Introduction to Cloud Computing Final Project – Guess the Capital

Estimated time needed: 30 minutes

In this final project, you will be deploying "Guess the Captial" on the cloud. It is a web application that asks you to guess the capital of a country from 4 choices.

You will use the source code and the steps provided to practice hands-on how an application can be developed and deployed on the cloud.

Objectives:

1. Clone the source code
2. Build Docker image
3. Deploy on Docker
4. Tag and Push image to IBM Cloud
5. Deploy on IBM Code Engine

Background

Docker

Containers are isolated environments that package applications and their dependencies. Each container runs as an isolated process on the host operating system.

Docker is an open-source platform that enables developers to automate the deployment and management of applications inside lightweight, isolated containers.

IBM Cloud

IBM Cloud is a cloud computing platform and suite of cloud-based services offered by IBM. It provides a range of infrastructure, platform, and software services to support the development, deployment, and management of various types of applications and workloads in the cloud.

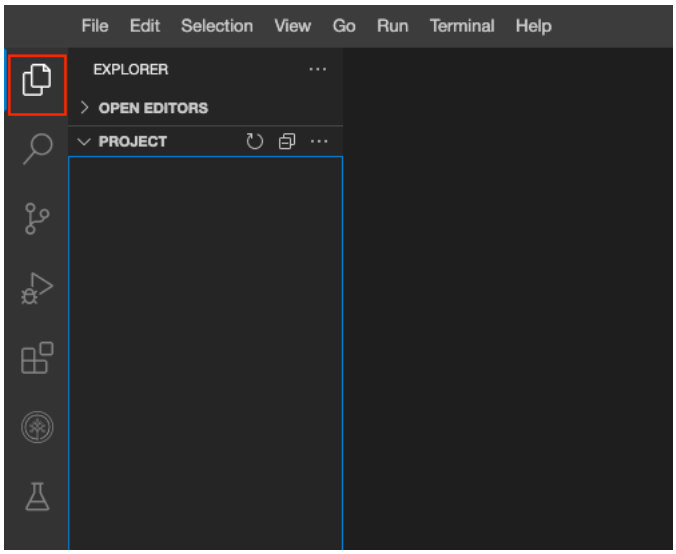
IBM Code Engine

IBM Cloud Code Engine is a serverless compute platform provided by IBM Cloud. It allows developers to deploy and run containerized applications without the need to manage the underlying infrastructure. Abstracting away the complexities of server provisioning, scaling, and maintenance, enabling developers to focus on writing code and building applications.

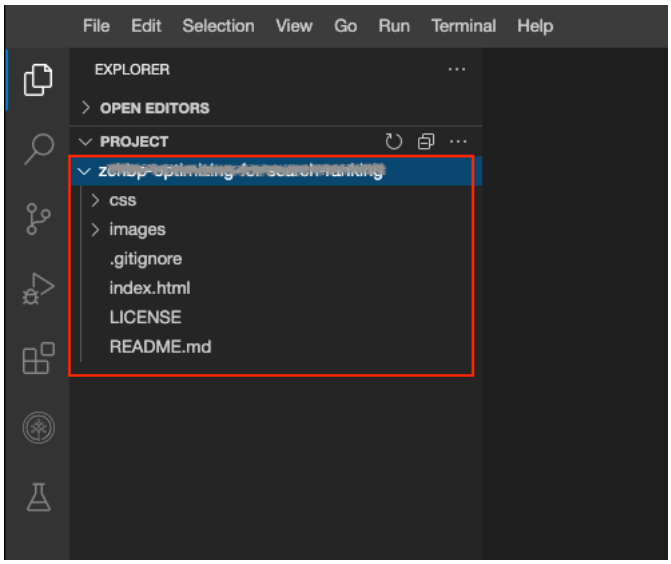
Working with files in Cloud IDE

If you are new to Cloud IDE, this section will show you how to create and edit files, which are part of your project, in Cloud IDE.

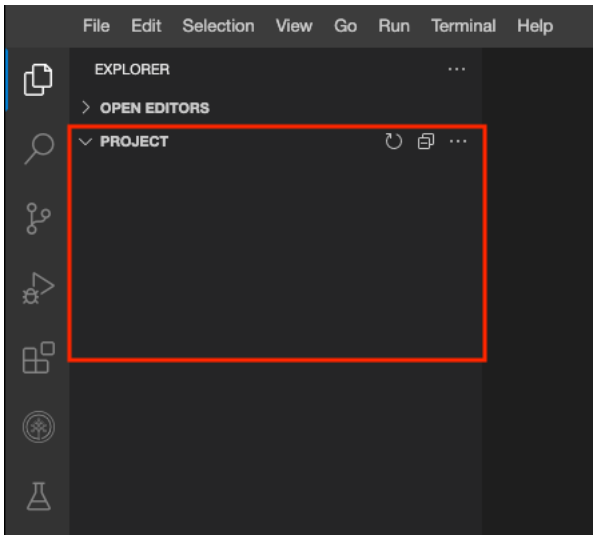
To view your files and directories inside Cloud IDE, click on this files icon to reveal it.



If you have cloned (using `git clone` command) boilerplate/starting code, then it will look like below:

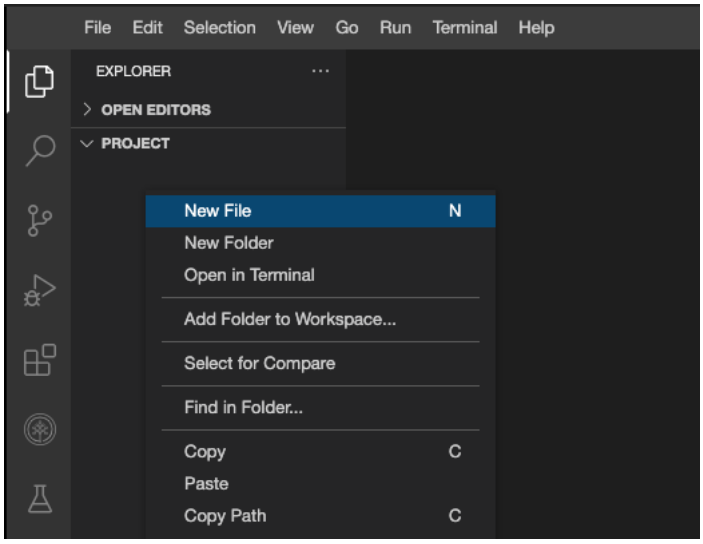


Otherwise a blank project looks like this:



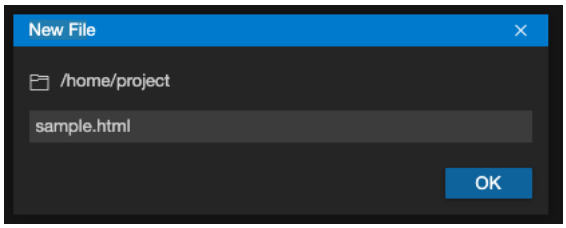
Create a new file

You can right-click and select the New File option to create a file in your project.

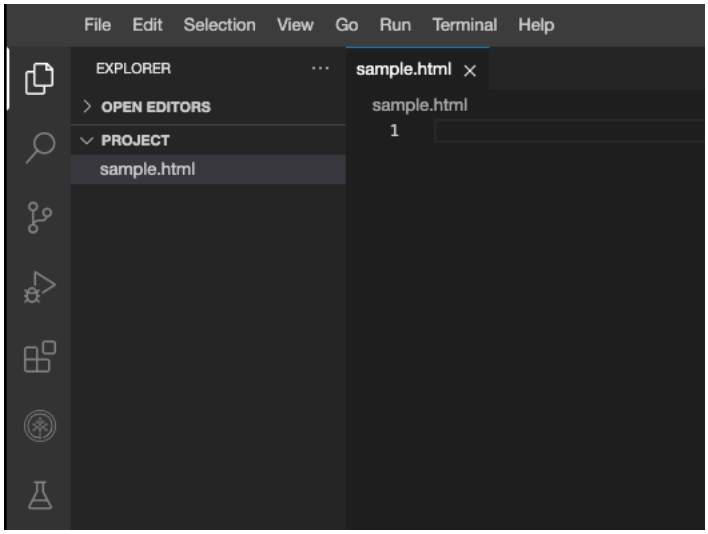


You can also choose `File -> New File` to do the same.

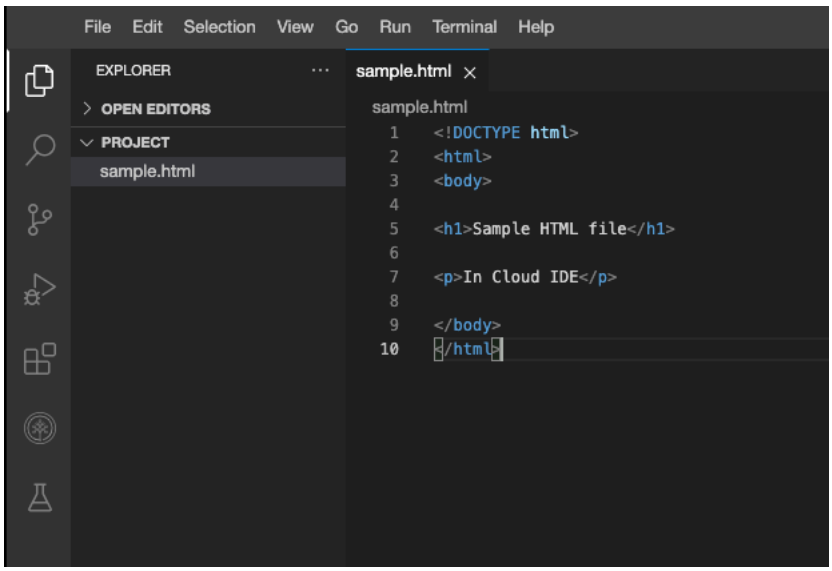
It will then prompt you to enter name of this new file. In the example below, we are creating `sample.html`.



Clicking on the file name `sample.html` in the directory structure will open the file on the right pane. You can create all different types of files; for example `FILE_NAME.js` for JavaScript file.

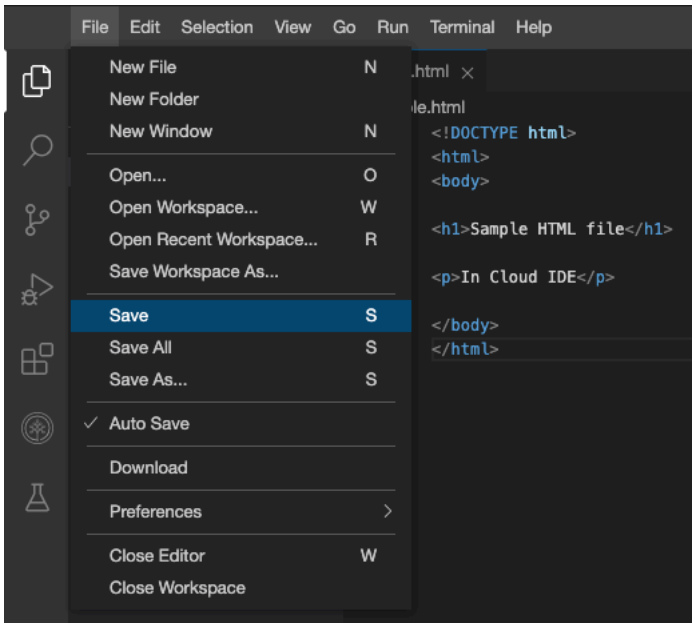


In the example, we just pasted some basic html code and then saved the file.



And saving it by:

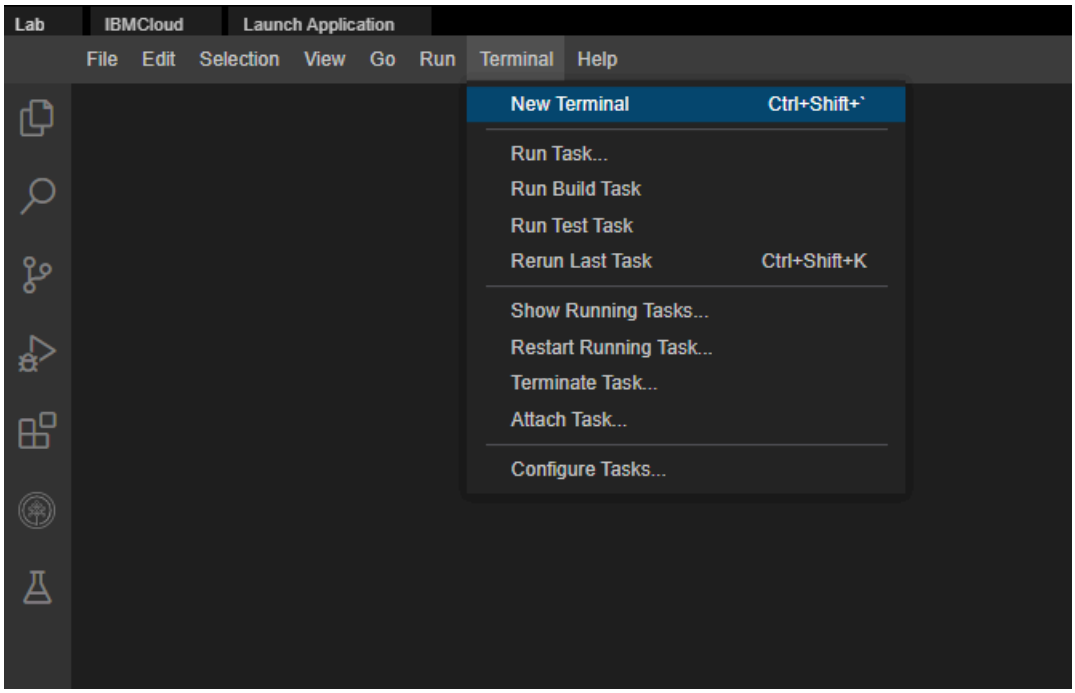
- Going in the menu.
- Press ⌘ + S on Mac or CTRL + S on Windows.
- Or it can Autosave it for you too.



Verify the environment and command line tools

1. Open a terminal window by using the menu in the editor: `Terminal > New Terminal`.

   Note:If the terminal is already opened, please skip this step.



2. Verify that `docker` CLI is installed.

   ```
   docker --version
   ```

You should see the following output, although the version may be different:

```
theia@theiadocker-[redacted]:/home/project$ docker --version
Docker version 20.10.7, build 20.10.7-0ubuntu5~18.04.3
```

3. Verify that `ibmcloud` CLI is installed.

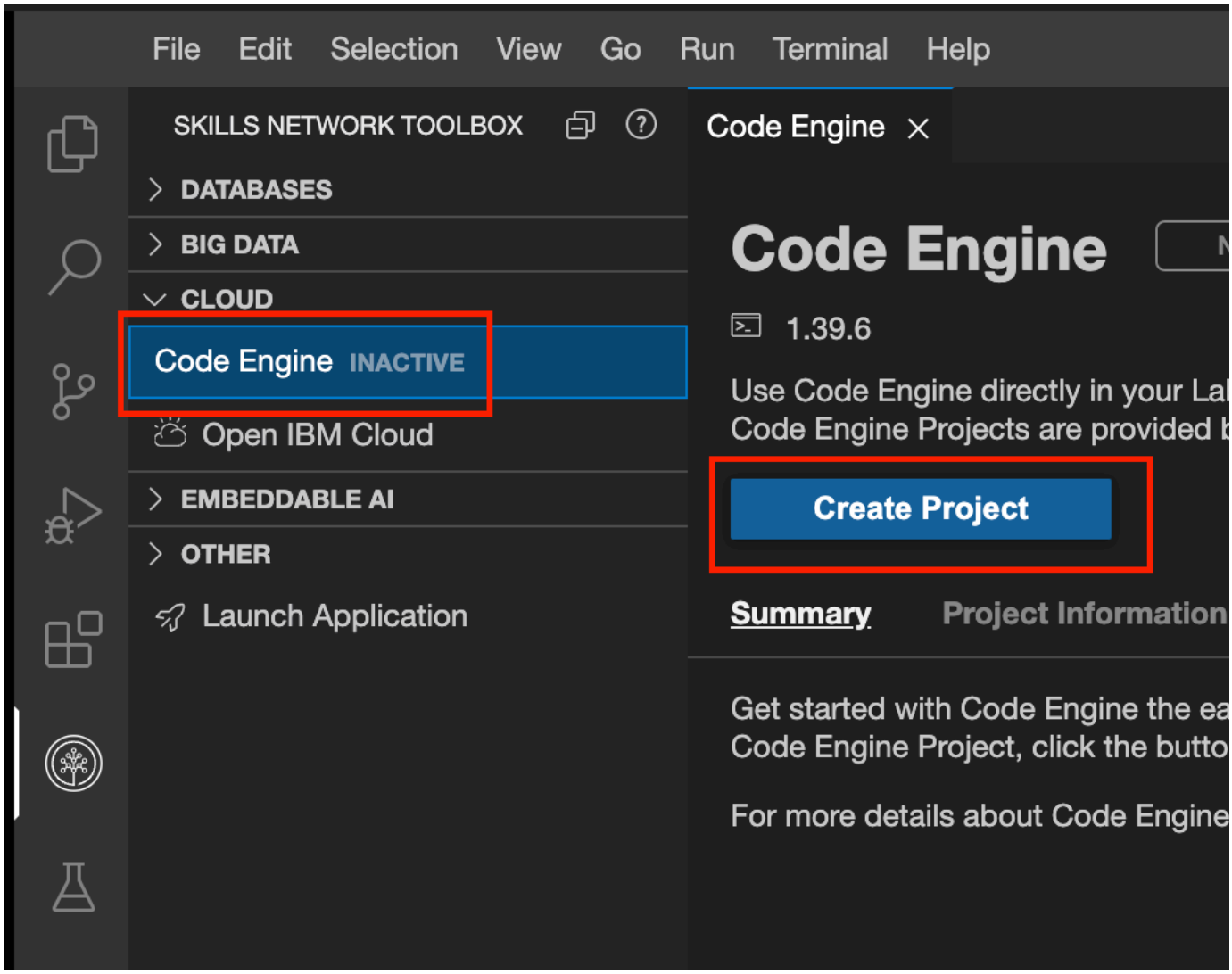
   ```
   ibmcloud version
   ```

You should see the following output, although the version may be different:

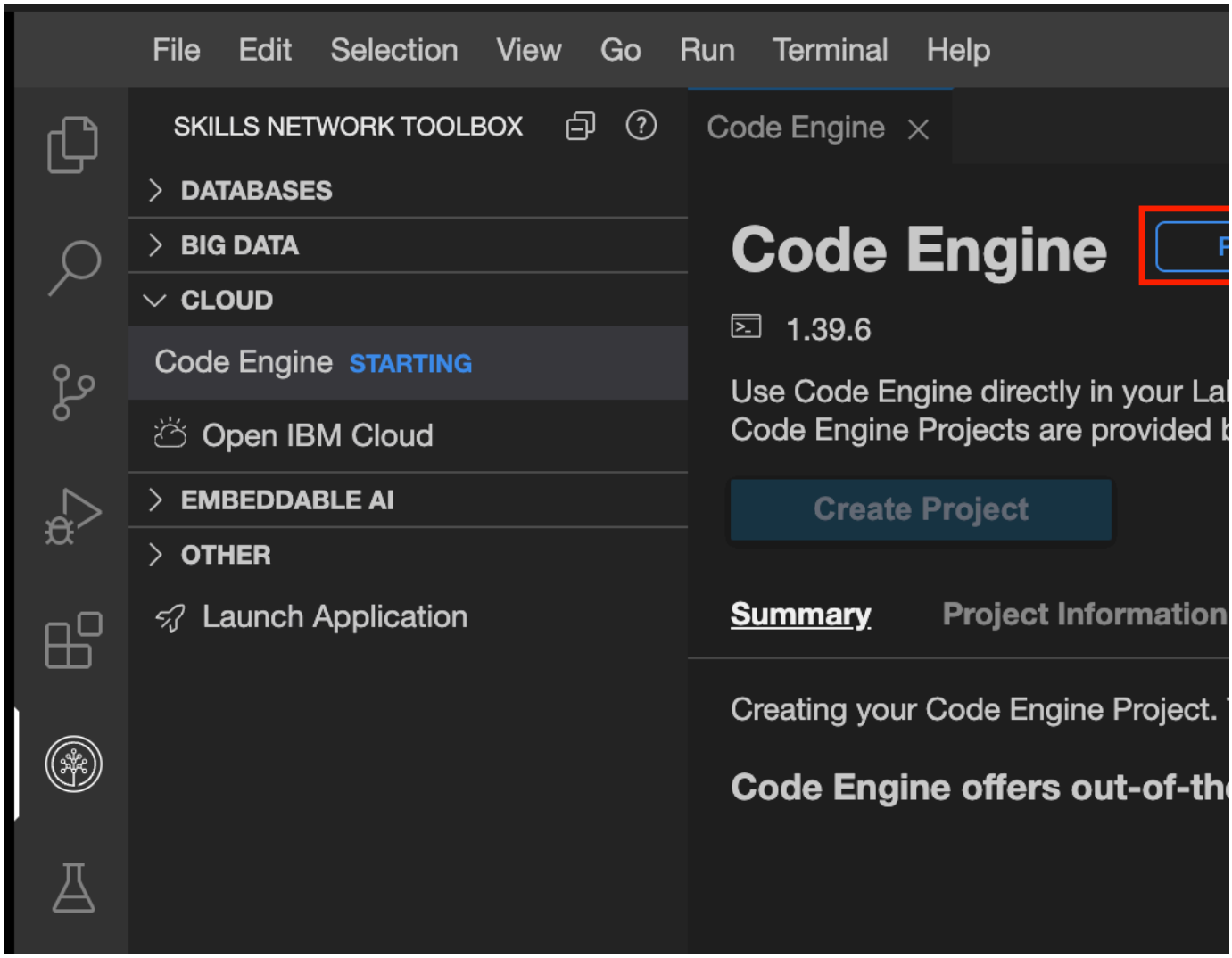
```
theia@theiadocker-[redacted]:/home/project$ ibmcloud version
ibmcloud version 2.1.1+19d7e02-2021-09-24T15:16:38+00:00
```

Start Code Engine

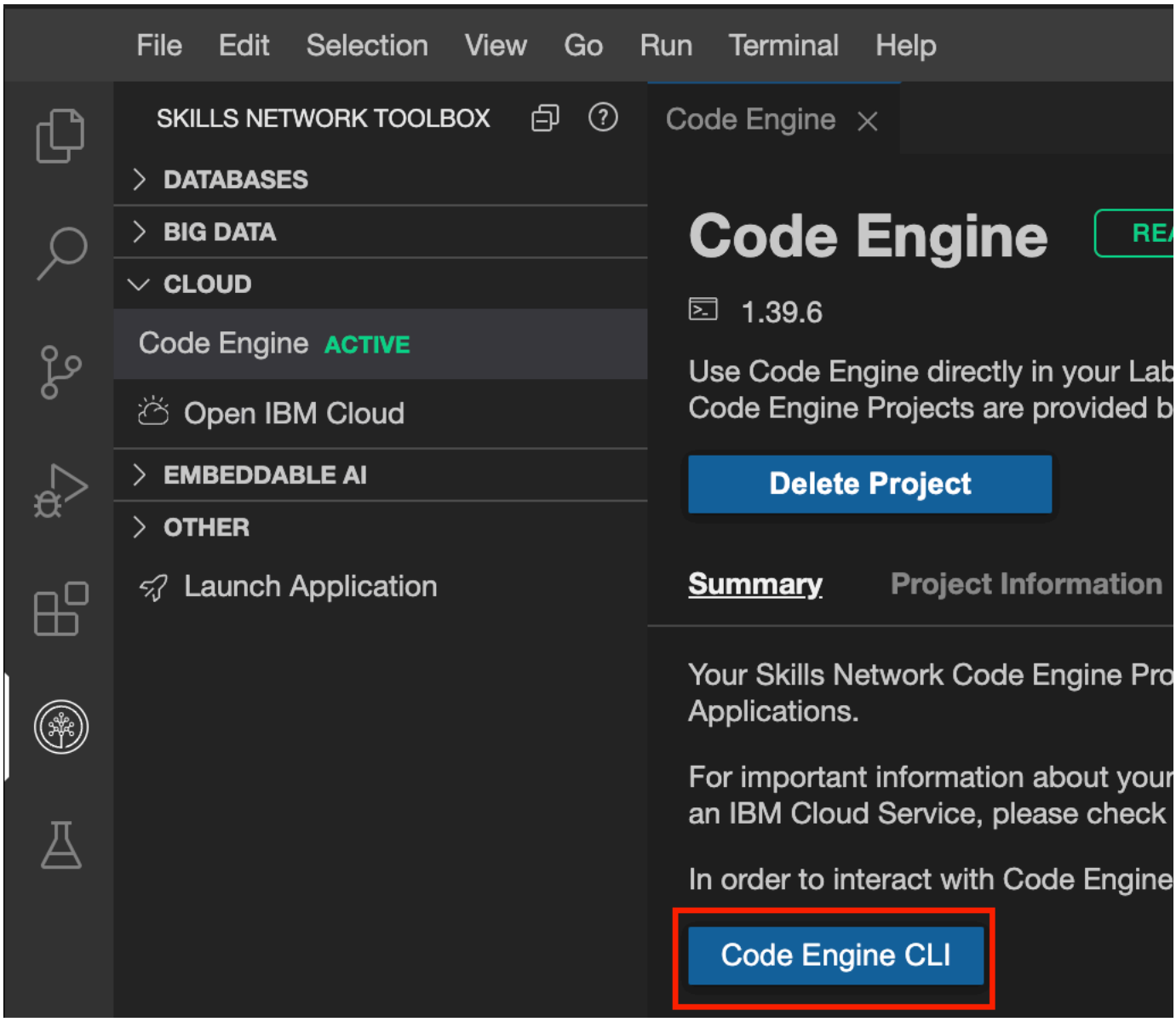
1. On the menu in your lab environment, click the `Cloud` dropdown menu and select `Code Engine`. The code engine setup panel appears. Click `Create Project` to begin.



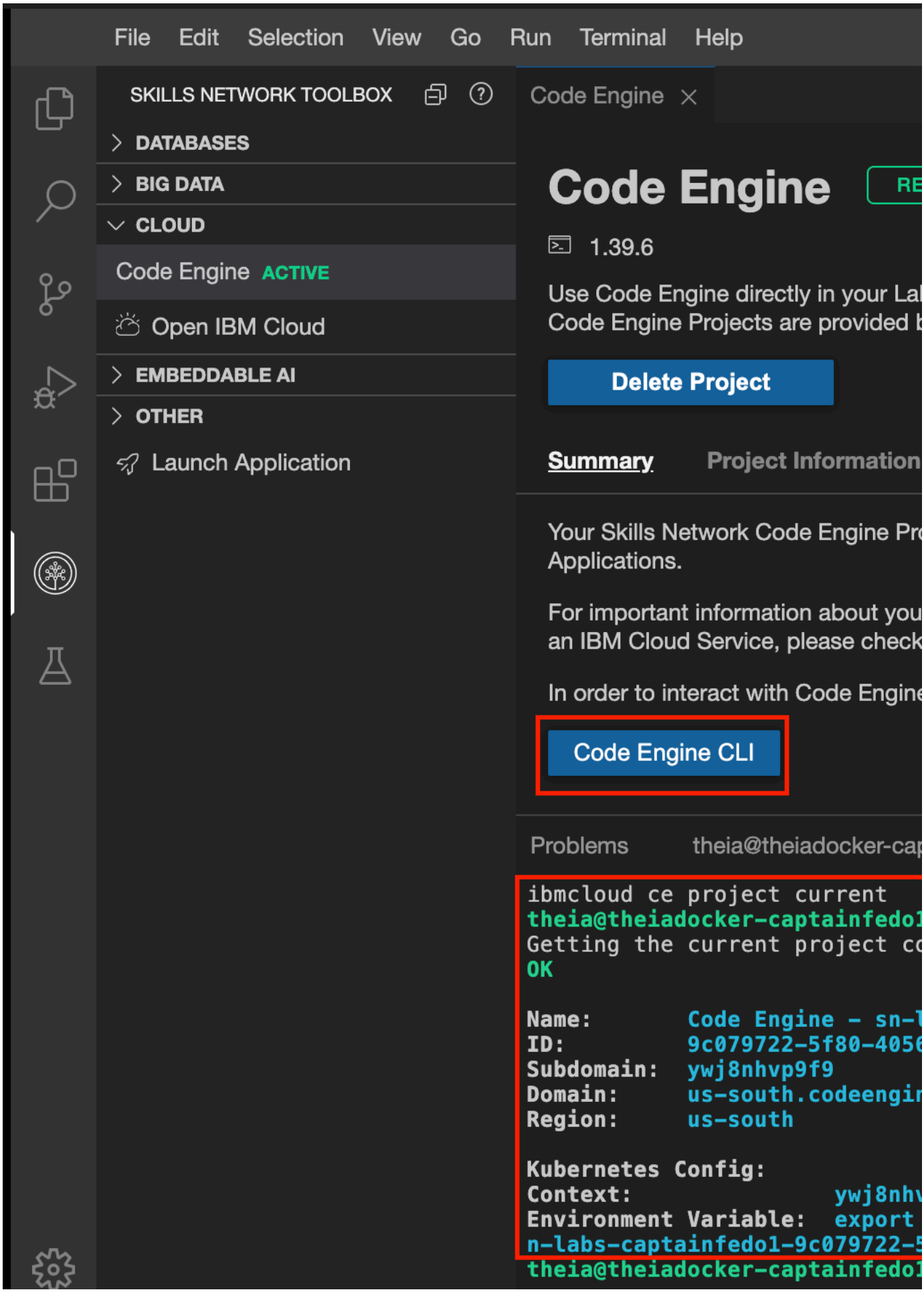
2. The code engine environment takes a while to prepare. You will see the progress status is indicated in the setup panel.



3. Once the code engine set up is complete, you can see that it is active. Click `Code Engine CLI` to begin the pre-configured CLI in the terminal as shown below.

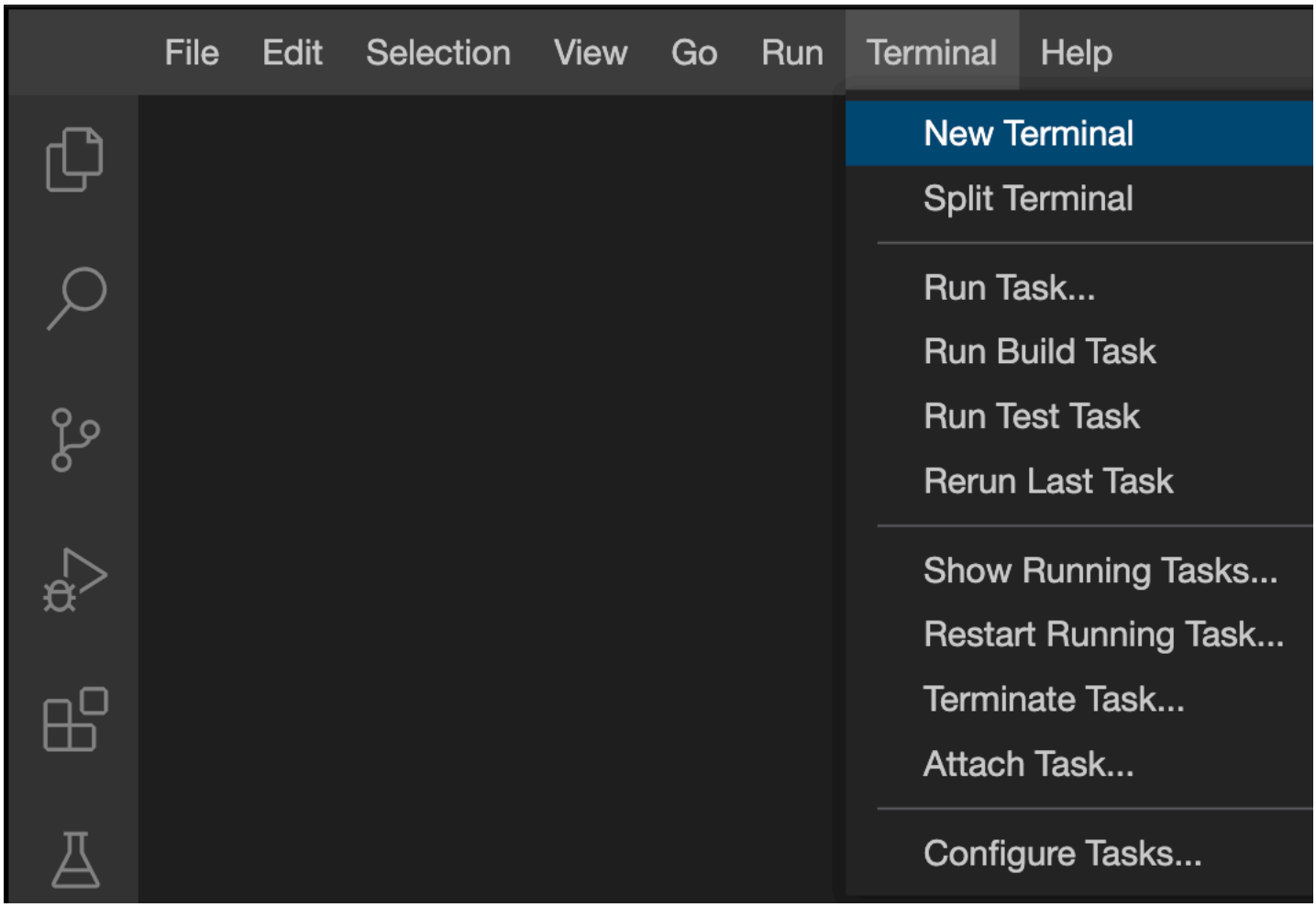


4. You will observe that the pre-configured CLI startup and the home directory are set to the current directory. As a part of the pre-configuration, the project has been set up, and Kubeconfig is set up. The details are shown on the terminal as follows.



Set-up : Create application

1. Open a terminal window by using the menu in the editor: **Terminal > New Terminal**.



2. If you are not currently in the project folder, copy and paste the following code to change to your project folder.

   ```
   cd /home/project
   ```

3. Run the following command to clone the Git repository that contains the starter code needed for this project if the Git repository doesn't already exist.

   ```
   [ ! -d 'fyidw-guess-the-capital' ] && git clone https://github.com/ibm-developer-skills-network/fyidw-guess-the-capital.git
   ```

4. Change to the directory **fyidw-guess-the-capital** to start working on the lab.

   ```
   cd fyidw-guess-the-capital
   ```

5. List the contents of this directory to see the artifacts for this lab.

   ```
   ls
   ```

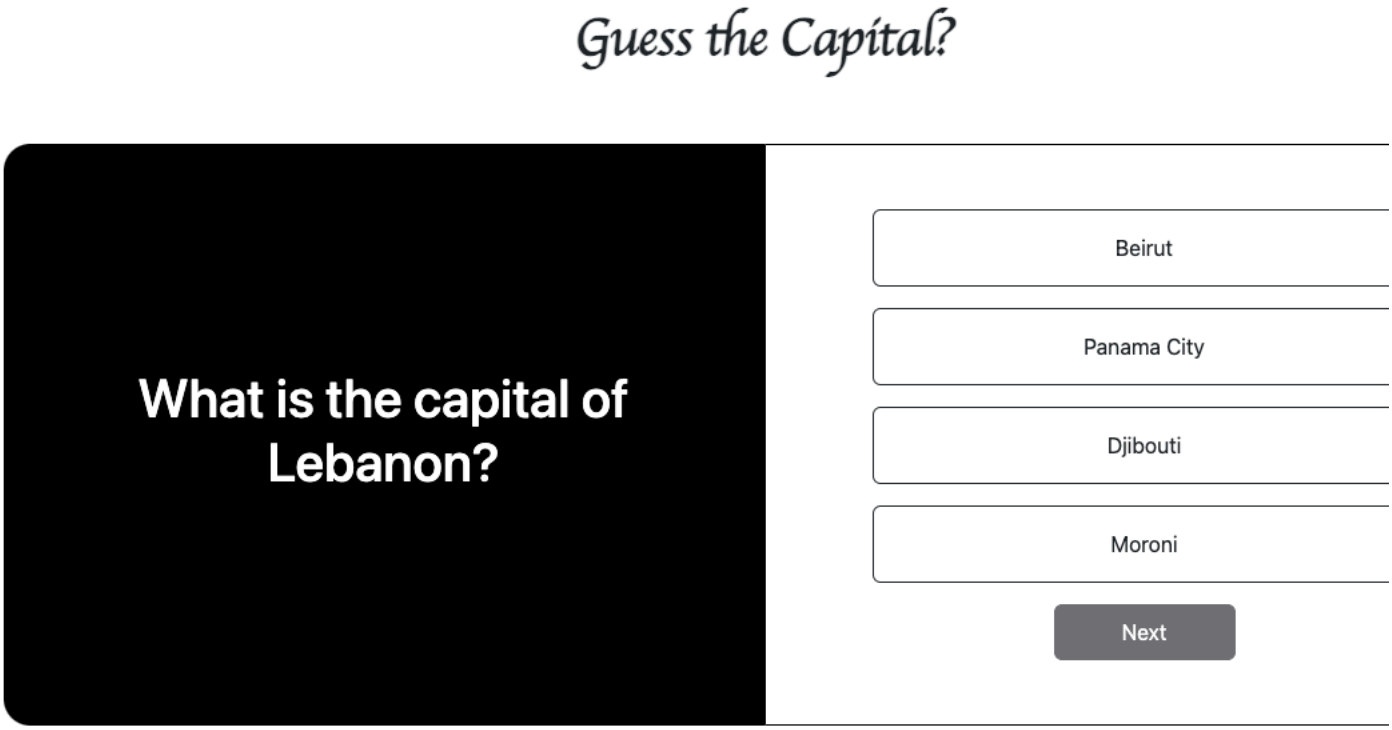
6. Run the following command on the terminal to host your web page.

   ```
   python3 -m http.server
   ```

7. To test your application in your browser, run the application first.

Launch Application

8. It will look like this:



9. In your terminal, press `CTRL + C` to stop your web server.

Task 1: Containerise the application

Let's start modernising our application. The first step towards it is to containerise it using Docker.

Create Dockerfile

Your tasks:

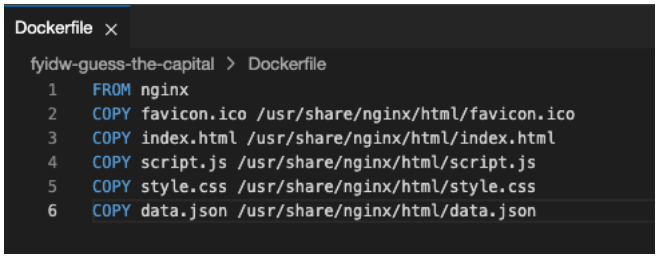
1. Paste the following content in

Open **Dockerfile** in IDE

Use the below as Dockerfile content.

```
FROM nginx
COPY favicon.ico /usr/share/nginx/html/favicon.ico
COPY index.html /usr/share/nginx/html/index.html
COPY script.js /usr/share/nginx/html/script.js
COPY style.css /usr/share/nginx/html/style.css
COPY data.json /usr/share/nginx/html/data.json
```

And it should look like below:



2. Build an image from a Dockerfile

```
docker build -t guess-the-capital .
```

Giving you the output similar to:

```
theia@theiadocker-[redacted]:/home/project/fyidw-guess-the-capital$ dock
[+] Building 12.2s (12/12) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 291B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/nginx:latest
 => [auth] library/nginx:pull token for registry-1.docker.io
 => [1/6] FROM docker.io/library/nginx@sha256:67f9a4f10d147a6e04629340e
 => => resolve docker.io/library/nginx@sha256:67f9a4f10d147a6e04629340e
 => => sha256:262696647b70a57f5f7dbf97a91091e7b51c1d2537dff72a 41.46MB
 => => sha256:67f9a4f10d147a6e04629340e6493c9703300ca23a2f7f3aa5 1.86kB
 => => sha256:73e957703f1266530db0aeac1fd6a3f87c1e59943f4c13eb34 1.78kB
 => => sha256:648e0aadf75ac2ef63c5390adc6dc14fde37a5ad88c2870e 29.12MB
 => => sha256:89da1fb6dcb964dd35c3f41b7b93ffc35eaf20bc61f2e1335f 8.15kB
 => => sha256:e66d0270d23f3038e0e8c94ee9244950fbfdb582476f61736b3c28 62
 => => sha256:55ac49bd649c325395133ae4f3640a07e28d9a25c4a56eb8ac3df9 95
 => => sha256:cbf42f5a00d268edb1684b8eb9039543669fc5f5d0aa801a01d346 36
 => => sha256:8015f365966bfa259003c319a44df5bb9290d279ca775b4f24 1.21kB
 => => sha256:4cadff8bc2aa83b23dd9e02a590174a84691f954eff4346888 1.40kB
 => => extracting sha256:648e0aadf75ac2ef63c5390adc6dc14fde37a5ad88c287
 => => extracting sha256:262696647b70a57f5f7dbf97a91091e7b51c1d2537dff7
 => => extracting sha256:e66d0270d23f3038e0e8c94ee9244950fbfdb582476f61
 => => extracting sha256:55ac49bd649c325395133ae4f3640a07e28d9a25c4a56e
 => => extracting sha256:cbf42f5a00d268edb1684b8eb9039543669fc5f5d0aa80
 => => extracting sha256:8015f365966bfa259003c319a44df5bb9290d279ca775b
 => => extracting sha256:4cadff8bc2aa83b23dd9e02a590174a84691f954eff434
 => [internal] load build context
 => => transferring context: 33.34kB
 => [2/6] COPY favicon.ico /usr/share/nginx/html/favicon.ico
 => [3/6] COPY index.html /usr/share/nginx/html/index.html
 => [4/6] COPY script.js /usr/share/nginx/html/script.js
 => [5/6] COPY style.css /usr/share/nginx/html/style.css
 => [6/6] COPY data.json /usr/share/nginx/html/data.json
 => exporting to image
 => => exporting layers
 => => writing image sha256:9f46c2925ff29c582eef7c32e63bc879fe3162cb49b
 => => naming to docker.io/library/guess-the-capital
```

3. List built images

```
docker images
```

```
theia@theiadocker-[redacted]:/home/project$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
guess-the-capital   latest    9a2dbca90e97   4 minutes ago    187MB
nginx               latest    eb4a57159180   7 days ago       187MB
```

4. Run the image

```
docker run -it -d -p 8080:80 guess-the-capital
```

5. Verify in browser

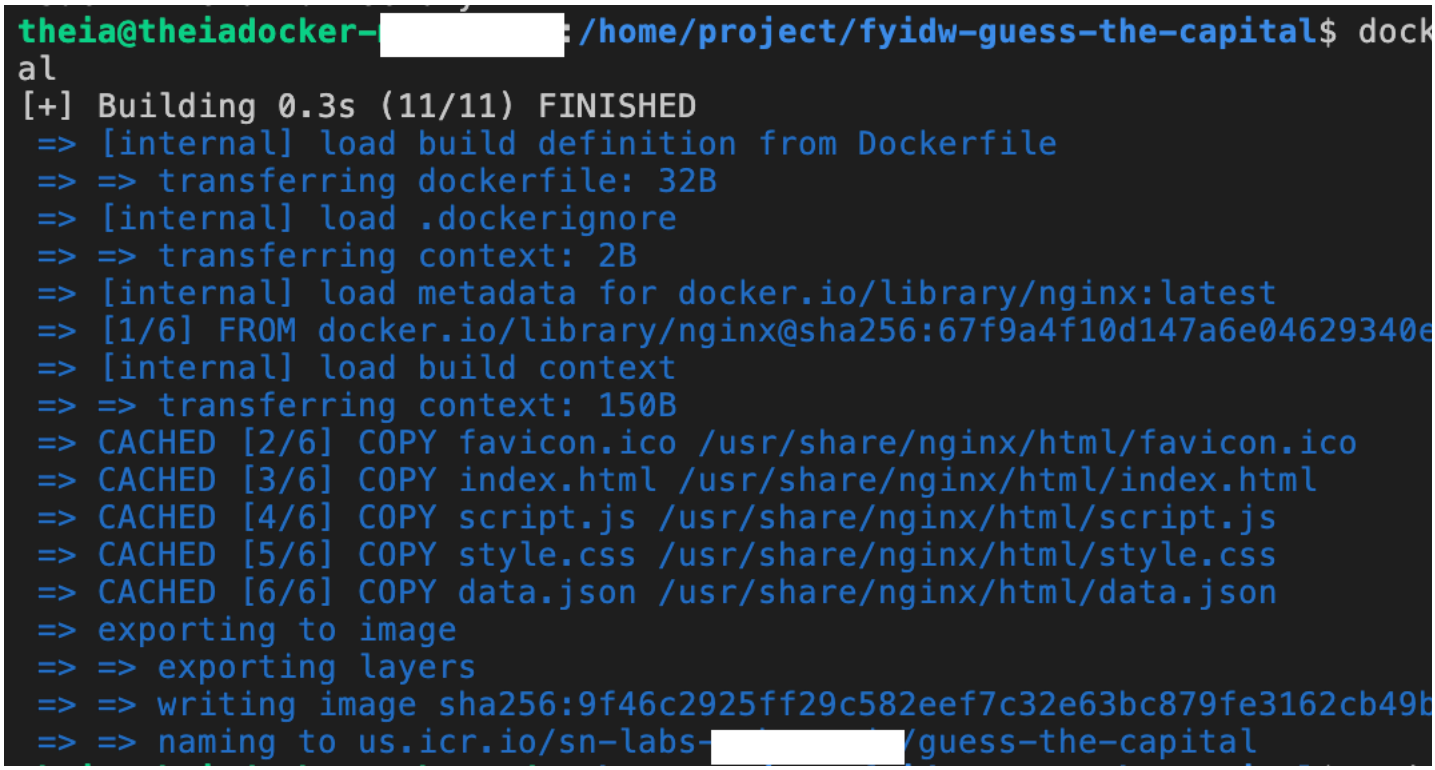
Launch Application

Task 2: Deploy on IBM Cloud

Let's start with launching Code Engine CLI.

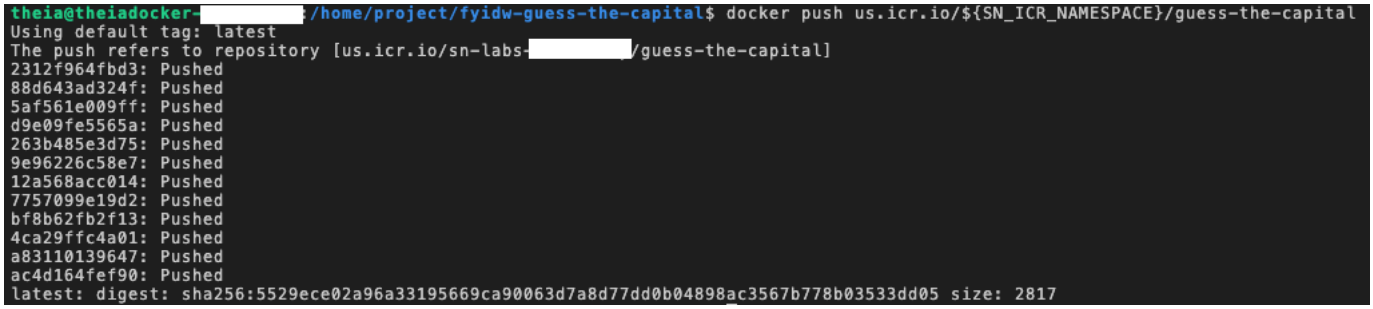
Create Code Engine Project in IDE

```
cd /home/project/fyidw-guess-the-capital
docker build . -t us.icr.io/${SN_ICR_NAMESPACE}/guess-the-capital
```



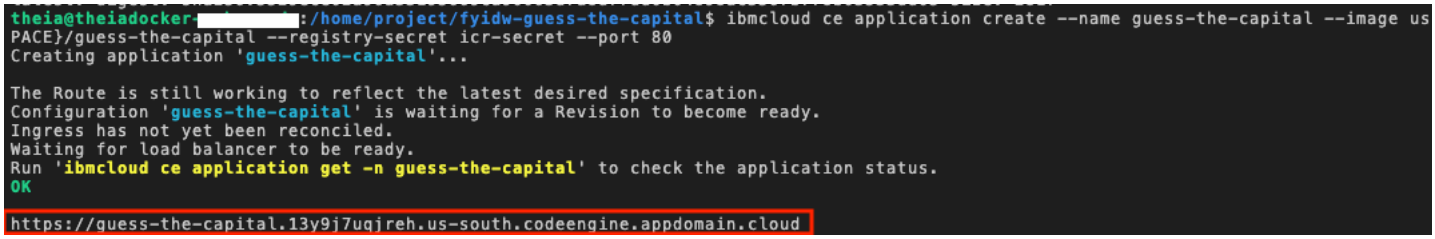
Push the image to IBM Cloud

```
docker push us.icr.io/${SN_ICR_NAMESPACE}/guess-the-capital
```



Deploy the image on IBM CE

```
ibmcloud ce application create --name guess-the-capital --image us.icr.io/${SN_ICR_NAMESPACE}/guess-the-capital --registry-secret ic
```



Take Cloud URL from the output; which looks something like: `https://guess-the-capital.somerandomalphanumeric.us-south.codeengine.appdomain.cloud` and open in your browser.

Optionally check the status

```
ibmcloud ce application get --name guess-the-capital
```

Congratulations

You have completed this final lab that showed you how to deploy and host a standard JavaScript application in Docker and on IBM Cloud.

Author(s)

Muhammad Yahya

(C) IBM Corporation 2023. All rights reserved.