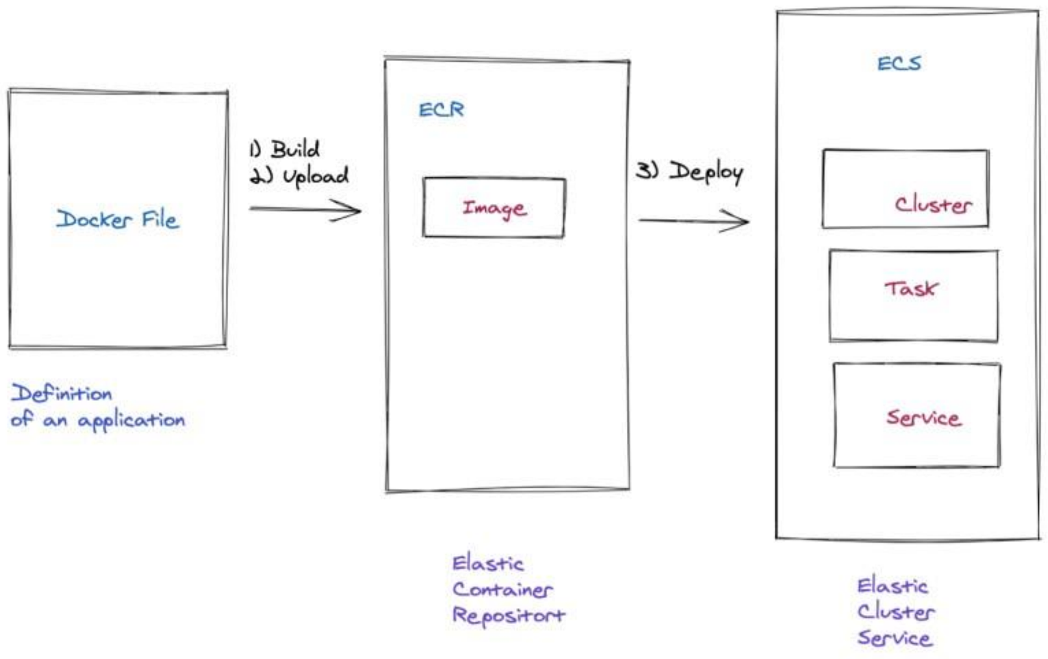
Project Deploying Docker containers on ECS

Name: Niranjan Chavan
Email: Niranjan_Chavan@epam.com

Deploying Docker containers on ECS



Creating Docker Image on ec2 instance:

```
sudo  yum update
aws configure list
mkdir project
sudo yum install docker
sudo systemctl status docker
sudo systemctl enable docker
sudo systemctl start docker
```

sudo vi package.json

```
{
  "name": "docker_web_app",
  "version": "1.0.0",
  "description": "Node.js on Docker",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

sudo vi server.js

```
"use strict"

const express = require("express")

// Constants const
PORT = 8080 const
HOST = "0.0.0.0"

// App const app =
express() app.get("/",
(req, res) => {
  res.send(`Hello World - ${new Date().toISOString()}`)
})

app.listen(PORT, HOST)
console.log(`Running on http://${HOST}:${PORT}`)
```

sudo vi Dockerfile

```
FROM node:14
# Create app directory
WORKDIR /usr/src/app
COPY package*.json ./

RUN npm install
COPY . .

EXPOSE 8080

CMD [ "node", "server.js" ]
```

```
sudo docker build -t node-web-app .
```

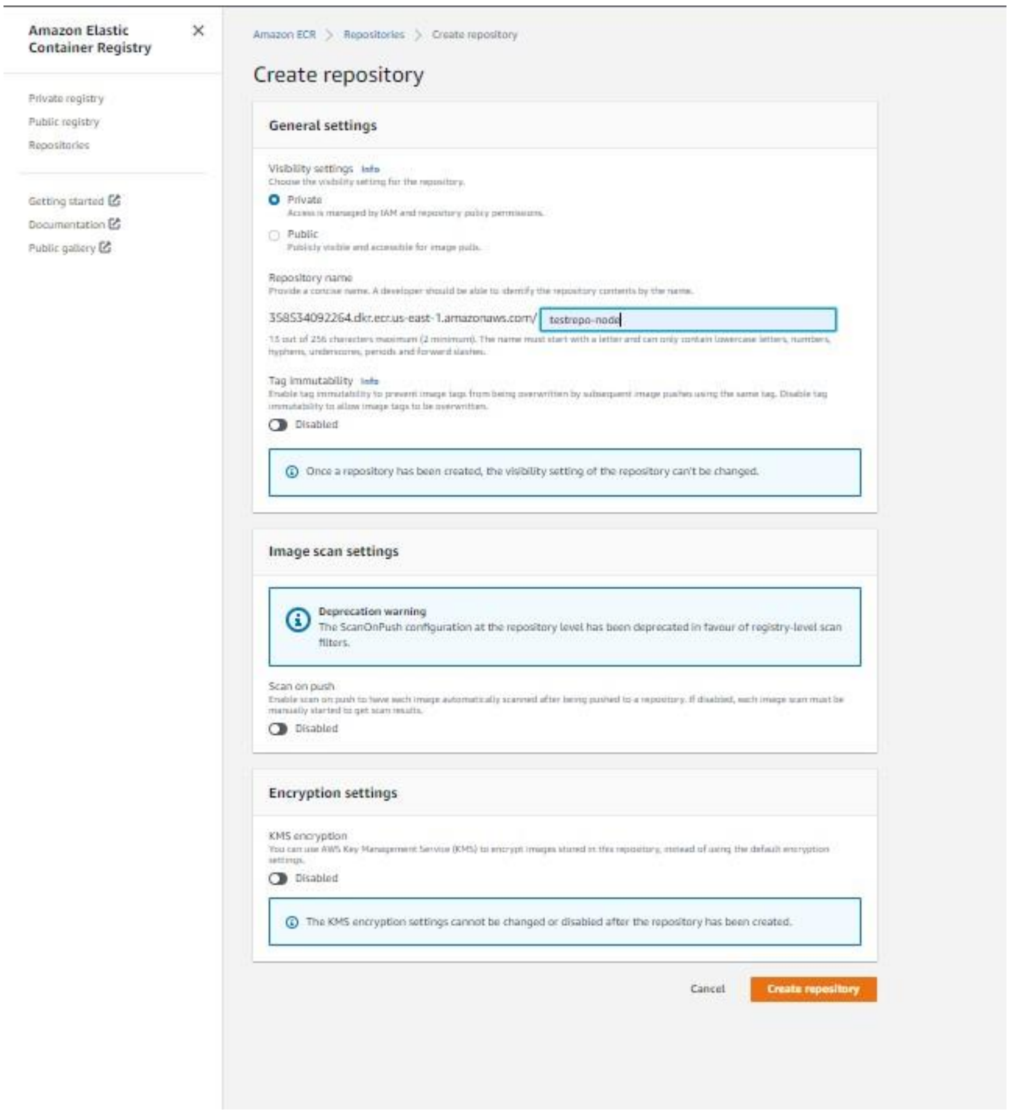
```
sudo docker run -p 80:8080 -d node-web-app curl
```

http://localhost:80

Result:

```
Hello World - 2021-02-11T05:06:12.739Z
```

Create your Amazon ECR in the AWS Console:



Create repository

General settings

Visibility settings Info
Choose the visibility setting for the repository.
[✓ Private] Access is managed by IAM and repository policy permissions.
Public Publicly visible and accessible for image pulls.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.
358534092264.dkr.ecr.us-east-1.amazonaws.com/ testrepo-node
13 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Tag immutability Info
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.
Disabled

Once a repository has been created, the visibility setting of the repository can't be changed.

Image scan settings

Deprecation warning
The ScanOnPush configuration at the repository level has been deprecated in favour of registry-level scan filters.

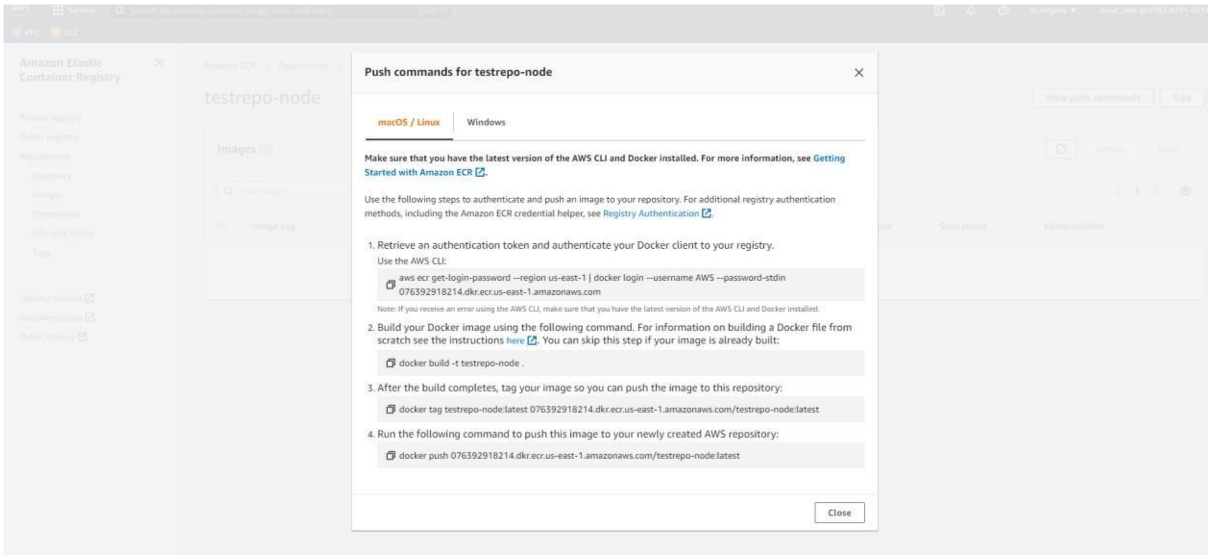
Scan on push
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.
Disabled

Encryption settings

KMS encryption
You can use AWS Key Management Service (KMS) to encrypt images stored in this repository, instead of using the default encryption settings.
Disabled

The KMS encryption settings cannot be changed or disabled after the repository has been created.

Viewing pushing Command



Push commands for testrepo-node

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see Getting Started with Amazon ECR.

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see Registry Authentication.

1. Retrieve an authentication token and authenticate your Docker client to your registry.
   Use the AWS CLI:
   ```
   aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin
   076392918214.dkr.ecr.us-east-1.amazonaws.com
   ```
   Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.
2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions here. You can skip this step if your image is already built:
   ```
   docker build -t testrepo-node .
   ```
3. After the build completes, tag your image so you can push the image to this repository:
   ```
   docker tag testrepo-node:latest 076392918214.dkr.ecr.us-east-1.amazonaws.com/testrepo-node:latest
   ```
4. Run the following command to push this image to your newly created AWS repository:
   ```
   docker push 076392918214.dkr.ecr.us-east-1.amazonaws.com/testrepo-node:latest
   ```

Uploading image using these command.

Copy the image URI: we need to keep this to create a task definition for the following steps.

Create an ECS Cluster

Go to the ECS home page and click on the create cluster button:

Choose EC2 Linux + Networking and then click next:

Then enter the following information:

- name of the cluster: ecs01
- EC2 instance type: t3-micro
- Number of instances: 1

sThen choose:

- Default VPC
- Auto assign IP: Enabled
- Security group: default
- Choose one of the subnet

Create a new Task definition

Click on new Task definition

Choose EC2

Then next

Choose NodeWebAppTask for the name of the task definition.

Enter 128for memory size.

Click Add Container:

Add the name of the container: NodeWebApp

Set the image URI that we have saved to add the end of the add image step

Set the port mappings 80:8080

Click create.

Create Application load Balancer attach it to service and check the DNS:

http://my-alb-1205436972.us-east-1.elb.amazonaws.com/

It is working 😊