launch template

Create new key pair

Network settings Info

Subnet | Info

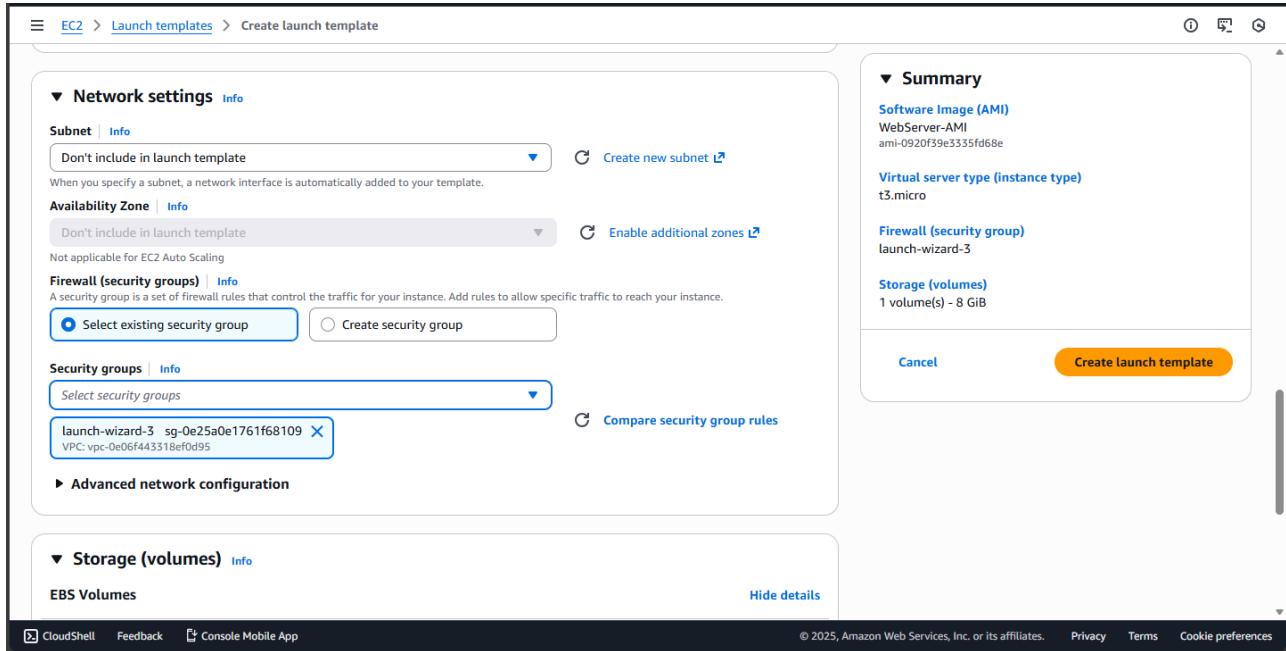
Don't include in launch template

Create new subnet

Create launch template

CloudShell Feedback Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



6B. Create Auto Scaling Group

1. Go to Auto Scaling → Create Auto Scaling Group
2. Name: WebApp-ASG
3. Select Launch Template: WebServer-LT
4. Choose VPC + two public subnets
5. Load Balancing:
 - o Attach to existing ALB
 - o Choose WebApp-TG
6. Group size:
 - o Desired: 1
 - o Min: 1
 - o Max: 3
7. Create

EC2 > Auto Scaling groups > Create Auto Scaling group

Choose instance launch options

Step 3 - optional

Integrate with other services

Step 4 - optional

Configure group size and scaling

Step 5 - optional

Add notifications

Step 6 - optional

Add tags

Step 7

Review

Name

Auto Scaling group name

Enter a name to identify the group.

Must be unique to this account in the current Region and no more than 255 characters.

Launch template Info

For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

Launch template

Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

[Create a launch template](#)

Version

[Create a launch template version](#)

| | | |
|-----------------------|--|-------------------------------|
| Description | Launch template | Instance type |
| - | WebServer-LT lt-0c7399cd892f63ddc | t3.micro |
| AMI ID | Security groups | Request Spot Instances |
| ami-0920f59e3335fd68e | - | No |

CloudShell Feedback Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 6 - optional

Add tags

Step 7

Review

instance type

t3.micro

Network Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC

Choose the VPC that defines the virtual network for your Auto Scaling group.

172.31.0.0/16 Default

[Create a VPC](#)

Availability Zones and subnets

Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

172.31.0.0/20 Default

172.31.80.0/20 Default

[Create a subnet](#)

Availability Zone distribution - new

Auto Scaling automatically balances instances across Availability Zones. If launch failures occur in a zone, select a strategy.

Balanced best effort

If launches fail in one Availability Zone, Auto Scaling will attempt to launch in another healthy Availability Zone.

Balanced only

If launches fail in one Availability Zone, Auto Scaling will continue to attempt to launch in the unhealthy Availability Zone to preserve balanced distribution.

CloudShell Feedback Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

[EC2](#) > [Auto Scaling groups](#) > Create Auto Scaling group

Step 1
 Choose launch template
 Choose instance launch options
 Integrate with other services
 Step 4 - optional
 Configure group size and scaling
 Step 5 - optional
 Add notifications
 Step 6 - optional
 Add tags
 Step 7
 Review

Integrate with other services - optional Info

Use a load balancer to distribute network traffic across multiple servers. Enable service-to-service communications with VPC Lattice. Shift resources away from impaired Availability Zones with zonal shift. You can also customize health check replacements and monitoring.

Load balancing Info

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

Select Load balancing options

No load balancer
Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer
Choose from your existing load balancers.

Attach to a new load balancer
Quickly create a basic load balancer to attach to your Auto Scaling group.

Attach to an existing load balancer

Select the load balancers to attach

Choose from your load balancer target groups
This option allows you to attach Application, Network, or Gateway Load Balancers.

Choose from Classic Load Balancers

Existing load balancer target groups
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups ▾

Webapp-tg | HTTP Application Load Balancer: WebApp-ALB

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

[EC2](#) > [Auto Scaling groups](#) > Create Auto Scaling group

Step 4 - optional
 Configure group size and scaling
 Step 5 - optional
 Add notifications
 Step 6 - optional
 Add tags
 Step 7
 Review

Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Desired capacity
Specify your group size.

Scaling Info

You can resize your Auto Scaling group manually or automatically to meet changes in demand.

Scaling limits
Set limits on how much your desired capacity can be increased or decreased.

| | |
|--|---|
| Min desired capacity <input type="text" value="1"/> Equal or less than desired capacity | Max desired capacity <input type="text" value="3"/> Equal or greater than desired capacity |
|--|---|

Automatic scaling - optional

Choose whether to use a target tracking policy Info

You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

No scaling policies
Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

Target tracking scaling policy
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

Instance maintenance policy Info

Control your Auto Scaling group's availability during instance replacement events. This includes health checks, instance refreshes, maximum instance lifetime

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

STEP 7 — Configure Auto Scaling Policies

Go to ASG → Automatic Scaling → Add Policy

Scale-Out Policy

- Metric: CPU Utilization
- Rule: > 60% for 2 minutes
- Action: Add +1 instance

Scale-In Policy

- Rule: < 20% for 5 minutes
- Action: Remove 1 instance

STEP 8 — Stress Test to Trigger Scaling

SSH into your running instance:

Install stress:

```
sudo amazon-linux-extras install epel -y
```

```
sudo yum install stress -y
```

Run CPU load:

```
stress --cpu 90 --timeout 300
```

Within 2–3 minutes:

- CPU becomes high
- ASG launches second instance
- Target Group shows both instances as healthy

STEP 9 — Test Load Balancing

Open ALB DNS Name in browser: <http://<alb-dns-name>>

Refresh many times — you will see:

Hello from Instance 1 — ip-XX-XX...

Hello from Instance 2 — ip-YY-YY...

- Confirms ALB working
- Confirms auto scaling working

After stress test ends → ASG scales back to 1 instance.

Extra points:

This exercise imitates a production-grade architecture used in companies.

What is a Target Group?

A Target Group is a collection of EC2 instances that receive traffic from the Application Load Balancer (ALB).

It defines:

- Which instances to send traffic to
- On which port (example: 80)
- Health check rules (path /, interval, threshold)

In simple words:

Target Group = list of servers behind the load balancer.

Why do we need two steps: Create Launch Template and Create Auto Scaling Group?

Because both have different roles:

Launch Template = WHAT to launch

It contains the configuration of one EC2 instance, such as:

- AMI
- Instance type
- Security Group
- Key pair
- Storage
- User data

This is just a blueprint.

Auto Scaling Group = WHEN & HOW MANY to launch

The ASG uses the Launch Template to automatically:

- Launch new instances
- Remove instances
- Maintain desired capacity
- Scale based on CPU/memory demand

Where to find `http://<alb-dns-name>`?

You can find the ALB DNS Name in the AWS Console:

Path:

EC2 → Load Balancers → Select your ALB → Description tab

There you will see:

DNS name: mywebapp-alb-12345678.ap-south-1.elb.amazonaws.com

Use it in your browser as:

`http://mywebapp-alb-12345678.ap-south-1.elb.amazonaws.com`

This is the public URL of your Load Balancer.

What is ALB DNS?

ALB DNS is the publicly accessible DNS name automatically created by AWS for your Application Load Balancer.

Example: myapp-alb-123456789.ap-south-1.elb.amazonaws.com

When a user types this URL:

- Traffic goes to the ALB
- ALB forwards to Target Group
- Target Group sends request to one of the EC2 instances

ALB DNS = The website URL of your Load Balancer.

DNS means Domain Name System.

DNS is a system that converts domain names to IP addresses.

Example: `www.amazon.com` → 52.95.120.1

A DNS Server is only one part of this system.

DNS = System that translates names to IPs.

DNS Server = Machine that performs the translation

DATE: 21-11-25

Exercise–14 Mini Project –

Deploying a Load-Balanced Web Application using Application Load Balancer (ALB), Auto Scaling Group (ASG), Custom AMI, and Target Group on AWS with CloudWatch Alarms & Stress Testing for Auto Scaling Validation.

Auto Scaling Group (ASG)

What: A service that automatically launches or terminates EC2 instances based on demand.

Why: Provides scalability so your application can handle variable traffic.

CPU-Based Scaling Policies

What: Rules that add or remove EC2 instances based on CPU usage thresholds.

Why: Ensures your application automatically scales up during heavy load and scales down to save cost.

Stress Testing (using stress/stress-ng tool)

What: A method to artificially increase CPU load on instances.

Why: Used to validate whether the Auto Scaling Group is scaling up/down correctly.

STEP 7A — Create Launch Template

Go to EC2 → Launch Templates → Create launch template

Template Name

- Launch template name: LT- WebServer

Choose AMI (Custom AMI)

Under Application and OS Images (AMI):

- Click My AMIs
- Select your custom AMI: AMI-WebServer (the AMI you created earlier)

Instance Type

Under Instance type: Select t3.micro

Key Pair

Under Key pair (login): Select your existing key pair (pemfile1.pem)

Security Group

Under Network settings → Firewall (security groups):

- Choose Select existing security group
- Select a security group that allows:
 - SSH (Port 22)

- HTTP (Port 80)

Example: launch-wizard-1 (already has both rules)

Storage

Leave default: 8 GiB gp3 (root volume)

Create Template

Click Create launch template

Launch Template is now ready to be used by the Auto Scaling Group.

Create launch template

Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.

Launch template name and description

Launch template name - *required*

LT-WebServer

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.

Template version description

A prod webserver for MyApp

Max 255 chars

Auto Scaling guidance | Info

Select this if you intend to use this template with EC2 Auto Scaling

Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

▶ Template tags

▶ Source template

Launch template contents

Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

▼ Application and OS Images (Amazon Machine Image) Info

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Summary

Software Image (AMI)
WebServer-AMI
ami-04b340607c6a88c7e

Virtual server type (instance type)
t3.micro

Firewall (security group)
-

Storage (volumes)
1 volume(s) - 8 GiB

[Cancel](#) [Create launch template](#)

Application and OS Images (Amazon Machine Image) Info

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 > Launch templates > Create launch template

▼ Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recents [My AMIs](#) Quick Start

Don't include in launch template Owned by me Shared with me

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

WebServer-AMI
ami-04b340607c6a88c7e
2025-11-28T04:18:04.000Z Virtualization: hvm ENA enabled: true Root device type: ebs Boot mode: uefi-preferred

Description

-

| Architecture | AMI ID |
|--------------|-----------------------|
| x86_64 | ami-04b340607c6a88c7e |

[CloudShell](#) [Feedback](#) [Console Mobile App](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

The screenshot shows the AWS EC2 'Create launch template' wizard. The current step is 'Network settings'. The configuration includes:

- Subnet:** Set to 'Don't include in launch template'.
- Availability Zone:** Set to 'Don't include in launch template'.
- Firewall (security groups):** A security group named 'launch-wizard-1' (sg-001d4e50a4144733c) is selected. It is associated with a VPC (vpc-0e06f443318ef0d95).
- Advanced network configuration:** An option to enable additional zones is present.

STEP 7B — Create Auto Scaling Group (ASG)

(Using the Launch Template you created in Step 7A)

Open ASG Wizard

Go to EC2 → Auto Scaling Groups → Create Auto Scaling Group Name and Choose Launch Template

Auto Scaling group name: ASG-WebServer

Launch template: Choose LT-WebServer

Click Next.

Select Network (VPC + Subnets)

Since we are using default VPC, select:

VPC: `vpc-0c952a6cabfbe594b (Default VPC)`

Subnets (select ANY 2)

Example: `ap-south-1a` and `ap-south-1b`

(These are public subnets in default VPC — good for web servers)

Click Next.

Attach Load Balancer

Under Load balancing options:

Select: Attach to an existing load balancer

Under Select load balancers to attach:

Choose: Choose from your load balancer target groups

Under dropdown:

Select your Target Group: `TG-WebServer` (or whatever name you created)

AWS will automatically show:

- Load balancer: `ALB-WebServer`
- Type: Application/HTTP

Click Next.

Configure Group Size

Set:

Desired capacity = 1

Minimum capacity = 1

Maximum capacity = 3

We are not adding scaling policies now (stress test comes later), so:

Select No scaling policies

Click Next.

Skip Notifications and Tags

No changes.

Click Next.

Review and Create

Check:

- Launch template = `LT-WebServer`
- Load balancer = `ALB-WebServer`
- Target group = `TG-WebServer`
- Subnets = 2 selected
- Desired = 1 instance

Then click: Create Auto Scaling Group

The screenshot shows the 'Create Auto Scaling group' wizard on the AWS Management Console. The current step is 'Step 1: Choose launch template'. On the left, a sidebar lists optional steps: Step 2 (Choose instance launch options), Step 3 - optional (Integrate with other services), Step 4 - optional (Configure group size and scaling), Step 5 - optional (Add notifications), Step 6 - optional (Add tags), Step 7, and Review. The main area is titled 'Choose launch template' with a 'Name' input field containing 'ASG-WebServer'.

Choose launch template Info
Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

Name
Auto Scaling group name
Enter a name to identify the group.

Must be unique to this account in the current Region and no more than 255 characters.

Launch template Info
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

Launch template
LT-WebServer
[Create a launch template](#) L C

Version
Default (1) L C
[Create a launch template version](#) L

Description
Launch template [LT-WebServer](#) L C
Instance type t3.micro

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the 'Create Auto Scaling group' wizard on the AWS Management Console. The current step is 'Step 4 - optional: Configure group size and scaling'. The sidebar shows steps 5 through 7 as completed. The main area displays configuration details: Launch template 'LT-WebServer' (version Default), Instance type 't3.micro', and a large empty 'Description' field.

Step 4 - optional
Configure group size and scaling
Add notifications
Add tags
Review

Launch template LT-WebServer L lt-02ccb953779fb9a0e
Instance type t3.micro

Description

Network Info
For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC
Choose the VPC that defines the virtual network for your Auto Scaling group.
 172.31.0.0/16 Default
[Create a VPC](#) L C

Availability Zones and subnets
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.
 L C

use1-az2 (us-east-1b) | subnet-0b614af667cff5875 X
172.31.80.0/20 Default

use1-az1 (us-east-1a) | subnet-05d7f1b9eaaf25b1d X
172.31.0.0/20 Default

[Create a subnet](#) L

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

[EC2](#) > [Auto Scaling groups](#) > Create Auto Scaling group

Step 1 Choose launch template
 Step 2 Choose instance launch options
 Step 3 - optional
Step 4 - optional
 Integrate with other services
 Step 5 - optional
 Configure group size and scaling
 Step 6 - optional
 Add notifications
 Step 7 Add tags
 Review

Integrate with other services - optional Info

Use a load balancer to distribute network traffic across multiple servers. Enable service-to-service communications with VPC Lattice. Shift resources away from impaired Availability Zones with zonal shift. You can also customize health check replacements and monitoring.

Load balancing Info

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

Select Load balancing options

No load balancer
 Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer
 Choose from your existing load balancers.

Attach to a new load balancer
 Quickly create a basic load balancer to attach to your Auto Scaling group.

Attach to an existing load balancer

Select the load balancers to attach

Choose from your load balancer target groups
 This option allows you to attach Application, Network, or Gateway Load Balancers.

Choose from Classic Load Balancers

Existing load balancer target groups

Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups ▾ 

WebApp-TG | HTTP 
 Application Load Balancer: WebApp-ALB

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

[CloudShell](#) [Feedback](#) [Console Mobile App](#)

[EC2](#) > [Auto Scaling groups](#) > Create Auto Scaling group

Choose instance launch options
 Step 3 - optional
 Integrate with other services
Step 4 - optional
 Configure group size and scaling
 Step 5 - optional
 Add notifications
 Step 6 - optional
 Add tags
 Step 7 Review

Group size Info

Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

Desired capacity type

Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Units (number of instances) ▾

Desired capacity

Specify your group size.
 1

Scaling Info

You can resize your Auto Scaling group manually or automatically to meet changes in demand.

Scaling limits

Set limits on how much your desired capacity can be increased or decreased.

| | |
|-------------------------------------|--|
| Min desired capacity 1 | Max desired capacity 3 |
| Equal or less than desired capacity | Equal or greater than desired capacity |

Automatic scaling - optional

Choose whether to use a target tracking policy Info

You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

No scaling policies
 Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

Target tracking scaling policy
 Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

[CloudShell](#) [Feedback](#) [Console Mobile App](#)

The screenshot shows the AWS Auto Scaling Groups creation wizard. It has two main sections:

- Step 5: Add notifications**: A "Notifications" section with a note "No notifications". There is an "Edit" button.
- Step 6: Add tags**: A "Tags (0)" section with a table for adding tags. It includes columns for "Key" and "Value", and a "Tag new instances" checkbox. Below it is a note "No tags". There is also an "Edit" button.

At the bottom, there are buttons for "Preview code", "Cancel", "Previous", and a prominent orange "Create Auto Scaling group" button.

ASG Creation Done

Check if ASG launched an instance.

Verify ASG Instance Creation

Go to EC2 → Auto Scaling Groups → ASG-WebServer

Open it and check: Under Activity tab

You should see: Successful — Launching a new EC2 instance

Under Instances tab: You should see an instance like: i-xxxxxxxxxxxx (Running)

The screenshot shows the "Activity" tab for the ASG-WebServer group. On the left is a navigation sidebar with categories like AMIS, AMI Catalog, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling (with "Auto Scaling Groups" selected). The main area has tabs for Integrations, Automatic scaling, Instance management, Instance refresh, Activity (which is selected), Monitoring, and Tags - moved. The "Activity notifications (0)" section shows a note "No notifications are currently specified" and a "Create notification" button. The "Activity history (1)" section shows a single entry:

| Status | Description | Cause | Start time |
|------------|---|---|-----------------|
| Successful | Launching a new EC2 instance: i-051beb0c60c18390f | At 2025-11-28T04:36:10Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 1. At 2025-11-28T04:36:13Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1. | 2025 Nov 28, AM |

This instance was launched automatically by the ASG using custom AMI through the Launch Template.

Now that ASG is created and one instance is running, we will add Auto Scaling Policies so the group can:

- Scale OUT (add more instances when CPU is high)
- Scale IN (remove instances when CPU is low)

STEP 8 — Configure Auto Scaling Policies (Scale Out & Scale In)

PART 1 – Create High-CPU Alarm (for Scale-OUT)

Open CloudWatch and start alarm

Go to CloudWatch → left menu Alarms → All alarms.

Click Create alarm.

Choose metric (CPU of ASG)

Click Select metric.

Navigate:

EC2 → By Auto Scaling Group (or By AutoScalingGroupName).

Click your group ASG-WebServer.

Select CPUUtilization.

Click Select metric.

Set condition (CPU > 60)

Statistic: Average

Period: 1 minute

Under Conditions:

Threshold type: Static

“Whenever CPUUtilization is...” → choose Greater

“than...” → 60

In Additional configuration, datapoints to alarm: 1 out of 1 (default is fine).

Click Next.

Skip actions (no SNS, no ASG here)

On Configure actions page, do NOT add anything.

Just click Next.

If a popup says “*No actions configured*”, click Proceed without actions.

Name and create

Alarm name: ASG-CPU-High-60

Description (optional): CPUUtilization > 60% for 1 minute (ASG-WebServer)

Click Next, then Create alarm.

You now have one metric alarm with no actions.

CloudWatch > Alarms > Create alarm

Specify metric and conditions

Step 1: Specify metric and conditions

Step 2: Configure actions

Step 3: Add alarm details

Step 4: Preview and create

Metric

Graph

Preview of the metric or metric expression and the alarm threshold.

Select metric

Cancel Next

Select metric

CPUUtilization

Percent

0.338
0.267
0.243
0.195

01:45 02:00 02:15 02:30 02:45 03:00 03:15 03:30 03:45 04:00 04:15 04:30 04:45

1h 3h 12h 1d 3d 1w Custom UTC timezone Line G

Browse (2) Multi source query Graphed metrics (1) Options Source

By Auto Scaling Group

Search for any metric, dimension, resource id or account id

CPUUtilization X Clear filters

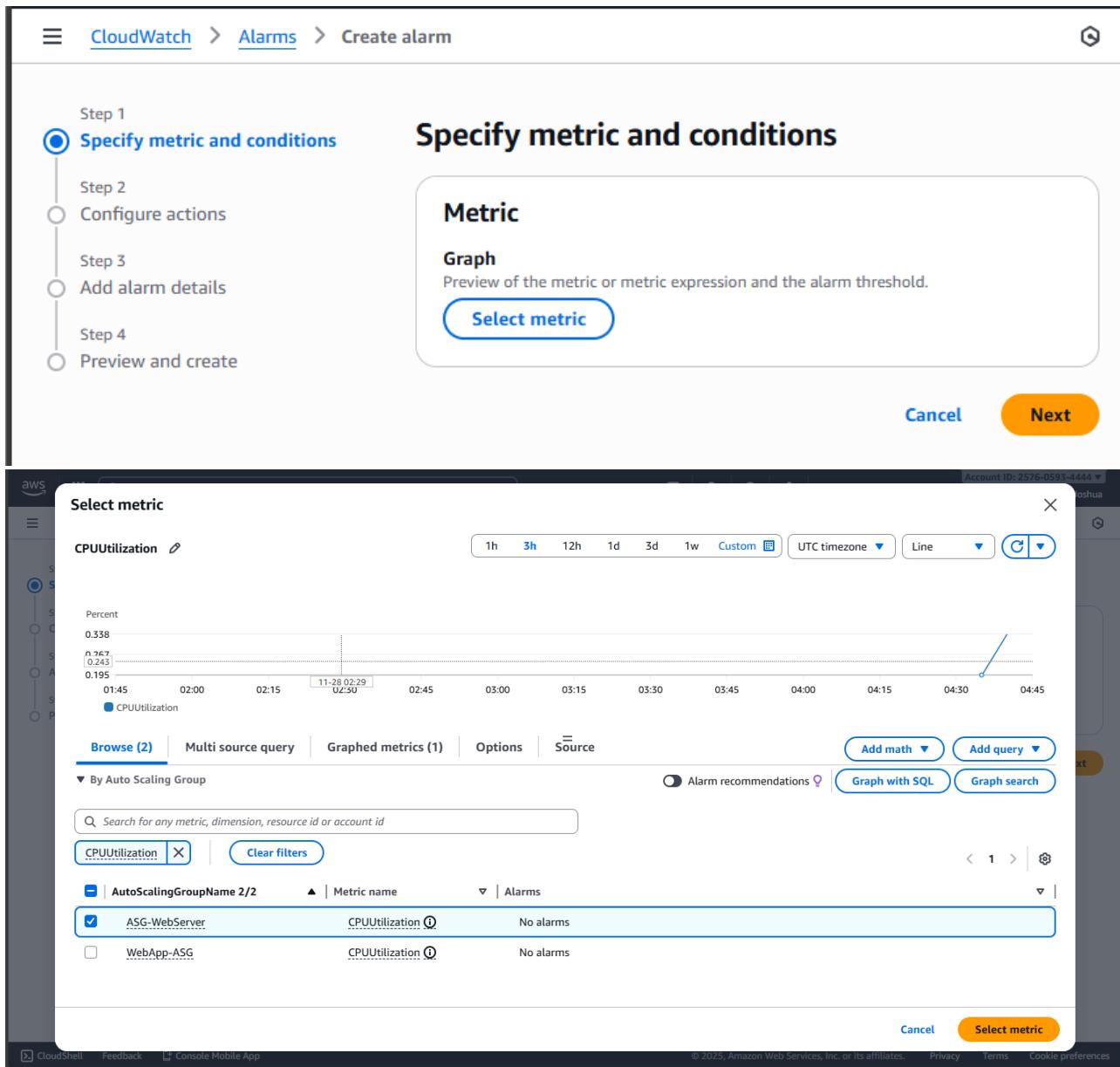
AutoScalingGroupName 2/2 Metric name Alarms

ASG-WebServer CPUUtilization ⓘ No alarms

WebApp-ASG CPUUtilization ⓘ No alarms

Cancel Select metric

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



CloudWatch > Alarms > Create alarm

Step 1 Specify metric and conditions

Step 2 Configure actions

Step 3 Add alarm details

Step 4 Preview and create

Specify metric and conditions

Metric

This alarm will trigger when the blue line goes above the red line for 1 datapoints within 1 minute.

Graph

Percent

60

30.1

0.195

02:00 02:30 03:00 03:30 04:00 04:30

CPUUtilization

Namespace
AWS/EC2

Metric name
CPUUtilization

AutoScalingGroupName
ASG-WebServer

Statistic
Average

Period
1 minute

Conditions

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

CloudWatch > Alarms > Create alarm

Step 1
Specify metric and conditions

Step 2
Configure actions

Step 3
Add alarm details

Step 4
Preview and create

Add alarm details

Name and description

Alarm name

Alarm description - optional [View formatting guidelines](#)

Edit **Preview**

CPUUtilization > 60% for 1 minute (ASG-WebServer)

Up to 1024 characters (49/1024)

Info Markdown formatting is only applied when viewing your alarm in the console. The description will remain in plain text in the alarm notifications.

Tags - optional

CloudShell Feedback [Console Mobile App](#) Privacy Terms Cookie preferences

© 2025, Amazon Web Services, Inc. or its affiliates.

CloudWatch > Alarms > Create alarm

Conditions

Threshold type

Static
Use a value as a threshold

Anomaly detection
Use a band as a threshold

Whenever CPUUtilization is...
Define the alarm condition.

Greater
> threshold

Greater/Equal
>= threshold

Lower/Equal
<= threshold

Lower
< threshold

than...
Define the threshold value.
60

Must be a number.

Additional configuration

Datapoints to alarm
Define the number of datapoints within the evaluation period that must be breaching to cause the alarm to go to ALARM state.
1 out of 1

Missing data treatment
How to treat missing data when evaluating the alarm.
Treat missing data as missing

No actions
You don't have any actions for this alarm.

Step 3: Add alarm details

Edit

Alarm details

Name
ASG-CPU-High-60

Description
CPUUtilization > 60% for 1 minute (ASG-WebServer)

Tags (0)

Markdown formatting is only applied when viewing your alarm in the console. The description will remain in plain text in the alarm notifications.

Cancel **Previous** **Create alarm**

CloudShell **Feedback** **Console Mobile App** © 2025, Amazon Web Services, Inc. or its affiliates. **Privacy** **Terms** **Cookie preferences**

PART 2 – Attach High-CPU alarm to ASG as Scale-OUT Policy

Open ASG automatic scaling

Go to EC2 → left menu Auto Scaling groups.

Click ASG-WebServer.

Go to the Automatic scaling tab.

Create Scale-OUT policy

Click Create dynamic scaling policy.

Policy type: Simple scaling

Scaling policy name: ScaleOut-CPU60

CloudWatch alarm:

Click the dropdown and select ASG-CPU-High-60.

Take the action:

Action: Add

Value: 1

Unit: capacity units

And then wait: 300 seconds (default is fine).

Click Create.

Now your ASG knows:

When alarm ASG-CPU-High-60 goes to ALARM → Add 1 instance.

PART 3 – Create Low-CPU Alarm (for Scale-IN)

Create another alarm in CloudWatch

Go again to CloudWatch → Alarms → Create alarm.

Select metric exactly like before:

EC2 → By Auto Scaling Group → ASG-WebServer → CPUUtilization → Select metric.

Set condition (CPU < 20)

Statistic: Average

Period: 1 minute

Conditions:

Threshold type: Static

“Whenever CPUUtilization is...” → Lower

“than...” → 20

Datapoints to alarm: 1 out of 1 (default).

Click Next.

Skip actions again

Do not add SNS / Auto Scaling / EC2 actions.

Click Next → if popup appears, Proceed without actions.

Name and create

Alarm name: ASG-CPU-Low-20

Description (optional): CPUUtilization < 20% for 1 minute (ASG-WebServer)
Next → Create alarm.

PART 4 – Attach Low-CPU alarm to ASG as Scale-IN Policy

Open ASG automatic scaling again

Back to EC2 → Auto Scaling groups → ASG-WebServer.

Go to Automatic scaling tab.

Create Scale-IN policy

Click Create dynamic scaling policy.

Policy type: Simple scaling

Scaling policy name: scalein-cpu20

CloudWatch alarm:

Choose ASG-CPU-Low-20 from the dropdown.

Take the action:

Action: Remove

Value: 1

Unit: capacity units

And then wait: 300 seconds.

Click Create.

CHECK the resources created in STEP 8-

In EC2 → Auto Scaling groups → ASG-WebServer → Automatic scaling

- You should see:
- ScaleOut-CPU60 – Simple scaling – Add 1 capacity units – Alarm: ASG-CPU-High-60
- scalein-cpu20 – Simple scaling – Remove 1 capacity units – Alarm: ASG-CPU-Low-20

In CloudWatch → Alarms

- ASG-CPU-High-60
- ASG-CPU-Low-20
- Both will show Actions: No actions – this is OK because ASG is using them.

STEP 9 — Perform Stress Testing for Auto Scaling Validation

To artificially increase CPU load on EC2 instances and observe automatic scaling behavior triggered by CloudWatch alarms.

Part A — Connect to the ASG-launched EC2 Instance

Go to AWS Console → EC2 → Instances
Select the instance launched by ASG
(NOT the original base instance you created manually)
Click Connect
Choose EC2 Instance Connect → Connect

Part B — Install Stress Testing Tool

For Amazon Linux 2023:
`sudo dnf install stress-ng -y`
Confirm installation:
`stress-ng --version`

Part C — Apply CPU Load (Trigger Scale Out)

Run stress tool for 2 minutes with 4 CPU workers:
`stress-ng --cpu 4 --timeout 120`

- ✓ This will spike CPU usage above **60%**
- ✓ Your Scale-Out Policy should activate
- ✓ CloudWatch alarm → ALARM state → ASG launches 1 more instance

Part D — Monitor Auto Scaling Activity

Check scaling progress in:

| Service | What to Monitor |
|--------------------------------------|--|
| EC2 → Auto Scaling Groups → Activity | Shows launching of new instance |
| EC2 → Instances | New instance appears with different hostname |
| CloudWatch → Alarms | High CPU alarm turns ALARM |

Scaling may take **2–4 minutes**

Refresh page to see updates

Part E — Test Load Balancing

Copy ALB DNS Name → paste in browser → refresh multiple times:
You should see:
Hello from Instance 1 — ip-xx-xx-xx-xx
Hello from Instance 2 — ip-aa-aa-aa-aa

- ✓ Confirms ALB is distributing traffic across multiple instances

Part F — Scale In (CPU comes down)

After 2 minutes, stress test stops automatically → CPU drops below 20%

- ✓ Low CPU alarm triggers
- ✓ ASG removes the extra instance

✓ Only 1 instance remains (desired capacity)
Scaling-in may take 3–5 minutes

Extra points

This exercise imitates a production-grade architecture used in companies.

What is a Target Group?

A Target Group is a collection of EC2 instances that receive traffic from the Application Load Balancer (ALB).

It defines:

- Which instances to send traffic to
- On which port (example: 80)
- Health check rules (path /, interval, threshold)

In simple words:

Target Group = list of servers behind the load balancer.

Why do we need two steps: Create Launch Template and Create Auto Scaling Group?

Because both have different roles:

Launch Template = WHAT to launch

It contains the configuration of one EC2 instance, such as:

- AMI
- Instance type
- Security Group
- Key pair
- Storage
- User data

This is just a blueprint.

Auto Scaling Group = WHEN & HOW MANY to launch

The ASG uses the Launch Template to automatically:

- Launch new instances
- Remove instances
- Maintain desired capacity
- Scale based on CPU/memory demand

Where to find `http://<alb-dns-name>`?

You can find the ALB DNS Name in the AWS Console:

Path: EC2 → Load Balancers → Select your ALB → Description tab

There you will see: DNS name: mywebapp-alb-12345678.ap-south-1.elb.amazonaws.com

Use it in your browser as: <http://mywebapp-alb-12345678.ap-south-1.elb.amazonaws.com>

This is the public URL of your Load Balancer.

What is ALB DNS?

ALB DNS is the publicly accessible DNS name automatically created by AWS for your Application Load Balancer.

Example: myapp-alb-123456789.ap-south-1.elb.amazonaws.com

When a user types this URL:

- Traffic goes to the ALB
- ALB forwards to Target Group
- Target Group sends request to one of the EC2 instances

ALB DNS = The website URL of your Load Balancer.

DNS means Domain Name System.

DNS is a system that converts domain names to IP addresses.

Example: www.amazon.com → 52.95.120.1

A DNS Server is only one part of this system.

DNS = System that translates names to IPs.

DNS Server = Machine that performs the translation.

Why Do we NEED CloudWatch alarms for ASG scaling exercise?

We are using "Simple Scaling"

- Based on CPU > 60 and CPU < 20
- Which requires manually created CloudWatch alarms

Without the alarms, ASG has NO trigger, so it cannot scale.

Delete these resources in this exact order

To avoid dependency errors:

1. Delete Auto Scaling Group (ASG)

- EC2 → Auto Scaling Groups
 - Select ASG-WebServer
 - Actions → Delete
- This will automatically terminate any instances managed by ASG.

2. Delete Launch Template

- EC2 → Launch Templates
- Select your template
- Actions → Delete template

3. Delete Application Load Balancer

- EC2 → Load Balancers
- Select your ALB

- Actions → Delete

4. Delete Target Group

- EC2 → Load Balancing → Target Groups
- Select your target group
- Actions → Delete

****5. Delete the Custom AMI**

- EC2 → AMIs
- Select your AMI → **Deregister**
- Check Snapshots → delete the related snapshot also.

6. Delete any Manual EC2 Instances and other resources

If you still have any instances running:

- EC2 → Instances → Select → **Terminate**

Resource clean-up NOTE:

- **ALB** charges per hour
- **ASG instances** can launch unexpectedly
- **Custom AMI snapshot** costs storage

Running EC2 will cost hourly

DATE: 21-11-25

Exercise–15: Hosting a WordPress Content Management Website on Amazon Lightsail.

Amazon LightSail is a beginner-friendly cloud platform in AWS that provides:

- Virtual servers (simplified EC2)
- Fixed monthly pricing (predictable cost)
- Simple UI and easy setup
- One-click application launch (WordPress, LAMP, Node.js, etc.)
- Built-in networking (Static IP, DNS, Firewall)
- Automatic SSH terminal
- Snapshots (backups)

It is ideal for:

- Personal websites
- Student projects
- Small web apps
- WordPress blogs
- Quick prototyping

Why Lightsail Instead of EC2?

Lightsail = “EC2 with all hard things simplified.”

| Feature | EC2 | Lightsail |
|---------|------------------------|---------------------------|
| Pricing | Pay per hour | Fixed monthly |
| Setup | Complex (VPC, SG, AMI) | Simple |
| SSH | Need .pem or PuTTY | Built-in SSH |
| DNS | Basic | Easy DNS panel |
| Apps | Install manually | One-click WordPress, LAMP |

3. What is WordPress in Lightsail?

WordPress is a full-stack content management system (CMS) pre-installed on Lightsail.

It includes:

Frontend (UI)

- HTML, CSS, JS
- Themes
- Templates
- Page layouts

Backend

- PHP (server-side logic)
- WordPress core engine

Database

- MySQL/MariaDB
- Stores pages, media, lessons, posts, settings.

Server

- Apache web server

Hosting

- Lightsail VM + static IP + firewall

Thus, WordPress is a full-stack application (LAMP stack).

4. Lightsail Free-Tier Eligibility

- 3 months free
- Only for the 512 MB RAM plan
- Suitable for WordPress, but slightly slow
- After 3 months → ~₹350–₹400/month
- recommended: 1 GB RAM plan.

PART A — Create Server on Lightsail

Step 1 — Open Lightsail Console

AWS Console → Search “Lightsail” → Open it.

Lightsail dashboard appears.

Step 2 — Create Instance

Click Create Instance.

Step 3 — Select Platform:

Choose Linux/Unix. (WordPress works best on Linux)

Step 4 — Select Blueprint → App + OS → WordPress

[Lightsail will install automatically:

- Apache
- PHP
- MySQL/MariaDB
- WordPress application

No manual setup required.]

Step 5 — Choose Instance Plan

512 MB RAM (free-tier for 3 months but slower)

Step 6 — Name the Instance

Give a name (example MyCMS)

Click Create Instance.

Step 7 — Wait Until Status = Running

Takes 1–2 minutes.

PART B — Access and Configure WordPress

Step 8 — Connect Through Browser-Based SSH

Click **Connect** → **Connect using SSH**

(Lightsail gives built-in SSH. No need for .pem file.)

Step 9 — Retrieve WordPress Admin Password

In SSH terminal, run:

`cat bitnami_application_password`

This outputs a long randomly generated password.

Copy it — you need it for login. [example: INBV03eMcWX:]

Step 10 — Open Your WordPress Website

Copy the **Public IP** from the instance details.

Visit: `http://<your-lightsail-ip>`

You will see a WordPress homepage.

Step 11 — Login to WordPress Admin

Visit: `http://<your-lightsail-ip>/wp-admin`

Login:

- Username: **user**
- Password: *paste the password from SSH*

You now have full access to your site's backend.

PART C — Build ‘CMS Website’ [Experiential learning]

Step 12 — Install a Clean Education Theme

Go to:

Dashboard → Appearance → Themes → Add New

Recommended theme: Astra (best)

Activate the theme. [install and activate]

Step 13 — Create Main Website Pages

Dashboard → Pages → Add New

Create:

1. Home
2. Courses & Subjects
3. Lesson Notes
4. Assignments
5. Study Material (PDFs/PPTs)
6. Contact

Step 14 — Create Lesson-Wise Pages

Step 15 — Upload PDFs, PPTs, Notes

Go to:

Dashboard → Media → Add New

Upload:

- PDFs
- DOCX files

- PPTs
- Images
- Notes

Then insert into pages using “Add Media”.

Step 16 — Create Menus

Dashboard → Appearance → Menus

Add:

Home | Courses | Notes | Assignments | Contact

Add submenus under Courses and Notes.

PART D — Secure & Finalize

Step 18 — Attach Static IP

Lightsail → Networking → Create Static IP → Attach to instance.

Reason:

Without this, your IP changes → site breaks.

Step 19 — Take Snapshot

Lightsail → Snapshots → Create snapshot.

Acts as a backup of your entire site.

DATE: 21-11-25

Exercise–16: Building a Basic Python Flask Web Application — Fundamentals of Web Requests & Form Handling (Pre-Requisite for AWS RDS Connectivity Lab)

[Prerequisite for AWS RDS exercise. This particular exercise is a local Python Flask exercise. No AWS service is used here.]

To design and implement a simple Web Application using Python Flask that:

- Displays an HTML form to accept Name and Password.
- Reads the form data on the server side when the form is submitted.
- Displays the submitted values on a new result page.

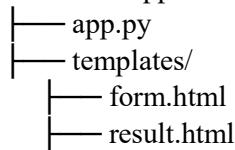
After completing this exercise, you will be able to:

- Understand how an HTML form sends data to the server.
- Explain how Flask receives form data using the request object.
- Read request parameters (request.form) in Flask.
- Understand basic frontend-backend communication.
- Handle a simple HTTP POST request in a Flask application.

Folder Structure

Create the following structure manually:

FlaskFormApp/



Requirements

- Python installed (3.x)
- Flask library
- Any code editor (VS Code / PyCharm / Notepad etc.)
- Web browser (Chrome / Edge / Firefox)

STEP 1: Install Flask

Open **Command Prompt / Terminal** and run:

pip install flask

STEP 2: Create Project Folder

Create a folder, for example:

C:\Users\MCA\FlaskFormApp

STEP 3: Create templates Folder

Inside C:\Users\MCA\FlaskFormApp, create a sub-folder named templates
FlaskFormApp/templates/
All HTML files will be placed in this templates folder.

STEP 4: Create HTML Form — templates/form.html

Create a new file form.html inside templates folder and type the following code:

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML Form</title>
</head>
<body>
    <h2>Enter Your Details</h2>
    <form action="/submit" method="post">
        Name: <input type="text" name="uname"><br><br>
        Password: <input type="password" name="pwd"><br><br>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

Note:

- action="/submit" – sends the form data to the /submit route.
- method="post" – form data is sent using HTTP POST method.
- name="uname" and name="pwd" – these names are used in Flask to read values.

STEP 5: Create Result Page — templates/result.html

Create another file result.html inside templates folder:

```
<!DOCTYPE html>
<html>
<head>
    <title>Result</title>
</head>
<body>
    <h2>Form Submission Result</h2>
    <p><strong>Name:</strong> {{ name }}</p>
    <p><strong>Password:</strong> {{ password }}</p>
```

```
</body>
</html>
```

Note:

- {{ name }} and {{ password }} are **placeholders** (Jinja2 template syntax) that Flask will fill with the values sent from app.py.

STEP 6: Create Flask Backend — app.py

In the FlaskFormApp folder, create app.py and type:

```
from flask import Flask, request, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('form.html')

@app.route('/submit', methods=['POST'])
def submit():
    name = request.form['uname']
    password = request.form['pwd']
    return render_template('result.html', name=name, password=password)

if __name__ == "__main__":
    app.run(debug=True)
```

Explanation:

- @app.route('/') → Home URL, shows the form using form.html.
- @app.route('/submit', methods=['POST']) → Handles form submission.
- request.form['uname'] → Reads the value of input with name="uname".
- render_template('result.html', name=name, password=password) → Sends values to result page.

STEP 7: Run the Application

- Open **Command Prompt / Terminal** inside the FlaskFormApp folder.
- Run:
- python app.py
- You will see something like:
- * Running on http://127.0.0.1:5000
- Open a browser and type the URL: <http://127.0.0.1:5000>
- To stop the Flask server, go to the terminal where it is running and press **Ctrl + C**.

← → ⌂ 127.0.0.1:5000 A☆ InPrivate ...

Enter Your Details

Name:

Password:

← ⌂ 127.0.0.1:5000/submit A☆ InPrivate ...

Form Submission Result

Name: Royal

Password: 1234

DATE: 26-11-25

Exercise-17: Flask Web App with Registration & Login Using AWS RDS (MySQL)
(Full Deployment on AWS EC2 + AWS RDS)

To design and deploy a Python Flask web application on an AWS EC2 instance that allows:

1. User Registration – store Name and Password in AWS RDS MySQL database.
2. User Login – validate user credentials from the RDS database.
3. Welcome Page – display a message on successful login.

This exercise demonstrates a complete cloud-based web application workflow using Flask + MySQL (RDS).

System Architecture

Flow: Browser → Flask App (on EC2) → MySQL Database (RDS)

- Frontend: HTML templates (registration form, login form, welcome page)
- Backend: Python Flask application running on EC2
- Database: AWS RDS – MySQL engine
- Hosting: Linux EC2 instance
- Web Server: Flask's built-in development server (Apache/NGINX not used in this exercise)

Folder Structure (on EC2 instance)

Create the following structure inside the EC2 instance:

FlaskLoginApp/

```
  └── app.py  
  └── db_config.py  
  └── templates/  
      ├── register.html  
      ├── login.html  
      ├── success.html  
      ├── error.html  
      └── welcome.html
```

From EC2 Create Database and Table in RDS MySQL

Where:

The SQL commands are executed from the EC2 instance, by connecting to the RDS MySQL database using the MySQL client.

When:

After the RDS MySQL instance status becomes “Available” and after allowing EC2 security group access in RDS inbound rules.

Create:

- A database: flaskdb
- A table: users

SQL: Create Database and Table

```
CREATE DATABASE flaskdb;
```

```
USE flaskdb;
```

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    password VARCHAR(50) NOT NULL
);
```

- id – unique identifier for each user (auto-increment primary key).
- name – username (cannot be NULL).
- password – password (cannot be NULL).

Note: For learning purposes, we store plain text. In real systems, passwords must be **hashed**.

Flask Application Code (Backend)

6.1 db_config.py

This file stores the **RDS MySQL connection details**.

```
# db_config.py
```

```
db_config = {
    'host': 'your-rds-endpoint.amazonaws.com',
    # e.g., flaskdb-instance.xxxxxxx.ap-south-1.rds.amazonaws.com
    'user': 'admin',           # master username created in RDS
    'password': 'YourPasswordHere', # master password created in RDS
    'database': 'flaskdb'       # database name created in MySQL
}
```

Replace host, user, and password with the actual values from the RDS instance.

6.2 app.py

This is the **main Flask application**.

```
# app.py
```

```
from flask import Flask, request, render_template
import mysql.connector
from db_config import db_config

app = Flask(__name__)

# Function to create a new database connection
def get_connection():
    return mysql.connector.connect(**db_config)

@app.route('/')
def home():
```

```
return render_template('register.html')

@app.route('/register', methods=['POST'])
def register():
    name = request.form['uname']
    password = request.form['pwd']

    conn = get_connection()
    cur = conn.cursor()

    sql = "INSERT INTO users (name, password) VALUES (%s, %s)"
    cur.execute(sql, (name, password))
    conn.commit()

    cur.close()
    conn.close()

    # Use HTML template for success page
    return render_template('success.html')

@app.route('/login')
def login():
    return render_template('login.html')

@app.route('/validate', methods=['POST'])
def validate():
    name = request.form['uname']
    password = request.form['pwd']

    conn = get_connection()
    cur = conn.cursor()

    sql = "SELECT * FROM users WHERE name = %s AND password = %s"
    cur.execute(sql, (name, password))
    user = cur.fetchone()

    cur.close()
    conn.close()

    if user:
        # Successful login -> welcome page
        return render_template('welcome.html', username=name)
    else:
        # Invalid login -> error page
        return render_template('error.html')
```

```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

7. HTML Templates (Frontend)

All HTML files go inside the templates/ folder.

7.1 templates/register.html

```
<!DOCTYPE html>
<html>
<head>
    <title>User Registration</title>
</head>
<body>
    <h2>User Registration</h2>

    <form action="/register" method="post">
        <label>Name:</label>
        <input type="text" name="uname" required><br><br>

        <label>Password:</label>
        <input type="password" name="pwd" required><br><br>

        <button type="submit">Register</button>
    </form>

    <br>
    <a href="/login">Already registered? Click here to Login</a>
</body>
</html>
```

7.2 templates/login.html

```
<!DOCTYPE html>
<html>
<head>
    <title>User Login</title>
</head>
<body>
    <h2>User Login</h2>

    <form action="/validate" method="post">
        <label>Name:</label>
        <input type="text" name="uname" required><br><br>
```

```
<label>Password:</label>
<input type="password" name="pwd" required><br><br>

<button type="submit">Login</button>
</form>

<br>
<a href="/">New user? Click here to Register</a>
</body>
</html>
```

7.3 templates/welcome.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Welcome</title>
</head>
<body>
    <h2>Welcome, {{ username }}!</h2>
    <p>You have successfully logged in.</p>
</body>
</html>
```

7.4 templates/success.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Registration Successful</title>
</head>
<body>
    <h3>Registration Successful!</h3>
    <a href="/login">Click here to Login</a>
</body>
</html>
```

7.5 templates/error.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Login Error</title>
</head>
```

```
<body>
  <h3>Invalid Username or Password</h3>
  <a href="/login">Try Again</a>
</body>
</html>
```

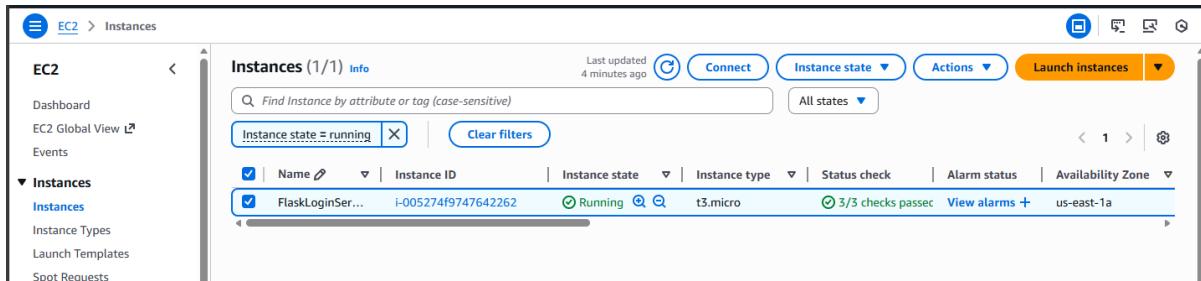
Deployment Steps –

PHASE 1 – Launch EC2 Linux Instance

- Open AWS Management Console → EC2 → Launch instance.
- Configuration:
 - Name: FlaskLoginServer
 - AMI: Amazon Linux 2023 (Free Tier eligible)
 - Instance type: t3.micro (Free Tier)
 - Key pair: create or select existing
 - Network: Default VPC and any public subnet
 - Auto-assign Public IP: Enable
- Security Group for EC2 – Inbound rules:

| Type | Port | Source | Purpose |
|------------|------|-----------|--------------------------------|
| SSH | 22 | My IP | To connect via SSH/EC2 Connect |
| Custom TCP | 5000 | 0.0.0.0/0 | To test Flask app in browser |

- Outbound: Allow all traffic.
- Click Launch instance.



PHASE 2 – Connect to EC2 and Install Dependencies

- Go to EC2 → Instances → Select instance → Connect.
- Choose EC2 Instance Connect (browser-based SSH) → Connect.
Or open PowerShell on windows and connect using ssh command
- Update packages:
`sudo yum update -y`
- Check Python version (Amazon Linux 2023 has Python 3):
`python3 --version`
- Install pip (if not already installed):
`sudo dnf install python3-pip -y`
- Install Flask and MySQL Connector for Python:
`pip3 install flask`
`pip3 install mysql-connector-python`

PHASE 3 – Create Flask Project Structure on EC2

- Create project folder:

- ```

mkdir FlaskLoginApp
cd FlaskLoginApp

```
- Create templates folder:  
`mkdir templates`
  - Create the Python and HTML files using nano:  
`nano db_config.py → paste code for db_config.py`  
`nano app.py → paste code for app.py`  
`nano templates/register.html → paste register page`  
`nano templates/login.html → paste login page`  
`nano templates/welcome.html → paste welcome page`

Save each file using: CTRL + O → Enter → CTRL + X

```

[ec2-user@ip-172-31-2-32 FlaskLoginApp]$ tree -a
.
└── app.py
└── db_config.py
└── templates
 ├── error.html
 ├── login.html
 ├── register.html
 ├── success.html
 └── welcome.html

1 directory, 7 files

```

#### PHASE 4 – Create RDS MySQL Instance

- Go to AWS Console → RDS → Databases → Create database.
    - Engine type: MySQL
    - Templates: Free tier
    - DB instance identifier: flaskdb-instance
    - Master username: admin (example)
    - Master password: <your-password>
    - DB instance class: db.t3.micro (Free tier)
    - Storage: e.g. 20 GB (Free tier)
    - Connectivity:
      - Connect to an EC2 compute resource → YES
      - Select the same EC2 instance (or its security group)
      - (AWS automatically performs the following:
        - Selects the same VPC as EC2
        - Creates/associates an RDS security group
        - Adds inbound rule: MySQL (3306) from EC2 security group
  - Public access: No
  - Click Create database.
- Wait until the status becomes “Available”.

The screenshot shows the AWS RDS Databases console. On the left, there's a sidebar with 'Aurora and RDS' selected. The main area is titled 'Databases (1)' and shows a table with one row for 'flaskdb-instance'. The columns include DB identifier, Status, Role, Engine, Upgrade rollout order, Region, and Size. The database is marked as 'Available' and is an 'Instance' type running MySQL Community Server (SECONDARY) in the us-east-1d region with a db.t2.micro size.

## PHASE 5 – Verify RDS Security Group (Auto-configured)

- Go to AWS Console → RDS → Databases → flaskdb-instance.
- Open Connectivity & security section.
- Under VPC security groups, click the linked security group name.
- In Inbound rules, verify that the following rule exists:

| Type           | Port | Source             |
|----------------|------|--------------------|
| MySQL / Aurora | 3306 | EC2 Security Group |

5. If the rule exists, EC2 to RDS connectivity is correctly configured.

The screenshot shows the AWS EC2 Security Groups console. On the left, there's a sidebar with 'EC2' selected. The main area is titled 'sg-0f087d057ac6edce8 - rds-ec2-3'. It shows details like the security group name (rds-ec2-3), owner (257605934444), security group ID (sg-0f087d057ac6edce8), and a description indicating it's attached to the RDS instance. Below this, the 'Inbound rules' tab is selected, showing a table with one rule. The rule has a security group rule ID (sgr-0b77f95139084a5f4), IP version (-), type (MySQL/Aurora), protocol (TCP), and port range (3306).

## PHASE 6 – Create Database and Table in RDS from EC2

- From the EC2 terminal, install MySQL client (MariaDB client):
 

```
sudo dnf install mariadb105 -y
if this fails, try: sudo dnf install mariadb -y
```
- Connect to the RDS MySQL instance from EC2:
 

```
mysql -h <RDS-endpoint> -u admin -p
Example: mysql -h flaskdb-instance.xxxxxx.ap-south-1.rds.amazonaws.com -u admin -p
```

```
[ec2-user@ip-172-31-6-43 FlaskLoginApp]$ mysql -h flaskdb.cczscwsuznp.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 27
Server version: 8.0.43 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> |
```

- Inside the MySQL prompt, run following commands:

```
CREATE DATABASE flaskdb;
```

```
USE flaskdb;
```

```
CREATE TABLE users (
 id INT AUTO_INCREMENT PRIMARY KEY,
 name VARCHAR(50) NOT NULL,
 password VARCHAR(50) NOT NULL
);
```

- Exit MySQL: exit;

```
MySQL [(none)]> CREATE DATABASE flaskdb
-> ;
Query OK, 1 row affected (0.012 sec)

MySQL [(none)]> USE flaskdb;
Database changed
MySQL [flaskdb]> CREATE TABLE users (
-> id INT AUTO_INCREMENT PRIMARY KEY,
-> name VARCHAR(50) NOT NULL,
-> password VARCHAR(50) NOT NULL
->);
Query OK, 0 rows affected (0.040 sec)
```

## PHASE 7 – Configure Flask to Use RDS and Run Application

- Open db\_config.py on EC2 and update with **correct RDS details**:  
nano db\_config.py  
Ensure:  

```
db_config = {
 'host': 'your-rds-endpoint.amazonaws.com',
 'user': 'admin',
 'password': 'YourPasswordHere',
 'database': 'flaskdb'
}
```

Save and exit.
- Start the Flask app:

```
cd ~/FlaskLoginApp
python3 app.py
You should see: * Running on http://0.0.0.0:5000
```

## PHASE 8 – Test the Application from Browser

- In your local system browser, open: http://<EC2-public-IP>:5000  
Example: http://13.232.122.81:5000  
You should see: Registration Page (default route /)



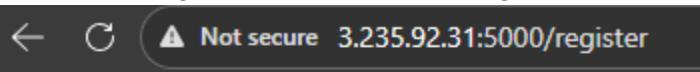
### User Registration

Name:

Password:

[Already registered? Click here to Login](#)

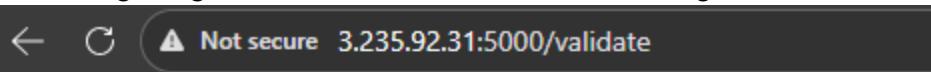
- After registration → success message with link to /login



### Registration Successful!

[Click here to Login](#)

- Login Page → if correct credentials → Welcome Page



### Welcome, joshua!

You have successfully logged in.

Check that data is stored in flaskdb.users table in RDS. - SELECT \* FROM users;

## Expected Output

1. Registration Page

- User enters Name and Password.
  - On submit, data is inserted into users table in RDS MySQL.
2. Login Page
    - User enters same Name and Password.
    - Flask executes SELECT query on users table to check match.
  3. Welcome Page
    - For valid credentials:  
“Welcome, <username>! You have successfully logged in.”
    - For invalid credentials:  
“Invalid Username or Password” message with link to try again.

## Result

A **fully working cloud-based registration and login system** was successfully deployed using:

- **Flask** as the backend web framework
- **HTML** templates for frontend pages
- **MySQL** database hosted on **AWS RDS**
- **AWS EC2** instance as the application server

This exercise demonstrates a **complete end-to-end mini project** on AWS using **Flask + MySQL (RDS)**.

**DATE: 28-12-25**

**Exercise-18:** Creating and Operating a NoSQL Key-Value Database using Amazon DynamoDB

To create an Amazon DynamoDB table and perform CRUD operations (Create, Read, Update, Delete) using the AWS Management Console, and understand Partition Key, Sort Key, Query vs Scan, and table cleanup.

**Phase 1: Create DynamoDB Table**

Create a DynamoDB table to store Student Records.

Store items like: USN, Name, Semester, Course, Attendance, IA1Marks

**Step 1: Open DynamoDB**

- Login to AWS Console
- Search → type DynamoDB → open Amazon DynamoDB

**Step 2: Create a Table**

- Left menu → Tables
- Click Create table
- Fill details:

Table details

- Table name: MCA\_StudentLabInternals
- Partition key (PK): USN (Type: String)
- Sort key (SK): CourseCode (Type: String)  
(Enable sort key option by setting sort key)

Why this design?

- One student can have multiple courses → Sort key separates records per course.
- This creates a composite primary key (PK + SK).

**Step 3: Table settings**

- Under Table settings, choose:  
Default settings (recommended for lab)
- Ensure Capacity mode is: On-demand (default already selected)

**Step 4: Create**

- Click Create table
- Wait until table status becomes Active

The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. The 'Table details' section has 'Table name' set to 'MCA\_StudentLabInternals'. The 'Partition key' section has 'USN' as the primary key of type 'String'. The 'Sort key - optional' section has 'CourseCode' as the secondary key of type 'String'. Under 'Table settings', the 'Default settings' option is selected. At the bottom, there are links for CloudShell, Feedback, and Console Mobile App, along with copyright information and privacy terms.

## Phase 2: Insert Items (Create Data)

### Step 5: Open Table and Add Items

- Click the table: MCA\_StudentLabInternals
- Click: Explore table items
- Click Create item

### Step 6: Add Item 1 (Student 1 - Course 1)

- You will see attributes:
  - USN (String)
  - CourseCode (String)

Enter:

- USN = 1MS24MCA001
- CourseCode = CCL301

Now add additional attributes (click **Add new attribute**):

- Name (String) = Arun
- Semester (Number) = 3
- Attendance (Number) = 86
- IA1Marks (Number) = 18

Click **Create item**

The screenshot shows the 'Create item' page in the Amazon DynamoDB console. At the top, there's a feedback banner: 'Share your feedback on Amazon DynamoDB. Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.' Below it, the 'Create item' section title is visible. A note says: 'You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. Learn more' with a link. To the right are 'Form' and 'JSON view' buttons. The main area is titled 'Attributes' with a 'Add new attribute' button. It lists six attributes:

| Attribute name        | Value       | Type   | Remove                  |
|-----------------------|-------------|--------|-------------------------|
| USN - Partition key   | 1MS24MCA001 | String |                         |
| CourseCode - Sort key | CCL301      | String |                         |
| Name                  | Abishek     | String | <button>Remove</button> |
| Sem                   | 3           | Number | <button>Remove</button> |
| Attendance            | 70          | Number | <button>Remove</button> |
| IA1 Marks             | 30          | Number | <button>Remove</button> |

At the bottom are 'Cancel' and 'Create item' buttons.

### Step 7: Add Item 2 (Same student - another course)

Repeat Create item with:

- USN = 1MS24MCA001
- CourseCode = DBS301
- Name = Arun
- Semester = 3
- Attendance = 88
- IA1Marks = 20

### Step 8: Add Item 3 (Another student)

Create item:

- USN = 1MS24MCA002
- CourseCode = CCL301
- Name = Chitra
- Semester = 3
- Attendance = 74
- IA1Marks = 14

You should now see 3 items in the table list.

Completed · Items returned: 0 · Items scanned: 0 · Efficiency: 100% · RCUs consumed: 2

### Table: MCA\_StudentLabInternals - Items returned (3)

Actions ▾

Scan started on December 19, 2025, 09:36:02

< 1 > |

|  | USN (String)                | CourseCode (String) | Attendance | IA1Marks | Name     |
|--|-----------------------------|---------------------|------------|----------|----------|
|  | <a href="#">1MS24MCA001</a> | DBS301              | 85         | 80       | Abhishek |
|  | <a href="#">1MS24MCA002</a> | CCL301              | 90         | 25       | Aditya   |
|  | <a href="#">1MS24MCA001</a> | CCL301              | 70         | 30       | Abhishek |

© 2025, Amazon Web Services, Inc. or its affiliates.

[Privacy](#)

[Terms](#)

[Cookie preferences](#)

### Phase 3: Read Data (Get / Query / Scan)

#### Step 9: Get a single item (exact PK + SK)

- In “Explore table items”, use search/filter:
- Use USN = 1MS24MC001 and CourseCode = CCL301
- Open the item → verify all attributes

#### Key concept:

To uniquely identify an item in a PK+SK table, you must provide **both**.

DynamoDB > Explore items > MCA\_StudentLabInternals

**DynamoDB**

- Dashboard
- Tables
- Explore items**
- PartiQL editor
- Backups
- Exports to S3
- Imports from S3
- Integrations
- Reserved capacity
- Settings

**DAX**

- Clusters
- Subnet groups
- Parameter groups
- Events

**Find tables**

**Select a table or index**: Table - MCA\_StudentLabInternals

**Select attribute projection**: All attributes

**Filters - optional**

| Attribute name | Condition | Type   | Value       |
|----------------|-----------|--------|-------------|
| USN            | Equal to  | String | 1MS24MCA001 |
| CourseCode     | Equal to  | String | CCL301      |

**Run** **Reset**

Completed · Items returned: 1 · Items scanned: 3 · Efficiency: 33.33% · RCUs consumed: 2

**Table: MCA\_StudentLabInternals - Items returned (1)**

Scan started on December 19, 2025, 09:58:27

< 1 > |

|  | USN (String)                | CourseCode (String) | Attendance | IA1Marks | Name     | Sem |
|--|-----------------------------|---------------------|------------|----------|----------|-----|
|  | <a href="#">1MS24MCA001</a> | 1                   | 70         | 30       | Abhishek | 3   |

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

#### Step 10: Query – fetch all courses for one student

- Click Run query
- Set Partition key condition: USN equals 1MS24MCA001
- Run

Expected output:

- Items for Arun in both CCL301 and DBS301

Key concept:

Query works only with Partition Key (and optional Sort Key conditions). It is efficient.

The screenshot shows the AWS DynamoDB console. On the left, there's a sidebar with various navigation links: Dashboard, Tables, Explore items (which is selected), PartitionQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Below that is a section for DAX with Clusters, Subnet groups, Parameter groups, and Events. The main content area is titled 'MCA\_StudentLabInternals'. It has tabs for 'Scan or query items' (selected) and 'View table details'. Under 'Scan or query items', there are sections for 'Partition key' (set to USN with value 1MS24MCA002) and 'Sort key - optional' (set to CourseCode with value CCL301). There's also a 'Filters - optional' section which is currently empty. At the bottom of this panel, there are buttons for 'Add filter', 'Actions', and 'Create item'. Below this panel, the table data is displayed with the following schema: USN (String), CourseCode (String), Attendance, IA1Marks, Name, and Sem. One row is shown: USN 1MS24MCA002, CourseCode CCL301, Attendance 90, IA1Marks 25, Name Aditya, and Sem 3.

#### Table: MCA\_StudentLabInternals - Items returned (1)

Query started on December 19, 2025, 09:48:57

|  | USN (String) | CourseCode (String) | Attendance | IA1Marks | Name   | Sem |
|--|--------------|---------------------|------------|----------|--------|-----|
|  | 1MS24MCA002  | CCL301              | 90         | 25       | Aditya | 3   |

#### Step 11: Scan (Not for large tables)

- Click Scan
- Run scan without filters

Expected output:

- All items (Arun + Chitra)

Key concept:

Scan reads the entire table → slow/costly for big tables.

#### Phase 4: Update Data

#### Step 12: Update IA1 marks of Chitra for CCL301

- Open item where:  
USN = 1MS24MCA002

- CourseCode = CCL301
- Click Edit
  - Change: IA1Marks from 14 to 16
  - Click Save changes

Verify updated value appears.

#### **Phase 5: Delete Data**

##### **Step 13: Delete one item (Arun's DBS301 record)**

- Select item:
  - USN = 1MS24MCA001
  - CourseCode = DBS301
- Click **Delete**
- Confirm delete

Now total items should reduce by 1.

#### **Phase 6: Cleanup (Must Do to avoid charges)**

##### **Step 15: Delete the Table**

- DynamoDB → Tables
- Select MCA\_StudentLabInternals
- Click Delete
- Type confirmation (if asked) and delete

Ensure table disappears from list.

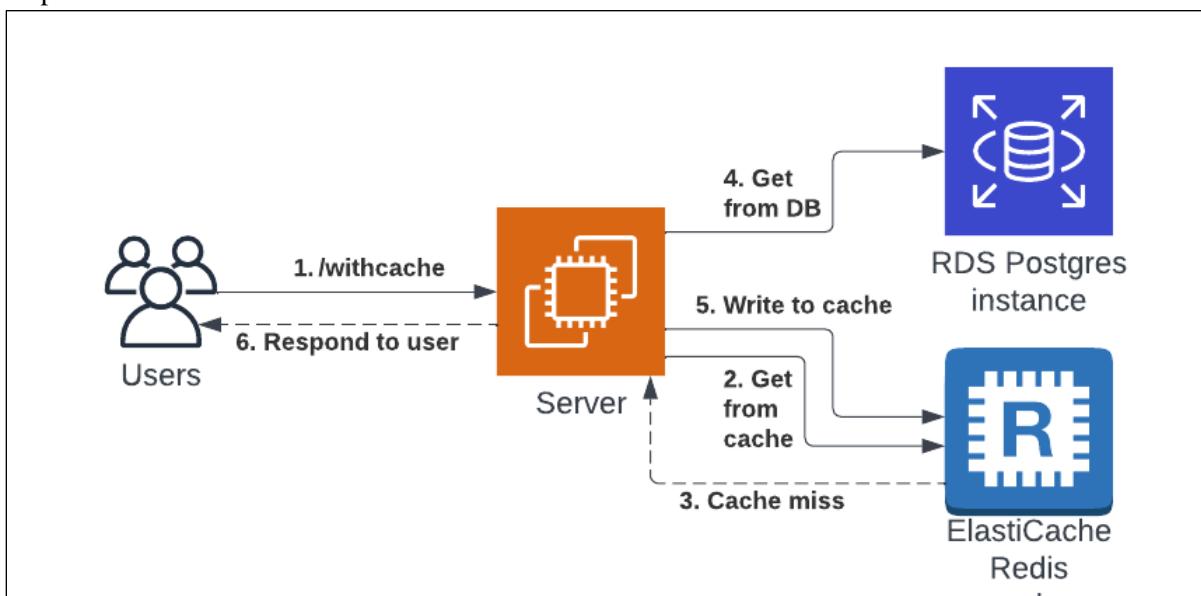
**DATE: 03-12-25**

### **Exercise-19: ElastiCache (Redis) as an In-Memory Cache**

To implement Amazon ElastiCache (Redis) as an in-memory caching service by deploying a Redis cluster and performing basic cache operations from an EC2 instance.

#### **Application-Level Data Flow (Logical)**

- Step 1: User requests data
- Step 2: Application checks ElastiCache (Redis)
- Step 3: If data exists → return from cache (FAST)
- Step 4: If data does not exist → fetch from RDS
- Step 5: Store fetched data in ElastiCache
- Step 6: Return data to user



This exercise provides **exposure** to an industry-used in-memory data store.

Redis improves performance by serving frequently accessed data from memory and is typically used **in front of databases** in real-world applications.

| Feature       | RDS       | ElastiCache |
|---------------|-----------|-------------|
| Storage       | Disk      | RAM         |
| Data lifetime | Permanent | Temporary   |
| Access speed  | Slower    | Very fast   |
| TTL support   | No        | Yes         |

Phase-1 → EC2 creation + valkey-cli installation

Phase-2 → Redis cache creation

Phase-3 → Connect EC2 → Redis engine and run commands

**Phase-1: Launch EC2 Client using Amazon Linux 2023 (for Valkey / Redis Client)**

## Purpose of Phase-1

To create an EC2 instance using **Amazon Linux 2023** that will act as a **client machine** to connect to Amazon ElastiCache (Redis OSS) using **valkey-cli**.

### Step 1: Select AWS Region

- Choose ONE region and stick to it for the entire exercise  
ap-south-1 (Mumbai) (*recommended*)
- Ensure ElastiCache and EC2 will be in the SAME region

### Step 2: Launch EC2 Instance

1. Go to **AWS Console → EC2**
2. Click **Launch instance**

### Step 3: Configure Instance Basics

- Name: RedisClient-AL2023
- AMI: Select Amazon Linux 2023 AMI
- Instance Type: Select t3.micro (Free Tier eligible)
- Key Pair: Select an existing key pair OR create a new one (RSA, .pem)
- Network & Security
  - Network
    - VPC: Default VPC
    - Subnet: No preference
    - Auto-assign public IP: Enabled
  - Security Group (VERY IMPORTANT)  
Create a new security group:
    - Security Group Name: SG-RedisClient
    - Description: Security group for Redis client EC2Inbound Rules

| Type | Port | Source             |
|------|------|--------------------|
| SSH  | 22   | Anywhere 0.0.0.0/0 |

Do NOT add Redis (6379) here (The client does not listen on 6379)

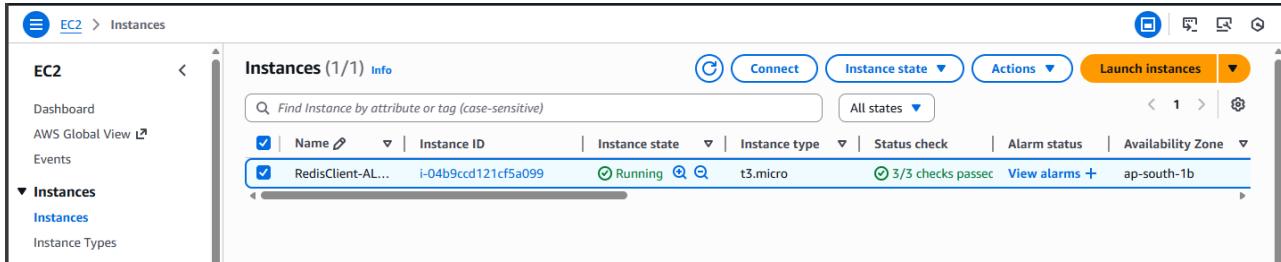
Outbound rules: Allow all (default)

### Step 5: Storage

- Keep default

### Step 6: Launch Instance

- Click Launch instance
- Wait until Instance State = Running



## Step 7: Connect to EC2

1. EC2 → Instances → select RedisClient-AL2023
2. Click Connect
3. Choose EC2 Instance Connect
4. Click Connect

You are now inside the EC2 terminal.

## Step 8: Install Valkey Client

Run the following commands:

```
sudo dnf update -y
sudo dnf install -y valkey
```

Verify installation:

```
valkey-cli --version
```

### Expected Output (example)

```
valkey-cli 8.x.x
```

This confirms the EC2 is ready as a Redis-compatible client.

```
Running transaction
 Preparing : 1/1
 Running scriptlet: valkey-8.0.6-3.amzn2023.0.2.x86_64 1/1
 Installing : valkey-8.0.6-3.amzn2023.0.2.x86_64 1/1
 Running scriptlet: valkey-8.0.6-3.amzn2023.0.2.x86_64 1/1
 Verifying : valkey-8.0.6-3.amzn2023.0.2.x86_64 1/1

 Installed:
 valkey-8.0.6-3.amzn2023.0.2.x86_64

 Complete!
[ec2-user@ip-172-31-14-150 ~]$ valkey-cli --version
valkey-cli 8.0.6
[ec2-user@ip-172-31-14-150 ~]$ |
```

Note: why install valkey and not redis??

Amazon Linux 2023 does not provide redis-cli directly.

AWS now provides **Valkey**, which is fully Redis-protocol compatible.

Hence, we use **valkey-cli** to connect to ElastiCache Redis.

## **Phase-2: Create Amazon ElastiCache (Redis OSS) Cluster**

### **Purpose of Phase-2**

To create an Amazon ElastiCache Redis OSS cluster that will act as an in-memory cache, accessible from the EC2 client created in Phase-1.

#### **Step 1: Confirm AWS Region**

- Ensure you are in the **same region** used in Phase-1  
EC2 and ElastiCache must be in the SAME region and VPC.

#### **Step 2: Open ElastiCache Console**

1. AWS Console → Services
2. Select ElastiCache
3. Click Create cache

#### **Step 3: Select Cache Engine**

- **Engine: Redis OSS**

#### **Step 4: Choose Deployment Settings**

- Deployment option: Node-based cluster
- Creation method: Easy create

Easy create is used to avoid advanced production settings.

#### **Step 5: Select Configuration**

- **Configuration: Demo**

This automatically selects:

- A small, low-cost node (e.g., cache.t4g.micro)
- Suitable for labs and practice

#### **Step 6: Provide Cluster Information**

- Cache name: lab-redis
- Description: Optional (lab-redis)

#### **Step 7: Configure Network and Subnet Group**

##### **Network**

- Network type: IPv4

##### **Subnet Group**

- Select: Create a new subnet group
  - Subnet group name: redis-subnet-group
  - VPC: Default VPC
  - Subnets: Leave AWS auto-selected subnets unchanged

No manual subnet selection required.

#### **Step 8: Configure Security**

##### **Security Group**

- Create or select a security group:
  - Name: SG-RedisCache

### Inbound Rule for Redis

Add one inbound rule to SG-RedisCache:

| Type       | Port | Source         |
|------------|------|----------------|
| Custom TCP | 6379 | SG-RedisClient |

This allows only the EC2 client to access Redis.

### Step 9: Authentication

- Authentication: Disabled

### Step 10: Create Cache

1. Review all settings
2. Click Create
3. Wait until Status = Available

This may take a few minutes.

| Cache name | Status    | Description                             | Engine version | Configuration   |
|------------|-----------|-----------------------------------------|----------------|-----------------|
| lab-redis  | Available | Easy created demo cluster on 2026-01... | 7.1.0          | cache.t4g.micro |

### Step 11: Note the Redis Endpoint

1. Click on the cache name lab-redis
2. Copy the Configuration Endpoint
  - Example: lab-redis.xxxxxx.cache.amazonaws.com

This endpoint will be used in **Phase-3** to connect from EC2.

Note:

We have created an in-memory Redis cache using Amazon ElastiCache.

It runs inside the AWS VPC and does not have a public IP.

Only our EC2 client is allowed to access it using port 6379.

## Phase-3: Connect EC2 to ElastiCache Redis and Execute Cache Commands

### Purpose of Phase-3

To connect the EC2 client (Amazon Linux 2023) to the ElastiCache Redis OSS cluster using valkey-cli and demonstrate basic in-memory cache operations.

**Pre-checks** - Before connecting, confirm:

- EC2 and ElastiCache are in the same AWS region
- EC2 security group = SG-RedisClient
- Redis security group = SG-RedisCache
- Redis security group allows port 6379 from SG-RedisClient
- Redis Status = Available
- You have copied the Primary Endpoint

### **Step 1: Connect to EC2**

1. AWS Console → **EC2**
2. Select instance RedisClient-AL2023
3. Click **Connect**
4. Choose **EC2 Instance Connect**
5. Click **Connect**

You are now inside the EC2 terminal.

### **Step 2: Verify Valkey Client**

Run: valkey-cli --version

Expected Output (example)

valkey-cli 8.x.x

This confirms the EC2 is ready to act as a Redis-compatible client.

### **Step 3: Connect to ElastiCache Redis**

Use the Configuration Endpoint from Phase-2.

valkey-cli -h <Configuration-ENDPOINT> -p 6379

Example: valkey-cli -h **lab-redis.xxxxxx.cache.amazonaws.com** -p 6379 --tls -c

### **Expected Result**

You should see a prompt like:

lab-redis.xxxxxx.cache.amazonaws.com:6379>

This means the connection is successful.

### **Step 4: Execute Redis / Valkey Commands**

These commands simulate what an application does internally.

#### **Test Connectivity**

PING

#### **Expected Output**

PONG

```
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> ping
PONG
```

#### **Store and Retrieve Data (SET / GET)**

SET course "Cloud Computing"

GET course

```
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> SET course "Cloud Computing"
OK
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> Get course "Cloud Computing"
```

#### Expected Output

"Cloud Computing"

Demonstrates key-value storage in memory.

#### Counter Example (INCR)

INCR visits

INCR visits

GET visits

#### Expected Output

"2"

NOTE: Common real-world use - page views, hit counters.

```
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> INCR visits
(integer) 1
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> INCR visits
(integer) 2
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> GET visits
"2"
```

#### 4.4 Set Data with Expiry (TTL)

SET notice "Results Published" EX 60

TTL notice

GET notice

#### Expected Output

- TTL shows a value  $\leq 60$
- GET returns:

"Results Published"

Shows temporary cache data.

```
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> SET notice "Results Published" EX 60
OK
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> TTL notice
(integer) 54
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> GET notice
"Results Published"
```

#### 4.5 Verify Automatic Expiry

Wait ~60 seconds, then run:

GET notice

#### Expected Output

(nil)

Confirms data is automatically removed from memory.

```
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> GET notice
(nil)
```

#### 4.6 Delete Data Manually

DEL course

GET course

##### Expected Output

(nil)

##### Step 5: Exit Redis Client

EXIT

```
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> DEL course
(integer) 1
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> GET course
(nil)
clustercfg.lab-redis.0e01xv.apsl.cache.amazonaws.com:6379> EXIT
[ec2-user@ip-172-31-14-150 ~]$
```

##### Note:

“The application first checks Redis.

If data is present, it is returned immediately from memory.

If not present, the application fetches data from the database and stores it in Redis with a TTL.

Redis automatically removes the data after expiry.”

#### Phase-3 Outcome

- EC2 successfully connected to ElastiCache Redis
- In-memory key-value operations verified
- Temporary storage and TTL behavior observed
- Redis used as a **cache**, not as a primary database

#### Common Errors & Quick Fix

| Problem            | Reason              | Fix                              |
|--------------------|---------------------|----------------------------------|
| Connection timeout | Wrong SG or region  | Check SG-RedisCache inbound rule |
| Command hangs      | Redis not Available | Wait & retry                     |
| (nil) output       | Key expired         | Expected behavior                |

#### Clean-Up (Mandatory)

##### Step 1: Delete Redis Cluster

- ElastiCache → Select lab-redis → Delete
- Disable snapshots

##### Step 2: Terminate EC2 Instance

- EC2 → Instances → Terminate RedisClient-EC2

##### Step 3: Optional

- Delete unused security groups

#### Redis Commands Explanation

- SET course "Cloud Computing"
- GET course

- **EXPIRE** course 30

- **TTL** course

- The above commands simulate application caching behavior.

The SET command represents storing frequently accessed data in cache.

The EXPIRE command shows that cached data is temporary and stored in memory.

After expiry, the application would fetch fresh data from the database again.

**DATE: 05-12-25**

**Exercise–20:** Deploy a simple Flask application on AWS Elastic Beanstalk and verify it using the public URL.

**Phase A — Create the Flask App**

**Step A1: On your system, create a folder**

Create a folder: eb-flask-lab

**Step A2: Create application.py**

Create a file named application.py:

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def home():
 return "Hello from Flask on Elastic Beanstalk!"

@app.route("/health")
def health():
 return "OK"

if __name__ == "__main__":
 app.run()
```

**Step A3: Create requirements.txt**

Create a file named **requirements.txt** and add:

```
Flask==3.0.3
gunicorn==22.0.0
```

**Step A4: Create Procfile (MOST IMPORTANT)**

Create a file named **exactly**:

Procfile (no extension)

Contents:

```
web: gunicorn application:app
```

[Windows Notepad method:

File name: Procfile, Save as type: All Files, Encoding: UTF-8]

## Step A5: Verify folder contents

Your folder must contain exactly these 3 files:

eb-flask-lab

```
└── application.py
└── requirements.txt
```

└── Procfile (Type should show as "File", NOT Procfile.txt)

## Phase B — Create the ZIP correctly (root-level ZIP)

### Step B1: Select the 3 files

Select:

- application.py
- requirements.txt
- Procfile

### Step B2: Create ZIP

Right click → Compress to → ZIP file

Rename the ZIP as:

eb-flask-lab.zip

### Step B3: Confirm ZIP contents (one-time check)

Open the ZIP and confirm it shows (at top level):

application.py

requirements.txt

Procfile

If you see a folder inside the ZIP, that is wrong.

| flask-ddb-lab.zip |                |               |                 |              |      |       |
|-------------------|----------------|---------------|-----------------|--------------|------|-------|
|                   | Name           | Type          | Compressed size | Password ... | Size | Ratio |
| Home              | application.py | Python.File   | 1 KB            | No           | 2 KB | 63%   |
| Gallery           | Procfile       | File          | 1 KB            | No           | 1 KB | 0%    |
| OneDrive          | requirements   | Text Document | 1 KB            | No           | 1 KB | 7%    |
| Desktop           | runtime        | Text Document | 1 KB            | No           | 1 KB | 0%    |

## Phase C — Deploy on Elastic Beanstalk (AWS Console)

### Step C1: Open Elastic Beanstalk

AWS Console → Search Elastic Beanstalk → Open

Ensure region is Asia Pacific (Mumbai) (ap-south-1)

### Step C2: Create Environment

Click Create environment

Environment tier - Select: Web server environment

#### Application information

- Application name: EB-Flask-Lab (or any name)

#### Environment information

- Environment name: EB-Flask-Lab-env (or any name)
- Domain: leave blank (auto)

## **Platform**

- Platform: Python
- Platform branch: latest Python on Amazon Linux
- Platform version: recommended

## **Step C3: Upload your code (IMPORTANT SETTINGS)**

In Application code:

1. Select: Upload your code
2. Version label: type v1
3. Source code origin: Local file  
(Do NOT choose “Public S3 URL”)
4. Click Choose file and select:
5. eb-flask-lab.zip

Presets

Select: Single instance (free tier eligible)

Click Next

## **Phase D — Configure Service Access (IAM Roles)**

On Configure service access page:

### **Step D1: Service role**

If dropdown shows “No options”:

- Click Create role (opens IAM)
  - Use case: Elastic Beanstalk
- 
- Keep default permissions (AWS auto selects)
  - Create role name:  
aws-elasticbeanstalk-service-role

Return to EB tab → click Refresh → select this role.

### **Step D2: EC2 instance profile**

If dropdown shows “No options”:

- Click Create role
- Use case: EC2
- Attach policy: AWSElasticBeanstalkWebTier
- Role name: aws-elasticbeanstalk-ec2-role

Return to EB tab → refresh → select this role.

### **Step D3: EC2 Key pair**

Leave blank (optional)

Click Next

Leave remaining pages as default → Create environment

**Platform software**

|                       |                      |
|-----------------------|----------------------|
| <b>Lifecycle</b>      | <b>Log streaming</b> |
| false                 | Disabled             |
| <b>NumProcesses</b>   | <b>NumThreads</b>    |
| 1                     | 15                   |
| <b>WSGIPath</b>       | <b>Proxy server</b>  |
| application           | nginx                |
| <b>Logs retention</b> | <b>Rotate logs</b>   |
| 7                     | Disabled             |
| <b>Update level</b>   | <b>X-Ray enabled</b> |
| minor                 | Disabled             |

**Environment properties**

| Source     | Key        | Value                  |
|------------|------------|------------------------|
| Plain text | PYTHONPATH | /var/app/venv/stagi... |

**Create**

## Part E — Wait for Deployment

### Step E1: Monitor

Open Events tab.

Final success condition:

- Green message: Environment successfully launched
- Health: OK
- Domain URL appears

## Phase F — Test the Application

### Step F1: Open the domain URL

Click the Domain link shown in EB.

Expected output: Hello from Flask on Elastic Beanstalk!

## **Step F2: Test /health**

In the browser address bar, append /health

Example: <http://<your-domain>.elasticbeanstalk.com/health>

Expected output: OK

A screenshot of a web browser displaying a JSON response. The URL in the address bar is "Not secure eb-flask-lab-env.eba-z7kj8y3q.ap-south-1.elasticbeanstalk.com/health". A "Pretty-print" checkbox is checked. The JSON content is: { "status": "ok" }.

## **Phase G — Cleanup (must do after lab)**

### **Step G1: Terminate Environment**

Elastic Beanstalk → Environment

**Actions → Terminate environment**

(Optional) After termination: Delete application if required.

### **Common Mistakes (quick checklist)**

Procfile must be Procfile (no .txt)

ZIP must contain files at root level (not folder)

Upload must be **Local file** (not Public S3 URL)

IAM roles must be created/selected if “No options”

#### 1. What is Elastic Beanstalk?

Elastic Beanstalk is a Platform as a Service (PaaS) provided by AWS that allows developers to deploy and manage applications without manually handling the underlying infrastructure such as EC2, load balancers, or auto scaling.

The user only uploads the application code, and Elastic Beanstalk automatically:

- Creates required AWS resources
- Deploys the application
- Handles scaling, monitoring, and health checks

#### 2. What does “Single instance environment” mean?

A Single instance environment means the application runs on one EC2 instance only, without a load balancer or auto scaling.

It is mainly used for:

- Learning and lab exercises
- Development and testing
- Low-traffic applications

It is cost-effective and simple, but not fault-tolerant.

### 3. Why do we need requirements.txt?

The requirements.txt file lists all Python libraries and their versions required for the application.

Elastic Beanstalk uses this file to:

- Automatically install dependencies using pip
- Ensure the application runs consistently across environments

Without requirements.txt, the application may fail due to missing modules.

### 4. What is Procfile used for?

A Procfile tells Elastic Beanstalk how to start the application.

It specifies:

- The process type (e.g., web)
- The command to run the application (e.g., Gunicorn for Flask)

Example:

```
web: gunicorn application:app
```

Without a Procfile, Elastic Beanstalk may not know which command to execute, leading to deployment errors.

### 5. What happens if you zip the parent folder instead of the files?

If the parent folder is zipped, Elastic Beanstalk cannot locate key files like:

- application.py
- requirements.txt
- Procfile

As a result:

- Deployment fails
- Application shows errors such as “*Application version not found*” or *502 Bad Gateway*

Elastic Beanstalk expects all required files at the root level of the ZIP file.

**DATE: 10-12-25**

## **Exercise-21:** Deploy a Flask App on AWS Elastic Beanstalk and Perform CRUD on DynamoDB (Using AWS SDK/boto3)

**Deploy a simple Python Flask web app on AWS Elastic Beanstalk (EB) that creates, reads, updates, and deletes items in DynamoDB using AWS APIs via boto3.**

### **A) LOCALLY — Prepare Folder + Code (Before AWS)**

#### **1) Create folder**

Create a folder named: flask-ddb-lab

#### **2) Inside it, create 4 files**

Your folder must look like:

```
flask-ddb-lab/
 application.py
 requirements.txt
 Procfile
 runtime.txt
```

#### **3) Paste code into application.py**

```
from flask import Flask, request, jsonify
import boto3
import os

app = Flask(__name__)

REGION = os.getenv("AWS_REGION", "ap-south-1")
TABLE_NAME = os.getenv("TABLE_NAME", "Students")

dynamodb = boto3.resource("dynamodb", region_name=REGION)
table = dynamodb.Table(TABLE_NAME)

@app.route("/")
def home():
 return "Flask + DynamoDB on Elastic Beanstalk is working!"

@app.route("/health")
def health():
 return jsonify(status="ok")

CREATE
@app.route("/student", methods=["POST"])
def create_student():
 data = request.get_json()
 table.put_item(Item={
```

```

 "StudentID": data["StudentID"],
 "Name": data["Name"],
 "Dept": data.get("Dept", ""))
 })
return jsonify(message="Student created"), 201

READ
@app.route("/student/<student_id>", methods=["GET"])
def get_student(student_id):
 resp = table.get_item(Key={"StudentID": student_id})
 item = resp.get("Item")
 if not item:
 return jsonify(error="Student not found"), 404
 return jsonify(item)

UPDATE
@app.route("/student/<student_id>", methods=["PUT"])
def update_student(student_id):
 data = request.get_json()
 resp = table.update_item(
 Key={"StudentID": student_id},
 UpdateExpression="SET #n = :n, Dept = :d",
 ExpressionAttributeNames={"#n": "Name"},
 ExpressionAttributeValues={":n": data["Name"], ":d": data.get("Dept", "")},
 ReturnValues="UPDATED_NEW"
)
 return jsonify(message="Student updated", updated=resp["Attributes"])

DELETE
@app.route("/student/<student_id>", methods=["DELETE"])
def delete_student(student_id):
 table.delete_item(Key={"StudentID": student_id})
 return jsonify(message="Student deleted")

```

#### **4) requirements.txt**

```
Flask==3.0.0
boto3==1.34.0
gunicorn==21.2.0
```

#### **5) Procfile**

```
web: gunicorn application:app
```

#### **6) runtime.txt**

```
python-3.11
```

## 7) Zip it

Select these 4 files **together** → right click → **Send to** → **Compressed (zipped) folder**.

Rename the zip: flask-ddb-lab.zip

When you open the zip, you must directly see:

- application.py
- requirements.txt
- Procfile
- runtime.txt

(No extra folder inside.)

## B) AWS — DynamoDB Table

### Create table

AWS Console → **DynamoDB** → Tables → **Create table**

- Table name: Students
  - Partition key: StudentID (String)
- Create → wait till **ACTIVE**

## C) AWS — IAM Role (for EB EC2 to access DynamoDB)

### Create role

AWS Console → **IAM** → Roles → **Create role**

- Trusted entity: **AWS service**
  - Use case: **EC2**
  - Attach policy: **AmazonDynamoDBFullAccess**
  - Role name: EB-EC2-DynamoDB-Role
- Create role

[once role is created, for next time it will be automatically selected. No need to create again]

The screenshot shows the AWS IAM 'Create role' wizard at Step 3: Name, review, and create. The role name is 'EB-EC2-SynamoDB-Role' and the description is 'Allows EC2 instances to call AWS services on your behalf.'

**Role details**

**Role name**  
Enter a meaningful name to identify this role.  
**EB-EC2-SynamoDB-Role**

Maximum 64 characters. Use alphanumeric and '+,-,\_' characters.

**Description**  
Add a short explanation for this role.  
**Allows EC2 instances to call AWS services on your behalf.**

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: \_+=,. @-/\[{}!#\$%^&\*();`

**Step 1: Select trusted entities** [Edit](#)

**Trust policy**

```
1 [{}
2 "Version": "2012-10-17",
3 "Statement": [
4 {
5 "Effect": "Allow",
6 "Principal": "
7 \"arn:aws:iam::
8 user:
9 ${AWS_ARN}
10 }
```

CloudShell Feedback [Console Mobile App](#) Privacy Terms Cookie preferences © 2025, Amazon Web Services, Inc. or its affiliates.

#### D) AWS — Elastic Beanstalk Environment (Sample app first)

##### Create EB app/environment

AWS Console → **Elastic Beanstalk** → **Create application**

- Environment tier: **Web server environment**
- Application name: flask-ddb-lab
- Platform: **Python**
- Application code: **Sample application**
- Presets: **Single instance (free tier eligible)**

##### Configure service access

On **Configure service access** page:

- Service role: leave default
  - **EC2 instance profile: select EB-EC2-DynamoDB-Role**
- Next → Next → Review → **Create environment**

Wait until you can open the EB domain and see sample page.

Environment successfully launched.

**Flask-ddb-lab-env**

**Environment overview**

Health: Pending

Domain: Flask-ddb-lab-env.eba-jtqiug36.ap-south-1.elasticbeanstalk.com

Environment ID: e-spixhdqwpf

Application name: flask-ddb-lab

**Platform**

Platform: Python 3.14 running on 64bit Amazon Linux 2023/4.9.0

Running version: -

Platform state: Supported

**Events** (10) **Info**

Time: December 30, 2025 10:17:40 (UTC+5:50) Type: INFO Details: Successfully launched environment: Flask-ddb-lab-env

Click on Domain: [Flask-ddb-lab-env.eba-jtqiug36.ap-south-1.elasticbeanstalk.com](https://flask-ddb-lab-env.eba-jtqiug36.ap-south-1.elasticbeanstalk.com)

AWS Elastic Beanstalk

Welcome to Your Elastic Beanstalk Application

Congratulations! Your Python application is now running on your own dedicated environment in the AWS Cloud.

Learn More

Benefits of AWS Elastic Beanstalk

Discover why thousands of developers rely on AWS Elastic Beanstalk to deploy and manage their applications.

The screenshot shows the AWS Elastic Beanstalk Events page for the environment 'Flask-ddb-lab-env-1'. The left sidebar shows navigation options like Applications, Environments, and Environment details for 'flask-ddb-lab'. The main content area displays a table of events with columns for Time, Type, and Details. There are 10 events listed, all of which are INFO type. The details column provides specific log messages for each event, such as 'Successfully launched environment: Flask-ddb-lab-env-1' and 'Added instance [i-0a31e37772ebb8107] to your environment'. The interface includes a search bar at the top and pagination controls at the bottom.

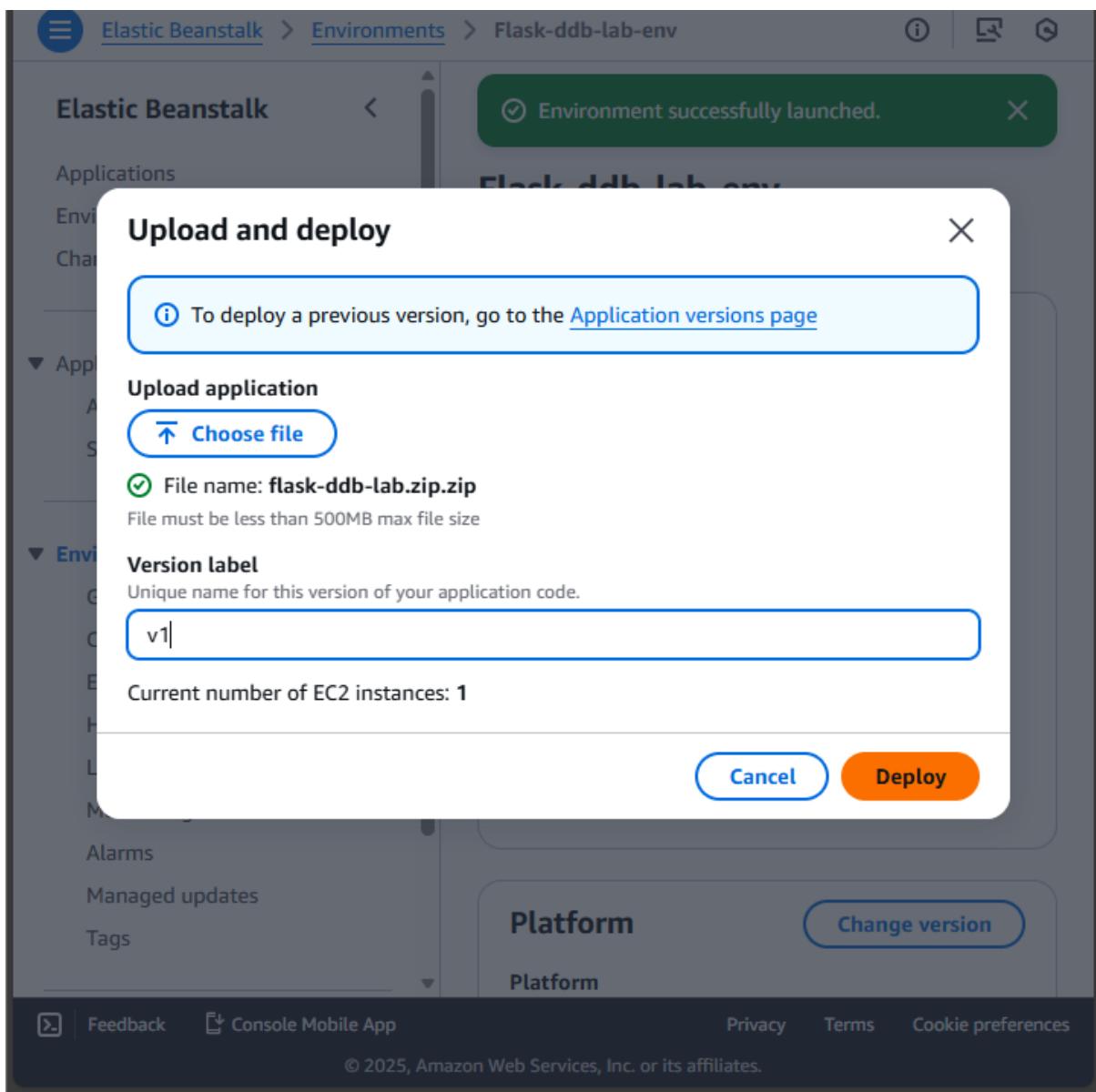
| Time                                  | Type | Details                                                                                                                     |
|---------------------------------------|------|-----------------------------------------------------------------------------------------------------------------------------|
| December 30, 2025 10:40:56 (UTC+5:30) | INFO | Successfully launched environment: Flask-ddb-lab-env-1                                                                      |
| December 30, 2025 10:40:05 (UTC+5:30) | INFO | Added instance [i-0a31e37772ebb8107] to your environment.                                                                   |
| December 30, 2025 10:39:49 (UTC+5:30) | INFO | Instance deployment completed successfully.                                                                                 |
| December 30, 2025 10:39:45 (UTC+5:30) | INFO | Instance deployment successfully generated a 'Procfile'.                                                                    |
| December 30, 2025 10:38:45 (UTC+5:30) | INFO | Waiting for EC2 instances to launch. This may take a few minutes.                                                           |
| December 30, 2025 10:38:29 (UTC+5:30) | INFO | Created EIP: 13.202.250.197                                                                                                 |
| December 30, 2025 10:38:14 (UTC+5:30) | INFO | Created security group named: awseb-e-d6qv4shuib-stack-AWSEBSecurityGroup-gsvglUeK5Bbj                                      |
| December 30, 2025 10:38:06 (UTC+5:30) | INFO | Environment health has transitioned to Pending. Initialization in progress (running for 9 seconds). There are no instances. |
| December 30, 2025 10:37:52 (UTC+5:30) | INFO | Using elasticbeanstalk-ap-south-1-749715477960 as Amazon S3 storage bucket for environment data.                            |
| December 30, 2025 10:37:52 (UTC+5:30) | INFO | createEnvironment is starting.                                                                                              |

## E) AWS — Deploy Your ZIP

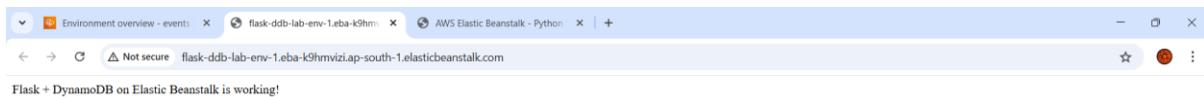
### Upload and deploy

EB Environment → Upload and deploy

- Upload: flask-ddb-lab.zip
  - Version label: v1
- Click **Deploy**



After deploy, opening the EB domain should show:  
“Flask + DynamoDB on Elastic Beanstalk is working!”



## F) Add Environment Variables in Elastic Beanstalk (MANDATORY)

### Step 1

Go to: AWS Console → Elastic Beanstalk → Environments → flask-ddb-lab-env

## Step 2

Click: Configuration

## Step 3

Under **Software**, click: Edit

## Step 4

Scroll to **Environment properties**

Add these two entries:

| Name       | Value      |
|------------|------------|
| TABLE_NAME | Student    |
| AWS_REGION | ap-south-1 |

**Environment properties**  
Use environment properties as either plain text environment variables or as references to secret ARN values that are stored in AWS Secrets Manager or AWS Systems Manager (SSM) Parameter Store. When using secrets, update your Elastic Beanstalk EC2 instance profile role with permissions to your secrets and parameters. [Learn more](#)

| Source     | Name       | Value                              | Remove |
|------------|------------|------------------------------------|--------|
| Plain text | PYTHONPATH | /var/app/venv/staging-LQM1lest/bin | Remove |
| Plain text | TABLE_NAME | Student                            | Remove |
| Plain text | AWS_REGION | ap-south-1                         | Remove |

[Add environment property](#)

[Cancel](#) [Continue](#) [Apply](#) [Activate Windows](#) [Go to Settings to activate Windows](#)

[CloudShell](#) [Feedback](#) [Console Mobile App](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

## Step 5

Click: Apply

## Step 6

Wait for **Environment update** to complete

(Status may show **Updating / Pending** — wait until it finishes)

## G) Verify Application After Env Update

Open the **EB Environment URL** in browser:

<http://<your-eb-domain>/>

Expected output: Flask + DynamoDB on Elastic Beanstalk is working!

## H) Test Health Endpoint (GET)

### Step 1

Open **AWS CloudShell** (or Command Prompt / PowerShell)

### Step 2

Run: curl http://<your-eb-domain>/health

Expected output: { "status": "ok" }

```
Use the --UseBasicParsing switch to avoid script code execution.

Do you want to continue?

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y

StatusCode : 200
StatusDescription : OK
Content : Flask + DynamoDB on Elastic Beanstalk is working!
RawContent : HTTP/1.1 200 OK
 Connection: keep-alive
 Content-Length: 49
 Content-Type: text/html; charset=utf-8
 Date: Tue, 30 Dec 2025 07:18:43 GMT
 Server: nginx

Forms : {}
Headers : {[Connection, keep-alive], [Content-Length, 49], [Content-Type, text/html; charset=utf-8], [Date, Tue, 30 Dec 2025 07:18:43 GMT]...}
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 49
```

## AWS CloudShell - Steps

1. In AWS Console (top right)
2. Search **CloudShell** in the search bar
3. Wait until terminal opens (you'll see \$ prompt)
4. Paste the following command (replace domain):

## I) CREATE Item in DynamoDB (POST)

### Step 1

Run:

```
curl -X POST http://Flask-ddb-lab-env-1.eba-k9hmviz.ap-south-1.elasticbeanstalk.com /Student \
-H "Content-Type: application/json" \
-d '{ "StudentID": "101", "Name": "Anita", "Dept": "MCA" }'
```

Expected output:

```
{"message":"Student created"}
```

#### **J) READ Item from DynamoDB (GET)**

Run:

```
curl http://<your-eb-domain>/student/101
```

Expected output (JSON with student data).

#### **K) UPDATE Item (PUT)**

Run:

```
curl -X PUT http://<your-eb-domain>/student/101 \
-H "Content-Type: application/json" \
-d '{"Name":"Anita S","Dept":"MCA-III"}'
```

Expected output:

```
{"message":"Student updated"}
```

#### **L) DELETE Item (DELETE)**

Run:

```
curl -X DELETE http://<your-eb-domain>/student/101
```

Expected output:

```
{"message":"Student deleted"}
```

#### **M) Verify in DynamoDB Console**

Go to:

AWS Console → DynamoDB → Tables → Students → Explore table items

Confirm record creation / update / deletion.

**DATE: 12-12-25**

**Exercise-22:** CloudFormation – Launch EC2 (Amazon Linux 2023) + Install Apache using UserData

Create an EC2 instance using AWS CloudFormation (YAML template) and automatically install Apache web server (httpd) using UserData, then verify the website from a browser.

#### Part A — Create Key Pair (One-time setup)

##### Step A1: Open EC2 Key Pairs

1. AWS Console → Search EC2
2. Left menu → Key Pairs (under “Network & Security”)
3. Click Create key pair

##### Step A2: Create key pair

- Name: pemkeypair (any name is fine)
- Key pair type: RSA
- Private key file format: .pem

Click Create key pair

The screenshot shows the AWS EC2 console with the 'Key pairs' section selected. A green success message at the top right says 'Successfully created key pair'. Below it is a table titled 'Key pairs (4)'. The table has columns for Name, Type, Created, Fingerprint, and ID. The rows show four key pairs: 'web', 'pemkeypair', 'nist', and 'ksa'. The 'pemkeypair' row is highlighted. The left sidebar shows other AWS services like AMIs, Volumes, and Network & Security.

| Name       | Type | Created                   | Fingerprint                                  | ID           |
|------------|------|---------------------------|----------------------------------------------|--------------|
| web        | rsa  | 2025/11/25 17:30 GMT+5:30 | 03:7cec:0e:32:b6:b6:16:fa:ed:09:bb:22:8...   | key-07341... |
| pemkeypair | rsa  | 2026/01/02 09:48 GMT+5:30 | 41:31:c8:6a:e3:8d:c0:a6:70:71:2b:7b:7e:7...  | key-0fa2a... |
| nist       | rsa  | 2025/12/12 09:35 GMT+5:30 | 2a:57:6e:06:eb:99:7e:9e:79:08:dc:ea:67:e...  | key-08b33... |
| ksa        | rsa  | 2025/11/28 09:19 GMT+5:30 | 8f:13:07:81:8a:04:cf:b5:e7:68:66:a9:78:bc... | key-05b04... |

A file will download like: pemkeypair.pem

**Note:** CloudFormation uses key pair NAME (example: pemkeypair), not the file name extension.

#### Part B — Create CloudFormation Template File (Amazon Linux 2023 + Apache)

##### Create YAML file on your computer

- Open Notepad
- Paste the full YAML template given below
- Save as: ec2-apache-al2023.yaml
  1. Save type: All files
  2. Encoding: UTF-8 (if asked)

## Full CloudFormation Template (Amazon Linux 2023)

Important:

- Do not add .pem anywhere.
- You will select Key Pair from dropdown during stack creation later on.

```
AWSTemplateFormatVersion: "2010-09-09"
```

```
Description: Launch EC2 (Amazon Linux 2023) and install Apache
```

Parameters:

KeyName:

Type: AWS::EC2::KeyPair::KeyName

Description: Select an existing EC2 Key Pair

AmazonLinux2023AMI:

Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>

Default: /aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86\_64

Resources:

WebServerSG:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Allow SSH and HTTP

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 22

ToPort: 22

CidrIp: 0.0.0.0/0

- IpProtocol: tcp

FromPort: 80

ToPort: 80

CidrIp: 0.0.0.0/0

WebServerInstance:

Type: AWS::EC2::Instance

Properties:

InstanceType: t3.micro

KeyName: !Ref KeyName

ImageId: !Ref AmazonLinux2023AMI

SecurityGroupIds:

- !Ref WebServerSG

UserData:

Fn::Base64: |

#!/bin/bash

dnf update -y

```
dnf install -y httpd
systemctl enable httpd
systemctl start httpd
echo "<h1>Apache Installed via CloudFormation (Amazon Linux 2023)</h1>" >
/var/www/html/index.html
```

Outputs:

WebsiteURL:

Description: Apache Website URL

Value: !Sub "http://\${WebServerInstance.PublicDnsName}"

## Part C — Create CloudFormation Stack (Console Steps)

### Step C1: Open CloudFormation

- AWS Console → Search CloudFormation
- Click Stacks
- Click Create stack → With new resources (standard)

### Step C2: Prepare template (select correct options)

- Under Prepare template:  
Select Choose an existing template
- Under Template source:  
Select Upload a template file
- Click Choose file → select ec2-apache-al2023.yaml
- Click Next

## Part D — Specify Stack Details

### Step D1: Stack name

- Stack name: EC2-Apache-AL2023

Click Next

### Step D2: Parameters

- Under **KeyName**, select your key pair name from dropdown [Example: pemkeypair]

Click Next

## Part E — Configure Stack Options (keep default)

- Leave everything as default
- Click Next

## Part F — Review and Create

- Scroll down
- Click **Create stack**

## Part G — Monitor Stack Creation

- Wait for Stack status to become: CREATE\_COMPLETE
- Open the stack → click Outputs tab
- Copy WebsiteURL
- Paste URL in browser

Expected Output in browser:

**Apache Installed via CloudFormation UserData (Amazon Linux 2023)!**

| Key        | Value                                           | Description        | Export name |
|------------|-------------------------------------------------|--------------------|-------------|
| Instanceld | i-007f400c15f76eeaa                             | EC2 Instance ID    | -           |
| WebsiteURL | http://ec2-98-92-41-106.compute-1.amazonaws.com | Apache Website URL | -           |



## Part H — Verify in EC2

Go to **EC2 → Instances**

Instance should be running

Security group should allow:

- SSH 22
- HTTP 80

## Troubleshooting (Common Errors)

**Website not opening**

Check:

- EC2 instance status checks are 2/2 passed
- Security group has port 80 open
- Wait 2–3 minutes (UserData takes time)

### **Stack went to ROLLBACK**

Go to stack → Events tab → check the failure reason

Most common reason: Wrong Key Pair selected / key pair does not exist

### **Clean-up**

To avoid charges:

1. CloudFormation → Stacks
2. Select EC2-Apache-AL2023
3. Click Delete
4. Confirm

This deletes EC2 + Security Group automatically.

**DATE: 17-12-25**

**Exercise-23:** EC2 + S3 (Static Content Pulled from S3 using CloudFormation)

Create an **S3 bucket** using CloudFormation.

Upload a static HTML file (index.html) to S3.

Launch **EC2 (Amazon Linux 2023) + Apache** using CloudFormation.

EC2 will **pull index.html from S3** and host it via Apache.

### **Pre-requisites**

- Region: **Mumbai (ap-south-1)**
- A Key Pair exists (example: pemkeypair)

### **One-time: Create Key Pair**

EC2 → Key Pairs → Create key pair

- Name: pemkeypair
- Format: .pem  
Download it and keep safe.

## **PART 1 — Stack-1: Create S3 Bucket (CloudFormation)**

### **Step 1.1: Create S3 template file**

Create a file: **stack1-s3-bucket.yaml** and paste:

AWSTemplateFormatVersion: "2010-09-09"

Description: Create an S3 bucket to store website content (index.html)

Resources:

WebsiteBucket:

Type: AWS::S3::Bucket

Outputs:

BucketName:

Description: S3 Bucket Name (use this to upload index.html)

Value: !Ref WebsiteBucket

## Step 1.2: Create stack

CloudFormation → Stacks → Create stack → With new resources

- Choose an existing template
- Upload a template file → stack1-s3-bucket.yaml
- Stack name: S3-Website-Bucket
- Create stack

## Step 1.3: Copy bucket name

Open stack → Outputs → copy BucketName

| Key        | Value                                        | Description                                    | Export name |
|------------|----------------------------------------------|------------------------------------------------|-------------|
| BucketName | s3-website-bucket-websitebucket-zwwwwezz2fh7 | S3 Bucket Name (use this to upload index.html) | -           |

## PART 2 — Upload Static Content to S3 (Manual)

### Step 2.1: Create index.html on your PC

Create a file named **index.html** with this content:

```
<!DOCTYPE html>

<html>
<head>
<title>EC2 + S3 Demo</title>
</head>
<body>
<h1>Hello! This page was pulled from S3 to EC2 automatically.</h1>
<p>Deployed using CloudFormation + UserData</p>
</body>
</html>
```

### Step 2.2: Upload to S3 bucket

S3 → Buckets → open your bucket (from Output) → Upload

- Upload **index.html**
- Keep it at **root** (no folder)
- Upload

Now S3 has: s3://<your-bucket-name>/index.html

## PART 3 — Stack-2: Launch EC2 + Apache + Pull from S3

This stack will:

- Create IAM Role (permission to read the bucket)
- Create Security Group (22, 80)
- Launch EC2 (Amazon Linux 2023)
- Install Apache
- Copy index.html from S3 to /var/www/html/index.html

### Step 3.1: Create EC2 template file

Create a file: **stack2-ec2-pull-from-s3.yaml** and paste:

```
AWSTemplateFormatVersion: "2010-09-09"
Description: Launch EC2 (Amazon Linux 2023), install Apache, and pull index.html from S3
```

Parameters:

KeyName:

Type: AWS::EC2::KeyPair::KeyName

Description: Select an existing EC2 Key Pair to enable SSH access

BucketName:

Type: String

Description: Enter the S3 bucket name created in Stack-1 (Output)

SubnetId:

Type: AWS::EC2::Subnet::Id

Description: Select a PUBLIC subnet (default VPC public subnet recommended)

VpcId:

Type: AWS::EC2::VPC::Id

Description: Select the VPC (default VPC recommended)

Resources:

WebServerRole:

Type: AWS::IAM::Role  
Properties:  
    AssumeRolePolicyDocument:  
        Version: "2012-10-17"  
        Statement:  
            - Effect: Allow  
                Principal:  
                    Service: ec2.amazonaws.com  
                    Action: sts:AssumeRole  
    Policies:  
        - PolicyName: S3ReadOnlyForWebsiteBucket  
            PolicyDocument:  
                Version: "2012-10-17"  
                Statement:  
                    - Effect: Allow  
                        Action:  
                            - s3:GetObject  
                            - s3>ListBucket  
                Resource:  
                    - !Sub "arn:aws:s3:::\${BucketName}"  
                    - !Sub "arn:aws:s3:::\${BucketName}/\*"

WebServerInstanceProfile:  
Type: AWS::IAM::InstanceProfile  
Properties:  
    Roles:  
        - !Ref WebServerRole

WebServerSG:  
Type: AWS::EC2::SecurityGroup  
Properties:  
    GroupDescription: Allow SSH (22) and HTTP (80)  
    VpcId: !Ref VpcId  
    SecurityGroupIngress:  
        - IpProtocol: tcp  
            FromPort: 22  
            ToPort: 22  
            CidrIp: 0.0.0.0/0  
        - IpProtocol: tcp  
            FromPort: 80  
            ToPort: 80  
            CidrIp: 0.0.0.0/0

WebServerInstance:  
Type: AWS::EC2::Instance  
Properties:

```

InstanceType: t3.micro
KeyName: !Ref KeyName
SubnetId: !Ref SubnetId
SecurityGroupIds:
- !Ref WebServerSG
IamInstanceProfile: !Ref WebServerInstanceProfile

Amazon Linux 2023 AMI for Mumbai (ap-south-1)
ImageId: ami-02b49a24cfb95941c

UserData:
Fn::Base64: !Sub |
#!/bin/bash
dnf update -y
dnf install -y httpd
systemctl enable httpd
systemctl start httpd

pull index.html from S3 to Apache web root
aws s3 cp s3://${BucketName}/index.html /var/www/html/index.html

systemctl restart httpd

Outputs:
WebsiteURL:
Description: Open this URL in browser
Value: !Sub "http://${WebServerInstance.PublicDnsName}"

```

### Step 3.2: Create stack

CloudFormation → Stacks → Create stack

- Upload template → stack2-ec2-pull-from-s3.yaml
- Stack name: EC2-Pull-S3-Website
- Parameters:
  - **KeyName:** select your key pair (e.g., pemkeypair)
  - **BucketName:** paste bucket name from Stack-1 Output
  - **VpcId:** select **default VPC**
  - **SubnetId:** select a **PUBLIC subnet** in default VPC  
(usually the subnet name shows “public” or you can pick any default subnet that gives public IP; default subnets generally work)

Create stack → wait for **CREATE\_COMPLETE**

The screenshot shows the AWS CloudFormation console with the stack named "EC2-Pull-S3-Website". The "Outputs" tab is active, showing a single output named "WebsiteURL" with the value "http://ec2-13-233-104-250.ap-south-1.compute.amazonaws.com". The status of the stack is "CREATE\_COMPLETE".

## PART 4 — Verify Output

Stack-2 → **Outputs** → copy **WebsiteURL** → open in browser.

Expected page:

“Hello! This page was pulled from S3 to EC2 automatically...”

The screenshot shows a web browser displaying the website URL "http://ec2-13-233-104-250.ap-south-1.compute.amazonaws.com". The page content is "Hello! This page was pulled from S3 to EC2 automatically.". Below the browser window, it says "Deployed using CloudFormation + UserData".

## Troubleshooting (comm

### 1) Website not opening

- Wait 1–2 minutes (UserData takes time)
- Ensure Security Group has **HTTP 80 open**
- Ensure you selected a **public subnet** (needs internet to reach S3)

### 2) Stack-2 fails with S3 access error

- BucketName typed wrong
- index.html not uploaded at root of bucket

## Clean-up

### Delete Stack-2 first

CloudFormation → select EC2-Pull-S3-Website → Delete

### **Empty the S3 bucket**

S3 → bucket → delete index.html (empty bucket)

### **Delete Stack-1**

CloudFormation → select S3-Website-Bucket → Delete

### **Viva questions**

1. What is the purpose of **UserData** in EC2?
2. Why do we need an **IAM Role + Instance Profile**?
3. Why is S3 bucket not made public in this lab?
4. What happens when a CloudFormation stack is deleted?
5. Which ports are required for web hosting and SSH?

**DATE: 19-12-25**

**Exercise-24:** AWS Lambda: Input Processing, Business Logic Execution, and CloudWatch Logging

Create one Lambda function that:

- prints a welcome message
- reads JSON input event
- performs business logic (grade calculation)
- writes logs → view them in CloudWatch Logs

### **Create the Lambda Function**

#### **Step 1: Open Lambda**

AWS Console → Search Lambda → Open AWS Lambda

#### **Step 2: Create function**

Click Create function

- Select - Author from scratch
- Function name: StudentGradeLogger
- Runtime: Python 3.11
- Architecture: x86\_64 (default)

#### **Step 3: Permissions (Execution Role)**

Under “Permissions”

- Choose: Create a new role with basic Lambda permissions

This automatically gives permission to write logs to CloudWatch.

Click **Create function**

### **Add Code (Business Logic + Logs)**

#### **Step 4: Paste this code**

In **Code** tab → `lambda_function.py` → paste and **Deploy**

```
import json

def lambda_handler(event, context):
 # Welcome message
 print("Lambda invoked successfully")

 # Read input from event
 student_name = event["StudentName"]
 marks = event["Marks"]

 # Business logic: grade calculation
 if marks >= 75:
 result = "Pass"
 grade = "A"
```

```

else:
 result = "Fail"
 grade = "F"

Log output
print("Student:", student_name)
print("Marks:", marks)
print("Grade:", grade)
print("Result:", result)

Return response
return {
 "statusCode": 200,
 "body": json.dumps({
 "StudentName": student_name,
 "Marks": marks,
 "Grade": grade,
 "Result": result
 })
}

```

Click **Deploy**.

### **Test with Input Events (JSON)**

#### **Step 5: Create a test event (Valid case)**

Go to Test (top right) → Configure test event

- Event name: ValidInput
- Paste this JSON:

```
{
 "StudentName": "Anita",
 "Marks": 86
}
```

Click **Save**

#### **Step 6: Run test**

Click **Test**

Expected response (sample):

- statusCode: 200
- body will include Grade A and Result Pass

Executing function: succeeded ([logs ↗](#))

▼ Details

```
{
 "statusCode": 200,
 "body": "{\"StudentName\": \"Anita\", \"Marks\": 86, \"Grade\": \"A\", \"Result\": \"Pass\"}"
}
```

Summary

Code SHA-256

R6fL2f5y9xHPTo4NojRtY2hz04DIAxWgwXm4xSMnwRQ=

Execution time

1 second ago

### Step 7: Test invalid input (Missing Marks)

Create another test event:

- Event name: MissingMarks

```
{
 "StudentName": "Rahul"
}
```

Run Test

Expected:

- statusCode: 400
- message: Missing 'Marks'

### Step 8: Test invalid marks range

Event name: InvalidMarks

```
{
 "StudentName": "Priya",
 "Marks": 150
}
```

Expected:

- statusCode: 400
- message: Marks must be 0–100

[View Logs in CloudWatch](#)

### Step 9: Open logs from Lambda directly

On the Lambda function page:

- Go to **Monitor** tab
- Click **View CloudWatch logs**

You will see:

- Log group: /aws/lambda/StudentGradeLogger
- Open latest **log stream**
- You should see all print() outputs:
  - “Lambda invoked successfully!”
  - “Received event...”
  - grade/result logs

- o error logs for invalid tests

The screenshot shows the AWS CloudWatch Log Management interface. The left sidebar navigation includes CloudWatch, Favorites and recents, Ingestion, Dashboards, Alarms, AI Operations, GenAI Observability, Application Signals (APM), Infrastructure Monitoring, Logs, Log Management (New), and Log Anomalies. The main content area is titled "Log events" and displays a table of log messages. The table has two columns: "Timestamp" and "Message". The log entries are:

| Timestamp                | Message                                                                                                                                  |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| 2026-01-20T16:21:01.589Z | INIT_START Runtime Version: python:3.14.v32 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:lee4e6d61a50fbb29d03b87572cc627d0a... |
| 2026-01-20T16:21:01.777Z | START RequestId: 214ddf0f-73cf-4ad5-a4ec-ea1098bf63f1 Version: \$LATEST                                                                  |
| 2026-01-20T16:21:01.778Z | Lambda invoked successfully                                                                                                              |
| 2026-01-20T16:21:01.778Z | Student: Anita                                                                                                                           |
| 2026-01-20T16:21:01.778Z | Marks: 86                                                                                                                                |
| 2026-01-20T16:21:01.778Z | Grade: A                                                                                                                                 |
| 2026-01-20T16:21:01.778Z | Result: Pass                                                                                                                             |

At the top right of the main area, there are buttons for Actions, Start tailing, Create metric filter, and time controls (Clear, 1m, 30m, 1h, 12h, Custom, UTC timezone).

### Cleanup (Avoid charges, keep account clean)

Lambda itself usually costs nothing in small use, but cleanup is good practice:

1. Lambda → Functions → select StudentGradeLogger → **Delete**
2. CloudWatch → Log groups → find /aws/lambda/StudentGradeLogger → **Delete**
3. IAM role created (optional):
  - o IAM → Roles → search role name created for Lambda → delete (only if not used elsewhere)

**DATE: 24-12-25**

**Exercise-25:** Mini Project: Event-Driven Notification System using AWS Lambda and Amazon SNS. Simulating real-time alerts from events using serverless computing.

- An event in JSON format is given as input to an AWS Lambda function.
- The Lambda function processes the event and generates a notification message.
- The message is published to an Amazon SNS topic.
- SNS sends the notification to the subscribed email address.
- The execution details are verified using CloudWatch Logs.

### Create SNS Topic + Email Subscription

#### Step 1: Open SNS

AWS Console → Search SNS → Open Simple Notification Service

#### Step 2: Create a Topic

SNS → Topics → Create topic

- Type: Standard
- Name: NotificationTopic

Click Create topic

The screenshot shows the AWS SNS Topics page. On the left, there's a sidebar with 'Amazon SNS' at the top, followed by 'Dashboard', 'Topics' (which is selected and highlighted in blue), 'Subscriptions', and 'Mobile' which has 'Push notifications' and 'Text messaging (SMS)' under it. The main area has a header 'Topics (1)'. Below it is a search bar and a table with one row. The table has columns for 'Name', 'Type', and 'ARN'. The single row shows 'NotificationTopic' as the name, 'Standard' as the type, and 'arn:aws:sns:us-east-1:257605934444:NotificationTopic' as the ARN. At the top right of the main area, there are buttons for 'Edit', 'Delete', 'Publish message', and 'Create topic'. There are also navigation arrows and a refresh icon.

#### Step 3: Copy Topic ARN

Open the created topic → copy Topic ARN

(You will paste it into Lambda code later)

#### Step 4: Create an Email Subscription

Inside the same topic:

Click Create subscription

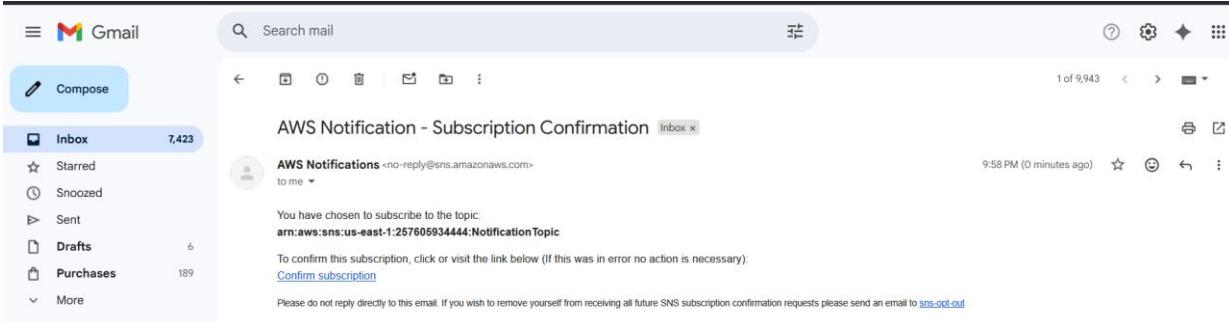
- Protocol: Email
- Endpoint: (your email id)

Click Create subscription

#### Step 5: Confirm Subscription

Open your email inbox → find AWS SNS confirmation mail → click Confirm subscription

→ Status in SNS should become Confirmed.



## Create Lambda Function

### Step 6: Open Lambda

AWS Console → Search Lambda → Open AWS Lambda

### Step 7: Create function

Click Create function

- Select: Author from scratch
  - Function name: NotificationSimulator
  - Runtime: Python 3.11
  - Permissions: Create a new role with basic Lambda permissions
- Click Create function

## Add Lambda Code (with SNS Publish)

### Step 8: Paste code in lambda\_function.py

Lambda → Code tab → open lambda\_function.py → remove existing code → paste:

Replace <SNS\_TOPIC\_ARN> with your copied Topic ARN.

```
import json
import boto3

sns = boto3.client('sns')

TOPIC_ARN = "arn:aws:sns:us-east-1:257605934444:NotificationTopic"

def lambda_handler(event, context):
 print("Notification Simulator invoked")

 event_type = event["eventType"]
 user = event["user"]

 if event_type == "ORDER_PLACED":
 message = f"Hi {user}, your order is placed successfully."
 priority = "NORMAL"

 elif event_type == "PAYMENT_FAILED":
 message = f"Hi {user}, your payment has failed. Please retry."
 priority = "HIGH"
```

```

elif event_type == "LOW_ATTENDANCE":
 message = f"Hi {user}, your attendance is low. Please take action."
 priority = "HIGH"

else:
 message = f"Hi {user}, unknown event received."
 priority = "LOW"

print("Event Type:", event_type)
print("Message:", message)
print("Priority:", priority)

Publish to SNS (Actual notification delivery)
sns.publish(TopicArn=TOPIC_ARN, Message=message, Subject=f"{event_type} [{priority}]")

return {
 "statusCode": 200,
 "body": json.dumps({
 "EventType": event_type,
 "Message": message,
 "Priority": priority
 })
}

```

Click **Deploy**

Wait for “Successfully updated function...”

### **Give Lambda Permission to Publish to SNS**

#### **Step 10: Open Lambda Execution Role**

Lambda → Configuration → Permissions

Click the Role name (execution role link)

#### **Step 11: Attach SNS Publish Policy**

IAM Role page → Add permissions → Attach policies

Attach: AmazonSNSFullAccess

Click Add permissions

Now Lambda can publish to SNS.

### **Test the Mini Project**

#### **Step 12: Create a test event**

Lambda → Test tab → Create new test event

Event name: PaymentFailed

Paste:

```
{
 "eventType": "PAYMENT_FAILED",
 "user": "Anita"
```

```
}
```

Click **Save**

### Step 13: Run Test

Click **Test**

Expected:

- Execution: **Succeeded**
- Email should arrive within a few seconds:  
Subject like: PAYMENT\_FAILED [HIGH]  
Message: "Hi Anita, your payment has failed. Please retry."

The screenshot shows the AWS Lambda Test interface. At the top, there are tabs: Code, Test (which is selected), Monitor, Configuration, Aliases, and Versions. Below the tabs, a green box indicates a successful execution: "Executing function: succeeded (logs [log](#))". Under "Details", a JSON object is shown: { "statusCode": 200, "body": "{\"Event Type\": \"PAYMENT\_FAILED\", \"Message\": \"Hi Anita, your payment has failed. Please retry.\", \"Priority\": \"HIGH\"}" }. In the "Summary" section, it says "Code SHA-256 R6fL2f5y9xHPTo4NojRtY2hz04DIAxWgwXm4xSMnwRQ=" and "Execution time 12 seconds ago".

The screenshot shows a Gmail inbox. The left sidebar lists "Compose", "Inbox 7,423", "Starred", "Snoozed", "Sent", "Drafts 6", "Purchases 189", and "More". The main area shows an email from "AWS Notifications <no-reply@sns.amazonaws.com>" with the subject "PAYMENT\_FAILED [HIGH]". The email body contains the message: "Hi Anita, your payment has failed. Please retry." Below the message, there is a unsubscribe link: <https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:257605934444:NotificationTopic:c4912bd1-190b-416b-b80a-8584a361da1d&Endpoint=joshuammz.2.7@gmail.com>. A note at the bottom of the email says: "Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>".

### Verify CloudWatch Logs

#### Step 14: View logs

Lambda → Monitor → View CloudWatch logs

Open latest log stream.

Verify these prints exist:

- Notification Simulator invoked
- Event Type:
- Message:
- Priority:

### Expected Outputs

1. Lambda test status: Succeeded
2. Email received from SNS with correct message
3. CloudWatch logs showing event type and generated message

### **Cleanup**

1. Delete Lambda function: NotificationSimulator
2. SNS → delete topic NotificationTopic (subscriptions removed automatically)
3. CloudWatch → delete log group for Lambda
4. (Optional) Delete IAM role if not used elsewhere

## **DATE: 26-12-25**

### **Exercise-26:** Analyze a CSV File in S3 Using Amazon Athena (No Server)

Upload a small CSV to **S3**, then use Athena to run SQL queries like count, average, max, group by.

#### **Exercise 26: Analyze a CSV File in S3 Using Amazon Athena (No Server)**

Upload a small CSV to **S3**, then use **Athena** to run SQL queries like **count, average, max, group by**.

#### **Part A — Create Sample CSV (on your PC)**

1. Open **Notepad**
2. Paste this data and save as: **students.csv**

StudentID,Name,Dept,Marks,Result

101,Anita,MCA,85,Pass

102,Ravi,MCA,72,Pass

103,Meera,MBA,64,Pass

104,John,MCA,35,Fail

105,Sneha,MBA,91,Pass

106,Arun,MCA,49,Fail

107,Kiran,MBA,58,Pass

108,Divya,MCA,77,Pass

#### **Part B — Create S3 Bucket and Upload CSV**

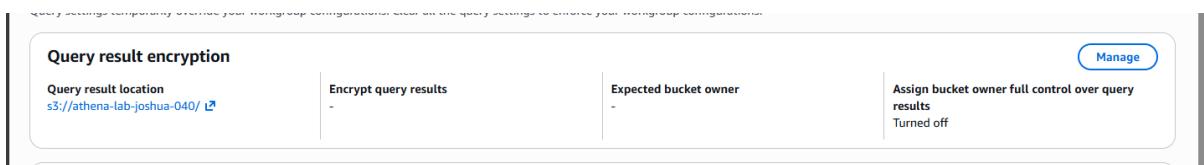
1. Go to **AWS Console → S3**
2. Click **Create bucket**
3. Bucket name: athena-lab-<yourname>-<number> (must be globally unique)
4. Keep defaults → Click **Create bucket**
5. Open the bucket → Click **Upload**
6. Upload **students.csv**
7. Click **Upload**

Now your CSV is in S3.

### Part C — Open Athena and Set Query Result Location (IMPORTANT)

1. Go to **AWS Console** → **Amazon Athena**
2. If it shows “Get started” / “Query editor”, open it.
3. It will ask to set a **Query result location**
4. Click the link/button like **Settings / Manage / Edit**
5. Set query result location to something like:
  - s3://your-bucket-name/athena-results/
6. Click **Save**

This is required so Athena can store query outputs.



### Part D — Create a Database in Athena

In Athena query editor, run:

```
CREATE DATABASE labdb;
```

Then on the left side, select:

- **Data source:** AwsDataCatalog
- **Database:** labdb

The screenshot shows the Amazon Athena Query editor interface. On the left, there's a sidebar titled 'Data' with dropdown menus for 'Data source' (set to 'AwsDataCatalog'), 'Catalog' (set to 'None'), and 'Database' (set to 'labdb'). On the right, the main area has tabs for 'Editor', 'Recent queries', 'Saved queries', and 'Query settings'. A message box at the top says: 'Athena now supports typeahead code suggestions to speed up SQL query development. Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.' Below this, a query editor window titled 'Query 1' shows a single line of SQL: '1 CREATE DATABASE labdb;'. There are also icons for running, saving, and deleting the query.

#### Part E — Create a Table for the CSV in S3

Run this (replace YOUR\_BUCKET\_NAME with your actual bucket name):

```
CREATE EXTERNAL TABLE IF NOT EXISTS students (
```

```
 StudentID int,
```

```
 Name string,
```

```
 Dept string,
```

```
 Marks int,
```

```
 Result string
```

```
)
```

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
```

```
WITH SERDEPROPERTIES (
```

```
 'separatorChar' = ',',
```

```
 'quoteChar' = "",
```

```
 'escapeChar' = '\\'
```

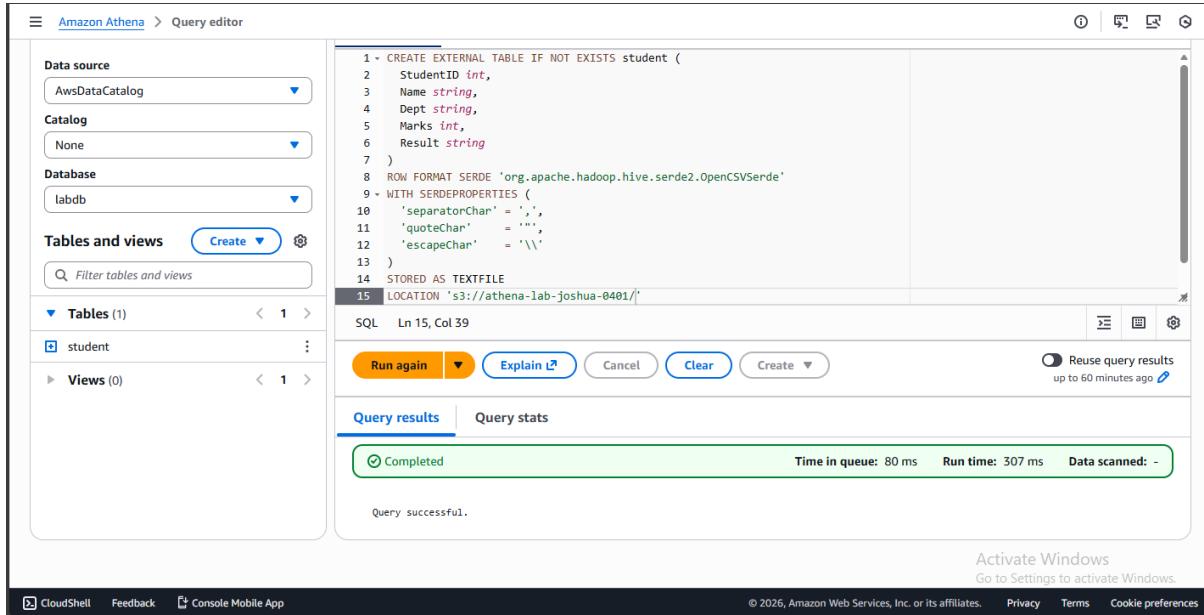
)

STORED AS TEXTFILE

LOCATION '<s3://athena-lab-josh-040/>'

TBLPROPERTIES ('skip.header.line.count'='1');

This tells Athena: "My CSV is in S3, treat it like a table."



The screenshot shows the Amazon Athena Query editor interface. On the left, the navigation pane displays the Data source (AwsDataCatalog), Catalog (None), and Database (labdb). Under Tables and views, there is one table named 'student'. The main area shows the SQL query for creating the table:

```
1 > CREATE EXTERNAL TABLE IF NOT EXISTS student (
2 StudentID int,
3 Name string,
4 Dept string,
5 Marks int,
6 Result string
7)
8 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
9 WITH SERDEPROPERTIES (
10 'separatorChar' = ',',
11 'quoteChar' = "'",
12 'escapeChar' = '\\'
13)
14 STORED AS TEXTFILE
15 LOCATION 's3://athena-lab-joshua-0401/'
```

Below the query, the status bar indicates the SQL line number (Ln 15) and column number (Col 39). The bottom section shows the Query results tab with a green status bar indicating the query is completed. The status bar also shows performance metrics: Time in queue: 80 ms, Run time: 307 ms, and Data scanned: -.

## Part F — Run Simple Analysis Queries (Core Part)

### 1) View all rows

SELECT \* FROM students;

Expected: 8 records.

**Query results**    **Query stats**

Completed    Time in queue: 150 ms    Run time: 510 ms    Data scanned: 0.21 KB

**Results (8)**    Copy    Download results CSV

| # | studentid | name  | dept | marks | result |
|---|-----------|-------|------|-------|--------|
| 1 | 101       | Anita | MCA  | 85    | Pass   |
| 2 | 102       | Ravi  | MCA  | 72    | Pass   |
| 3 | 103       | Meera | MBA  | 64    | Pass   |
| 4 | 104       | John  | MCA  | 35    | Fail   |
| 5 | 105       | Sneha | MBA  | 91    | Pass   |
| 6 | 106       | Arun  | MCA  | 49    | Fail   |
| 7 | 107       | Kiran | MBA  | 58    | Pass   |
| 8 | 108       | Divya | MCA  | 77    | Pass   |

## 2) Total students

SELECT COUNT(\*) AS total\_students FROM students;

Expected: 8

**Query results**    **Query stats**

Completed    Time in queue: 115 ms    Run time: 425 ms    Data scanned: 0.21 KB

**Results (1)**    Copy    Download results CSV

| # | total_students |
|---|----------------|
| 1 | 8              |

## 3) Pass vs Fail count

SELECT Result, COUNT(\*) AS cnt

FROM students

GROUP BY Result;

Expected (based on data): Pass = 6, Fail = 2

| Query results                    |        | Query stats                                                  |
|----------------------------------|--------|--------------------------------------------------------------|
| <span>Completed</span>           |        | Time in queue: 104 ms Run time: 350 ms Data scanned: 0.21 KB |
| <b>Results (2)</b>               |        | <span>Copy</span> <span>Download results CSV</span>          |
| <input type="text"/> Search rows |        |                                                              |
| #                                | Result | cnt                                                          |
| 1                                | Pass   | 6                                                            |
| 2                                | Fail   | 2                                                            |

#### 4) Average marks overall

```
SELECT AVG(Marks) AS avg_marks
FROM students;
```

| Query results                    |           | Query stats                                                  |
|----------------------------------|-----------|--------------------------------------------------------------|
| <span>Completed</span>           |           | Time in queue: 116 ms Run time: 599 ms Data scanned: 1.91 KB |
| <b>Results (1)</b>               |           | <span>Copy</span> <span>Download results CSV</span>          |
| <input type="text"/> Search rows |           |                                                              |
| #                                | avg_marks |                                                              |
| 1                                | 66.375    |                                                              |

#### 5) Department-wise average marks

```
SELECT Dept, AVG(Marks) AS avg_marks
FROM students
GROUP BY Dept;
```

| Query results                    |      | Query stats                                                  |
|----------------------------------|------|--------------------------------------------------------------|
| <span>Completed</span>           |      | Time in queue: 112 ms Run time: 386 ms Data scanned: 2.00 KB |
| <b>Results (3)</b>               |      | <span>Copy</span> <span>Download results CSV</span>          |
| <input type="text"/> Search rows |      |                                                              |
| #                                | Dept | avg_marks                                                    |
| 1                                | MBA  | 71.0                                                         |
| 2                                |      |                                                              |
| 3                                | MCA  | 63.6                                                         |

#### 6) Top scorer

```
SELECT Name, Dept, Marks
```

FROM students

ORDER BY Marks DESC

LIMIT 1;

Expected: Sneha (91)

| Results (1) |       |      |       |
|-------------|-------|------|-------|
| #           | Name  | Dept | Marks |
| 1           | Sneha | MBA  | 91    |

## 7) List failed students

SELECT StudentID, Name, Dept, Marks

FROM students

WHERE Result = 'Fail';

Expected: John, Arun

| Results (2) |           |      |      |
|-------------|-----------|------|------|
| #           | StudentID | Name | Dept |
| 1           | 104       | John | MCA  |
| 2           | 106       | Arun | MCA  |

## Cleanup (to avoid any charges)

1. In Athena, you can keep DB/table (no cost by itself), but clean S3:
2. Go to **S3 bucket**
3. Delete:
  - o students.csv
  - o athena-results/ folder contents (query outputs)
4. Delete the bucket (must be empty to delete)

### **Cost Note (Simple)**

- **S3 storage:** tiny (almost negligible for this)
- **Athena:** charges based on **data scanned**; with this tiny CSV it's usually minimal, but **always delete outputs and bucket** after lab.

**DATE: 31-12-25**

### Exercise-27: Smart Sensor Monitoring System using AWS IoT Core

Cloud-Based IoT Data Ingestion and Processing using AWS - To understand how IoT device data is sent to the cloud, processed automatically, and logged using AWS managed services.

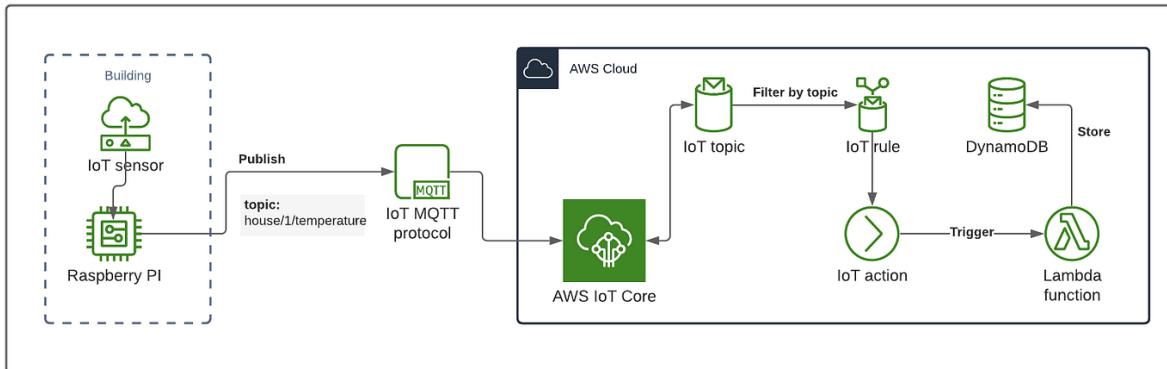
In this lab, a simulated IoT device sends sensor data (such as temperature and humidity) in JSON format to the AWS cloud.

The data is received by AWS IoT Core, which acts as the central message broker for IoT devices.

Whenever new sensor data arrives:

- AWS IoT Core detects the event
- The data is processed automatically
- The processed output is stored or logged in the cloud for monitoring and verification

This exercise demonstrates how real-time device data can be handled without managing servers, highlighting the principles of IoT, cloud computing, and event-driven architecture.



### Concepts Covered

- Internet of Things (IoT)
- Device-to-Cloud communication
- Event-driven processing
- JSON data format
- Serverless cloud services
- Real-time data handling

## **Input**

Simulated sensor data (example: temperature and humidity values)

## **Output**

- IoT message successfully received by AWS
- Automatic processing triggered
- Cloud logs showing received and processed data

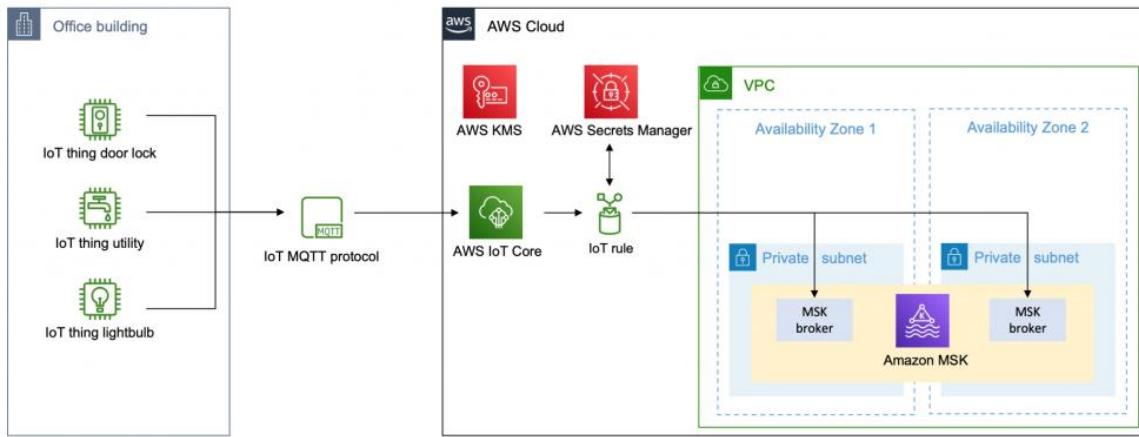
## **Learning Outcome**

After completing this lab, students will be able to:

- Explain how IoT devices communicate with the cloud
- Understand the role of AWS IoT Core in IoT solutions
- Describe event-based data processing in cloud environments
- Relate IoT concepts to real-world smart systems (smart homes, healthcare, industry)

## **STEP 1: Open IoT Service**

1. Login to AWS Management Console
2. Search for **AWS IoT Core**
3. Open the IoT Core dashboard



## STEP 2: Create an IoT Thing (Device)

1. Go to **Manage** → **Things**
2. Click **Create things**
3. Choose **Create single thing**
4. Give a name (example: TempSensor01)
5. Skip advanced settings
6. Create the thing

This represents a sensor device (even though no real hardware is used).

The screenshot shows the AWS IoT Things management interface. The left sidebar includes links for Device Advisor, MQTT test client, Device Location, Query connectivity status, Manage, All devices, Things, Thing groups, Thing types, Fleet metrics, Greengrass devices, LPWAN devices, and Software packages. A success message at the top right states: "You successfully created certificate f23905058f7eb6b62ae1c97a016764d0ece0faa64857f825110e2752a577ec0." Below this, the main area displays "Things (1) Info". It shows a table with one row for "TempSensor01", with columns for Name, Type, Group, Billing, and Searchable attributes. Buttons for Advanced search, Run aggregations, Run connectivity status query, Edit, Delete, and Create things are available. A search bar and pagination controls are also present.

## STEP 3: Create Device Certificate

1. Choose **Auto-generate certificate**
2. Activate the certificate

3. Download:

- o Device certificate
- o Private key
- o Root CA

4. Attach the certificate to the Thing

Certificate = identity of the device

#### STEP 4: Create and Attach IoT Policy

1. Go to Secure → Policies

2. Create a new policy

3. Allow:

- o Connect
- o Publish
- o Subscribe
- o Receive

4. Use \* (for lab simplicity)

5. Attach the policy to the certificate

Policy = permission for device to talk to AWS

The screenshot shows the AWS IoT Certificates page. The left sidebar has a 'Certificates' section under 'Security'. The main area shows a success message: 'Successfully attached the policy TempSensorPolicy to certificate f23905058f7eb6b62ae1c97a016764d0ece0cfaa64857fb25110e2752a577ec0.' Below this, the 'Certificates' tab is selected, showing one certificate listed: 'f23905058f7eb6b62ae1c97a016764d0ece0cfaa64857fb25110e2752a577ec0' with ARN 'arn:aws:iot:us-east-1:257605934444:cert/f23905058f7eb6b62ae1c97'. A status bar at the bottom right indicates '1' certificate.

#### STEP 5: Test Device Messages (No Hardware)

1. Go to Test → MQTT test client

2. Subscribe to a topic

Example:

3. sensor/temperature
4. Publish a message:
5. {
6. "deviceId": "TempSensor01",
7. "temperature": 32,
8. "humidity": 65
9. }

10. Verify message appears instantly

IoT data successfully reached the cloud

The screenshot shows two instances of the AWS IoT MQTT test client interface. Both instances have a green banner at the top stating "Successfully attached the policy TempSensorPolicy to certificate f23905058f7eb6b62ae1c97a016764d0ece0cfaa64857fb25110e2752a577ec0." Below the banner, there are connection status indicators (QoS levels 0, 1, and 2) and a "Disconnect" button.

**Top Instance (Subscription):**

- Connection details:** Connected.
- Subscribe to a topic:** Topic filter: sensor/temperature.
- Additional configuration:** A "Subscribe" button.

**Bottom Instance (Publishing):**

- Connection details:** Connected.
- Subscribe to a topic:** Topic name: sensor/temperature.
- Message payload:**

```
{
 "deviceId": "TempSensor01",
 "temperature": 32,
 "humidity": 65
}
```
- Additional configuration:** A "Publish" button.

## STEP 6: Create a Rule (Automation)

1. Go to Message Routing → Rules

2. Create a rule
3. Rule query example:
4. SELECT \* FROM 'sensor/temperature'
5. Choose action:
  - Send to **CloudWatch Logs**  
*(or Lambda / DynamoDB if required)*

Rule = “when data arrives, do something”

### **STEP 7: Verify Output**

1. Open **CloudWatch → Logs**
2. Check log group created by IoT Rule
3. Confirm sensor data entries

Automatic processing confirmed

### **What Students Finally Demonstrate**

Device creation  
Secure communication  
JSON sensor data  
Event-based automation  
Serverless processing

Sensor data is sent to AWS IoT Core, where rules automatically process and store the data without using any server.

**CloudWatch**

Favorites and recent

- Ingestion
- Dashboards
- ▶ Alarms △ 0 ○ 0 ○ 0
- ▶ AI Operations
- ▶ GenAI Observability
- ▶ Application Signals New (APM)
- ▶ Infrastructure Monitoring
- ▼ Logs
  - Log Management New
  - Log Anomalies
  - Live Tail
  - Logs Insights
  - Contributor Insights
- ▶ Metrics New
- ▶ Network Monitoring

CloudShell Feedback Console Mobile App

**Log group details** Info

CloudWatch Logs offers three log classes: Standard, Infrequent Access, and Delivery. [Learn more about the features offered by each log class.](#)

**Log group name** IoT\_Sensor\_Logs

**Retention setting** Never expire

**Log class** Info Standard

**KMS key ARN - optional**

**Deletion protection** Deletion protection is turned off by default. Deletion protection helps prevent accidental or unintended deletion of log groups that store critical operational or compliance data.

**Tags**

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

Activate Windows Go to Settings to activate Windows.

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

**AWS IoT** > Message routing > Rules > Create rule

Send message data to CloudWatch logs

**Log group name** Info IoT\_Sensor\_Logs

**Batch mode** The payload that contains a JSON array of records will be sent to CloudWatch via a batch call.

Use batch mode

**Create role**

**IAM role** Choose a role or create a new one. AWS IoT will assume this role when it sends messages to CloudWatch Logs.

**Role name** IoTCloudWatchRole

Enter a unique role name that contains alphanumeric characters, hyphens, and underscores. A role name can't contain any spaces.

**Error action - optional**

You can optionally set an action that will be executed when something goes wrong with processing your rule. If two rule actions in the same rule fail, the error action receives one message that contains both errors.

**AWS IoT** > Message routing > Rules

Manage

▼ All devices

Things

Thing groups

Thing types

Fleet metrics

▶ Greengrass devices

▶ LPWAN devices

Software packages

▶ Remote actions

▼ Message routing

Successfully created rule SensorToCloudWatch.

**Rules (1) Info**

Rules allow your things to interact with other services. Rules are analyzed and perform specific actions based on messages published by your devices.

| Name               | Status | Rule topic         | Created date                           |
|--------------------|--------|--------------------|----------------------------------------|
| SensorToCloudWatch | Active | sensor/temperature | January 13, 2026, 12:02:02 (UTC+05:30) |

**AWS IoT > MQTT test client**

Successfully created rule SensorToCloudWatch.

Connect

- Connect one device
- Connect many devices
- Domain configurations

Test

- Device Advisor
- MQTT test client**
- Device Location
- Query connectivity status

Manage

- All devices
- Things
- Thing groups
- Thing types
- Fleet metrics
- Greengrass devices
- LPWAN devices
- Software packages
- Remove actions

CloudShell Feedback Console Mobile App

View rule X

**Connection details** Connected

To disconnect from the MQTT test client, choose Disconnect. To re-establish a connection, you can update the details and choose Connect.

**Subscribe to a topic** **Publish to a topic**

**Topic name**  
The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.

X

**Message payload**

```
{
 "deviceId": "TempSensor01",
 "temperature": 32,
 "humidity": 65
}
```

**Additional configuration**

**Publish**

Activate Windows  
Go to Settings to activate Windows.

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

**CloudWatch > Log management > IoT\_Sensor\_Logs**

Log group "IoT\_Sensor\_Logs" has been created.

**Retention**  
Never expire

**Stored bytes**  
-

**Deletion protection**  
 Off

**Configure**  
[Anomaly detection](#) [Configure](#)

**Log streams** (5)

By default, we only load the most recent log streams.

Exact match  Show expired [Info](#)

| Log stream                    | Last event time                 |
|-------------------------------|---------------------------------|
| SensorToCloudWatch-206645551  | 2026-01-13 12:09:18 (UTC+05:30) |
| SensorToCloudWatch-1906457266 | 2026-01-13 12:09:05 (UTC+05:30) |
| SensorToCloudWatch-572705605  | 2026-01-13 12:07:28 (UTC+05:30) |
| SensorToCloudWatch-880663087  | 2026-01-13 12:07:22 (UTC+05:30) |
| SensorToCloudWatch-1062684149 | 2026-01-13 12:03:59 (UTC+05:30) |

Activate Windows  
Go to Settings to activate Windows.

https://us-east-1.console.aws.amazon.com/cloudwatch/home?region=us...  
© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

**CloudWatch > Log management > IoT\_Sensor\_Logs > SensorToCloudWatch--1062684149**

Log group "IoT\_Sensor\_Logs" has been created.

**Log events**

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

Actions ▾ Start tailing ▾ Create metric filter

Display ▾

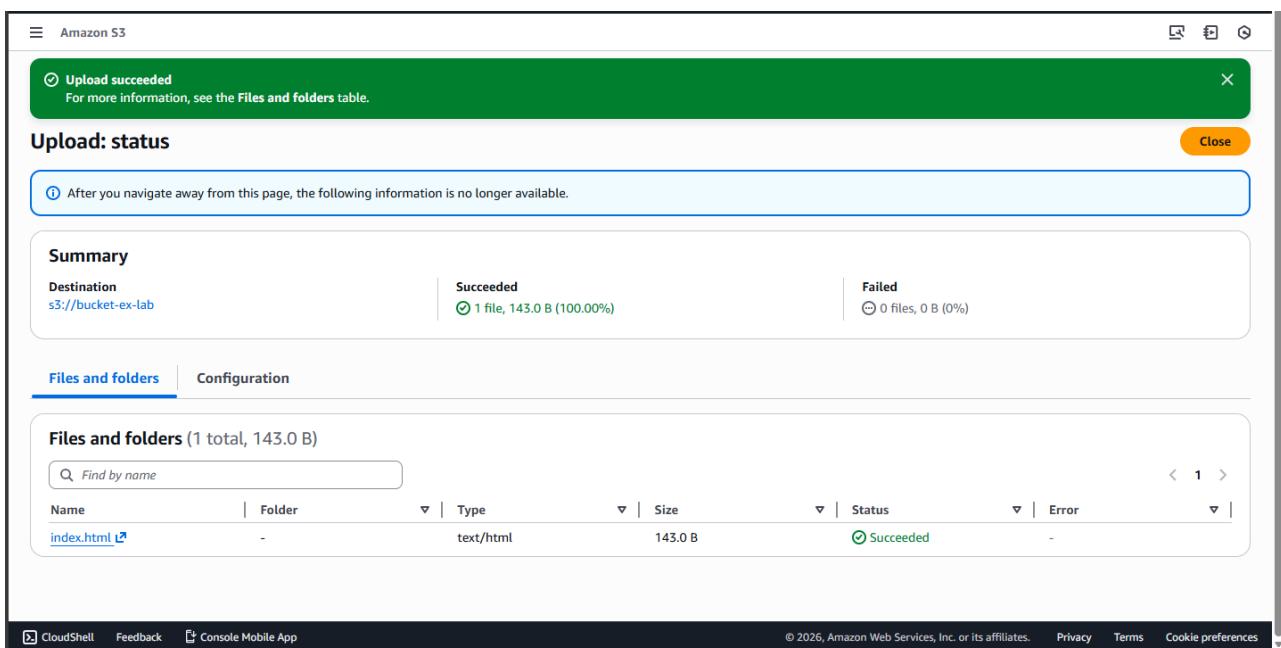
| Timestamp                     | Message                                                                                                                               |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| 2026-01-13T12:03:59.319+05:30 | No more records within selected time range <a href="#">Retry</a><br>{ "deviceId": "TempSensor01", "temperature": 32, "humidity": 65 } |
|                               | No more records within selected time range <a href="#">Auto retrying...</a> <a href="#">Pause</a>                                     |

**DATE: 02-01-26**

## Exercise-28: Accelerate an S3 Static Website Using Amazon CloudFront (CDN)

### Pre-requisite

- You already have an **S3 bucket** with at least:
  - index.html
  - (optional) error.html



The screenshot shows the Amazon S3 console interface. At the top, a green success message box displays: "Upload succeeded. For more information, see the Files and folders table." Below this, a "Upload: status" section indicates "Succeeded" with "1 file, 143.0 B (100.00%)" and "Failed" with "0 files, 0 B (0%)". A "Summary" section shows the destination as "s3://bucket-ex-lab". The "Files and folders" tab is selected, showing a table with one item: "index.html" (text/html, 143.0 B, Status: Succeeded). The table has columns for Name, Folder, Type, Size, Status, and Error. At the bottom of the page, there are links for CloudShell, Feedback, and Console Mobile App, along with copyright information and links for Privacy, Terms, and Cookie preferences.

### Part A - Ensure S3 is ready

- Open S3 → your bucket
- Go to Properties → Static website hosting
- Enable it, set:
  - Index document: index.html
  - Error document: error.html (optional)

**Note down the S3 Website endpoint shown there.**

## **Part B - Create CloudFront Distribution**

- Go to **CloudFront → Create distribution**
- **Origin domain**
  - Select your **S3 bucket** (not the website endpoint)
- **Origin access**
  - Choose **Origin access control (OAC)** (recommended)
  - Click **Create control setting** if asked
- **Viewer protocol policy**
  - Choose **Redirect HTTP to HTTPS**
- **Default root object**
  - Set: index.html
- Click **Create distribution**

**Copy the Distribution domain name** (looks like dxxxxx.cloudfront.net)

## **Part C - Allow CloudFront to read the bucket (important)**

After creating distribution, CloudFront usually shows a message like “**S3 bucket policy needs update**”.

- Click **Copy policy**
- Go to **S3 → Bucket → Permissions → Bucket policy**
- Paste and **Save**

This step ensures **public users cannot directly access S3**, but CloudFront can.

## **Part D - Test**

- Wait until Distribution status becomes **Enabled** (and “Deployed”)
- Open in browser:
  - <https://dxxxxx.cloudfront.net>

Expected:

- Your index.html loads via CloudFront.

## **Viva**

- **S3** stores the website files
- **CloudFront** caches them at edge locations (faster)
- **OAC** ensures S3 is not public; CloudFront accesses it securely

**DATE: 02-01-26**

## **Exercise-29 Mini Project –**

### **Global Static Website Delivery using S3 + CloudFront with Cache Update Demo**

Host a static website in S3, deliver it through CloudFront (CDN), then do a content update and observe caching behavior (old page still showing). Finally, force the latest version using CloudFront Invalidations.

#### **A) Prerequisites**

- AWS Account
- An S3 bucket
- 2 small HTML files ready:
  - index.html
  - about.html

#### **index.html (Version 1)**

```
<html>
 <body>
 <h1>My CloudFront Mini Project</h1>
 <h2>Version: V1</h2>
 <p>Loaded via CloudFront CDN</p>
 </body>
</html>
```

#### **about.html**

```
<html>
 <body>
 <h1>About Page</h1>
 <h2>Version: V1</h2>
 </body>
</html>
```

## B) Step-by-step Execution

### Step 1: Create S3 Bucket

Go to S3 → Create bucket

Bucket name: cf-mini-project-<yourname>

Region: (keep default)

Block Public Access: Keep ON

Click Create bucket

### Step 2: Upload Website Files

Open your bucket → Upload

Upload:

1. index.html
2. about.html

Click Upload

The screenshot shows the Amazon S3 console interface. On the left, there's a navigation sidebar with options like 'Amazon S3', 'Buckets', 'Access management and security', 'Storage management and insights', and links to 'CloudShell', 'Feedback', and 'Console Mobile App'. The main area is titled 'cf-mini-project-joshua' and shows 'Objects (2)'. It lists two files: 'about.html' and 'index.html'. Both files are of type 'html' and were uploaded on January 20, 2026, at 12:23:47 (UTC-05:30). The file sizes are 89.0 B and 143.0 B respectively. The storage class is Standard. There are also buttons for 'Actions', 'Create folder', and 'Upload'.

| Name       | Type | Last modified                          | Size    | Storage class |
|------------|------|----------------------------------------|---------|---------------|
| about.html | html | January 20, 2026, 12:23:47 (UTC-05:30) | 89.0 B  | Standard      |
| index.html | html | January 20, 2026, 12:23:48 (UTC-05:30) | 143.0 B | Standard      |

### Step 3: Create CloudFront Distribution (with secure access)

Go to CloudFront → Create distribution

Origin domain

- Select your S3 bucket (from dropdown)

#### Origin access

- Choose Origin access control (OAC)
- Click Create control setting (if asked) → Create

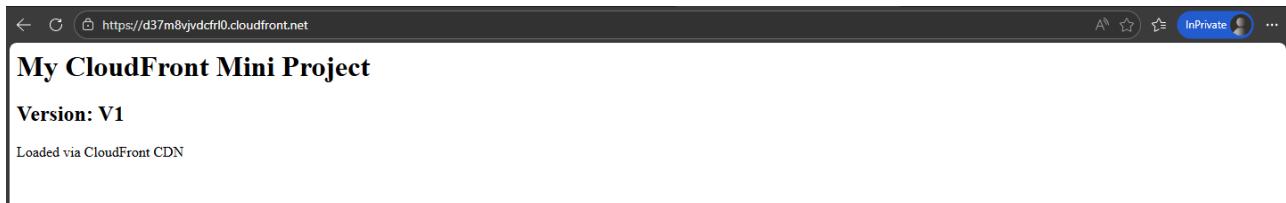
#### Viewer protocol policy

- Select Redirect HTTP to HTTPS

#### Default root object

- Enter: index.html

#### Click Create distribution



### Step 4: Add Bucket Policy for OAC (important)

After distribution creation, CloudFront page will show a message like:

- “S3 bucket policy needs to be updated”

Click Copy policy

Go to S3 → your bucket → Permissions → Bucket policy

Paste policy → Save changes

Now: S3 is NOT public, but CloudFront can fetch objects.

### Step 5: Test the Website Using CloudFront

Go back to CloudFront

Open your distribution

Copy the Distribution domain name, like:

- dxxxxx.cloudfront.net

Open in browser:

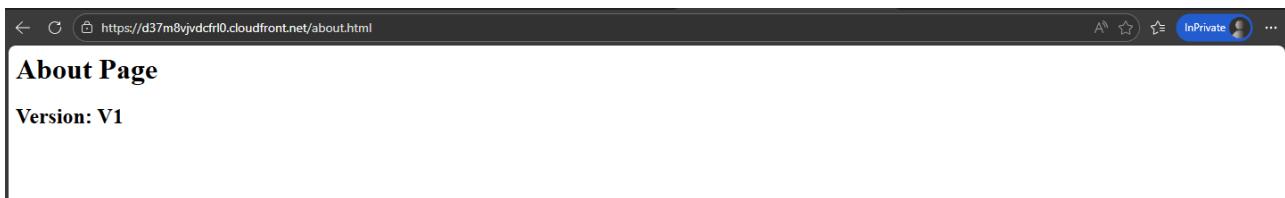
- <https://dxxxxx.cloudfront.net>

Also test:

- <https://dxxxxx.cloudfront.net/about.html>

Screenshot evidence:

- CloudFront distribution details page (domain + status)
- Browser output showing **Version V1**



### C) Cache Demo (Version Update + See Old Content)

#### Step 6: Update Website Content in S3 (V2)

Edit your index.html and change:

From:

<h2>Version: V1</h2>

To:

<h2>Version: V2</h2>

Now re-upload updated index.html:

S3 → bucket → Upload → add updated index.html

Upload

#### Step 7: Observe CloudFront Caching

Immediately refresh:

- <https://dxxxxx.cloudfront.net>

Many times you may still see **V1** (cached), even though S3 has V2.

This is the caching behavior demonstration.

Tip for students:

- Do a hard refresh: Ctrl + Shift + R
- Still might show V1 because cache is on CloudFront edge.

Take screenshot if it still shows V1.



#### D) Force Latest Content Using Invalidation

##### Step 8: Create CloudFront Invalidation

CloudFront → your distribution

Go to Invalidations tab

Click Create invalidation

In “Object paths”, enter:

- /index.html  
(or use /\* to clear everything)

Click **Create invalidation**

Wait until status becomes **Completed**.

##### Step 9: Verify Latest Version

Refresh the CloudFront URL again: <https://dxxxxx.cloudfront.net>

Now it should show **Version: V2**

Screenshot evidence:

- Invalidiation status “Completed”
- Browser showing V2

### **Cleanup (to avoid charges)**

1. CloudFront: Disable distribution → wait → Delete distribution
2. S3: Empty bucket → Delete bucket

### **Mini-Project Output Checklist:**

S3 bucket file list (index.html, about.html)

CloudFront distribution domain name

Browser output **V1**

S3 updated file showing **V2**

Browser still showing **V1** (optional if observed)

CloudFront invalidation completed

Browser output **V2**

### **Viva Questions**

1. Why use CloudFront instead of accessing S3 directly?
2. What does “cache” mean?
3. Why was invalidation needed?
4. What is OAC and why is it useful?

**DATE: 07-01-26**

## **Exercise–30 Mini Project – Deploying a Containerized App on AWS using Amazon EKS**

- Create an EKS cluster on AWS
- Add worker nodes (managed node group)
- Connect using kubectl
- Deploy Nginx and expose it

### **Region**

Use **Mumbai (ap-south-1)** (or keep consistent with your account).

### **Prerequisites**

- Logged in with root email + password
- **AWS CloudShell**

### **Method: CloudShell + eksctl**

#### **Step A1: Open CloudShell**

AWS Console (Mumbai region) → click **CloudShell** icon (terminal opens in browser)

#### **Step A2: Check tools**

Run:

```
aws --version
```

```
kubectl version --client
```

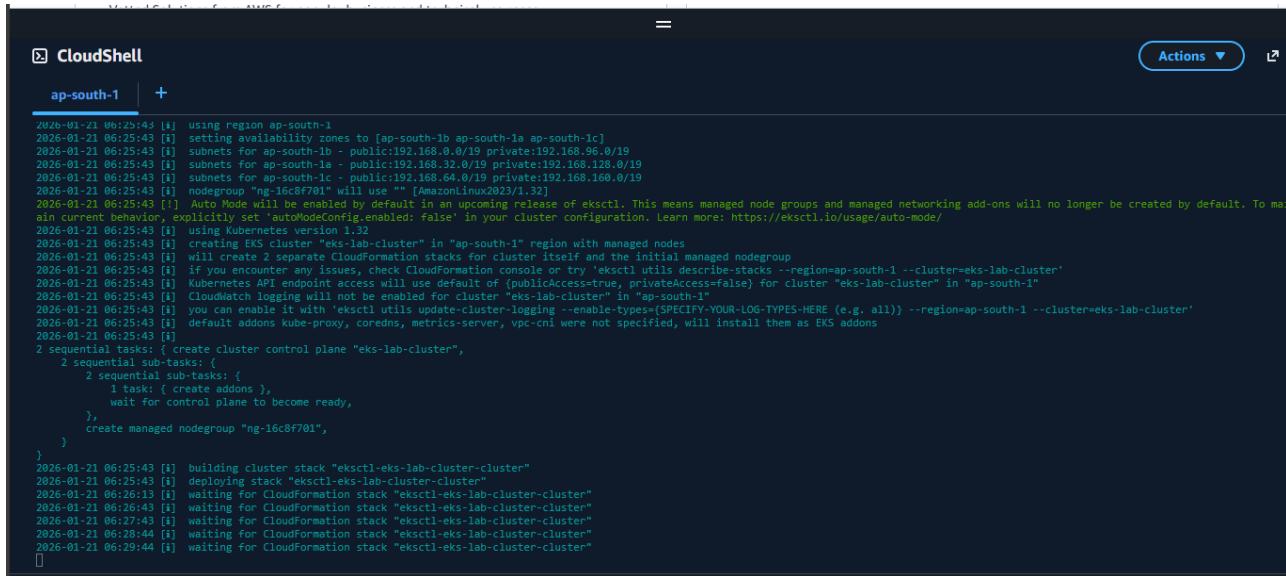
```
eksctl version
```

If eksctl is not found, install it ([CloudShell often has it](#)).

## Step A3: Create EKS cluster + node group (single command)

Run:

```
eksctl create cluster \
--name eks-lab-cluster \
--region ap-south-1 \
--nodes 2 \
--node-type t3.medium \
--managed
```



The screenshot shows a terminal window titled "CloudShell" with the region set to "ap-south-1". The command "eksctl create cluster" is being run, followed by several options: --name eks-lab-cluster, --region ap-south-1, --nodes 2, --node-type t3.medium, and --managed. The terminal output displays log messages from 2026-01-21 at 06:25:43, detailing the creation of the EKS cluster and its associated CloudFormation stacks. It mentions setting availability zones, creating subnets, and deploying managed node groups. A note at the bottom indicates that Managed Node Groups will be enabled by default in an upcoming release of eksctl, and that managed networking add-ons will no longer be created by default. It also provides a link to the usage documentation.

### If eksctl asks / fails due to IAM (rare, but in case)

Run: aws sts get-caller-identity

If it shows:

- Your root account ID
- ARN ending with :root

Then IAM is 100% fine.

This automatically:

- Creates a **new VPC** with correct subnets/routes
- Creates the **EKS cluster**
- Creates the **Managed Node Group (worker nodes)**
- Configures kubeconfig for you

Wait until it completes (it will print “cluster created”)

#### Step A4: Verify nodes

kubectl get nodes

Expected: 2 nodes in **Ready** state.

```
~ $ kubectl get nodes
NAME STATUS ROLES AGE VERSION
ip-192-168-20-92.ap-south-1.compute.internal Ready <none> 2m58s v1.32.9-eks-ecaa3a6
ip-192-168-87-217.ap-south-1.compute.internal Ready <none> 2m58s v1.32.9-eks-ecaa3a6
~ $
```

#### Step A5: Deploy Nginx

kubectl create deployment webapp --image=nginx

kubectl get pods

```
~ $ kubectl create deployment webapp --image=nginx
deployment.apps/webapp created
~ $ kubectl get pods
NAME READY STATUS RESTARTS AGE
webapp-6fddc68b96-d7d5n 0/1 ContainerCreating 0 0s
~ $
```

#### Step A6: Expose using LoadBalancer

kubectl expose deployment webapp --type=LoadBalancer --port=80

kubectl get svc

Wait until EXTERNAL-IP becomes a value (not <pending>).

```
~ $ kubectl expose deployment webapp --type=LoadBalancer --port=80
service/webapp exposed
~ $ kubectl get svc
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
kubernetes ClusterIP 10.100.0.1 <none> 443/TCP 10m
webapp LoadBalancer 10.100.216.214 <pending> 80:32686/TCP 1s
~ $
```

#### Step A7: Open in browser

Copy the EXTERNAL-IP and open:

<http://<EXTERNAL-IP>>

Expected: **Nginx welcome page**

