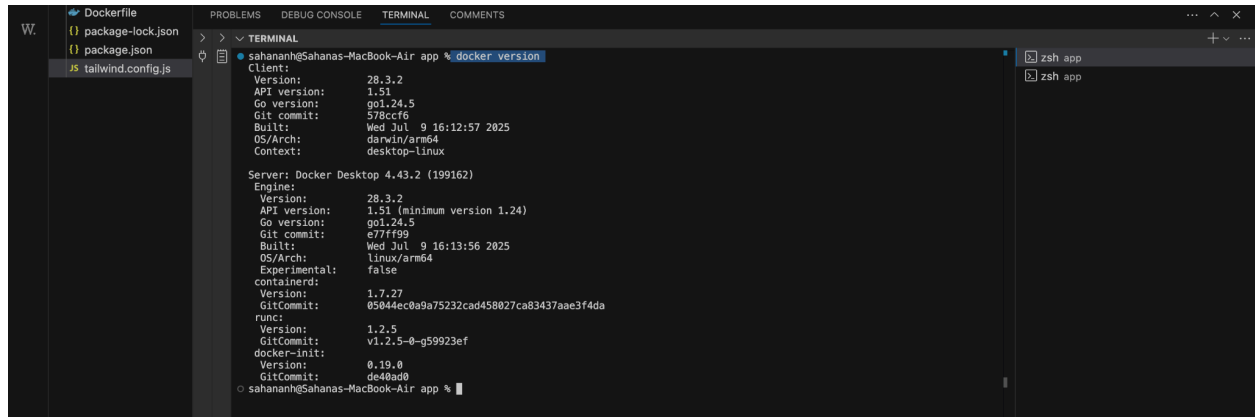


AWS - Elastic Container Service (ECS)

Pre-requisite

Install docker

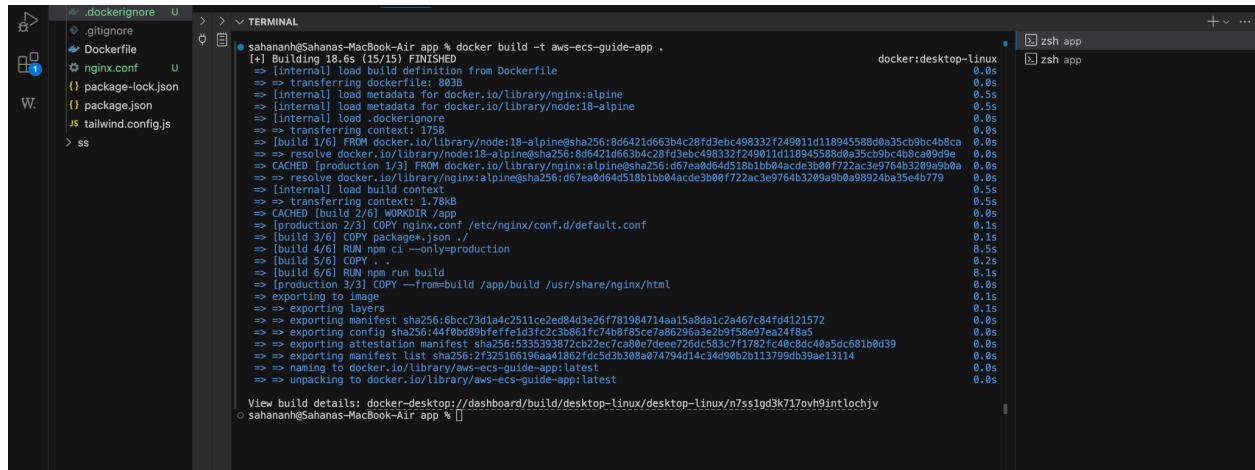


The screenshot shows a terminal window with the command `docker version` executed. The output displays the client and server versions, along with various system details like OS, architecture, and build information.

```
Client:
Version:      28.3.2
API version:  1.51
Go version:   go1.24.5
Git commit:   578ccf6
Built:        Wed Jul 9 16:12:57 2025
OS/Arch:      darwin/arm64
Context:      desktop-linux

Server: Docker Desktop 4.43.2 (199162)
Engine:
Version:      28.3.2
API version:  1.51 (minimum version 1.24)
Go version:   go1.24.5
Git commit:   e77ff99
Built:        Wed Jul 9 16:13:56 2025
OS/Arch:      linux/arm64
Experimental: false
containerd:
Version:      1.7.27
GitCommit:    05044ec0a9a75232cad458027ca83437aae3f4da
runc:
Version:      1.2.5
GitCommit:    v1.2.5-0-g59923ef
docker-init:
Version:      0.19.0
GitCommit:    de40bad9
```

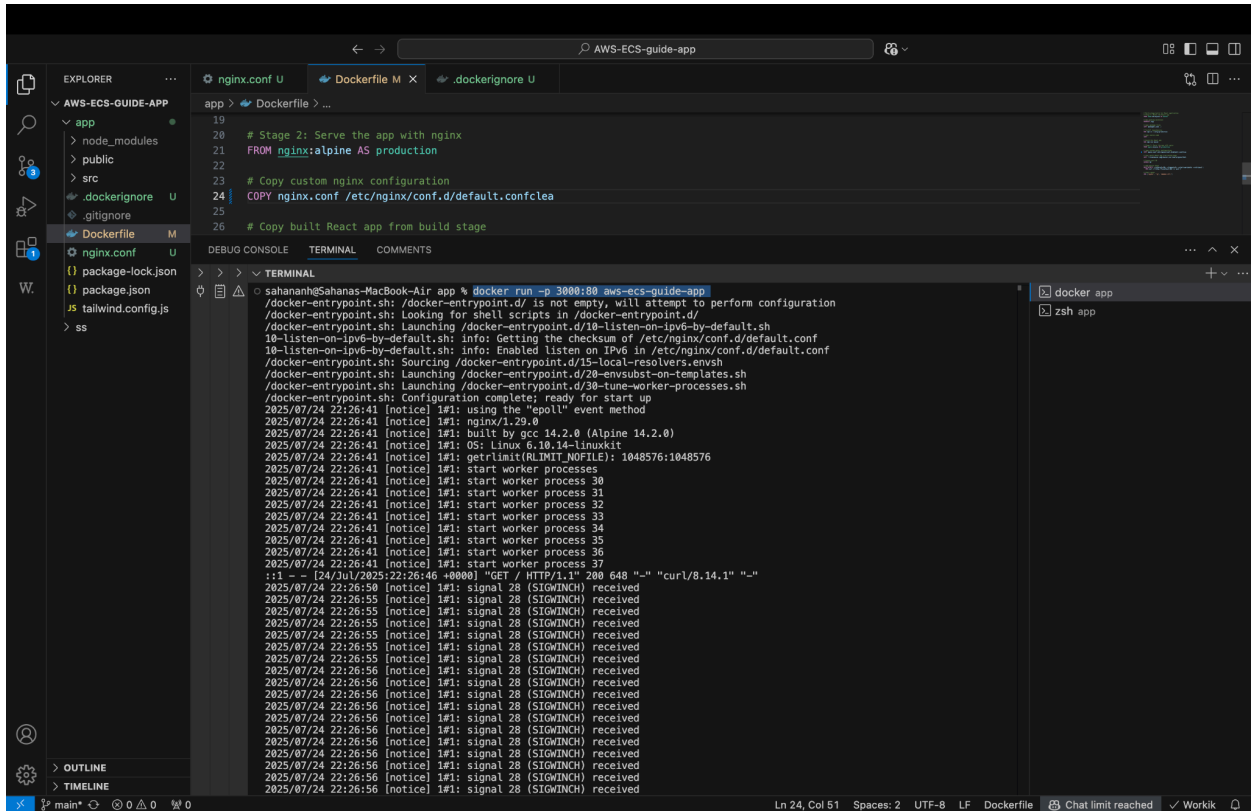
Build the Docker Image Locally



The screenshot shows a terminal window with the command `docker build -t aws-ecs-guide-app .` executed. The output displays the build process, including the transfer of the Dockerfile, loading of metadata, and the final export of the image.

```
[+] Building 18.6s (15/15) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 803B
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [internal] load .dockerignore
=> transferring context: 175B
=> [build 1/6] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28f43ebc498337f249011d118945588d0a35cb9bc4b8ca
=> => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28f43ebc498337f249011d118945588d0a35cb9bc4b8ca
=> CACHED [production 1/3] FROM docker.io/library/nginx:alpine@sha256:d67ea0d64d518b1bb04acde3b00f722ac3e9764b3209a9b0ea
=> => resolve docker.io/library/nginx:alpine@sha256:d67ea0d64d518b1bb04acde3b00f722ac3e9764b3209a9b0ea
=> [internal] load build context
=> transferring context: 1.78kB
=> CACHED [build 2/6] WORKDIR /app
=> [production 2/3] COPY nginx.conf /etc/nginx/conf.d/default.conf
=> [build 3/6] COPY package*.json ./
=> [build 4/6] RUN npm ci --only=production
=> [build 5/6] COPY . .
=> [build 6/6] RUN npm run build
=> [production 3/3] COPY --from=build /app/build /usr/share/nginx/html
=> exporting to image
=> exporting layers
=> exporting manifest sha256:6bcc73d1a4c2511ce2ed84d3e26f781984714aa15a8da1c2a467c84fd4121572
=> exporting config sha256:44f6b0898feffed2f1c2c3b861fc7486f85ce7a86298a3e2b9f58e97ce24f8a5
=> exporting attestation manifest sha256:5335393872cb22ec7ca80e7deee72ddc583c7f1f782c48c8dc40a5dc681b0d039
=> exporting manifest list sha256:2f325166196aa41862fcd5d3b308aa07494d14c34090b2b113799db39ae13114
=> naming to docker.io/library/aws-ecs-guide-app:latest
=> unpacking to docker.io/library/aws-ecs-guide-app:latest
```

Run the image



Login to ECR (Elastic Container Registry)

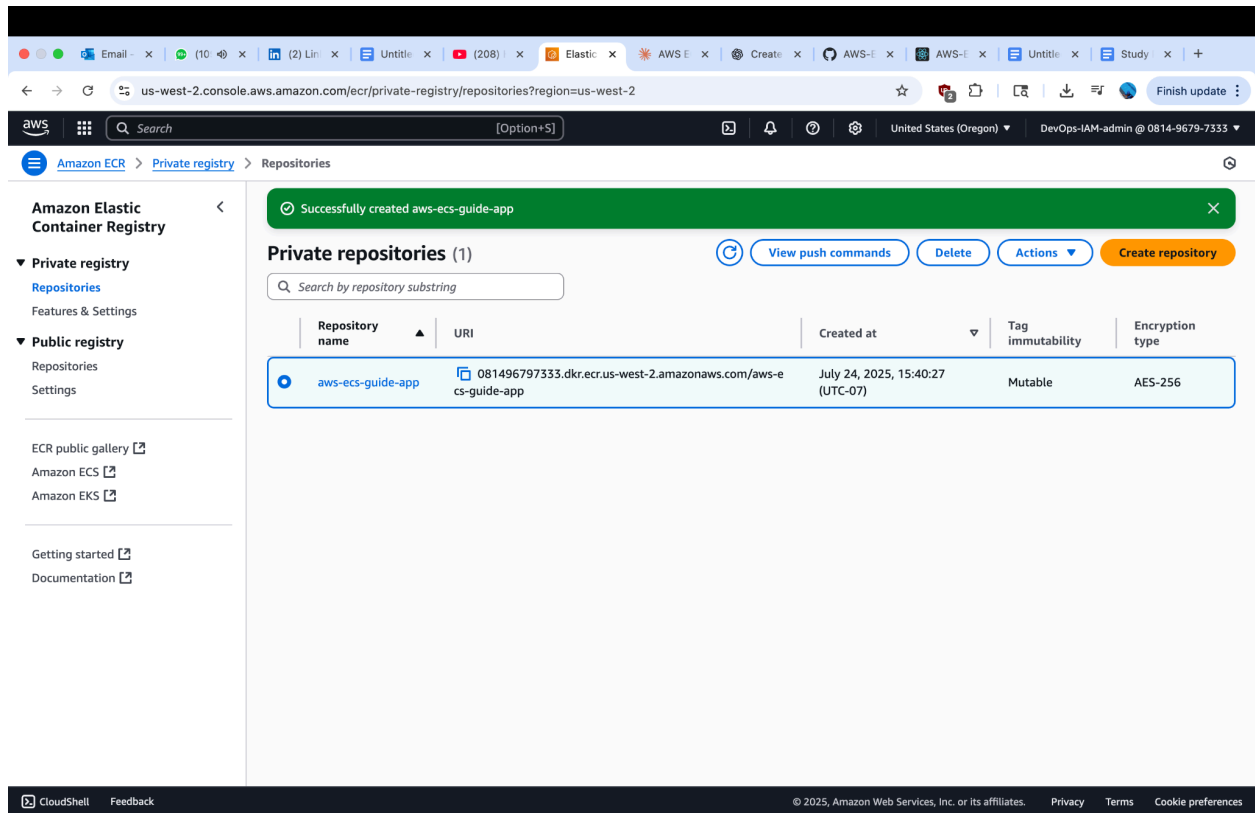
Before that, Make sure your **AWS CLI** is configured and has valid credentials
get your **AWS account number** by running the below command

```
aws sts get-caller-identity
```

Get the account id and run below cmd, to login to ECR. Make sure the region is correct.

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin  
<your-account-id>.dkr.ecr.us-east-1.amazonaws.com
```

Once logged in. Go to AWS console and create ECR repository and keep an eye on the region



Now , upload the docker image from local to ECR using the below cmd

Tag

```
docker tag <image-name>:latest
```

```
<your-account-id>.dkr.ecr.us-east-1.amazonaws.com/aws-ecs-guide-app:latest
```

Push

```
docker push <your-account-id>.dkr.ecr.us-east-1.amazonaws.com/aws-ecs-guide-app:latest
```

Once pushed, you can see the latest image on the ECR

aws Search [Option+S] United States (Oregon) DevOps-IAM-admin @ 0814-9679-7333

Amazon ECR > Private registry > Repositories > aws-ecs-guide-app

Amazon Elastic Container Registry

▼ Private registry

- Repositories
- Summary
- Images**
- Permissions
- Lifecycle Policy
- Repository tags
- Features & Settings

▼ Public registry

- Repositories
- Settings

ECR public gallery [↗](#)

Amazon ECS [↗](#)

Amazon EKS [↗](#)

Getting started [↗](#)

Documentation [↗](#)

Images (3) [Refresh](#) [Delete](#) [Details](#) [Scan](#) [View push commands](#)

Search artifacts

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Last recorded pull time
<input checked="" type="checkbox"/>	latest	Image Index	July 24, 2025, 16:56:28 (UTC-07)	23.16	Copy URI	sha256:60244daef57f4bf...	-
<input type="checkbox"/>	-	Image	July 24, 2025, 16:56:28 (UTC-07)	23.16	Copy URI	sha256:417d077a82e85e...	-
<input type="checkbox"/>	-	Image	July 24, 2025, 16:56:28 (UTC-07)	0.00	Copy URI	sha256:f00e57b2adb468...	-

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

Next, navigate to ECS and create a cluster, choose EC2/fargate , vpc, subnet, security group and click on create. It creates a cluster and ec2 instance specified.

us-west-2.console.aws.amazon.com/ecs/v2/clusters?region=us-west-2

aws Search [Option+S] United States (Oregon) DevOps-IAM-admin @ 0814-9679-7333

Amazon Elastic Container Service > Clusters

Amazon Elastic Container Service

Clusters

- Namespaces
- Task definitions
- Account settings

Amazon ECR [↗](#)

Repositories [↗](#)

AWS Batch [↗](#)

Documentation [↗](#)

Discover products [↗](#)

Subscriptions [↗](#)

Tell us what you think

Clusters (1) [Info](#) [Refresh](#) [Create cluster](#)

Search clusters

Cluster	Services	Tasks	Container instances	CloudWatch monitoring	Capacity
cluster1-ecs-guide	0	No tasks running	1 EC2	Default	ASG

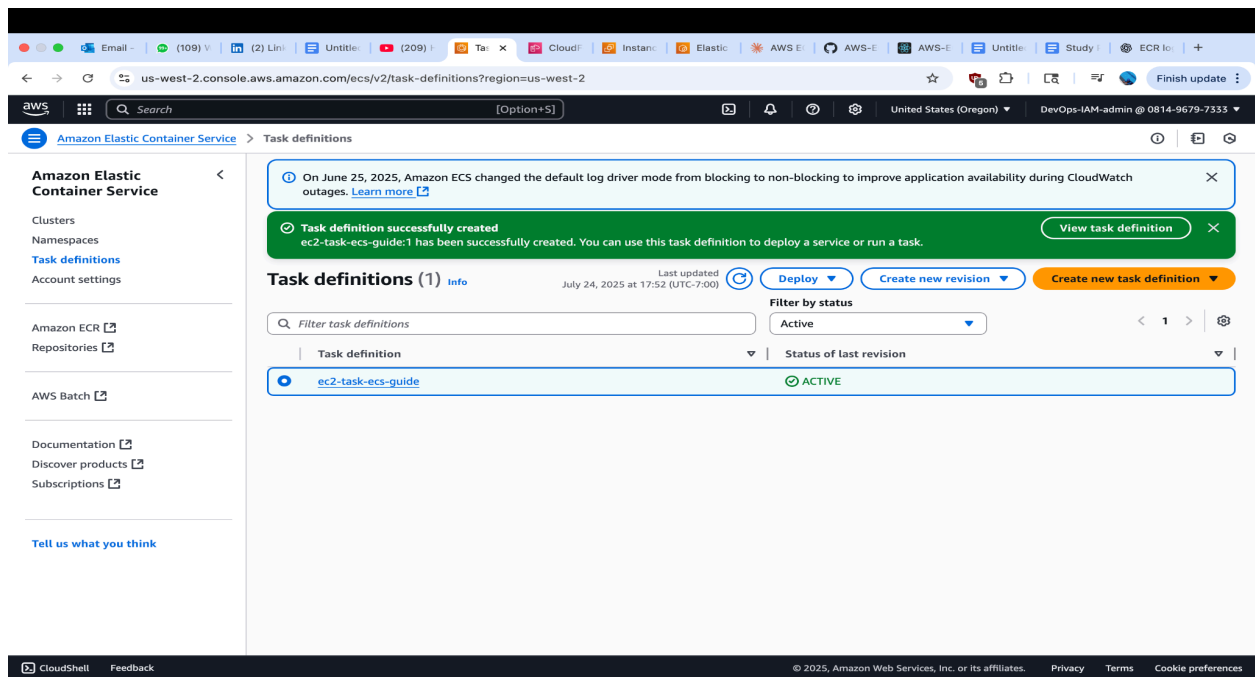
CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Verify the ec2 instance

The screenshot shows the AWS Management Console for the 'us-west-2' region. The left sidebar contains navigation links for EC2, including Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, and Elastic IPs. The main content area is titled 'Instances (1)' and shows a table with one instance. The instance is named 'ECS Instanc...', has an Instance ID of 'i-08ffc7d37940d154c', is in a 'Running' state, and has an Instance type of 't2.micro'. The Status check is 'Initializing', and the Alarm status is 'View alarms +'. The Availability Zone is 'us-west-2b' and the Public IP is 'ec2-35-9!'. The console also shows a 'Select an instance' section below the table.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
ECS Instanc...	i-08ffc7d37940d154c	Running	t2.micro	Initializing	View alarms +	us-west-2b	ec2-35-9!

Go and create a task and give the ECR repository URL while defining the container.



Once the task is created , go to the cluster and deploy a task by clicking on run task. After selecting the created task, the application in ECR that is docker image is going to deploy in the machine.

