Docker Image Deployment Guide

Overview

This document provides step-by-step instructions for building, running, and deploying a Docker container locally and then pushing it to **AWS Elastic Container Registry (ECR)** for deployment on **AWS Elastic Container Service (ECS)**.

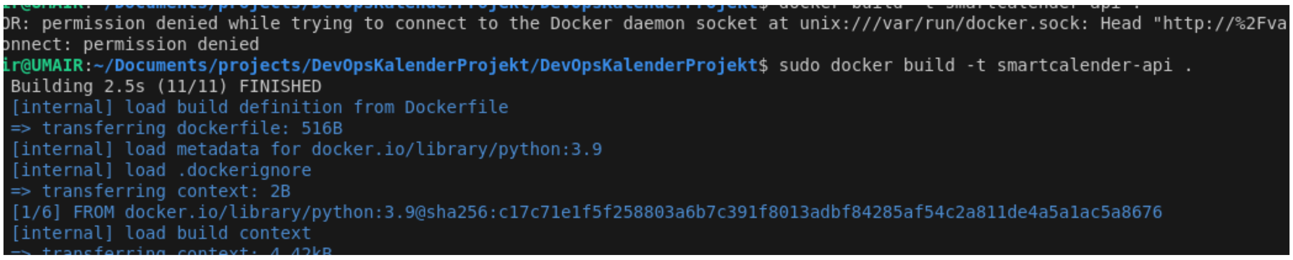
---

1. Build and Run Docker Image Locally

1.1 Build the Docker Image

To create a Docker image for your FastAPI application, run:

```
sudo docker build -t my-fastapi-app .
```



```
OR: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http://%2Fva
onnect: permission denied
ir@UMAIR:~/Documents/projects/DevOpsKalenderProjekt/DevOpsKalenderProjekt$ sudo docker build -t smartcalender-api .
 Building 2.5s (11/11) FINISHED
 [internal] load build definition from Dockerfile
 => transferring dockerfile: 516B
 [internal] load metadata for docker.io/library/python:3.9
 [internal] load .dockerignore
 => transferring context: 2B
 [1/6] FROM docker.io/library/python:3.9@sha256:c17c71e1f5f258803a6b7c391f8013adbf84285af54c2a811de4a5a1ac5a8676
 [internal] load build context
```

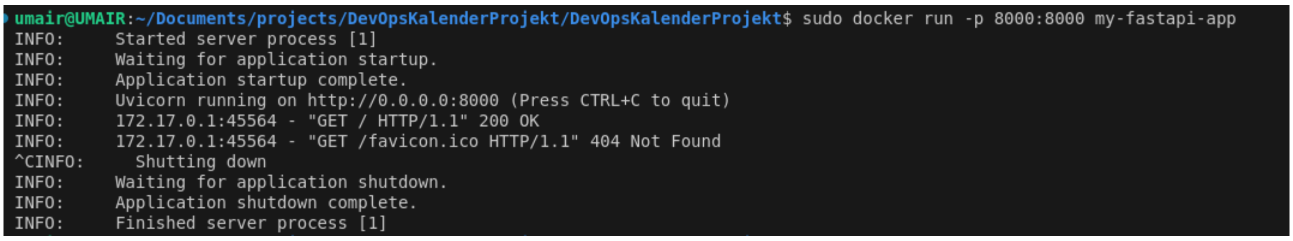
This command:

- Uses the `Dockerfile` in the current directory (`.`) to build the image.
- Tags the image as `my-fastapi-app`.

1.2 Run the Docker Container Locally

Execute the following command to run the container:

```
sudo docker run -p 8000:8000 my-fastapi-app
```



```
umair@UMAIR:~/Documents/projects/DevOpsKalenderProjekt/DevOpsKalenderProjekt$ sudo docker run -p 8000:8000 my-fastapi-app
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     172.17.0.1:45564 - "GET / HTTP/1.1" 200 OK
INFO:     172.17.0.1:45564 - "GET /favicon.ico HTTP/1.1" 404 Not Found
^CINFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [1]
```

This maps:

- Port `8000` inside the container to port `8000` on the local machine.

The application should now be accessible at:

```
http://localhost:8000
```

---

2. Push Docker Image to AWS Elastic Container Registry (ECR)

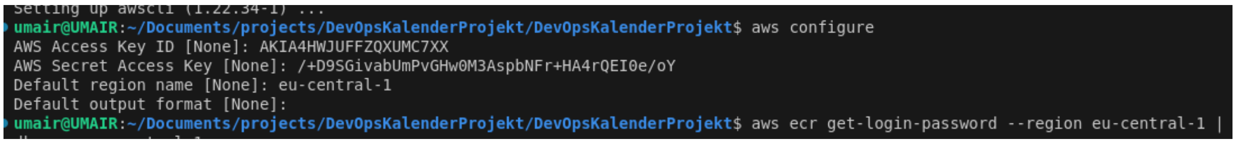
2.1 Configure AWS CLI

Ensure AWS CLI is configured with valid credentials:

```
aws configure
```

This prompts for:

- AWS **Access Key**
- AWS **Secret Key**
- AWS **Region** (e.g., `eu-central-1`)



```
Setting up awscli (1.22.34-1) ...
umair@UMAIR:~/Documents/projects/DevOpsKalenderProjekt/DevOpsKalenderProjekt$ aws configure
AWS Access Key ID [None]: AKIA4HWJUFFZQXUMC7XX
AWS Secret Access Key [None]: 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
Default region name [None]: eu-central-1
Default output format [None]:
umair@UMAIR:~/Documents/projects/DevOpsKalenderProjekt/DevOpsKalenderProjekt$ aws ecr get-login-password --region eu-central-1 |
```

-

2.2 Authenticate Docker with AWS ECR

Use the following command to authenticate Docker with AWS ECR:

```
aws ecr get-login-password --region eu-central-1 | docker login --username AWS
--password-stdin 841162697075.dkr.ecr.eu-central-1.amazonaws.com
```

This logs Docker into AWS ECR to allow pushing images.

2.3 Build the Docker Image for AWS ECR

Run:

```
docker buildx build --platform linux/amd64 -t smartcalender-api .
```

This tags the image as `smartcalender-api`.

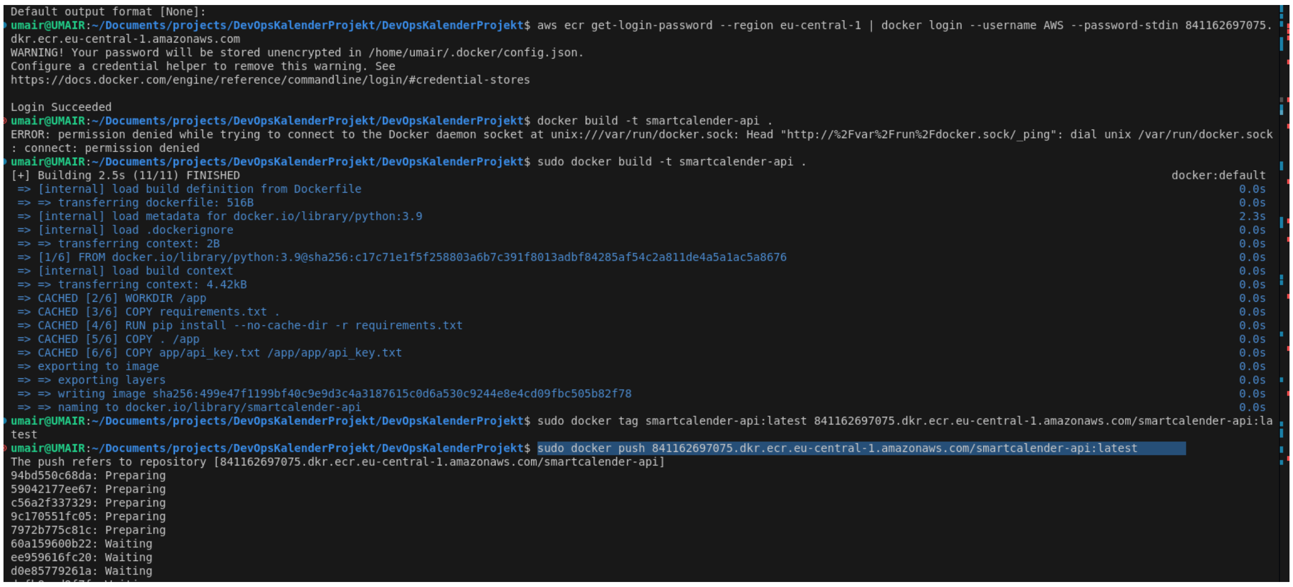
2.4 Tag the Docker Image for AWS ECR

```
docker tag smartcalender-api:latest
841162697075.dkr.ecr.eu-central-1.amazonaws.com/smartcalender-api:latest
```

This assigns the correct tag needed to push the image to ECR.

2.5 Push the Docker Image to AWS ECR

docker push 841162697075.dkr.ecr.eu-central-1.amazonaws.com/smartcalender-api:latest



```
Default output format [None]:
umair@UMAIR:~/Documents/projects/DevOpsKalenderProjekt/DevOpsKalenderProjekt$ aws ecr get-login-password --region eu-central-1 | docker login --username AWS --password-stdin 841162697075.
dkr.ecr.eu-central-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/umair/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
umair@UMAIR:~/Documents/projects/DevOpsKalenderProjekt/DevOpsKalenderProjekt$ docker build -t smartcalender-api .
ERROR: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http://%2Fvar%2Frun%2Fdocker.sock/_ping": dial unix /var/run/docker.sock
: connect: permission denied
umair@UMAIR:~/Documents/projects/DevOpsKalenderProjekt/DevOpsKalenderProjekt$ sudo docker build -t smartcalender-api .
[+] Building 2.5s (11/11) FINISHED                                                                                        docker:default
 => [internal] load build definition from Dockerfile                                                                               0.0s
 => => transferring dockerfile: 516B                                                                                               0.0s
 => [internal] load metadata for docker.io/library/python:3.9                                                                      2.3s
 => [internal] load .dockerignore                                                                                                  0.0s
 => => transferring context: 2B                                                                                                    0.0s
 => [1/6] FROM docker.io/library/python:3.9@sha256:c17c71e1f5f258803a6b7c391f8013adbf84285af54c2a811de4a5a1ac5a8676                0.0s
 => [internal] load build context                                                                                                  0.0s
 => => transferring context: 4.42kB                                                                                                0.0s
 => CACHED [2/6] WORKDIR /app                                                                                                      0.0s
 => CACHED [3/6] COPY requirements.txt .                                                                                           0.0s
 => CACHED [4/6] RUN pip install --no-cache-dir -r requirements.txt                                                                0.0s
 => CACHED [5/6] COPY . /app                                                                                                       0.0s
 => CACHED [6/6] COPY app/api_key.txt /app/app/api_key.txt                                                                         0.0s
 => exporting to image                                                                                                             0.0s
 => => exporting layers                                                                                                            0.0s
 => => writing image sha256:499e47f1199bf40c9e9d3c4a3187615c0d6a530c9244e8e4cd09fbc505b82f78                                       0.0s
 => => naming to docker.io/library/smartcalender-api                                                                               0.0s
umair@UMAIR:~/Documents/projects/DevOpsKalenderProjekt/DevOpsKalenderProjekt$ sudo docker tag smartcalender-api:latest 841162697075.dkr.ecr.eu-central-1.amazonaws.com/smartcalender-api:la
test
umair@UMAIR:~/Documents/projects/DevOpsKalenderProjekt/DevOpsKalenderProjekt$ sudo docker push 841162697075.dkr.ecr.eu-central-1.amazonaws.com/smartcalender-api:latest
The push refers to repository [841162697075.dkr.ecr.eu-central-1.amazonaws.com/smartcalender-api]
94bd550c68da: Preparing
59042177ee67: Preparing
c56a2f337329: Preparing
9c170551fc05: Preparing
7972b775c81c: Preparing
60a159600b22: Waiting
ee959616fc20: Waiting
d0e85779261a: Waiting
```

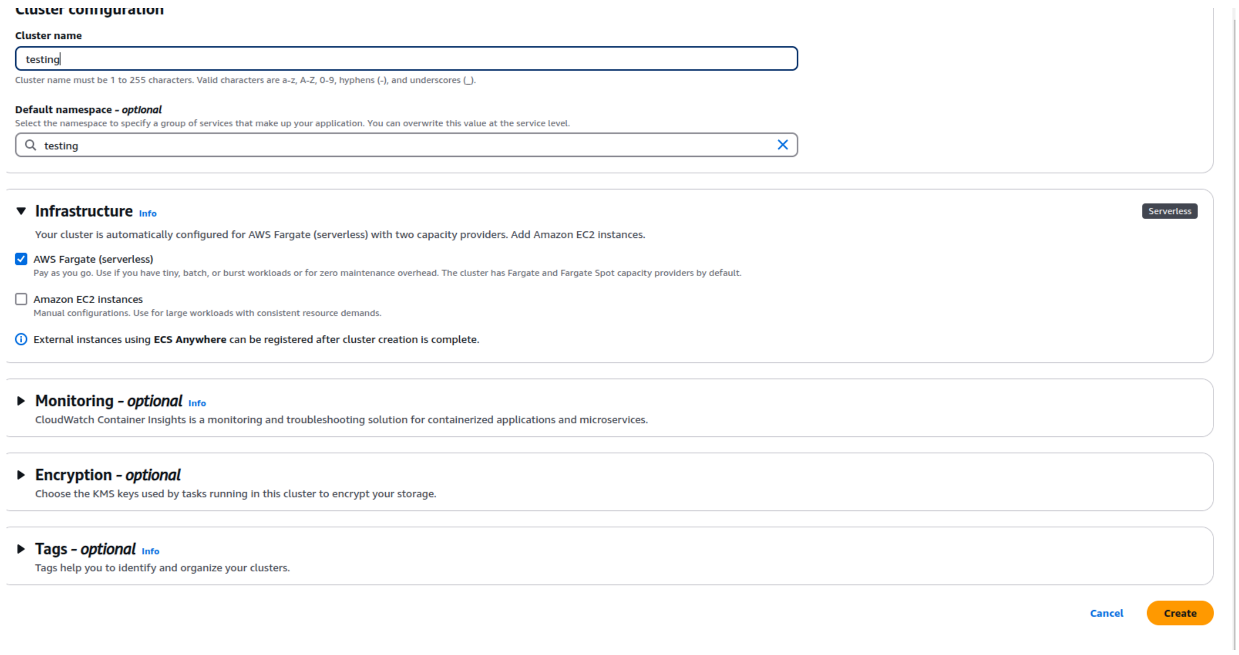
---

3. Deploy Docker Image to AWS ECS

After pushing the image to **AWS ECR**, use it in **AWS ECS** by:

1. Creating an **ECS Cluster**.

   Give name of cluster and click on create



Cluster configuration

Cluster name

testing

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.

testing

Infrastructure Info

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.

Encryption - *optional*

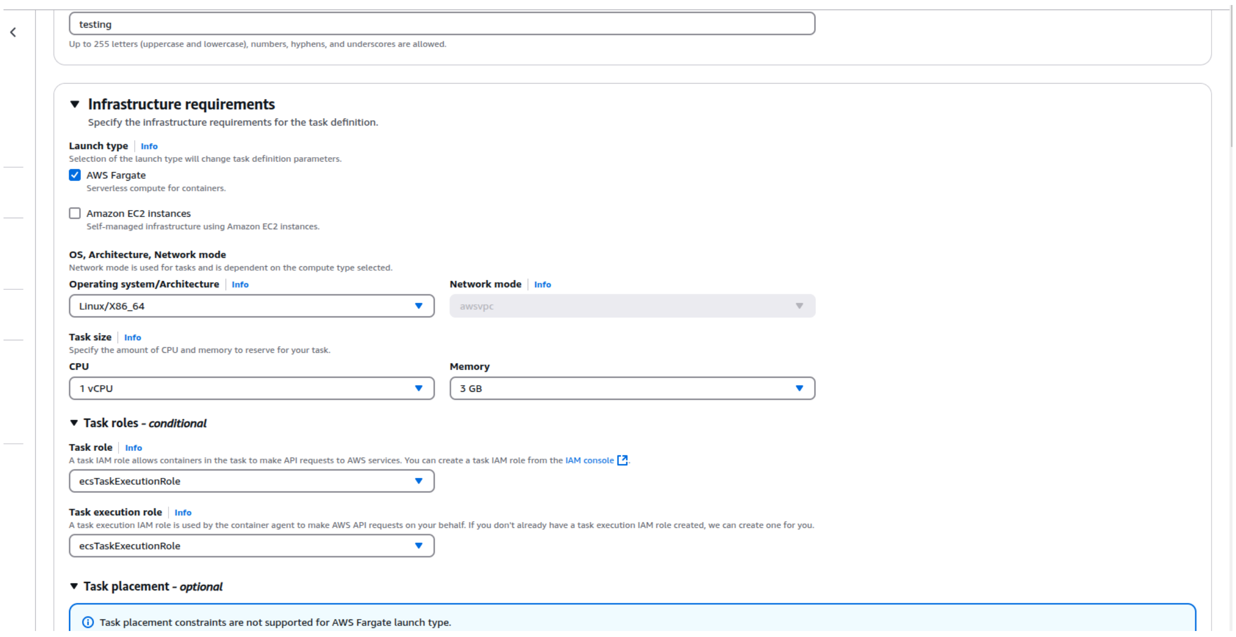
Choose the KMS keys used by tasks running in this cluster to encrypt your storage.

Tags - *optional* Info

Tags help you to identify and organize your clusters.

2. Defining a **Task Definition** referencing the ECR image.

   Give name of the **Task definition family**
   select **Task role** and **Task execution role** from dropdown



testing

Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

Infrastructure requirements
Specify the infrastructure requirements for the task definition.

Launch type | Info
Selection of the launch type will change task definition parameters.

☑ AWS Fargate
Serverless compute for containers.

☐ Amazon EC2 instances
Self-managed infrastructure using Amazon EC2 instances.

OS, Architecture, Network mode
Network mode is used for tasks and is dependent on the compute type selected.

Operating system/Architecture | Info: Linux/X86_64

Network mode | Info: awsvpc

Task size | Info
Specify the amount of CPU and memory to reserve for your task.

CPU: 1 vCPU

Memory: 3 GB

Task roles - *conditional*

Task role | Info
A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the IAM console

ecsTaskExecutionRole

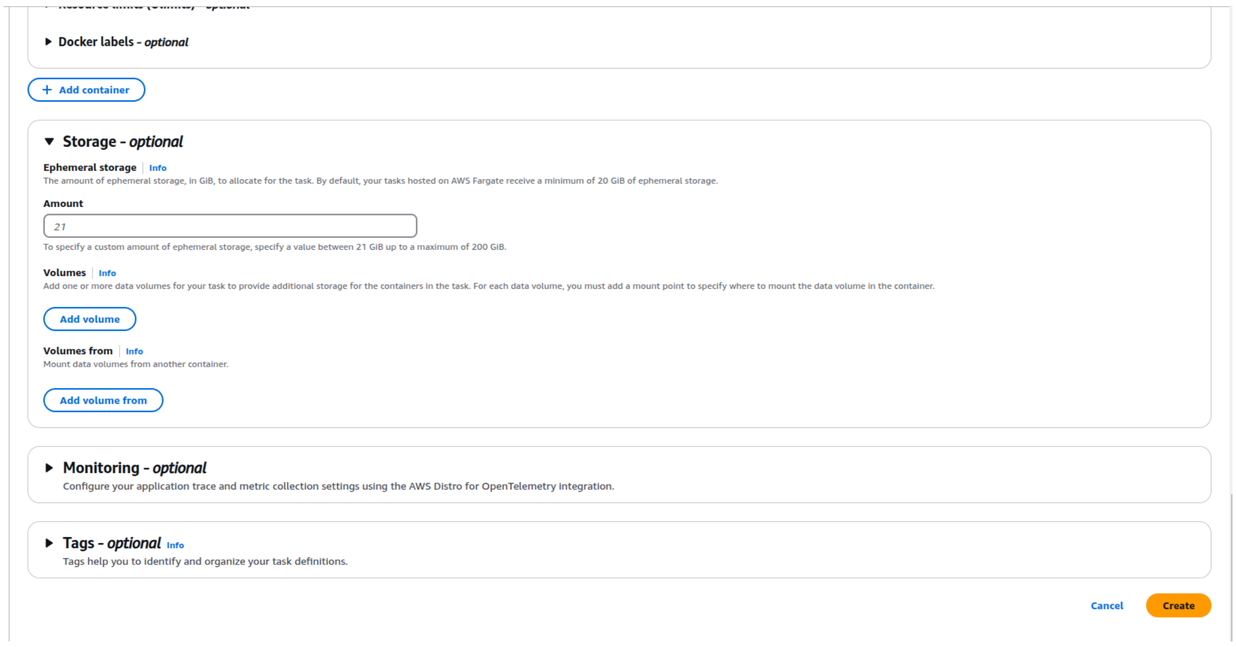
Task execution role | Info
A task execution IAM role is used by the container agent to make AWS API requests on your behalf. If you don't already have a task execution IAM role created, we can create one for you.

ecsTaskExecutionRole

Task placement - *optional*

Task placement constraints are not supported for AWS Fargate launch type.

select the **CPU and Memory** according to your requirement



Docker labels - *optional*

Add container

Storage - *optional*

Ephemeral storage | Info
The amount of ephemeral storage, in GiB, to allocate for the task. By default, your tasks hosted on AWS Fargate receive a minimum of 20 GiB of ephemeral storage.

Amount

21

To specify a custom amount of ephemeral storage, specify a value between 21 GiB up to a maximum of 200 GiB.

Volumes | Info
Add one or more data volumes for your task to provide additional storage for the containers in the task. For each data volume, you must add a mount point to specify where to mount the data volume in the container.

Add volume

Volumes from | Info
Mount data volumes from another container.

Add volume from

Monitoring - *optional*
Configure your application trace and metric collection settings using the AWS Distro for OpenTelemetry integration.

Tags - *optional* Info
Tags help you to identify and organize your task definitions.

Cancel | Create

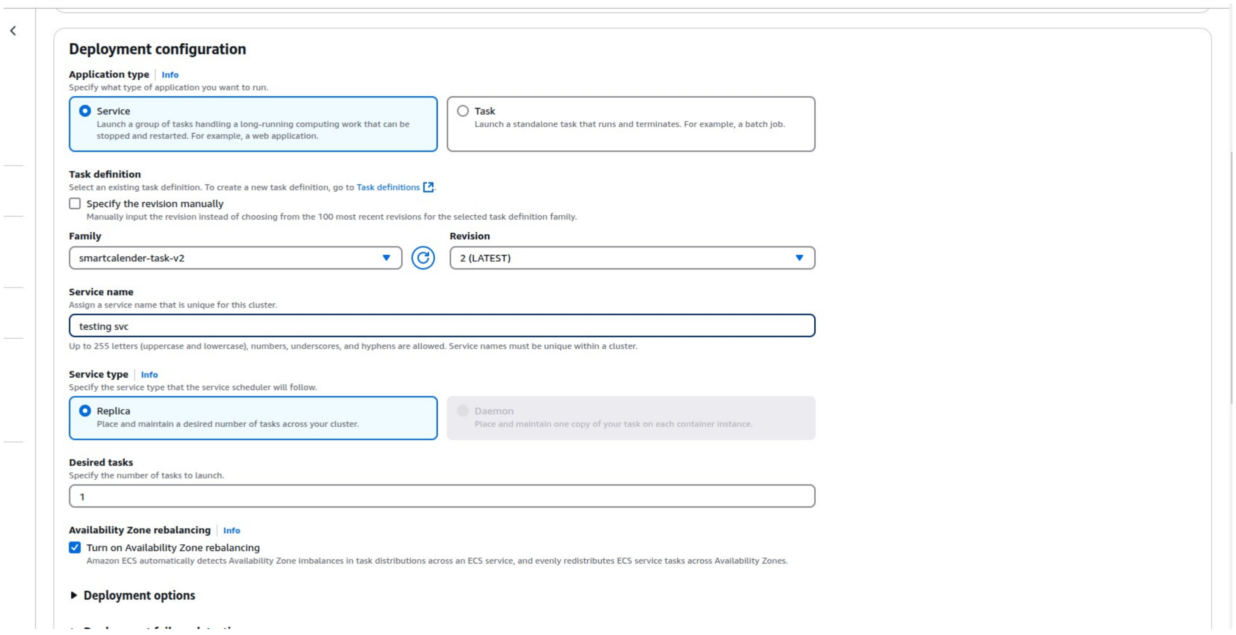
   leave everything same and click on done

3. Running an **ECS Service** using the Task Definition.

GO the cluster and and click on create to create service

select the task definition from **Family field** and select the **revision** from dropdown

give name of Service name in Service field



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: smartcalender-task-v2
Revision: 2 (LATEST)

Service name
Assign a service name that is unique for this cluster.
testing svc
Up to 255 letters (uppercase and lowercase), numbers, underscores, and hyphens are allowed. Service names must be unique within a cluster.

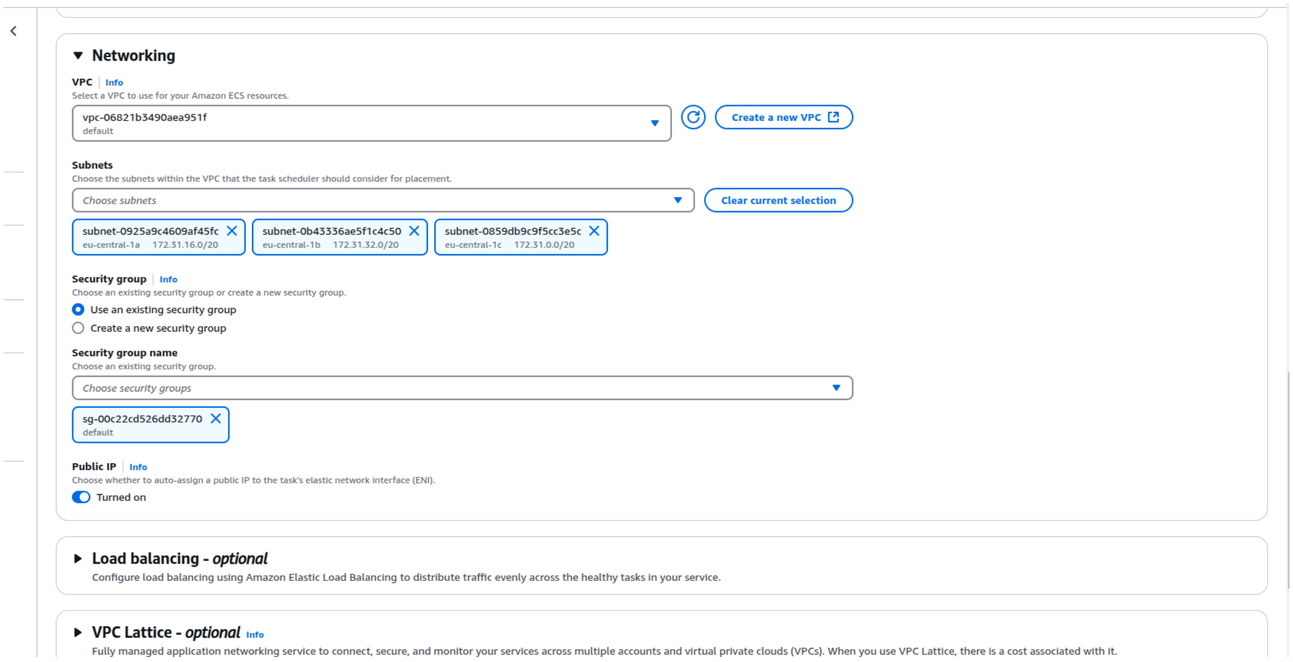
Service type | Info
Specify the service type that the service scheduler will follow.
[✓ Replica] Place and maintain a desired number of tasks across your cluster.
Daemon: Place and maintain one copy of your task on each container instance.

Desired tasks
Specify the number of tasks to launch.
1

Availability Zone rebalancing | Info
[✓] Turn on Availability Zone rebalancing
Amazon ECS automatically detects Availability Zone imbalances in task distributions across an ECS service, and evenly redistributes ECS service tasks across Availability Zones.

▸ Deployment options

In Networking tab select the default vpc and subnet



Networking

VPC | Info
Select a VPC to use for your Amazon ECS resources.
vpc-06821b3490aea951f
default

Subnets
Choose the subnets within the VPC that the task scheduler should consider for placement.
Choose subnets
subnet-0925a9c4609af45fc eu-central-1a 172.31.16.0/20
subnet-0b43336ae5f1c4c50 eu-central-1b 172.31.32.0/20
subnet-0859db9c9f5cc3e5c eu-central-1c 172.31.0.0/20

Security group | Info
Choose an existing security group or create a new security group.
[✓ Use an existing security group]
Create a new security group

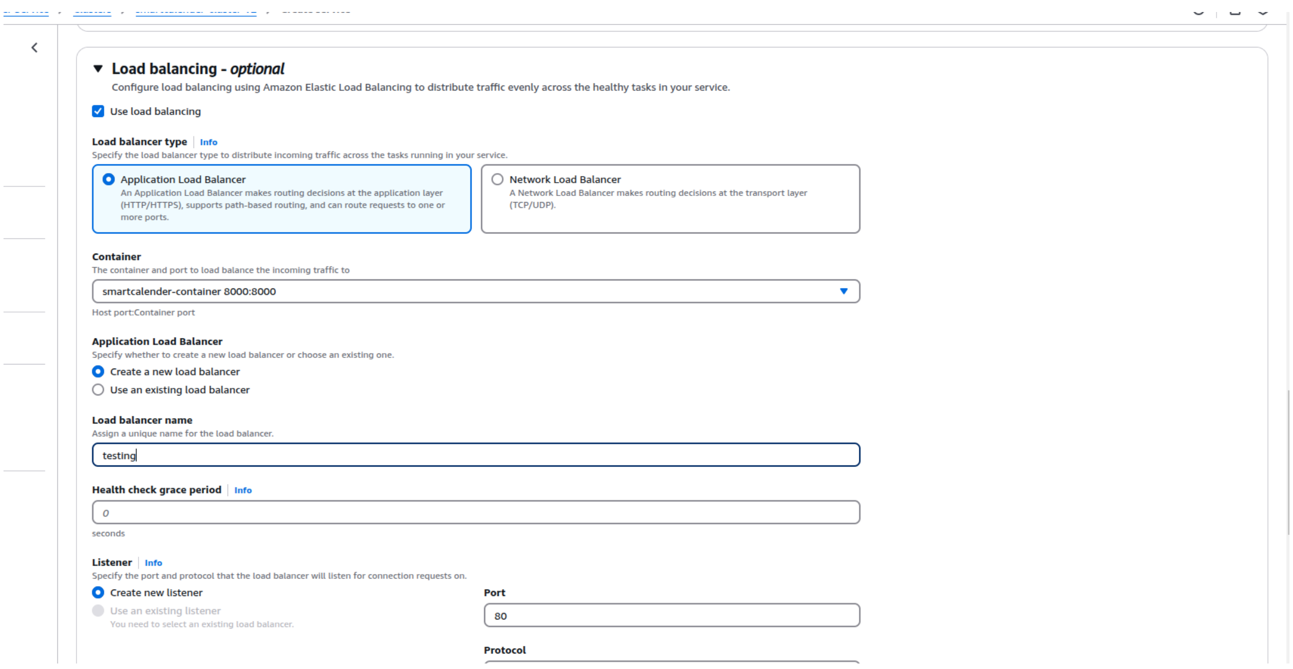
Security group name
Choose an existing security group.
Choose security groups
sg-00c22cd526dd32770
default

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

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

▸ **VPC Lattice - *optional*** Info
Fully managed application networking service to connect, secure, and monitor your services across multiple accounts and virtual private clouds (VPCs). When you use VPC Lattice, there is a cost associated with it.

In load balancer section
give the name loadbalancer and leave other thing remain same



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

smartcalender-container 8000:8000

Host port:Container port

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 name

Assign a unique name for the load balancer.

testing

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]

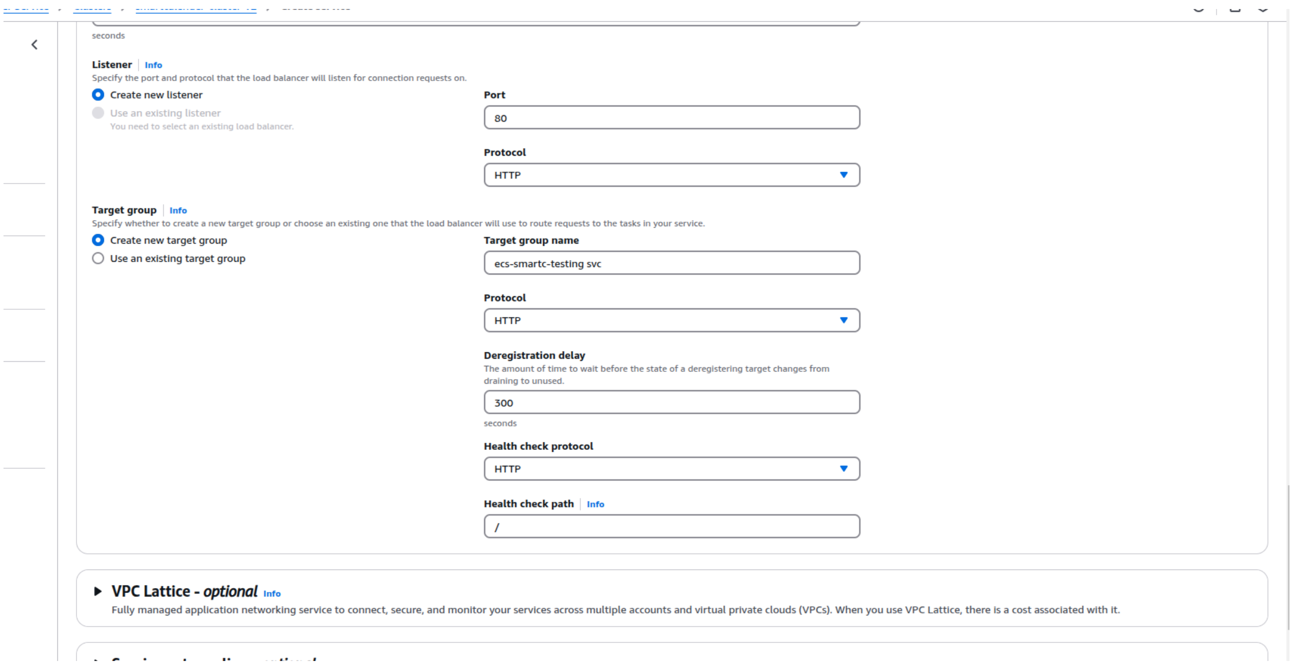
Use an existing listener — You need to select an existing load balancer.

Port

80

Protocol

Leave everything same and click on create service


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
  You need to select an existing load balancer.

Port: 80

Protocol: HTTP

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: ecs-smartc-testing svc

Protocol: HTTP

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

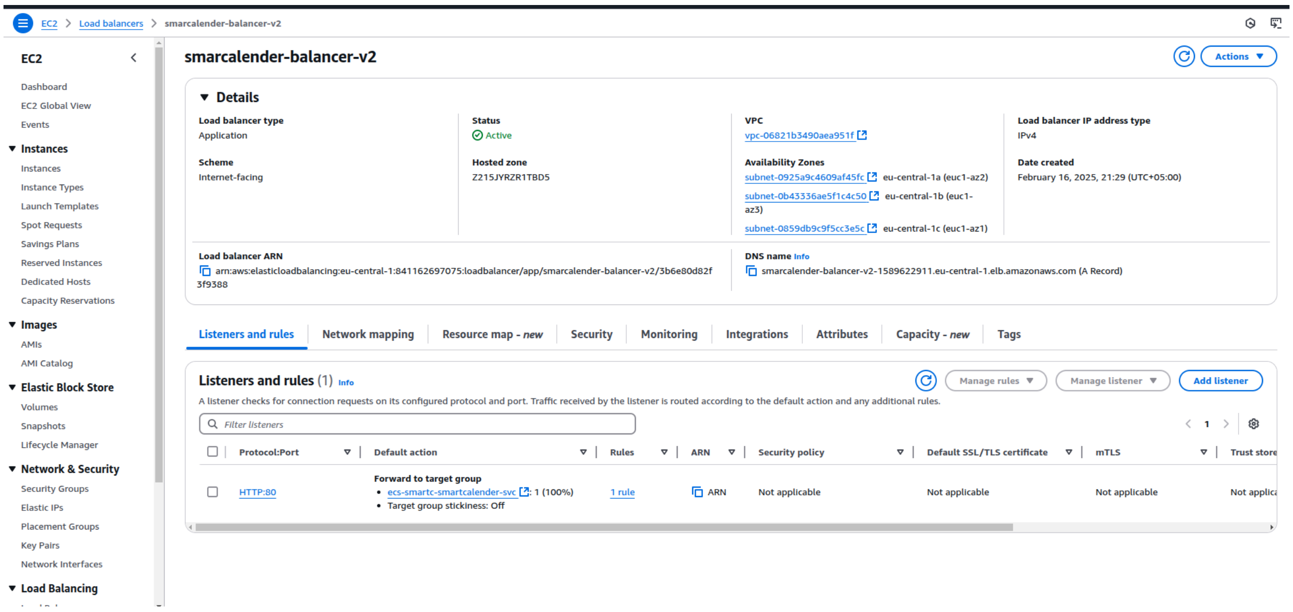
Health check path Info: /

VPC Lattice - *optional* Info
Fully managed application networking service to connect, secure, and monitor your services across multiple accounts and virtual private clouds (VPCs). When you use VPC Lattice, there is a cost associated with it.

To copy the **Application Load Balancer (ALB) DNS name**, follow these steps based on your setup:

Go to the AWS Management Console → Navigate to **EC2**.

- In the left panel, click on **Load Balancers** (under Load Balancing).
- Locate your **Application Load Balancer (ALB)**.
- Click on the ALB name to open its details.
- Find the **DNS name** under the **Basic Configuration** section.
- Click the **copy icon** next to the DNS name to copy it.



smarcalender-balancer-v2

Details

Load balancer type Application	**Status** Active	**VPC** vpc-06821b3490aea951f	**Load balancer IP address type** IPv4
Scheme Internet-facing	**Hosted zone** Z215JYRZR1TBD5	**Availability Zones** subnet-0925a9c4609af45fc eu-central-1a (euc1-az2); subnet-0b43336ae5f1c4c50 eu-central-1b (euc1-az3); subnet-0859db9c9f5cc3e5c eu-central-1c (euc1-az1)	**Date created** February 16, 2025, 21:29 (UTC+05:00)

Load balancer ARN
arn:aws:elasticloadbalancing:eu-central-1:841162697075:loadbalancer/app/smarcalender-balancer-v2/3b6e80d82f3f9388

DNS name Info
smarcalender-balancer-v2-1589622911.eu-central-1.elb.amazonaws.com (A Record)

Listeners and rules | Network mapping | Resource map - *new* | Security | Monitoring | Integrations | Attributes | Capacity - *new* | Tags

Listeners and rules (1) Info
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store
HTTP:80	**Forward to target group** • ecs-smartc-smartcalender-svc: 1 (100%) • Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable	Not applicable	Not applica