CLOUD FOUNDRY

Pivotal Cloud Foundry Developer

Lab Instructions

Application Deployment using Pivotal Cloud Foundry

Version 1.11.a

Pivotal

Copyright Notice

Table of Contents

Chapter 1. Manifest

Estimated Time: 10 minutes

1.1. Preface

You can imagine how tedious things would be if we had to repeatedly pass a handful of command arguments to the cf cli each time we push an application. There's a better way: application manifests allow you to consolidate into a single file all of our application's deployment metadata, including the application name, memory, path, hostname, and much more. We can associate services and environment variables to our application in a single manifest file.

Don't forget to take the time to read through the excellent documentation on manifests that's referenced below. There are lots of manifest file attributes you'll want to learn about. If you're new to yaml, take a little time to familiarize yourself with the syntax.

In this lab you'll reverse-engineer an application manifest file from one of the running applications you deployed in the last lab. That's a pretty convenient feature.

1.2. Reverse-Engineer a manifest from a running application

1. Read about application manifests

2. Change directories to the `attendee-service` application.

   Example:

```
Unix, Mac: cd ~/pivotal-cloud-foundry-developer-workshop/articulate/
Windows:   cd C:\pivotal-cloud-foundry-developer-workshop\articulate\
```

3. Generate a manifest.

```
cf create-app-manifest attendee-service -p ./manifest.yml
```

4. Edit the manifest.

   - Increase the `instances` to `2`.

   - Add a manifest attribute entry to set the path to the application's artifact (`attendee-service-0.1.jar`).

5. Push `attendee-service`.

```
cf push
```

6. Confirm the `attendee-service` is running and the latest push parameters are reflected.

```
cf app attendee-service
```

1.2.1. Questions

- Assuming you have application running with a bound service (MySQL). You `push` the application again with a manifest, but said service is not specified in the manifest. Is the service still bound to the application?
- What are some `cf` commands that don't have to be explicitly made when using manifests?
- Why did the `create-app-manifest` command not write the `path` attribute for me? In other words, why did I have to specify it manually?

1.2.2. Cleanup

1. Edit the manifest.
   - Decrease the `instances` to `1`
2. Push `attendee-service`.

```
cf push
```

1.3. Create a manifest file by hand

Now that we have a manifest file for the `attendee-service` application, it'd be nice to also have one for `articulate`.

1. Navigate to the `articulate` application directory:

```
cd .../pivotal-cloud-foundry-developer-workshop/articulate/
```

2. This time, try to write the manifest by hand. It's good practice, especially if you're not familiar with the yaml format. Remember, yaml files are indented with two spaces, and should not have any tabs.

   Use the manifest documentation as a reference.

- Review the various manifest property options. Which should you specify?
- You'll likely need to set the application name, path, what else?
- Also, make sure to specify the service binding directly in the manifest. That will save you the step of having to manually `bind-service` if you ever redeploy the application from scratch, or to a new space.

3. Once you're satisfied with your `manifest.yml`, test it with a `cf push` and see if it properly deploys your application as before.

There's much more to manifest files. You can..

- define multiple applications in a single manifest file
- setup one manifest file per environment, and use the -f flag with the `cf push` command to specify which to use
- there's even a mechanism for manifest files to inherit settings from another file

Congratulations! You've completed this lab.