



CLOUD **FOUNDRY**

Pivotal Cloud Foundry Developer

Lab Instructions

Application Deployment using Pivotal Cloud
Foundry

Version 1.11.a

Pivotal

Copyright Notice

- Copyright © 2017 Pivotal Software, Inc. All rights reserved. This manual and its accompanying materials are protected by U.S. and international copyright and intellectual property laws.
- Pivotal products are covered by one or more patents listed at <http://www.pivotal.io/patents>.
- Pivotal is a registered trademark or trademark of Pivotal Software, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies. The training material is provided “as is,” and all express or implied conditions, representations, and warranties, including any implied warranty of merchantability, fitness for a particular purpose or non-infringement, are disclaimed, even if Pivotal Software, Inc., has been advised of the possibility of such claims. This training material is designed to support an instructor-led training course and is intended to be used for reference purposes in conjunction with the instructor-led training course. The training material is not a standalone training tool. Use of the training material for self-study without class attendance is not recommended.
- These materials and the computer programs to which it relates are the property of, and embody trade secrets and confidential information proprietary to, Pivotal Software, Inc., and may not be reproduced, copied, disclosed, transferred, adapted or modified without the express written approval of Pivotal Software, Inc.

Table of Contents

1. Services	1
1.1. Preface	1
1.2. Exercises	1
1.2.1. Review articulate dependencies	1
1.2.2. Push the attendee-service application	2
1.2.3. Add a Managed Service	3
1.2.4. Add a User-Provided Service Instance	5
1.3. Beyond the class	6

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.

Let's think of services as external application dependencies like a datastore or messaging system.

But it can also represent many other things that we would not typically think of such as [application performance monitoring](#) and [auto scaling](#).

First Name	Last Name	Email Address
John	Doe	john.doe@example.com
Jane	Smith	jane.smith@example.com
Michael	Johnson	michael.johnson@example.com
Sarah	Williams	sarah.williams@example.com
David	Brown	david.brown@example.com
Emily	Miller	emily.miller@example.com
James	Wilson	james.wilson@example.com
Alice	Moore	alice.moore@example.com
Robert	Taylor	robert.taylor@example.com
Olivia	Anderson	olivia.anderson@example.com
William	Thomas	william.thomas@example.com
Isabella	Clark	isabella.clark@example.com
Benjamin	White	benjamin.white@example.com
Mia	Green	mia.green@example.com
Ethan	Black	ethan.black@example.com
Ava	Gray	ava.gray@example.com
Noah	Red	noah.red@example.com
Charlotte	Blue	charlotte.blue@example.com
Liam	Orange	liam.orange@example.com
Amelia	Purple	amelia.purple@example.com
Lucas	Yellow	lucas.yellow@example.com
Sophia	Pink	sophia.pink@example.com
Mason	Green	mason.green@example.com
Evelyn	Blue	evelyn.blue@example.com
Logan	Orange	logan.orange@example.com
Aria	Purple	aria.purple@example.com
Sebastian	Yellow	sebastian.yellow@example.com
Madison	Pink	madison.pink@example.com
Carter	Green	carter.green@example.com
Isabella	Blue	isabella.blue@example.com
Wyatt	Orange	wyatt.orange@example.com
Chloe	Purple	chloe.purple@example.com
Grayson	Yellow	grayson.yellow@example.com
Lily	Pink	lily.pink@example.com
Jack	Green	jack.green@example.com
Skylar	Blue	skylar.blue@example.com
Levi	Orange	levi.orange@example.com
Grace	Purple	grace.purple@example.com
Matthew	Yellow	matthew.yellow@example.com
Kennedy	Pink	kennedy.pink@example.com
Nathan	Green	nathan.green@example.com
Alaina	Blue	alaina.blue@example.com
Christopher	Orange	christopher.orange@example.com
Savannah	Purple	savannah.purple@example.com
Andrew	Yellow	andrew.yellow@example.com
Brooklyn	Pink	brooklyn.pink@example.com
Joshua	Green	joshua.green@example.com
Madelyn	Blue	madelyn.blue@example.com
Isaac	Orange	isaac.orange@example.com
Chloe	Purple	chloe.purple@example.com
Grayson	Yellow	grayson.yellow@example.com
Lily	Pink	lily.pink@example.com
Jack	Green	jack.green@example.com
Skylar	Blue	skylar.blue@example.com
Levi	Orange	levi.orange@example.com
Grace	Purple	grace.purple@example.com
Matthew	Yellow	matthew.yellow@example.com
Kennedy	Pink	kennedy.pink@example.com
Nathan	Green	nathan.green@example.com
Alaina	Blue	alaina.blue@example.com
Christopher	Orange	christopher.orange@example.com
Savannah	Purple	savannah.purple@example.com
Andrew	Yellow	andrew.yellow@example.com
Brooklyn	Pink	brooklyn.pink@example.com
Joshua	Green	joshua.green@example.com
Madelyn	Blue	madelyn.blue@example.com
Isaac	Orange	isaac.orange@example.com
Chloe	Purple	chloe.purple@example.com
Grayson	Yellow	grayson.yellow@example.com
Lily	Pink	lily.pink@example.com
Jack	Green	jack.green@example.com
Skylar	Blue	skylar.blue@example.com
Levi	Orange	levi.orange@example.com
Grace	Purple	grace.purple@example.com
Matthew	Yellow	matthew.yellow@example.com
Kennedy	Pink	kennedy.pink@example.com
Nathan	Green	nathan.green@example.com
Alaina	Blue	alaina.blue@example.com
Christopher	Orange	christopher.orange@example.com
Savannah	Purple	savannah.purple@example.com
Andrew	Yellow	andrew.yellow@example.com
Brooklyn	Pink	brooklyn.pink@example.com
Joshua	Green	joshua.green@example.com
Madelyn	Blue	madelyn.blue@example.com
Isaac	Orange	isaac.orange@example.com
Chloe	Purple	chloe.purple@example.com
Grayson	Yellow	grayson.yellow@example.com
Lily	Pink	lily.pink@example.com
Jack	Green	jack.green@example.com
Skylar	Blue	skylar.blue@example.com
Levi	Orange	levi.orange@example.com
Grace	Purple	grace.purple@example.com
Matthew	Yellow	matthew.yellow@example.com
Kennedy	Pink	kennedy.pink@example.com
Nathan	Green	nathan.green@example.com
Alaina	Blue	alaina.blue@example.com
Christopher	Orange	christopher.orange@example.com
Savannah	Purple	savannah.purple@example.com
Andrew	Yellow	andrew.yellow@example.com
Brooklyn	Pink	brooklyn.pink@example.com
Joshua	Green	joshua.green@example.com
Madelyn	Blue	madelyn.blue@example.com
Isaac	Orange	isaac.orange@example.com
Chloe	Purple	chloe.purple@example.com
Grayson	Yellow	grayson.yellow@example.com
Lily	Pink	lily.pink@example.com
Jack	Green	jack.green@example.com
Skylar	Blue	skylar.blue@example.com
Levi	Orange	levi.orange@example.com
Grace	Purple	grace.purple@example.com
Matthew	Yellow	matthew.yellow@example.com
Kennedy	Pink	kennedy.pink@example.com
Nathan	Green	nathan.green@example.com
Alaina	Blue	alaina.blue@example.com
Christopher	Orange	christopher.orange@example.com
Savannah	Purple	savannah.purple@example.com
Andrew	Yellow	andrew.yellow@example.com
Brooklyn	Pink	brooklyn.pink@example.com
Joshua	Green	joshua.green@example.com
Madelyn	Blue	madelyn.blue@example.com
Isaac	Orange	isaac.orange@example.com
Chloe	Purple	chloe.purple@example.com
Grayson	Yellow	grayson.yellow@example.com
Lily	Pink</	

First Name	Last Name	Email Address
<div></div>		
Refresh		

Application Environment Information	
Application Name:	no name environment variable
Instance Index:	no index environment variable
Container Address:	localhost
Cell Address:	localhost
Java Version:	1.8.0_45
Services	
None	
Description	
The 12 Factor App	



- _____

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](#)):



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
```

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


Articulate
Scale & HA
Services
Blue-Green
Spring Boot ▾

Services

By now we understand a bit about how applications are being managed in Pivotal Cloud Foundry, what about services?

Let's think of services as external application dependencies like a datastore or messaging system.

But it can also represent many other things that we would not typically think of such as [application performance monitoring](#) and [auto scaling](#).

Attendees Database Tool

First Name

Last Name

Email

Add

First Name	Last Name	Email Address

Refresh

Application Environment Information

Application Name: articulate

Instance Index: 2

Container Address: 10.254.0.98:8080

Cell Address: 10.10.115.85:62529

Java Version: 1.8.0_71

Services

user-provided: attendee-service

Description

The 12 Factor App

Provided to you by Pivotal!

- Review the environment.

```
cf env articulate
```

- 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.