How to dockerize your Node.js Express application to AWS ECR

In this article, we will learn how to build a user input application using Node Express, Docker, and AWS services. Indeed, we will first create a Node Express application in our local environment, then a Docker file to create a Docker image and a container, and finally an AWS ECR to pull our image. We will test our container by launching an EC2 instance in AWS.

Check out the application we are going to build via this link

http://ec2-3-227-16-138.compute-1.amazonaws.com:3000/

What you will learn:

* Basic node express application
* Docker
* Some AWS services

Prerequisites:

AWS account

Express: A backend web-application framework for Node.js.

Node.js: Runtime for building web applications.

Let's get started:

I Node Express application

1 – Workspace installation:

Create a new folder and navigate into it

You can do this from the command line using:

run node-ikapp

Text

Description automatically generated with medium confidence

Run node –version =🡺 to verify node version:

Text

Description automatically generated

If you don’t have it let install via <https://nodejs.org/en/download/>;

Install node express: npm install express

We installed node js and node Express, let build our Node Express application.

2 Building Node express sample application:

Open a CLI in your project directory and run the following commands:

Touch app.js

Touch index.html

Run code. to open the project in visual studio

As you could see, the project doesn’t contains a package json yet . Let run npm init to initialize our project. Keep pushing the enter keyboard until the process finish

Text

Description automatically generated

Wonderful, let build our application.

Open app.js. and add this:

Text

Description automatically generated

These few lines of code are:

We first imported Express (first line of code), create a simple route handler for the HTTP GET/request to index our HTML file, and finally start a server listening for incoming request on port and to display a message: “Hey readers our application is running on port 5000”

Open index.html, add this:

This is a simple HTML form. The form action will be redirected to the route /docker-demo-user

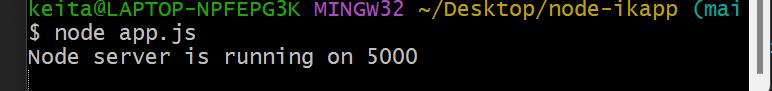
Text

Description automatically generated

Text

Description automatically generated

Run node app.js to check out our application :



A picture containing graphical user interface

Description automatically generated

After user submit the query, the URL will be redirected to /docker-demo-user

Graphical user interface, text, application, email

Description automatically generated Our application is up running perfectly on our local environment (Laptop). At this stage, it is not shareable. How to make this application shareable with countless pull request?

Answering this question will drive us into our main topic: Docker

II Docker:

Docker is a tool designed to make it easier for developers to develop, ship, and run applications by using containers. Containers allow devs to package an application with all its requirements and configurations, such as libraries and other dependencies and deploy it as a single package. By doing so, developers can rest easy knowing that the software will work properly on other systems besides the one they used for writing the code.

Let’s go back to our project and within the same directory to create a Dokerfile

Graphical user interface, text, application

Description automatically generated

Dockerfile is basically a text file. It contains some set of instructions. Automation of docker image creation.

A screenshot of a computer

Description automatically generated with medium confidence

From = For a base image, it must be on top of the docker file

WORDIR= Work directory

ADD package\*.json = Files haven’t changed , so it uses the same image layer built the first without re-running anything .

Run npm install will install npm

CMD will run node app.js

As you could notice, we got our application running locally, let create an image from it and make it available anywhere.

There are several dockers registries provider such as: docker Hub, quay.io and Amazon ECR. We are going to use AWS ECR to create a repository and from that repository will create an image for our application.

III AWS services

1. ECR

If you are not admin, you should give to yourself full control over these services: ECR, EC2,VPC.

In AWS management console go ahead to create a public repository.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

We create a repository let build an image and push it into this repository. Open the project folder with gitbash and run the following commands:

* Dowload AWS CLI
* Run this command: *aws configure*
* It will prompt you to enter the following:
* AWS Access Key ID [None]: your key
* AWS Secret Access Key [None]:your key
* Default region name [None]: us-west-2
* Default output format [None]: json

Now you can login with AWS CLI and do the following:

##### a) **Retrieve an authentication token and authenticate your Docker client to your registry**

Text

Description automatically generated

##### 2) **Build your Docker image using the following command**

docker build -t ikeita .

Run this command to check out your image: docker image ls

A picture containing chart

Description automatically generated

Perfect!!, we got our image created. Because we didn’t customize tag, the default tag will attach on it: latest.

##### 3) **After the build completes, tag your image so you can push the image to this repository:**

docker tag ikeita:latest public.ecr.aws/t4r2x1j8/ikeita:latest

##### 4) **Run the following command to push this image to your newly created AWS repository:**

docker push public.ecr.aws/t4r2x1j8/ikeita:latest

If check your repository, you will see the image we pushed.

Graphical user interface

Description automatically generated with medium confidence

For now, we got our docker image in AWS ECR. We are going to launch an instance, install docker on it and containerize our image.

2 VPC :

Create a default VPC

Graphical user interface, text, application

Description automatically generated

3 Launch an ec2 instance

Use a default vpc, default security group (allow port 22 and port 80)

Graphical user interface, text, application

Description automatically generated

Select the instance we just launched, click on connect, EC2 instance Connect and connect

Graphical user interface, text, application, email

Description automatically generated

We got an Amazon linux running

A screenshot of a computer

Description automatically generated with medium confidence

Run the following commands:

Sudo su: to get root privilege

Yum update -y: to update the machine

After updating the machine, install Docker on our machine:

Yum install docker

Docker version

Text

Description automatically generated

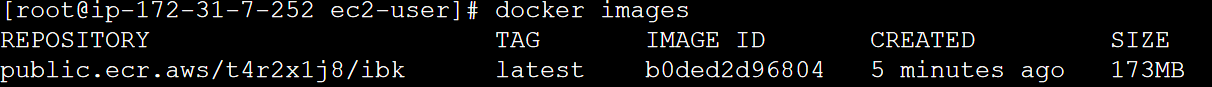
Let pull our image from ECR repository

To pull it, we first must start Docker.

Run this: systemctl start docker

Run this command to pull image: docker pull public.ecr.aws/t4r2x1j8/ibk :latest which is your repository URL + your image tag.

Run docker images to list images:



IV Docker Containers:

A Docker container is an open-source software development platform. Its main benefit is to package applications in containers allowing them to be portable to any system running a Linux or Windows operating system (OS)

Run this: docker run -t -p 3000:5000 public.ecr.aws/t4r2x1j8/ibk:latest

Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

A picture containing text

Description automatically generated

<http://ec2-3-227-16-138.compute-1.amazonaws.com:3000/>

Text

Description automatically generated with medium confidence