

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

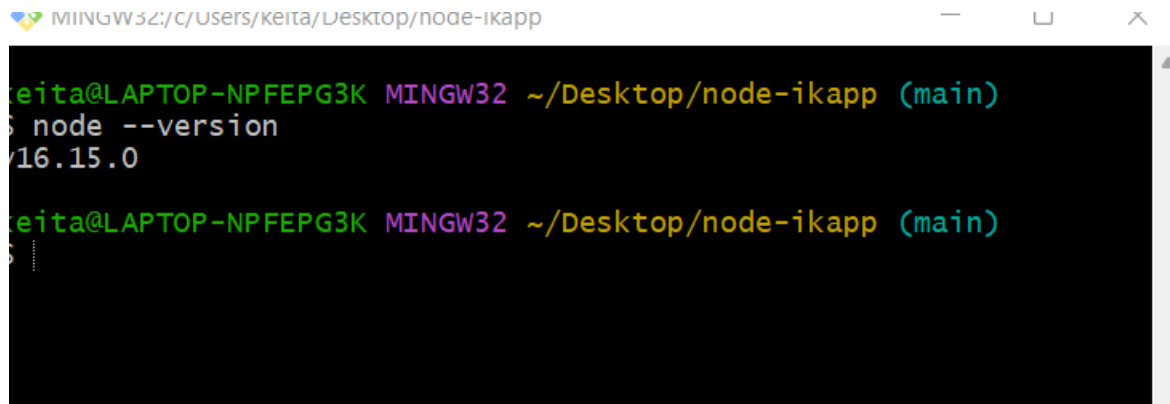
You can do this from the command line using:

run node-ikapp

```
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop (main)
$ mkdir node-ikapp

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop (main)
$ |
```

Run `node --version` => to verify node version:

A screenshot of a terminal window with a black background and green text. The window title is 'MINGW32: C:/Users/keita/Desktop/node-ikapp'. The prompt is 'keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/node-ikapp (main)'. The command 'node --version' has been entered, and the output 'v16.15.0' is displayed. The prompt is shown again on the next line.

```
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/node-ikapp (main)
$ node --version
v16.15.0
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/node-ikapp (main)
$
```

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:

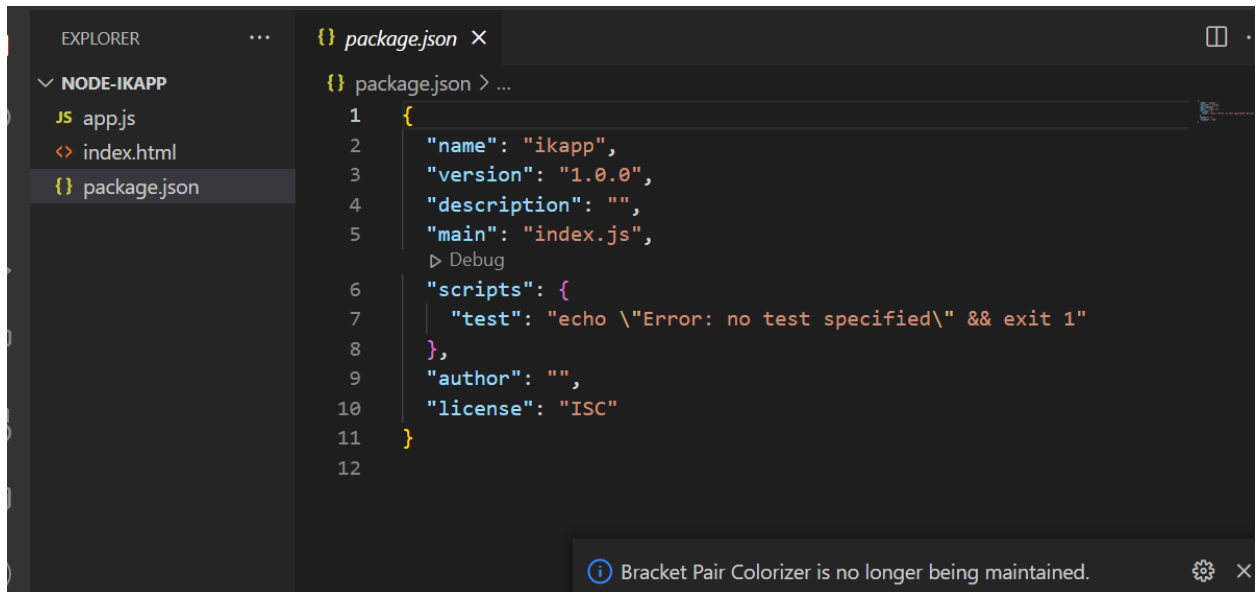
Open a CLI in your project directory and run the following commands:

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



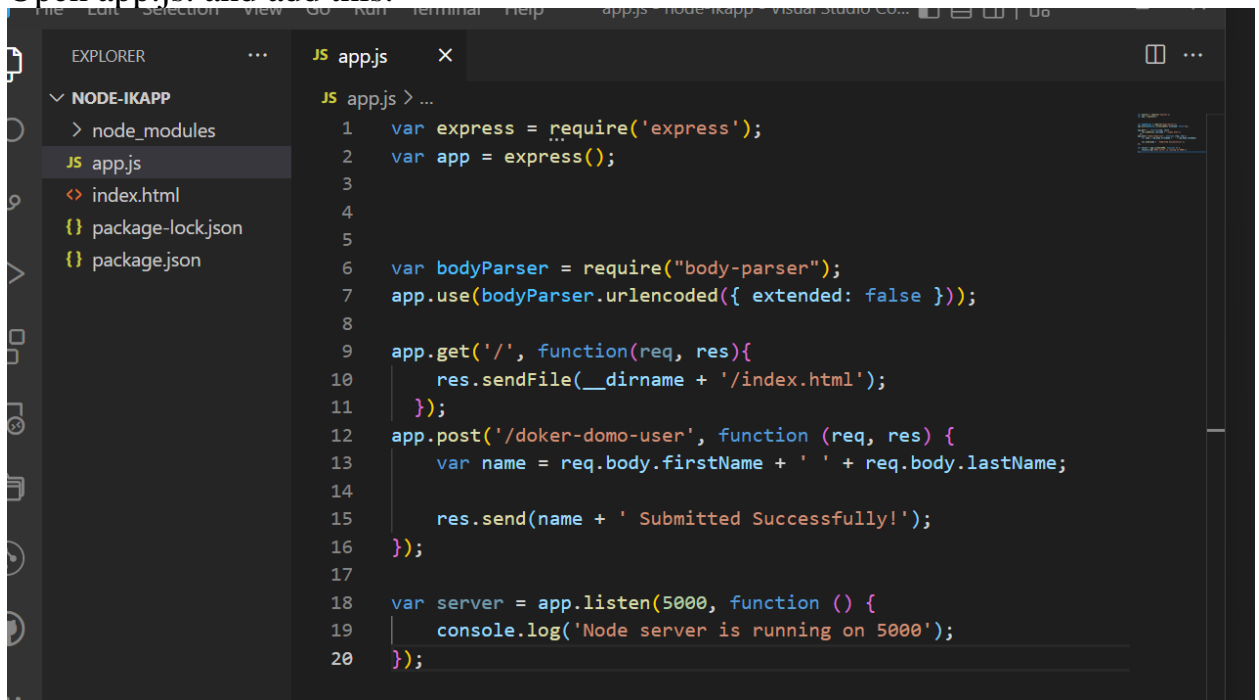
The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying the file structure of a project named 'NODE-IKAPP'. The files listed are 'app.js', 'index.html', and 'package.json'. The 'package.json' file is selected and its content is displayed in the main editor. The code is as follows:

```
1 {  
2   "name": "ikapp",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "author": "",  
10  "license": "ISC"  
11 }  
12
```

A notification at the bottom right states: "Bracket Pair Colorizer is no longer being maintained."

Wonderful, let build our application.

Open app.js. and add this:



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying the file structure of a project named 'NODE-IKAPP'. The files listed are 'node_modules', 'app.js', 'index.html', 'package-lock.json', and 'package.json'. The 'app.js' file is selected and its content is displayed in the main editor. The code is as follows:

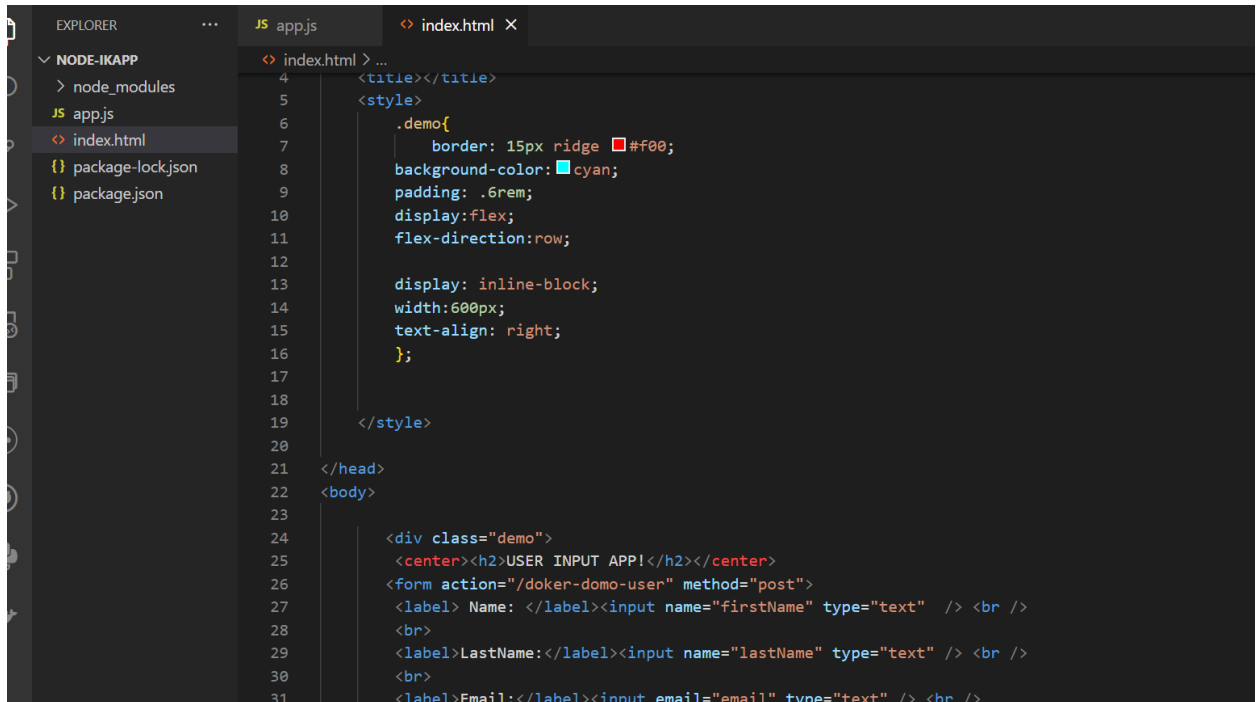
```
1 var express = require('express');  
2 var app = express();  
3  
4  
5  
6 var bodyParser = require("body-parser");  
7 app.use(bodyParser.urlencoded({ extended: false }));  
8  
9 app.get('/', function(req, res){  
10   res.sendFile(__dirname + '/index.html');  
11 });  
12 app.post('/doker-domo-user', function (req, res) {  
13   var name = req.body.firstName + ' ' + req.body.lastName;  
14  
15   res.send(name + ' Submitted Successfully!');  
16 });  
17  
18 var server = app.listen(5000, function () {  
19   console.log('Node server is running on 5000');  
20 });
```

These few lines of code are:

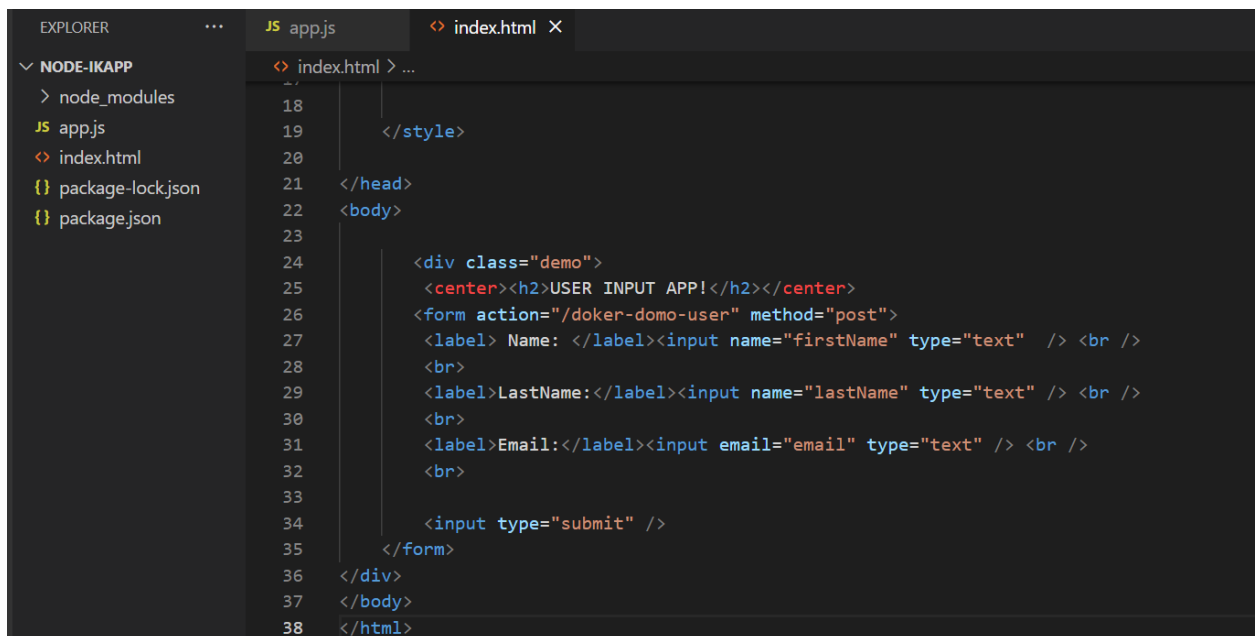
We first imported Express (first line of code), create a simple route handler for the HTTP GET/request to index our HTML file, and finally start a server listening for incoming request on port and to display a message: “Hey readers our application is running on port 5000”

Open index.html, add this:

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

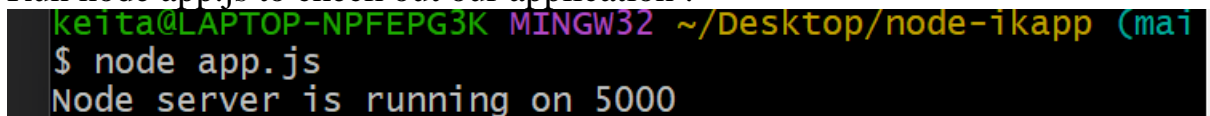


```
index.html > ...
4 <title></title>
5 <style>
6   .demo{
7     border: 15px ridge #f00;
8     background-color: cyan;
9     padding: .6rem;
10    display: flex;
11    flex-direction: row;
12
13    display: inline-block;
14    width: 600px;
15    text-align: right;
16  };
17
18 </style>
19
20 </head>
21 <body>
22
23
24 <div class="demo">
25   <center><h2>USER INPUT APP!</h2></center>
26   <form action="/docker-demo-user" method="post">
27     <label> Name: </label><input name="firstName" type="text" /> <br />
28     <br>
29     <label>LastName:</label><input name="lastName" type="text" /> <br />
30     <br>
31     <label>Email:</label><input email="email" type="text" /> <br />
32   </form>
33 </div>
34 </body>
35 </html>
```

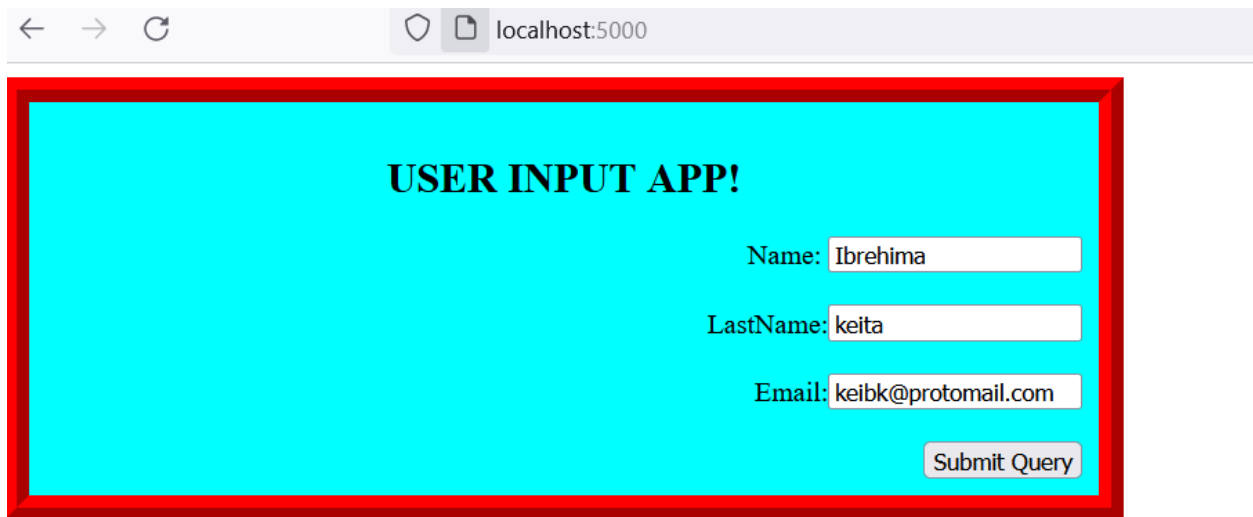


```
index.html > ...
18
19 </style>
20
21 </head>
22 <body>
23
24 <div class="demo">
25   <center><h2>USER INPUT APP!</h2></center>
26   <form action="/docker-demo-user" method="post">
27     <label> Name: </label><input name="firstName" type="text" /> <br />
28     <br>
29     <label>LastName:</label><input name="lastName" type="text" /> <br />
30     <br>
31     <label>Email:</label><input email="email" type="text" /> <br />
32     <br>
33     <input type="submit" />
34   </form>
35 </div>
36 </body>
37 </html>
```

Run node app.js to check out our application :



```
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/node-ikapp (mai
$ node app.js
Node server is running on 5000
```



USER INPUT APP!

Name:

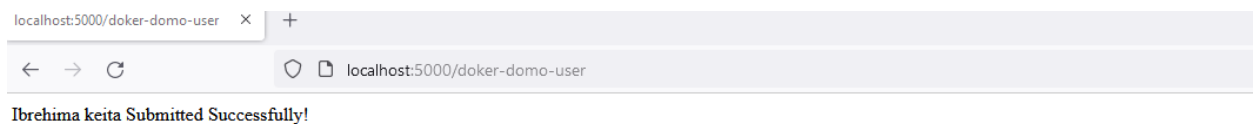
LastName:

Email:

Author: Ibrehima

keibk@protonmail.com

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



localhost:5000/docker-demo-user

Ibrehima keita Submitted Successfully!

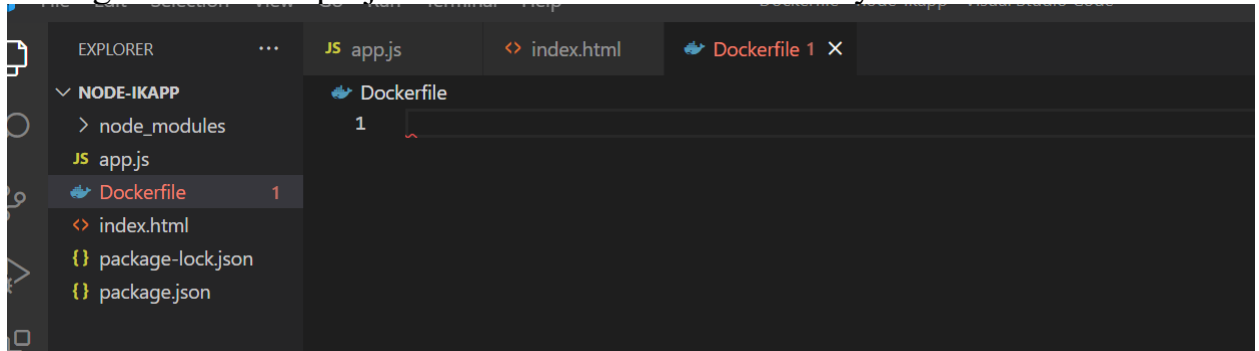
Our application is up running perfectly on our local environment (Laptop). At this stage, it is not shareable. How to make this application shareable with countless pull request?

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

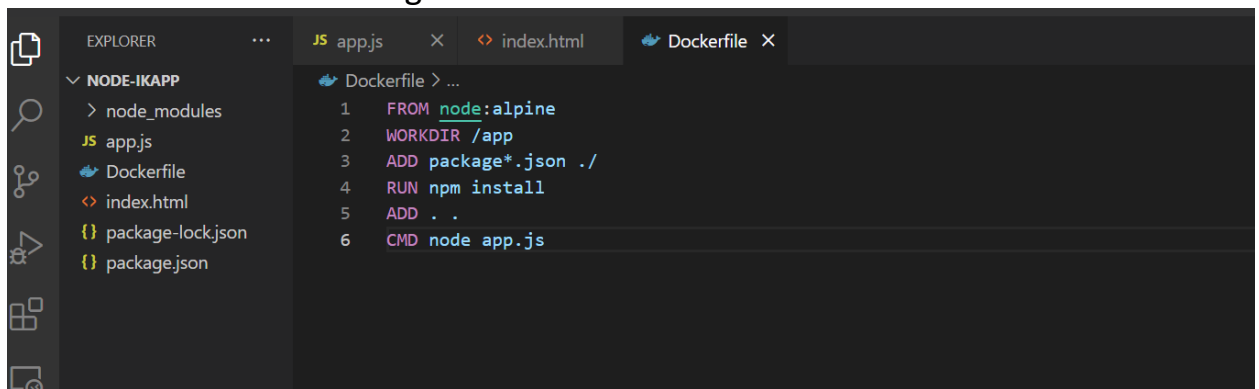
II Docker:

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 Dockerfile



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



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.

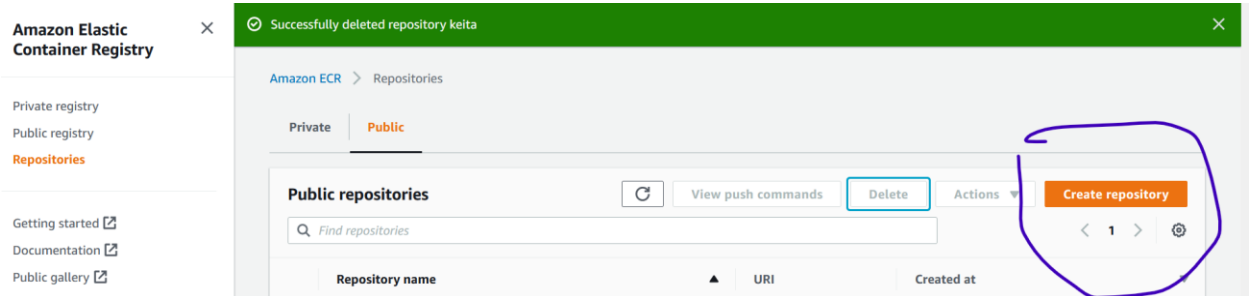
There are several dockers registries provider such as: docker Hub, quay.io and Amazon ECR. We are going to use AWS ECR to create a repository and from that repository will create an image for our application.

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.



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:

- Download 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**

```
MINGW32/c/Users/keita/Desktop/node-ikapp

ita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/node-ikapp (mai
aws ecr-public get-login-password --region us-east-1 | docker login --u
ername AWS --password-stdin public.ecr.aws/t4r2x1j8
gin Succeeded

gging in with your password grants your terminal complete access to you
account.
r better security, log in with a limited-privilege personal access toke
Learn more at https://docs.docker.com/go/access-tokens/

ita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/node-ikapp (main)
```

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

`docker build -t ikeita .`

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

```
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/node-ikapp (main)
$ docker images
REPOSITORY          TAG          IMAGE ID          CREATED
SIZE
ibk                  latest       b0ded2d96804     11 seconds ago
173MB
```

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.

Amazon Elastic Container Registry

Private registry

Public registry

Repositories

Images

Gallery detail

Permissions

Tags

Getting started [↗](#)Documentation [↗](#)Public gallery [↗](#)

Amazon ECR > Repositories > ibk

ibk

[View public listing](#) [↗](#)[View push commands](#)[Edit](#)

Images (1)

[↻](#)[Delete](#)

<

1

>

[⚙](#)

<input checked="" type="checkbox"/>	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input checked="" type="checkbox"/>	latest	Image	July 29, 2022, 19:05:10 (UTC-04)	53.35	Copy URI	sha256:8

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

[New VPC Experience](#)
Tell us what you think

VPC dashboard

EC2 Global View [↗](#) **New**

Filter by VPC:

[Select a VPC](#)Your VPCs (1/1) [Info](#)[↻](#)[Actions](#)[Create VPC](#)

<

1

>

[⚙](#)

<input checked="" type="checkbox"/>	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
<input checked="" type="checkbox"/>	default-vpc	vpc-0c54a30c94bb272f7	Available	172.31.0.0/16	-

3 Launch an ec2 instance

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

EC2 Image Builder

[New EC2 Experience](#)
Tell us what you think

EC2 Dashboard

EC2 Global View

Events

Tags

Limits

Instances (1) [Info](#)[↻](#)[Connect](#)[Instance state](#)[Actions](#)[Launch instances](#)

<

1

>

[⚙](#)

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	docker-demo	i-0c71eebf751249960	Running	t2.micro	Initializing	No alarms

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



Connect to instance [Info](#)

Connect to your instance i-0c71eebf751249960 (docker-demo) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID

i-0c71eebf751249960 (docker-demo)

Public IP address

3.227.16.138

User name

ec2-user

Connect using a custom user name, or use the default user name ec2-user for the AMI used to launch the instance.

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel

Connect

We got an Amazon linux running

EC2 Image Builder

```
  _ |   _ |   )
  _ | ( _ |   /   Amazon Linux 2 AMI
  _ | \ _ |   |
```

```
https://aws.amazon.com/amazon-linux-2/
12 package(s) needed for security, out of 22 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-7-252 ~]$
```

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

```

Dependency Installed:
  containerd.x86_64 0:1.4.13-3.amzn2      libcgrouper.x86_64 0:0.41-21.amzn2      pigz.x86_64 0:2.3.4-1.amzn2
  runc.x86_64 0:1.0.3-3.amzn2

Complete!
[root@ip-172-31-7-252 ec2-user]# docker version
Client:
Version:      20.10.13
API version:  1.41
Go version:   go1.16.15
Git commit:   a224086
Built:        Thu Mar 31 19:20:32 2022
OS/Arch:      linux/amd64
Context:      default
Experimental: true
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
[root@ip-172-31-7-252 ec2-user]#

```

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:

```

[root@ip-172-31-7-252 ec2-user]# docker images
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
public.ecr.aws/t4r2x1j8/ibk  latest      b0ded2d96804     5 minutes ago   173MB

```

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`

```

[root@ip-172-31-7-252 ec2-user]# docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS        PORTS                               NAMES
c547a19a4606   public.ecr.aws/t4r2x1j8/ibk:latest  "docker-entrypoint.s..."  51 minutes ago
Up 51 minutes  0.0.0.0:3000->5000/tcp, :::3000->5000/tcp  peaceful_dewdney
[root@ip-172-31-7-252 ec2-user]#

```

i-0c71eebf751249960 (docker-demo)

PublicIPs: 3.227.16.138 PrivateIPs: 172.31.7.252



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

USER INPUT APP!

Name: LastName: Email:

Author: Ibrehima

keibk@protonmail.com

ec2-3-227-16-138.compute-1.amazonaws.com:3000/doker-domo-user

Ibrehima keita Submitted Successfully!

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

```
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/node-ikapp (main)
$ date
Sat Jul 30 02:28:13 EDT 2022

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/node-ikapp (main)
$
```