

Activity 0: Environment Setup

Objective

Launch an AWS EC2 instance (Ubuntu 24.04 LTS) and install Docker so that all subsequent lab activities can be executed on this environment.

By the end of this activity you will have:

- A running Ubuntu VM on AWS EC2.
 - Docker installed and verified.
 - SSH access working from your lab environment.
-

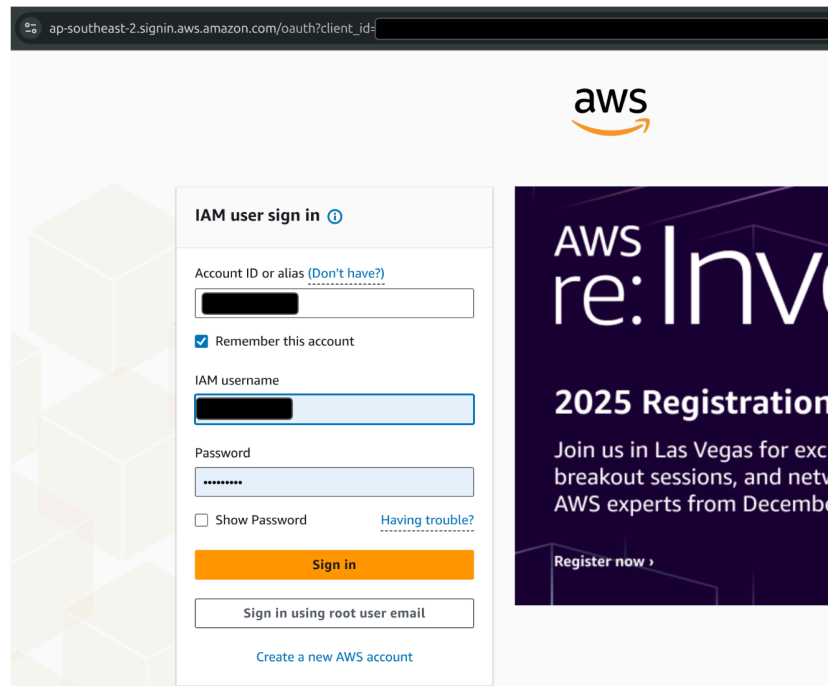
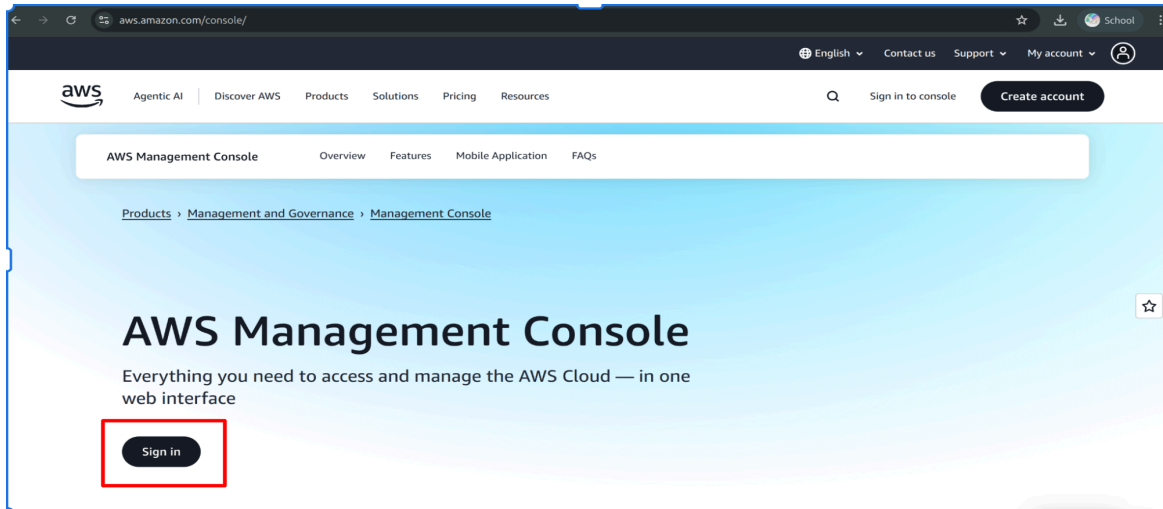
Prerequisites

- An AWS account with EC2 permissions.
 - Local SSH client (Linux/macOS: builtin; Windows: WSL, PowerShell + OpenSSH, or Git Bash).
 - Minimum VM requirements: **2 vCPU, 4 GiB RAM, 20 GiB disk**.
 - Recommended instance type: **t3.medium** (stop/terminate when not in use to avoid charges).
-

Step-by-step Guidance

Step 1 — Sign in to AWS Management Console

1. Open: <https://aws.amazon.com/console/>
2. Click **Sign in to the console** and log in with your AWS account.



Step 2 — Navigate to EC2 and Launch Instance

1. In the Console, go to **All Services** → **Compute** → **EC2**.
2. Click **Launch instance**.

The screenshot shows the AWS Management Console interface. At the top, the URL is `ap-south-1.console.aws.amazon.com/console/services?region=ap-south-1`. The navigation bar includes the AWS logo, a search bar, and a hamburger menu icon (labeled 1). Below the navigation bar, the left sidebar shows 'Console Home' and 'All services' (labeled 2). The main content area displays 'All services' with a 'Services by category' section. Under the 'Compute' category, 'EC2' is highlighted (labeled 3). Below this, the 'Resources' section shows a table of EC2 resources in the Asia Pacific (Mumbai) Region. The table includes columns for various resource types and their counts. The 'Launch instance' section is visible, with a 'Launch instance' button (labeled 4) and a 'Migrate a server' button. The 'Service health' section shows the AWS Health Dashboard and the status of the service, which is 'operating normally'.

ap-south-1.console.aws.amazon.com/console/services?region=ap-south-1

aws

Search [Alt+S]

Console Home > All services

Console Home

myApplications

All services

All services

Services by category

Compute

EC2

Lightsail

Lambda

Resources

EC2 Global View

You are using the following Amazon EC2 resources in the Asia Pacific (Mumbai) Region:

Instances (running)	0	Auto Scaling Groups	0	Capacity Reservations	0
Dedicated Hosts	0	Elastic IPs	0	Instances	0
Key pairs	0	Load balancers	0	Placement groups	0
Security groups	1	Snapshots	0	Volumes	0

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Launch instance

Migrate a server

Service health

AWS Health Dashboard

Region

Asia Pacific (Mumbai)

Status

✓ This service is operating normally.

Step 3 – Name and Tags

1. In the **Name and Tags** section, set **Name** to something like: `docker-lab-vm`.
2. (Optional) Add additional tags for owner, course, etc.

Name and tags [Info](#)

Name

docker-lab-vm

Add additional tags

Step 4 – Choose Amazon Machine Image (AMI)

1. Select **Ubuntu Server 24.04 LTS (HVM), SSD Volume Type**.
2. Make sure Architecture is selected as **64-bit(x86)**.

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](#).

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

Debian

debian

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-0f918f7e67a3323f0 (64-bit (x86)) / ami-02f607855bfce66b6 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

DescriptionUbuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture	AMI ID	Publish Date	Username	
64-bit (x86)	ami-0f918f7e67a3323f0	2025-06-10	ubuntu	<div>Verified provider</div>

Step 5 – Choose Instance Type

1. Choose **t3.medium** (2 vCPU, 4 GiB RAM).
[This choice ensures one has enough resources for the lab as mentioned in prerequisite.]
⚠ This instance type may incur charges. Stop/terminate when not required.
2. Prefer an equivalent **Free-tier eligible** AMI if available.

▼ **Instance type** [Info](#) | [Get advice](#)

Instance type

t3.small Free tier eligible

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

On-Demand Ubuntu Pro base pricing: 0.0259 USD per Hour

On-Demand Linux base pricing: 0.0224 USD per Hour On-Demand SUSE base pricing: 0.0534 USD per Hour

On-Demand RHEL base pricing: 0.0512 USD per Hour On-Demand Windows base pricing: 0.0408 USD per Hour

Additional costs apply for AMIs with pre-installed software

Step 6 — Create (or Use) Key Pair

1. Under **Key pair (login)** select **Create new key pair**.
 2. Choose **RSA** and **.pem** format. Name it (e.g., **docker-lab-key**).
 3. Click **Create key pair** and download the **.pem** file.
- ⚠ Save this **.pem** file securely — this is your only download opportunity.

Create key pair

✕

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

docker-lab-key

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 7 – Configure Network / Security Group

1. Create a new security group (e.g., `docker-lab-sg`).
 2. Add rules:
 - **SSH** – TCP 22 – Source `0.0.0.0/0` (or restrict to your IP)
 - **HTTP** – TCP 80 – Source `0.0.0.0/0`

*[Here, we want to allow SSH and HTTP to the VM from **Anywhere**]*
 3. (Optional) Add additional rules as required by your lab.
- ⚠ For real production or tighter security, restrict SSH to your IP only.

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*
`docker-lab-sg`

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 `./:/0#,@!+=8;()$*`

Description - *required* [Info](#)
`docker-lab-sg allows HTTP and SSH connections from anywhere created on 2025-08-11T12:15`

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0, SSH connections from anywhere for ...)

Type	Info	Protocol	Info	Port range	Info	Description - optional	Info
ssh		TCP		22		SSH connections from anywhere for lab	
Source type	Info	Source	Info				
Anywhere		<input type="text" value="0.0.0.0/0"/>					

▼ Security group rule 2 (TCP, 80, 0.0.0.0/0, HTTP connections from anywhere for ...)

Type	Info	Protocol	Info	Port range	Info	Description - optional	Info
HTTP		TCP		80		HTTP connections from anywhere for lab	
Source type	Info	Source	Info				
Anywhere		<input type="text" value="0.0.0.0/0"/>					

Step 8 – Configure Storage

1. Set root volume to **20 GiB (gp3)**.
[AWS EBS(Elastic Block Storage) gp3 volumes are generally cheaper and offer better performance than gp2 volumes]
2. Confirm and proceed.

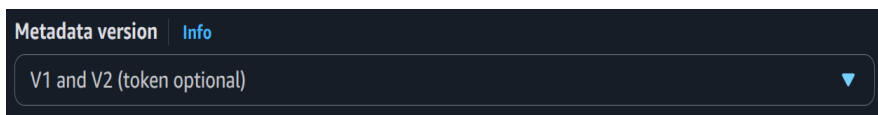
▼ **Configure storage** [Info](#)

1x GiB Root volume, 3000 IOPS, Not encrypted

Step 9 — Make Metadata accessible

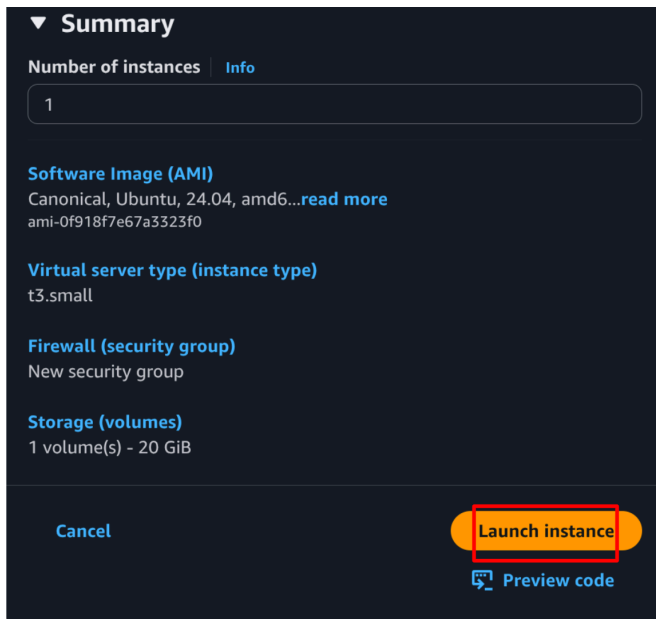
1. In the **Advanced details** Info select **Metadata Version**
2. Choose **V1 and V2 (token optional)**

[V2 enforces token-based access for security where in lab setting we intentionally make token optional using V1 so that no need to attach a token with any ssh request into our ec2 instance]



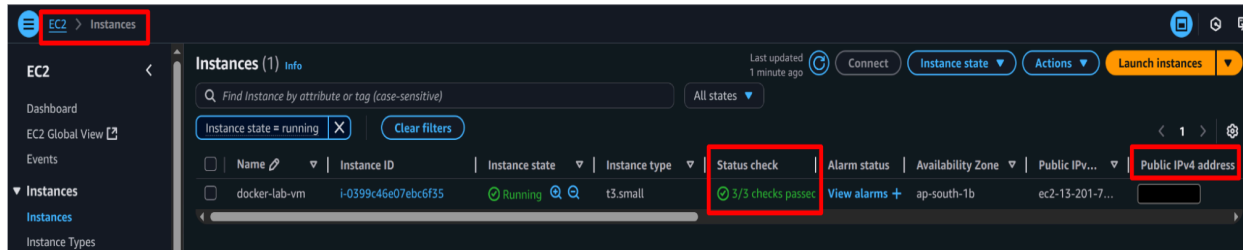
Step 10 — Launch Instance

1. Review configuration and click **Launch instance**.



Step 11 – Note Public IP

1. In **Instances** view wait until the instance state is **running** and status checks show **3/3 checks passed**.
2. In **EC2** → **Instances**, copy the **Public IPv4 address** of your instance.



Step 12 – Prepare SSH Key Locally and Connect

Since cLab is a containerised application, direct file transfer from your local machine is limited. Follow these steps:

1. Open the **cLab** app and enter the current lab → current activity.
2. Click the **POP TERMINAL** button (opens a terminal at **labDirectory**).
3. Run:

```
Shell
chmod 770 secret-key.pem
```

4. Click **OPEN DIRECTORY** in cLab (opens **labDirectory**).
5. Open the downloaded **.pem** file (**docker-lab-key.pem**) from your local machine, copy its full content, and paste it into **secret-key.pem** inside **labDirectory**. Save the file.
6. Again, in the popped terminal, run:

```
Shell
chmod 400 secret-key.pem
```

- This ensures the private key has the correct permissions.
7. From the **popped terminal** in cLab (**labDirectory**), run:

Shell

```
ssh -i secret-key.pem ubuntu@<public-ip-address>
```

Replace **<public-ip-address>** with the value noted in **Step 11**.

Step 13 – Install Docker on the EC2 Instance

Step 13.0 – Login to the EC2 instance (ubuntu)

Shell

```
ssh -i secret-key.pem ubuntu@<public-ip-address>
```

Step 13.1 – Update Ubuntu packages and install prerequisites

- **apt-transport-https** : for https package retrieval
- **curl** : for data transfer
- **conntrack** : for network connection tracking

Shell

```
sudo apt-get update -y  
sudo apt-get upgrade -y  
sudo apt-get install -y apt-transport-https curl conntrack
```

Step 13.2 – Download Docker installation script

Shell

```
curl -fsSL https://get.docker.com -o get-docker.sh
```

Step 13.3 – Run the Docker installation script

Shell

```
sudo sh get-docker.sh
```

Step 13.4 – Allow current user to run Docker without sudo

Shell

```
sudo usermod -aG docker $USER && newgrp docker
```

⚠ If `docker` commands require `sudo` after `usermod`, **log out** and log back in or use `newgrp docker` to refresh group membership.

Step 14 – Verify Docker Installation

Check Docker version

```
Shell
docker --version
```

Expected output (sample):

```
None
Docker version 28.3.3, build 980b856
```

⚠ If you see `docker: command not found`, the installation was unsuccessful.

Step 15 – Verify Docker with Hello World

```
Shell
docker run hello-world
```

Expected output (excerpt):

```
None
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Step 16 — Configure for Evaluation

- Open the `data.json` file in `labDirectory`.
 - Paste your AWS EC2 **Public IPv4 address** into the appropriate field.
 - Save the file.
-

Evaluation (what the grader checks)

The evaluation script will:

1. SSH into your EC2 instance using the provided `public-ip` and `secret-key.pem`.
 2. Verify:
 - The AWS **EC2 instance** is set up correctly.
 - **Docker** is installed and configured correctly.
-

Notes & Tips

- Keep `secret-key.pem` private and do not share it publicly.
 - If you use a restricted SSH source IP for better security, ensure the grader's IP (or the mechanism used by the autograder) can connect.
-

Stop / Terminate Instance (to avoid charges)

- After completing evaluation, it is important to stop the instance so that no extra costs are incurred.
- To **stop** the instance (preserve disk/data): In EC2 Console → select instance → **Instance state** → **Stop instance**.
- To **terminate** (delete resources): **Instance state** → **Terminate instance**.



Congratulations — You've completed Activity 0!

===== END OF DOCUMENT =====