CLOUDFOUNDRY

Pivotal Cloud Foundry Developer

Lab Instructions

Application Deployment using Pivotal Cloud Foundry

Version 1.11.a

Pivotal

Copyright Notice

Pivotal

Table of Contents

Chapter 1. Services

Estimated Time: 25 minutes

1.1. Preface

Up until now, we've focused on the deployment of applications. But as we know, applications often use backing services: databases, message brokers, and other applications' services (to name a few).

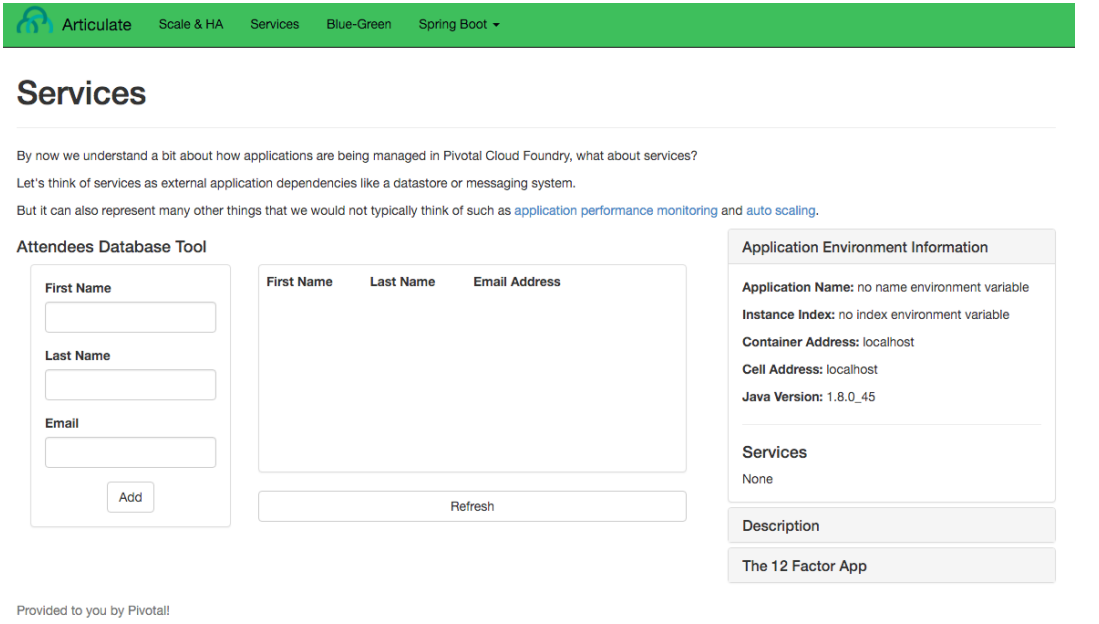
In this lab, you'll have the opportunity to experiment with both managed services and user-provided services: you'll create a mysql backing service to persist information about attendees, and you'll create a user-provided service to allow one application to consume the services of another without hard-coding its route.

In the process, you'll learn new cf commands: `create-service`, `create-user-provided-service`, and `bind-service`. Don't forget: `cf help` is there to help us understand how to invoke each command.

1.2. Exercises

1.2.1. Review `articulate` dependencies

`articulate` exposes functionality to add attendees on the *Services* page. However, `articulate` doesn't do this alone. It makes REST calls to the `attendee-service` application. To learn more about services, let's provision the `attendee-service` application.



1.2.2. Push the `attendee-service` application

1. The attendee-service JAR, `attendee-service-0.1.jar` is in you lab files under its own directory. If so, jump to step (3)

2. Alternatively, you can Download the `attendee-service` application, create `pivotal-cloud-foundry-developer-workshop/attendee-service/` directory and copy the JAR there.

> **Note**
>
> If your browser warns you about downloading this file please proceed to download it.

3. The application Source is not required, but you may be curious how it works as you move through the labs.

4. Push the `attendee-service` application.

```
$> cd .../pivotal-cloud-foundry-developer-workshop/attendee-service/
$> cf push attendee-service -p ./attendee-service-0.1.jar -m 768M --random-route
```


Does `attendee-service` start up correctly? Why not?

1.2.3. Add a Managed Service

1. Review the documentation on managing service instances
2. Review what services are available in the marketplace.

```
cf marketplace
```

3. There exist two distinct Cloud Foundry managed services that provision MySQL databases:
   a. p-mysql is a Pivotal product, and
   b. cleardb is a company that specializes in cloud-based MySQL instances

   Option A: Pivotal Web Services

   Pivotal Web Services offers the *cleardb* managed service in its marketplace.

```
cf create-service cleardb spark attendee-mysql
```

   Option B: Pivotal Cloud Foundry

   If working in a Pivotal Cloud Foundry environment that is not PWS, you are likely to have the *p-mysql* managed service.

```
cf create-service p-mysql 100mb attendee-mysql
```

4. Now we need to bind the application with the service instance.

```
cf bind-service attendee-service attendee-mysql
```

Note

You can ignore the *TIP: Use 'cf restage attendee-service' to ensure your env variable changes take effect* message at this time.

5. Restart the application.

```
cf restart attendee-service
```


6. View the `attendee-service` in a browser.

   You should see a response similar to the following (the output in the screenshot is rendered by the JSON Formatter for Chrome):


```
attendee-service-surfy-glt.pcfi1.fe.gopivotal.com
{
  "_links": {
    "attendees": {
      "href": "http://attendee-service-surfy-glt.pcfi1.fe.gopivotal.com/attendees{?page,size,sort}",
      "templated": true
    },
    "profile": {
      "href": "http://attendee-service-surfy-glt.pcfi1.fe.gopivotal.com/profile"
    }
  }
}
```


   The root endpoint of this application is rather uninteresting. Instead, visit the `/attendees` endpoint. An http GET to the `attendees` endpoint will fetch all attendees in the database and display them in JSON format. Note that in the response, the `totalElements` is currently 0.

   This application implements a RESTful API. This means that you should be able to submit http POST to the same `attendees` endpoint with a body containing the JSON representation of the `Attendee` model type to create such a record. This can be done programmatically, or via REST client tools such as Postman or command-line tools such as curl or httpie. Here's an example using httpie:

```
http post {{attendees_app_uri}}/attendees firstName=Eitan lastName=Suez emailAddress=esuez@pivotal.io
```

How does this work?

1. Read about how twelve-factor apps handle backing services and configuration.
2. Read about VCAP_SERVICES.
3. View the environment for `attendee-service`.

```
cf env attendee-service
```

4. Different languages/frameworks will have various ways to read environment variables. `attendee-service` takes advantage of a Java Buildpack feature called Auto-Reconfiguration that will automatically re-write bean definitions to connect with services bound to an application.

1.2.3.1. Questions

- After binding a service to an application why is the application restarted/restaged?
- In this case, why could we `restart` vs `restage`?

1.2.4. Add a User-Provided Service Instance

In the enterprise, not all services will be provisioned by Pivotal Cloud Foundry.

For example, consider your Oracle RAC cluster. How can we connect our applications running on Pivotal Cloud Foundry to these external systems?

Additionally, how can we easily connect applications together running on the platform?

`articulate's` default configuration for the `attendee-service uri` is `http://localhost:8181/` . The subsequent steps will allow you to override the default configuration with your own.

1. Read about user-provided service instances.
2. Create a user-provided service instance.

```
cf create-user-provided-service attendee-service -p uri
```

   This will create an interactive prompt. For the value of `uri`, enter **your `attendee-service` application**'s base url:

```
uri> https://{{attendees_app_uri}}/
```

3. Bind `articulate` to the `attendee-service` user-provided service.

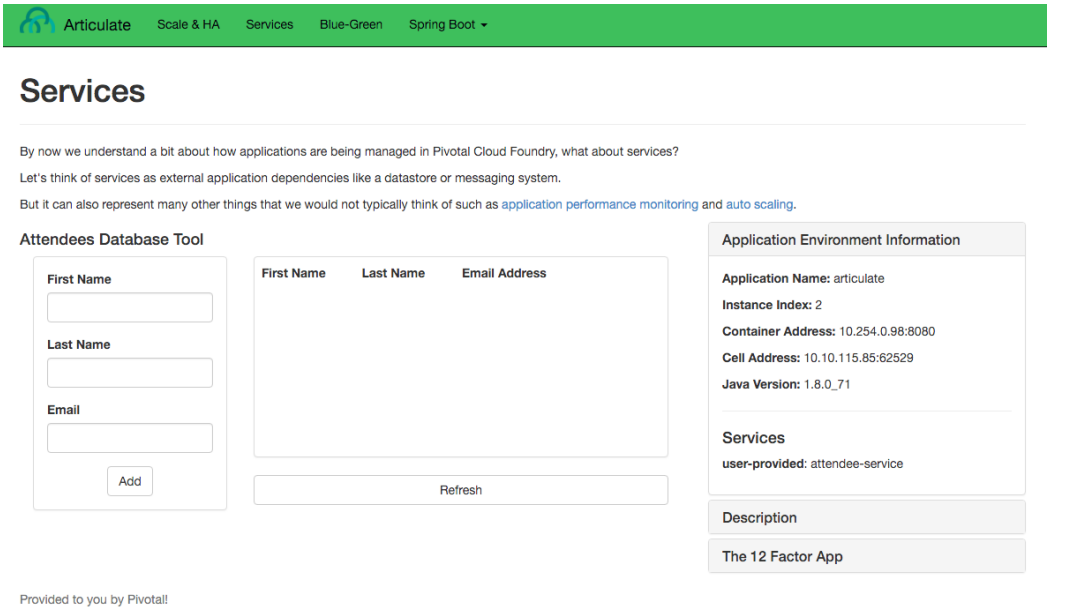
```
cf bind-service articulate attendee-service
```

> **Note**
>
> You can ignore the *TIP: Use 'cf restage articulate' to ensure your env variable changes take effect* message at this time.

4. Restart the application.

```
cf restart articulate
```


5. Refresh the `articulate` *Services* page. You can now see the `attendee-service` listed under `Services`.



6. Review the environment.

```
cf env articulate
```

7. Add some attendees.

Note

If you can't add attendees, review the `articulate` logs and the user-provided service instance configuration.

1.2.4.1. Questions

- From an application perspective, are managed services instances different from user-provided service instances?

1.3. Beyond the class

- Use Spring Music and a User Provided Service Instance to connect to a database (MySQL or Oracle) in your environment.

