

REST API DEPLOYMENT

Deployed a containerized Node.js REST API on AWS using a scalable, production-ready architecture. The API is hosted on ECS Fargate, exposed securely through an Application Load Balancer, and managed via Amazon ECR.

Tech Stacks Used

- **Node.js (Express.js)** – REST API application
- **Docker** – Containerization
- **Amazon EC2** – Virtual server for Docker image build
- **Amazon ECR** – Container image repository
- **Amazon ECS (Fargate)** – Serverless container orchestration
- **Application Load Balancer (ALB)** – Traffic routing and health checks
- **AWS VPC & Security Groups** – Networking and access control
- **AWS CLI** – Image push and infrastructure interaction

STEP 1: Launch an EC2 Virtual Server (Docker Build Server)

This EC2 instance will:

- Run Docker
- Build your website image
- Push image to ECR
(No Docker Desktop needed)

1.1 Go to EC2 Console

AWS Console → **EC2** → **Instances** → **Launch instance**

The screenshot shows the AWS EC2 Instances page. The left sidebar has 'EC2' selected under 'Instances'. The main area is titled 'Instances Info' with a search bar and filters for 'Name', 'Instance ID', 'Instance state', 'Instance type', 'Status check', 'Alarm status', and 'Availability Zone'. A message says 'No instances' and 'You do not have any instances in this region'. A blue 'Launch instances' button is at the bottom.

1.2 Choose AMI (Operating System)

Select:

Ubuntu

Why:

- AWS optimized
- Stable
- Docker works perfectly

The screenshot shows the 'Launch an instance' wizard. Step 1: Set instance details. It shows a summary of 1 instance, the software image (Canonical, Ubuntu, 24.04, amd64), virtual server type (t3.micro), storage (1 volume(s) - 8 GiB), and a preview code. The 'Quick Start' tab is selected in the application and OS images section, which lists recent AMIs like Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian.

1.3 Choose Instance Type

Select:

t2.micro or t3.micro
(Free-tier eligible)

Click **Next / Continue**

EC2 > Instances > Launch an instance

Summary

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 24.04, amd64 noble image
ami-02b8269d5e85954ef

Virtual server type (instance type): t3.micro

Firewall (security group): New security group

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

Launch Instance | **Preview code**

1.4 Key Pair (VERY IMPORTANT)

- Select **Create new key pair**
- Key pair type: RSA
- File format: **.pem**
- Name: **deployment-project-key**

Download and save this file safely
(You cannot download it again)

EC2 > Instances > Launch an instance

Create key pair

Key pair name: deployment-project-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

Note: 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**

1.5 Network Settings

Configure:

- VPC: Use your created VPC - **deployment-project-VPC**
- Subnet: Public subnet
- Auto-assign public IP: **Enabled**

VPC > Your VPCs > Create VPC

IPv4 CIDR block Info
Determine the starting IP and the size of your VPC using CIDR notation.
 65,536 IPs
CIDR block size must be between /16 and /28.

IPv6 CIDR block Info
 No IPv6 CIDR block
 Amazon-provided IPv6 CIDR block

Tenancy Info

Encryption settings - optional

Number of Availability Zones (AZs) Info
Choose the number of AZs in which to provision subnets. We recommend at least two AZs for high availability.

Customize AZs

Number of public subnets Info
The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet.

Preview

VPC Show details
Your AWS virtual network
deployment-project-VPC-vpc

Subnets (4)
Subnets within this VPC

- ap-south-1a
 - deployment-project-VPC-subnet-
 - deployment-project-VPC-subnet-
- ap-south-1b
 - deployment-project-VPC-subnet-
 - deployment-project-VPC-subnet-

Route tables (3)
Route network traffic to resources

- deployment-project-VPC-rtb-pub
- deployment-project-VPC-rtb-prv
- deployment-project-VPC-rtb-prv

VPC > Your VPCs > Create VPC

Number of public subnets Info
The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet.

Number of private subnets Info
The number of private subnets to add to your VPC. Use private subnets to secure backend resources that don't need public access.

Customize subnets CIDR blocks

NAT gateways (\$ - updated) Info
NAT gateway allows private resources to access the internet from any availability zone within a VPC, providing a single managed internet exit point for the entire region. Additional charges apply.
 None Regional - new Zonal

Introducing regional NAT gateway X
AWS now offers a multi-AZ NAT Gateway, eliminating the need for separate NAT Gateways across availability zones.

VPC endpoints Info
Endpoints can help reduce NAT gateway charges and improve security by accessing S3 directly from the VPC. By default, full access policy is used. You can customize this policy at any time.
 None S3 Gateway

DNS options Info
 Enable DNS hostnames
 Enable DNS resolution

Additional tags

Preview

Subnets (4)
Subnets within this VPC

- ap-south-1a
 - deployment-project-VPC-subnet-
 - deployment-project-VPC-subnet-
- ap-south-1b
 - deployment-project-VPC-subnet-
 - deployment-project-VPC-subnet-

Route tables (3)
Route network traffic to resources

- deployment-project-VPC-rtb-public
- deployment-project-VPC-rtb-private1-
- deployment-project-VPC-rtb-private2-

Network connections (2)
Connections to other networks

- deployment-project-VPC-igw
- deployment-project-VPC-vpc-e-s2

VPC > Your VPCs > Create VPC

NAT gateways (\$ - updated) Info
NAT gateway allows private resources to access the internet from any availability zone within a VPC, providing a single managed internet exit point for the entire region. Additional charges apply.
 None Regional - new Zonal

Introducing regional NAT gateway X
AWS now offers a multi-AZ NAT Gateway, eliminating the need for separate NAT Gateways across availability zones.

VPC endpoints Info
Endpoints can help reduce NAT gateway charges and improve security by accessing S3 directly from the VPC. By default, full access policy is used. You can customize this policy at any time.
 None S3 Gateway

DNS options Info
 Enable DNS hostnames
 Enable DNS resolution

Additional tags

Preview

Subnets (4)
Subnets within this VPC

- ap-south-1a
 - deployment-project-VPC-subnet-
 - deployment-project-VPC-subnet-
- ap-south-1b
 - deployment-project-VPC-subnet-
 - deployment-project-VPC-subnet-

Route tables (3)
Route network traffic to resources

- deployment-project-VPC-rtb-public
- deployment-project-VPC-rtb-private1-
- deployment-project-VPC-rtb-private2-

Network connections (2)
Connections to other networks

- deployment-project-VPC-igw
- deployment-project-VPC-vpc-e-s2

[Cancel](#) [Preview code](#) [Create VPC](#)

The screenshot shows the 'Create VPC resources' step in the VPC creation wizard. It lists 20 successful actions, such as creating the VPC, enabling DNS hostnames, and verifying route table creation. A prominent orange 'View VPC' button is located at the bottom right.

The screenshot displays the 'Your VPCs' section of the VPC dashboard. It shows a table with two entries: 'vpc-029195c9a66d47cc5' and 'deployment-project-VPC'. The 'deployment-project-VPC' row is selected. Below the table, a detailed view for 'deployment-project-VPC' is shown, including tabs for Details, Resource map, CIDRs, Flow logs, Tags, and Integrations.

The screenshot shows the 'Launch an instance' wizard. In the 'Network settings' section, it configures a VPC (selected), subnet (selected), and assigns a public IP (selected). On the right, the 'Summary' section shows 1 instance being launched with the AMI 'ami-02b8269d5e85954ef' and instance type 't3.micro'. A large orange 'Launch instance' button is at the bottom right.

1.6 Security Group Configuration

Create a new security group:

Inbound Rules

Add these rules:

```

SSH    | TCP | 22 | 0.0.0.0/0
HTTP   | TCP | 80 | 0.0.0.0/0
All TCP | TCP | 0.0.0.0/0

```

Why:

- SSH → connect to EC2
- HTTP → test website temporarily

EC2 > Instances > Launch an instance

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

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](#)

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type	Protocol	Port range
Info ssh	Info TCP	Info 22

Source type [Info](#)

Description - optional [Info](#)

⚠️ 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.

Summary

Number of instances [Info](#)

Software Image (AMI)
Canonical, Ubuntu, 24.04, amd64... [read more](#)
ami-02bb8269d5e85954ef

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

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

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](#)

Inbound Security Group Rules

▼ Security group rule 2 (TCP, 0-65535, 0.0.0.0/0)

Type	Protocol	Port range
Info All TCP	Info TCP	Info 0-65535

Source type [Info](#)

Description - optional [Info](#)

▼ Security group rule 3 (TCP, 80, 0.0.0.0/0)

Type	Protocol	Port range
Info HTTP	Info TCP	Info 80

Source type [Info](#)

Description - optional [Info](#)

⚠️ 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.

[Add security group rule](#)

Summary

Number of instances [Info](#)

Software Image (AMI)
Canonical, Ubuntu, 24.04, amd64... [read more](#)
ami-02bb8269d5e85954ef

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

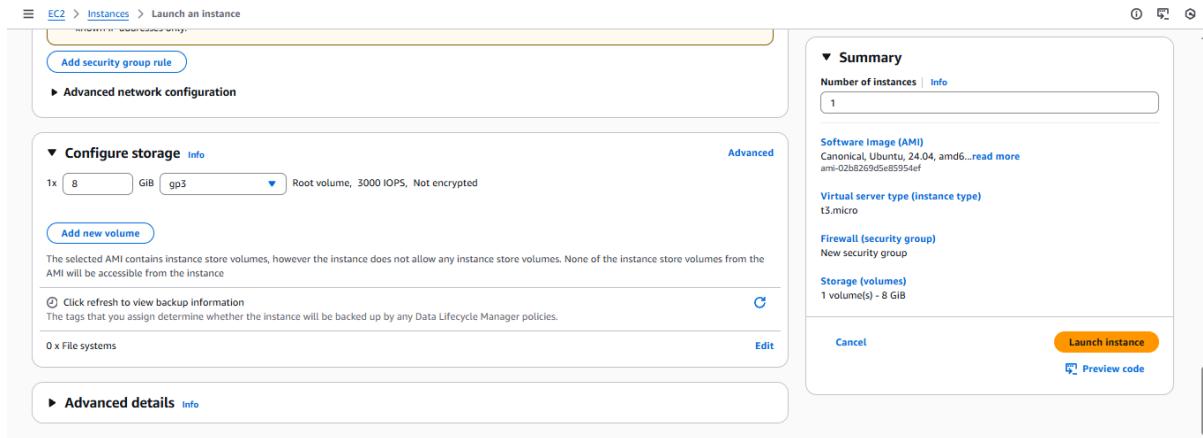
Storage (volumes)
1 volume(s) - 8 GiB

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

1.7 Storage

Leave default:

8 GB gp3

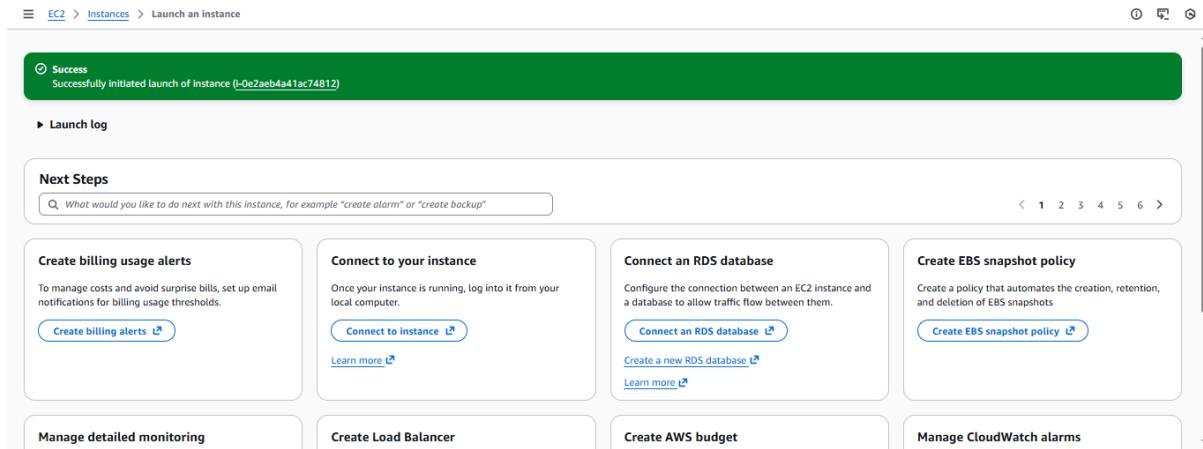


1.8 Launch Instance

Click **Launch Instance**

Wait until:

- Instance state = **Running**
- Status checks = **3/3 passed**



The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like Dashboard, AWS Global View, Events, Instances (selected), Images, and Elastic Block Store. The main area displays a table of instances. One instance is selected, labeled 'deployment-project' with the ID 'i-0e2aeab4a41ac74812'. The instance is shown as 'Running' with a Public IPv4 address of '3.110.207.94'. The 'Details' tab is selected in the instance summary panel.

STEP 2: Connect to EC2 (Using AWS Console)

You are already inside the EC2 terminal.
Now we prepare this virtual server to act as a **Docker machine**.

This screenshot is identical to the one above, showing the AWS EC2 Instances page with the same instance details and selection.

The screenshot shows the 'EC2 Instance Connect' dialog box. It has tabs for Session Manager, SSH client, and EC2 serial console. The 'Session Manager' tab is active. It shows the instance ID 'i-0e2aeab4a41ac74812' and a connection type section where 'Public IPv4 address' is selected. A note at the bottom says, 'Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.' There are 'Cancel' and 'Connect' buttons at the bottom right.

```
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Sat Dec 20 04:39:57 UTC 2025

System load: 0.15 Temperature: -273.1 C
Usage of /: 25.9% of 6.71GB Processes: 111
Memory usage: 22% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 10.0.25.153

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Sat Dec 20 04:39:58 2025 from 13.233.177.5
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-25-153:~$
```

STEP 3: Install Docker on EC2 (Virtual Docker)

3.1 Update the EC2 System

Run this **exact command** in the EC2 terminal:

```
sudo su
apt update
```

What this does

- Updates system packages
- Prevents Docker installation issues

Wait until it completes.

```
Swap usage: 0%          IPv4 address for ens5: 10.0.25.153

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

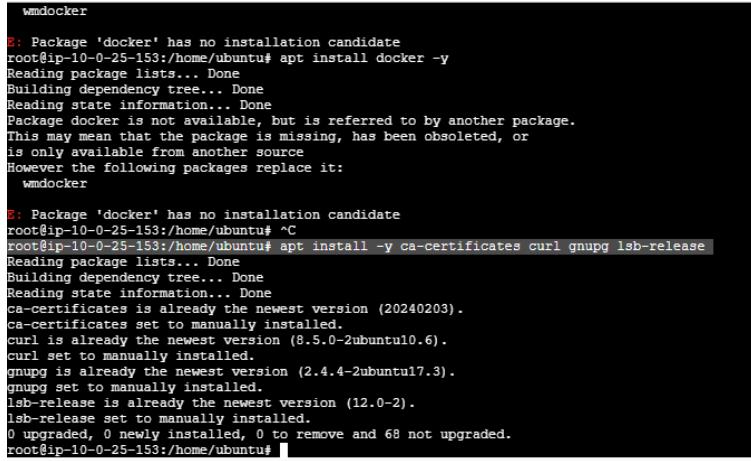
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Sat Dec 20 04:40:25 2025 from 13.233.177.4
ubuntu@ip-10-0-25-153:~/home/ubuntu$ sudo su
root@ip-10-0-25-153:~/home/ubuntu# apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1684 kB]
```

3.2 Install Required Packages

```
sudo apt install -y ca-certificates curl gnupg lsb-release
```

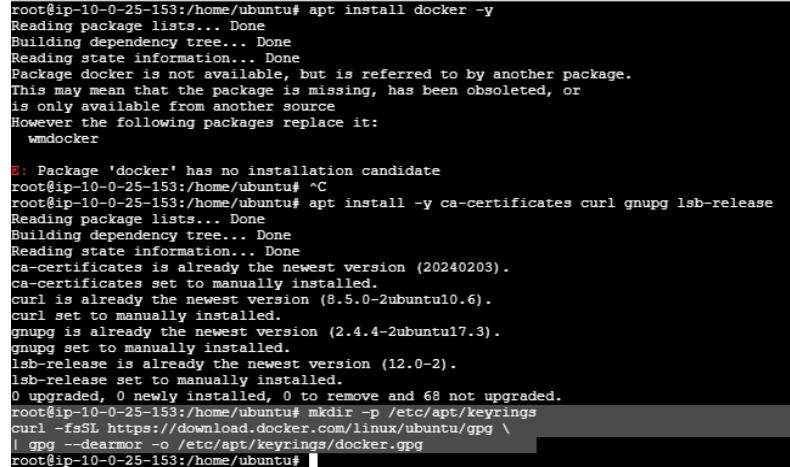


```
wmdocker
E: Package 'docker' has no installation candidate
root@ip-10-0-25-153:/home/ubuntu# apt install docker -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package docker is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source
However the following packages replace it:
  wmdocker

E: Package 'docker' has no installation candidate
root@ip-10-0-25-153:/home/ubuntu# ^C
root@ip-10-0-25-153:/home/ubuntu# apt install -y ca-certificates curl gnupg lsb-release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
gnupg is already the newest version (2.4.4-2ubuntu17.3).
gnupg set to manually installed.
lsb-release is already the newest version (12.0-2).
lsb-release set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 68 not upgraded.
root@ip-10-0-25-153:/home/ubuntu#
```

3.3 Add Docker's Official GPG Key

```
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
| gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```



```
root@ip-10-0-25-153:/home/ubuntu# apt install docker -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package docker is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source
However the following packages replace it:
  wmdocker

E: Package 'docker' has no installation candidate
root@ip-10-0-25-153:/home/ubuntu# ^C
root@ip-10-0-25-153:/home/ubuntu# apt install -y ca-certificates curl gnupg lsb-release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
gnupg is already the newest version (2.4.4-2ubuntu17.3).
gnupg set to manually installed.
lsb-release is already the newest version (12.0-2).
lsb-release set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 68 not upgraded.
root@ip-10-0-25-153:/home/ubuntu# mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
| gpg --dearmor -o /etc/apt/keyrings/docker.gpg
root@ip-10-0-25-153:/home/ubuntu#
```

3.4 Add Docker Repository

```
echo \
"deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.gpg] \
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" \
| tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```

is only available from another source
However the following packages replace it:
  wmdocker

E: Package 'docker' has no installation candidate
root@ip-10-0-25-153:/home/ubuntu# ^C
root@ip-10-0-25-153:/home/ubuntu# apt install -y ca-certificates curl gnupg lsb-release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
gnupg is already the newest version (2.4.4-2ubuntu17.3).
gnupg set to manually installed.
lsb-release is already the newest version (12.0-2).
lsb-release set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 68 not upgraded.
root@ip-10-0-25-153:/home/ubuntu# mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
| gpg --dearmor -o /etc/apt/keyrings/docker.gpg
root@ip-10-0-25-153:/home/ubuntu#
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" \
| tee /etc/apt/sources.list.d/docker.list > /dev/null
root@ip-10-0-25-153:/home/ubuntu# 

```

3.5 Update Package List Again

`apt update`

```

curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
gnupg is already the newest version (2.4.4-2ubuntu17.3).
gnupg set to manually installed.
lsb-release is already the newest version (12.0-2).
lsb-release set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 68 not upgraded.
root@ip-10-0-25-153:/home/ubuntu# mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
| gpg --dearmor -o /etc/apt/keyrings/docker.gpg
root@ip-10-0-25-153:/home/ubuntu#
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" \
| tee /etc/apt/sources.list.d/docker.list > /dev/null
root@ip-10-0-25-153:/home/ubuntu# apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.5 kB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [41.1 kB]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 89.6 kB in 0s (185 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
68 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-10-0-25-153:/home/ubuntu# 

```

3.6 Install Docker Engine (Correct Package)

`apt install -y docker-ce docker-ce-cli containerd.io`

```

lsb-release set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 68 not upgraded.
root@ip-10-0-25-153:/home/ubuntu# mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
| gpg --dearmor -o /etc/apt/keyrings/docker.gpg
root@ip-10-0-25-153:/home/ubuntu#
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" \
| tee /etc/apt/sources.list.d/docker.list > /dev/null
root@ip-10-0-25-153:/home/ubuntu# apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.5 kB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [41.1 kB]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 89.6 kB in 0s (185 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
68 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-10-0-25-153:/home/ubuntu# apt install -y docker-ce docker-ce-cli containerd.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-buildx-plugin docker-ce-rootless-extras docker-compose-plugin libslirp0 pigz slirp4netns
Suggested packages:

```

3.7 Start and Enable Docker

```
systemctl start docker  
systemctl enable docker
```

```
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.  
Setting up docker-compose-plugin (5.0.0-1~ubuntu.24.04-noble) ...  
Setting up docker-ce-cli (5:29.1.3-1~ubuntu.24.04-noble) ...  
Setting up libalilrp0:amd64 (4.7.0-1~ubuntu3) ...  
Setting up pigz (2.8-1) ...  
Setting up docker-ce-rootless-extras (5:29.1.3-1~ubuntu.24.04-noble) ...  
Setting up slirp4netns (1.2.1-1build2) ...  
Setting up docker-ce (5:29.1.3-1~ubuntu.24.04-noble) ...  
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.  
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.  
Processing triggers for man-db (2.12.0-4build2) ...  
Processing triggers for libc-bin (2.39-0ubuntu3.6) ...  
Scanning processes...  
Scanning linux images...  
  
Running kernel seems to be up-to-date.  
  
No services need to be restarted.  
  
No containers need to be restarted.  
  
No user sessions are running outdated binaries.  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
root@ip-10-0-25-153:/home/ubuntu# systemctl start docker  
root@ip-10-0-25-153:/home/ubuntu# systemctl enable docker  
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.  
Executing: /usr/lib/systemd/systemd-sysv-install enable docker  
root@ip-10-0-25-153:/home/ubuntu#
```

Check status:

```
systemctl status docker
```

You should see:

active (running)

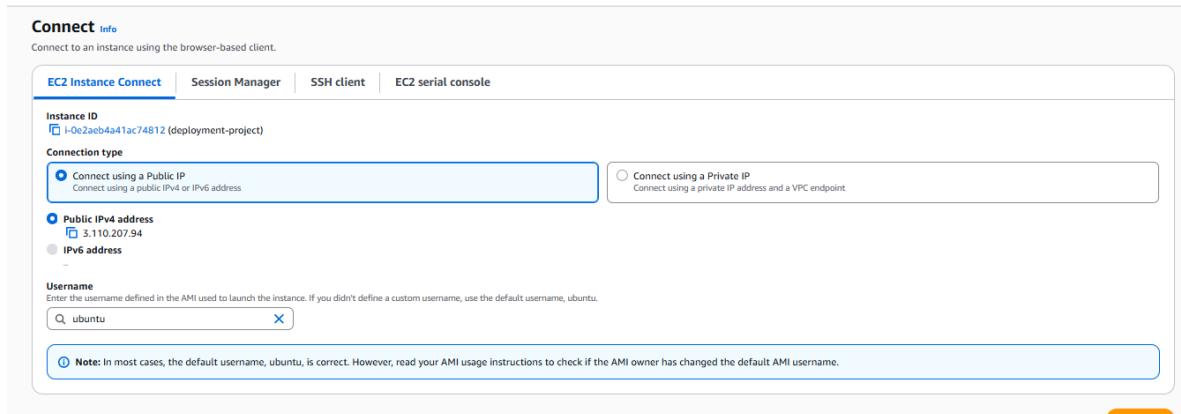
```
No services need to be restarted.  
No containers need to be restarted.  
No user sessions are running outdated binaries.  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
root@ip-10-0-25-153:/home/ubuntu# systemctl start docker  
root@ip-10-0-25-153:/home/ubuntu# systemctl enable docker  
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.  
Executing: /usr/lib/systemd/systemd-sysv-install enable docker  
root@ip-10-0-25-153:/home/ubuntu# systemctl status docker  
● docker.service - Docker Application Container Engine  
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)  
   Active: active (running) since Sat 2025-12-20 05:00:05 UTC; 4min 52s ago  
     Docs: https://docs.docker.com  
          >Main PID: 3402 (dockerd)  
          Tasks: 9  
         Memory: 41.4M (peak: 42.3M)  
            CPU: 528ms  
          CGroup: /system.slice/docker.service  
                  └─3402 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock  
  
Dec 20 05:00:04 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:04.761182295Z" level=info msg="Restoring containers: start."  
Dec 20 05:00:04 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:04.809726955Z" level=info msg="Deleting nftables IPv4 rules" error="exit status 1"  
Dec 20 05:00:04 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:04.818586659Z" level=info msg="Deleting nftables IPv6 rules" error="exit status 1"  
Dec 20 05:00:05 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:05.250529855Z" level=info msg="Loading containers: done."  
Dec 20 05:00:05 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:05.610740122Z" level=info msg="Docker daemon is ready to handle requests from /var/run/dockerd socket"  
Dec 20 05:00:05 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:05.628321097Z" level=info msg="Completed buildkit initialization"  
Dec 20 05:00:05 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:05.343116001Z" level=info msg="Daemon has completed initialization"  
Dec 20 05:00:05 ip-10-0-25-153 systemd[1]: Started docker.service - Docker Application Container Engine.  
Dec 20 05:00:05 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:05.345229359Z" level=info msg="API listen on /run/docker.sock"  
  
root@ip-10-0-25-153:/home/ubuntu# usermod -aG docker ubuntu  
root@ip-10-0-25-153:/home/ubuntu#
```

3.8 Allow Docker Without sudo (VERY IMPORTANT)

```
usermod -aG docker ubuntu
```

```
root@ip-10-0-25-153:/home/ubuntu# systemctl enable docker  
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.  
Executing: /usr/lib/systemd/systemd-sysv-install enable docker  
root@ip-10-0-25-153:/home/ubuntu# systemctl enable docker  
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.  
Executing: /usr/lib/systemd/systemd-sysv-install enable docker  
root@ip-10-0-25-153:/home/ubuntu# systemctl status docker  
● docker.service - Docker Application Container Engine  
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)  
   Active: active (running) since Sun 2025-12-21 05:00:05 UTC; 4min 52s ago  
     Docs: https://docs.docker.com  
          >Main PID: 3402 (dockerd)  
          Tasks: 9  
         Memory: 41.4M (peak: 42.3M)  
            CPU: 528ms  
          CGroup: /system.slice/docker.service  
                  └─3402 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock  
  
Dec 20 05:00:04 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:04.761182295Z" level=info msg="Restoring containers: start."  
Dec 20 05:00:04 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:04.809726955Z" level=info msg="Deleting nftables IPv4 rules" error="exit status 1"  
Dec 20 05:00:04 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:04.818586659Z" level=info msg="Deleting nftables IPv6 rules" error="exit status 1"  
Dec 20 05:00:05 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:05.250529855Z" level=info msg="Loading containers: done."  
Dec 20 05:00:05 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:05.610740122Z" level=info msg="Docker daemon is ready to handle requests from /var/run/dockerd socket"  
Dec 20 05:00:05 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:05.628321097Z" level=info msg="Completed buildkit initialization"  
Dec 20 05:00:05 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:05.343116001Z" level=info msg="Daemon has completed initialization"  
Dec 20 05:00:05 ip-10-0-25-153 systemd[1]: Started docker.service - Docker Application Container Engine.  
Dec 20 05:00:05 ip-10-0-25-153 dockerd[3402]: time="2025-12-20T05:00:05.345229359Z" level=info msg="API listen on /run/docker.sock"  
  
root@ip-10-0-25-153:/home/ubuntu# usermod -aG docker ubuntu  
root@ip-10-0-25-153:/home/ubuntu#
```

Now disconnect and reconnect the EC2 console session.



3.9 Verify Docker Installation

After reconnecting, run:

```
docker --version
```

```
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Sat Dec 20 05:09:40 UTC 2025

System load: 0.13 Temperature: -273.1 C
Usage of /: 34.9% of 6.71GB Processes: 115
Memory usage: 29% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 10.0.25.153

Expanded Security Maintenance for Applications is not enabled.

74 updates can be applied immediately.
28 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sat Dec 20 04:45:34 2025 from 13.233.177.4
ubuntu@ip-10-0-25-153:~$ docker --version
Docker version 29.1.3, build f52814d
ubuntu@ip-10-0-25-153:~$
```

STEP 4: Create REST API Project Directory

```
mkdir rest-api
cd rest-api
```

```
root@ip-10-0-25-153:/home/ubuntu# mkdir rest-api
root@ip-10-0-25-153:/home/ubuntu# cd rest-api/
root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

STEP 5: Create a Simple REST API (Node.js – Industry Standard)

5.1 Install Node.js (Only if not installed)

Check first:

node -v

If not installed:

```
sudo apt install nodejs npm -y
```

```
Scanning linux images... [=====]
Scanning linux images... [=====

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-10-0-25-153:/home/ubuntu/rest-api# apt install nodejs npm -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nodejs is already the newest version (18.19.1+dfsg-6ubuntu5).
npm is already the newest version (9.2.0~ds1-2).
0 upgraded, 0 newly installed, 0 to remove and 44 not upgraded.
root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

5.2 Initialize Node Project

```
npm init -y
```

This creates:

package.json

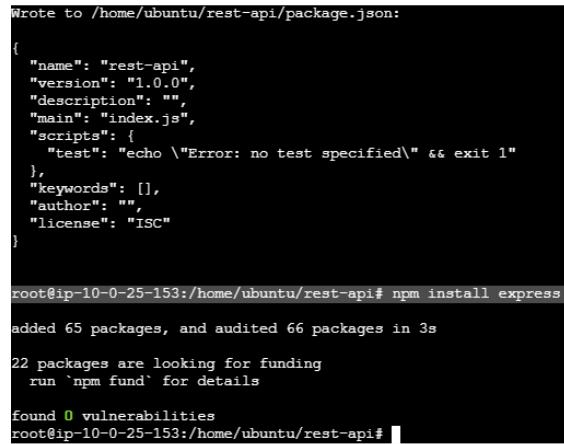
```
root@ip-10-0-25-153:/home/ubuntu/rest-api# apt install nodejs npm -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nodejs is already the newest version (18.19.1+dfsg-6ubuntu5).
npm is already the newest version (9.2.0-ds1-2).
0 upgraded, 0 newly installed, 0 to remove and 44 not upgraded.
root@ip-10-0-25-153:/home/ubuntu/rest-api# npm init -y
Wrote to /home/ubuntu/rest-api/package.json:

{
  "name": "rest-api",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

5.3 Install Express (REST framework)

```
npm install express
```

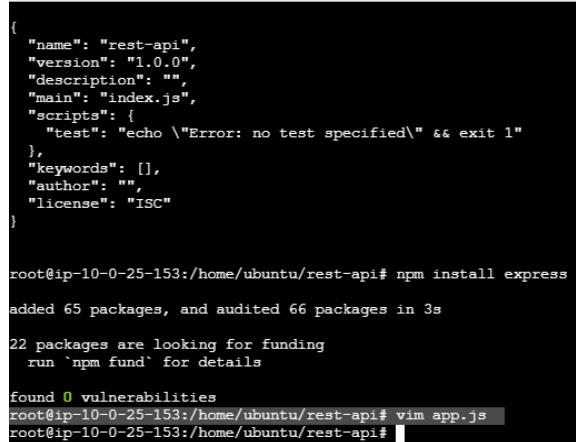


```
Wrote to /home/ubuntu/rest-api/package.json:
{
  "name": "rest-api",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

root@ip-10-0-25-153:/home/ubuntu/rest-api# npm install express
added 65 packages, and audited 66 packages in 3s
22 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

5.4 Create API File

```
vim app.js
```



```
{
  "name": "rest-api",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

root@ip-10-0-25-153:/home/ubuntu/rest-api# npm install express
added 65 packages, and audited 66 packages in 3s
22 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
root@ip-10-0-25-153:/home/ubuntu/rest-api# vim app.js
root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

Paste this **simple REST API**:

```
const express = require("express");
const app = express();

app.use(express.json());

app.get("/", (req, res) => {
  res.json({ message: "REST API is running successfully" });
});
```

```

app.get("/health", (req, res) => {
  res.status(200).json({ status: "UP" });
});

app.get("/api/users", (req, res) => {
  res.json([
    { id: 1, name: "jeevadharsan" },
    { id: 2, name: "user" }
  ]);
});

const PORT = 3000;
app.listen(3000, "0.0.0.0", () => {
  console.log("Server running on port 3000");
});

```

Save & exit.

```

const app = express();
app.use(express.json());

app.get("/", (req, res) => {
  res.json({ message: "REST API is running successfully" });
});

app.get("/health", (req, res) => {
  res.status(200).json({ status: "UP" });
});

app.get("/api/users", (req, res) => {
  res.json([
    { id: 1, name: "Abi" },
    { id: 2, name: "DevOps User" }
  ]);
});

const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});

```

STEP 6: Test REST API Locally (WITHOUT DOCKER)

`node app.js`

In another terminal tab (or stop and use curl):

`curl http://localhost:3000/health`

Expected output:

`{"status": "UP"}`

Stop the app:

Ctrl + C

```
added 65 packages, and audited 66 packages in 3s
22 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
root@ip-10-0-25-153:/home/ubuntu/rest-api# vim app.js
root@ip-10-0-25-153:/home/ubuntu/rest-api# vim app.js
root@ip-10-0-25-153:/home/ubuntu/rest-api# node app.js
Server running on port 3000
curl http://localhost:3000/health

^C
root@ip-10-0-25-153:/home/ubuntu/rest-api# curl http://localhost:3000/health
curl: (7) Failed to connect to localhost port 3000 after 0 ms: Couldn't connect to server
root@ip-10-0-25-153:/home/ubuntu/rest-api# node app.js
Server running on port 3000
q
^Z
[2]+  Stopped                  node app.js
root@ip-10-0-25-153:/home/ubuntu/rest-api# node app.js
Server running on port 3000
root@ip-10-0-25-153:/home/ubuntu/rest-api# curl http://localhost:3000/health
```

We don't get any output in this step, because our security group is not listening to port 3000, it's listening to 80. So, if we have a **server running on port 3000**, it's correct.

STEP 7: Create Dockerfile for REST API

Now we move from **application validation → containerization**

STEP 7.1: Stop the API

Make sure the app is stopped before Docker build:

Ctrl + C

```
System information as of Wed Dec 24 11:36:29 UTC 2025
System load: 0.08      Temperature:      -273.1 C
Usage of /: 65.0% of 6.71GB  Processes:        125
Memory usage: 42%          Users logged in:   1
Swap usage: 0%            IPv4 address for ens5: 10.0.25.153

Expanded Security Maintenance for Applications is not enabled.

44 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Wed Dec 24 11:14:24 2025 from 13.233.177.5
ubuntu@ip-10-0-25-153:~$ sudo su
root@ip-10-0-25-153:/home/ubuntu# cd rest-api/
root@ip-10-0-25-153:/home/ubuntu/rest-api# node app.js &
[1] 10200
root@ip-10-0-25-153:/home/ubuntu/rest-api# Server running on port 3000
```

```
System load: 0.08      Temperature: -273.1 C
Usage of /: 65.0% of 6.71GB  Processes: 125
Memory usage: 42%          Users logged in: 1
Swap usage: 0%            IPv4 address for ens5: 10.0.25.153

Expanded Security Maintenance for Applications is not enabled.

44 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Wed Dec 24 11:14:24 2025 from 13.233.177.5
ubuntu@ip-10-0-25-153:~$ sudo su
root@ip-10-0-25-153:/home/ubuntu# cd rest-api/
root@ip-10-0-25-153:/home/ubuntu/rest-api# node app.js &
[1] 10200
root@ip-10-0-25-153:/home/ubuntu/rest-api# Server running on port 3000
^C
[1]+ Done                  node app.js
root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

STEP 7.2: Create Dockerfile

Ensure you are inside `rest-api` directory:

```
cd ~/rest-api
```

Create Dockerfile:

```
vim Dockerfile
```

```
System load: 0.08      Temperature: -273.1 C
Usage of /: 65.0% of 6.71GB  Processes: 125
Memory usage: 42%          Users logged in: 1
Swap usage: 0%            IPv4 address for ens5: 10.0.25.153

Expanded Security Maintenance for Applications is not enabled.

44 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Wed Dec 24 11:14:24 2025 from 13.233.177.5
ubuntu@ip-10-0-25-153:~$ sudo su
root@ip-10-0-25-153:/home/ubuntu# cd rest-api/
root@ip-10-0-25-153:/home/ubuntu/rest-api# node app.js &
[1] 10200
root@ip-10-0-25-153:/home/ubuntu/rest-api# Server running on port 3000
^C
[1]+ Done                  node app.js
root@ip-10-0-25-153:/home/ubuntu/rest-api# vim Dockerfile
root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

Paste this **exactly**:

```
FROM node:18-alpine
WORKDIR /app

COPY package*.json .
RUN npm install --only=production

COPY . .
```

```
EXPOSE 3000  
CMD [ "node", "app.js" ]
```

Save and exit.

```
FROM node:18-alpine  
  
WORKDIR /app  
  
COPY package*.json ./  
RUN npm install --only=production  
  
COPY . .  
  
EXPOSE 3000  
  
CMD [ "node", "app.js" ]  
  
~
```

STEP 7.3: Build Docker Image

```
docker build -t rest-api:v1 .
```

```
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker build -t rest-api:v1 .  
[*] Building 4.2s (4/9)  
=> [internal] load build definition from Dockerfile  
=> transferring dockerfile: 178B  
=> [internal] load metadata for docker.io/library/node:18-alpine  
=> [internal] load .dockerignore  
=> transferring context: 2B  
=> resolving docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e  
=> sha256:25ff2da83641908f65c3a74d80409db1b62ccfaab220b9ea70b80df5a2e0549 446B / 446B  
=> sha256:1e5a4c89ceec5c0826c540ab06d4bb491c96edca01837ef430bd47f0d26702d6e3 1.26MB / 1.26MB  
=> sha256:f18232174bc91741fd3da96d85011092101a032a93a30a55b79e99e63c235e870 0B / 3.64MB  
=> sha256:dd71dde834b5c203d162902e6b8994cb2309ae049a0eabc4fea161b2b5a3d0e 0B / 40.01MB  
=> [internal] load build context  
=> transferring context: 2.14MB
```

Verify:

```
docker images
```

```
=> extracting sha256:f18232174bc91741fd3da93a30a55b79e99e63c235e870 1.4s  
=> extracting sha256:dd71dde834b5c203d162902e6b8994cb2309ae049a0eabc4fea161b2b5a3d0e 0.1s  
=> extracting sha256:1e5a4c89ceec5c0826c540ab06d4bb491c96edca01837ef430bd47f0d26702d6e3 0.0s  
=> extracting sha256:25ff2da83641908f65c3a74d80409d6b1b62ccfaab220b9ea70b80df5a2e0549 0.3s  
=> [internal] load build context 0.2s  
=> transferring context: 2.14MB 0.2s  
[2/5] WORKDIR /app 0.1s  
[3/5] COPY package*.json ./ 0.1s  
[4/5] RUN npm install --only=production 2.9s  
[5/5] COPY . . 0.3s  
=> exporting to image 0.3s  
=> exporting layers 0.3s  
=> exporting manifest sha256:b967d34b2845c81cceba9990989a57e3580288d2d70907042b9e420cf42640b32 0.0s  
=> exporting config sha256:50a812f20066ea00305e219e72fb0deafac2c65f8964f69fe492377003769bc1 0.0s  
=> exporting attestation manifest sha256:c42b5a89a0f64b8cd68f8be14a6c28a1f367164095e353442101307545c9a7 0.0s  
=> exporting manifest list sha256:bb909131dd4a4583d4a74daf709dc0e81e4317cdc0901cc6985af66b7e74f9f6 0.0s  
=> naming to docker.io/library/rest-api:v1 0.0s  
=> unpacking to docker.io/library/rest-api:v1 0.3s  
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker images  
[ Info - In Use ]  
IMAGE ID          REPOSITORY          TAG      IMAGE SIZE  EXTRAS  
333673271195.dkr.ecr.ap-south-1.amazonaws.com/frontend-aitech:latest b5e16a3bbfcfcb 131MB 47.4MB U  
frontend-aitech:latest b5e16a3bbfcfcb 131MB 47.4MB U  
rest-api:v1 bb909131dd4a 194MB 46.9MB U  
root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

You should see:

```
rest-api    v1
```

STEP 7.4: Run Container Locally

```
docker run -d -p 3000:3000 rest-api:v1
```

When you run the above command, you'll get an error sometimes. Verify the below image, if there is any error.

```
docker: Error response from daemon: failed to set up container networking: driver failed programming external connectivity on endpoint wonderful_bouman (0182b6e183a/f27c92606277552f403385e380702672d465df9ba18a7c10d34): failed to bind host port 0.0.0.0:3000/tcp: address already in use
Run 'docker run --help' for more information
root@ip-10-0-25-153:/home/ubuntu/rest-api# sudo lsof -i :3000
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 9786 root 25u IPv6 27143 0t0 TCP *:3000 (LISTEN)
curl 9809 root 5u IPv6 29561 0t0 TCP ip6-localhost:46526->ip6-localhost:3000 (ESTABLISHED)
curl 9820 root 5u IPv6 27184 0t0 TCP ip6-localhost:56380->ip6-localhost:3000 (ESTABLISHED)
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
root@ip-10-0-25-153:/home/ubuntu/rest-api# pkill node
root@ip-10-0-25-153:/home/ubuntu/rest-api# sudo lsof -i :3000
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 9786 root 25u IPv6 27143 0t0 TCP *:3000 (LISTEN)
curl 9809 root 5u IPv6 29561 0t0 TCP ip6-localhost:46526->ip6-localhost:3000 (ESTABLISHED)
curl 9820 root 5u IPv6 27184 0t0 TCP ip6-localhost:56380->ip6-localhost:3000 (ESTABLISHED)
root@ip-10-0-25-153:/home/ubuntu/rest-api# ls -l /var/run/docker.sock
root@ip-10-0-25-153:/home/ubuntu/rest-api# kill -9 9786
root@ip-10-0-25-153:/home/ubuntu/rest-api# sudo lsof -i :3000
root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

In this, kill -9 <PID>.

Again, run this command:

```
docker run -d -p 3000:3000 rest-api:v1
```

Verify container:

```
docker ps
```

```
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 9786 root 25u IPv6 27143 0t0 TCP *:3000 (LISTEN)
curl 9809 root 5u IPv6 29561 0t0 TCP ip6-localhost:46526->ip6-localhost:3000 (ESTABLISHED)
curl 9820 root 5u IPv6 27184 0t0 TCP ip6-localhost:56380->ip6-localhost:3000 (ESTABLISHED)
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
root@ip-10-0-25-153:/home/ubuntu/rest-api# pkill node
root@ip-10-0-25-153:/home/ubuntu/rest-api# sudo lsof -i :3000
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 9786 root 25u IPv6 27143 0t0 TCP *:3000 (LISTEN)
curl 9809 root 5u IPv6 29561 0t0 TCP ip6-localhost:46526->ip6-localhost:3000 (ESTABLISHED)
curl 9820 root 5u IPv6 27184 0t0 TCP ip6-localhost:56380->ip6-localhost:3000 (ESTABLISHED)
root@ip-10-0-25-153:/home/ubuntu/rest-api# ls -l /var/run/docker.sock
root@ip-10-0-25-153:/home/ubuntu/rest-api# kill -9 9786
root@ip-10-0-25-153:/home/ubuntu/rest-api# sudo lsof -i :3000
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker run -d -p 3000:3000 rest-api:v1
c6707c53ca3ea4affb95163df2ce882bc1d214f04c07286ab4671426e9029e
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
c6707c53ca3e "docker-entrypoint.s..." 6 seconds ago Up 6 seconds 0.0.0.0:3000->3000/tcp, [:]:3000->3000/tcp dazzling_faraday
root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

STEP 7.5: Test Containerized API

```
curl http://localhost:3000/health
```

Expected:

```
{"status": "UP"}
```

```

node 9786 root 25u IPv6 27143 0t0 TCP *:3000 (LISTEN)
curl 9809 root 5u IPv6 29561 0t0 TCP ip6-localhost:46526->ip6-localhost:3000 (ESTABLISHED)
curl 9820 root 5u IPv6 27184 0t0 TCP ip6-localhost:56380->ip6-localhost:3000 (ESTABLISHED)
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
root@ip-10-0-25-153:/home/ubuntu/rest-api# kill node
root@ip-10-0-25-153:/home/ubuntu/rest-api# sudo lsof -i :3000
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 9786 root 25u IPv6 27143 0t0 TCP *:3000 (LISTEN)
curl 9809 root 5u IPv6 29561 0t0 TCP ip6-localhost:46526->ip6-localhost:3000 (ESTABLISHED)
curl 9820 root 5u IPv6 27184 0t0 TCP ip6-localhost:56380->ip6-localhost:3000 (ESTABLISHED)
root@ip-10-0-25-153:/home/ubuntu/rest-api# lsof -i :3000
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 9786 root 25u IPv6 27143 0t0 TCP *:3000 (LISTEN)
curl 9809 root 5u IPv6 29561 0t0 TCP ip6-localhost:46526->ip6-localhost:3000 (ESTABLISHED)
curl 9820 root 5u IPv6 27184 0t0 TCP ip6-localhost:56380->ip6-localhost:3000 (ESTABLISHED)
root@ip-10-0-25-153:/home/ubuntu/rest-api# kill -9 9786
root@ip-10-0-25-153:/home/ubuntu/rest-api# sudo lsof -i :3000
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker run -d -p 3000:3000 rest-api:v1
c6707c53ca3ea4fffb95163df2ce82bc1d2124f04c0728eb4671426e9029e
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
c6707c53ca3ea4fffb95163df2ce82bc1d2124f04c0728eb4671426e9029e
rest-api:v1 "docker-entrypoint.s..." 6 seconds ago Up 6 seconds 0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp dazzling_faraday
root@ip-10-0-25-153:/home/ubuntu/rest-api# curl http://localhost:3000/health
[{"status": "UP"}]

```

STEP 8: Push REST API Docker Image to Amazon ECR

Now we move from **local container** → **AWS-managed container registry**.

This step is **mandatory** for ECS Fargate.

STEP 8.1: Create ECR Repository (REST API)

AWS Console → **ECR** → **Create repository**

Repository name:

rest-api

- Visibility: Private
- Encryption: Default
- Click **Create repository**

Copy the repository URI, it will look like:

333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest-api

Private repositories (1)					
<input type="text"/> Search by repository substring					
Repository name		URI	Created at	Tag immutability	Encryption type
<input type="radio"/>	frontend-aitech	333673271195.dkr.ecr.ap-south-1.amazonaws.com/fron tend-aitech	20 December 2025, 12:27:07 (UTC+05:5)	Mutable	AES-256

Create a private repository

General settings

Repository name

Enter a concise name. Repositories support namespaces, which you can use to group similar repositories.

333673271195.dkr.ecr.ap-south-1.amazonaws.com/ rest-api

8 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers and special characters _-./.

Image tag settings Info

Image tag mutability

Choose the tag mutability setting.

Mutable

Image tags can be overwritten.

Immutable

Image tags can't be overwritten.

Mutable tag exclusions

Tags that match these filters will be immutable (can't be overwritten). Using wildcards (*) will match zero or more image tag characters.

[Add filter](#)

Encryption settings Info

⚠️ The encryption settings for a repository can't be changed once the repository is created.

Encryption configuration

By default, repositories use the industry standard Advanced Encryption Standard (AES) encryption. You can optionally choose to use a key stored in the AWS Key Management Service (KMS) to encrypt the images in your repository.

AES-256

Industry standard Advanced Encryption Standard (AES) encryption

AWS KMS

AWS Key Management Service (KMS)

▶ Image scanning settings - deprecated

[Cancel](#) [Create](#)

Successfully created private repository, rest-api					
Private repositories (2)					
Repository name		URI	Created at	Tag immutability	Encryption type
<input type="radio"/>	frontend-aitech	333673271195.dkr.ecr.ap-south-1.amazonaws.com/fron tend-aitech	20 December 2025, 12:27:07 (UTC+05:5)	Mutable	AES-256
<input checked="" type="radio"/>	rest-api	333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest- api	24 December 2025, 17:37:06 (UTC+05:5)	Mutable	AES-256

STEP 8.2: Authenticate Docker to ECR

Run on EC2:

```
aws ecr get-login-password --region ap-south-1 \
| docker login --username AWS --password-stdin
333673271195.dkr.ecr.ap-south-1.amazonaws.com
```

Expected:

Login Succeeded

```
curl 9809 root 5u IPv6 29561 0t0 TCP ip6-localhost:46526->ip6-localhost:3000 (ESTABLISHED)
curl 9820 root 5u IPv6 27184 0t0 TCP ip6-localhost:56380->ip6-localhost:3000 (ESTABLISHED)
root@ip-10-0-25-153:/home/ubuntu/rest-api# lsof -i :3000
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
node 9786 root 25u IPv6 27143 0t0 TCP *:3000 (LISTEN)
curl 9809 root 5u IPv6 29561 0t0 TCP ip6-localhost:46526->ip6-localhost:3000 (ESTABLISHED)
curl 9820 root 5u IPv6 27184 0t0 TCP ip6-localhost:56380->ip6-localhost:3000 (ESTABLISHED)
root@ip-10-0-25-153:/home/ubuntu/rest-api# kill -9 9786
root@ip-10-0-25-153:/home/ubuntu/rest-api# sudo lsof -i :3000
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker run -d -p 3000:3000 rest-api:v1
c6707c53ca3ea4fffb95163df2ce882bc1d214f04c07286ab4671426e9029e
root@ip-10-0-25-153:/home/ubuntu/rest-api# curl http://localhost:3000/health
{"status": "UP"}root@ip-10-0-25-153:/home/ubuntu/.aws/ aws ecr get-login-password --region ap-south-1 < /dev/null
| docker login --username AWS --password-stdin 333673271195.dkr.ecr.ap-south-1.amazonaws.com

WARNING! Your credentials are stored unencrypted in '/root/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

STEP 8.3: Tag REST API Image

```
docker tag rest-api:v1 \
333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest-api:latest
```

Verify:

docker images

```
root@ip-10-0-25-153:/home/ubuntu/rest-api# sudo lsof -i :3000
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker run -d -p 3000:3000 rest-api:v1
c6707c53ca3ea4fffb95163df2ce882bc1d214f04c07286ab4671426e9029e
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
c6707c53ca3e "docker-entrypoint.s..." 6 seconds ago Up 6 seconds 0.0.0.0:3000->3000/tcp, [:]:3000->3000/tcp dazzling_faraday
root@ip-10-0-25-153:/home/ubuntu/rest-api# curl http://localhost:3000/health
{"status": "UP"}root@ip-10-0-25-153:/home/ubuntu/.aws/ aws ecr get-login-password --region ap-south-1 < /dev/null
| docker login --username AWS --password-stdin 333673271195.dkr.ecr.ap-south-1.amazonaws.com

WARNING! Your credentials are stored unencrypted in '/root/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker tag rest-api:v1 \
333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest-api:latest
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker images



| IMAGE                                                                 | ID            | DISK  | USAGE  | CONTENT SIZE | EXTRA |
|-----------------------------------------------------------------------|---------------|-------|--------|--------------|-------|
| 333673271195.dkr.ecr.ap-south-1.amazonaws.com/frontend-alitech:latest | b5e16a3bbfcfb | 131MB | 47.4MB | 46.9MB       | U     |
| 333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest-api:latest         | bb909131ddda4 | 194MB | 46.9MB | 46.9MB       | U     |
| frontend-alitech:latest                                               | b5e16a3bbfcfb | 131MB | 47.4MB | 46.9MB       | U     |
| rest-api:latest                                                       | bb909131ddda4 | 194MB | 46.9MB | 46.9MB       | U     |


root@ip-10-0-25-153:/home/ubuntu/rest-api#
```

STEP 8.4: Push Image to ECR

docker push

```
333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest-api:latest
```

```

https://docs.docker.com/go/credential-store/
Login Succeeded
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker tag rest-api:v1 \
333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest-api:latest
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker images
IMAGE          ID      DISK USAGE   CONTENT SIZE  EXTRA
333673271195.dkr.ecr.ap-south-1.amazonaws.com/frontend-aitech:latest  b5e16a3bbfcf  131MB     47.4MB    U
333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest-api:latest        bb90913ldda4  194MB     46.9MB    U
frontend-aitech:latest          b5e16a3bbfcf  131MB     47.4MB    U
rest-api:v1                    bb90913ldda4  194MB     46.9MB    U
root@ip-10-0-25-153:/home/ubuntu/rest-api# docker push 333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest-api:latest
The push refers to repository [333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest-api]
36124f414498: Pushed
11b279d0efc34: Pushed
f18232174bc9: Pushed
dd71dde834b5: Pushed
ba40c37a9e9c9: Pushed
25ff2da83641: Pushed
b83a1367db4e: Pushed
c22ed34e5763: Pushed
1e5a4c89ceec5: Pushed
latest: digest: sha256:bb909131dd44583d4a74daf0b709dcec81e4317cdc0901cc6985af66b7e74f9f size: 856
root@ip-10-0-25-153:/home/ubuntu/rest-api#


```

STEP 8.5: Verify in AWS Console

Go to:

ECR → rest-api → Images

You should see:

- Image tag: `latest`
- Status: Available

Image tags	Type	Created at	Image size	Image digest	Last pulled at
latest	Image Index	24 December 2025, 17:43:01 (UTC+05:5)	46.90	sha256:bb909131dd44583d4a74daf0b709dcec81e4317cdc0901cc6985af66b7e74f9f	-
-	Image	24 December 2025, 17:43:01 (UTC+05:5)	0.00	sha256:c42b5ab...	-
-	Image	24 December 2025, 17:43:01 (UTC+05:5)	46.90	sha256:b967d34...	-

STEP 9: Create ECS Task Definition (REST API)

The **task definition** is the **blueprint** that tells ECS:

- Which image to run

- Which port to expose
- How much CPU/RAM to allocate
- How logs are handled

The screenshots show the 'Create cluster' wizard in the AWS ECS console. The first two steps are visible, and the third step shows the successful creation of the cluster.

Step 1: Cluster configuration

Cluster name: rest-api

Service Connect defaults – optional:

Infrastructure – advanced

Select a method of obtaining compute capacity:

- Fargate only: Serverless – you don't think about creating or managing servers. Great for most common workloads.
- Fargate and managed instances: Managed instances - Amazon ECS will manage patching and scaling on your behalf while giving you configurability about the types of instances. Great for more advanced workloads.
- Fargate and Self-managed instances: Self-managed instances - you must ensure the instances are patched and scaled properly, and you have full control over the instances.

Monitoring – optional:

Encryption – optional:

Tags – optional

Create

Step 2: Infrastructure – advanced

Select a method of obtaining compute capacity:

- Fargate only: Serverless – you don't think about creating or managing servers. Great for most common workloads.
- Fargate and managed instances: Managed instances - Amazon ECS will manage patching and scaling on your behalf while giving you configurability about the types of instances. Great for more advanced workloads.
- Fargate and Self-managed instances: Self-managed instances - you must ensure the instances are patched and scaled properly, and you have full control over the instances.

Monitoring – optional:

Encryption – optional:

Tags – optional

Create

Step 3: Confirmation

Cluster rest-api has been created successfully.

Clusters (2)

Cluster	Services	Tasks	Container Instances	CloudWatch monitoring	Capacity provi
frontend-aitech-cluster	1	0 pending 1 running	0 EC2	Default	No default four
rest-api	0	No tasks running	0 EC2	Default	No default four

View cluster **Create cluster**

STEP 9.1: Open Task Definitions

AWS Console → **ECS** → **Task definitions** → **Create new task definition**

Choose:

- **Launch type:** **AWS Fargate**

STEP 9.2: Task Definition Configuration

Setting	Value
Task definition family	rest-api-task
Launch type	AWS Fargate
Operating system	Linux
Architecture	x86_64
Network mode	awsvpc

STEP 9.3: Task Size

Resource	Value
CPU	0.25 vCPU
Memory	0.5 GB

(Perfect for a lightweight REST API)

STEP 9.4: Task Execution Role

When prompted:

- ✓ Select **Create new role**
(or)
- ✓ Select **ecsTaskExecutionRole** (recommended)

This role allows ECS to:

- Pull image from **ECR**
- Send logs to **CloudWatch**

STEP 9.5: Container Configuration

Click **Add container**

Field	Value
Container name	rest-api
Image URI	333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest-api:latest
Essential	Yes

Port Mapping

Setting	Value
Container port	3000
Protocol	TCP

Amazon Elastic Container Service > Create new task definition

Container – 1 Info

Container details
Specify a name, container image and whether the container should be marked as essential. Each task definition must have at least one essential container.

Name rest-api **Essential container** Yes

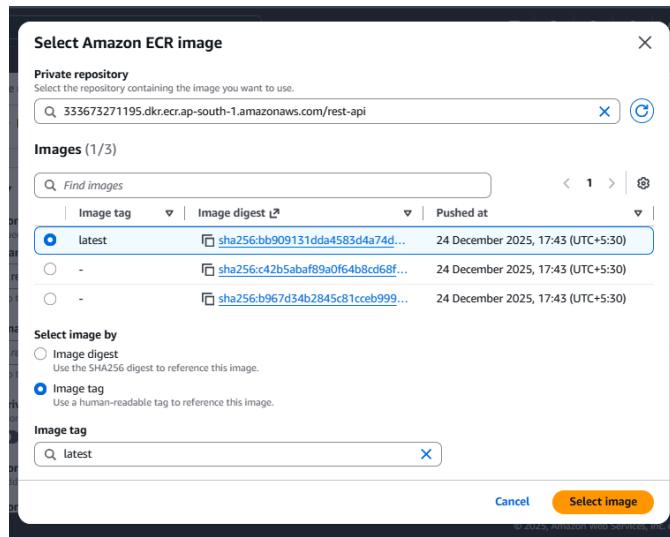
Image URI 333673271195.dkr.ecr.ap-south-1.amazonaws.com/rest-api:latest **Browse ECR images**

Private registry Info
Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.
 Private registry authentication

Port mappings Info
Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol
80	TCP	container-port-protocol	HTTP

Documentation Discover products Subscriptions



Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol
80	TCP	container-port-protocol	HTTP
3000	TCP	container-port-protocol	HTTP

Add port mapping

Read-only root file system Info
When this parameter is turned on, the container is given read-only access to its root file system.
 Read only

Resource allocation limits - conditional Info
Container-level CPU, GPU and memory limits are different from task-level values. They define how many resources are allocated for the container. If the container attempts to exceed the memory specified by the hard limit, the container is terminated.

CPU	GPU	Memory hard limit	Memory soft limit
1 in vCPU	1	3 in GB	1 in GB

Environment variables - optional

Environment variables Info
Add individual...

Amazon Elastic Container Service > Create new task definition

Express Mode Clusters Namespaces Task definitions Account settings

Amazon ECR Repositories

AWS Batch Documentation Discover products Subscriptions

STEP 9.6: Logging (IMPORTANT)

Enable CloudWatch Logs

Setting	Value
Log group	/ecs/rest-api-task

Region	ap-south-1
Stream prefix	ecs
Auto-create group	Yes

The screenshot shows the 'Log collection' section of the AWS CloudWatch Log Configuration interface. It includes a table for defining log collection rules:

Key	Value type	Value
awslogs-group	Value	/ecs/rest-api-task
awslogs-region	Value	ap-south-1
awslogs-stream-prefix	Value	ecs
awslogs-create-group	Value	true

Below the table are buttons for 'Add log configuration option' and 'Remove'. A note about a 'Restart policy - optional' is also present.

The screenshot shows the 'Task definitions' section of the AWS Task Definition configuration interface. It includes sections for 'Storage - optional' and 'Monitoring - optional'.

Storage - optional:

- Ephemeral storage**: An input field set to 21.
- Volumes**: Buttons for 'Add volume' and 'Volumes from'.

Monitoring - optional:

- A note: 'Mount data volumes from another container.'
- Buttons for 'Add volume from'.

STEP 9.7: Create Task Definition

Click **Create**

✓ You should now see:

`rest-api-task : Revision 1`

STEP 10: Create ECS Service (REST API) + Attach to ALB

The **ECS Service**:

- Keeps your REST API running
- Replaces crashed containers automatically
- Connects your API to the **Application Load Balancer**

STEP 10.1: Create ECS Service

Go to:

AWS Console → ECS → Clusters → *your cluster* → Create service

Choose:

Setting	Value
Launch type	Fargate
Task definition family	rest-api-task
Revision	Latest
Service name	rest-api-service

Desired tasks | 1

Service name	ARN	Status	Schedule...	Last updated	Actions
rest-api-task		Active		24 December 2025, 18:12 (UTC+5:30)	Manage tags Update Delete service Create

Create service

Service details

Task definition family
Select an existing task definition. To create a new task definition, go to [Task definitions](#).
rest-api-task

Task definition revision [Latest]
Select the task definition revision from the 100 most recent entries, or enter a revision. Leave the field blank to use the latest revision.
Q 1

Service name
Assign a service name that is unique for this cluster.
rest-api-task-service-x8czm65h

Environment
Existing cluster

Environment
Existing cluster

Compute configuration - advanced

Compute options [Info]
To ensure task distribution across your compute types, use appropriate compute options.

Capacity provider strategy [Info]
Specify a launch strategy to distribute your tasks across one or more capacity providers.

Launch type
Launch tasks directly without the use of a capacity provider strategy.

Capacity provider strategy [Info]
Select either your cluster default capacity provider strategy or select the customised option to configure a different strategy.

Use cluster default
No default capacity provider strategy configured for this cluster.

Use custom (Advanced)

Capacity provider

Base	Weight	
FARGATE	0	1

The image contains two side-by-side screenshots of the AWS Elastic Container Service (ECS) console, specifically focusing on deployment configuration.

Screenshot 1 (Top): Deployment Configuration - Scheduling Strategy

- Capacity provider:** FARGATE (Base: 0, Weight: 1)
- Platform version:** LATEST
- Troubleshooting configuration - recommended:** A link to troubleshooting information.
- Deployment configuration - Scheduling strategy:**
 - Replica (selected):** Place and maintain a desired number of tasks across your cluster.
 - Daemon:** Place and maintain one copy of your task on each container instance.

Screenshot 2 (Bottom): Deployment Configuration - Desired Tasks

- Desired tasks:** 1
- Availability Zone re-balancing:** Turn on Availability Zone re-balancing (checkbox checked).
- Health check grace period:** 0 seconds
- Deployment options:** A link to deployment options.

Left sidebar (common to both):

- Amazon Elastic Container Service (selected)
- Express Mode
- Clusters** (selected)
- Namespaces
- Task definitions
- Account settings
- Amazon ECR
- Repositories
- AWS Batch
- Documentation
- Discover products
- Subscriptions

STEP 10.2: Networking Configuration

Setting	Value
VPC	deployment-project-VPC
Subnets	Public subnets (same as ALB)
Security group	deployment-project-sg
Auto-assign public IP	Enabled

Make sure:

- Subnets are **public**
- Security group allows **port 3000** from ALB SG

The screenshot shows the AWS CloudFormation template editor with the 'Networking' section open. On the left, a sidebar lists navigation options: Express Mode, Clusters, Namespaces, Task definitions, Account settings, Amazon ECR, Repositories, AWS Batch, Documentation, Discover products, and Subscriptions. The main area displays configuration for a VPC:

- VPC**: A dropdown menu set to 'vpc-0bba916360bb4e70d deployment-project-VPC'. To its right is a 'Create a new VPC' button.
- Subnets**: A list of two subnets: 'subnet-03f425e660184ff8b deployment-project-VPC-subnet-public2-ap-south-1b 10.0.16.0/20' and 'subnet-0624ce757b655fff9 deployment-project-VPC-subnet-public1-ap-south-1a 10.0.0.0/20'. Below this is a 'Clear current selection' button.
- Security group**: A section with 'Use an existing security group' selected (radio button). It shows two security groups: 'sg-0bd44cf7bd5b0a40d deployment-project-sg' and 'sg-0dcc14845b66358c6 default'.
- Security group name**: A dropdown menu labeled 'Choose security groups'.
- Public IP**: A section with 'Turned on' selected (radio button).
- Service Connect - optional**: A note stating 'Service Connect allows for service-to-service communications with automatic discovery using short names and standard ports.'
- Service discovery - optional**: A note stating 'Service discovery uses Amazon Route 53 to create a namespace for your service, which allows it to be discoverable via DNS.'

STEP 10.3: Load Balancer Configuration (IMPORTANT)

Enable load balancing.

Choose:

- **Application Load Balancer**
- Select existing ALB:
frontend-aitech-alb

The screenshot shows the AWS CloudFormation template editor with the 'Load balancing - optional' section open. On the left, a sidebar lists navigation options: Express Mode, Clusters, Namespaces, Task definitions, Account settings, Amazon ECR, Repositories, AWS Batch, Documentation, Discover products, and Subscriptions. The main area displays configuration for load balancing:

- Use load balancing**: A checked checkbox.
- VPC**: A dropdown menu set to 'vpc-0bba916360bb4e70d'.
- Load balancer type**: A section with 'Application Load Balancer' selected (radio button). It describes an Application Load Balancer making routing decisions at the application layer (HTTP/HTTPS), supporting path-based routing, and can route requests to one or more ports. The 'Network Load Balancer' option is also listed.
- Container**: A section for specifying the container and port to load balance the incoming traffic to. It shows 'rest-api 3000:3000'.
- Host port:Container port**: A dropdown menu set to 'rest-api 3000:3000'.
- Application Load Balancer**: A section for creating a new load balancer or choosing an existing one.

STEP 10.4: Target Group Configuration

Setting	Value
Target type	IP
Target group name	rest-api-tg
Protocol	HTTP
Port	3000
Health check path	/health

- ✓ This is critical — your API health endpoint is `/health`

The screenshot shows two pages of the AWS ECS console side-by-side.

Left Page (Load Balancer Configuration):

- Load balancer:** Choose an existing load balancer to distribute traffic. View existing load balancers and create new one in EC2 Console.
- Listener:** Specify the port and protocol that the load balancer will listen for connection requests on.
 - Create new listener
 - Use an existing listener (selected)
 - Protocol: HTTP-80
- Listener rules for 80:HTTP (1):** Traffic received by the listener is routed according to its rules. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last.

Priority	Rule path	Target group
default	/	frontend-aitech-tg
- Target group:** Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.
 - Create new target group
 - Target group name: frontend-aitech-tg

Right Page (Target Group Configuration):

- Target group:** Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.
 - Create new target group
 - Target group name: rest-api-tg
- Protocol:** HTTP
- Port:** 3000
- Deregistration delay:** The amount of time to wait before the state of a deregistering target changes from draining to unused.
 - 300 seconds
- Health check protocol:** HTTP

The screenshot shows the AWS Lambda service configuration page. On the left, a sidebar lists navigation options: Express Mode, Clusters, Namespaces, Task definitions, Account settings, Amazon ECR, Repositories, AWS Batch, Documentation, Discover products, and Subscriptions. The main content area displays function details: Protocol (HTTP), Port (3000), Deregistration delay (300 seconds), Path pattern (/api*) with Priority 1, Health check protocol (HTTP), and Health check path (/health). Below this, two optional features are shown in callout boxes: VPC Lattice (fully managed application networking service) and Service auto scaling (automatically adjust service's desired count up and down based on CloudWatch alarms).

STEP 10.6: Create Service

Click **Create service**

Wait until:

- Desired tasks: 1
- Running tasks: 1
- Health status: **Healthy**

The screenshot shows the AWS ECS console. On the left, there's a sidebar with options like Express Mode, Clusters (which is selected), Namespaces, Task definitions, Account settings, Amazon ECR, Repositories, AWS Batch, Documentation, and Discover products. The main area displays a service named 'rest-api-task-service-x8czm65h'. It has one task listed under the 'Tasks' section, which is currently running. The 'Services' section shows the service is active. Below the main table, there are tabs for Services, Tasks, Infrastructure, Metrics, Scheduled tasks, Configuration, Event history, and Tags. At the top right, there are buttons for Manage tags, Update, Delete service, and Create.

✓ VERIFY DEPLOYMENT

1 Get ALB DNS name

frontend-aitech-alb-xxxx.ap-south-1.elb.amazonaws.com

2 Test REST API via ALB

<http://<ALB-DNS>/api/health>

Expected output:

```
{"status": "UP"}
```