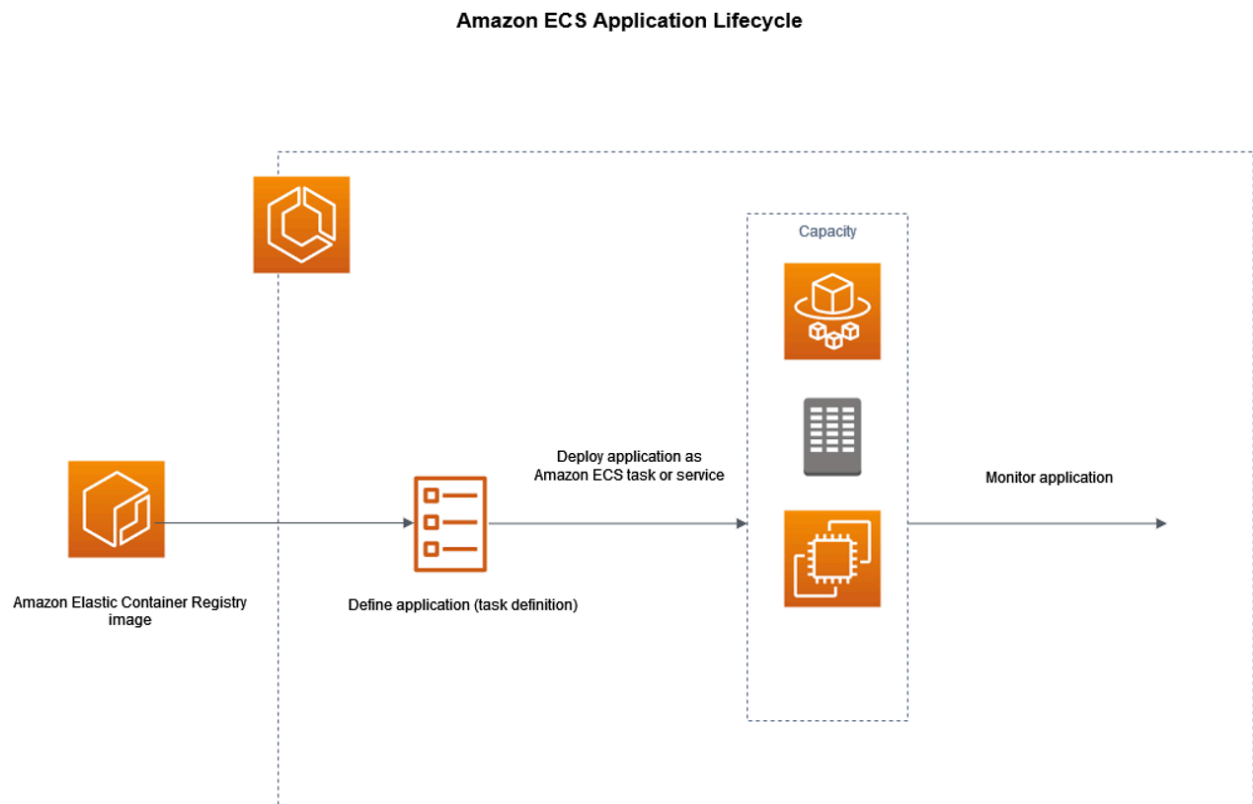


**TASK: i have to launch an ECS server, then store an apache or nginx image in ECR, ECR will call the image in ecs**

What is an NGINX image?

The NGINX image is a pre-built Docker image that contains the NGINX web server and its dependencies. This image is maintained and provided by the NGINX team or community contributors on Docker Hub. It allows you to run an instance of the NGINX web server in a containerized environment, providing a lightweight, portable way to deploy web applications.



After creation and storage of image, we create an Amazon ECS task definition-blueprint

Is a text file in JSON format that describes params and one or more containers that form your app

Specific params available for task def depend on needs of the specific apps

After task definition: deploy it as a service or task on your cluster

Cluster: local grouping of tasks or services that runs on cap infra registered to a cluster

Task is instantiation of a task def within a cluster

Use amazon ECS service to run and maintain desired number of tasks simultaneously in Amazon ECS cluster <- Amazon ECS service scheduler launches another instance based on task definition if any task fails or stops

### HOW IT WILL WORKS:

Dev pull Docker images and resources from ECR or other repos

This defines the application

Service ingests container images

Arranges/composes containers and resources into an app

Everything is gathered and services implemented

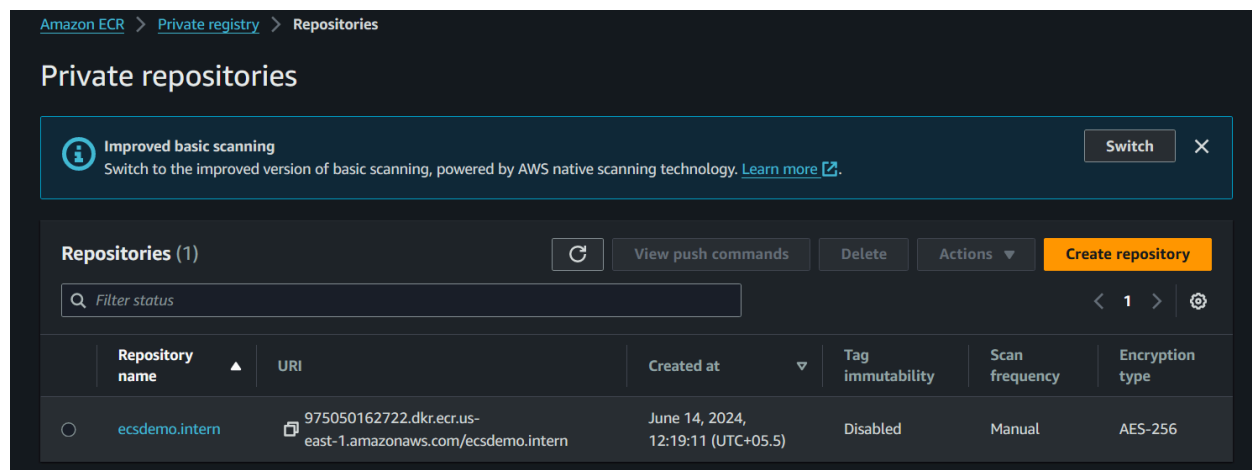
Containers deployed on EC2 or Fargate

ECS scales the app and continuously manages availability of cont

<https://medium.com/@ahmedSalem2020/deploying-a-docker-nginx-image-using-amazon-ecr-and-ecs-119e7d9b78d9>

CREATED AN ECR REPOSITORY:

[975050162722.dkr.ecr.us-east-1.amazonaws.com/ecsdemo.intern](https://975050162722.dkr.ecr.us-east-1.amazonaws.com/ecsdemo.intern)



INSTALLED DOCKER:

```
C:\Users\advik>docker --version
Docker version 26.1.1, build 4cf5afa

C:\Users\advik>docker-compose --version
Docker Compose version v2.27.0-desktop.2

C:\Users\advik>docker ps
CONTAINER ID   IMAGE                                COMMAND                  NAMES                  CREATED          STATUS
4ece778eb123   infiniflow/ragflow:v1.0             "/entrypoint.sh"        ragflow-server         2 months ago    Up 2 minut
es             0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp, 0.0.0.0:9380->9380/tcp
3db511b70c63   mysql:5.7.18                        "docker-entrypoint.s..." ragflow-mysql          2 months ago    Up 2 minut
es (healthy)   0.0.0.0:5455->3306/tcp
bb3c28565b79   quay.io/minio/minio:RELEASE.2023-12-20T01-00-02Z "/usr/bin/docker-ent..." ragflow-minio          2 months ago    Up 2 minut
es             0.0.0.0:9000-9001->9000-9001/tcp
319d3b839183   docker.elastic.co/elasticsearch/elasticsearch:8.11.3 "/bin/tini -- /usr/l..." ragflow-es-01          2 months ago    Up 2 minut
es (healthy)   9300/tcp, 0.0.0.0:1200->9200/tcp
```

UPDATED CLI2 BUT ERROR:

```
Requirement already satisfied: six>=1.5 in c:\users\advik\appdata\roaming\python\python311\site-packages (from python-dateutil<3.0.0,>=2.1->botocore==1.34.126->awscli) (1.16.0)

advik@advika MINGW64 ~
$ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 975050162722.dkr.ecr.us-east-1.amazonaws.com
bash: aws: command not found
Error: Cannot perform an interactive login from a non TTY device
```

CREATED EC2 SERVER:

The screenshot shows the 'Create new key pair' form in the AWS Management Console. The 'Key pair name' field is filled with 'demoecs.intern.key'. The 'Key pair type' is set to 'RSA'. The 'Private key file format' is set to '.pem'. The form includes instructions for key pair names and file formats.

Successfully initiated launch of instance ([i-020e5743ed1d94db4](#))

The screenshot shows the AWS Management Console 'Instances' page. The instance 'Intern\_ECS' (ID: i-020e5743ed1d94db4) is listed as 'Running'. Below the table, the details for this instance are shown, including its public and private IP addresses, DNS names, and instance type.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
Intern_...	i-020e5743ed1d94db4	Running	t2.micro	2/2 checks passer	View alarms +	us-east-1b	ec2-3-88-216

**i-020e5743ed1d94db4 (Intern\_ECS)**

Details	Status and alarms	Monitoring	Security	Networking	Storage	Tags
<b>Instance summary</b> Instance ID: i-020e5743ed1d94db4 (Intern_ECS) IPv6 address: - Hostname type: IP name: ip-172-31-20-87.ec2.internal Answer private resource DNS name	Public IPv4 address: 3.88.216.175   open address Instance state: Running Private IP DNS name (IPv4 only): ip-172-31-20-87.ec2.internal Instance type	Private IPv4 addresses: 172.31.20.87 Public IPv4 DNS: ec2-3-88-216-175.compute-1.amazonaws.com   open address Elastic IP addresses				

From <https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:>

## SETTING UP DOCKER AND AWS CLI ON WSL:

```
Get:48 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Package
s [70.1 kB]
Get:49 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-e
n [14.3 kB]
Get:50 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Compone
nts [212 B]
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Compone
nts [208 B]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f M
etadata [116 B]
Fetched 29.1 MB in 6s (5033 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
77 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-20-87:~$ sudo apt install awscli
```

need both of these tools to authentic

```
ubuntu@ip-172-31-20-87:~$ sudo docker --version
Docker version 26.1.4, build 5650f9b
ubuntu@ip-172-31-20-87:~$ /usr/local/bin/aws --version
aws-cli/2.16.8 Python/3.11.8 Linux/6.8.0-1008-aws exe/x86_64.ubuntu.24
```

## CONFIGURING AWS

```
root@ip-172-31-20-87:~# aws configure
AWS Access Key ID [None]: AKIA6GBMEYIRDQK3CIRV
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
root@ip-172-31-20-87:~# aws ecr get-login-password --region us-east-1 | docker l
ogin --username AWS --password-stdin 975050162722.dkr.ecr.us-east-1.amazonaws.co
m
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ip-172-31-20-87:~#
```

Created an ECS cluster with launch type EC2 INSTANCES

# 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

fargate-ec2-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 fargate-ec2-cluster



## ▼ Infrastructure Info

Customized

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.

Auto Scaling group (ASG) Info

Use Auto Scaling groups to scale the Amazon EC2 instances in the cluster.

Create new ASG



## fargate-ec2-cluster ASG



Update cluster

Delete cluster

### Cluster overview

ARN

arn:aws:ecs:us-east-1:97505016272:cluster/fargate-ec2-cluster

Status

Active

CloudWatch monitoring

Default

Registered container instances

-

### Services

Draining

-

Active

-

### Tasks

Pending

-

Running

-

### Encryption

Managed storage

-

Fargate ephemeral storage

-

Services

Tasks

Infrastructure

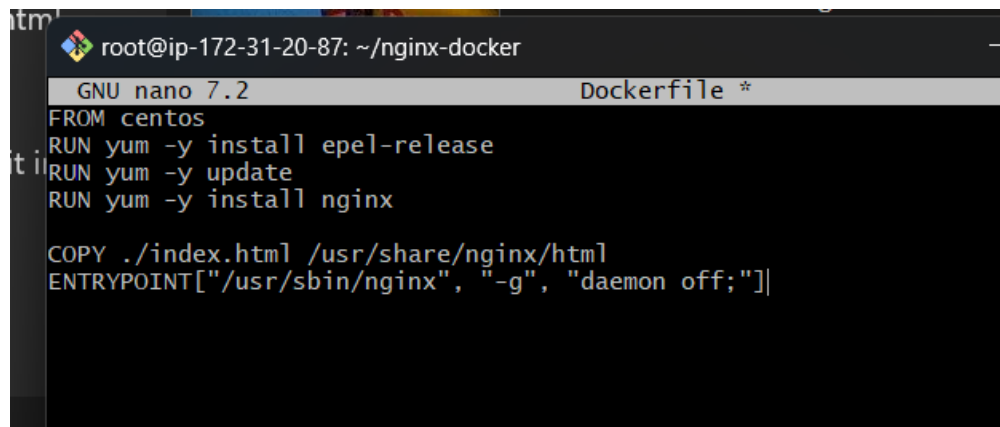
Metrics

Scheduled tasks

Tags

```
root@ip-172-31-20-87:~/nginx-docker# sudo docker run --name  
cont1 -d -p 80:80 nginx  
86377d09c0bf52a5edacaa0a5003db137c5377f40056f671b31b7ecb8  
73ff475
```

Great! You have successfully started the NGINX container named `cont1`. To ensure that your custom `index.html` file is served by the NGINX container, you'll need to mount the file into the container. Let's verify the container is running and then serve the custom content.

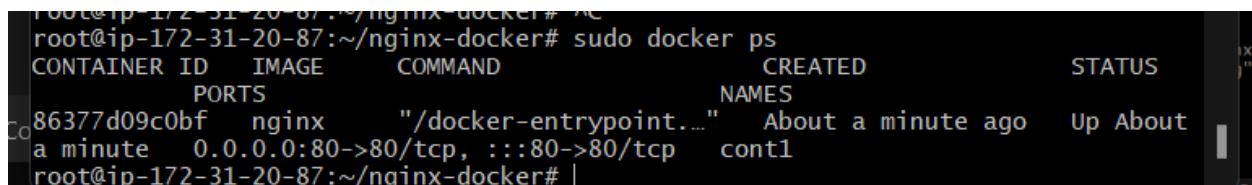


```
root@ip-172-31-20-87: ~/nginx-docker  
GNU nano 7.2 Dockerfile *  
FROM centos  
RUN yum -y install epel-release  
RUN yum -y update  
RUN yum -y install nginx  
  
COPY ./index.html /usr/share/nginx/html  
ENTRYPOINT[ "/usr/sbin/nginx", "-g", "daemon off;"]
```

- 1> specifies base image for Docker image- CentOS is popular Linux distribution
- 2 and 3> update software repo, RUN cmd execs cmds in the container
- Installs extra packages for enterprise linux repo

THIS IS MAKING YOU BUILD THE DOCKER IMAGE

USE VI OVER NANO, lesser commands



```
root@ip-172-31-20-87:~/nginx-docker# sudo docker ps  
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS  
86377d09c0bf   nginx    "/docker-entrypoint...." About a minute ago Up About  
a minute      0.0.0.0:80->80/tcp, :::80->80/tcp cont1  
root@ip-172-31-20-87:~/nginx-docker#
```

```

root@ip-172-31-20-87:~/nginx-docker# docker tag nginx:latest 975050162722.dkr.ecr.us-east-1.amazonaws.com/ecsdemo.intern:latest
root@ip-172-31-20-87:~/nginx-docker# docker push 975050162722.dkr.ecr.us-east-1.amazonaws.com/ecsdemo.intern:latest
The push refers to repository [975050162722.dkr.ecr.us-east-1.amazonaws.com/ecsdemo.intern]
10655d686986: Pushed
8dd5fd695861: Pushed
eddb6eb0845b: Pushed
8162731f1e8d: Pushed
cddaf363c4d4: Pushed
409a3bc90254: Pushed
1387079e86ad: Pushed
latest: digest: sha256:80550935209dd7f6b2d7e8401b9365837e3edd4b047f5a1a7d393e9f04d34498 size: 1778
root@ip-172-31-20-87:~/nginx-docker#

```

Docker tag cmd for creating a new tag for existing image

Nginx latest- source image that you want to tag, nginx is the image name and latest is the tag  
97505etx is the target name and tag for the image, latest will be the tag for the image in the ECR repo

THE NGINX IMAGE HAS BEEN BUILT AND IS NOW UPLOADED ON ECR

The screenshot shows a web browser window with the URL `1216.175`. The page displays a "Welcome to nginx!" message, indicating that the nginx web server is successfully installed and working. Below the welcome message, there is a note about further configuration and links to online documentation and support at [nginx.org](https://nginx.org) and [nginx.com](https://nginx.com). A "Thank you for using nginx." message is also present.

Overlaid on the bottom right of the browser window is a terminal window titled `root@ip-172-31-20-87: ~/nginx-docker`. The terminal shows the output of the command `sudo docker logs cont1`. The logs indicate that the container's entrypoint script is running, performing various configuration steps such as setting up shell scripts, enabling IPv6 listening, and sourcing environment files. The logs conclude with a message stating "Configuration complete; ready for start up" and show the start of the nginx service, including the version (1.27.0) and the start of worker processes.

GETTING ECS TO CALL THE IMAGE FROM ECR

Amazon ECR > Private registry > Repositories > ecsdemo.intern

ecsdemo.intern

View push commandsEdit

Images (1)

Search artifacts

< 1 > ⚙

<input type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input type="checkbox"/>	latest	Image	June 14, 2024, 14:57:13 (UTC+05.5)	70.99	Copy URI	sha256:80550935209dd7...

Choose one or more policies to attach to your new role.

amazonecstas

Filter by Type

All types

1 match

< 1 > ⚙

<input checked="" type="checkbox"/>	Policy name	Type	Description
<input checked="" type="checkbox"/>	AmazonECSTaskEx...	AWS managed	Provides access to other AWS service resources that are required to run Amazon ECS tasks

▶ Set permissions boundary - optional

Created a role  
Attached that task role to the cluster I made on ECS

ECR\_calls\_ECS\_try

Info

Allows ECS tasks to call AWS services on your behalf.

Summary

Creation date

June 14, 2024, 15:10 (UTC+05:30)

Last activity

-

ARN

arn:aws:iam::975050162722:role/ECR\_calls\_ECS\_try

Maximum session duration

1 hour

Permissions

Trust relationships

Tags

Access Advisor

Revoke sessions

Permissions policies (1)

Info

Refresh

Simulate

Remove

A

You can attach up to 10 managed policies.

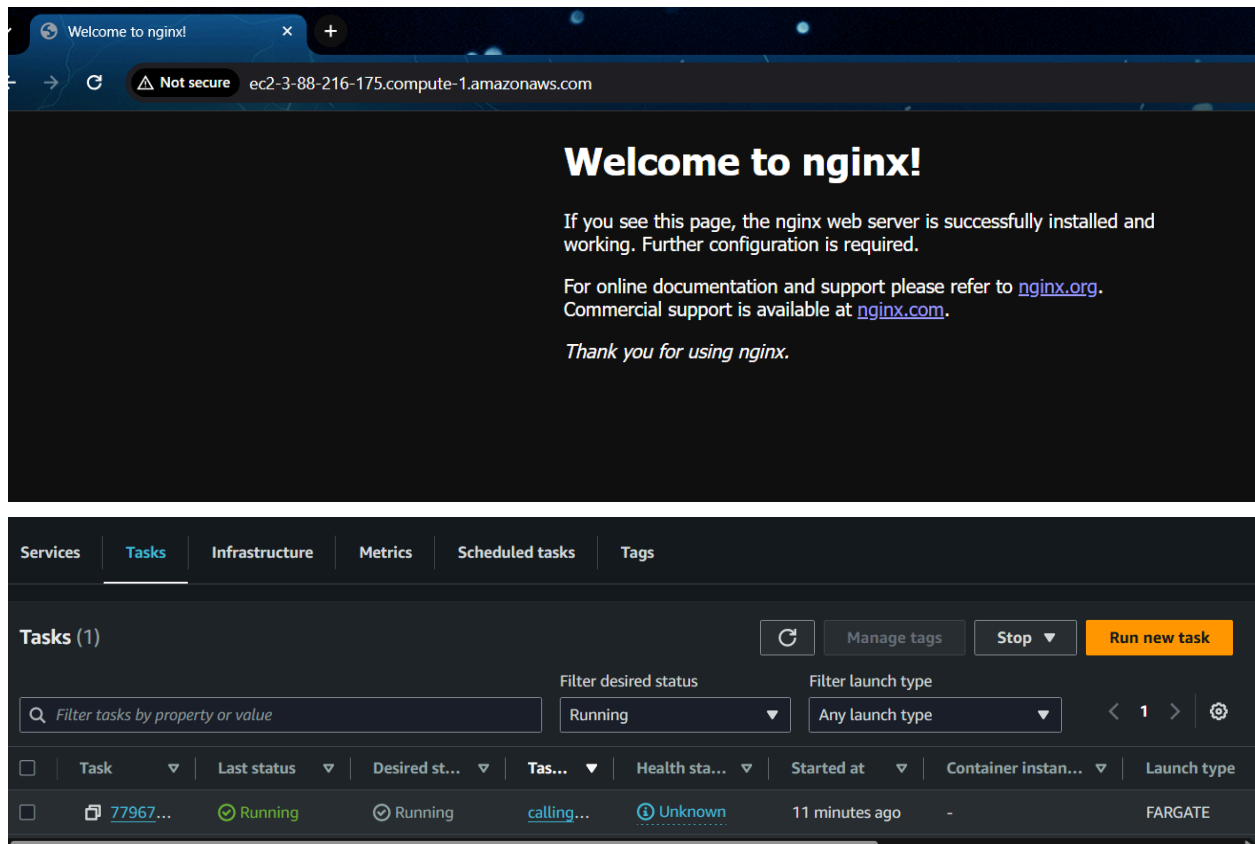
Search

Filter by Type

All types

<input type="checkbox"/>	Policy name	Type	Attached entities
<input type="checkbox"/>	AmazonECS_FullAccess	AWS managed	1





If I understand this correctly, using only ecr, you can call your nginx image from a public ip address but after having ecs call ecr to fetch the image pushed in one of its repositories can we use the public ip DNS?

ECS will handle the deployment and management of containers on EC2 instances within your cluster

When you specify a task definition in ECS that references an NGINX image in ECR, ECS will automatically pull the image from ECR and run it on the EC2 instances in your ECS cluster.

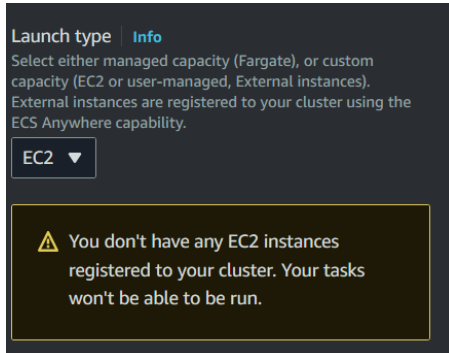
ECS seamlessly integrates with ECR, allowing you to specify ECR repository URLs directly in your ECS task definitions. This eliminates the need for manual steps to pull images onto your EC2 instances before running containers.

## LAUNCHING EC2 CLUSTERS INSTEAD OF FARGATE CLUSTERS

Registering ec2 instance to ECS cluster

Attach the IAM role to the EC2 instance

```
sudo mkdir -p /etc/ecs
```



```
e952d1229dfe0746c55b7cda55d25d07af51c17007f6076cd7c1b27dab20592a
ubuntu@ip-172-31-16-142:~$ sudo docker ps | grep ecs-agent
e952d1229dfe  amazon/amazon-ecs-agent:latest  "/agent"  32 seconds ago  Rest
arting (1) 5 seconds ago          ecs-agent
```