Pivotal Cloud Platform Deep Dive

Part 4: Custom Buildpacks and Data Services

Ravi Gadhia, rgadhia@pivotal.io
Field Engineering
Los Angeles

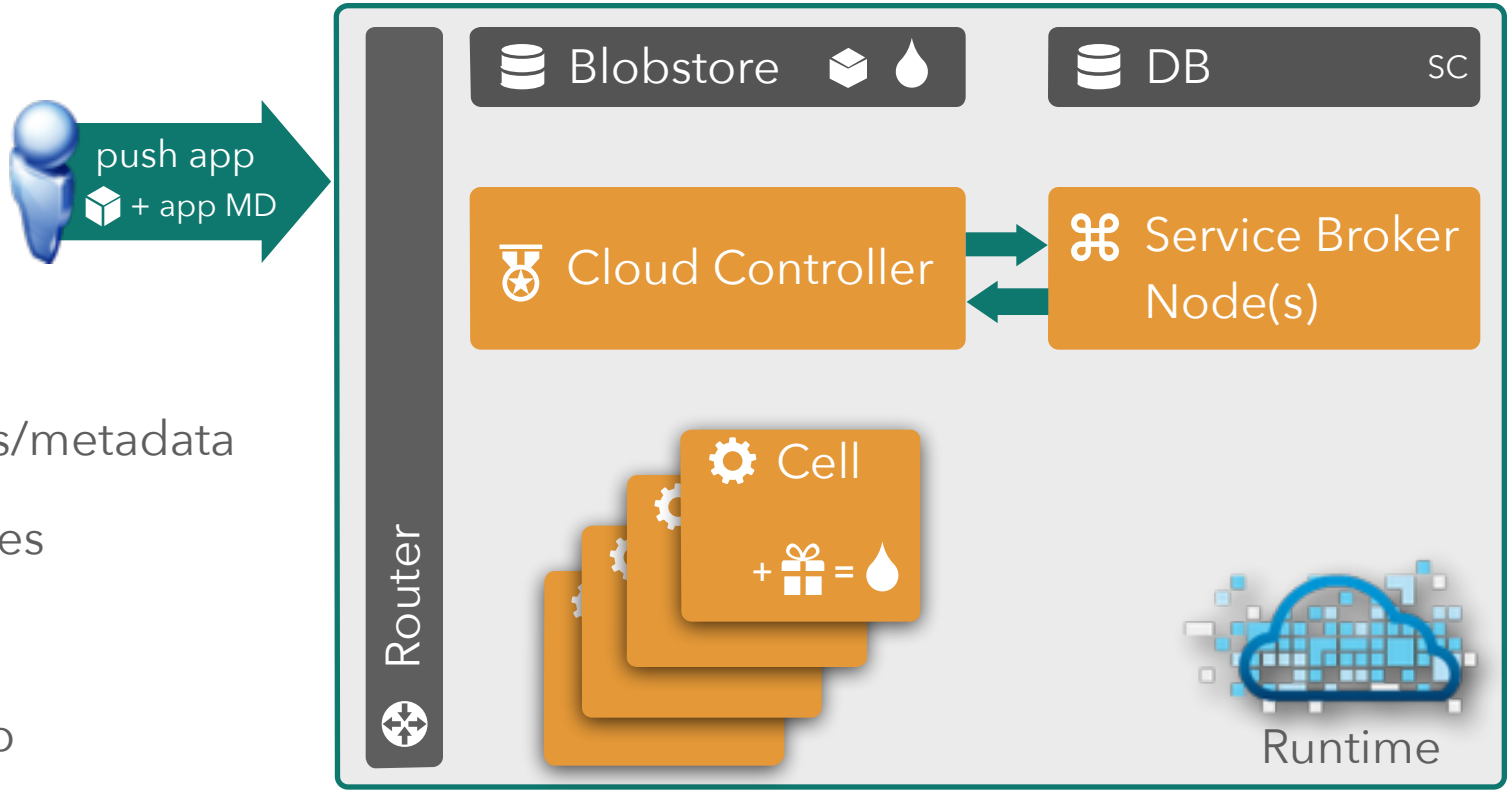
Pivotal

Buildpacks and Services

- Buildpacks
- Java Buildpack Deep Dive
- User-Provided Services
- Managed Services
- DEMO: Deploy and Test HaaSh (HashMap as a Service)

Pivotal

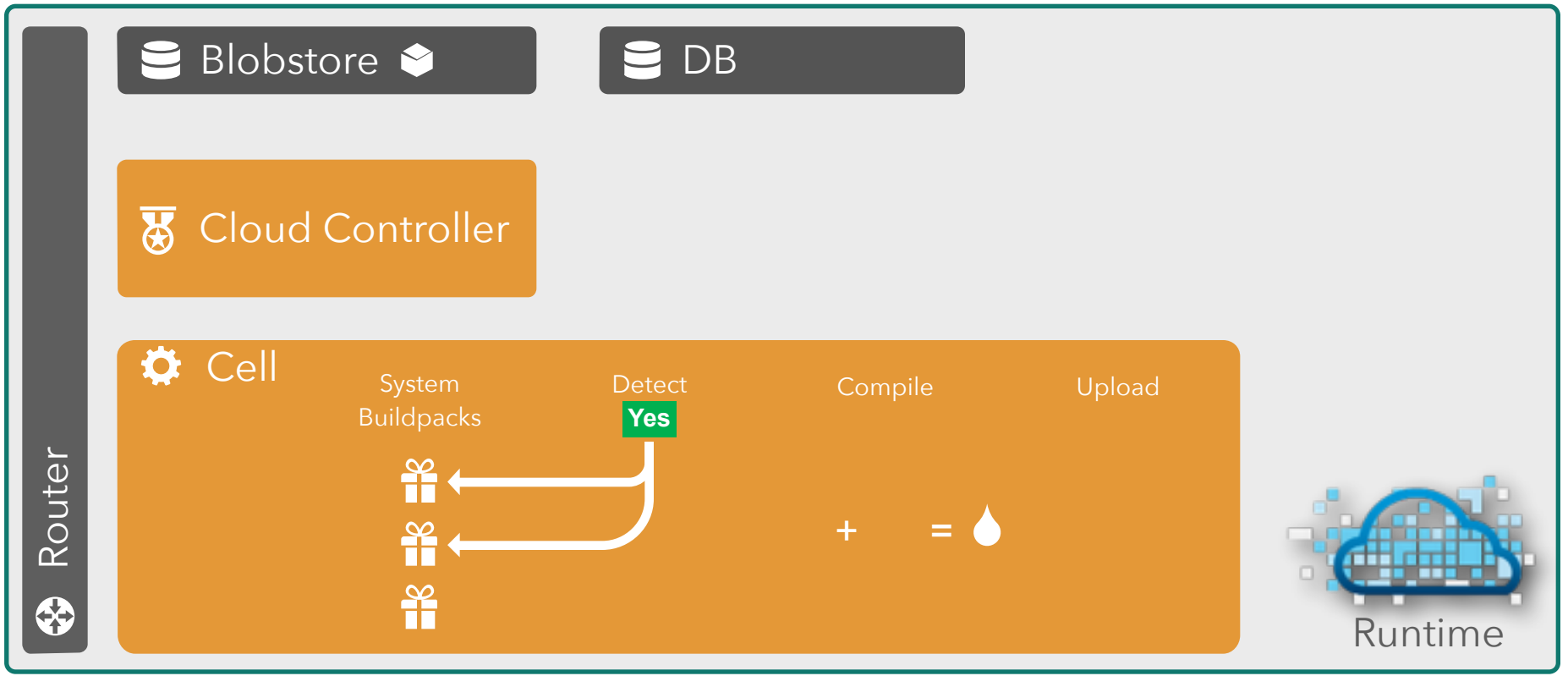
REVIEW: Deploying Applications to Cloud Foundry Runtime



1. Upload bits/metadata
2. Bind services
3. Stage app
4. Deploy app

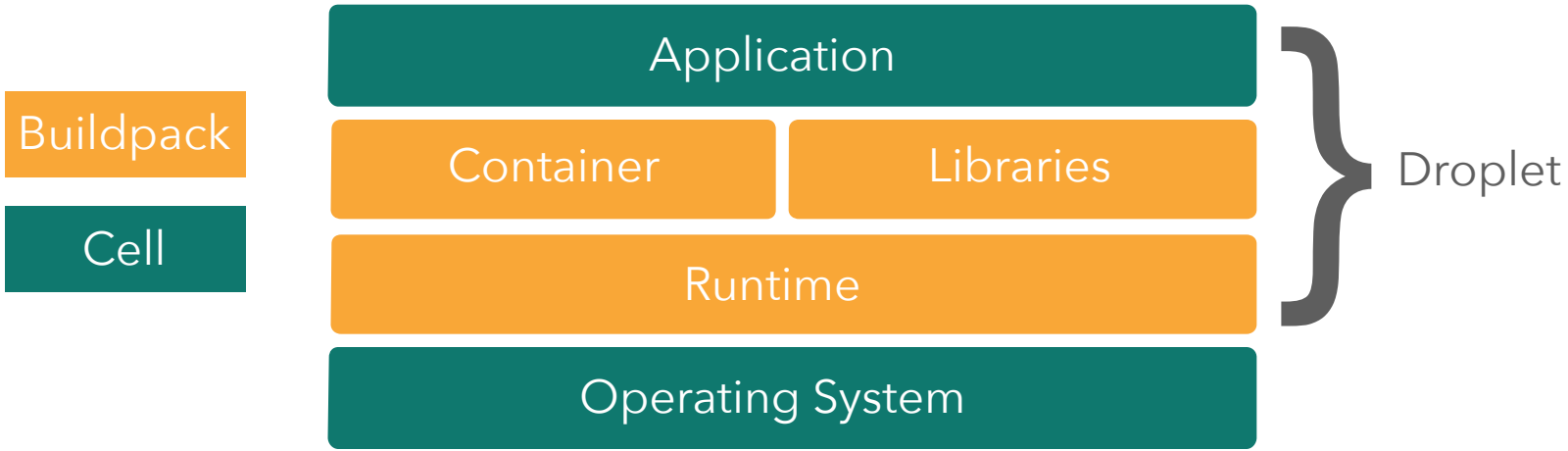
Pivotal

REVIEW: Stage an App



Staging and Buildpacks

Buildpacks are responsible for preparing the machine image for an application.



Types of Buildpacks

System

- deployed with Cloud Foundry

Admin

- uploaded to Cloud Foundry

BYO

- specified at app push

Pivotal

Managing Admin Buildpacks

```
$ cf buildpacks
$ cf create-buildpack <name> <path to bits> <position>
$ cf update-buildpack <name> [-p <path>] [-i <position>]
$ cf delete-buildpack <name>
```

Pivotal

Buildpack selection

`$ cf push`

The application is tested against buildpacks in priority order.

`$ cf push -b <url>`

The buildpack is referenced by a Git URL.

`$ cf push -b <name>`

The buildpack is referenced by name.

Pivotal

Tested Buildpacks

https://github.com/cloudfoundry-community/cf-docs-contrib/wiki/Buildpacks

Containers

Languages

Pivotal

Buildpack API

/bin/detect app_directory

Inspect app bits to determine buildpack applicability

/bin/compile app_directory cache_directory

Download and install runtime, container, packages, libraries; install app bits as necessary

/bin/release app_directory

Build app start command

Pivotal

/bin/detect

Inspect the app bits to determine if the buildpack knows how to handle the application

Ruby	`Gemfile exists`
node.js	package.json exists
python	`setup.py exists`

On match, return exit code 0 and

write to STDOUT a string identifying the buildpack

(often just the name of the language supported)

Pivotal

/bin/detect

$ cf push

Cell iterates over admin and system buildpacks calling

`/bin/detect` scripts

until one of them returns exit code 0

$ cf push -b <url|name>

`/bin/detect` is

not called

Pivotal

/bin/compile

Download and install any necessary

- runtime (Java VM, Ruby interpreter, JavaScript interpreter)
- container (web server)
- support libraries, packages, modules (Ruby gems, NPM packages)

… and then installing the app bits into the runtime or container

Pivotal

/bin/compile Caching

Runtime, container, and support packages are downloaded from sources external to Cloud Foundry

Cell provides a location for storing downloaded artifacts to speed subsequent staging operations

Pivotal

/bin/release

Build a YAML-formatted hash with three possible keys

```
addons: []
config_vars: {}
default_process_types:
  web: <start command>
```

On Cloud Foundry, currently only the web: value is used to get the start command for the app

Pivotal

Buildpacks and Services

- ~~Buildpacks~~
- Java Buildpack Deep Dive
- User-Provided Services
- Managed Services
- DEMO: Deploy and Test HaaSh (HashMap as a Service)

Pivotal

Java Buildpack

Supports a variety of JVM languages, containers, and frameworks with a modular, configurable, and extensible design

Pivotal

Java Buildpack Concepts

Containers

How an application is run

Frameworks

Additional application transformations

JREs

Java Runtimes

Pivotal

Java Buildpack Concepts

Containers

Java `main()`
Tomcat
Groovy
Spring Boot CLI
Play

Frameworks

Spring config
Play config
Play JPA config
New Relic agent
AppDynamics agent

JREs

OpenJDK

Pivotal

Container Detection Criteria

Java `main()`	`META-INF/MANIFEST.MF` exists with `Main-class` attribute set
Tomcat	`WEB-INF` directory exists
Groovy	`.groovy` file with a `main()` method, or `.groovy` file with no classes, or `.groovy` file with a shebang (`#!`) declaration
Spring Boot CLI	one or more POGO `.groovy` files with no `main()` method, and no `WEB-INF` directory
Spring Boot Embedded	`start` script and `lib/spring-boot-*.jar` exist
Play	`start` script and `lib/play.play_*.jar` exist
Ratpack	`start` script and `ratpack-core-*.jar` exist

Choose zero or one

Pivotal

Framework Detection Criteria

Spring	`spring-core*.jar` exists
Play config	Play application detected
Play JPA config	play-java-jpa plugin exists in app
Spring Insight	Insight service bound to app
New Relic agent	New Relic service bound to app
AppDynamics agent	AppDynamics service bound to app

Choose all that apply

Pivotal

/bin/compile Output Example


```
-----> Downloaded app package (18M)
-----> Downloading OpenJDK 1.7.0_21 JRE (17.5s)
       Expanding JRE to .java (1.4s)
-----> Downloading Auto Reconfiguration 0.7.1 (1.4s)
       Modifying /WEB-INF/web.xml for Auto Reconfig
-----> Downloading Tomcat 7.0.42 (3.5s)
       Expanding Tomcat to .tomcat (0.2s)
       Downloading Buildpack Tomcat Support 1.1.1 (0.0s)
-----> Uploading droplet (55M)
```

Cell

Buildpack

Cell

Pivotal

See What's Going On – 'cf files' deprecated

```
$ cf files <app-name> app


.buildpack-diagnostics/
.java/          ——— Buildpack-installed runtime
.lib/           ——— Buildpack-installed support libraries
.tomcat/        ——— Buildpack-installed container
META-INF/
WEB-INF/        ——— Cell-downloaded application files
assets/
```

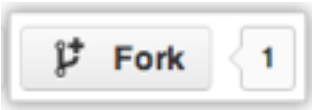
Pivotal

Customization

Two ways to customize the Java buildpack

Configure artifacts used by standard JREs, Containers, and Frameworks

Extend the buildpack with your own JREs, Containers, and Frameworks



Customization is done by forking the buildpack

Pivotal

Customization by Configuration

Configuration files in `java-buildpack/config` determine the behavior of a JRE, Container, or Framework

```
# http://download.pivotal.io.s3.amazonaws.com/openjdk/lucid/x86_64/index.yml
---
1.6.0_27: http://download.pivotal.io.s3.amazonaws.com/openjdk/lucid/x86_64/openjdk-1.6.0_27.tar.gz
1.7.0_21: http://download.pivotal.io.s3.amazonaws.com/openjdk/lucid/x86_64/openjdk-1.7.0_21.tar.gz
1.7.0_25: http://download.pivotal.io.s3.amazonaws.com/openjdk/lucid/x86_64/openjdk-1.7.0_25.tar.gz
1.8.0_M6: http://download.pivotal.io.s3.amazonaws.com/openjdk/lucid/x86_64/openjdk-1.8.0_M6.tar.gz
1.8.0_M7: http://download.pivotal.io.s3.amazonaws.com/openjdk/lucid/x86_64/openjdk-1.8.0_M7.tar.gz
```

```
# cloudfoundry/java-buildpack/config/openjdk.yml
---
version: 1.7.0_+
repository_root: "http://download.pivotal.io.s3.amazonaws.com/openjdk/{platform}/{architecture}"
memory_sizes:
memory_heuristics:
  heap: 0.75
  permgen: 0.1
  stack: 0.05
  native: 0.1
```

Pivotal

Customization by Extension

Implement a JRE, Container, or Framework support class as one Ruby file in the appropriate directory

```
cloudfoundry/java-buildpack/lib/java_buildpack
    └─jre
    └─container
    └─framework
```

(with additional support classes as necessary)

Pivotal

Customization by Extension

Support class types have similar interfaces, following the buildpack scripts naming conventions

```
# initialize the support class with platform information provided in context
# context includes app_dir, lib_dir, environment, java_home, java_opts,
# vcap_application, vcap_services
def initialize(context)

# return a String or an Array<String> that uniquely identifies the container/framework/jre,
# or nil
def detect

# download and unpack the container/framework/jre, and transform the application as necessary
def compile

# create and return the command to run the application with (containers) or add
# options to context[:java_opts] (frameworks)
def release
```

Pivotal

Customization by Extension

Add new support class to config/components.yml

```
---
containers:
  - "JavaBuildpack::Container::Groovy"
  - "JavaBuildpack::Container::JavaMain"
  - "JavaBuildpack::Container::SpringBootCLI"
  - "JavaBuildpack::Container::Tomcat"
  - "JavaBuildpack::Container::PlayFramework"
jres:
  - "JavaBuildpack::Jre::OpenJdk"
frameworks:
  - "JavaBuildpack::Framework::AppDynamicsAgent"
  - "JavaBuildpack::Framework::JavaOpts"
  - "JavaBuildpack::Framework::NewRelicAgent"
  - "JavaBuildpack::Framework::PlayAutoReconfiguration"
  - "JavaBuildpack::Framework::PlayJpaPlugin"
  - "JavaBuildpack::Framework::SpringAutoReconfiguration"
```

Pivotal

Customization

Much more information and documentation included in the GitHub repository

https://github.com/cloudfoundry/java-buildpack

Pivotal

Buildpacks and Services

- ~~Buildpacks~~
- ~~Java Buildpack Deep Dive~~
- User-Provided Services
- Managed Services
- DEMO: Deploy and Test HaaSh (HashMap as a Service)

Pivotal

Use Cases: User Provided Service Instances

- Typically legacy or **existing instances of a service** (databases, queues, mail, etc) where applications connect to the **same instance**
- **Credential passing** when you need to inject the same credential set into an application

Pivotal

```
$ cf cups
FAILED
Incorrect Usage.

NAME:
   create-user-provided-service - Make a user-provided service instance available to cf apps

ALIAS:
   cups

USAGE:
   cf create-user-provided-service SERVICE_INSTANCE [-p PARAMETERS] [-l SYSLOG-DRAIN-URL]

   Pass comma separated parameter names to enable interactive mode:
   cf create-user-provided-service SERVICE_INSTANCE -p "comma, separated, parameter, names"

   Pass parameters as JSON to create a service non-interactively:
   cf create-user-provided-service SERVICE_INSTANCE -p '{"name":"value","name":"value"}'

EXAMPLE:
   cf create-user-provided-service oracle-db-mine -p "host, port, dbname, username, password"
   cf create-user-provided-service oracle-db-mine -p '{"username":"admin","password":"pa55woRD"}'
   cf create-user-provided-service my-drain-service -l syslog://example.com


OPTIONS:
   -p   Parameters
   -l   Syslog Drain Url
```

Buildpacks and Services

- ~~Buildpacks~~
- ~~Java Buildpack Deep Dive~~
- ~~User Provided Services~~
- Managed Services
- Choose Your Own Lab:
  - Customize Java Buildpack
  - Deploy and Test HaaSh (HashMap as a Service)

Pivotal

Managed Services

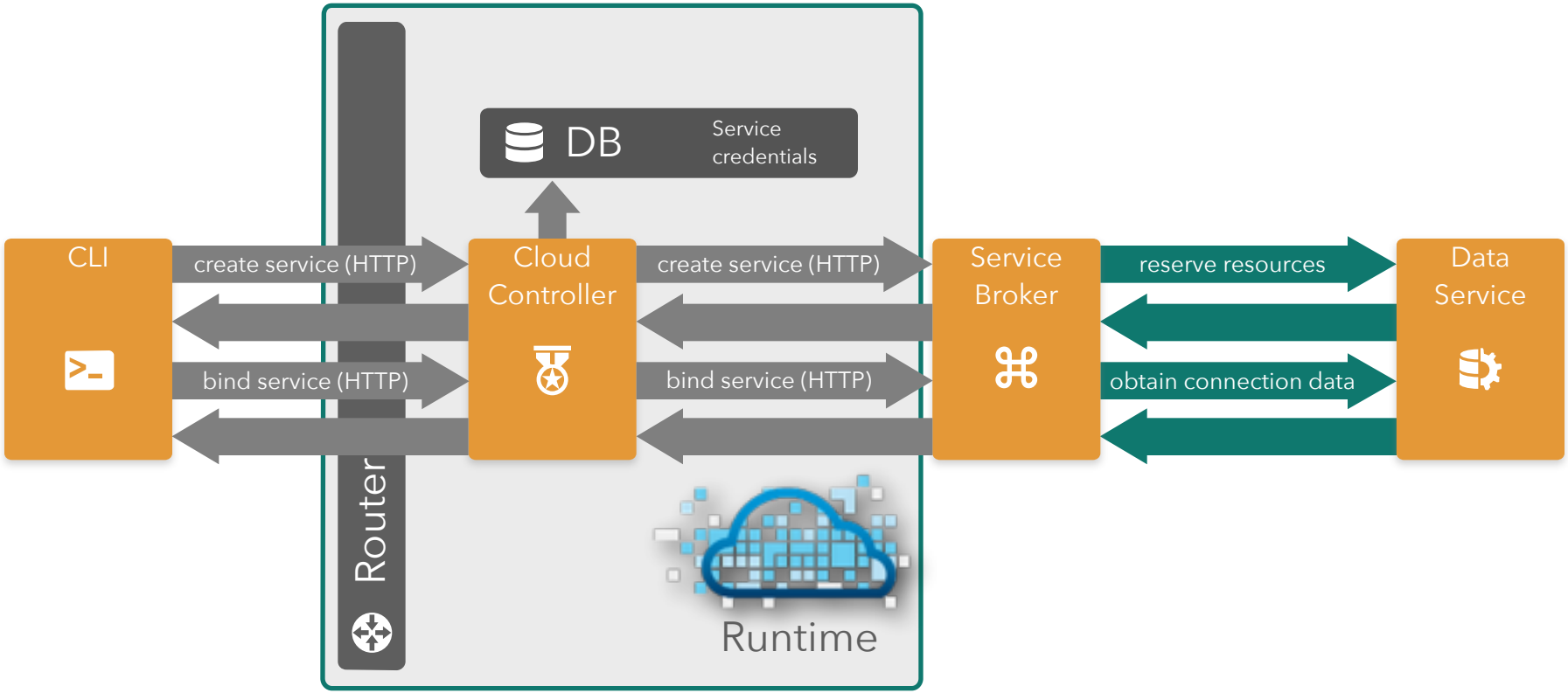
Managed Services are integrated with Cloud Foundry by implementing a documented API for which the cloud controller is the client

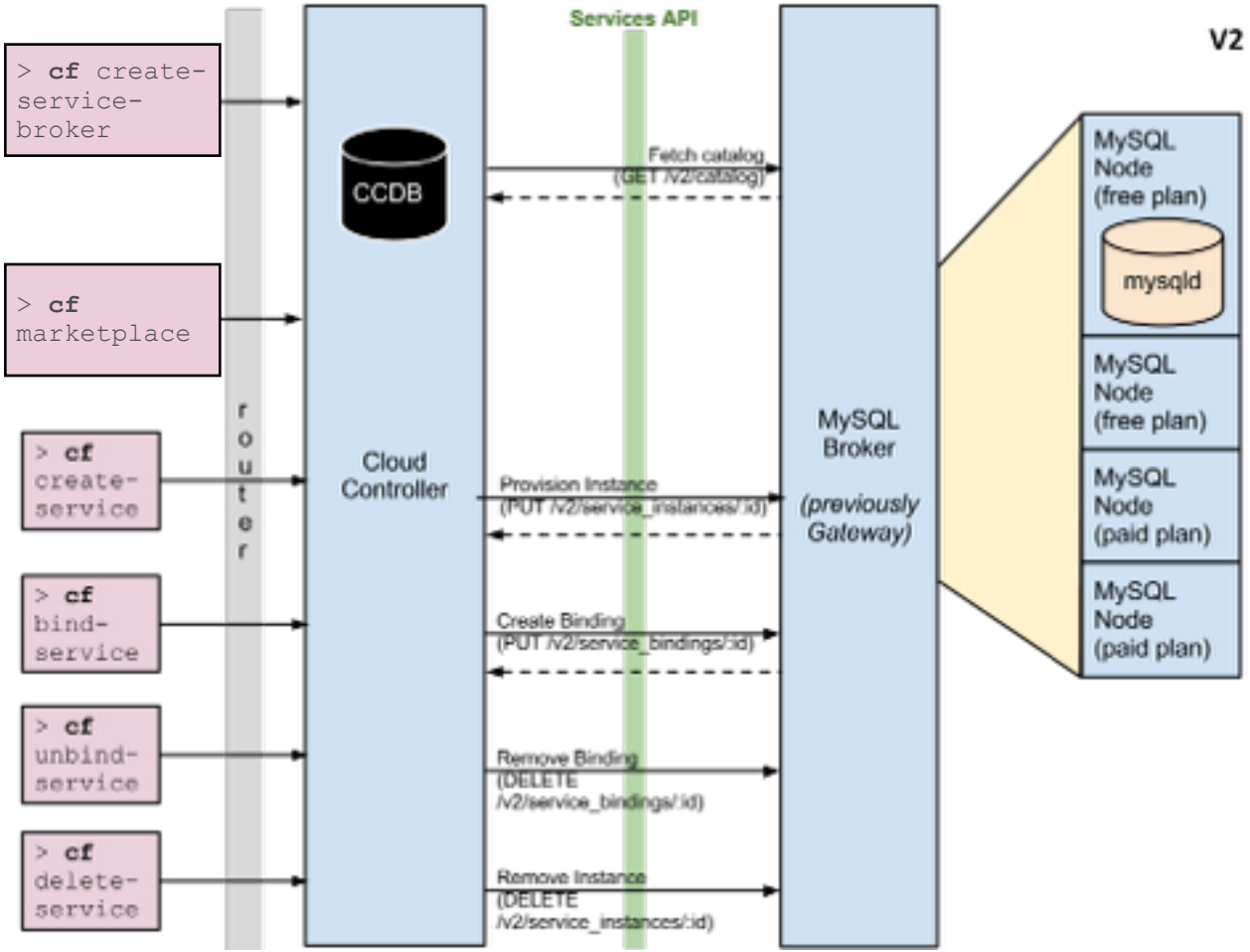
Service Broker is a component which implements the required API.

- Service brokers advertise a catalog of service offerings and service plans to Cloud Foundry, and receive calls from the Cloud Controller for five functions: fetch catalog, create, bind, unbind, and delete.

Pivotal

REVIEW: Create and Bind Services





Pivotal

Server Broker API

- GET /v2/catalog – **Fetch catalog**—services and plans available from this broker.
- PUT /v2/service_instances/:id – **Create** a new service instance.
- PUT /v2/service_instances/:instance_id/service_bindings/:id – Create a new **bind**ing to a service instance.
- DELETE /v2/service_instances/:instance_id/service_bindings/:id – **Unbind** from a service instance.
- DELETE /v2/service_instances/:id – **Delete** a service instance.

Pivotal

Server Broker Registration

- Make the service broker known to the Cloud Controller
  - *cf create-service-broker <broker name> <username> <password> <broker base URI>*
  - Broker should ONLY allow access to those requestors it shared its credential with (Basic Auth)
  - See: http://docs.pivotal.io/pivotalcf/services/managing-service-brokers.html#register-broker
- Make 'plans' accessible to users in a specific org/space
  - Somewhat cumbersome: need to "hand parse" JSON to find service plan UUID as registered with the CC
  - See: http://docs.pivotal.io/pivotalcf/services/access-control.html#make-plans-public

Need admin creds/role in order to introduce a service broker to the Cloud Controller!

Pivotal

Service Broker Implementation

- Service implementation is up to the service provider/developer.
- Cloud Foundry only requires that the service provider implement the service broker API.
- A broker can be implemented as a separate application, or by adding the required http endpoints to an existing service.

Pivotal

Service Broker Implementation

- ***Best Practice***: Each binding is represented by its own *credentials* =>
  - T=1: create service instance
    - ***Neither App-1 or App-2 has access to the service instance***
  - T=2: bind App-1 to service instance
    - ***Only App-1 can access the service instance***
  - T=3: bind App-2 to service instance
    - ***Both App-1 and App-2 have access to the service instance***
  - T=4 unbind App-1 from service instance
    - ***Only App-2 can access the service instance***

Pivotal

Service Instance Provisioning Examples

The result of provisioning varies by service type, although there are a few common actions that work for many services. For a MySQL service, provisioning could result in:

- An empty dedicated mysqld process running on its own VM.
- An empty dedicated mysqld process running in a lightweight container on a shared VM.
- An empty dedicated mysqld process running on a shared VM.
- An empty dedicated database, on an existing shared running mysqld.
- A database with business schema already there.
- A copy of a full database, for example a QA database that is a copy of the production database.
- For non-data services, provisioning could just mean getting an account on an existing system.

Pivotal

Service Broker Deployment Models

Because Cloud Foundry only requires that a service implements the broker API in order to be available to Cloud Foundry end users, many deployment models are possible. The following are examples of valid deployment models:

- Entire service (service backend + broker) packaged and deployed by BOSH alongside Cloud Foundry
- Broker (and optionally service) pushed as an application to Cloud Foundry user space (**this is the approach we'll take in the demo…)**
- Entire service, including broker, deployed and maintained outside of Cloud Foundry by other means

Pivotal

Resources

- Service Broker API: http://docs.pivotal.io/pivotalcf/services/api.html
- Managing Service Brokers: http://docs.pivotal.io/pivotalcf/services/managing-service-brokers.html
- Binding Credentials: http://docs.pivotal.io/pivotalcf/services/binding-credentials.html
- Tiny sample application: https://github.com/cloudfoundry-samples/spring-hello-env

Pivotal

Buildpacks and Services

- ~~Buildpacks~~
- ~~Java Buildpack Deep Dive~~
- ~~User-Provided Services~~
- ~~Managed Services~~
- DEMO: Deploy and Test HaaSh (HashMap as a Service)

Pivotal

THANK YOU!

Pivotal