Cloud Foundry Services

Creating and Binding Services

Making services available to your applications

Pivotal

Roadmap

- **Provisioning Services**
- Using the CLI
- Using the Pivotal CF App Manager Console
- Binding to a Service
- User Provided Services

Pivotal™

Service vs. Service Instance

- **Services** provision **services instances**
  - For example
    - ClearDB service provisions MySQL databases.
    - Offers different plans (fees, SLAs)
  - You may get a dedicated server, or share a multi-tenant server

Pivotal™

Provisioning – Operator View

- Available services depend on CF setup
  - Must be installed and configured by CF Ops
    - Either via Pivotal CF Operator's Console (*Ops Manager*)
    - Using **cf** CLI
    - Or using the BOSH provisioning tool
- Once Ops have deployed a service to your CF instance
  - It appears in the ***marketplace***
  - Can be made available to your application = **provisioning**

OPERATOR

cf create-service-broker

cf enable-service-access

OPERATOR

Service “Tiles” in PCF Ops Manager

Only installed services appear in the “marketplace”

PCF Ops Manager

admin

Available Products

Ops Manager Director
No upgrades available

Pivotal Elastic Runtime
No upgrades available

Jenkins Enterprise by CloudBees for Pivotal CF
No upgrades available

MySQL for Pivotal Cloud Foundry
No upgrades available

Pivotal Ops Metrics
No upgrades available

Import a Product

Download PCF compatible products at Pivotal Network

Installation Dashboard

Ops Manager Director for vmware vSphere
v1.4.0.0

Pivotal Elastic Runtime
v1.4.0.0

Jenkins Enterprise by CloudBees for Pivotal CF
v1.3.8.3

MySQL for Pivotal Cloud Foundry
v1.5.0.0

Pivotal Ops Metrics
v1.4.0.0

No updates

Apply changes

Recent Install Logs

Operations staff import service “tiles”, configure them and **Apply Changes** to install.

Provisioning – Developer View

- *Only* concerned with what a developer has to do
  - Create (provision) a service
  - Bind it to your application

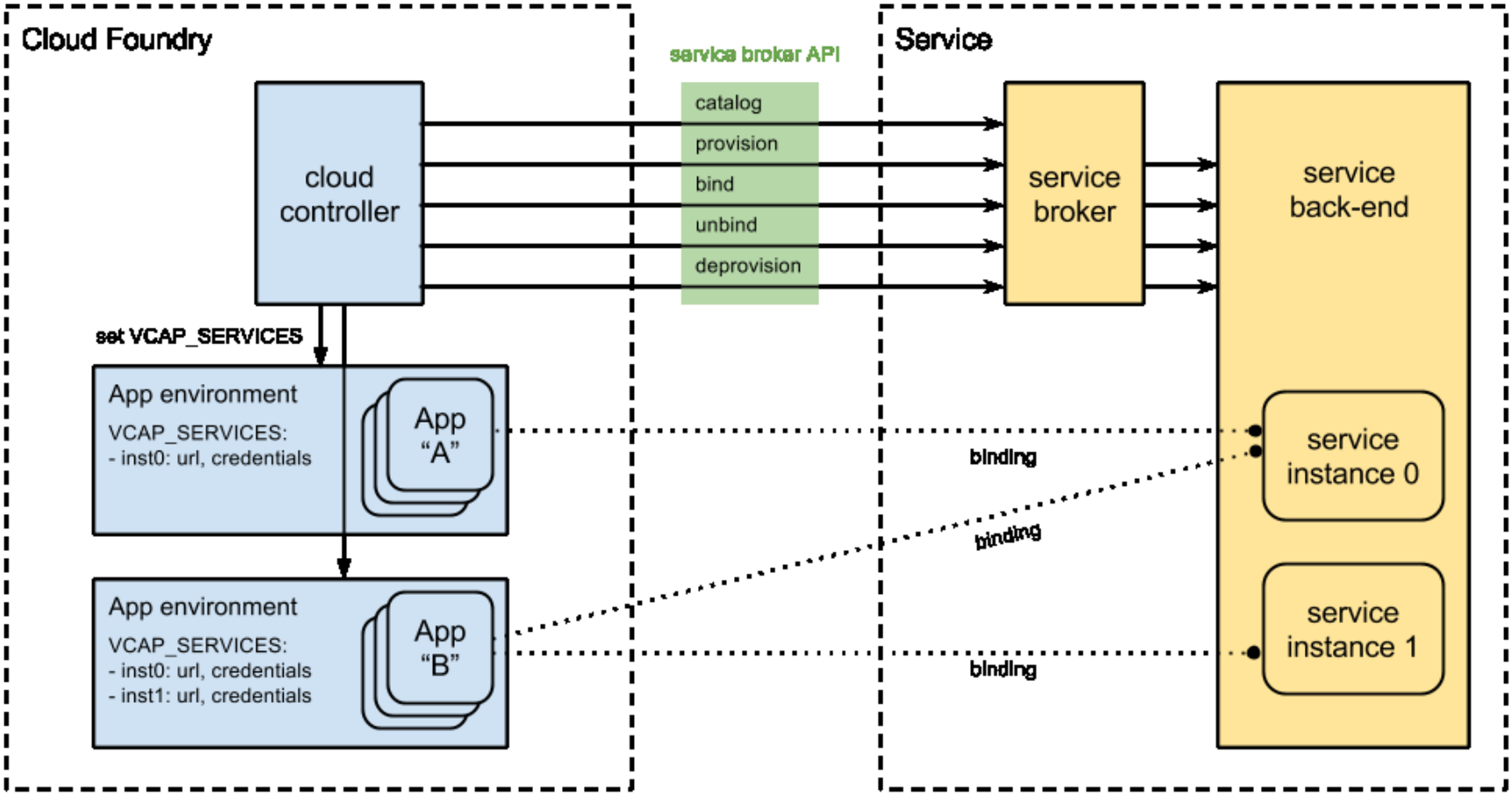
DEVELOPER

cf create <myService>

cf bind <myApp> <myService>

Pivotal™

Services Overview – Service Brokers



Creating a Service Instance

- Actually we are *Provisioning* an *instance* of a service
  - It must already exist in CF marketplace
- Use App Manager or **`cf create-service`**
  - Allows selection of service and plan
- Service instance becomes available to *current space*
  - And any applications running in it
  - For multiple spaces, run **`create-service`** in *each* space

Pivotal™

Roadmap

- Provisioning Services
- **Using the CLI**
- Using the Pivotal CF App Manager Console
- Binding to a Service
- User Provided Services

Pivotal™

Finding Available Services

Command Line Interface

- Check marketplace for available services
  - Essentially a service catalog

```
example$ cf marketplace
Getting services from marketplace in org pivotaledu / space development as user@domain...
OK

service       plans                                                    description

blazemeter    free-tier, basic1kmr, pro5kmr, pp10kmr, hv40kmr          The JMeter Load Testing Cloud
cleardb       spark, boost, amp, shock                                 Highly available MySQL for your Apps
cloudamqp     lemur, tiger, bunny, rabbit, panda                       Managed HA RabbitMQ servers in the cloud
cloudforge    free, standard, pro                                      Development Tools In The Cloud
elephantsql   turtle, panda, hippo, elephant                           PostgreSQL as a Service
ironmq        pro_platinum, pro_gold, large, medium, small, pro_silver Powerful Durable Message Queueing Service
ironworker    large, pro_gold, pro_platinum, pro_silver, small, medium Scalable Background and Async Processing
...
```

Finding Existing Service Instances
Command Line Interface

- List existing services instance
  - In *current space*
- In this example: one service instance called mysql

```
example$ cf services
Getting services in org pivotaledu / space development as user@domain...
OK

name        service     plan    bound-apps
mydb        cleardb     spark   booking-app-123
```

- Remember, to change spaces
  - cf target -s [space-name]

Pivotal

Provisioning a new Service Instance
Command Line Interface

- Provision a new service instance
  - Added to *current space*
  - Give it a name
  - Choose the correct plan or contract
- Usage
  - cf create-service [service-name] [plan-name] [instance-name]

```
example$ cf create-service elephantsql turtle mypg
Creating service mypg in org pivotaledu / space development as user@domain...
OK
```

Pivotal

Finding Existing Service Instances
Command Line Interface

- List service instances again for *current* space
  - New service instance now appears

```
example$ cf services
Getting services in org pivotaledu / space development as user@domain...
OK

name        service     plan    bound-apps
mydb        cleardb     spark   booking-app-123
mypg        elephantsql turtle
```

Service created

Roadmap

- Provisioning Services
- Using the CLI
- **Using the Pivotal CF App Manager Console**
- Binding to a Service
- User Provided Services

Pivotal™

Provisioning Service Instances
GUI

Pivotal Web Services

pivotaledu > development

ORG

pivotaledu

SPACES

development

production

staging

Marketplace

Docs

Support

Tools

Blog

Status

SPACE

development

APPLICATIONS LEARN MORE

STATUS	APP	INSTANCES	MEMORY
STOPPED	booking-app-123 booking-app-123.cfapp....	1	1GB

SERVICES ADD SERVICE

SERVICE INSTANCE	SERVICE PLAN	BOUND APPS
mysql Manage | Documentation | Support | Delete	ClearDB MySQL Database spark	1

Marketplace

Services

Finding Available Services

Service Selection

Pivotal Web Services

pivotaledu > Marketplace

ORG

pivotaledu

SPACES

development

production

staging

Marketplace

Docs

Support

Tools

Blog

Status

Services Marketplace

Get started with our free marketplace services. Upgrade to gain access to premium service plans.

BlazeMeter
The JMeter Load Testing Cloud

ClearDB MySQL Database
Highly available MySQL for your Apps.

CloudAMQP
Managed HA RabbitMQ servers in the cloud

CloudForge
Development Tools In The Cloud

ElephantSQL
PostgreSQL as a Service

VIEW PLAN OPTIONS

IronMQ
Powerful Durable Message Queueing Service

IronWorker
Scalable Background and Async Processing

d Impact
d and on-demand performance testing

Choose to check available plans

Provisioning a new Service Instance

Pick a Plan

Pivotal Web Services

pivotaledu > Marketplace > ElephantSQL

ORG

pivotaledu

SPACES

development

production

staging

Marketplace

Docs

Support

Tools

Blog

Status

ElephantSQL

PostgreSQL as a Service

ABOUT THIS SERVICE

The most advanced open-source database, hosted in the cloud.

Documentation | Support

SERVICE PLANS

Choose the plan

Tiny Turtle	free
Pretty Panda	$19.00/MONTH
Happy Hippo	$99.00/MONTH
Enormous Elephant	$499.00/MONTH

PLAN FEATURES

- Shared high performance cluster
- 20 MB data
- 4 concurrent connections

SELECT THIS PLAN

Provisioning a new Service Instance

Provision (Create) Service

Pivotal Web Services

pivotaledu > Marketplace > ElephantSQL > Add a new Service Instan…

ORG

pivotaledu

SPACES

development

production

staging

Marketplace

Docs

Support

Tools

Blog

Status

ElephantSQL

PostgreSQL as a Service

ABOUT THIS SERVICE

The most advanced open-source database, hosted in the cloud.

Documentation | Support

COMPANY

84codes AB

SERVICE PLAN

Tiny Turtle free

CONFIGURE INSTANCE

Instance Name: mypg

Add to Space: development

Bind to App: [do not bind]

SUBSCRIPTION TERMS

- A monthly subscription charge is added to the bill at the start of every monthly service period.
- Cancel a service subscription by deleting the instance.
- Credits are not issued for the unused portion of a monthly subscription period.
- Your subscription will be billed monthly starting today.

CANCEL ADD

Specify instance name

Create the service

Provisioning a new Service Instance
Complete

Pivotal Web Services

pivotaledu > development

Service instance mypg created.

Success

ORG

pivotaledu

SPACES

development

production

staging

Marketplace

Docs

Support

Tools

Blog

Status

SPACE

development

APPLICATIONS LEARN MORE

STATUS	APP	INSTANCES	MEMORY
STOPPED	booking-app-123 booking-app-123.cfapp...	1	1GB

SERVICES ADD SERVICE

SERVICE INSTANCE	SERVICE PLAN	BOUND APPS
mysql Manage | Documentation | Support | Delete	ClearDB MySQL Database spark	1
mypg Manage | Documentation | Support | Delete	ElephantSQL turtle	0

Service available

Pivotal

Roadmap

- Provisioning Services
- Using the CLI
- Using the Pivotal CF App Manager Console
- **Binding to a Service**
- User Provided Services

Pivotal™

Accessing Service Instances from an App?

Traditional way

- Traditionally, for an application to access a service instance, connection properties are required
- For example: a database instance
  - Need to know service address / port, credentials
    - such as a JDBC connection
- May be hard-coded, provided through the environment or a configuration file
- Typically service-specific code is required

Pivotal™

Accessing Service Instances from an App?

Traditional way

- Configuration files

```
development:
  adapter: mysql2
  encoding: utf8
  database: pivotaldb
  username: pivotal
  password: pivotal
  host: myDbHost
  port: 3306
```

Ruby

```
datasource {
    driverClassName = "com.mysql.jdbc.Driver"
    username = "pivotal"
    password = "pivotal"
    url = "jdbc:mysql://myDbHost:3306/pivotaldb"
}
```

Groovy

```
datasource.driverClassName="com.mysql.jdbc.Driver"
datasource.username="pivotal"
datasource.password="pivotal"
datasource.url="jdbc:mysql://myDbHost:3306/pivotaldb"
```

Java

Accessing Service Instances from an App?
The CloudFoundry way

- In CloudFoundry, you **bind** the service instance to apps
  - Connection credentials are negotiated / defined for you
  - Application code only needs service name and type/kind
    - Example: a Postgres instance with name "mypg"
  - Service details injected into application by CF
    - **`VCAP_SERVICES`**
  - Any changes (host/port/credentials) are managed external to the application.

Pivotal™

Example VCAP_SERVICES Property

```
VCAP_SERVICES=
{
  cleardb-n/a: [
    {
      name: "cleardb-1",
      label: "cleardb-n/a",
      plan: "spark",
      credentials: {
        name: "ad_c6f4446532610ab",
        hostname: "us-cdbr-east-03.cleardb.com",
        port: "3306",
        username: "b5d435f40dd2b2",
        password: "ebfc00ac",
        uri: "mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-
                        03.cleardb.com:3306/ad_c6f4446532610ab",
        jdbcUrl: "jdbc:mysql://b5d435f40dd2b2:ebfc00ac@us-
                 cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab"
      }
  ...
```

ClearDB is the MySQL instance offered through App Direct

Using a Service – Cloud Foundry

Binding using the CLI

- **Binding** *associates* an application to a service instance.
  - Use `cf bind-service`
  - Syntax
    - `cf bind-service [app_name] [service_name]`

```
example$ cf bind-service booking-app-456 mypg
Binding service mypg to booking-app-456 in org pivotaledu / space development
as user@domain...
OK
TIP Use 'cf restage' to ensure your env variable changes take effect
```

Note

Using a Service – Cloud Foundry

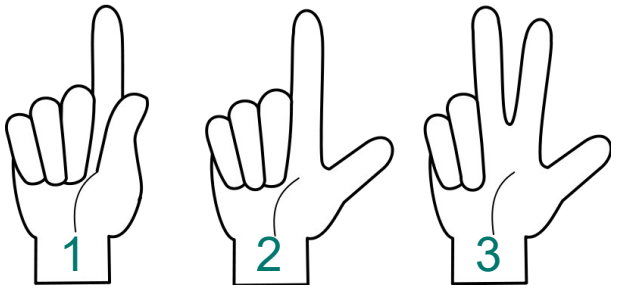
Binding using a Manifest

- Add a *services* section to you application in the manifest
  - Example `manifest.yml`

```
---
applications:
- name: booking-app-456
  memory: 256M
  instances: 2
  host: booking-app-456
  domain: cfapps.io
  path: target/booking-app.war
  # services, one per line
  services:
  - mypg
  - mydb
```

Using a Service
The CloudFoundry way

- Cloud Foundry provides Application with **`VCAP_SERVICES`** environment variable
  - Which contains connection details / credentials in JSON.
- How can an application obtain the credentials?
- Three options:
  1. Manual
     - Explicit low-level code
  2. Custom library
     - Explicit code, higher level interface
  3. Auto configuration
     - CF does it for you



Using a Service – Application View
1. Manually

- Manual configuration
  - Access **`VCAP_SERVICES`** environment variable
  - In your code, parse the JSON (see next slide)
- Very low-level but works in most languages
  - Fall-back option when options 2 and 3 aren't possible

Pivotal™

Recall: VCAP_SERVICES Property

```
VCAP_SERVICES=
{
  cleardb-n/a: [
    {
      name: "cleardb-1",
      label: "cleardb-n/a",
      plan: "spark",
      credentials: {
        name: "ad_c6f4446532610ab",
        hostname: "us-cdbr-east-03.cleardb.com",
        port: "3306",
        username: "b5d435f40dd2b2",
        password: "ebfc00ac",
        uri: "mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-
                        03.cleardb.com:3306/ad_c6f4446532610ab",
        jdbcUrl: "jdbc:mysql://b5d435f40dd2b2:ebfc00ac@us-
                 cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab"
      }
  ...
```

Just a very long string in JSON format

Parse to extract these credentials

Using a Service – Application View

2. Custom Library

- Avoid manual parsing using a cloud-aware library
  - Cloud foundry aware helper code
    - Language/framework dependent
    - Parses **VCAP_SERVICES** for you
  - *JVM:* use Spring Cloud project
  - *Node.js:* use *cfruntime* object

Derived from VCAP_SERVICES

```
for (ServiceInfo service : cloud.getServiceInfos() ) {
    if (service instanceof MysqlServiceInfo)
        connectionURI = ((MysqlServiceInfo)service).getJdbcUri();
    } ...
```

Java Example

Using a Service – Application View

3. Auto-configuration

- Cloud Foundry creates the service connection for you
  - Not always supported, depends on:

1) The buildpack
   - Some buildpacks support auto-configuration, others do not.
2) The framework
   - Spring, Grails, Lift, Rails currently supported.
   - NOT (currently) supported for Node.js, Sinatra, Rack …
3) The uniqueness of the service type
   - For example, can auto-configure ONE database connection
     - CF doesn't know which is which if there are two or more

Accessing Connection Information

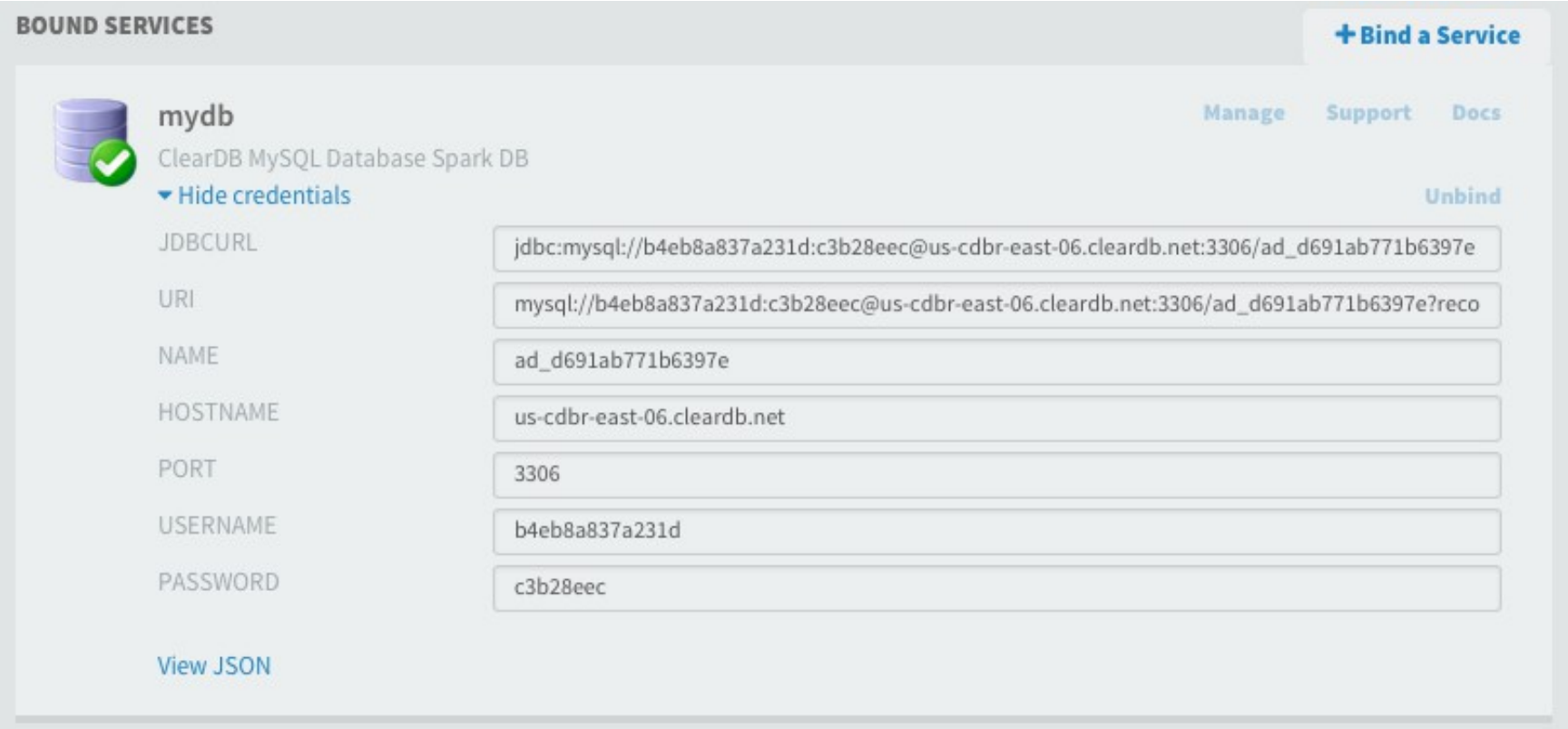
- Recall
  - Connection information once bound is in **`VCAP_SERVICES`**
  - Every application's environment is logged at startup

- Once application is *staged*, view connection information using
  - *`cf env [app-name]`*
  - Look for **`VCAP_SERVICES`** in the output

Pivotal™

Accessing Connection Information - 2

- Connection information also available via App Manager:



Roadmap

- Provisioning Services
- Using the CLI
- Using the Pivotal CF App Manager Console
- Binding to a Service
- **User Provided Services**

Pivotal™

User Provided Service Instances

- **User-provided service instances** are service instances
  - Already provisioned outside of Cloud Foundry
  - Behave like other service instances once created
  - Are little more than predefined configurations
    - A "*mock*" service for providing credentials
- When bound they provide service instance configuration (including credentials) to applications
  - Avoids hard coding service instance endpoints

http://docs.cloudfoundry.org/devguide/services/user-provided.html

Pivotal™

Use Cases

User Provided Service Instances

- These are typically **legacy** or **existing instances** of a **service** (databases, queues, email, etc)
  - Applications connect to the *same* instance
    - With CF services, applications get *different* instances
  - Typically used with CF *on-premise*
    - Easy integration of your CF PaaS with your existing systems

- **Credential passing** used to inject the *same* credential set into each application

Pivotal™

Defining User Provided Services – 1

- Use `cf create-user-provided-service` command
  - Provide name and parameters/credentials
  - All applications bound to *same* instance in *same* way

```
$ cf cups mydb -p "hostname, port, username, password, name"

hostname> db.example.com

port> 1234

username> dbuser

password> dbpasswd

name> mydb
Creating user provided service mydb ... OK
```

Or use alias: cf cups

Specify *any* list of parameters here

Prompts for parameters values

Defining User Provided Services – 2

- *Or* define within application's manifest.yml:

```
---
applications:
- name: spring-music
  memory: 512M
  instances: 1
  host: spring-music
  domain: cfapps.io
  path: build/libs/spring-music.war
  services:
    mydb:
      label: user-provided
      credentials:
        uri: postgres://dbuser:dbpass@db.example.com:1234/dbname
        username: pivotal
        password: pivotal
```

User Provided Services - Accessing

- Bound service properties available in **VCAP_SERVICES** environment variable
- In your code
  - Access variable
  - Parse JSON
  - Use to connect

```
{
  user-provided: [
    {
      name: "mydb",
      label: "user-provided",
      tags: [ ],
      credentials: {
        hostname: "db.example.com",
        port: "1234",
        username: "dbuser",
        password: "dbpasswd",
        name: "mydb"
      }
    }
  ]
}
```

Example: Application with Multiple Services

```
VCAP_SERVICES: {
  "rediscloud": [
   {
    "credentials": {
      "hostname": "redisvr...com",
      "password": "wU974wucDT45Jc",
      "port": "19016"
    },
    "label": "rediscloud",
    "name": "session-replication",
    "plan": "25mb",
    "tags": [
     "Data Stores",
     "Cloud Databases",
     "Developer Tools",
     "Data Store",
     "key-value",
     "redis"
    ]
   }
  ],
```

```
"user-provided": [
   {
    "credentials": {
     "uri": "http://review.cfapps.io"
    },
    "label": "user-provided",
    "name": "reviews",
    "syslog_drain_url": "",
    "tags": []
   },
   {
    "credentials": {
     "uri": "http://products.cfapps.io"
    },
    "label": "user-provided",
    "name": "products",
    "syslog_drain_url": "",
    "tags": []
   }
  ]
 }
```

What you have learned

- Provisioning Services
  - Using the CLI
  - Using the Pivotal CF App Manager Console
- Binding to a Service
- User Provided Services

Pivotal™

Lab

Creating a service and binding to it

Pivotal™