Deploy, Update, and Scale Microservices with Serverless Framework in Python

Estimated Time: 30 minutes

In this lab, you will learn how to create a UK University information application composed of two microservices on Code Engine, and how to scale and update each microservice independently of the other. IBM Cloud Code Engine has been made available to you through this lab environment.

> This application uses a JSON file which is obtained from this link, filtering out on universities in the UK.

Learning Objectives:

After completing this lab you will be able to:

1. Start Code Engine service to create an application
2. Use the code engine service to deploy two microservices that are part of an application.
3. Scale one of the microservices to have more than one instance.
4. Update one of the microservices without having to stop and redeploy it.

Deploying the microservices

1. Go to the Code engine CLI terminal. If you don't have one, click below to set it up.

▼ Click here to see how to set up one

1. On the menu in your lab environment, Click the `Cloud` dropdown and choose `Code Engine`. The code engine setup panel comes up. Click `Create Project`.

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 setup is complete, you can see that it is active. Click on `Code Engine CLI` to begin the pre-configured CLI in the terminal below.

4. You will observe that the pre-configured CLI starts up and the home directory is 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.

2. In the Code Engine CLI, clone the repository https://github.com/ibm-developer-skills-network/gentu-microservices_practiceProj.git by running the following command.

```
git clone https://github.com/ibm-developer-skills-network/gentu-microservices_practiceProj.git
```

3. Change to the `universities` directory in the project that you just cloned.

```
cd gentu-microservices_practiceProj/universities
```

4. List to see the directories inside it each of which have the microservices that we will deploy on Code Engine.

```
ls
```

You will see two directories listed.

- listing
- websites

The `listing` directory contains the microservice that will list all universities or a list of universities based on keyword search.

The `websites` directory contains the microservice that will provide the list of websites given a college name. You will later update the application to return websites for all colleges that match a keyword search.

5. Run the following command to deploy the `listing` as a microservice.

```
ibmcloud ce app create --name listing --image us.icr.io/${SN_ICR_NAMESPACE}/listing --registry-secret icr-secret --port 5000 --build-context-dir listing --build-source .
```

This takes a while.

6. Now run the following command to get the details about the app. This will also list URL that you can access the endpoints through. Copy the URL.

```
ibmcloud ce app get -n listing
```

7. Now try to access the endpoint you copied previously by replacing it in the following command. This command will fetch all the colleges which have the name `Trinity` in it.

```
curl <your deplymenturl>/colleges/Trinity
```

This will take a while as no instances are running.

8. Now get the details of the deployment again.

```
ibmcloud ce app get --name listing -q
```

You can see that the instances are listed now as you just accessed the endpoint. But this instance will be terminated again when there are no more requests. You can verify the same by running the same command after a gap of 5 mins.

9. Run the following command to deploy the `websites` as a microservice.

```
ibmcloud ce app create --name websites --image us.icr.io/${SN_ICR_NAMESPACE}/websites --registry-secret icr-secret --port 5000 --build-context-dir websites --build-source .
```

This takes a while and once it successfully deploys, you will get a URL that you can access the endpoints through. Copy the URL.

10. Now try to access the endpoint you copied previously by replacing it in the following command. This command will fetch all the websites for the college `Trinity College of Music`.

```
curl <your deplymenturl>/websites/Trinity%20College%20of%20Music
```

%20 is URL encoding for blank space.

11. There is no keyword search for the websites. Only if the college name exactly matches the value that is passed, the websites are fetched. Try to access the endpoint you copied previously by replacing it in the following command. This command will fetch no websites.

```
curl <your deplymenturl>/websites/Trinity
```

It will return nothing.

Scale and update the microservices in the application

Unlike a monolithic application, having the application composed of multiple microservices is very advantageous. It allows us to scale the microservices and/or update them independently without affecting the whole application, having to redeploy it or restart it. For example, if one service is seeing a lot of hits then we can scale just that one.

In this example, we will imagine two scenarios:

1. The `listings` API endpoint gets a lot of hits and has to be scaled.
2. After creating the `websites` API endpoint, based on user experience, you decide to offer a keyword search instead of an exact word search.
3. Run the following command to scale the `listing` application to three instances.

```
ibmcloud ce app update --name listing --min 2
```

This will scale the minimum number of instances to 2. At any point in time, there will be 2 instances of the application running.

2. Run the following command to see if the `listing` application has scaled.

```
ibmcloud ce app get --name listing -q
```

Wait for the application to scale and all instances to run.

3. Now when you run the `curl` command to access the endpoint, it will be much faster as the instance is already running 2 instances.

Updating the microservice

You will now update the microservice serving `websites` API Endpoint to take keyword search as a parameter instead of giving the whole college name.

1. Click on the button below to open `app.py` in the websites directory.

Open **app.py** in IDE

2. Replace the implementation of the API endpoint, with the following code. This will enable the microservice to provide keyword searches for websites.

The updated code should be as below.

```
from flask import Flask
from flask_cors import CORS
import json
app = Flask("University Websites")
CORS(app)
with open("UK_Universities.json", "r") as unifile:
    data = json.load(unifile)
@app.route("/websites/<name>")
def getCollegesWebsites(name):
    webpages = []
    for college in data:
        if college["name"].__contains__(name):
            for website in college["web_pages"]:
                webpages.append(website)
    return json.dumps({"Webpages": webpages},indent=4)
if __name__ == "__main__":
    app.run(debug=True, port=5000)
```

3. Run the following command to update the application.

```
ibmcloud ce app update --name websites --image us.icr.io/${SN_ICR_NAMESPACE}/websites --registry-secret icr-secret --port 5000 --build-context-dir websites --build-source .
```

4. This will update the existing application. To verify the same run the following command. You may recollect from the previous exercise that this did not return any websites as the previous code did not support keyword search.

```
curl <your deplymenturl>/websites/Trinity
```

You will see the websites of all colleges which have `Trinity` in their name listed.

Practice Exercise:

1. Scale the `websites` microservice to have 2 minimum instances.

   NOTE: Before scaling the website service, check the number of instances currently running for both the listing and website services. If 3–4 instances are running, wait approximately 5 minutes, as these instances will be terminated when there are no active requests. You can verify this by running the following command after the 5-minute wait. Begin scaling only when there are 2 or fewer instances to ensure sufficient memory is available for the new instances.

   ```
   ibmcloud ce app get --name websites -q
   ```

2. Update the `listings` microservices to retrieve the list of universities in a case-insensitive manner. Verify the same passing “TriniTY”, as a parameter. If successfully updated, the endpoint should return all the universities with the work ‘Trinity’.

Congratulations! You have completed this lab successfully and deployed your application composed of two microservices on Code Engine, scaled on microservice, and updated the other microservice.

Author(s)

Lavanya T S