

# Final Project - Part B and C: Tax Calculator using Python



Estimated time needed: 45 minutes

In Part B and C of this final project for the Cloud Native, DevOps, Agile, and NoSQL; you will work on Tax Calculator and enhance it using the skills you learned in this course.

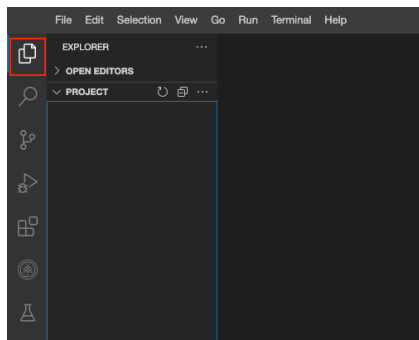
## Objectives:

1. Containerize the application
2. Run unit tests as part of the pipeline
3. Deploy the application when all tests pass
4. Replace the hard-coded configuration (tax brackets)
5. Deploy the application again when all tests pass

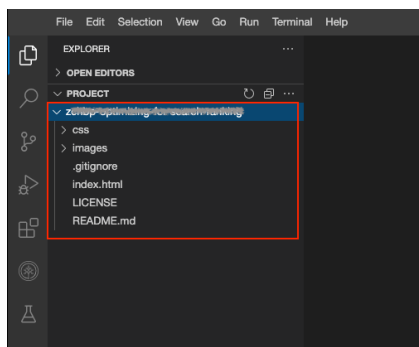
## Working with files in Cloud IDE

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.

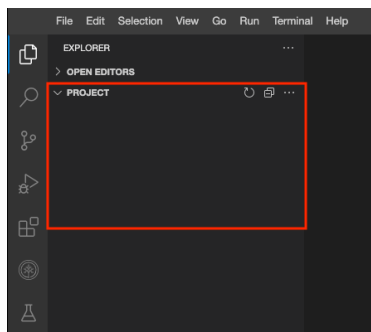
To view your files and directories inside Cloud IDE, click this files icon to reveal it.



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

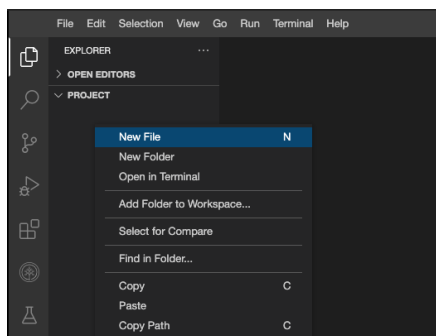


Otherwise, a blank project looks like this:



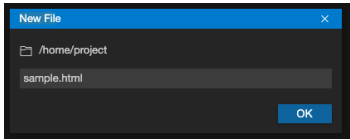
## Create a new file

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

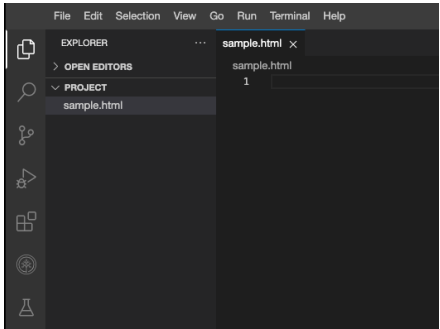


You can also choose `File -> New File` to do the same.

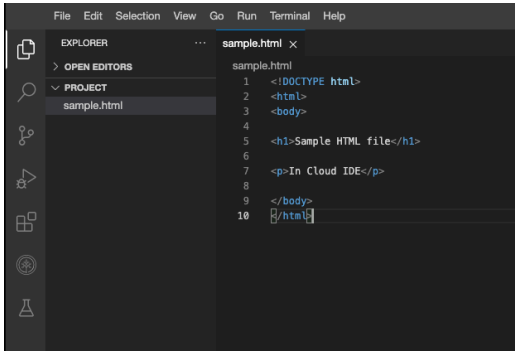
It will then prompt you to enter the name of this new file. In the example below, we are creating `sample.html`.



Clicking 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 files.

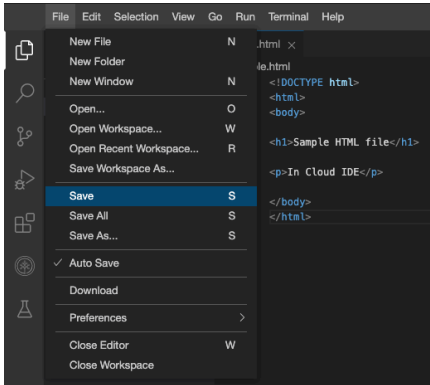


In the example, we just pasted some basic html code and then saved the file.



And saving it by:

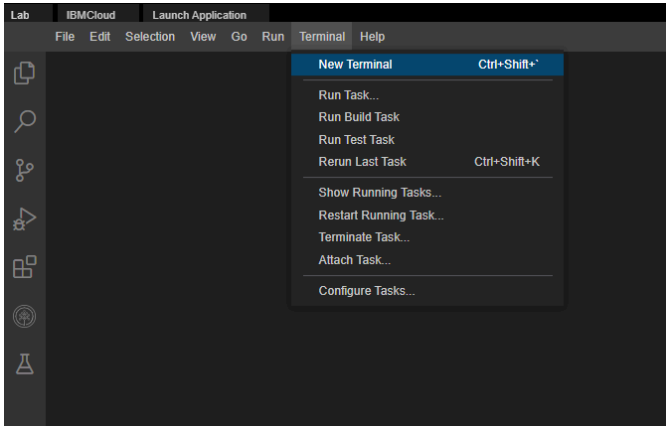
- Using the menu
- Press `command + S` on Mac or `CTRL + S` on Windows
- Or it can Autosave it for you too



## 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@theiaopenshift-rajashreep:/home/project$ docker --version
Docker version 28.0.1, build 068a01e
```

3. Verify that ibmCloud CLI is installed.

```
ibmcloud version
```

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

```
theia@theiaopenshift-rajashreep:/home/project$ ibmcloud version
ibmcloud version 2.20.0+f382323-2023-09-19T20:06:39+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.

The screenshot shows the Skills Network Toolbox interface. On the left, the 'CLOUD' dropdown menu is expanded, and 'Code Engine INACTIVE' is selected. The main panel displays the 'Code Engine' setup page. At the top, it says 'Code Engine' with a 'NOT READY' status. Below this, it shows the version '1.39.6' and a description: 'Use Code Engine directly in your Lab environment. To deploy serverless apps using Code Engine, Code Engine Projects are provided by Skills Network at no charge.' A prominent 'Create Project' button is visible. At the bottom, there are tabs for 'Summary', 'Project Information', and 'Details'. The 'Summary' tab is active, showing instructions on how to get started with Code Engine.

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

The screenshot shows the Skills Network Toolbox interface. On the left, the 'Code Engine' option is now 'STARTING' in the 'CLOUD' dropdown menu. The main panel displays the 'Code Engine' setup page. At the top, it says 'Code Engine' with a 'PREPARING' status. Below this, it shows the version '1.39.6' and a description: 'Use Code Engine directly in your Lab environment. To deploy serverless apps using Code Engine, Code Engine Projects are provided by Skills Network at no charge.' A 'Create Project' button is visible. At the bottom, there are tabs for 'Summary', 'Project Information', and 'Details'. The 'Summary' tab is active, showing instructions on how to get started with Code Engine.

3. Once the code engine setup 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.

FileEditSelectionViewGoRunTerminalHelp

SKILLS NETWORK TOOLBOX

> DATABASES

> BIG DATA

> CLOUD

Code Engine ACTIVE

Open IBM Cloud

> EMBEDDABLE AI

> OTHER

Launch Application

Code Engine

READY TO USE

1.39.6

Use Code Engine directly in your Lab environment. To deploy serverless apps using Code Engine Code Engine Projects are provided by Skills Network at no charge.

Delete Project

SummaryProject InformationDetails

Your Skills Network Code Engine Project is now ready to use. You can now create and manage Applications.

For important information about your project view the Project Information section. For more details about an IBM Cloud Service, please check out the Details section.

In order to interact with Code Engine please click the following button:

Code Engine CLI

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.

FileEditSelectionViewGoRunTerminalHelp

SKILLS NETWORK TOOLBOX

> DATABASES

> BIG DATA

> CLOUD

Code Engine ACTIVE

Open IBM Cloud

> EMBEDDABLE AI

> OTHER

Launch Application

Code Engine

READY TO USE

1.39.6

Use Code Engine directly in your Lab environment. To deploy serverless apps using Code Engine Code Engine Projects are provided by Skills Network at no charge.

Delete Project

SummaryProject InformationDetails

Your Skills Network Code Engine Project is now ready to use. You can now create and manage your Applications.

For important information about your project view the Project Information section. For more details about an IBM Cloud Service, please check out the Details section.

In order to interact with Code Engine please click the following button:

Code Engine CLI

Problems

theia@theiadocker-captainfedo1: /home/project

theia@theiadocker-captainfedo1: /home/project

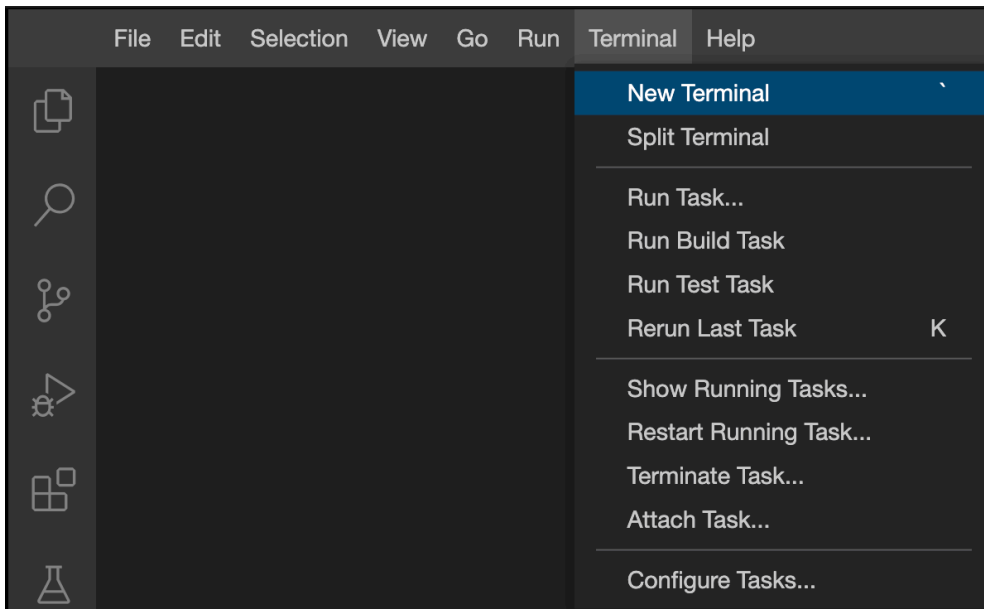
ibmcloud ce project current  
theia@theiadocker-captainfedo1:/home/project\$ ibmcloud ce project current  
Getting the current project context...  
OK  
  
Name: Code Engine - sn-labs-captainfedo1  
ID: 9c079722-5f80-4056-bed6-798cdb0acf04  
Subdomain: ywj8nhvp9f9  
Domain: us-south.codeengine.appdomain.cloud  
Region: us-south  
  
Kubernetes Config:  
Context: ywj8nhvp9f9  
Environment Variable: export KUBECONFIG="/home/theia/.bluemix/plugins/code-engine-n-labs-captainfedo1-9c079722-5f80-4056-bed6-798cdb0acf04.yaml"  
theia@theiadocker-captainfedo1:/home/project\$

Another way to get to it is by clicking the following action:

Create Code Engine Project in IDE

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.

```
git clone https://github.com/ibm-developer-skills-network/zntwk-tax_calculator.git
```

4. Change to the directory **zntwk-tax\_calculator** to start working on the lab.

```
cd zntwk-tax_calculator
```

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

```
ls
```

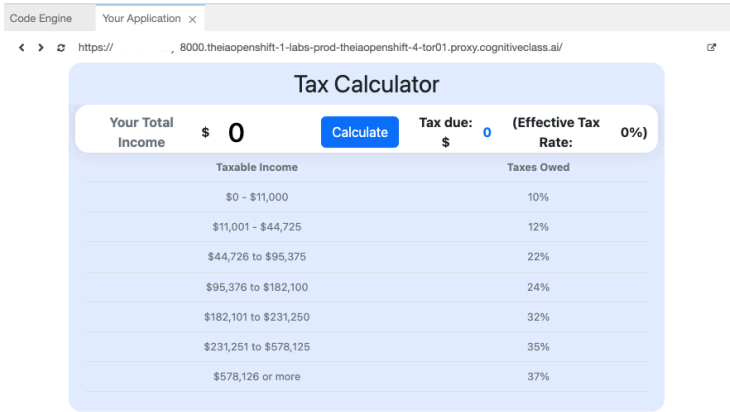
6. Run the following command on the terminal to host your web page.

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

7. To test your application in the 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.

## Creating Tax calculator App

Note:

1. Please ensure that all updates to the `tasks.yaml`, `pipeline.yaml`, and `run.yaml` files are properly saved.
2. Save the text files, URLs and capture screenshots as mentioned in the respective tasks.
3. As mentioned in the “*Final Project Overview and Review Criteria*”, you can submit your project deliverables through either Option 1: AI-Graded Submission and Evaluation or Option 2: Peer-Graded Submission and Evaluation.
4. **For Option 1: AI-Graded Submission and Evaluation:**
  - Submission requires the terminal output, code snippet, URLs, along with screenshot for **Task 10**.
5. **For Option 2: Peer-Graded Submission and Evaluation:**
  - Submission requires screenshots for **Tasks 1 to 10**

## Task 3: Manual unit test

### Unit Tests

Unit tests are a type of software testing where individual components, or units, of a software application, are tested in isolation to ensure that they function correctly. The goal of unit testing is to verify that each unit of code performs as expected and meets its specified requirements.

### Jasmine

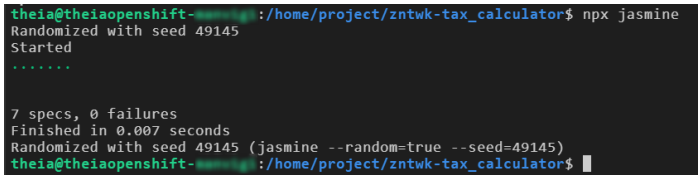
Jasmine is a popular testing framework for JavaScript that provides a behavior-driven development (BDD) style syntax for writing unit tests.

### Steps

```
npm install
npm install jasmine
```

Which should produce an output like the below showing all 7 unit tests have passed:

```
Randomized with seed 78367
Started
.....
7 specs, 0 failures
Finished in 0.013 seconds
Randomized with seed 78367 (jasmine --random=true --seed=78367)
```



Assessment:

**For Option 1 - AI Graded Submission and Evaluation:** Copy and paste the terminal output and save it in a textfile named `01-jasmine-tests-passing` for the Final Project Submission and Evaluation.

**For Option 2 - Peer Graded Submission and Evaluation:** Take a screenshot of the output and save it as `01-jasmine-tests-passing.png` for Peer Assignment.

## Task 4: Containerize the application

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

### What is Docker

Docker is an open-source platform and toolset that allows you to automate the deployment, scaling, and management of applications using containerization. Containers are lightweight, isolated, and portable environments that package applications and their dependencies, enabling them to run consistently across different computing environments.

### Create Dockerfile

Your task:

1. Define `nginx` as parent image.
2. Copy each file individually to `/usr/share/nginx/html/`.

[Open Dockerfile in IDE](#)

▼ Click here for Hint

`FROM` is a keyword that you need to use, where as `nginx` is the image.

`COPY filename.ext /usr/share/nginx/html/filename.ext` can help you copy files to your container.

▼ Click here for the solution  
Use the following 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 taxCalculator.js /usr/share/nginx/html/taxCalculator.js
```

Assessment

**For Option 1 - AI Graded Submission and Evaluation:** Copy and paste the code of **Dockerfile** and save it in a text file named **02-dockerfile** for the Final Project Submission and Evaluation.

**For Option 2 - Peer Graded Submission and Evaluation:** Take a screenshot of the output and save it as **02-dockerfile.png** for Peer Assignment.

Build Docker image

Now that we have created the Dockerfile, let's build our custom image. You will name the image **tax-calculator**.

▼ Click here for Hint

```
docker build -t IMAGE DIRECTORY
```

▼ Click here for Solution

```
docker build -t tax-calculator .
```

```
theia@theiaopenshift-: /home/project/zntwk-tax_calculator$ docker build -t tax-calculator .
[+] Building 14.5s (11/11) FINISHED
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 304B 0.0s
=> [internal] load metadata for docker.io/library/nginx:latest 1.5s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/6] FROM docker.io/library/nginx:latest@sha256:088eea90c3d0a540ee5686e7d7471acbd40 11.4s
=> => resolve docker.io/library/nginx:latest@sha256:088eea90c3d0a540ee5686e7d7471acbd406 0.0s
=> => sha256:c29f5b76f736a8b555fd191c48d6581bb918bcd605a7cbcc76205dd6a 28.21MB / 28.21MB 2.8s
=> => sha256:e19db8451adb4b9c6dc54aaec5c9fd8917d5267d249a977b95dab00ce 43.95MB / 43.95MB 4.0s
=> => sha256:088eea90c3d0a540ee5686e7d7471acbd4063b6e97eaf49b5e651665eb7 2.29kB / 2.29kB 0.0s
=> => sha256:97662d24417b316f60607afbca9f226a2ba58f09d642f27b8e197a89859 8.58kB / 8.58kB 0.0s
=> => sha256:24ff42a0d907c9f6e59a6ce8ad8691f8b72ffad5e1e55584612b86cb632f5cf 627B / 627B 0.4s
=> => sha256:c558df2179491b5490a7b895e422fffcddcfac6c045d49907048d32a2ab578 955B / 955B 0.9s
=> => sha256:976e8f6b25dd4b8451fa52d71524f61103d32dca1af6b1ddb70b0ea848a3e37 403B / 403B 1.4s
=> => sha256:6c78b0ba1a32d228d04680a4dc7bd67a21f2e30f6fc032b5648d0cae756 1.21kB / 1.21kB 1.9s
=> => sha256:84cade77a8319515342dbc0f9d9ce4bbddd2d95f93646091b61a4ca3181 1.40kB / 1.40kB 2.4s
=> => extracting sha256:c29f5b76f736a8b555fd191c48d6581bb918bcd605a7cbcc76205dd6aef3260 2.5s
=> => extracting sha256:e19db8451adb4b9c6dc54aaec5c9fd8917d5267d249a977b95dab00ce960e4bf 1.4s
=> => extracting sha256:24ff42a0d907c9f6e59a6ce8ad8691f8b72ffad5e1e55584612b86cb632f5cfa 0.0s
=> => extracting sha256:c558df2179491b5490a7b895e422fffcddcfac6c045d49907048d32a2ab5784 0.0s
=> => extracting sha256:976e8f6b25dd4b8451fa52d71524f61103d32dca1af6b1ddb70b0ea848a3e373 0.0s
=> => extracting sha256:6c78b0ba1a32d228d04680a4dc7bd67a21f2e30f6fc032b5648d0cae756fe14 0.0s
=> => extracting sha256:84cade77a8319515342dbc0f9d9ce4bbddd2d95f93646091b61a4ca318164a44 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 23.84kB 0.0s
=> [2/6] COPY favicon.ico /usr/share/nginx/html/favicon.ico 0.1s
=> [3/6] COPY index.html /usr/share/nginx/html/index.html 0.0s
=> [4/6] COPY script.js /usr/share/nginx/html/script.js 0.0s
=> [5/6] COPY style.css /usr/share/nginx/html/style.css 0.0s
=> [6/6] COPY taxCalculator.js /usr/share/nginx/html/taxCalculator.js 0.0s
=> exporting to image 1.1s
```

Assessment:

**Option 1 - AI Graded Submission and Evaluation:** Copy and paste the terminal output and save it in a text file named **03-docker-build-output** for the Final Project Submission and Evaluation.

**Option 2 - Peer Graded Submission and Evaluation:** Take a screenshot of the output and save it as **03-docker-build-output.png** for Peer Assignment.

List Docker images and verify you see **tax-calculator**.

```
docker images
```

Deploy Docker image

Once you build the container, deploy the container on port 8080.

▼ Click here for a hint

```
docker run FLAGS PORT_MAPPING IMAGE
```

- -i is short for –interactive. Keep STDIN open even if unattached.
- -t is short for –tty. Allocates a pseudo terminal that connects your terminal with the container's TTY.
- -p –port. 8080:80 maps the Docker port 80 to port 8080 on your Cloud IDE instance.

▼ Click here for the solution

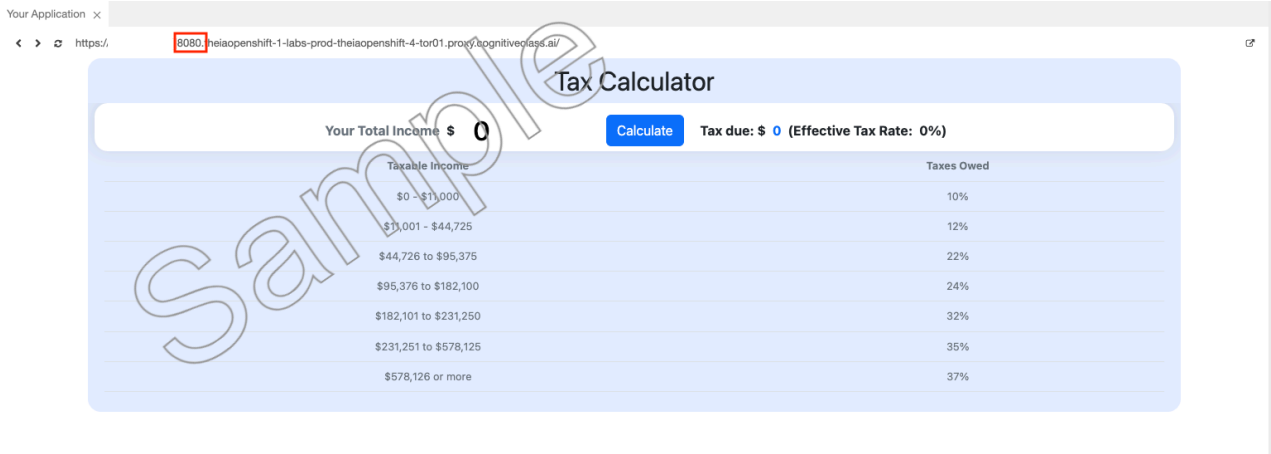
```
docker run -it -d -p 8080:80 tax-calculator
```

Assessment:

**For Option 1 - AI Graded Submission and Evaluation:** Copy and paste the terminal output in a text file and save it as 04-docker-image for the Final Project Evaluation.

Test the application in a browser.

Launch Application



**For Option 2 - Peer Graded Submission and Evaluation:** Take a screenshot of the output and save it as 04-docker-image-browser.png for Peer Assignment.

And finally, clean up running containers.

```
docker ps -aq | xargs docker stop | xargs docker rm
```

Task 5: Manual deploy on IBM Cloud

As part of this learning platform, you are provided with basic infrastructure on IBM Cloud. You don't need an external sign up, and all setup has been done for you already.

Create Code Engine Project in IDE

Build and Tag Docker image

The Docker build command builds Docker images from a Dockerfile.

▼ Click here for a hint

```
docker build DIRECTORY FLAGS us.icr.io/${SN_ICR_NAMESPACE}/tax-calculator
```

- --tag , -t Name and optionally a tag in the name:tag format

▼ Click here for the solution

```
cd /home/project/zntwk-tax_calculator
docker build . -t us.icr.io/${SN_ICR_NAMESPACE}/tax-calculator
```

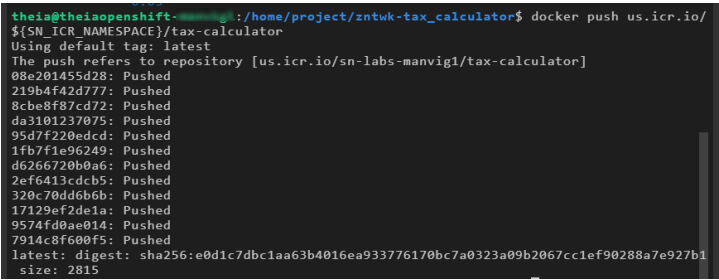
Push the image to IBM Cloud

To upload an image to a registry, use docker push.

Let's push the image to IBM Cloud.

▼ Click here for the solution

```
docker push us.icr.io/${SN_ICR_NAMESPACE}/tax-calculator
```



Assessment:

**For Option 1 - AI Graded Submission and Evaluation:** Copy and paste the terminal output and save it in a text file named 05-docker-icr-push for the Final Project Submission and Evaluation.

**For Option 2 - Peer Graded Submission and Evaluation:** Take a screenshot of the output and save it as 05-docker-icr-push.png for Peer Assignment.

Deploy the image to IBM Cloud

Deploy the image on IBM Cloud Code Engine.

```
ibmcloud ce application create --name tax-calculator --image us.icr.io/${SN_ICR_NAMESPACE}/tax-calculator --registry-secret icr-secret --port 80
```

Check the status

```
ibmcloud ce application get --name tax-calculator
```

```

awslogs:loggroupshift --log-group-name /home/project/2018-tax-calculator --log-type application --name tax-calculator
OK

Name: tax-calculator
ID: 8000131-8756-4564-8368-37228674789
Code Engine: us-labs-mav1g1
Project ID: cfc1848b-6364-45ff-b73c-538af4f2533
E3s
URL: https://api.us-south.codeengine.cloud.ibm.com
Cluster Local URL: https://tax-calculator-lsw-amthb21.us-south.codeengine.appdomain.cloud
Console URL: https://cloud.ibm.com/codeengine/project/us-south/cfc1848b-6364-45ff-b73c-538af4f2533/application/tax-calculator/configurations
Status Summary: Application deployed successfully

Environment Variables:
Type Name Value
Literal CE_APP https://api.us-south.codeengine.cloud.ibm.com
Literal CE_DOMAIN tax-calculator
Literal CE_DOMAIN us-south.codeengine.appdomain.cloud
Literal CE_PROJECT_ID cfc1848b-6364-45ff-b73c-538af4f2533
Literal CE_REGION us-south
Literal CE_SUBDOMAIN lsw-amthb21
Image: us.icr.io/sb-labs-000001/tax-calculator

Resource Allocation:
CPU: 1
Ephemeral Storage: 4800
Memory: 4G
Registry Secrets:
icr-secret
Port: 80

Revisions:
tax-calculator-000001:
Age: E3s
Latest: true
Traffic: 100%
Image: us.icr.io/sb-labs-000001/tax-calculator (pinned to eb01c7)
Running Instances: 1

Routes:
Concurrency: 100
Maximum Scale: 10
Minimum Scale: 0
Scale Down Delay: 100
Timeout: 300

Readiness Probe:
Type: tcp
Port: 8 (use listening port)

Conditions:
Type OK Age Reason
ConfigurationsReady true 51s
Ready true 48s
RoutesReady true 46s

Events:
Type Reason Age Source Messages
Normal Created 64s service-controller Created Configuration "tax-calculator"
Normal Created 64s service-controller Created Route "tax-calculator"

Instances:
Name Revision Running Status Restarts Age
tax-calculator-00001-deployment-5f6b5f4dc8-9rcvc tax-calculator-00001 3/3 Running 0 E3s

For troubleshooting information visit: https://cloud.ibm.com/docs/codeengine?topic=codeengine-troubleshoot-apps.
Run ibmcloud ce application events -n tax-calculator to get the system events of the application instances.
The ibmcloud ce application logs -n tax-calculator to follow the logs of the application instances.

```

**Assessment:**

**For Option 1 - AI Graded Submission and Evaluation:** Copy and paste the terminal output and save it in a text file named `06-deployed-on-cloud` for the Final Project Submission and Evaluation.

**For Option 2 - Peer Graded Submission and Evaluation:** Take a screenshot of the output and save it as 06-deployed-on-cloud.png for Peer Assignment.

Take Cloud URL from the output; which looks something like the following: `https://tax-calculator.somerandomalphanumeric.us-south.codeengine.appdomain.cloud`

### Task 6: Setup Pipeline

You will now create the pipeline to perform manual actions as part of Continuous Deployment.

Open a new terminal and go to `/home/project`.

```
cd /home/project
```

The GitHub repository of the project is available on the URL mentioned below.

As a first step you need to Fork this repository by clicking on the Fork button at the top-right corner of the repository page and keep the repository name unchanged.

[https://github.com/ibm-developer-skills-network/juphc-tc\\_pipeline](https://github.com/ibm-developer-skills-network/juphc-tc_pipeline)

You can find instructions on how to fork the repository by visiting exercise 2 at the following [link](#).

**Note:** Ensure your forked repository is Public. This action will create a copy of the forked repository within your GitHub account.

Open a new terminal and clone the forked repository and change to the directory **juphc-tc pipeline** to start working on the pipeline.

```
cd juphc-tc_pipeline
```

You can also see your code in the Project structure.

## Task 7: Add Tasks for Pipeline use

Open **tasks.yaml** in IDE

## Create npminstall task

Add a new task called `npminstall`, using the image `node:20-buster-slim`, running these commands:

```
set -e
echo "***** Environment *****"
npm --version
pwd
echo "***** Cleaning npm cache *****"
npm cache clean --force
```

```
echo "***** Running npm install *****"
npm install
```

▼ Click here for Hint

```
apiVersion: tekton.dev/v1beta1
kind: Task
metadata:
  name: npm
spec:
  workspaces:
    - name: source
  steps:
    - name: TASK_NAME
      image: IMAGE_NAME
      workingDir: ${workspaces.source.path}
      script: |
        #!/bin/bash
        COMMANDS
```

▼ Click here for Solution

```
apiVersion: tekton.dev/v1beta1
kind: Task
metadata:
  name: npm
spec:
  workspaces:
    - name: source
  steps:
    - name: npminstall
      image: node:20-buster-slim
      workingDir: ${workspaces.source.path}
      script: |
        #!/bin/bash
        set -e
        echo "***** Environment *****"
        npm --version
        pwd
        echo "***** Cleaning npm cache *****"
        npm cache clean --force
        echo "***** Running npm install *****"
        npm install
---
```

## Create the Jasmine task

Add a new task called jasmine, using the image node:20-buster-slim, running these commands:

```
set -e
echo "***** Running jasmine tests *****"
npx jasmine
```

▼ Click here for a hint

```
apiVersion: tekton.dev/v1beta1
kind: Task
metadata:
  name: npm
spec:
  workspaces:
    - name: source
  steps:
    - name: TASK_NAME
      image: IMAGE_NAME
      workingDir: ${workspaces.source.path}
      script: |
        #!/bin/bash
        COMMANDS
```

▼ Click here for the solution

```
apiVersion: tekton.dev/v1beta1
kind: Task
metadata:
  name: jasmine
spec:
  workspaces:
    - name: source
  steps:
    - name: jasmine tests
      image: node:20-buster-slim
      workingDir: ${workspaces.source.path}
      script: |
        #!/bin/bash
        set -e
        echo "***** Running jasmine tests *****"
        npx jasmine
```

Please push your code to your forked GitHub repository, you can do so by following the instructions provided in this [link](#).

```

---
apiVersion: tekton.dev/v1beta1
kind: Task
metadata:
  name: npm
spec:
  workspaces:
  - name: source
  steps:
  - name: npminstall
    image: node:20-buster-slim
    workingDir: ${workspaces.source.path}
    script: |
      #!/bin/bash
      set -e
      echo "***** Environment *****"
      npm --version
      pwd
      echo "***** Cleaning npm cache *****"
      npm cache clean --force
      echo "***** Running npm install *****"
      npm install

---
apiVersion: tekton.dev/v1beta1
kind: Task
metadata:
  name: jasmine
spec:
  workspaces:
  - name: source
  steps:
  - name: jasminetests
    image: node:20-buster-slim
    workingDir: ${workspaces.source.path}
    script: |
      #!/bin/bash
      set -e
      echo "***** Running jasminetests *****"
      npx jasmine

```

#### Assessment:

**For Option 1 - AI Graded Submission and Evaluation:** Copy and paste the the public Github URL of the tasks.yaml and save it in a text file for the Final Project Submission and Evaluation.

**For Option 2 - Peer Graded Submission and Evaluation:** Take a screenshot of the tasks.yaml and save it as 07-tasks.yaml.png for Peer Assignment.

## Create a Secret

Workspaces allow Tasks to declare parts of the filesystem that need to be provided at runtime by TaskRuns. A TaskRun can make these parts of the filesystem available in many ways:

- using a read-only ConfigMap or Secret
- an existing PersistentVolumeClaim shared with other Tasks
- create a PersistentVolumeClaim from a provided VolumeClaimTemplate
- or simply an emptyDir that is discarded when the TaskRun completes

The first step you need to create that secret, as mentioned above.

**Secret:** A Secret is an object that contains a small amount of sensitive data, such as a password, a token, or a key.

You need the following secret to be able to deploy to IBM Container Registry.

```

kubectl create secret docker-registry docker-reg-creds \
--docker-server=us.icr.io \
--docker-username=iamapikey \
--docker-password=$IBM_CLOUD_API_KEY

```

## Task 8: Create pipeline

Because our DevOps process has been streamlined with pipeline for testing, packaging and deploying, we simply need to trigger it to deploy with full confidence that it will work as long as it meets all the criteria.

Similar to boilerplate.yaml for Tasks, you are provided with pipeline.yaml.

### Call npminstall task

Add a new task called npminstall where taskRef/ name npm running after clone:

▼ Click here for a hint

```

- name: NAME
  workspaces:
  - name: source
    workspace: pipeline-workspace
  taskRef:
    name: TASKREF-NAME
  runAfter:
  - RUNNING-AFTER

```

▼ Click here for a solution

```

- name: npminstall
  workspaces:
  - name: source
    workspace: pipeline-workspace
  taskRef:
    name: npm
  runAfter:
  - clone

```

### Call jasmine task

Add a new task called tests where taskRef/ name jasmine running after npminstall:

▼ Click here for a hint

```

- name: NAME
  workspaces:
  - name: source
    workspace: pipeline-workspace
  taskRef:
    name: TASKREF-NAME
  runAfter:
  - RUNNING-AFTER

```

▼ Click here for the solution

```
- name: tests
workspaces:
  - name: source
    workspace: pipeline-workspace
taskRef:
  name: jasmine
runAfter:
  - npminstall
```

## Call buildah task

Add a new task called build where taskRef/ name buildah with kind ClusterTask, passing the IMAGE parameter to the value \$(params.build-image), running after tests.

Additionally, instead of just pipeline-workspace, you will also need to add dockerconfig-ws too.

▼ Click here for a hint

```
- name: NAME
workspaces:
  - name: source
    workspace: pipeline-workspace
  - name: dockerconfig
    workspace: dockerconfig-ws
taskRef:
  name: TASKREF-NAME
  kind: KIND
params:
  - name: PARAM-NAME
    value: "PARAM-VALUE"
runAfter:
  - RUNNING-AFTER
```

▼ Click here for the solution

```
- name: build
workspaces:
  - name: source
    workspace: pipeline-workspace
taskRef:
  name: buildah
  kind: ClusterTask
params:
  - name: IMAGE
    value: "us.icr.io/$(context.pipelineRun.namespace)/$(params.app-name)"
runAfter:
  - tests
```

Note: Please push your code to your forked GitHub repository.

```
pipeline.yaml x
to-pipeline > pipeline.yaml
23 |
24 | - name: clone
25 |   workspaces:
26 |     - name: output
27 |       workspace: pipeline-workspace
28 |   taskRef:
29 |     name: git-clone
30 |     kind: ClusterTask
31 |   params:
32 |     - name: url
33 |       value: $(params.repo-url)
34 |     - name: revision
35 |       value: $(params.branch)
36 |   runAfter:
37 |     - init
38 | - name: npminstall
39 |   workspaces:
40 |     - name: source
41 |       workspace: pipeline-workspace
42 |   taskRef:
43 |     name: npm
44 |   runAfter:
45 |     - clone
46 | - name: tests
47 |   workspaces:
48 |     - name: source
49 |       workspace: pipeline-workspace
50 |   taskRef:
51 |     name: jasmine
52 |   runAfter:
53 |     - npminstall
54 | - name: build
55 |   workspaces:
56 |     - name: source
57 |       workspace: pipeline-workspace
58 |     - name: dockerconfig
59 |       workspace: dockerconfig-ws
60 |   taskRef:
61 |     name: buildah
62 |     kind: ClusterTask
63 |   params:
64 |     - name: IMAGE
65 |       value: "us.icr.io/$(context.pipelineRun.namespace)/$(params.app-name)"
66 |   runAfter:
67 |     - tests
```

Assessment:

**For Option 1 - AI Graded Submission and Evaluation:** Copy and paste the public Github URL of pipeline.yaml and save it in a text file for the Final Project Submission and Evaluation.

**For Option 2 - Peer Graded Submission and Evaluation:** Take a screenshot of the pipeline.yaml and save it as 08-pipeline-changes.png for Peer Assignment.

▼ Click here for see final pipeline.yaml

```
apiVersion: tekton.dev/v1beta1
kind: Pipeline
metadata:
  name: tc-pipeline
```

```
spec:
  workspaces:
    - name: pipeline-workspace
    - name: dockerconfig-ws
  params:
    - name: repo-url
    - name: app-name
      default: tax-calculator
    - name: branch
      default: main
  tasks:

    - name: init
      workspaces:
        - name: source
          workspace: pipeline-workspace
      taskRef:
        name: cleanup

    - name: clone
      workspaces:
        - name: output
          workspace: pipeline-workspace
      taskRef:
        name: git-clone
        kind: ClusterTask
      params:
        - name: url
          value: ${params.repo-url}
        - name: revision
          value: ${params.branch}
      runAfter:
        - init
    - name: npminstall
      workspaces:
        - name: source
          workspace: pipeline-workspace
      taskRef:
        name: npm
      runAfter:
        - clone
    - name: tests
      workspaces:
        - name: source
          workspace: pipeline-workspace
      taskRef:
        name: jasmine
      runAfter:
        - npminstall
    - name: build
      workspaces:
        - name: source
          workspace: pipeline-workspace
        - name: dockerconfig
          workspace: dockerconfig-ws
      taskRef:
        name: buildah
        kind: ClusterTask
      params:
        - name: IMAGE
          value: "us.icr.io/${context.pipelineRun.namespace}/${params.app-name}"
      runAfter:
        - tests
```

Task 9: Create run.yamll

Now you need to create tc-pipeline/run.yamll that will run your pipeline.

[Open run.yamll in IDE](#)

A PipelineRun allows you to instantiate and execute a Pipeline on-cluster. A Pipeline specifies one or more Tasks in the desired order of execution. A PipelineRun executes the Tasks in the Pipeline in the order they are specified until all Tasks have executed successfully or a failure occurs.

In your run.yaml file, create a definition of kind PipelineRun, with the following:

Path	Value
kind	PipelineRun
metadata.generateName	tc-pipeline-run-
spec.serviceAccountName	pipeline
spec.pipelineRef.name	tc-pipeline
workspaces.name	pipeline-workspace
workspaces.persistentVolumeClaim.claimName	pipelinerun-pvc
workspaces.name	dockerconfig-ws
workspaces.secret.secretName	docker-reg-creds

Along with following params:

param.name	param.value
repo-url	"https://github.com/ibm-developer-skills-network/zmtwk-tax_calculator.git"
branch	"main"
app-name	"tax-calculator"

▼ Click here for a hint

```
apiVersion: tekton.dev/v1beta1
kind: KIND
metadata:
  generateName: GENERATE_NAME
spec:
  serviceAccountName: SERVICE_ACCOUNT
  pipelineRef:
    name: PIPELINE_NAME
  params:
    - name: FIRST_PARAM
      value: FIRST_PARAM_VALUE
    - name: SECOND_PARAM
      value: SECOND_PARAM_VALUE
  workspaces:
    - name: WORKSPACE_1
      persistentVolumeClaim:
        claimName: PERSISTENT_VOLUME_CLAIM_NAME
    - name: WORKSPACE_2
      secret:
        secretName: SECRET_NAME
```

▼ Click here for the solution

```
apiVersion: tekton.dev/v1beta1
kind: PipelineRun
metadata:
  generateName: tc-pipeline-run-
spec:
  serviceAccountName: pipeline
  pipelineRef:
    name: tc-pipeline
  params:
```

```
- name: repo-url
  value: "https://github.com/ibm-developer-skills-network/zntwk-tax_calculator.git"
- name: branch
  value: "main"
- name: app-name
  value: "tax-calculator"
workspaces:
- name: pipeline-workspace
  persistentVolumeClaim:
    claimName: pipelinerun-pvc
- name: dockerconfig-ws
  secret:
    secretName: docker-reg-creds
```

Note: Please push your code to your forked GitHub repository.

juphc-tc\_pipeline >  run.yaml > ...

```
1  apiVersion: tekton.dev/v1beta1
2  kind: PipelineRun
3  metadata:
4    generateName: tc-pipeline-run-
5  spec:
6    serviceAccountName: pipeline
7    pipelineRef:
8      name: tc-pipeline
9    params:
10     - name: repo-url
11       value: "https://github.com/ibm-developer-skills-network/zntwk-tax_calculator.git"
12     - name: branch
13       value: "main"
14     - name: app-name
15       value: "tax-calculator"
16    workspaces:
17     - name: pipeline-workspace
18       persistentVolumeClaim:
19         claimName: pipelinerun-pvc
20     - name: dockerconfig-ws
21       secret:
22         secretName: docker-reg-creds
```

#### Assessment:

**For Option 1 - AI Graded Submission and Evaluation:** Copy and paste the the public Github URL of run.yaml and save it in a text file for the Final Project Submission and Evaluation.

**For Option 2 - Peer Graded Submission and Evaluation:** Take a screenshot of the run.yaml and save it as 09-run.yaml.png for Peer Assignment.

## Task 10: Apply, Create, and Deploy

Now you have all you need to build and push using pipeline. Let's start it:

#### Kubectl apply

Apply manages applications through files defining Kubernetes resources. It creates and updates resources in a cluster through running kubectl apply. This is the recommended way of managing Kubernetes applications on production.

#### Apply Persistent Volume Claim

▼ Click here for a hint

```
kubectl apply -f FILENAME.EXT
```

▼ Click here for the solution

```
kubectl apply -f pvc.yaml
```

#### Install ibmcloud task in our namespace.

▼ Click here for hint

Apply the official Tekton Catalog task manifest for ibmcloud to your Kubernetes cluster using kubectl

▼ Click here for solution

```
kubectl apply -f https://raw.githubusercontent.com/tektoncd/catalog/main/task/ibmcloud/0.1/ibmcloud.yaml
```

Apply tasks.yaml and pipeline.yaml

▼ Click here for a hint

```
kubect1 apply -f FILANAME.EXT
```

▼ Click here for the solution

```
kubect1 apply -f tasks.yaml
kubect1 apply -f pipeline.yaml
```

kubect1 create run.yaml

The command, set kubect1 apply, is used at a terminal's command-line window to create or modify Kubernetes resources defined in a manifest file. This is called a declarative usage. The state of the resource is declared in the manifest file, then kubect1 apply is used to implement that state.

In contrast, the command, set kubect1 create, is the command you use to create a Kubernetes resource directly at the command line. This is an imperative usage. You can also use kubect1 create against a manifest file to create a new instance of the resource.

▼ Click here for a hint

```
kubect1 create -f FILANAME.EXT -n NAMESPACE
```

▼ Click here for the solution

```
kubect1 create -f run.yaml -n $SN_ICR_NAMESPACE
```

Monitor the run

You retrieve last logs with the following command:

```
tkn pipelinerun logs --last
```

Deploy on the cloud

Before you issue the IBM Code Engine CLI command to deploy your tagged application, make sure your Code Engine has correct details in the context.

Create Code Engine Project in IDE

Once the code engine setup 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.

File

Edit

Selection

View

Go

Run

Terminal

Help

SKILLS NETWORK TOOLBOX

> DATABASES

> BIG DATA

> CLOUD

Code Engine ACTIVE

Open IBM Cloud

> EMBEDDABLE AI

> OTHER

Launch Application

Code Engine

Code Engine

1.39.6

Use Code Engine directly in your Lab environment. To deploy serverless apps using Code Engine Code Engine Projects are provided by Skills Network at no charge.

Delete Project

SummaryProject InformationDetails

Your Skills Network Code Engine Project is now ready to use. You can now create and manage yo Applications.

For important information about your project view the Project Information section. For more detail an IBM Cloud Service, please check out the Details section.

In order to interact with Code Engine please click the following button:

Code Engine CLI

Verify

ibmcloud ce project current

Deploy

ibmcloud ce application create --name tax-calculator --image us.icr.io/\${SN\_ICR\_NAMESPACE}/tax-calculator --registry-secret icr-secret --port 80

Note: if you encounter an error related to the application name, it may be because you have previously deployed an application with the same name. In such cases, please rename the application to proceed.

Verify in the browser

You deployed from another branch so you can see the difference.

Assessment: Take a screenshot of the final output and save it as 10-final-output.png for both the evaluation options.

Tax Calculator

Your Total Income \$ 0

Calculate

Tax due: \$ 0 (Effective Tax Rate: 0%)

Taxable Income	Taxes Owed
\$0 - \$11,000	10%
\$11,001 - \$44,725	12%
\$44,726 to \$95,375	22%
\$95,376 to \$182,100	24%
\$182,101 to \$231,250	32%
\$231,251 to \$578,125	35%
\$578,126 or more	37%

Summary

Congratulations

You have now completed all of the enhancements needed for Tax Calculator.

In this lab, you have gained an understanding of the requirements to obtain and test a code template for the course assessment feature. You have efficiently implemented these features, ensuring requirements alignment while maintaining good project organization and documentation.

Author(s)

Muhammad Yahya