

Deploy a Static Web App on AWS with Docker, Amazon ECR, and ECS

Deploying a static web app on AWS using Docker, Amazon Elastic Container Registry (ECR), and Amazon Elastic Container Service (ECS) provides scalability, reliability, and efficiency in managing your application.

When inputting the below docker syntax to build the container image, I came across the following error ‘failed to authorise’. After a bit of research, I noticed that my docker credentials in VSCode were not up to date.

```
Waiting for authentication in the browser...
Error response from daemon: Get "https://registry-1.docker.io/v2/": unauthorized: incorrect username or password
PS C:\Users\8rooo\Desktop\docker-projects\Jupiter-website> docker build -t jupiter .
[+] Building 0.9s (3/3) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 714B
=> ERROR [internal] load metadata for docker.io/library/amazonlinux:2
=> [auth] library/amazonlinux:pull token for registry-1.docker.io
-----
> [internal] load metadata for docker.io/library/amazonlinux:2:
-----
Dockerfile:1
-----
1 | >>> FROM amazonlinux:2
2 |
3 |     # Install dependencies
-----
ERROR: failed to solve: amazonlinux:2: failed to resolve source metadata for docker.io/library/amazonlinux:2: failed to a
https://auth.docker.io/token?scope=repository%3Alibrary%2Famazonlinux%3Apull&service=registry.docker.io: 401 Unauthorized
PS C:\Users\8rooo\Desktop\docker-projects\Jupiter-website>
```

To resolve this, I had to log into docker via the docker extension in VSCODE. Please do the following:

- In the Docker extension panel, look for the "**Registries**" section. It's typically near the bottom of the Docker sidebar.
- Hover over the "**Registries**" heading and click the **plus (+) icon** that appears.
- Select "**Docker Hub**" from the dropdown menu.
- Enter your Docker Hub **username** and press Enter.
- Enter your Docker Hub **password**

Now your credentials have been updated you can rerun the following command ‘`docker build -t Jupiter .`’ You should have the following output:

```

6  EXPOSE 80
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
RUNNING: daemon is not using the default seccomp profile
5 C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> docker build -t jupiter .
+] Building 84.5s (11/11) FINISHED
-> [internal] load build definition from Dockerfile
-> > transferring dockerfile: 714B
-> [internal] load metadata for docker.io/library/amazonlinux:2
-> [internal] load .dockerignore
-> > transferring context: 2B
-> [1/7] FROM docker.io/library/amazonlinux:2@sha256:e43d4316208b9ef2ddbe5e5b605c944073b76292d73c2f5868312d25bc1e6c
-> > > resolving docker.io/library/amazonlinux:2@sha256:e43d4316208b9ef2ddbe5e5b605c944073b76292d73c2f5868312d25bc1e6c
-> > sha256:899946e4a240e49763e4d2464f581b66a1bd29bf9f38cc3e3503b261565914b88e62_689B / 62_689B
-> > extracting sha256:899946e4a240e49763e4d2464f581b66a1bd29bf9f38cc3e3503b261565914b88e62_689B
-> > > yum update -y && yum install httpd && yum search wget && yum install wget -y && yum install unzip -y
-> [3/7] RUN cd /var/www/html
-> [4/7] RUN wget https://github.com/azeezsalu/jupyter/archive/refs/heads/main.zip
-> [5/7] RUN unzip main.zip
-> [6/7] RUN cp -r jupyter-main/* /var/www/html/
-> [7/7] RUN rm -rf jupyter-main main.zip
-> exporting to image
-> > exporting layers
-> > > exporting config sha256:5761f7c4ed13935c38814e828f280690332bc3f661d68f983e72163fc68752a
-> > > exporting config sha256:948779cd1bf5e637882342fb773b1c2cb26c32ea3f99d574a23b87598f624e6
-> > > exporting attestation manifest sha256:c28de1e322301164f38403c3e3503b261565914b88e62_689B / 62_689B
-> > > exporting manifest list sha256:f700889902ce9488ac5c420ec7a80c8eebec207f3cc949ff88c037e77ee0300d
-> > > naming to docker.io/library/jupyter:latest
-> > unpacking to docker.io/library/jupyter:latest
5 C:\Users\Brooo\Desktop\docker-projects\Jupiter-website>
6 C:\Users\Brooo\Desktop\docker-projects\Jupiter-website>
```

Based on the dockerfile I put together you can see all the steps the file went through. To check the image created you can type the following command in the terminal ‘docker image ls’.

```

PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website>
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website>
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> docker image ls
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
jupiter        latest       f7d0889902ce  About an hour ago  965MB
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> []
```

Now I've have build the docker image, we are going to run image as a container via port 80 which will allow us to access the site via the internet.

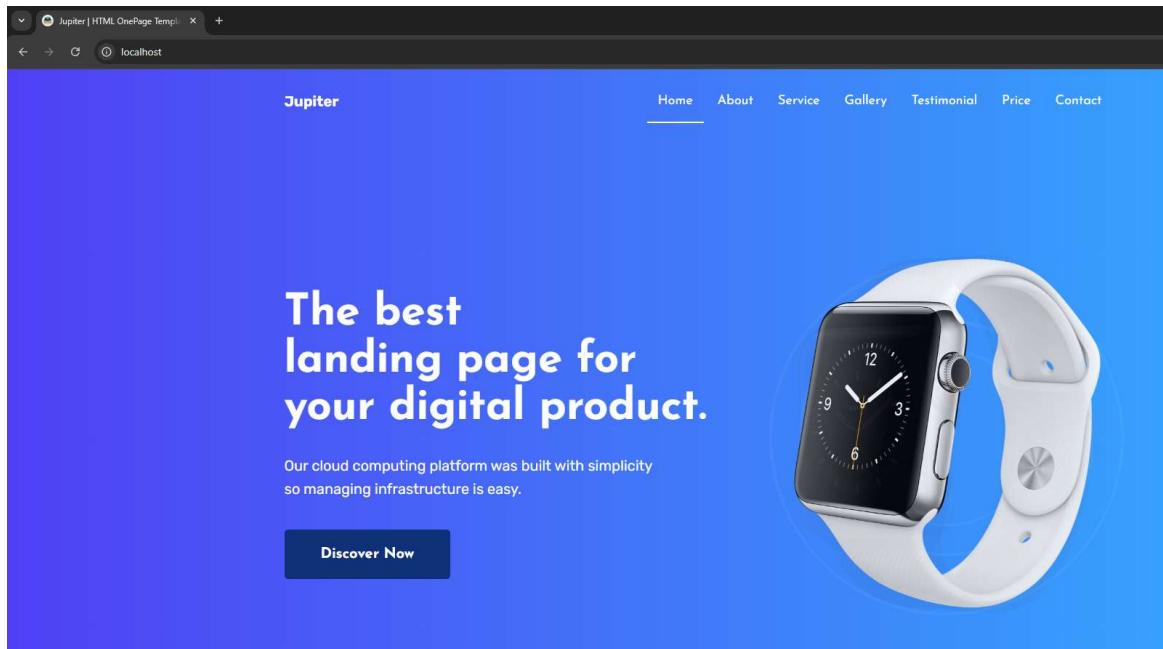
In the VSCode terminal write ‘**docker run -dp 80:80 jupiter**’. Docker run will create and run the container from the image we created earlier. ‘-d’ runs the container in detached mode which means the container runs in the background instead of occupying the terminal session. ‘-p 80:80’ maps the containers to port 80 to the hosts port 80 (your local machine). ‘**jupiter**’ this is the name of the docker image we created earlier and docker will run this image as a container.

```

REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
jupiter        latest       f7d0889902ce  22 hours ago  965MB
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website>

> docker run -dp 80:80 jupiter
02b08f80f2e0238bbcfc095bcd29873353cbd06afe429f62b9d7d0b16e5060af2
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website>
```

Based on our recent command to run the docker image, we should be able to access the Jupiter website via ‘<http://localhost:80>’. Write this in your search engine and you should see the below website.



GREAT! Your site is now up and running via the localhost. To stop the container you can write the following commands in the VSCode terminal.

1. Docker ps – to show the current running containers.

```
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
584493cb228c jupiter "/usr/sbin/httpd -D ..." 4 minutes ago Up 4 minutes 0.0.0.0:80->80/tcp youthful_ardinghelli
```

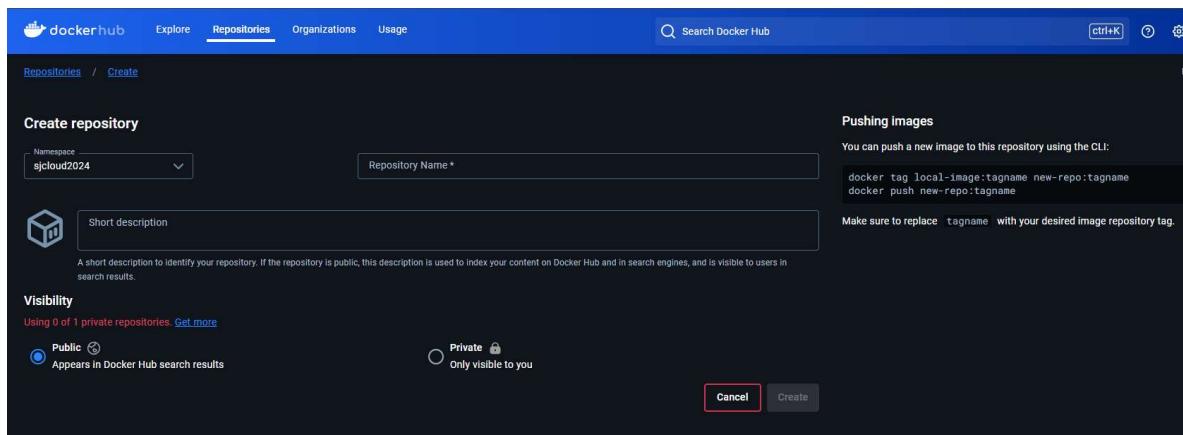
As you can see, I have one container running which is the Jupiter container.

2. Docker stop <container ID> - this command will stop the container.

```
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
584493cb228c jupiter "/usr/sbin/httpd -D ..." 4 minutes ago Up 4 minutes 0.0.0.0:80->80/tcp youthful_ardinghelli
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> docker stop 584493cb228c
584493cb228c
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> []
```

Create a Repository in Docker Hub

To create a repository in docker hub, you will need to go to ‘hub.docker.com’ in your search engine. Once you are logged in, go to your repository and click create repository. Give your repo a name and description, followed by making the visibility public and click create.



Congrats! You have now created a docker repository.

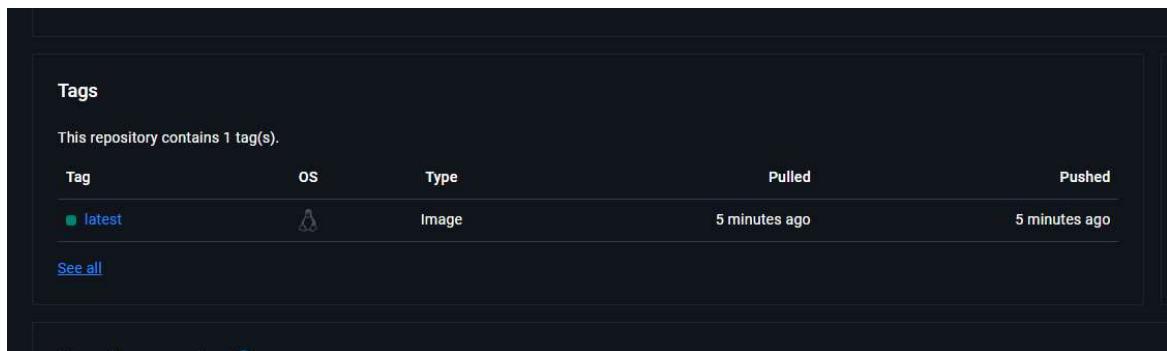
Push the Image to the Docker Hub Repository

In this section, I'm going to change the name of our image and push the docker image to the docker repository we created in docker hub. To do this you will need to do the following:

1. Log into docker via the VSCode terminal – this is to ensure I'm logged in and the image will be pushed to docker. Write ‘docker -u <yourusername>’ and write your password. Once completed you should get a response saying, ‘login succeeded’.

```
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> docker login -u sjcloud2024
Password:
Login Succeeded
```

2. Changing the name of our image – to do this we you will need to enter the following command into the VSCode terminal: ‘docker tag jupiter <yourusername/Jupiter>’. Once entered you can run the ‘docker image ls’ command to see the changes.
3. Pushing the image to your docker repo – run the command ‘docker push sjcloud2024/jupiter’ in the terminal. Once you run and all checks have been pushed, you can go to your repo in docker and you should see the image tagged as ‘latest’.



Install the AWS Command Line (CLI) on windows computer and create an IAM user

As this documentation is for showing the deployment of a static website on AWS with docker, I will not be going through the process of installing the AWS CLI and creating an IAM user with admin access.

Create Access Key for an IAM User

In this section we will go through the process of creating an access key for the IAM user created earlier.

1. Go to users within the AWS management console and click on the user your created for programmatic access.

The screenshot shows the AWS Identity and Access Management (IAM) service. On the left, there's a sidebar with a search bar and links for Dashboard, Access management (with sub-links for User groups, Users, Roles, and Policies), and a main 'Users' link which is currently selected. The main content area is titled 'Users (3)' with a 'Info' link. It contains a table with three rows, each representing a user: 'CodeBuild-User', 'SJCLOUD-PROGRAMMATIC-U...', and 'SJCLOUD2024'. Each row has a checkbox and a path indicator ('/').

2. Once within the user page, click on security credentials and scroll down to access keys and click create access key.

The screenshot shows the 'Security credentials' tab for a specific IAM user. At the top, there are tabs for Permissions, Groups, Tags, Security credentials (which is selected and highlighted in blue), and Last Accessed. Below this, the 'Console sign-in' section shows a 'Console sign-in link' (https://975049930561.signin.aws.amazon.com/console) and a 'Console password' status (Not enabled). There are buttons for 'Enable console access' and 'Assign MFA device'. The 'Multi-factor authentication (MFA) (0)' section indicates no MFA devices assigned. The 'Access keys (0)' section shows a note about using access keys for programmatic calls and a button to 'Create access key'. It also notes that there are no access keys present.

3. Click Command line interface (CLI) and click the confirmation below.

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.

Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

Application running outside AWS
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

Other
Your use case is not listed here.

⚠ Alternatives recommended

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

Confirmation

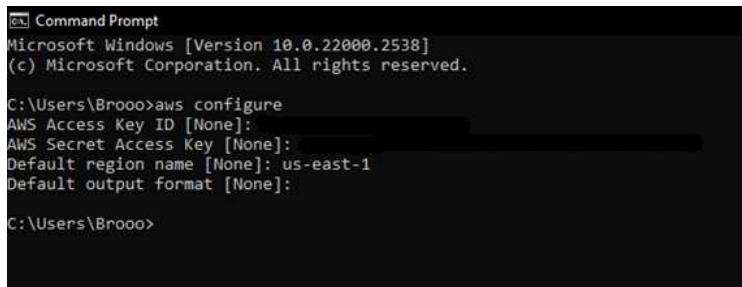
I understand the above recommendation and want to proceed to create an access key.

[Cancel](#) [Next](#)

4. The next page will ask you to provide a tag description for your access. We will leave this blank and click next.
5. GREAT! You have now created your access keys for your user. Save your access key and secret access key as you will need them later. IMPORTANT note your access keys will only be shown once, and you will not be able to see them again. Make sure you save them down.

Create a profile and configure access keys for programmatic access

1. Open the command prompt and write ‘aws configure’ and press enter. You will get a prompt to enter your access key ID and secret access key. Use the access keys created from the previous step. You will also be asked to provide a default region name. In this example I used ‘us-east-1’. The following prompt will ask for a default output format and for this you can leave blank and press enter.

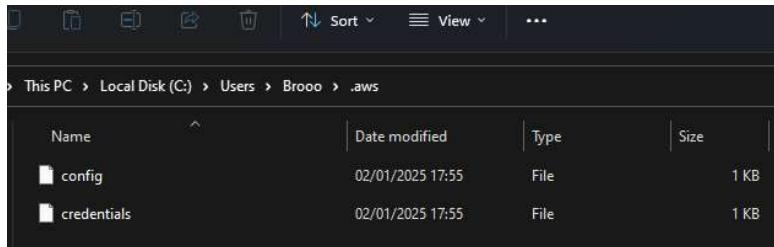


```
Command Prompt
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Brooo>aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: us-east-1
Default output format [None]

C:\Users\Brooo>
```

2. To check your access keys have been stored, go to your ‘C drive’, click ‘users’, then click on your computer user name, followed by ‘.aws’ and you will be able to see your credentials and config file saved in this folder.



Create an Amazon ECR Repository to Store your Image

In this section we are going to create an ECR repository and store our image in the repo.

1. Got to google and write ‘aws cli create ecr repository’. Click on the first link which says aws documentation. You should see the below page.

AWS CLI COMMAND REFERENCE Home User Guide Forum GitHub

aws

Table Of Contents

- create-repository
 - Description
 - Synopsis
 - Options
 - Global Options
 - Examples
 - Output

Quick search

Search box Search

Feedback

Did you find this page useful?
Do you have a suggestion to improve the documentation?
Give us feedback.
If you would like to suggest an improvement or fix for the AWS CLI, check our contributing guide on GitHub.

[aws . ecr]

create-repository

Description

Creates a repository. For more information, see Amazon ECR repositories in the *Amazon Elastic Container Registry User Guide*.

See also: [AWS API Documentation](#)

Synopsis

2. Scroll down to 'examples' and the first example will show the aws cli command to create a repository.

Reference Home User Guide Forum GitHub

Star | 15.683

Examples

Note:

To use the following examples, you must have the AWS CLI installed and configured. See the [Getting started guide](#) in the *AWS CLI User Guide* for more information.

Unless otherwise stated, all examples have unix-like quotation rules. These examples will need to be adapted to your terminal's quoting rules. See [Using quotation marks with strings](#) in the *AWS CLI User Guide*.

Example 1: To create a repository

The following `create-repository` example creates a repository inside the specified namespace in the default registry for an account.

```
aws ecr create-repository \
--repository-name project-a/sample-repo
```

Output:

```
{
```

3. Copy the cli command and paste it into your computer CMD. I changed the name of the repo and added a region as I want the repo to be in 'us-east-1'. See below for reference: (`aws ecr create-repository --repository-name jupiter --region us-east-1`)

```
C:\Users\Brooo>aws ecr create-repository --repository-name jupiter --region us-east-1
{
  "repository": {
    "repositoryArn": "arn:aws:ecr:us-east-1:975049930561:repository/jupiter",
    "registryId": "975049930561",
    "repositoryName": "jupiter",
    "repositoryUri": "975049930561.dkr.ecr.us-east-1.amazonaws.com/jupiter",
    "createdAt": "2025-01-03T18:27:09.090000+00:00",
    "imageTagMutability": "MUTABLE",
    "imageScanningConfiguration": {
      "scanOnPush": false
    },
    "encryptionConfiguration": {
      "encryptionType": "AES256"
    }
  }
}

C:\Users\Brooo>
```

4. Next you will need to save the output and store this in your notepad, as we will need this information for the next step.
5. Finally, we can go to the aws console to see if our ECR has been created. In the management console, search 'ECR' and you should see the repository we created earlier under private.

| Repository name | URI | Created at | Tag immutability | Encryption type |
|-----------------|--|-------------------------------------|------------------|-----------------|
| jupiter | 975049930561.dkr.ecr.us-east-1.amazonaws.com/jupiter | January 03, 2025, 18:27:09 (UTC-00) | Mutable | AES-256 |

Push the Image to your ECR Repository

Now we have created our repository in ECR, we will push our image to the ECR repository we created in aws.

1. Go to VSCode and open the terminal. Make sure you are in the Jupiter-website directory. Next you will need to write 'docker image ls' to list all the images within docker. The first image is the image we rename and the second image is the image we originally created.

```
* History restored
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> docker image ls
error during connect: Head "http://%2F%2Fpipe%2FdockerDesktopLinuxEngine/_ping": open //./pipe/dockerDesktopLinuxEngine: The system cannot find the file specified.
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> docker image ls
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
jupiter             latest   f7d0889902ce  5 days ago   965MB
s3cloud2024/jupiter latest   f7d0889902ce  5 days ago   965MB
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website>
```

Note: when I first enter the command 'docker image ls' I got an error. This was because I was not signed into my docker desktop.

2. Next, we need to tag our image to push the image to ECR. In the terminal write 'docker tag <repositoryUri>' which can be found in the information you saved in your notepad from the previous section and press enter.

```
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> docker tag jupiter 975049930561.dkr.ecr.us-east-1.amazonaws.com/jupiter
error during connect: Head "http://%2F%2Fpipe%2FdockerDesktopLinuxEngine/_ping": open //./pipe/dockerDesktopLinuxEngine: The system cannot find the file specified.
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> docker image ls
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
jupiter             latest   f7d0889902ce  5 days ago   965MB
s3cloud2024/jupiter latest   f7d0889902ce  5 days ago   965MB
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website>
```

3. Now you have submitted the tag via the terminal. Write 'docker image ls' to see the tagged image we created.
4. Next, we will need to log into aws via the VSCode terminal before we will be able to push the image to ECR. To do this you will need to enter the following command in the terminal – 'aws ecr get-login-password | docker login --username AWS --password'

stdin <aws_account_id>.dkr.ecr.<region>.amazonaws.com'. You will need to change the account id to your account id and this is the first 12 digits of your URI. You will also need to change the region.

```
REPOSITORY          TAG      IMAGE ID   CREATED    SIZE
s1cloud2024/jupiter        latest   f7d0889902ce  5 days ago  965MB
975049930561.dkr.ecr.us-east-1.amazonaws.com/jupiter  latest   f7d0889902ce  5 days ago  965MB
jupiter            latest   f7d0889902ce  5 days ago  965MB
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> aws ecr get-login-password | docker login --username AWS --password-stdin 975049930561.dkr.ecr.us-east-1.amazonaws.com
Login Succeeded
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website>
```

- Now we have logged in, we can push the container image to AWS. In the terminal write the following command 'docker push <URI from aws console>' and press enter.

```
Login Succeeded
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website> docker push 975049930561.dkr.ecr.us-east-1.amazonaws.com/jupiter
Using default tag: latest
The push refers to repository [975049930561.dkr.ecr.us-east-1.amazonaws.com/jupiter]
8fd0463e2b94: Pushed
de8d86f9ab94: Pushed
4f4fb700ef54: Pushed
e510c31b85b0: Pushed
2c42bfedf808: Pushed
f56855f48270: Pushed
503d83be2189: Pushed
899046e4a240: Pushed
latest: digest: sha256:f7d0889902ce9488ac5c420ec7a80c8eebec207f3cc949f88c0d37e77ee0300d size: 856
PS C:\Users\Brooo\Desktop\docker-projects\Jupiter-website>
```

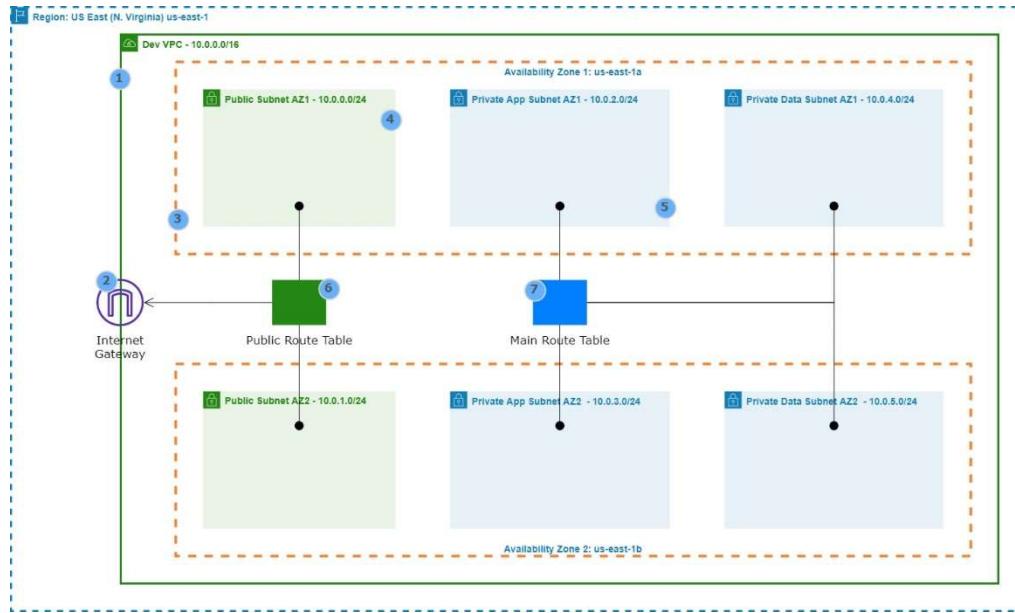
- Let's check our container image has been pushed to our ECR repository by going to the management console. Make sure to refresh the page, click on the Jupiter repository and you should see your image and the tag for that image is called 'latest'.

| Images (3) | | | | | |
|--|-----------|---------------|-------------------------------------|-----------|--|
| <input type="button" value="Search artifacts"/> <input type="button" value="Delete"/> <input type="button" value="Details"/> <input type="button" value="Scan"/> <input type="button" value="View"/> | | | | | |
| <input type="checkbox"/> | Image tag | Artifact type | Pushed at | Size (MB) | Image URI |
| <input type="checkbox"/> | latest | Image Index | January 06, 2025, 18:35:57 (UTC-00) | 262.75 | <input type="button" value="Copy URI"/> sha256:f7d0889902ce9488ac5c420ec7a80c... |
| <input type="checkbox"/> | - | Image | January 06, 2025, 18:35:56 (UTC-00) | 262.75 | <input type="button" value="Copy URI"/> sha256:5761f7c4ed13935c38814ef828f280... |
| <input type="checkbox"/> | - | Image | January 06, 2025, 18:35:56 (UTC-00) | 0.00 | <input type="button" value="Copy URI"/> sha256:e20dc1e532301164ff30403c3e350... |

Build a Three-Tier AWS Network VPC

To complete this project, we will start by a three tier VPC architecture. In a three tier vpc architecture your vpc is divided into the following tiers:

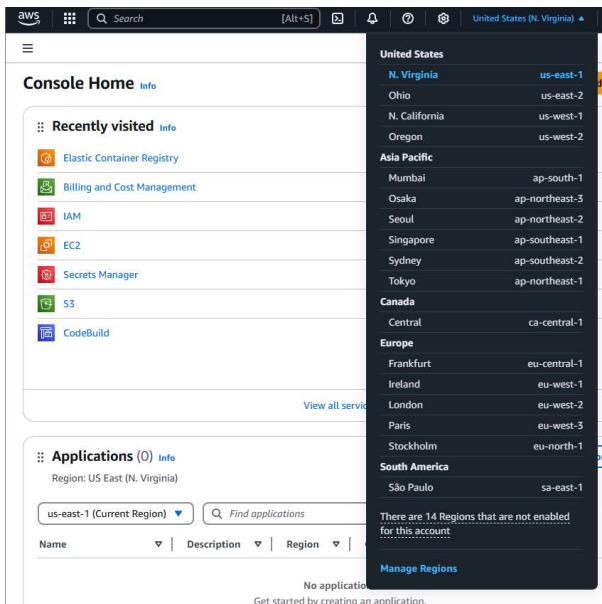
1. Tier 1 – Public subnet: this will host resources such as NAT gateway, load balancers and bastion host.
2. Tier 2 – Private subnet: this is the subnet that will host our web servers e.g. EC2 instances
3. Tier 3 – private subnet: this subnet will host our database



All 3 subnets will be duplicated across multiple availability zones for high availability and fault tolerance. Finally, we'll create an internet gateway and route table to allow the resources in our vpc to access the internet.

Creating a VPC

The first thing you will want to do is select the region you want your vpc to be places. In this example I will be using 'us-east-1'.



Once your region has been selected, navigate to the vpc dashboard by using the search bar. Click vpc then click create vpc.

Give your vpc a tag name and input your chosen IPV4 CIDR block. In this example I have chosen 10.0.0.0/16. The next three setting can be left as default and click create vpc.

The screenshot shows the 'Create VPC' configuration page. It includes sections for 'VPC settings', 'Tags', and 'AWS Lambda function VPC settings'. In the 'VPC settings' section, the 'Resources to create' dropdown is set to 'VPC only'. The 'Name tag - optional' field contains 'p1 VPC'. The 'IPv4 CIDR block' dropdown is set to 'IPv4 CIDR manual input' and the value '10.0.0.0/16' is entered. The 'IPv6 CIDR block' dropdown is set to 'No IPv6 CIDR block'. In the 'Tenancy' dropdown, 'Default' is selected. The 'Tags' section shows a single tag 'Name: p1 VPC'.

Great! You have now created your vpc. Next, we'll be enabling DNS host name in our vpc. To do this first you will need to filter by your vpc.

The screenshot shows the AWS VPC dashboard. A green success message at the top right says "You successfully created vpc-0c7342300bd2337d7 / p1 VPC". The main panel displays the details of the newly created VPC, "vpc-0c7342300bd2337d7 / p1 VPC". The "Details" section includes fields like VPC ID (vpc-0c7342300bd2337d7), State (Available), DNS resolution (Enabled), Main network ACL (acl-029ed873b156abe29), IPv6 CIDR (Network border group), Block Public Access (Off), DHCP option set (dopt-08784d203daa3a710), IPv4 CIDR (10.0.0.0/16), and Route 53 Resolver DNS Firewall rule groups. Below the details are tabs for Resource map, CIDRs, Flow logs, Tags, and Integrations. On the left sidebar, there are sections for EC2 Global View, Security (Network ACLs, Security groups), PrivateLink and Lattice (Endpoints, Endpoint services, Service networks), and Resource gateways.

Then click actions, edit VPC setting and then enable DNS hostnames.

The screenshot shows the "Edit VPC settings" page for the VPC "vpc-0c7342300bd2337d7". The "DNS settings" section contains two checked checkboxes: "Enable DNS resolution" and "Enable DNS hostnames". At the bottom of the page are "Cancel" and "Save" buttons.

The next step is to create an internet gateway for our vpc. Make sure your vpc is still selected via the filter on the left hand side. On the left hand side under ‘virtual private cloud’ click internet gateway, then click create internet gateway. You will need to give your internet gateway a name. In this example I have called my internet gateway ‘p1 internet gateway’.

Create internet gateway [Info](#)

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.

Internet gateway settings

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

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.

| Key | Value - optional |
|-----------------------------------|---|
| <input type="text" value="Name"/> | <input type="text" value="p1 internet gateway"/> Remove |

[Add new tag](#)
You can add 49 more tags.

[Cancel](#) [Create internet gateway](#)

In this next step we will need to attach our internet gateway to our vpc. Click actions then ‘attach to vpc’. At this point you should be able to search for your vpc which was created earlier. In my case, I can see my vpc called ‘p1 vpc’. Note to remember: you can only attach one internet gateway to one vpc.

Attach to VPC (igw-08a358ce73c821bd3) [Info](#)

VPC
Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.

Available VPCs
Attach the internet gateway to this VPC.

| Select a VPC |
|---|
| <input type="text" value="vpc-0c7342300bd2337d7 - p1 VPC"/> |

[AWS Command Line Interface command](#)

[Cancel](#) [Attach internet gateway](#)

Now we are going to we are going to create our two public subnets in our two availability zones. On the left hand side make sure to filter by you new vpc and then below click on subnets. Next click ‘create subnet’. Make sure to pick the vpc you created earlier. In my case my vpc is called ‘p1 VPC’. Under subnet settings you will need to give your subnet a name and in our reference architecture our first subnet is called ‘public subnet AZ1’. The availability zone will be ‘us-east-1a’ and the subnet CIDR block will be ‘10.0.0.0/24’. Leave all tag options as default and click create subnet.

[Subnets](#) > Create subnet

VPC

VPC ID
Create subnets in this VPC.

vpc-0c7342300bd2337d7 (p1 VPC) ▾

Associated VPC CIDRs

IPv4 CIDRs
10.0.0.0/16

Subnet settings

Specify the CIDR blocks and Availability Zone for the subnet.

Subnet name
Create a tag with a key of 'Name' and a value that you specify.
public subnet AZ1
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.
US East (N. Virginia) / us-east-1a ▾

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
10.0.0.0/24 256 IPs
◀ ▶ ⌂ ⌃ ⌄

▼ **Tags - optional**

To create your second subnet, follow the same steps as above however, the availability zone will be 'us-east-1b' and the CIDR block will be '10.0.1.0/24'. See below for reference.

[Subnets](#) / [Create subnet](#)

VPC

VPC ID

Create subnets in this VPC.

vpc-0c7342300bd2337d7 (p1 VPC) ▾

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.
public subnet AZ2

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.
US East (N. Virginia) / us-east-1b ▾

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

To see the new public subnets you created, you will need to clear all filters and filter by your chosen VPC and then you will see the two public subnets you created.

The screenshot shows the AWS VPC dashboard. On the left, under 'Virtual private cloud', there is a list of 'Your VPCs'. A specific VPC is selected, showing its ID (vpc-0c7342300bd2337d7), name (p1 VPC), and owner (975049930561). On the right, the 'Subnets' section displays two subnets: 'public subnet AZ1' and 'public subnet AZ2'. Both subnets are listed as 'Available' and are associated with the VPC 'vpc-0c7342300bd2337d7'. A success message at the top indicates that a new subnet was created successfully.

Next this we are going to do is enable to the auto assign the IP settings. This means any time we launch an EC2 instance in the public subnets we created, the EC2 instances will be assigned an IPV4 address.

To do this, select your first public subnet and click actions at the top then click edit subnet settings. In the next page under 'auto-assign IP setting' tick the box enable auto-assign public IPV4 address. Then press save.

This screenshot shows the 'Edit subnet settings' page for the subnet 'public subnet AZ1'. It includes sections for 'Subnet' (with Subnet ID: subnet-0842b9a1837ed819b and Name: public subnet AZ1) and 'Auto-assign IP settings' (with options for enabling auto-assign public IPv4 address and customer-owned IPv4 address). The 'Enable auto-assign public IPv4 address' checkbox is checked.

Make sure to do the same actions for your second public subnet.

Creating a route table

To create a route table, click on route tables in the left-hand side of the management console, also make sure your filter is still filtered by your chosen vpc. You will notice that you have one route table already in your list. This table was created when we created the vpc earlier. Note that this table is the main route table and is private by default.

Click create route table, give you route table a name and select the your vpc from the drop down list, then press create route table.

This screenshot shows the 'Create route table' page. It displays a summary of the route table 'rtb-093503d663828153c' with the message 'public route table was created successfully.' The 'Details' section shows the Route table ID (rtb-093503d663828153c), Main status (No), Owner ID (975049930561), and VPC (vpc-0c7342300bd2337d7 | p1 VPC). The 'Routes' section lists a single route for destination 10.0.0.0/16 with target 'local', status 'Active', and propagation 'No'.

Great! You have created a public route table. Next, we will need to add a public route to our new route table. The public route will allow our route table to route traffic to the internet. To do this click 'edit route', add route and in the destination under route 2 type '0.0.0.0/0'. The target will be our internet gateway in the drop-down list and below this you should be able to select your internet gateway.

Route 1
Destination 10.0.0.0/16
Target local
Status Active

Route 2
Destination 0.0.0.0/0
Target Internet Gateway
Status Active

Propagated No

Propagated No

Add route Remove

Cancel Preview Save changes

Associate the two public subnets with the public route table

By associating our two public subnets with the public route table will ensure all our resources in the public subnets can access the internet or be accessed by it. To do this, within the public route table we created click 'edit subnet associations' within the explicit subnet associations. Once within the edit subnet association page, you will be able to see the public subnet your created. Tick both subnets and click 'save associations'.

| Available subnets (2/2) | | | | | |
|-------------------------------------|-------------------|--------------------------|-------------|-----------|------------------------------|
| | Name | Subnet ID | IPv4 CIDR | IPv6 CIDR | Route tab |
| <input checked="" type="checkbox"/> | public subnet AZ2 | subnet-0c4e9525271730596 | 10.0.1.0/24 | - | Main (rtb-093503d663828153c) |
| <input checked="" type="checkbox"/> | public subnet AZ1 | subnet-0842b9a1837ed819b | 10.0.0.0/24 | - | Main (rtb-093503d663828153c) |

Selected subnets

subnet-0c4e9525271730596 / public subnet AZ2 X subnet-0842b9a1837ed819b / public subnet AZ1 X

Cancel Save associations

Now within the subnet associations tab, you should see both public subnets listed under the 'explicit subnet associations'.

Creating private subnets

Navigate to the subnet tab on the left-hand side of the management console. Click create subnets and make sure you select your vpc ID. Under subnet settings we are going to call our first private subnet ‘private app subnet AZ1’, this will be in the ‘us-east-1a’ availability zone, the IPV4 subnet CIDR block will be ‘10.0.2.0/24’.

Create subnet [Info](#)

VPC

VPC ID
Create subnets in this VPC.

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.

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.

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.

IPv4 subnet CIDR block
 256 IPs

You will need to do this step three more times as we need four private subnets. The details for the private subnets are within the main architecture above. Once all subnets have been created, filter by your vpc and your subnet page should look like below.

The screenshot shows the AWS VPC dashboard with the Subnets table. A green banner at the top indicates "You have successfully created 1 subnet: subnet-07f7d64248fd75bb3". The Subnets table lists six subnets:

| Name | Subnet ID | State |
|-------------------------|--------------------------|-----------|
| Private app subnet AZ1 | subnet-007a0403734ca1d74 | Available |
| Private app subnet AZ2 | subnet-04c9518d6f775fe4c | Available |
| Private data subnet AZ1 | subnet-014ea7194cd22fd7e | Available |
| Private data subnet AZ2 | subnet-07f7d64248fd75bb3 | Available |
| public subnet AZ1 | subnet-0842b9a1837ed819b | Available |
| public subnet AZ2 | subnet-0c4e9525271730596 | Available |

You should have 6 subnets:

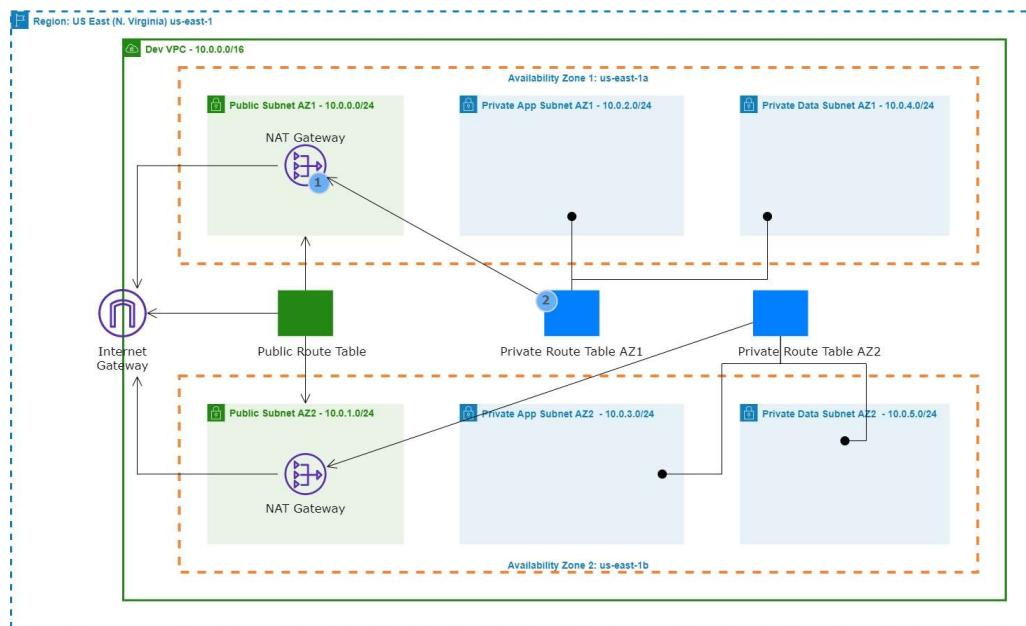
- Two private app subnets
- Two private data subnets
- Two public subnets

Create NAT Gateways

In the section we are going to create a NAT Gateway in the first and second availability zones using the below reference architecture. With reference to the below architecture, we will create a NAT gateway in the public subnet AZ1 and create a route table that we'll call 'private route table AZ1'.

Once we the route table has been created, we'll add route to the route table to route traffic to the internet through the NAT gateway. We'll also associate the route table to the private App subnet AZ1 and private Data subnet AZ1. This process will be replicated in the second availability zone.

Note: A Nat Gateway enables instances in private subnets to access the internet or other AWS services without exposing those instances to incoming internet traffic.



To create our first NAT gateway, navigate the vpc created earlier and on the left-hand side of the management console, you should be able to see 'NAT gateway'. Give your NAT Gateway a name, under subnet pick the public subnet we created and under elastic IP allocation ID click allocate elastic IP.

Elastic IP address 54.144.209.21 (eipalloc-080e27f82c675a8df) allocated. X

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.
 Select Allocate Elastic IP

Additional settings Info

Great! You have created a NAT gateway. Next, we will create a route table, and we'll call this route table 'private route table AZ1'. Navigate to the route table page and click create route table. Give your table a name and select the correct vpc which should be the vpc we created earlier. Once this has been done, you can click create route table.

Route table settings

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

VPC
The VPC to use for this route table.

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="text" value="Private Route Table AZ1"/> <input type="button" value="Remove"/> |

Add new tag
You can add 49 more tags.

Create route table

Now we're going to add a route to our private route table AZ1 to route traffic to the internet through the NAT gateway in the public subnet AZ1. To do this, click actions and then 'edit routes'. The destination will be '0.0.0.0/0' and the target will be 'NAT gateway' and you should be able to select the public NAT gateway AZ1 we created.

Edit routes

Route 1
Destination 10.0.0.0/16
Target local
Propagated No

Route 2
Destination 0.0.0.0/0
Target NAT Gateway
Propagated No

Add route

Next, we'll associate the private route table AZ1 to the private app subnet AZ1 and private data subnet AZ1. You will need to click 'subnet associations' and then under 'explicit subnet associations' click 'edit subnet associations'. Once in the edit subnet associations page, you will need to tick the private app subnet AZ1 and private data subnet AZ1 then click save associations.

Edit subnet associations
Change which subnets are associated with this route table.

Available subnets (2/6)

| Name | Subnet ID | IPv4 CIDR | IPv6 CIDR | Route table ID |
|---|-------------------------|-------------|-----------|-------------------|
| <input checked="" type="checkbox"/> Private app subnet AZ1 | subnet-007a0403734ca... | 10.0.0.0/24 | - | Main (rtb-0788de) |
| <input type="checkbox"/> Private app subnet A22 | subnet-04e951866f75... | 10.0.3.0/24 | - | Main (rtb-0788de) |
| <input checked="" type="checkbox"/> Private data subnet AZ1 | subnet-014ea7194cd22... | 10.0.4.0/24 | - | Main (rtb-0788de) |
| <input type="checkbox"/> Private data subnet AZ2 | subnet-07f765424bf07... | 10.0.5.0/24 | - | Main (rtb-0788de) |
| <input type="checkbox"/> public subnet AZ1 | subnet-0842b9a1837ed... | 10.0.0.0/24 | - | rtb-093503d6638 |
| <input type="checkbox"/> public subnet AZ2 | subnet-0c4e952527173... | 10.0.1.0/24 | - | rtb-093503d6638 |

Selected subnets

| | |
|---|--|
| subnet-007a0403734ca1d74 / Private app subnet AZ1 | subnet-014ea7194cd22fd7e / Private data subnet AZ1 |
|---|--|

Save associations

Now we will create the second NAT gateway in the public subnet AZ2. To do this please following the same steps we did to create the first NAT gateway.

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.

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

Next, we'll create the second private route table. This process will be the same as what we did to create the first private route table. Please refer to above steps.

Create route table Info

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Route table settings

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

VPC
The VPC to use for this route table.

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="text" value="Private Route Table AZ2"/> <input type="button" value="Remove"/> |

You can add 49 more tags.

Following the creation of the second NAT gateway and second private route table, we'll need to create a route to the second route table to route traffic to the NAT gateway in the public subnet AZ2. As mentioned above this will be the same process as before, therefore please refer to above steps.

Note: this route will be linked to the NAT gateway in AZ2.

Edit routes

Route 1
Destination 10.0.0.0/16
Target local

Propagated
No

Route 2
Destination 0.0.0.0/0
Target NAT Gateway

Use: "nat-02bbd64c35982253a"

The final process is to associate the private route table to the private app subnet AZ2 and the private data subnet AZ2. This again will be the same steps we took above to create the associations for AZ1, however, this is new related to AZ2.

Available subnets (2/6)

| Name | Subnet ID | IPv4 CIDR | IPv6 CIDR | Route table |
|-------------------------|--------------------------|-------------|-----------|-------------|
| Private data subnet AZ2 | subnet-07f7d64248fd7... | 10.0.5.0/24 | - | rtb-012fe1 |
| public subnet A22 | subnet-0cce952527173... | 10.0.1.0/24 | - | rtb-093503 |
| public subnet A21 | subnet-0842b9a1837ed... | 10.0.0.0/24 | - | rtb-093503 |
| Private data subnet AZ1 | subnet-014ea7194cd22... | 10.0.4.0/24 | - | rtb-039cf0 |
| Private app subnet A21 | subnet-007a0403734ca... | 10.0.2.0/24 | - | rtb-039cf0 |
| Private app subnet AZ2 | subnet-04c9518d6f775fe4c | 10.0.3.0/24 | - | rtb-012fe1 |

Selected subnets

- subnet-04c9518d6f775fe4c / Private app subnet AZ2
- subnet-07f7d64248fd75bb3 / Private data subnet AZ2

Buttons: Cancel, Save associations

Create Security Group

In this section we are going to be creating two security groups:

1. Application load balancer security group – open port 80 and 443 from anywhere
2. Container security group – opening port 80 and 443, source of traffic will be from the ALB security group

Navigate to the VPC dashboard in the management console. Then filter by the vpc we created earlier in this documentation. Once filtered scroll down on the left-hand side and click ‘security groups’. We’ll call our security group ‘ ALB security group’ and select our VPC from the dropdown list. We’ll also need to add two inbound rules:

1. Type: HTTP – Source: 0.0.0.0/0
2. Type: HTTPS – Source: 0.0.0.0/0

Basic details

Security group name: ALB Security Group
Name cannot be edited after creation.

Description: ALB Security Group

VPC Info: vpc-0c7342300bd2337d7 (p1 VPC)

Inbound rules

Inbound rule 1

| | | |
|---------------|--------------|------------------------|
| Type | Protocol | Port range |
| HTTP | TCP | 80 |
| Source type | Source | Description - optional |
| Anywhere-IPv4 | Q. 0.0.0.0/0 | 0.0.0.0/0 |

Inbound rule 2

| | | |
|---------------|--------------|------------------------|
| Type | Protocol | Port range |
| HTTPS | TCP | 443 |
| Source type | Source | Description - optional |
| Anywhere-IPv4 | Q. 0.0.0.0/0 | |

Next, we're going to create the container security group. This will be the same process has creating to ALB security group. However, the inbound rules will be a little different. The first inbound rule will be 'HTTP' on port and the source will be our ALB security group. The next inbound rule will be 'HTTPS' on port 443 and the source will also be the ALB security group. Once completed you can click create security group.

Basic details

Security group name [Info](#)
Container Security Group
Name cannot be edited after creation.

Description [Info](#)
Container Security Group

VPC [Info](#)
vpc-0c7342300bd2337d7 (p1 VPC)

Inbound rules [Info](#)

| Type | Protocol | Port range | Source | Description - optional |
|-------|----------|------------|--------------------------------|------------------------|
| HTTP | TCP | 80 | Custom sg-0bb658eb6c73a4071 | |
| HTTPS | TCP | 443 | Custom sg-0bb658eb6c73a4071 | |

Creating an application load balancer

In this section, we'll be creating an application load balancer. This will be used to route traffic to the ECS task we'll create later.

Navigate to the EC2 section within the management console. Before creating the application load balancer, we'll need to create a target group and when the ECS creates our tasks it will add those tasks to the target group, then the application load balancer will route traffic to the tasks in the target group. On the left-hand side scroll down and under load balancing you should see 'target groups'.

1. Basic configuration will be 'IP addresses'
2. Give your target group a name of your choice. In my case I called my target group 'P1-target-group'

Specify group details

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

Basic configuration

Settings in this section can't be changed after the target group is created.

Choose a target type

Instances

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.

IP addresses

- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

Lambda function

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

Application Load Balancer

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

Target group name

P1-target-group

- Under the VPC section select the VPC you created earlier and all other setting can remain as default.
- On the following page which is the register target section make sure your VPC has been selected from the drop-down list. You can leave all other settings as default.

As you can see under the review targets, we have no targets as we have not yet created our ECS tasks. Once the ECS task has been created the ECS service will automatically add the targets to this section.

Now let's create our application load balancer!

Make sure you select application load balancer, give your alb a name and the scheme will be internet-facing.

Create Application Load Balancer

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

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.

P1-alb

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

Scheme

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

Internet-facing

- Serves internet-facing traffic.
- Has public IP addresses.
- DNS name is publicly resolvable.
- Requires a public subnet.

Internal

- Serves internal traffic.
- Has private IP addresses.
- DNS name is publicly resolvable.
- Compatible with the IPv4 and DualStack IP address types.

Load balancer IP address type

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

For a public IPv4 address, and private IPv4 and IPv6 addresses. Compatible with **Internet-facing** load balancers only.

In the network mapping section, make sure you select your VPC from the drop down list. Under mapping select your two availability zones.

Note: your subnets should be your PUBLIC subnets and NOT your private subnets

The screenshot shows the 'Network mapping' section of the AWS Load Balancer configuration. It includes:

- VPC:** A dropdown menu showing 'p1 VPC' with the value 'vpc-0c7342300bd2337d7'. Below it, 'IPv4 VPC CIDR: 10.0.0.0/16' is listed.
- Mappings:** A table for selecting Availability Zones and subnets. It lists two rows:
 - us-east-1a (use1-az2):** Subnet 'subnet-0842b9a1837ed819b' mapped to 'public subnet AZ1' (IPv4 subnet CIDR: 10.0.0.0/24).
 - us-east-1b (use1-az4):** Subnet 'subnet-0c4e9525271730596' mapped to 'public subnet AZ2' (IPv4 subnet CIDR: 10.0.1.0/24).
- IPv4 address:** Assigned by AWS.

Next remove the default security group and select the ALB security group from the dropdown list. The default action for listeners and routing will be the target group we created earlier.

The screenshot shows the 'Security groups' and 'Listeners and routing' sections of the AWS Load Balancer configuration. It includes:

- Security groups:** A dropdown menu showing 'ALB Security Group' with the value 'sg-0bb658eb6c73a4071' and 'VPC: vpc-0c7342300bd2337d7'.
- Listeners and routing:** A table for defining listeners and their actions. It lists one row:
 - Listener HTTP:80:** Protocol 'HTTP' (dropdown), Port '80' (input), and '1-65535' (input). Default action is 'P1-target-group' (Forward to) via 'HTTP' (dropdown).
 - Create target group [?]**
- Listener tags - optional:** A note about adding tags to categorize resources.
- Add listener tag:** A button to add more tags.
- Add listener:** A button to add a new listener.

Once the above has been completed, you can scroll down and click create load balancer.

Create an ECS cluster

In the previous topic, we pushed our image to the ECR repository and now we are ready to run our ECS fargate container.

In the search bar, search for ECS and on the left hand side select clusters. Within the create cluster page you will need to give your cluster a name. In my case I called my cluster 'jupiter-cluster'. Under infrastructure I tick the AWS fargate (serverless). This is because we do not want to deal with the underlying infrastructure and want AWS to handle this. In a real work environment, you are also likely to choose this option.

The screenshot shows the 'Create cluster' wizard. The first section, 'Cluster configuration', has 'Cluster name' set to 'jupiter-cluster'. The second section, 'Infrastructure', is set to 'Serverless' and has 'AWS Fargate (serverless)' checked. The third section, 'Monitoring - optional', has 'CloudWatch Container Insights' described as a monitoring and troubleshooting solution for containerized applications and microservices.

Create cluster

Create cluster Info

An Amazon ECS cluster groups together tasks, and services, and allows for shared capacity and common configurations. All of your tasks, services, and capacity must belong to a cluster.

Cluster configuration

Cluster name
jupiter-cluster

Cluster name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Default namespace - optional
Select the namespace to specify a group of services that make up your application. You can overwrite this value at the service level.

Q jupiter-cluster X

▼ Infrastructure Info Serverless

Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances.

AWS Fargate (serverless)
Pay as you go. Use if you have tiny, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate Spot capacity providers by default.

Amazon EC2 instances
Manual configurations. Use for large workloads with consistent resource demands.

ⓘ External instances using **ECS Anywhere** can be registered after cluster creation is complete.

► Monitoring - optional Info

CloudWatch Container Insights is a monitoring and troubleshooting solution for containerized applications and microservices.

All other options can be left as default and click create.

Create task definition

Now we have created the ECS cluster, we will need to create a task definition. Give your task definition a name, make sure under infrastructure AWS fargate has been selected. Your operating system/architecture will be linux/X86_64 if you created your docker image on a windows computer. However, if you created your docker image on a Mac computer select linux/ARM64.

The screenshot shows the 'Create new task definition' page in the AWS Management Console. On the left, there's a sidebar with navigation links for Clusters, Namespaces, Task definitions (which is selected), Account settings, and other services like AWS Copilot, ECR, and AWS Batch. The main content area is titled 'Create new task definition' and contains two sections: 'Task definition configuration' and 'Infrastructure requirements'. In the 'Task definition configuration' section, the 'Task definition family' is set to 'jupiter-task-definition'. In the 'Infrastructure requirements' section, the 'Launch type' is set to 'AWS Fargate'. The 'Operating system/Architecture' dropdown is set to 'Linux/X86_64', and the 'Network mode' dropdown is set to 'awsvpc'. There are also fields for 'Memory' (set to 3 GB) and 'CPU' (set to 1). A note at the bottom says 'PI requests to AWS services. You can create a task IAM role for your task.'

Under container – 1, give your container an name and also input your URI. This can be found in your AWS ECR. Navigate to the ECR repository and select the repo you created earlier. Once within this repo, you will need to select the ‘latest’ image tag from the images. Within the image index you will be able to see the URI.

The screenshot shows the 'Image Index' page in the AWS Management Console. The left sidebar has sections for Private registry (Repositories, Summary, Images, Permissions, Lifecycle Policy, Repository tags, Features & Settings) and Public registry (Repositories, Settings). The main content area shows details for a repository named 'jupiter'. Under 'Image tags', it lists 'latest' with a URI of '975049930561.dkr.ecr.us-east-1.amazonaws.com/jupiter:latest'. Under 'Digest', it shows 'sha256:f7d0889902ce9488ac5c420ec7a80c8eebec207f3cc949f88c0d37e77ee0300d'. The 'General information' section includes 'Artifact type' (Image Index), 'Repository' (jupiter), 'Pushed at' (January 06, 2025, 18:35:57 (UTC-00)), and 'Size (MB)' (262.75).

Paste the URI in the image URI.

Amazon Elastic Container Service > Create new task definition

Container - 1 Info Essential container Remove

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 jupiter

Image URI 975049930561.dkr.ecr.us-east-1.amazonaws.com/jupiter:latest

Essential container Yes

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 |
|----------------|----------|
| 80 | TCP |

| Port name | App protocol |
|-------------------------|--------------|
| container-port-protocol | HTTP |

Remove

At this point you can leave all other settings as default and click create.

Create a Service

Now we've created the task definition, we'll create the ECS service that will create the fargate containers.

With in the ECS page, click clusters, then select the Jupiter cluster. Scroll to the bottom of the page and click create under the services tab. The setting within 'environment' should be left as default and under 'deployment configuration' select 'service'. Click the dropdown list under 'family' and select your 'task definition'. You will also need to give a 'service name', in my case i put 'jupiter-service'.

Deployment configuration

Application type | Info

Specify what type of application you want to run.

Service

Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.

Task

Launch a standalone task that runs and terminates. For example, a batch job.

Task definition

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

Specify the revision manually

Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.

Family

jupiter-task-definition

Revision

1 (LATEST)

Service name

Assign a service name that is unique for this cluster.

jupiter-service

Up to 255 letters, underscores and lowercase numbers are allowed. Service names must begin with a letter.

All other setting can be left as default. However, when you get to the networking section you will need to select your vpc from the dropdown list, remove all subnets selected and only select your ‘private app subnet AZ1’ and ‘private app subnet AZ2’. You will also need to remove the default security group and select your container security group from the dropdown list. Next you will also need to turn off the public IP as our container will be in a private subnet.

Networking

VPC | **Info**
Select a VPC to use for your Amazon ECS resources.

vpc-0c7342300bd2337d7
p1 VPC

Create a new VPC

Subnets
Choose the subnets within the VPC that the task scheduler should consider for placement.

Choose subnets ▾ **Clear current selection**

subnet-04c9518d6f775fe4c X
Private app subnet AZ2
us-east-1b 10.0.3.0/24

subnet-007a0403734ca1d74 X
Private app subnet AZ1
us-east-1a 10.0.2.0/24

Security group | **Info**
Choose an existing security group or create a new security group.

Use an existing security group
 Create a new security group

Security group name
Choose an existing security group.

Choose security groups ▾

sg-0a53db8de955d6e61 X
Container Security Group

Public IP | **Info**
Choose whether to auto-assign a public IP to the task's elastic network interface (ENI).

Turned off

Next, we'll need to adjust the setting within the load balancer section. Select:

1. Use load balancing
2. Load balancer type: application load balancer
3. Under Application load balancer – select ‘use an existing load balancer’ and select your load balancer from the dropdown list
4. Under listener select ‘use an existing listener’ and select your ‘80:HTTP’ listener from the dropdown list.
5. Select ‘use an existing target group’ and from the dropdown list select your target group created earlier.

Clusters / **jupiter-cluster** / **Create service**

Load balancing - optional
Configure load balancing using Amazon Elastic Load Balancing to distribute traffic evenly across the healthy tasks in your service.

Use load balancing

Load balancer type | **Info**
Specify the load balancer type to distribute incoming traffic across the tasks running in your service.

Application Load Balancer
An Application Load Balancer makes routing decisions at the application layer (HTTP/HTTPS), supports path-based routing, and can route requests to one or more ports.

Network Load Balancer
A Network Load Balancer makes routing decisions at the transport layer (TCP/UDP).

Container
The container and port to load balance the incoming traffic to

jupiter 80:80

Host port:Container port

Clusters > [jupiter-cluster](#) > Create service

Application Load Balancer
Specify whether to create a new load balancer or choose an existing one.

Create a new load balancer
 Use an existing load balancer

Load balancer
Select the load balancer you wish to use to distribute incoming traffic across the tasks running in your service.

P1-alb

Health check grace period | [Info](#)
0 seconds

Listener | [Info](#)
Specify the port and protocol that the load balancer will listen for connection requests on.

Create new listener
 Use an existing listener

Listener
80:HTTP

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.

| Evaluation order | Rule path | Target group |
|------------------|-----------|-----------------|
| default | / | P1-target-group |

Target group | [Info](#)
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
 Use an existing target group

Target group name
P1-target-group

Health check path

The setting we'll need to adjust next will be the 'service auto scaling':

1. Select use service auto scaling
2. Minimum number of tasks will be (1) and maximum number will be (2)
3. Scaling policy type will be 'target tracking'
4. Give your policy a name. In my case I called mine 'jupiter'
5. ECS service metric: ECSServiceAverageCPUUtilization
6. Target value: 70
7. Scale-out cooldown period: 300
8. Scale-in cooldown period: 300

Clusters > [jupiter-cluster](#) > Create service

Service auto scaling - optional
Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. You can modify your service auto scaling configuration at any time to meet the needs of your application.

Use service auto scaling
Configure service auto scaling to adjust your service's desired count

| | |
|--|--|
| Minimum number of tasks The lower boundary to which service auto scaling can adjust the desired count of the service. 1 | Maximum number of tasks The upper boundary to which service auto scaling can adjust the desired count of the service. 2 |
|--|--|

Scaling policy type | [Info](#)
Create either a target tracking or step scaling policy.

Target tracking
Increase or decrease the number of tasks that your service runs based on a target value for a specific metric.

Step scaling
Increase or decrease the number of tasks that your service runs based on a set of scaling adjustments, known as step adjustments, that vary based on the size of the alarm breach.

Policy name
jupiter

ECS service metric
ECSServiceAverageCPUUtilization

Target value
70

Scale-out cooldown period
300

Scale-in cooldown period
300

Finally, you can click create and after a few mins your ECS service should be created. When created you should be able to see your service under the services tab.

The screenshot shows the AWS ECS Services page. At the top, there are buttons for 'Manage tags', 'Update', 'Delete service', and a prominent orange 'Create' button. Below these are search and filter fields: 'Filter services by value' (with a placeholder 'Filter services by value'), 'Filter launch type' (set to 'Any launch type'), and 'Filter service type' (set to 'Any service type'). A navigation bar below these includes icons for Service name, ARN, Status (Active), and Service. The main list displays one service: 'jupiter-service' with ARN 'arn:aws:ecs:us-east-1:123456789012:service/jupiter-service' and status 'Active'. The word 'REPLICAS' is partially visible to the right.

Register a new domain name in route 53

In the section we are going to register a new domain name in route 53, which will allow our end users to access our website using our domain instead of the DNS name of the load balancer.

First you will need to check if your chosen domain name is available via the register domain search.

The screenshot shows the 'Register domain' search interface. It has a header 'Register domain' with a cursor icon. Below it is a sub-header 'Find and register an available domain, or transfer your existing domains to Route 53.' A search input field contains the placeholder 'Enter a domain name'. Below the input field is a note: 'Each label (each part between dots) can be up to 65 characters long and must start with a-z or 0-9. Maximum length: 255 characters, including dots. Valid characters: a-z, 0-9, and - (hyphen)'. A 'Check' button is located at the bottom left of the search area.

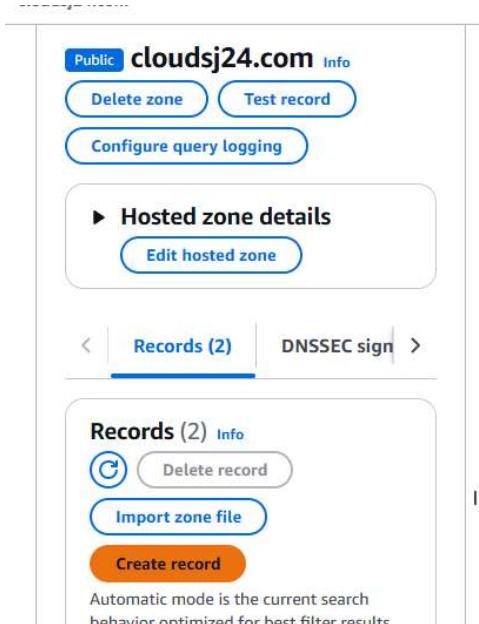
My domain name was available and my purchase price was \$14. Go through the process of purchasing your domain name. The process to officially registering your domain name should take around 10-15mins to change the status to successful.

The screenshot shows the 'Registered domains' page. On the left, a sidebar menu for 'Route 53' includes 'Dashboard', 'Hosted zones', 'Health checks', 'Profiles' (with 'New' highlighted), 'IP-based routing', and 'CIDR collections'. The main content area is titled 'Registered domains' with a 'Info' link. It features buttons for 'Download billing report', 'Transfer in', and 'Register domains'. A search bar 'Search domains by name' is present. A table lists registered domains: 'cloudsj24.com' with expiration date 'January 20, 2026', 'Auto-renew' status 'Off', and 'Transfer lock' status 'Off'.

Create a Record in route 53

In the section we'll create a record set in route 53 to point our domain name to the application load balancer.

Navigate to the Route 53 in the management console and in the main dashboard click 'hosted zone'. Next you will need to click into your registered domain name and then click 'create record'.

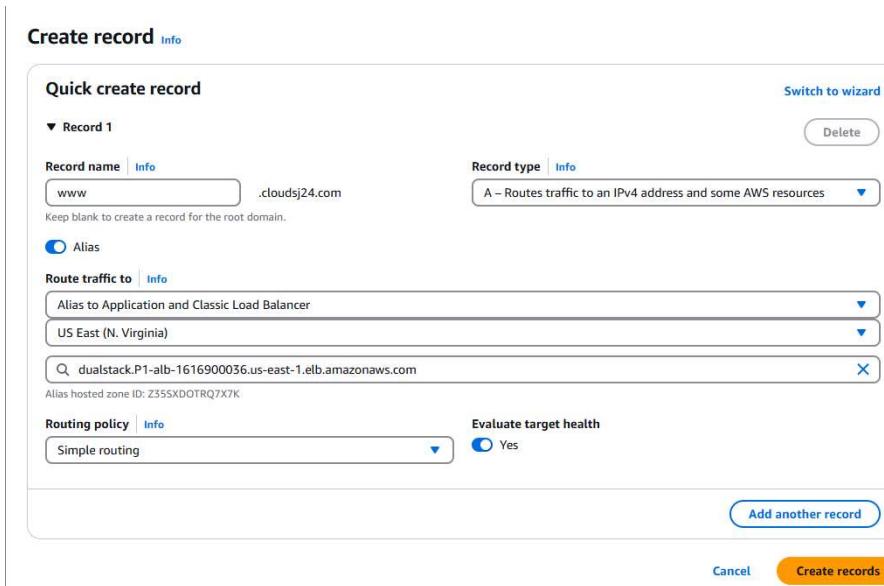


The screenshot shows the AWS Route 53 interface. At the top, there's a header with 'Public' and the domain name 'cloudsj24.com' with an 'Info' link. Below it are three buttons: 'Delete zone', 'Test record', and 'Configure query logging'. Underneath is a box titled 'Hosted zone details' with an 'Edit hosted zone' button. Below this is a navigation bar with 'Records (2)' selected. The main area is titled 'Records (2)' with an 'Info' link. It contains a 'Delete record' button, an 'Import zone file' button, and the orange 'Create record' button. A note below says 'Automatic mode is the current search behavior optimized for best filter results.'

Under the create record section you will need to adjust the following settings:

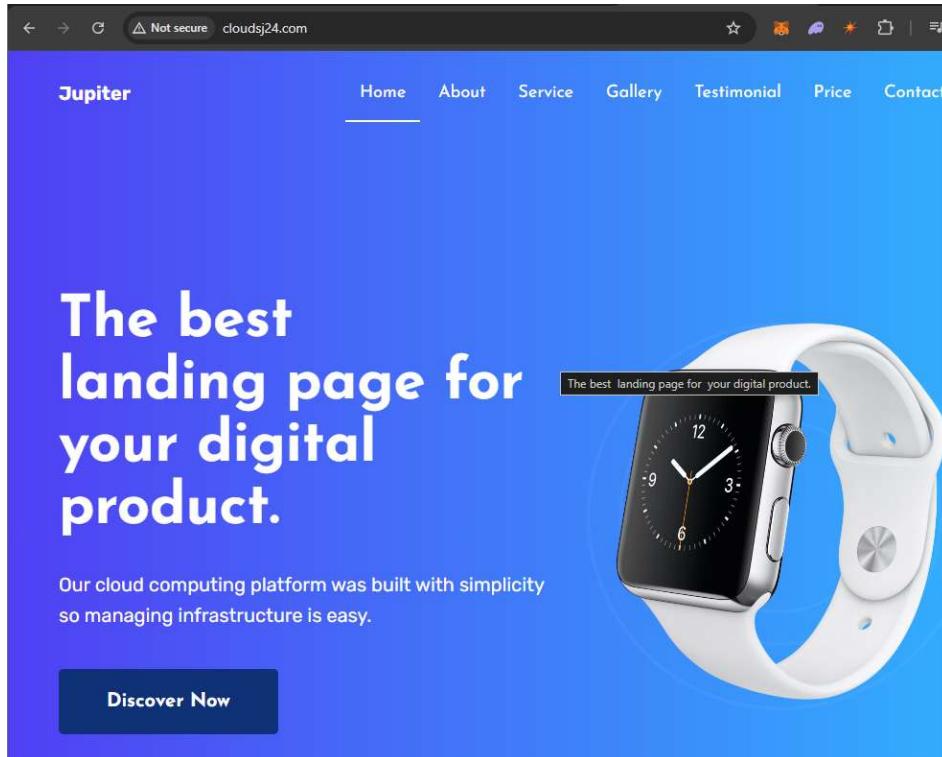
1. Record name: www
2. Record type: A
3. Make sure Alias has been turned on
4. Route traffic to: Alias to application and classic load balancer, select the region you created your load balancer in, then select your load balancer

All other setting can be left as default.



The screenshot shows the 'Create record' wizard for the 'cloudsj24.com' hosted zone. The 'Quick create record' step is active. In the 'Record name' field, 'www' is entered. The 'Record type' dropdown is set to 'A - Routes traffic to an IPv4 address and some AWS resources'. In the 'Route traffic to' section, 'Alias' is selected, and 'Alias to Application and Classic Load Balancer' is chosen from the dropdown. The 'Region' dropdown shows 'US East (N. Virginia)'. The 'Evaluate target health' checkbox is checked. The 'Routing policy' dropdown is set to 'Simple routing'. At the bottom right are 'Cancel' and 'Create records' buttons.

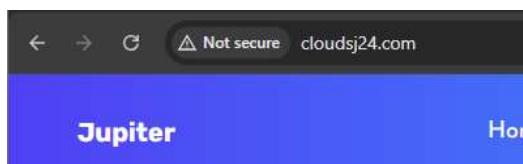
To check your domain is working, select your domain name from the records tab, then copy the record name in my case it will be 'www.cloudsj24.com'. Search for your domain name in the web browser and your domain name should show your website.



Register for an SSL certificate in AWS certificate manager

In this section, we'll register for a free SSL certificate from the AWS certificate manager and we'll use the SSL certificate to encrypt all communications between the web browser and our web servers.

Currently for our website the communication between the website and web servers are not secure.



Note this is called encryption in transit.

Now search for 'certificate manager' in the management console then click 'request a certificate'. We are going to request a 'public certificate'. Next you will need to enter your domain name and also add another name to the certificate. The second domain name will start with a '*' As this will cover the www as the beginning of your domain name. All other setting can be left as default and click request.

Domain names
Provide one or more domain names for your certificate.

Fully qualified domain name | Info

cloudsj24.com Remove

*.cloudsj24.com Remove

Add another name to this certificate

You can add additional names to this certificate. For example, if you're requesting a certificate for "www.example.com", you might want to add the name "example.com" so that customers can reach your site by either name.

Validation method Info

Select a method for validating domain ownership.

DNS validation - recommended

Choose this option if you are authorized to modify the DNS configuration for the domains in your certificate request.

Email validation

Choose this option if you do not have permission or cannot obtain permission to modify the DNS configuration for the domains in your certificate request.

Key algorithm Info

Select an encryption algorithm. Some algorithms may not be supported by all AWS services.

RSA 2048

RSA is the most widely used key type.

ECDSA P 256

Equivalent in cryptographic strength to RSA 3072.

ECDSA P 384

Equivalent in cryptographic strength to RSA 7680.

Great! You have created an SSL certificate. Now you will see that the status is saying 'pending validation'. This is because we need to create a record set in route 53 to validate the domain name belongs to us.

To do this, click 'create record in route 53'.

c5811cb9-da74-4da2-9a70-aeb266e77251 Delete

Certificate status

| Identifier | Status |
|--------------------------------------|--|
| c5811cb9-da74-4da2-9a70-aeb266e77251 | Pending validation <small>Info</small> |

ARN

arn:aws:acm:us-east-1:975049930561:certificate/c5811cb9-da74-4da2-9a70-aeb266e77251

Type

Amazon Issued

Domains (2)

Create records in Route 53 Export to CSV

| Domain | Status | Renewal status |
|-----------------|--------------------|----------------|
| cloudsj24.com | Pending validation | - |
| *.cloudsj24.com | Pending validation | - |

Make sure to select your domain and the wild card then click 'create record'. Next you will need to refresh the page and the status of your certificate should be 'issued'.

The screenshot shows the AWS Certificate Manager (ACM) interface. At the top, there's a breadcrumb navigation: 'c5811cb9-da74-4da2-9a70-aeb266e77251'. To the right are three small icons: a help icon, a copy icon, and a delete icon. Below this, the certificate identifier '5811cb9-da74-4da2-9a70-aeb266e77251' is displayed, along with a 'Delete' button. A large section titled 'Certificate status' contains details: 'Identifier' (c5811cb9-da74-4da2-9a70-aeb266e77251), 'Status' (Issued), 'ARN' (arn:aws:acm:us-east-1:975049930561:certificate/c5811cb9-da74-4da2-9a70-aeb266e77251), and 'Type' (Amazon Issued). Below this is a table titled 'Domains (2)' with two entries: 'cloudsj24.com' and '*.cloudsj24.com', both of which have a status of 'Success'. There are buttons for 'Create records in Route 53' and 'Export to CSV'.

Create an HTTPS (SSL) listener for an application load balancer

In this section we will use the SSL certificate we registered to secure all communications to our website. To do this this in the management console search for 'EC2' and then on the left hand side scroll down and click load balancers.

Select your load balancer created earlier and scroll down to 'listeners and rules'. What we are going to do first is create the HTTPS listener. The HTTPS listener is going to secure the communication between the our end user and our website.

This means any time you see the lock icon in the web browser URL, this means the website is secure. In our case our website is not secure and by creating an HTTPS listener this will secure our website.

Now under the 'listeners and rules' tab select 'add listener'. From the protocol dropdown list select 'HTTPS', the target group will be the target group we created earlier in the documentation.

EC2 > Load balancers > P1-alb > Add listener

Listener details: HTTPS:443

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Listener configuration

The listener will be identified by the protocol and port.

| | |
|--|--|
| Protocol Used for connections from clients to the load balancer. | Port The port on which the load balancer is listening for connections. |
| HTTPS | 443 1-65535 |

Default actions | Info

The default action is used if no other rules apply. Choose the default action for traffic on this listener.

Authentication | Info

Authentication requires IPv4 connectivity to authentication endpoints. Learn more [\[?\]](#)

Use OpenID or Amazon Cognito
Include authentication using either OpenID Connect (OIDC) or Amazon Cognito.

Routing actions

Forward to target groups Redirect to URL Return fixed response

Forward to target group | Info

Choose a target group and specify routing weight or [Create target group](#) [\[?\]](#).

| Target group | Weight | Percent | |
|--|--------|---|------|
| P1-target-group Target type: IP, IPv4 | HTTP | <input checked="" type="radio"/> 1 0-999 | 100% |

[Add target group](#)

You can add up to 4 more target groups.

Target group stickiness | Info

Enables the load balancer to bind a user's session to a specific target group. To use stickiness the client must support cookies. If you want to bind a user's session to a specific target, turn on the Target Group attribute Stickiness.

Turn on target group stickiness

Next you will need to scroll down and under 'default SSL/TLS server certificate' make sure the certificate source 'from ACM' has been selected, and from the dropdown list select the SSL certificate created earlier. This should be the name of your domain name.

Secure listener settings | Info

Security policy | Info

Your load balancer uses a Secure Socket Layer (SSL) negotiation configuration called a security policy to manage SSL connections with clients. Compare security policies [\[?\]](#)

| | |
|--------------------------|---|
| Security category | Policy name |
| All security policies | ELBSecurityPolicy-TLS13-1-2-2021-06 (recommended) |

Default SSL/TLS server certificate

The certificate used if a client connects without SNI protocol, or if there are no matching certificates. You can source this certificate from AWS Certificate Manager (ACM), Amazon Identity and Access Management (IAM), or import a certificate. This certificate will automatically be added to your listener certificate list.

Certificate source

From ACM From IAM Import certificate

Certificate (from ACM)

The selected certificate will be applied as the default SSL/TLS server certificate for this load balancer's secure listeners.

| | |
|------------------------------------|----------------------------------|
| cloudsj24.com | <input checked="" type="radio"/> |
| c5811cb9-da74-4da2-9a70-aeb266e... | [?] |

[Request new ACM certificate](#) [\[?\]](#)

Client certificate handling | Info

Client certificates are used to make authenticated requests to remote servers. Learn more [\[?\]](#)

Mutual authentication (mTLS)
Mutual TLS (Transport Layer Security) authentication offers two-way peer authentication. It adds a layer of security over TLS and allows your services to verify the client that's making the connection.

Now if you scroll down to 'listeners and rules' you should have two listeners, one on port 80 and one on port 443. The listener on port 443 is forwarding traffic to our target group and this is also being replicated by the listener on port 80. However, we need to modify the listener on port 80, so that any time a user tries to access our site on port 80 they will be redirected to port 443 for HTTPS.

Select the 'HTTP:80' listener, click the manager listener dropdown list and select 'edit listener'. Once in the edit listener page, under default actions select 'redirect to URL' and under redirect to URL select 'full URL' then save changes.

Listener details

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Listener ARN

arn:aws:elasticloadbalancing:us-east-1:975049930561:listener/app/P1-alb/e2313305f5e3e86a/635dbce79a737ee5

Listener configuration

The listener will be identified by the protocol and port.

Protocol

Used for connections from clients to the load balancer.

HTTP

Port

The port on which the load balancer is listening for connections.

80

1-65535

Default actions | Info

The default action is used if no other rules apply. Choose the default action for traffic on this listener.

Routing actions

Forward to target groups

Redirect to URL

Return fixed response

Redirect to URL | Info

Redirect client requests from one URL to another. You cannot redirect HTTPS to HTTP. To avoid a redirect loop, you must modify at least one of the following components: protocol, port, hostname or path. Components that you do not modify retain their original values.

URI parts

Full URL

Full URL | Info

Enter the redirect URL.

https://#{host}#/#{path}#{query}

protocol://hostname:port/path?query

Status code

301 - Permanently moved

Now if you go back to your ALB, you will see that under 'listeners and Rules' the HTTP:80 listener is being redirected to HTTPS 443. This is all we need to do to encrypt the communication between the end user and website.

To confirm your site is secure, go back to your website and you should see the lock icon in URL.



