A PRACTICAL EXPLORATION OF CLOUD FOUNDRY AND CONCOURSE

AIM

To learn new technology, the most efficient way is to practise it. The aim of this practical exploration is to publish a sample .net web application to Cloud Foundry via Concourse, which is also a Pivotal product.

CREATE A SAMPLE .NET WEB PROJECT

To start the exploration, we need a .net web project with some unit tests. I created two projects, one called Concourse Api which uses the default web project template. The other was called Concourse.Test using the unit test template.

CONCOURSE CI

Concourse CI is a CI/CD solution built by Pivotal. It integrates well with other Pivotal products. However, Concourse is also a stand-alone product which means you can also use it on other platforms. You can think of Concourse as a Jenkins alternative or competitor.

INSTALL CONCOURSE

Before installing Concourse, please make sure your machine/environment has the following tools installed:

1. Docker
2. Docker Compose

Download the docker compose file and navigate to the file location and run the following command:

Spin up a container

```
docker-compose up -d
```

Using this command, the Docker-Compose file should be called **"docker-compose.yml"**, which is the default file name of the docker-compose command configuration file. Otherwise, you should use the following command to notify the tool where the configuration file is located:

Spin up a container with specific configuration file

```
docker-compose -f docker-compose-quickstart.yml up -d
```

After applying the command, you can navigate to http://localhost:8080 to check your installation.

Note
In this step, we only need to ensure the Concourse is successfully installed. You don't need to log in into the server at this stage.

Besides installing the Concourse server on your machine, you also need to install two CLI tools, **fly** and **concourse**. You can download these tools here.

BASIC CONCOURSE COMMANDS

To start with learning Concourse, I recommend you read this tutorial which comes highly recommended by Concourse. Here, I only list some common usages. **Fly** CLI will be used to manipulate the Concourse server.

Login to the Concourse server with fly CLI

```
fly -t poc login -c http://localhost:8080
```

poc is a target variable which the user can define, thus you can assign any value which fit into your scenario.

The first time you execute this command you also need to ensure the fly CLI tool has the same version as Concourse. The following command also needs to be applied if it is your first time logging in:

Ensure fly CLI have the same version as Concourse

```
fly -t poc sync
```

Run a task on Concourse with fly CLI

For a CI/CD server, to run a task is a basic operation. Running a task on Concourse, a docker container will be created to execute the command. This architecture gives developers peace of mind on the task execution environment. There is low overhead cost and good performance when spinning up a virtual environment in docker, which also means the concourse server doesn't need to utilise too many resources.

hello_world.yml

```
platform: linux

image_resource:
  type: docker-image
  source: {repository: busybox}

run:
  path: echo
  args: [hello world]
```

As this is a YAML file, we can see a Linux container will be created and we will execute a simple command: **echo hello world**.

To run this simple task on Concourse, we can use the following command with fly:

Run a simple task on Concourse with fly

```
fly -t poc e -c task_hello_world.yml
```

Set up a pipeline on Concourse with fly

The pipeline is also described by a YAML file. A simple CI/CD pipeline normally will have two resources. One is your code repository and the other is the delivery destination. In this example, the code repository is a GIT repository and the delivery destination is Pivotal Cloud Foundry. Multiple tasks will be executed when the pipeline is trigged. These operations include Build, Test, Release.
To use this pipeline configuration, you need to have a Pivotal Cloud Foundry account and put your **username** and **password** in the configuration.

pipeline.yml

```
resources:
- name: code-source
  type: git
  source:
      uri: https://github.com/skyline9002/PivotalPoc
      branch: master
- name: pcf
  type: cf
  source:
      api: https://api.run.pivotal.io
      skip_cert_check: false
      username: xxxxxxx
      password: xxxxxxx
      organization: pivotal-simon-huang
      space: development

jobs:
-  name: aspnetcore-unit-tests
   plan:
      - get: code-source
      trigger: true
      - task: run-tests
      privileged: true
      config:
            platform: linux
            inputs:
            - name: code-source
            image_resource:
            type: docker-image
            source:
            repository: microsoft/aspnetcore-build
            run:
            path: sh
            args:
            - -exc
            - |
            cd ./code-source/Concourse.Test
            dotnet restore
            dotnet test

-  name: deploy-to-prod
   plan:
      - get: code-source
      trigger: true
      passed: [aspnetcore-unit-tests]
      - put: pcf
      params:
      manifest: code-source/manifest.yml
```

In the deploy-to-prod task, we can see the pipeline notify Pivotal Cloud Foundry to provision resources and deploy the application. The **manifest.yml** is shown as follows:

manifest.yml

```
applications:
- name: dotnet-poc-simonhuang
  memory: 256M
  instances: 1
  buildpack: dotnet_core_buildpack
  path: /tmp/build/put/code-source/Concourse.Api
```

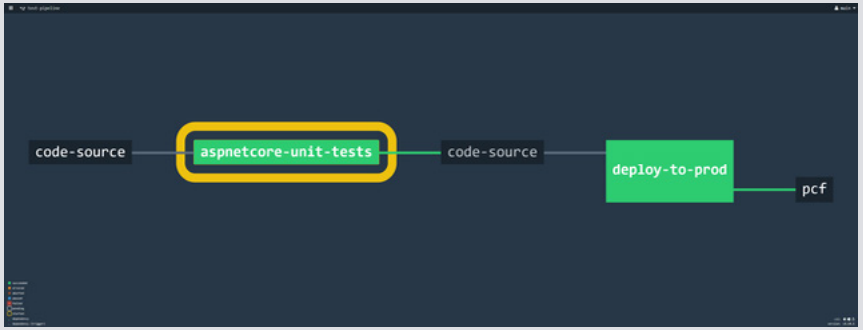
Note
Please note your manifest.yml should be in the root directory of your code repository. Otherwise, please modify to the correct path in your manifest.yml
To set up this pipeline in your Concourse server, namely your local machine in this example, use the following command.

Setup a pipeline on Concourse via fly

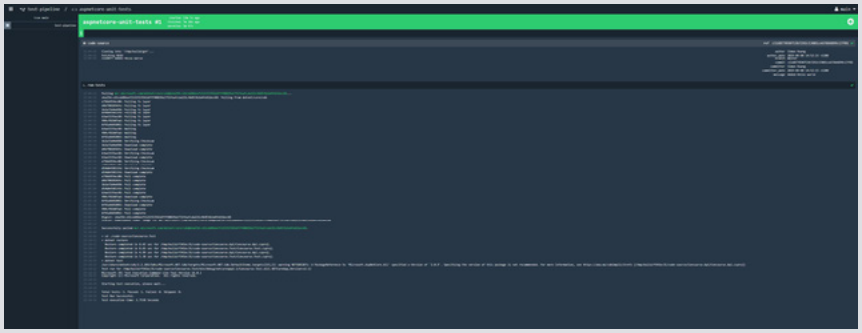
```
fly -t poc set-pipeline -c pipeline.yml -p test-pipeline
```

Then you can check your pipeline from: http://localhost:8080/teams/main/pipelines/test-pipeline

You can see the overall pipeline status from each step. A green colour means the step ran successfully, yellow highlighted means the step is running and red means there have been some blockers in the step. Also, for further details of the task running, you can click on the step status to check the log.



This screenshot shows the pipeline established successfully and running



This screenshot showed the log of details execution. We can see unit tests executed in the pipeline and passing.

FINAL STEPS

As a final step of the exercise, you should see your pipeline tasks running with a cheerful green colour. Up to this point, you have successfully deployed a .net application to Cloud Foundry with Concourse. Furthermore, you can log in to your Pivotal account to check your running instance. The Pivotal platform will assign a random public URL to your running instance which is the entry to your web application. Also, when you try to push some modified code into the master branch, you will see the pipeline triggered again. Woo hoo!

@AssurityConsulting

@AssurityNZ

Assurity Consulting Ltd

assurity.nz

AUCKLAND

Level 6
22 Fanshawe Street
PO Box 106 949
Auckland 1143

t. (64) 9 354 4901

WELLINGTON

Level 6, Harbour Tower
2 Hunter Street
PO Box 25 440
Wellington 6140

t. (64) 4 473 0901

CHRISTCHURCH

Level 2
53 Victoria Street
PO Box 25 443
Christchurch 8144

t. (64) 3 379 9146

assurity