

# Services

## Table of Contents

Requirements

What you will learn

Exercises

- Review articulate dependencies

- Push the attendee-service application

- Add a Managed Service

  - How does this work?

  - Questions

- Add a User Provided Service Instance

  - Questions

Beyond the class

Estimated Time: 25 minutes



## Requirements

This lab is part of a workshop available at <https://github.com/Pivotal-Field-Engineering/DevNexus2017>  
General Pre-Requisites are available in the Readme.md

In general the Labs should be done in numerical order since they are interdependent.

# What you will learn

---

- How to create a managed service instance
- How to create a user provided service instance
- How to bind an application to a service instance

## Exercises

---

### 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.





## 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 Address
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Add"/>		
<input type="button" value="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

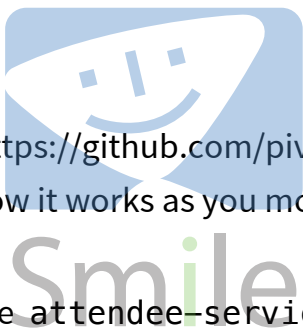
### Description

The 12 Factor App

Provided to you by Pivotal!

## Push the attendee-service application

- 1) Find the (/sample-apps/attendee-service/attendee-service-0.0.1-SNAPSHOT.jar) application. This directory is a subdirectory of the "DevNexus2017" git project that you downloaded in Lab 1.



Source (<https://github.com/pivotal-enablement/attendee-service>) is not required, but you may be curious how it works as you move through the labs.

- 2) Push the attendee-service application.

```
$ cd sample-apps/attendee-service  
$ cf push attendee-service -p ./attendee-service-0.0.1-SNAPSHOT.jar -m 512M --random-route
```

Does attendee-service start up correctly? Why not?

## Add a Managed Service

1) Review the documentation (<http://docs.pivotal.io/pivotalcf/devguide/services/managing-services.html>) on managing service instances

2) Review what services are available in the marketplace.

```
$ cf marketplace
```

3) Provision a MySQL service instance.

### Pivotal Cloud Foundry:

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

### Pivotal Web Services:

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

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

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

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 response similar to the following (pic is using the JSON Formatter for Chrome (<https://chrome.google.com/webstore/detail/json-formatter/bcjindcccaagfpapjjmafapmmgkkhgoa?hl=en>)):



## How does this work?

Smile

1) Read about how twelve-factor apps handle backing services (<http://12factor.net/backing-services>) and configuration (<http://12factor.net/config>).

2) Read about VCAP\_SERVICES (<https://docs.pivotal.io/pivotalcf/devguide/deploy-apps/environment-variable.html#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 (<https://github.com/cloudfoundry/java-buildpack>) feature called Auto-Reconfiguration (<https://github.com/cloudfoundry/java-buildpack-auto-reconfiguration>) that will automatically re-write bean definitions to connect with services bound to an application.

## Questions

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

## 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?

1) Read about user provided service instances

(<http://docs.pivotal.io/pivotalcf/devguide/services/user-provided.html>).

2) Create a user provided service. This will create an interactive prompt. Don't use the literal below for the value of `uri`, use your `attendee-service uri`. Also make sure to specify `http`.

**NOTE:** This will not work with `https`.

```
$cf create-user-provided-service attendee-service -p uri  
uri> http://attendee-service-surfy-glt.pcfi1.fe.gopivotal.com
```

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

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

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`.





## 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 Address
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Add"/>		
<input type="button" value="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!

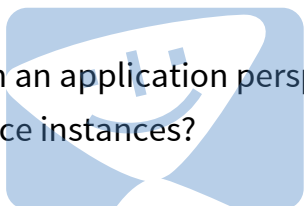
## 6) Review the environment.

```
$ cf env articulate
```

## 7) Add some attendees.

## Questions

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



## Beyond the class

- Use Spring Music (<https://github.com/cloudfoundry-samples/spring-music>) and a User



Provided Service Instance to connect to a database (MySQL or Oracle) in your environment.

---

[Back to TOP](#)

---

© Copyright Pivotal. All rights reserved.

