## **AWS ECS With Fargate**

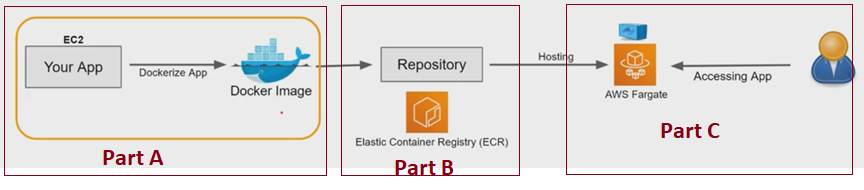
Source : Youtube: <https://youtu.be/JzsSjcyN3MI>

AWS Docs: <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html>

### Objective:

Timeline

Description automatically generated with medium confidence



### Steps: Part A – Create App and Dockerize App to create Docker Image

1. **Create EC2 Instance & Install Docker** 
   1. Launch an instance with the Amazon Linux AMI. (Mention Role To Connect with ECR - AmazonEC2ContainerRegistryFullAccess)
   2. Update the installed packages
   3. Install the most recent Docker Engine package
   4. Start the Docker service.
   5. Add the ec2-user to the docker group so you can execute Docker commands without using sudo.
   6. Log out and log back in again to pick up the new docker group permissions.
   7. Verify that the ec2-user can run Docker commands without sudo.

**sudo yum update -y**

**sudo amazon-linux-extras install docker**

**sudo service docker start**

**sudo usermod -a -G docker ec2-user**

**docker info**

1. **Create a Docker image of a simple Hello World Web Application**
   1. Create a file called Dockerfile - Describing base image to use for your Docker image and what you want installed and running on it
   2. Edit the Dockerfile you just created and add the orange content
   3. Build the Docker image from your Dockerfile.
   4. Run docker images to verify that the image was created correctly.
   5. Run the newly built image

* **touch Dockerfile**
* **nano Dockerfile**
* **docker build -t hello-world .**
* **docker images --filter reference=hello-world**
* **docker run -t -i -p 80:80 hello-world**

FROM ubuntu:18.04

*# Install dependencies*

RUN apt-get update && \

apt-get -y install apache2

*# Install apache and write hello world message*

RUN echo 'Hello World!' > /var/www/html/index.html

*# Configure apache*

RUN echo '. /etc/apache2/envvars' > /root/run\_apache.sh && \

echo 'mkdir -p /var/run/apache2' >> /root/run\_apache.sh && \

echo 'mkdir -p /var/lock/apache2' >> /root/run\_apache.sh && \

echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run\_apache.sh && \

chmod 755 /root/run\_apache.sh

EXPOSE 80

CMD /root/run\_apache.sh

### Steps: Part B: Create ECR & Push Docker Image in ECR

1. **Create ECR and Push this Image to ECR**
   1. Create a ECR Either thru CLI or Console
   2. Tag the hello-world docker image with the ECR URI value from the previous step
   3. Run the aws ecr get-login-password command with pipe ((You may be asked to setup your region using AWS Configure)
   4. Push the image to Amazon ECR with the ECR URI value from the earlier step.

* **aws ecr create-repository --repository-name** *hello-repository* **--region** *region*
* **docker tag hello-world <ECR URI>**
* **aws ecr get-login-password | docker login --username AWS --password-stdin 791309171132.dkr.ecr.ap-south-1.amazonaws.com/amecswithfargate-ecr-repository**
* **docker push <ECR URI>**

### Steps: Part C: Host Docker Image as Docker Container in **ECS Fargate**

1. **Create ECS and Host this Docker Image in Fargate**
   1. **Create ECS Cluster**
   2. **Create Task Definition (Choose Fargate instead of EC2)**
      1. **Add one Container In it – Here specify earlier created Docker Container Image Name and URI (Can be found under ECR)**
   3. **Run Task**
   4. **Then Browse Public API, you’ll find the output of Web Application, which we dockerized in Step #2.**

#### **Create ECS Cluster**

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

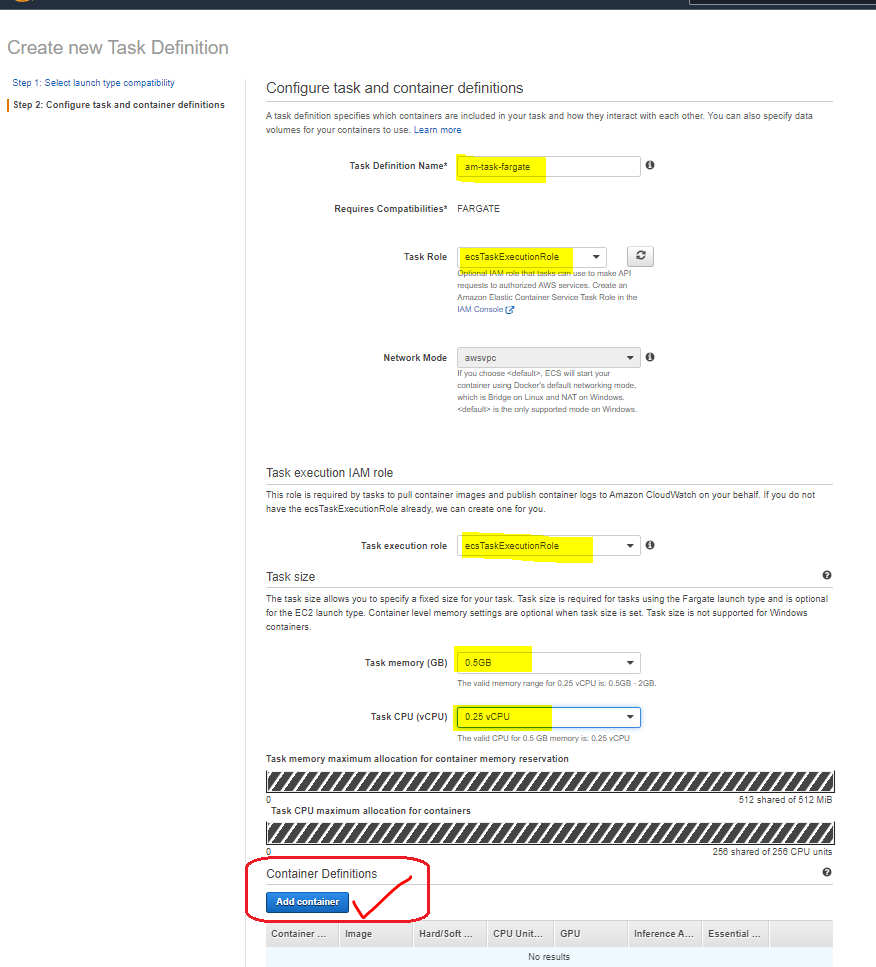
#### **Create Task Definition (Choosing Fargate or EC2)**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, Word

Description automatically generated



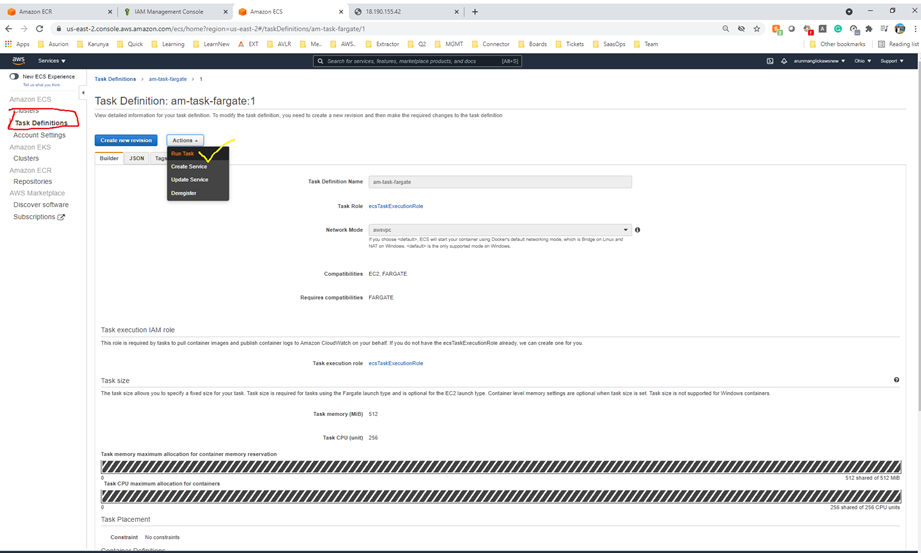
Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

#### **Run Task**



Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

#### **Browse Public API**

Graphical user interface, application, Word

Description automatically generated