



CLOUD **FOUNDRY**

# Pivotal Cloud Foundry Developer

Hands-On Workshop

Application Deployment using Pivotal Cloud  
Foundry

Pivotal

# Copyright Notice

Copyright © 2015 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.



CLOUD **FOUNDRY**

# Pivotal Cloud Foundry Developer

A three-day workshop on using PCF to  
develop and deploy enterprise applications

Course Overview

**Pivotal**<sup>TM</sup>

# Introductions

- Name
- Role / title
- Background in Software Development
- Experience with command-line or “shell”
- What you would like to get out of this course

# Logistics

- Instructor introduction
  - Course registration (if needed)
  - Courseware
  - Internet access
  - Phones on silent
- Working hours  
Lunch and breaks  
Toilets/Restrooms  
Fire alarms  
Emergency exits  
*Any other questions?*



Pivotal™

# How You will Benefit

*Gain a solid understanding of Pivotal Cloud Foundry ...  
... from the point of view of a developer*

- Learn to use Pivotal Cloud Foundry to develop, test and deploy web and other applications
- Gain hands-on experience
  - Generous mixture of presentation and labs
- Access to experienced, certified instructors



# Covered in this section

- **Agenda**
- About Pivotal

# Course Agenda: Day 1

- Introduction to Cloud Foundry and PCF
- Concepts
- Getting Started with `cf` CLI
- Logging, Scaling and High Availability
- Services



Pivotal™

# Course Agenda: Day 2

- Environment Variables
- Manifests
- Application Security Groups
- Log Draining
- Blue-Green (Rolling) Deployments
- Introduction to Microservices
- Application Autoscaler
- Application Performance Monitoring
- Metrics



# Course Agenda: Day 3

- Buildpacks
- Service Brokers
- Continuous Delivery
- Route Services
- Advanced Features
  - Tasks, TCP routing, Volume Services
  - Container to Container routing

3

Pivotal™

# Covered in this section

- Agenda
- **Getting the Most from this Course**
- About Pivotal

# Approach & Methodology

- Course is designed to challenge you
  - You will *learn by doing*, as well as from the presentations
- To get through the labs, you will have to:
  - Ask questions
  - Collaborate with colleagues
  - Follow links to, and **read official product documentation**,
  - Use “`cf help`” and “`cf help <command>`”
  - Experiment

# Tips for Working Through a Lab Exercise

- *Take the time to read the referenced documentation*
- For every new `cf` command
  - Check out the `cf help` for that command.
- **Example:**
  - Look up “services” in the online documentation
  - You need to use `cf create-service` for the first time
    - Run `cf help create-service` to see how it works

# Covered in this section

- Agenda
- Getting the Most from this Course
- **About Pivotal**

# About Pivotal

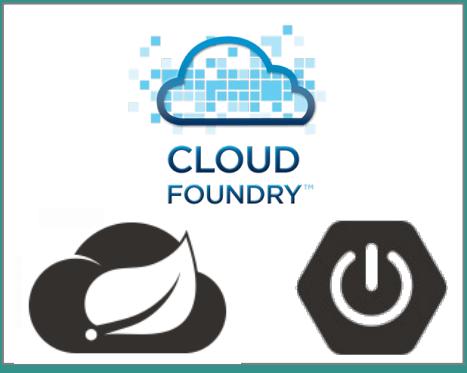
- Pivotal Software
  - Joint venture by EMC, VMware and General Electric
  - Formed in 2013, privately held
- Several products supporting cloud deployment
  - Cloud Foundry itself
  - Spring Projects
  - Gemfire Distributed Data Grid
  - RabbitMQ Messaging (AMQP)



# The Pivotal World

## Cloud Foundry

*Cloud Independence  
Microservices  
Continuous Delivery  
Dev Ops*



## Development

*Frameworks  
Services  
Analytics*



## Big Data Suite

*High Capacity  
Real-time Ingest  
SQL Query  
Scale-out Storage*



Pivotal **Labs**

*Working with clients to build better apps more quickly*

# Covered in this section

- Agenda
- About Pivotal

Let's get on with the course...!





CLOUD **FOUNDRY**

# What is Cloud Foundry?

## Overview

Introduction to Cloud Technologies

Pivotal™

# What is Cloud Foundry?

- **Introduction to Cloud**
- Industry Trends
- Cloud Foundry
- Pivotal Cloud Foundry



Pivotal™



CLOUD FOUNDRY

# “The Cloud”

- Means many things to many people.
  - Distributed applications accessible over a network
    - Typically, but not necessarily, The Internet
  - An application and/or its platform
  - Resources on demand, inherently virtualized
  - Public, private or both (hybrid)

“The cloud is not about *where* computing is done, it’s about *how* computing is done”

*Paul Maritz Executive Chairman, Pivotal*

Pivotal™

# Types of Cloud Computing: IaaS

- Infrastructure as a Service
  - Replacement for physical hardware
  - Provides virtual hardware, OS, Network
  - Amazon Web Services (AWS), RackSpace, Microsoft Azure, Google Cloud Platform



Windows Azure



Google Cloud Platform

Pivotal

# Types of Cloud Computing: PaaS

- Platform as a Service
  - More than a raw machine with OS
  - Provides ready-made platform for running apps
  - CloudFoundry, Heroku, Google App Engine, Amazon Elastic Beanstalk



Pivotal™

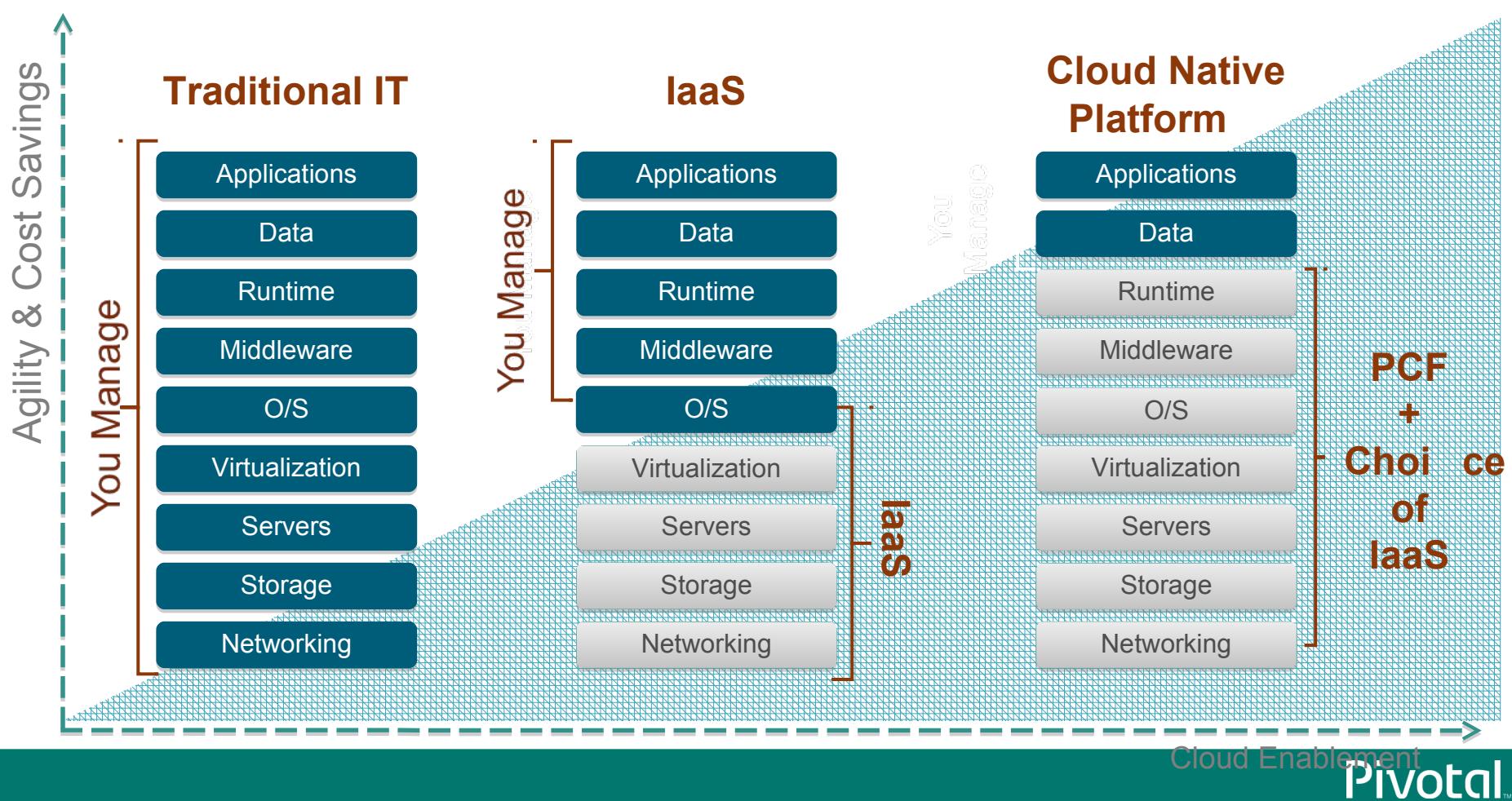
# Types of Cloud Computing: SaaS

- Software as a Service
  - Complete software application
  - SalesForce.com, Google Apps, hundreds of examples



Pivotal

# The Power of the Platform



# Deploying an Application

## IaaS

- Provision a VM
- Install Application Runtime
- Deploy Application
- Configure Load Balancer
- Configure SSL Termination
- Service Connectivity
- Configure Firewall

## Pivotal Cloud Foundry

- `cf create-service`
- `cf push`
- `cf bind-service`

Common application needs  
*handled by the platform*

You can focus on *business value*

# Scaling an Application

## IaaS

- Provision a VM
- Install Application Runtime
- Deploy Application
- Configure Load Balancer
- Configure SSL Termination
- Service Connectivity
- Configure Firewall

## Pivotal Cloud Foundry

- `cf scale`
- Or bind to auto-scaling service

# What is Cloud Foundry?

- Introduction to Cloud
- **Industry Trends**
- Cloud Foundry
- Pivotal Cloud Foundry

# Software is Changing Industries



Square



U B E R



NETFLIX



TESLA

Pivotal™

# Mobile Trends in the Enterprise

- The client has changed ...



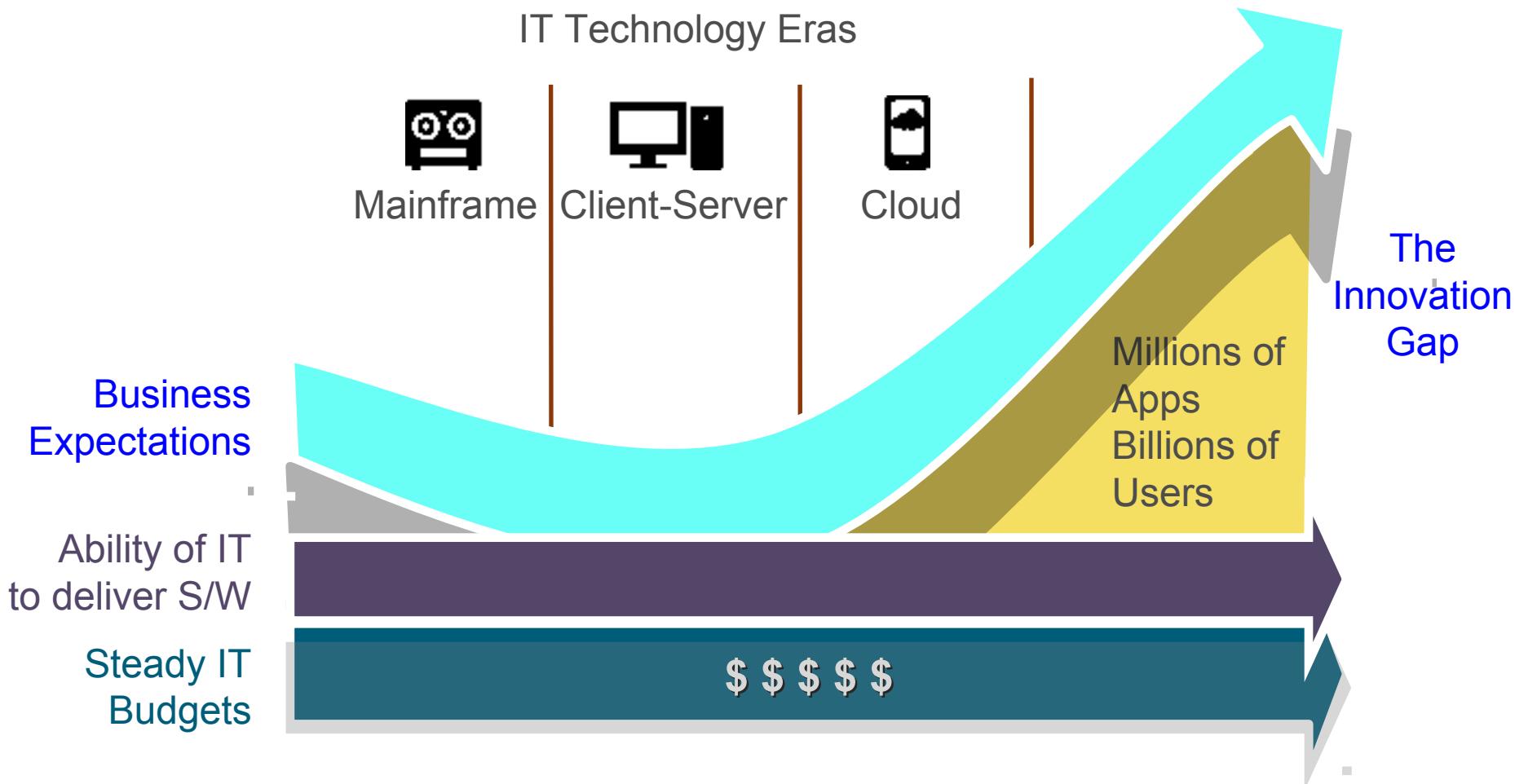
2013  
20% of  
enterprise apps  
are mobile



2017  
90% of  
enterprise apps  
will be desktop  
*and* mobile

Source: Gartner predicts

# Business Needs Exceed IT Expectations

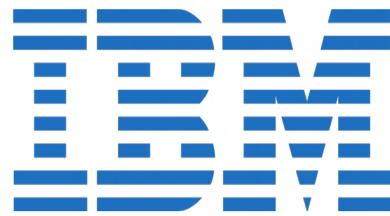


Gartner, 2013: "Hunting and Harvesting in a Digital World: The 2013 CIO Agenda"

Pivotal™

# Software Continues to be Constrained by Silos

Yesterday



Today



**ORACLE**



Google Cloud Platform

Windows  
Server

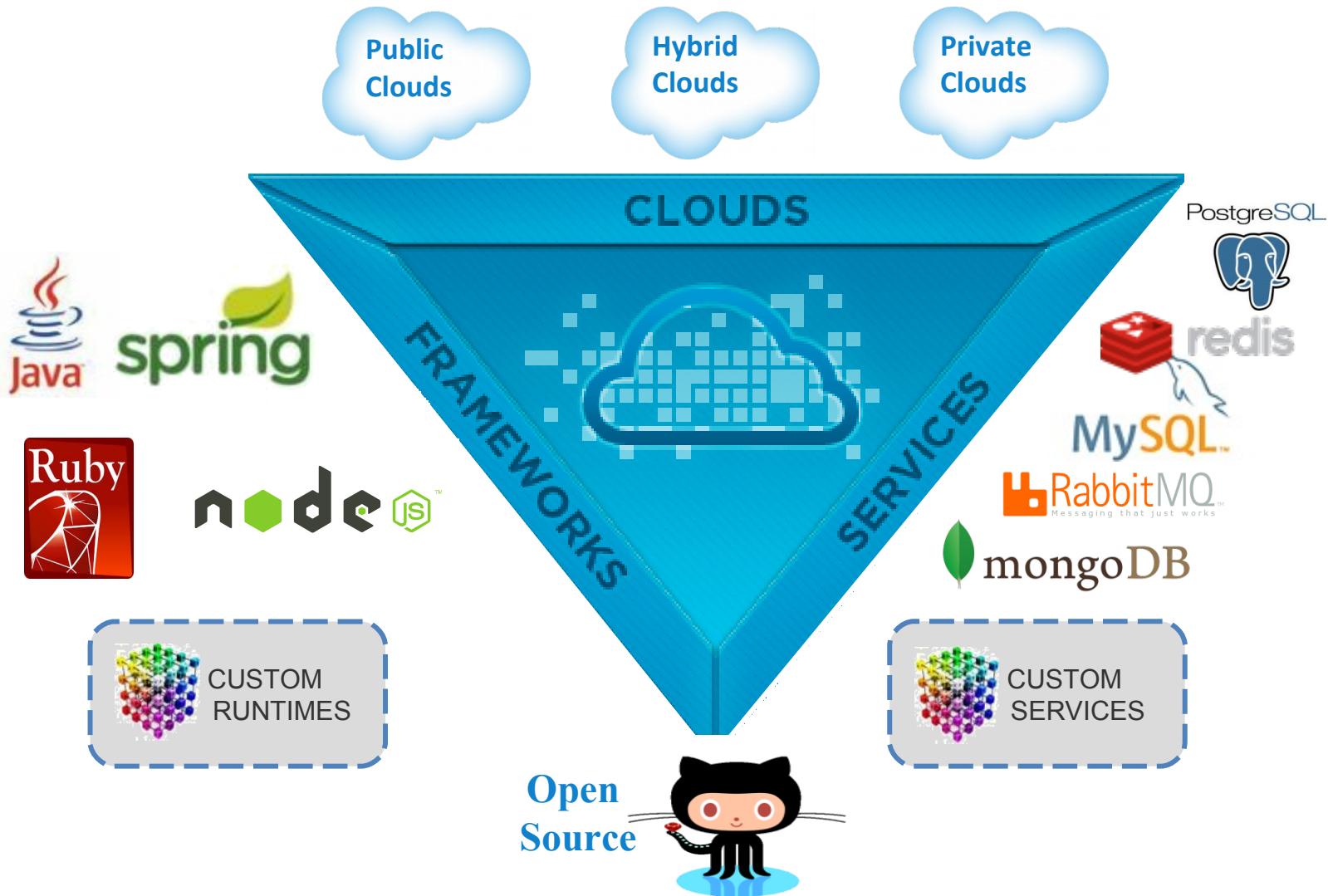


Microsoft Azure

# What is Cloud Foundry?

- Introduction to Cloud
- Industry Trends
- **Cloud Foundry**
- Pivotal Cloud Foundry

# Cloud Foundry – The Open PaaS



# Why Cloud Foundry?

- Open Source
  - Reduce vendor-lock
- Public OR Private OR Hybrid
- Language Independence
  - Via Buildpacks
- Wide, growing range of services



# Cloud Foundry: Foundation Open Governance

PLATINUM



GOLD



SILVER



JPMORGAN CHASE & CO.



resilient scale



TOSHIBA

Pivotal™

# A Major Cloud Foundry Site

- Largest Chinese Internet search site
  - One billion page views per day
  - Ad-Words facility powered by Cloud Foundry



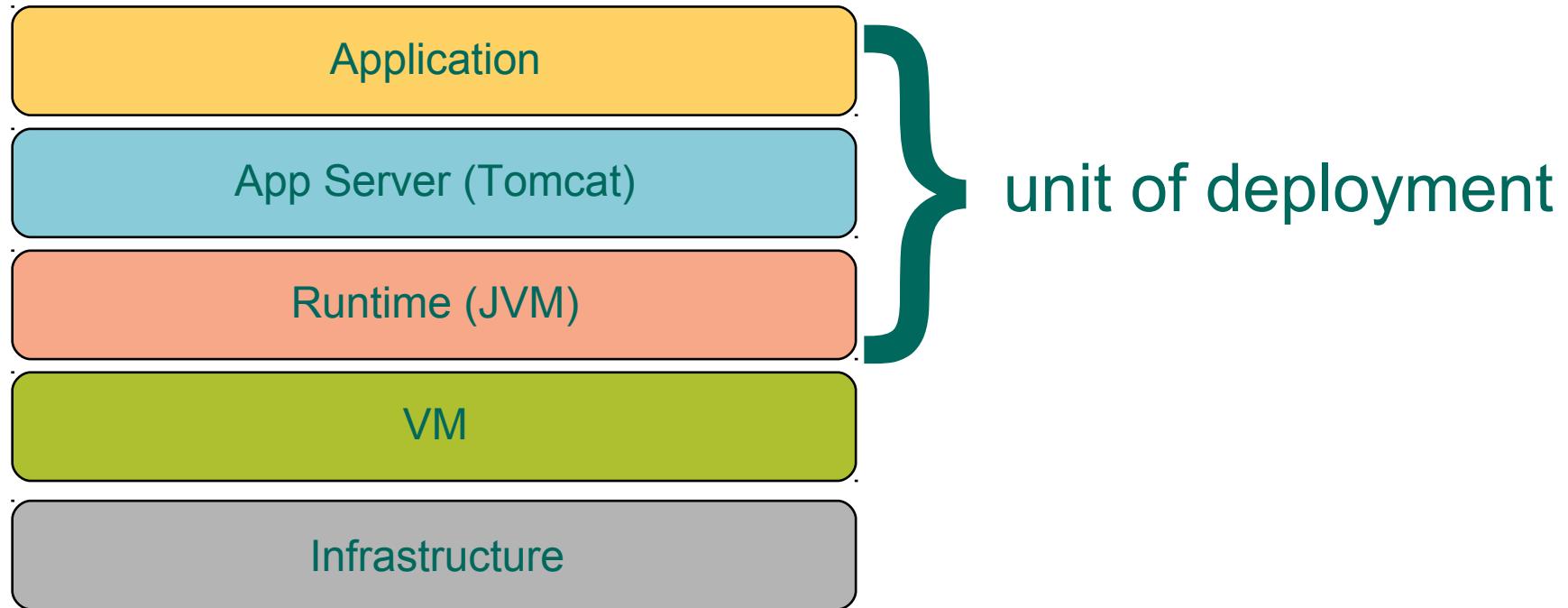
[百科](#) [文库](#) [hao123](#) | [更多>>](#)

# Cloud Foundry Platform as a Service

- **Application** is the new unit of deployment and control
  - Abstracting VMs and Middleware
  - Abstracting containers and processes
  - Data as a service
  - Eliminate bottleneck of provisioning & deployment



# IaaS: VM-centric Deployment

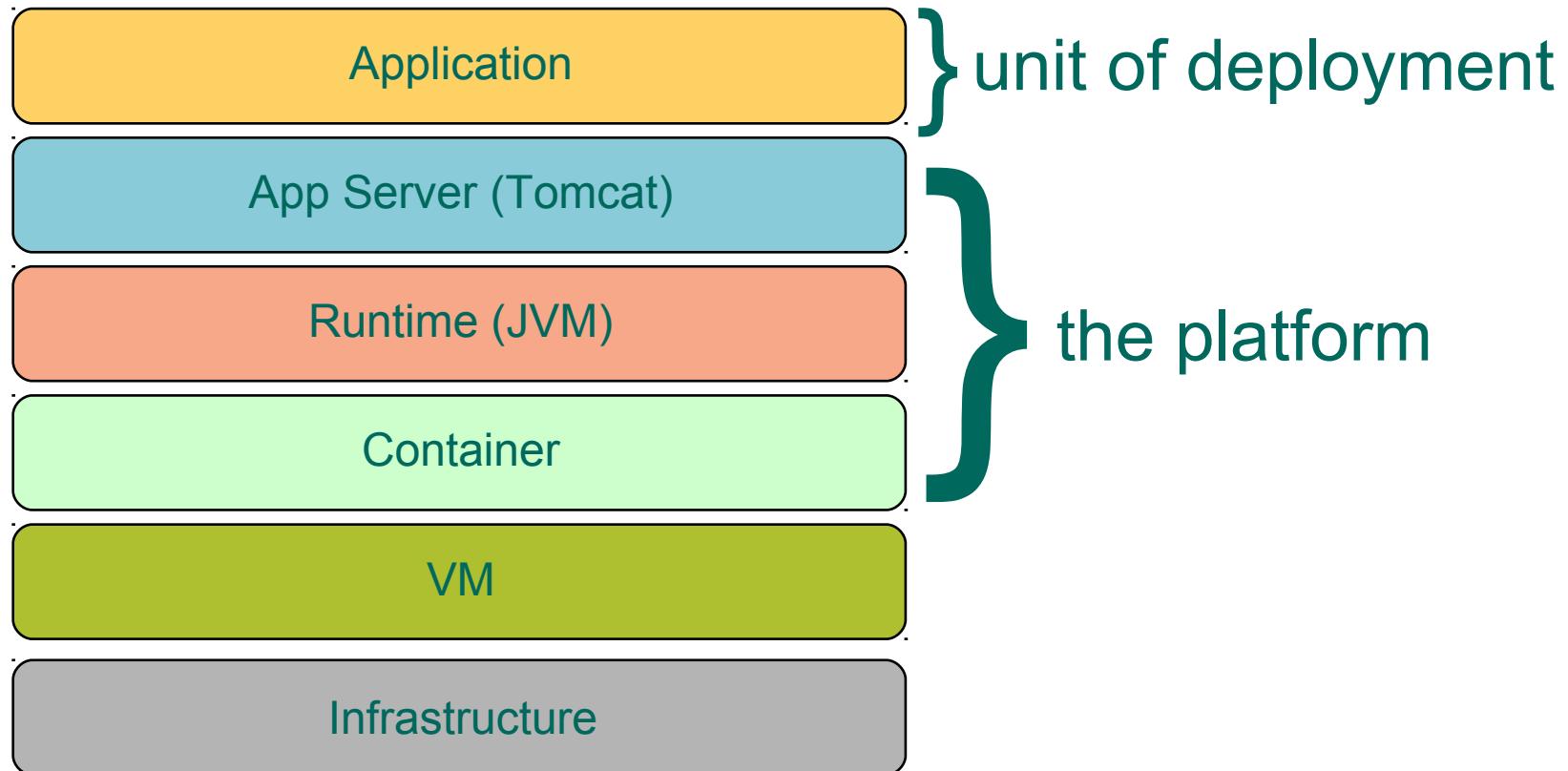


Scaling = creating VMs from blueprints/templates



CLOUD FOUNDRY

# PaaS: App-centric Deployment



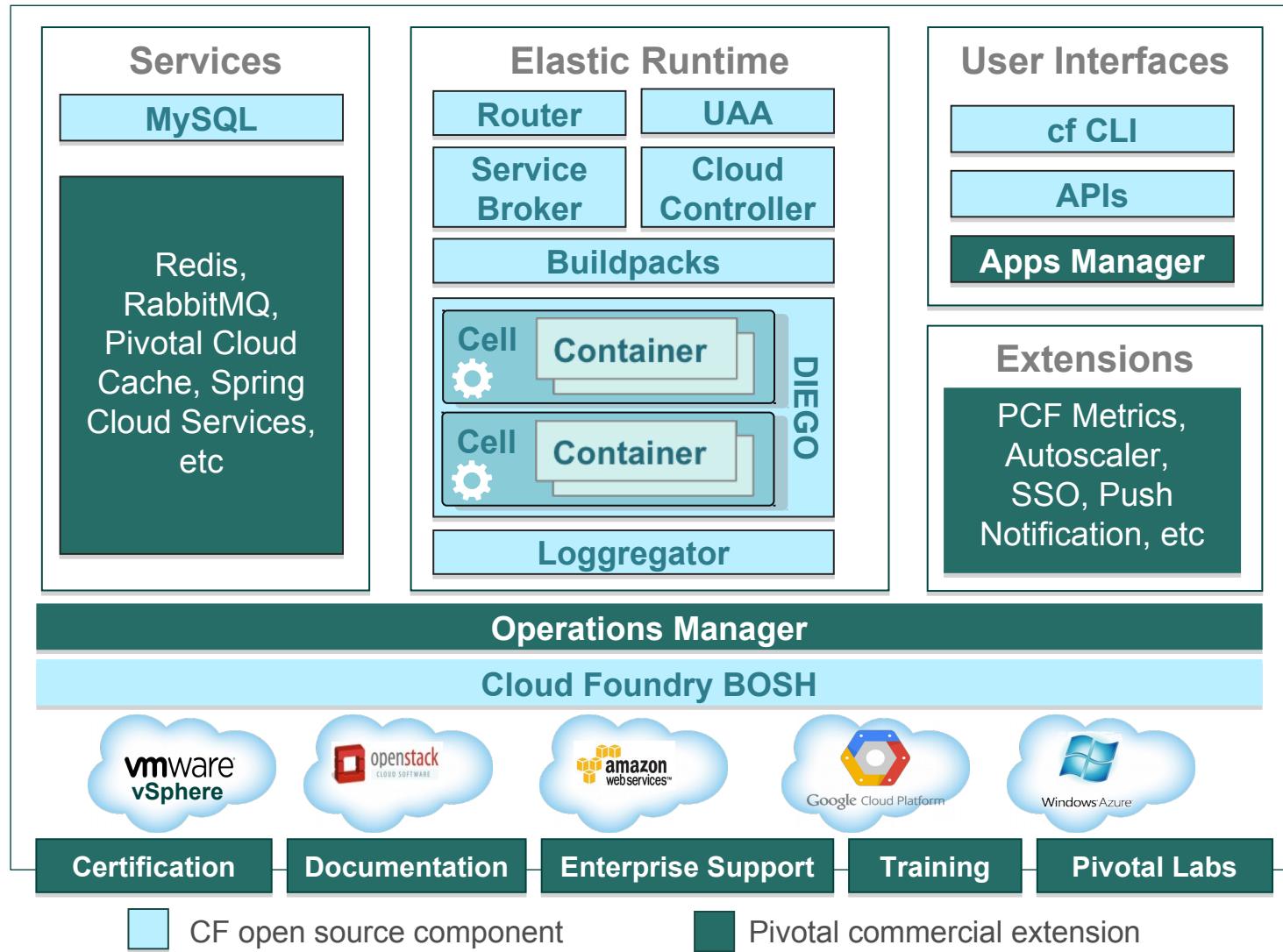
Scaling = creating containers in a VM pool

Pivotal™

# What is Pivotal Cloud Foundry?

- Introduction to Cloud
- Industry Trends
- Cloud Foundry
- **Pivotal Cloud Foundry**

# PCF Architecture



# Which Cloud Foundry Product?



- ◆ Open Source
- ◆ Setup and run a PaaS for yourself
- ◆ No paid support
- ◆ No tools



- PWS**
- ◆ Pivotal Web Services
  - ◆ Public CF PaaS run/managed by Pivotal
  - ◆ Hosted on AWS
  - ◆ No guaranteed SLAs, not for production
  - ◆ Support offered

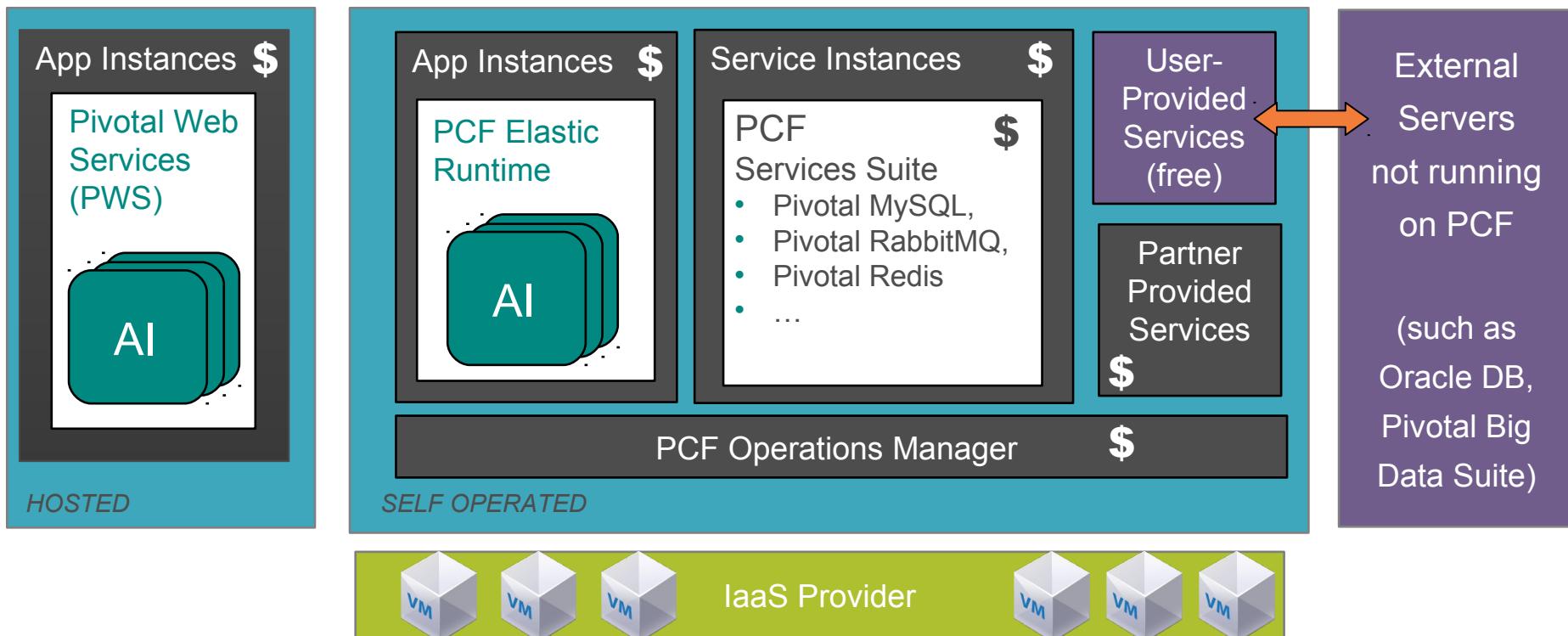
- PWS-E**
- ◆ PWS Enterprise
  - ◆ For existing Pivotal customers to use PWS



- ◆ Commercial product
- ◆ Run private PaaS in-house, or
- ◆ Create public PaaS (ISP managed)
- ◆ Sophisticated Web Console (App Mgr)
- ◆ Additional Tools (Ops Mgr)
- ◆ Less hassle
- ◆ Support offered

# What We Charge For

- Each App Instance, each Ops Manager, each managed service





# Core Tenets of PCF

## Radically Simple, Developer Friendly

- Push an app – “it just works”
- Supporting wide ranging use cases
- Support new application patterns – microservices
- Easy to add/customize

## Broad Ecosystem of Services

Data  
Services

App  
Services

Mobile  
Services

## Operational Benefits for every Application

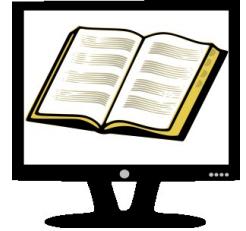
- Highly scalable
- Self healing
- Logging & audit trail
- Existing enterprise policies – user access & authorization
- Application monitoring
- Operational metrics
- App uptime SLAs

Deploy, Operate Update, Scale Platform on Any IaaS

# How can I use PCF?

- PCF is the commercial PaaS based on CF
- Many options:
  - Public – Pivotal Web Services: [run.pivotal.io](https://run.pivotal.io)
  - Basis of other public PaaS offerings
  - Run it in-house as the basis of your private cloud
  - Run it yourself on third-party cloud provider
    - Open Stack, AWS ...





# Documentation

- CF docs can be found on three different sites, depending on the context:
  - <http://docs.cloudfoundry.org>
    - Open-source CF project docs
  - <http://docs.pivotal.io/pivotalcf>
    - Setting up and running PCF
  - <http://docs.run.pivotal.io>
    - Information about running on Pivotal Web Services

# An Explanation of the Various Web Sites

- A web-search for “Cloud Foundry” can be confusing!

URL	Details
cloudfoundry.org	<ul style="list-style-type: none"><li>• The open source project's home page</li><li>• No hosting of any kind here.</li><li>• Documentation</li></ul>
blog.cloudfoundry.org	<ul style="list-style-type: none"><li>• Technical blog</li></ul>
github.com/cloudfoundry	<ul style="list-style-type: none"><li>• The location of the source</li><li>• Download, build, and run if you like!</li></ul>
network.pivotal.io	<ul style="list-style-type: none"><li>• Pivotal products downloads</li><li>• Also documentation, including for PCF</li></ul>
run.pivotal.io	<ul style="list-style-type: none"><li>• Pivotal Web Services (PWS)</li><li>• Pivotal's hosted environment, runs PivotalCF</li></ul>

# Summary

- Cloud Foundry is “the Open PaaS”
- PCF is Pivotal's Cloud Foundry distribution
  - Supported, easier to install / manage
- Various documentation resources available

# Lab

## Setup Lab Environment



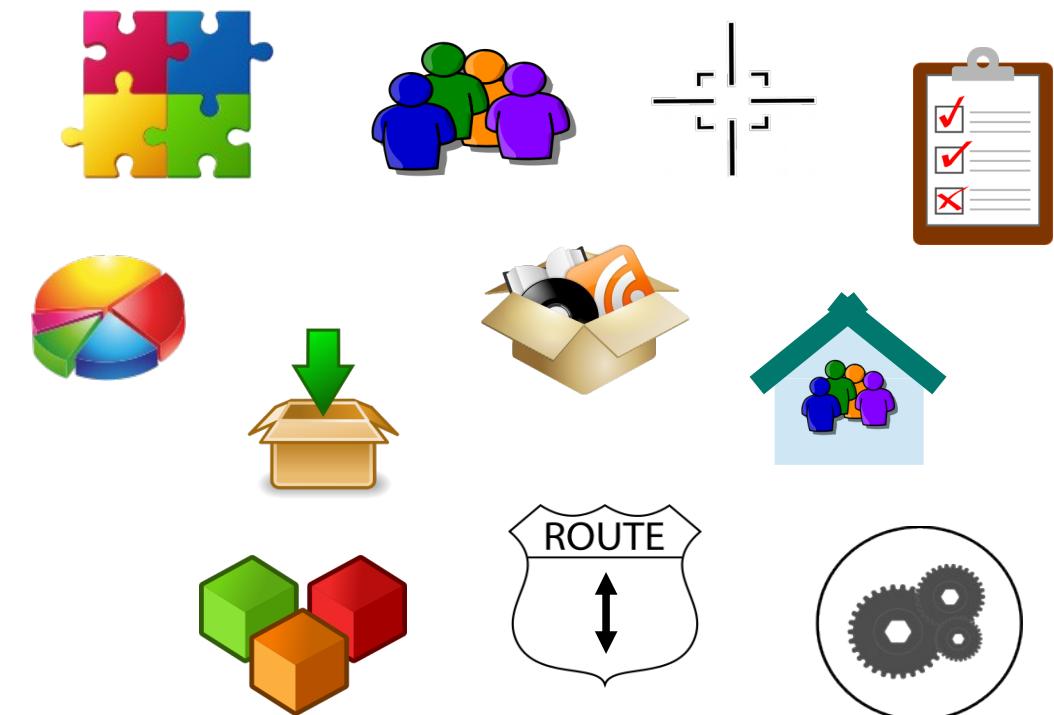
# Cloud Foundry Concepts

## Terms We Use

Organization, Space, Role, Route, Application

# Overview

- After completing this lesson, you should understand:
  - Applications
  - Buildpacks
  - Manifests
  - Organizations
  - Spaces
  - Users and Roles
  - Quotas
  - Domains
  - Routes
  - Services



# Roadmap

- **Applications, Buildpacks, Manifests**
- Organizations, Spaces, Users, and Roles
- Domains and Routes
- Services



# Applications

- PaaS exists to deploy *applications*
  - In Cloud Foundry, the application is the unit of deployment
  - Developers focus on apps, not runtimes or services
- Cloud Foundry is development *agnostic*
  - Not limited to specific language
  - Doesn't mandate the runtime environment you get
  - Some are “out-of-the-box”
  - You can add more

# Buildpacks and Manifests



- Buildpacks
  - Allow CF to support multiple languages and deployment environments
    - Buildpacks for Java, Ruby, JavaScript ...
    - Buildpacks for Tomcat, Rails, Node.JS ...
- Manifests
  - A deployment “blueprint” for an application
  - *Repeatable*: redeploy using same manifest

```
---  
applications:  
- name: nodetestdh01  
  memory: 64M  
  instances: 2  
  host: crn    # unique  
  domain: cfapps.io  
  path: .
```

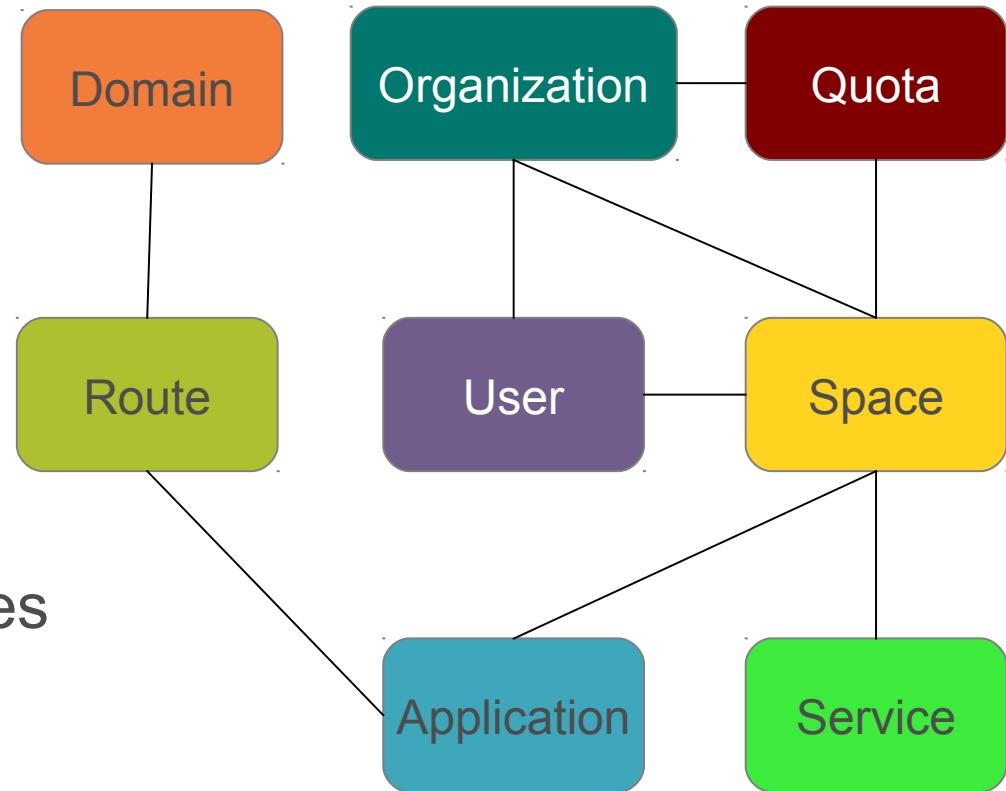
# Roadmap

- Applications, Buildpacks, Manifests
- **Organizations, Spaces, Users, and Roles**
- Domains and Routes
- Services



# Organizations

- Unit of *Tenancy* for Cloud Foundry
- Contains spaces and users
  - Which own routes, applications and services
- Quotas restrict resources
  - For orgs and spaces
- Domain(s)
  - Define routes to apps

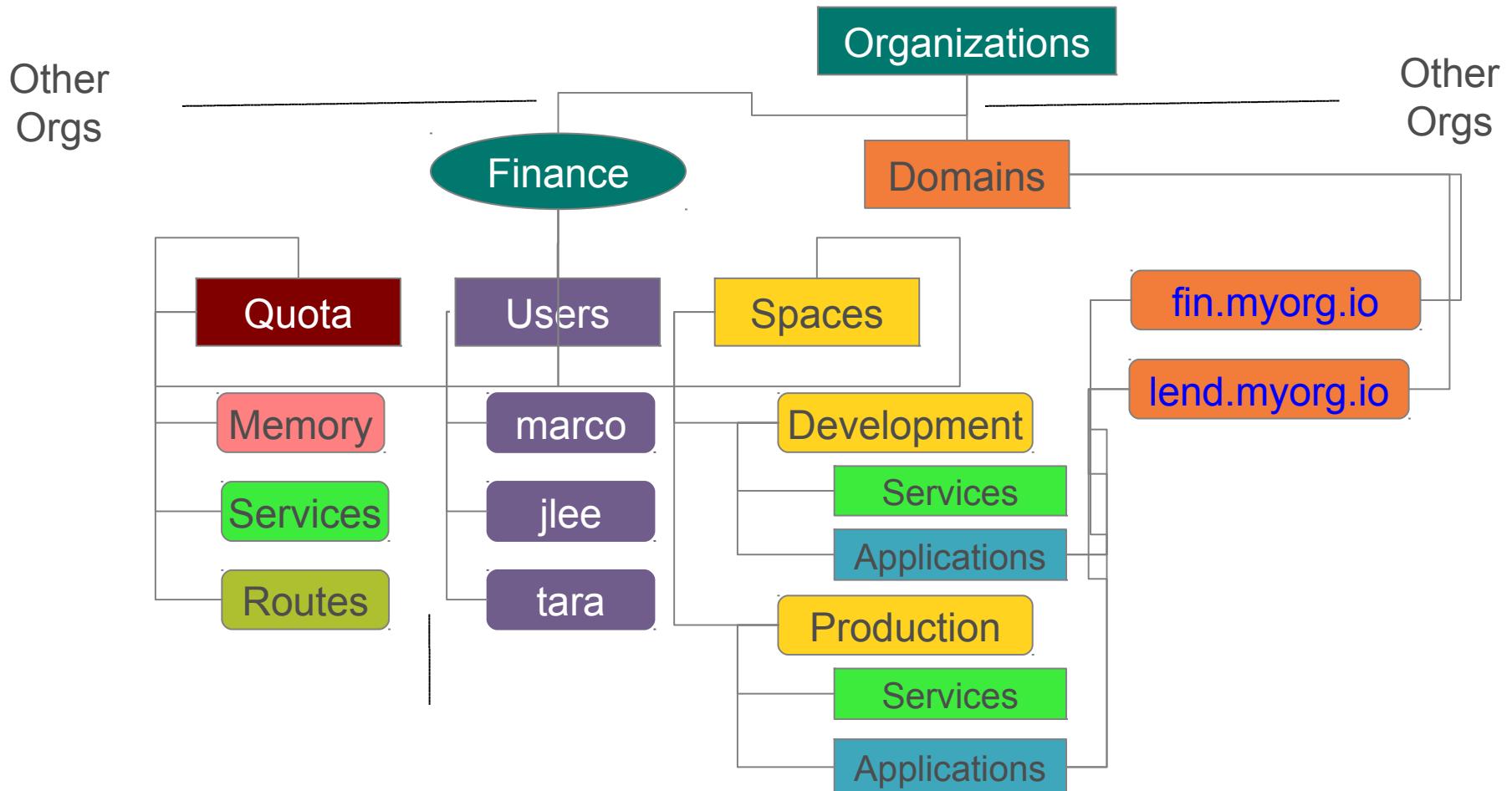




# Organizations

- CF is designed for use by Organizations
  - Typically a company, department, application suite or large project
- Designed to support collaboration
  - Potentially many users
- Defines one or more domains
  - Cloud Foundry instance defines default domain for all organizations
    - For PWS: [cfapps.io](https://cfapps.io)
    - You may add additional domains
- Defines Security, Quotas

# Example Organization



# *Example: Bank Using PCF*

What Organizations might they create?

Divisional

HR, Back-office, Branches ...

By Project

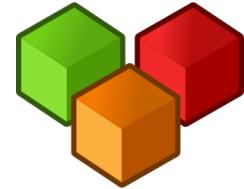
Online Banking, Mobile Banking, International Payments

By application suite

Office Tools, Branch Advisor, Mortgage Assist

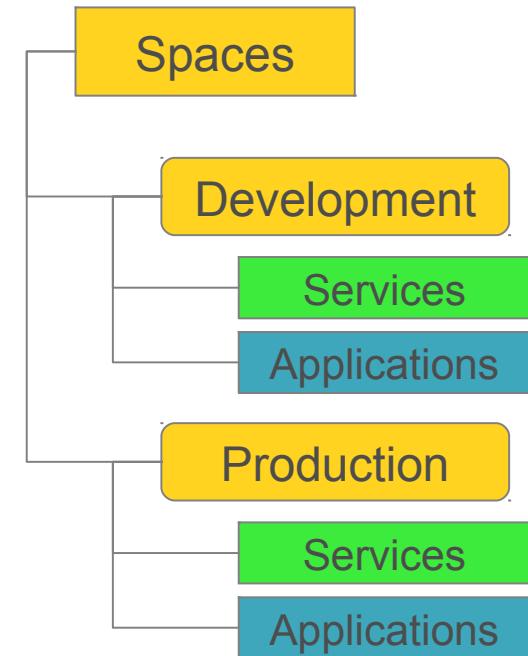
Each would have same/similar spaces

Dev, Test, QA, UAT, Performance, Staging, Production



# Spaces

- Organization contains multiple spaces
  - Default PWS space: *development*
  - Users can create additional spaces
- Provides set of users access to shared location
  - Application development
  - Functionality and/or Performance Testing
  - Quality Assurance
  - Deployment to production
  - Maintenance
- Applications and services scoped to a space
  - Many applications can run/scale within a space





# Users and Roles

- Members of an organization
  - You can invite users to share your “cloud”
- Have specific roles
  - Roles control access to domains and spaces
  - Roles therefore control who has permission
    - To manage routes (see later slides)
    - To deploy applications
    - To add/bind/remove services
- ***Don't*** need to be CF User to access deployed apps
  - Each application does its ***own*** user-management



# Organization Roles

- Organization Manager
  - Can invite/manage users, select/change the plan, establish spending limits
- Organization Auditor
  - View only access to all org and space info, settings, reports



# Application Space Roles

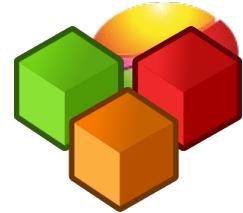
- Space Manager
  - Can invite/manage users, enable features for a given space
- Space Developer
  - Can create, delete, manage applications and services, full access to all usage reports and logs
- Space Auditor
  - View only access to all space information, settings, reports, logs



# Administrator User / Roles

- Special Administrator user / role defined
  - Defined for the Cloud Foundry installation
  - Separate from users defined at Organization / Space
- Several **cf** commands restricted to Administrator only
  - Setting organization and space quotas
  - Defining security groups
  - Administering services
  - Adding, modifying and removing user accounts
- Use without the right role, CLI returns:

```
Server error, status code: 403, error code: 10003, message: You are not  
authorized to perform the requested action
```

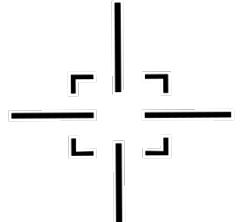


# Quotas

- Restrictions on available resources
  - Total memory available to all applications
  - Total number of routes
  - Max application instance size
  - Total number of services
- Can view
  - Using the CLI
  - Using App Manager (on org homepage)

# Roadmap

- Applications, Buildpacks, Manifests
- Organizations, Spaces, Users, and Roles
- **Domains and Routes**
- Services



# Domains

- Deployed applications are associated with a URL
  - All requests to that URL redirect to the application
- Each Cloud Foundry instance has a default app domain
  - PWS has [cfapps.io](#)
- Custom Domains
  - Register your own domain
  - Give it to Cloud Foundry to use and manage
- Subdomains
  - Each application has a *unique* sub-domain
    - Its URL is therefore *sub-domain.domain*
    - Example: deploy an app to <http://myapp.cfapps.io>



# Routes

- Define how to get to an application
  - A unique *route* exists to each *application* in every *space*
  - Behind the scenes CF uses a router
    - maps incoming requests to the right application
- Domain can be mapped to *multiple* spaces
  - Route can *only* be mapped to *one* space
  - Same application *can* be deployed in *multiple* spaces
    - Each must have a *different, unique* URL
    - Development space route: <http://myapp-test.cfapps.io>
    - Production space route: <http://myapp.cfapps.io>

# Roadmap

- Applications, Buildpacks, Manifests
- Organizations, Spaces, Users, and Roles
- Domains and Routes
- **Services**

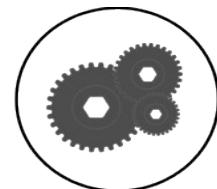
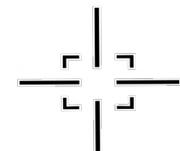
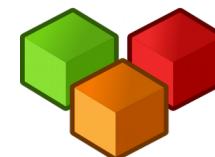


# Services

- Any type of add-on that can be provisioned along side your apps.
  - database, messaging, mail, third-party SaaS provider
- Services are usually “bound” to 1 or more applications
  - Connection info and credentials are put in an environment variable: **VCAP\_SERVICES**
  - **Note:**
    - All configuration data to CF applications should be passed via environment variables
    - Can't use configuration files: *no file system*

# Summary

- After completing this lesson, you should have learned about:
  - Applications, Buildpacks, Manifests
  - Organizations, Users, Roles ,Quotas
  - Domains, Spaces, Routes
  - Services



# Getting Started with Cloud Foundry

Deploying your First Application

Setup, Deploy and Manage

Version 1.12.a

Pivotal™

# Overview

- After<sup>2017</sup> completing this lesson, you should be able to:
  - Deploy an application to CloudFoundry using CLI
  - Manage application instances using App Manager

# PCF Environment

## Option 1 – PWS

- Pivotal Web Services (PWS)
  - Public PCF Installation running at: <http://run.pivotal.io/>
- *Sign up NOW*
  - Apps Manager URL: [console.run.pivotal.io](http://console.run.pivotal.io)
    - Click “**SIGN UP FOR FREE**” to setup a new account and your username/password
  - API URL: [api.run.pivotal.io](http://api.run.pivotal.io)
  - Create an “org”, give it a suitable identifying name
  - Create a “space” within your org named “development”

# PCF Environment

## Option 2 – Company PCF Account

- Using your company's own PCF setup
- *You should know*
  - Apps Manager URL: [login.???](#)
  - API URL: [api.???](#)
  - Create an “org” for the course, if you wish
    - Give it a suitable identifying name
  - Create a “space” within your org named “development”

# Roadmap

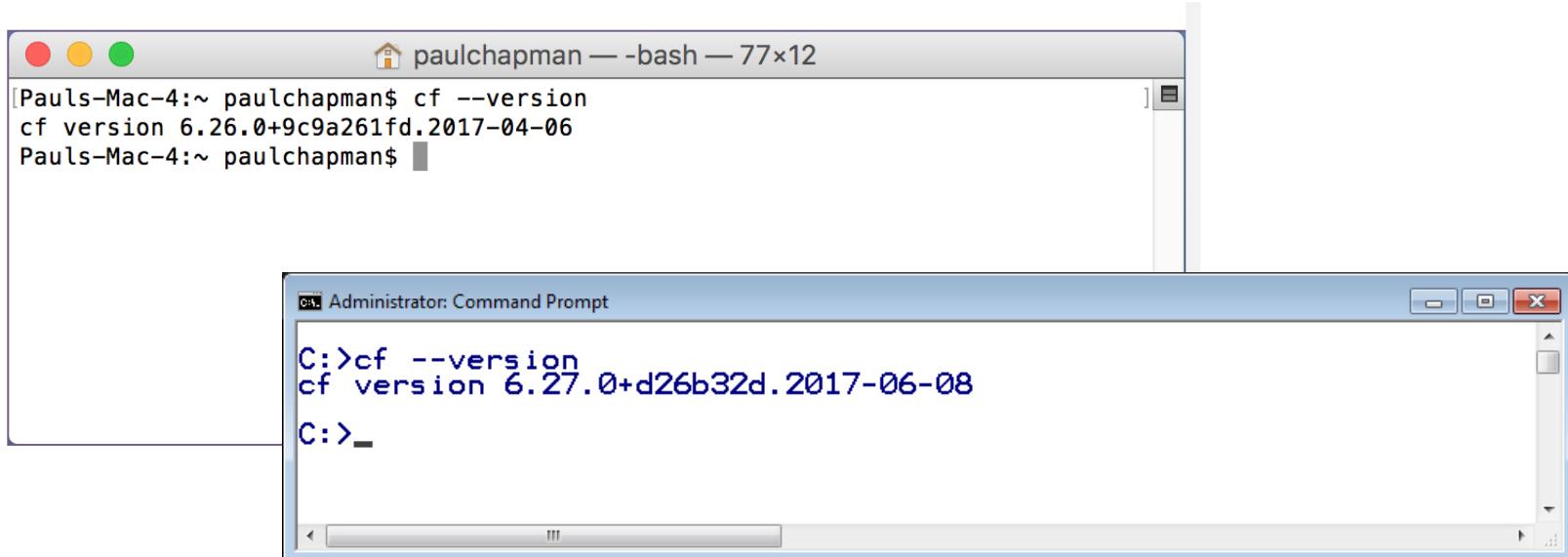
- **Getting Started with the Command Line Interface**
- Login
- Deploying an Application
- Managing Application Instances

# The Command Line Interface

- Several interface options exist for Cloud Foundry
  - Command Line Interface (CLI)
  - Web-based *Application Manager* Console
  - IDE Plugin: Eclipse, STS, IntelliJ, etc
- Primary access is done via the CLI
  - Make sure you have it installed
    - Installation was covered in the “Welcome” module

# DO NOW – Test the CLI Utility

- It is called **cf**
  - Open a Command/Shell window
  - At the prompt type: **cf --version**



# DO NOW – Getting Help

- Get help at any time via `cf help`
  - Or for a particular command: `cf help <command>`
- Perform these steps on your computer:
  - Open a Command prompt or Terminal window
  - Issue the `cf help` command
  - What about `cf help -a`?
  - Get help on the *login* command: `cf help login`
- Answer these questions:
  - What option do you use to specify username?
  - Is specifying the password option encouraged?

# Roadmap

- Getting Started with the Command Line Interface
- **Login**
- Deploying an Application
- Managing Application Instances

# Login to Cloud Foundry

- Need to tell **cf**
  - What cloud foundry instance you are using
  - What your account details are
  - Use **cf login**

Color highlighting  
MacOS, Linux only

```
> cf login -a api.run.pivotal.io -u <username>
API endpoint: api.run.pivotal.io
Authenticating...
OK

Targeted org Cloud Foundry Course
Targeted space development

API endpoint: https://api.run.pivotal.io (API version: 2.0.0)
User:      qzqz2020@yahoo.com.au
Org:       Cloud Foundry Course
Space:    development
```

*Will prompt for anything you  
don't specify  
No -p? Prompts for password*

# Cloud Foundry URLs

- To access CF you need to know 3 URLs
  - *API Endpoint*
    - Identifies your CF instance
    - Used to deploy applications, manage spaces, routes ...
    - The `cf` utility makes *RESTful* requests to this URL
      - Actually to the Cloud Controller
  - *Apps Manager*
    - Application management dashboard (console)
    - *Pivotal CF only*
  - *Apps Domain*
    - Used to access deployed applications
    - *May be same as System Domain*

# Cloud Foundry URLs

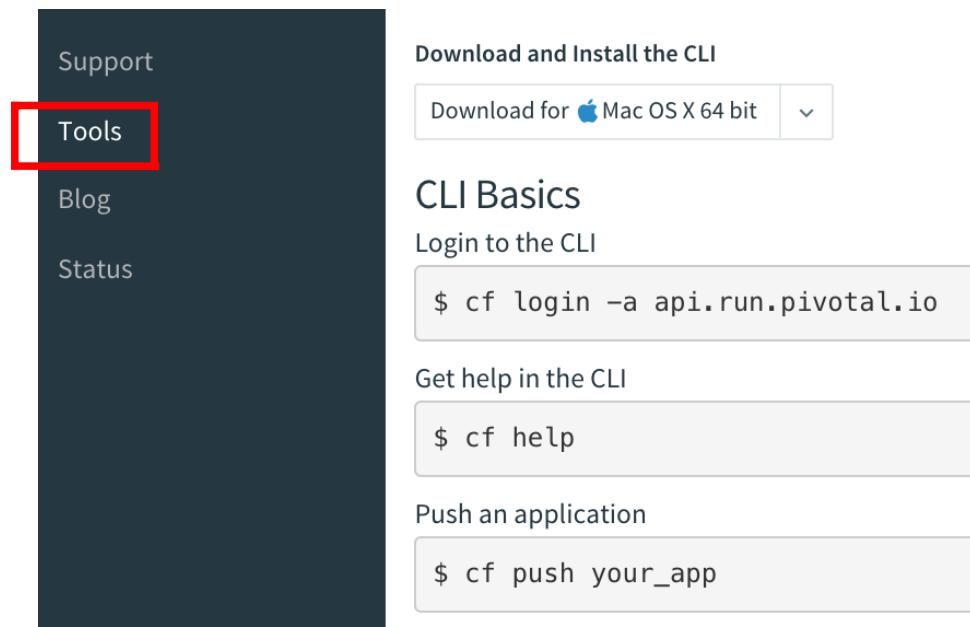
For simplicity, most examples in this section show PWS URLs

- System & App domains defined when CF was installed
- *If using PWS*
  - System domain: `run.pivotal.io`
  - API Endpoint: `api.run.pivotal.io`
  - Apps Manager: `console.run.pivotal.io`
  - Apps domain: `cfapps.io`
- *Your own CF installation*
  - System domain: `<your-cf-system-domain>`
  - API Endpoint: `api.<your-cf-system-domain>`
  - Apps Manager: `login.<your-cf-system.domain>`
  - Apps domain: `<your-cf-apps-domain>`



# Finding the API Endpoint URL

- URL of Cloud Controller in your Cloud Foundry instance
  - On Apps Manager home-page on first login
  - Or click *Tools*
  - Shows how to run **cf login**, including the API Endpoint



# DO NOW – Login

- Perform these steps on your computer:
  - Login with **cf login** command
    - Specify CF instance using **-a <API-URL>**
      - For PWS: **-a api.run.pivotal.io**
    - Specify email / password
    - If prompted, select an organization and space

```
$> cf login -a <API-URL> -u <your-email-or-username>  
API endpoint: api.run.pivotal.io  
...
```

- Firewall issues?  
<http://docs.cloudfoundry.org/devguide/installcf/http-proxy.html>

# The `.cf` Directory

- `cf` creates a `.cf` directory in your *home* directory
  - Remembers your CF API Endpoint
    - Don't need to specify `-a` option at next login

```
C:>dir .cf
Volume in drive C is VMWARE-QVTRTJ6E
Volume Serial Number is E8C5-12AF

Directory of C:\Users\paulchapman\.cf

12/21/2015  04:51 PM    <DIR>          .
12/21/2015  04:51 PM    <DIR>          ..
07/01/2017  02:58 AM            1,070 config.json
12/21/2015  04:51 PM    <DIR>          plugins
                           1 File(s)           1,070 bytes
                           3 Dir(s)   3,632,525,312 bytes free
```

# DO NOW – Viewing .cf folder

- Perform these steps on your computer:
  - Find the `.cf` folder / directory on your computer
    - You won't (yet) have all the files shown on previous slide
  - Open the `config.json` file, observe the contents
    - Note the tokens
- To override the location, set `CF_HOME`

# DO NOW – Current Target

- When you first login you see output like this:
  - Notice it shows *current* organization and space
  - At any time, run `cf target` to get same information

```
API endpoint: https://api.run.pivotal.io (API version: 2.6.0)
User:          pchapman@pivotal.io
Org:          pivotaledu
Space:        development
```

- By default your organization only has one space
  - Development
- **Note:** On PWS you are setup as your own organization



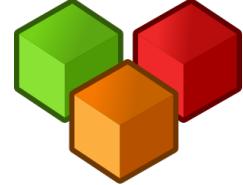
# DO NOW – Viewing Organization(s)

- Commands
  - **cf orgs** All orgs for current user
  - **cf org <org-name>** Shows specified org

```
>$ cf org pivotaledu
Getting info for org myorg as user@somedomain.com
OK

pivotaledu:
  domains:      cfapps.io
  quota:        paid (10240M memory limit, Unlimited instance
                  memory limit, 1000 routes, -1 services,
                  paid services allowed)
  spaces:       development, production, staging
  space quotas:

>$
```



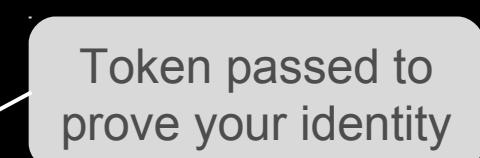
# Managing Spaces

- To see all the spaces in an organization
  - `cf spaces`
- Create a *new* space (in current organization by default)
  - `cf create-space <space-name>`
  - `cf create-space <space-name> -o <org-name>`
- Use `target` command to *change* space (or organization)
  - `cf target -s <space-name>`
  - `cf target -o <org-name>`

# Do NOW – Seeing cf Work

- Makes requests of Cloud Foundry using REST HTTP
- To see this use **cf -v <command>**

```
C:> cf -v orgs
REQUEST: [2017-06-11T01:24:22+10:00]
GET /v2/info HTTP/1.1
Host: api.run.pivotal.io
Accept: application/json
Authorization: [PRIVATE DATA HIDDEN]
Connection: close
Content-Type: application/json
User-Agent: cf/6.27.0+d26b32d.2017-06-08 (go1.8.3; 386 windows)
```



Token passed to prove your identity

# Response from Cloud Foundry

**RESPONSE:** [2017-06-11T01:32:17+10:00]

HTTP/1.1 200 OK

Connection: close

Content-Length: 3050

Content-Type: application/json; charset=utf-8

Date: Sat, 10 Jun 2017 15:32:15 GMT

Server: nginx

X-Content-Type-Options: nosniff

X-Ratelimit-Limit: 20000

X-Ratelimit-Remaining: 19997

X-Ratelimit-Reset: 1497112254

X-Vcap-Request-Id: fbbbc027-901b-4a23-6e76-dfefc1a92995

X-Vcap-Request-Id: 8b57571a-d0c1-491a-62e3-2496d50e48ce::c36971b5...

{

  "total\_results": 1,

  "total\_pages": 1,

  "prev\_url": null,

  "next\_url": null,

  "resources": [ { ... *organization data here* } ]

}

JSON data  
returned

# Roadmap

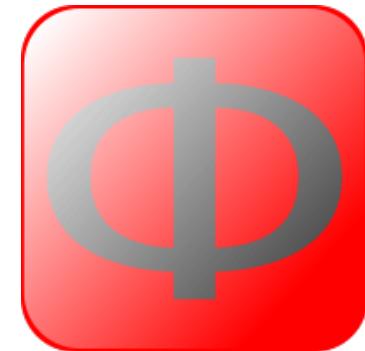
- Get Setup
- Login
- **Deploying an Application**
- Managing Application Instances

# Deploy Using the CLI

- You need a deployable application
  - For example with Java: a jar or war
    - Ant, Maven or Gradle build-tools can make it for us
    - Cloud Foundry doesn't care how you build your application
  - Other languages (Ruby, Node.js, etc.): the source will do

# The cf push Philosophy

- Onsi Fakhouri (Cloud Foundry PM)
  - *Here is my source code*
  - *Run it on the cloud for me*
  - *I do not care how*
- The architecture of CF is fascinating
  - And we *will* cover it
  - But ultimately irrelevant
- I just want to push an application
  - I no longer need to know
    - *how that happens, how it is packaged or how it is run*



*Haiku*

# Deploy (*push*) to Cloud Foundry

- Deploy by running **cf push <name-of-your-app>**
  - Many options
    - -i Number of instances
    - -m Memory limit (e.g. 256M, 1024M, 1G)
    - -n hostName (application subdomain)
    - -p Local path to app directory, jar, war, zip file ...
    - ... Others will be covered later
- Your application appears in Cloud Foundry under the name you specify here

# Domains and URLs

- Every CF instance is assigned a domain at installation
  - Known as the *Apps Domain*
  - For PWS this is ***cfapps.io***
- When you deploy, your application gets a unique route (URL) to access it: **hostname + app domain name**
  - By default, hostname = application name
  - Make sure hostname is **unique**
    - *cf push* returns an HTTP 400 error if not
- PWS example:
  - ***cf push spring-music ...***
  - gets route: ***spring-music.cfapps.io***

# Examples of Using cf push

- Fully specified (recommended)

```
cf push spring-music -i 1  
                  -m 512M  
                  -n spring-music-678  
                  -p build/libs/spring-music.war
```

- Deploys war file (specify path if needed)
- 1 instance, 512M memory
- Name: **spring-music**
  - Appears as **spring-music** in Cloud Foundry
- Hostname: **spring-music-678**
  - Creates URL (PWS): **spring-music-678.cfapps.io**

*Specify unique sub-domain  
by adding numbers, initials ...*



Or use  
**--random-route**

# What Happens ?

- **cf** connects to Cloud Foundry using your credentials
- It 'pushes' your application to CF and tells it to deploy it
  - The whole application is uploaded – takes a while
  - CF “stages” your application
    - Recognizes Java WAR file, prepares a “droplet” with a JRE and Tomcat server
    - “Droplet” is deployed to a container and starts running
    - All requests to the *Deployed URL* route to your application
- Whole process logged on screen
  - See next 3 slides

# What Happens - 1

URL: `spring-music-678.cfapps.io`

```
cf push spring-music -n spring-music-678 -i 1 -m 512M  
-p pre-built/spring-music.war
```

```
> cf push spring-music -n spring-music-678 -p build/libs/spring-music.war -i 1 -m 512M
```

```
Creating app spring-music in org your-org / space development as your-id@company.io...  
OK
```

Updates CF metadata  
(app name, instances, memory)

```
Using route spring-music-678.cfapps.io
```

Establish & bind route

```
Binding spring-music-678.cfapps.io to spring-music...
```

Uploads war

```
Uploading spring-music...
```

```
Uploading app files from: pre-built/spring-music.war
```

```
Uploading 574.8K, 95 files
```

```
Done uploading
```

```
OK
```

```
Starting app spring-music in org your-org / space development as your-id@company.io...
```

```
...
```

Next ...

# What Happens - “Staging”

CF must prepare the app before its first run

```
...  
Starting app spring-music in org your-org / space development as your-id@company.io...  
OK  
  
Downloading dotnet_core_buildpack_beta...  
Downloading java_buildpack...  
... lots more buildpacks ...  
----> Java Buildpack Version: v2.7.1 | https://github.com/cloudfoundry/java-buildpack#fee275a  
----> Downloading Open Jdk JRE 1.8.0_XX from https://download.run.pivotal.io/.../openjdk-1.8.0\_XXX.tar.gz (6.1s)  
      Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.3s)  
----> Downloading Open JDK Like Memory Calculator 2.0.0.2_RELEASE from https://java-buildpack.cloudfoundry.org/memory-calculator/trusty/x86\_64/memory-calculator-... (found in cache)  
      Memory Settings: -Xss228K -Xmx317161K -XX:MaxMetaspaceSize=64M -Xms317161K ...  
----> Downloading Spring Auto Reconfiguration 1.7.0_RELEASE from  
https://download.run.pivotal.io/auto-reconfiguration/auto-reconfiguration-1.7.0\_RELEASE.jar (0.2s)  
----> Downloading Tomcat Instance 8.XXX from https://download.run.pivotal.io/tomcat/tomcat-8.XXX.tar.gz (1.1s)  
      Expanding Tomcat to .java-buildpack/tomcat (0.1s)  
----> Downloading Tomcat Lifecycle Support 2.4.0_RELEASE from  
https://download.run.pivotal.io/tomcat-lifecycle-support/tomcat-lifecycle-support-XXX.jar (0.0s)  
...  
----> Uploading droplet (73M)  
...  
  
Next ...      creates “Droplet”, uploads to blobstore      Buildpack obtains Tomcat      Buildpack configures Java      “Buildpack” selected and executed      Reconfigure Spring for cloud environment
```

# What Happens - “Start”

```
...  
0 of 1 instances running, 1 starting  
0 of 1 instances running, 1 starting  
1 of 1 instances running
```

Cloud Foundry runs the “Droplet” on a “container”

App started

Command that was executed

OK

```
App spring-music was started using this command `JAVA_HOME=$PWD/.java-buildpack/open_jdk_jre JAVA_OPTS="--  
Djava.io.tmpdir=$TMPDIR -XX:OnOutOfMemoryError=$PWD/.java-buildpack/open_jdk_jre/bin/killjava.sh  
-Xmx382293K -Xms382293K -XX:MaxMetaspaceSize=64M -XX:MetaspaceSize=64M -Xss995K  
-Daccess.logging.enabled=false -Dhttp.port=$PORT" $PWD/.java-buildpack/tomcat/bin/catalina.sh run`
```

Showing health and status for app **spring-music** in org **your-org** as **your-id@company.io**...

OK

```
requested state: started  
instances: 1/1  
usage: 512M x 1 instances  
urls: spring-music-678.cfapps.io  
last uploaded: Tue Mar 17 17:58:35 UTC 2015  
buildpack: java-buildpack=v3.9-offline-https://github.com/cloudfoundry/java-buildpack.git#b050954  
open-jdk-like-jre=1.8.0_101 open-jdk-like-memory-calculator=2.0.2_RELEASE spring-auto...
```

Health Check

	state	since	cpu	memory	disk
#0	running	2015-03-17 01:59:35 PM	0.0%	474.4M of 512M	150.3M of 1G

Done! 1 application instance running on **spring-music-678.cfapps.io**

# Application State and Logs

- Run `cf apps`

```
> cf apps
Getting apps in org pivotaledu / space development as kkrueger@pivotal.io...
OK

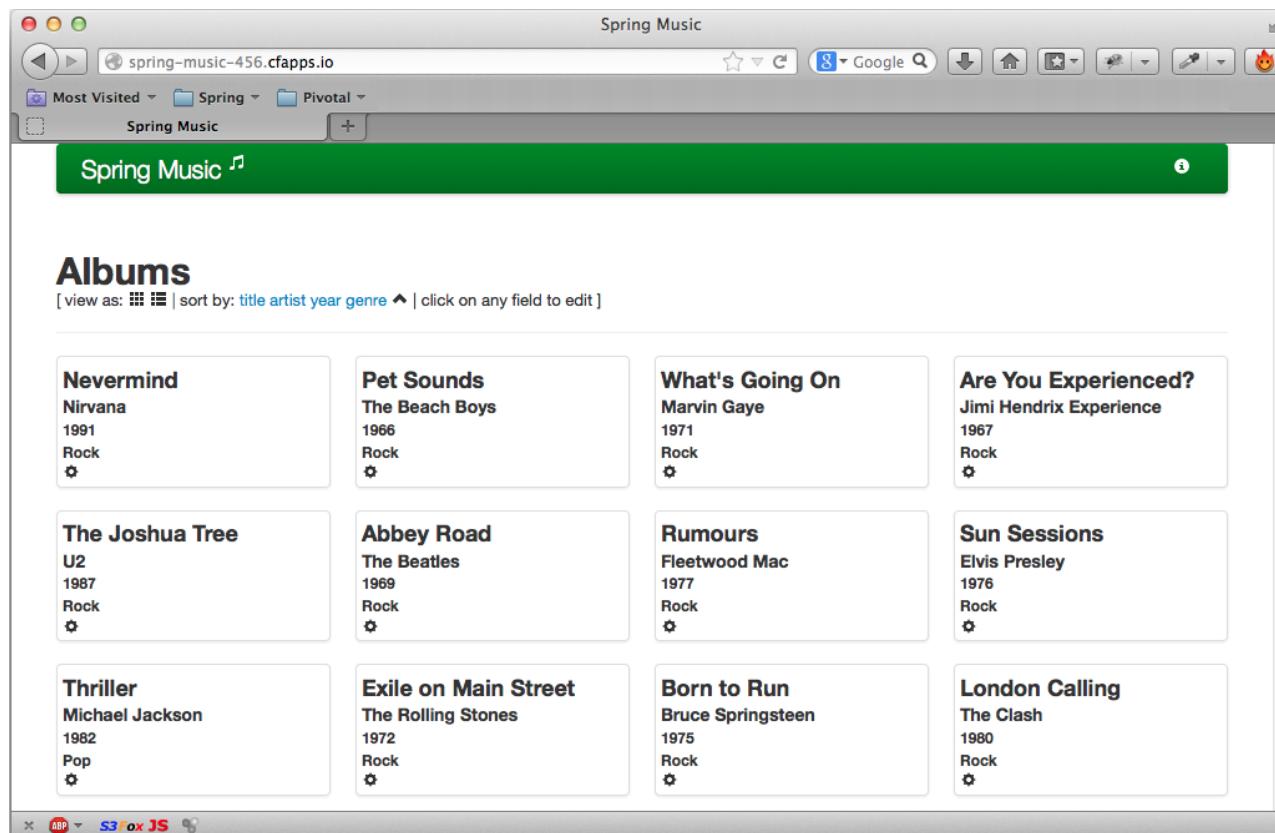
name          requested state    instances   memory   disk      urls
spring-music  started           1/1        512M     1G       spring-music-678.cfapps.io
```

- `cf logs spring-music`

```
> cf logs spring-music
Connected, tailing logs for app spring-music in org pivotaledu / space development as
kkhueger@gopivotal.com...
2014-06-07T23:01:47.68-0400 [RTR]      OUT spring-music-678.cfapps.io -
[08/06/2014:03:01:47 +0000] "GET /assets/js/status.js HTTP/1.1" 200 844 "http://spring-
music-678.cfapps.io/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5)
AppleWebKit/537.73.11 (KHTML, like Gecko) Version/6.1.1 Safari/537.73.11"
10.10.66.34:64401 vcap_request_id:73037523-63ef-498f-6cd8-d3b48fe69e84
response_time:0.003693009 app_id:314f0434-d2c9-446c-ab4a-6c310878ca80
2014-06-07T23:01:48.47-0400 [RTR]      OUT spring-music-678.cfapps.io -
[08/06/2014:03:01:48 +0000] "GET /assets/templates/header.html HTTP/1.1" 200 1060
"http://spring-music-678.cfapps.io/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5)
AppleWebKit/537.73.11 (KHTML, like Gecko) Version/6.1.1 Safari/537.73.11"
10.10.66.34:64324 vcap_request_id:39fbb3f2-46fb-4bd7-78d6-8994fafade9f
response_time:0.004132254 app_id:314f0434-d2c9-446c-ab4a-6c310878ca80
```

# See The Application Running

- Open a browser window to [spring-music-678.cfapps.io](http://spring-music-678.cfapps.io)



Pivotal™

# Configuring a Deployed Application

- Change the number of instances
  - `cf scale <app> -i <new-value>`
  - Two instances: `cf scale spring-music -i 2`
  - New instances added, or some existing instances stopped
- Change the memory allocation
  - `cf scale <app> -m <new-value>`
  - 1024M: `cf scale spring-music -m 1024M`
  - Requires a restart to take effect

# Stopping and Starting

## `cf stop`

- Sends SIGTERM message to application
- Sends SIGKILL 10 seconds later if still running

## `cf start`

- Starts existing application

## `cf restart`

- `cf stop` followed by `cf start`

## `cf restage`

- Repeats the staging process, and starts the app.
- Useful when environment variables / bound services change
  - (Covered later)



# Adding / Removing Routes

- Add a new domain mapping
  - `cf map-route <app> <domain> -n <hostname>`
  - `cf map-route spring-music cfapps.io -n mymusic`
    - `mymusic.cfapps.io` also maps to `spring-music`
- Remove mapping
  - `cf unmap-route <app> <domain> -n <hostname>`
  - `cf unmap-route spring-music cfapps.io -n spring-music-678`
    - `spring-music-678.cfapps.io` no longer maps to `spring-music`

# Cleaning Up Unused Routes



- Routes tend to accumulate over time
  - Applications in other Orgs / Spaces cannot use these routes
- Find all other routes used in a space:
  - `cf routes`
- Remove route:
  - `cf delete-route`
- Very Useful! Remove unused routes:
  - `cf delete-orphaned-routes`

# Roadmap

- Get Setup
- Login
- Deploying an Application
- **Managing Application Instances**

# Apps Manager

- Login to Cloud Foundry using your web-browser
  - Pivotal Web Services: <http://run.pivotal.io>
    - Your Cloud Foundry instance URL will be different
    - `login.<your-cf-domain>`
  - Use the username and password you registered with
  - Our new application should show green in the Apps Manager
- Next slide ...

**NOTE:** Only Pivotal CF comes with the Apps Manager  
Open Source Cloud Foundry *does not*

# Apps Manager Home Page

- At a glance view of all your applications
  - Shows current space

The screenshot shows the Pivotal Apps Manager interface. On the left, there's a sidebar with a 'P' logo, 'Pivotal Web Services' text, 'ORG' dropdown set to 'pivotaledu', and a 'SPACES' section with 'development' selected (highlighted in green). The main area shows the 'pivotaledu > development' space. It displays a summary: 1 Running, 0 Stopped, 0 Crashed. Below this, there are tabs for 'App (1)', 'Services', 'Security', and 'Settings'. The 'App (1)' tab is active, showing a table with one row for 'spring-music'. The table columns are NAME, INSTANCES, MEMORY, LAST PUSH, and ROUTE. The 'spring-music' row shows 'Running' status, 1 instance, 512MB memory, 17 hours ago last push, and the URL <http://spring-music-pc.cfapps.io>. A red arrow points from the text 'Click to select different space' to the 'development' link in the sidebar. Another red arrow points from the text 'Click application name to see its dashboard (next slide)' to the 'spring-music' application name. A third red arrow points from the text 'URL of your application' to the application URL in the table.

Click to select different space

Click application name to see its dashboard (next slide)

URL of your application

NAME	INSTANCES	MEMORY	LAST PUSH	ROUTE
spring-music ● Running	1	512MB	17 hours ago	<a href="http://spring-music-pc.cfapps.io">http://spring-music-pc.cfapps.io</a> ➔

# Application Dashboard

Application state

APP  
spring-music

Stop

Restart

Status

Running

LOGS

Go to App Home Page

View App ↗

Overview Services Route (1) Env Variables Logs Settings Buildpack: java-buildpack=v3.9-o...

Select tabs for more details

Events Last Push: 03:08 PM 10/03/16

Started app pchapman@gopivotal.com 10/03/2016 at 09:38:16 AM UTC

Updated app pchapman@gopivotal.com 10/03/2016 at 09:37:52 AM UTC

Renamed app to spring-music pchapman@gopivotal.com 10/03/2016 at 09:37:42 AM UTC

Renamed app to spring-music pchapman@gopivotal.com 10/03/2016 at 09:37:01 AM UTC

Scaling

Cancel Scale App

Instances: 1 Memory Limit: 512 MB Disk Limit: 1 GB

Scaling control

Status View in PCF Metrics ↗

# ▲	STATUS	CPU	MEMORY	DISK	UPTIME
0	● Runn... 0%	355.57 MB	154.55 MB	5 hr 14 ...	

Get runtime info

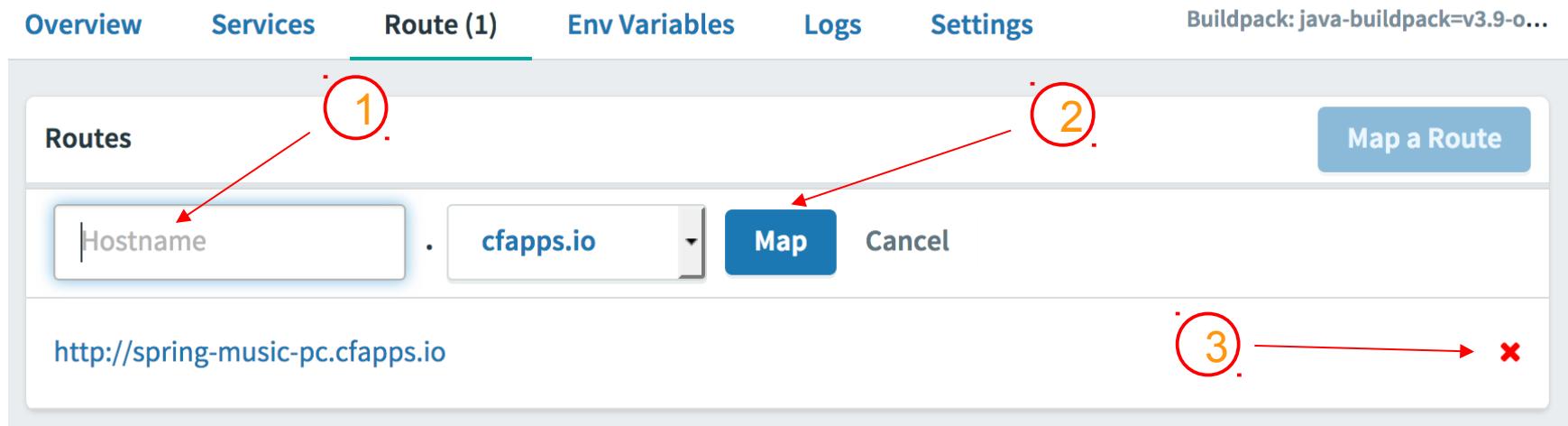
Recent events

Application Information

The dashboard provides a central hub for managing and monitoring the 'spring-music' application. It includes a top navigation bar with tabs for Overview, Services, Route (1), Env Variables, Logs (selected), and Settings. A status bar at the bottom indicates the buildpack used: 'Buildpack: java-buildpack=v3.9-o...'. The main area features a summary card with recent events, scaling controls, and detailed runtime metrics. Red annotations highlight specific features: 'Recent events' points to the event log; 'Scaling control' points to the scaling interface; 'Application Information' points to the runtime metrics table; and 'Get runtime info' points to the 'View in PCF Metrics' link.

# Change Mapped URL

- Enter new domain (1)
  - Remember to make it *unique*
- Click MAP (2)
- To remove a mapping (3)
  - Click **red cross** at far right of same line



# Monitoring Instances

- The very bottom panel shows all your instances
  - Provides statistics
  - Updated live (slight time-lag)

Status						<a href="#">View in PCF Metrics</a>
# ▲	STATUS	CPU	MEMORY	DISK	UPTIME	
0	● Running	0%	362.85 MB	154.55 MB	5 hr 35 min	
1	● Crashed	0%	0 Bytes	0 Bytes	0 min	

# Summary

- After completing this lesson, you should have learned:
  - How to Deploy an application to CloudFoundry using CLI
  - Managing application instances using Apps Manager



Pivotal™

# Lab

Push an existing application to Pivotal CF

# PCF Architecture

Production features

PCF Internal Architecture

# Learning Objectives

- After completing this lesson, you should be able to:
  - List the challenges of Cloud Native Apps
  - Describe Cloud Foundry's Elastic Runtime Architecture



# Agenda

- **Cloud Native Apps**
- Elastic Runtime Architecture

# Fundamental Changes

*Cloud Native applications operate in a different environment*

# Distributed System

- Distributed systems are hard to build, test, manage, and scale

# “Ephemeral” (Temporary) Infrastructure

- Virtual machines and containers are temporary

# Immutable Infrastructure

- Updates to systems and applications are *not* done in-place
- New, updated instances are created *instead*

# Changes in App Design

*Architecting for the Cloud*

# The 12 Factor App

- Methodology for building web apps suitable for running on cloud platforms
  - Most are best practices for *any* development
  - Some specific to a Cloud deployment

<http://12factor.net/>

# The 12 Factor Design Patterns

Codebase

Dependencies

Config

Backing Services

Build, release, run

Processes

Port binding

Concurrency

Disposability

Dev/prod parity

Logs

Admin Processes

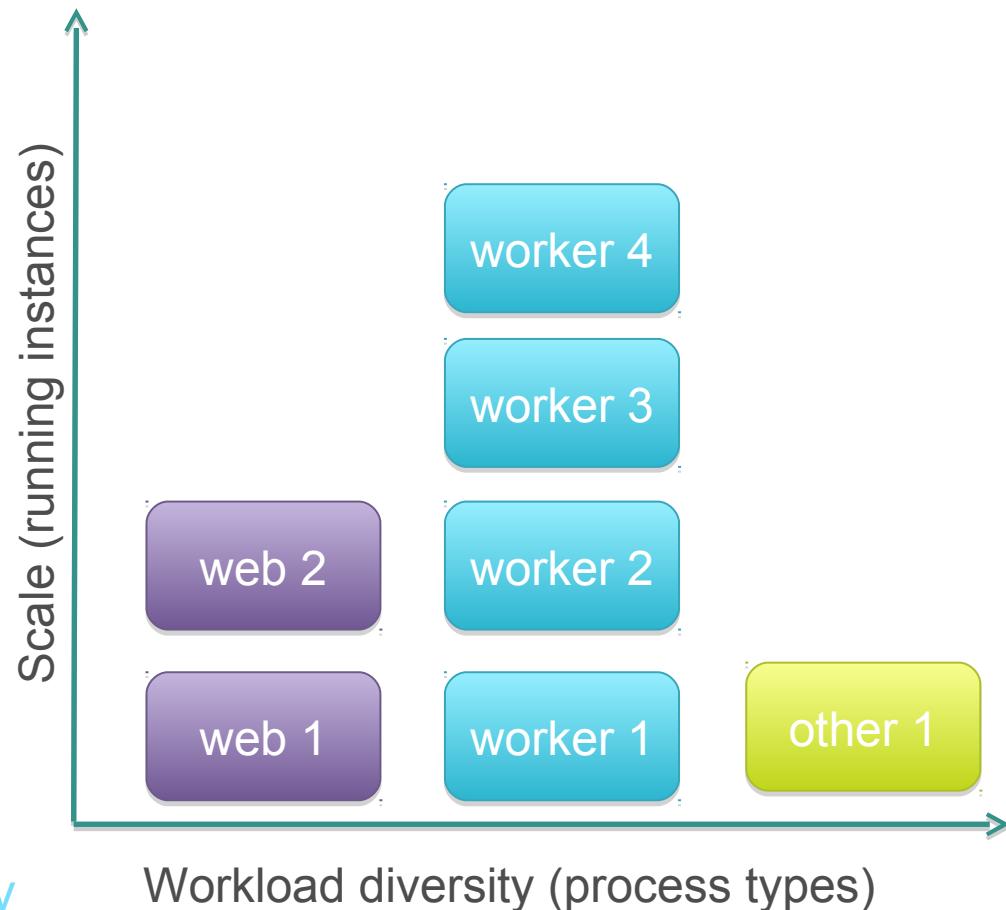
# Processes

- Execute the app as one or more stateless processes

<http://12factor.net/processes>

# Concurrency

- Scale *out* via the process model



<http://12factor.net/concurrency>

# Disposability

- Maximize robustness with fast startup and graceful shutdown

<http://12factor.net/disposability>

# Logs

- Treat logs as event streams

<http://12factor.net/logs>

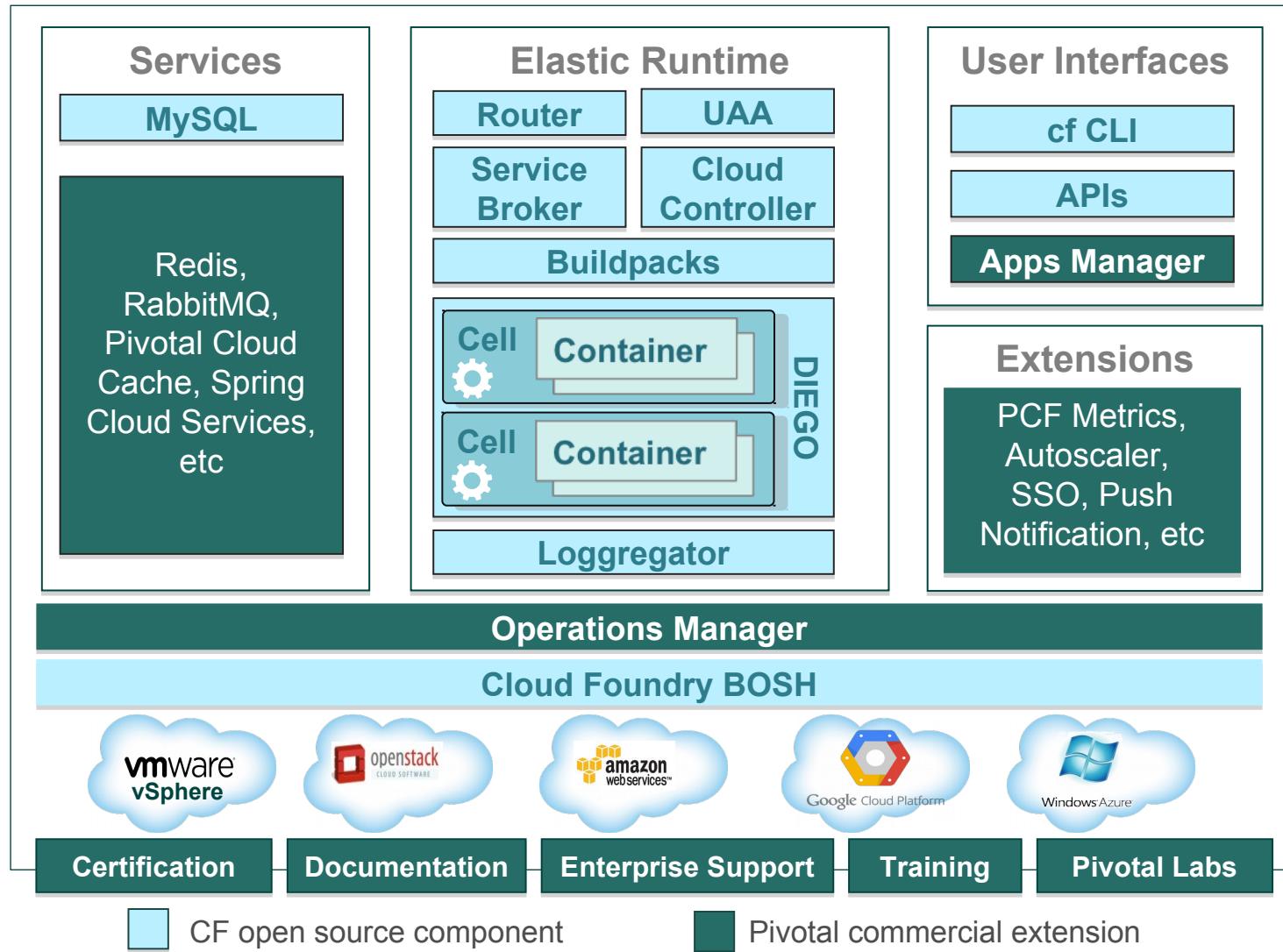
# Agenda

- Cloud Native Apps
- **Elastic Runtime Architecture**

# Elastic Runtime

- What is the Elastic Runtime?
- *The Elastic Runtime is Cloud Foundry*

# PCF Architecture



# Elastic Runtime Architecture

- Elastic Runtime Subsystems
  - Diego
  - Loggregator
  - Cloud Controller API
  - Routing
- Key Sequence Flows through the ER

Elastic Runtime  
=  
Cloud Foundry  
OSS

~ 30 VMs

# Diego

*Cloud Foundry's internal architecture*

<https://docs.pivotal.io/pivotalcf/concepts/diego/diego-architecture.html>

# Diego

- Schedules Tasks and Long-Running Processes (LRPs)
- **TASK**
  - Is guaranteed to run at most once.
  - **Example:** staging an application
- **LRP**
  - Typically represented as a web app
  - LRP<sup>s</sup> can have multiple instances
  - PCF will attempt to keep them running forever

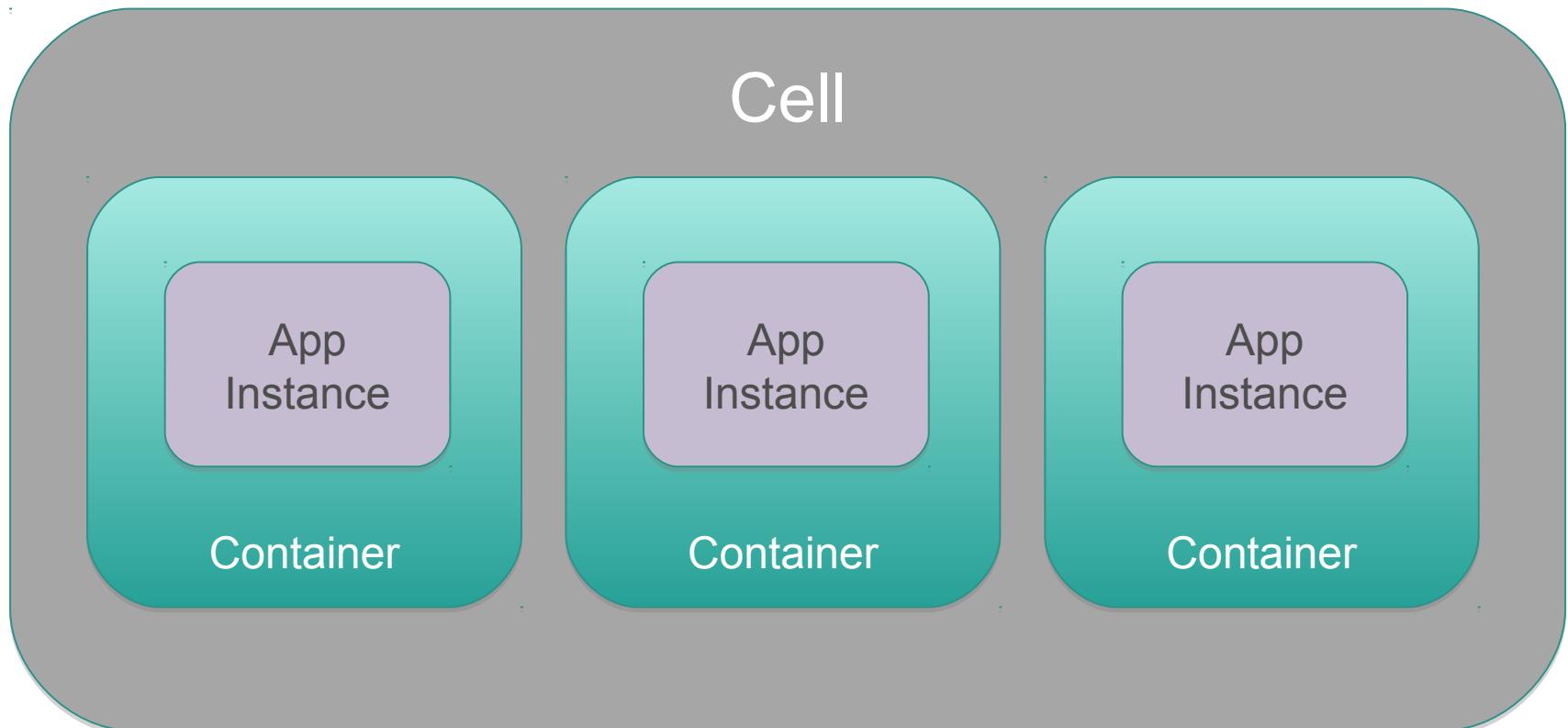
# Diego: Container

- An application instance is run within an immutable container



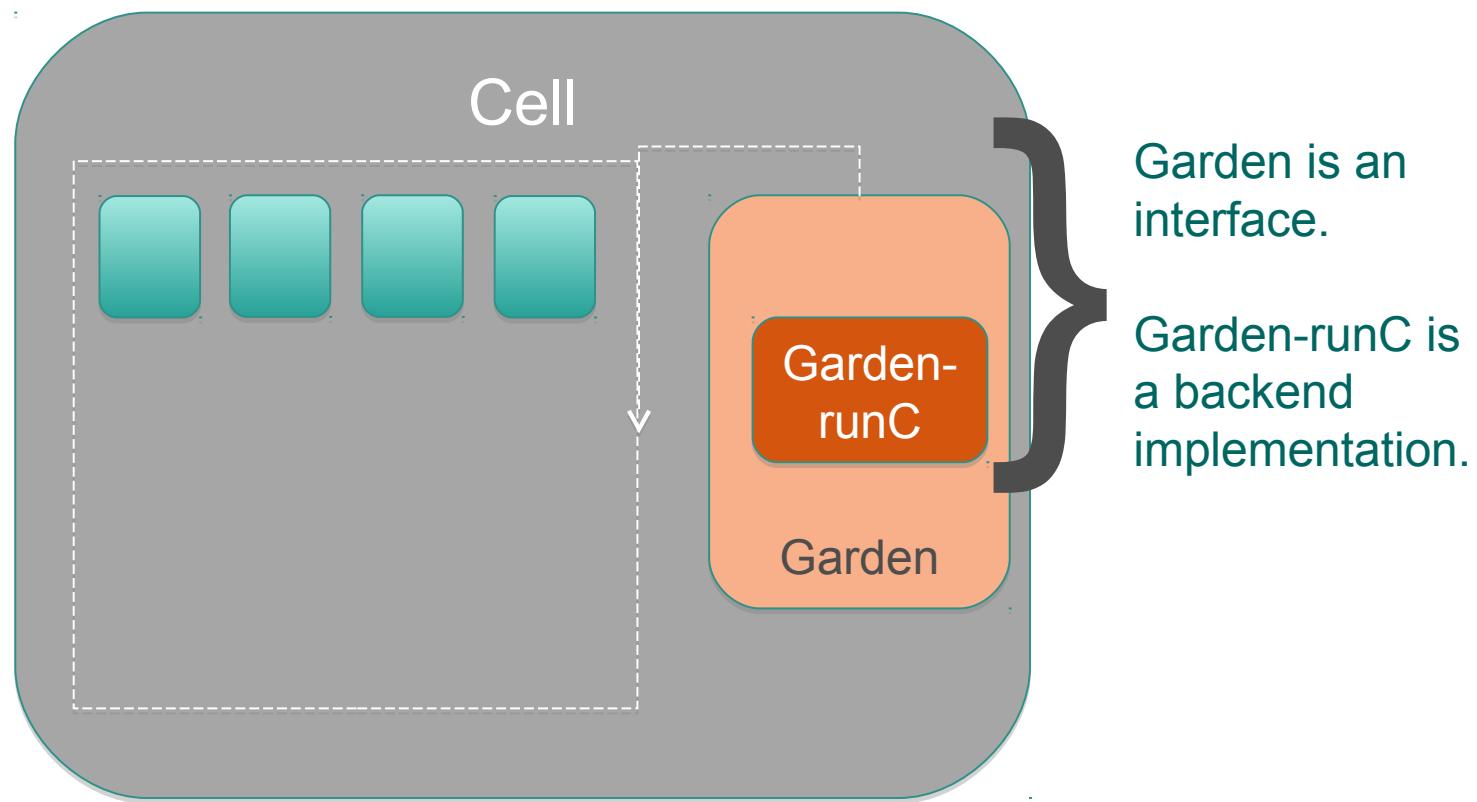
# Diego: Cell

- Containers are run within a Cell (Virtual Machine)



# Diego: Garden

- Containers are managed by Garden

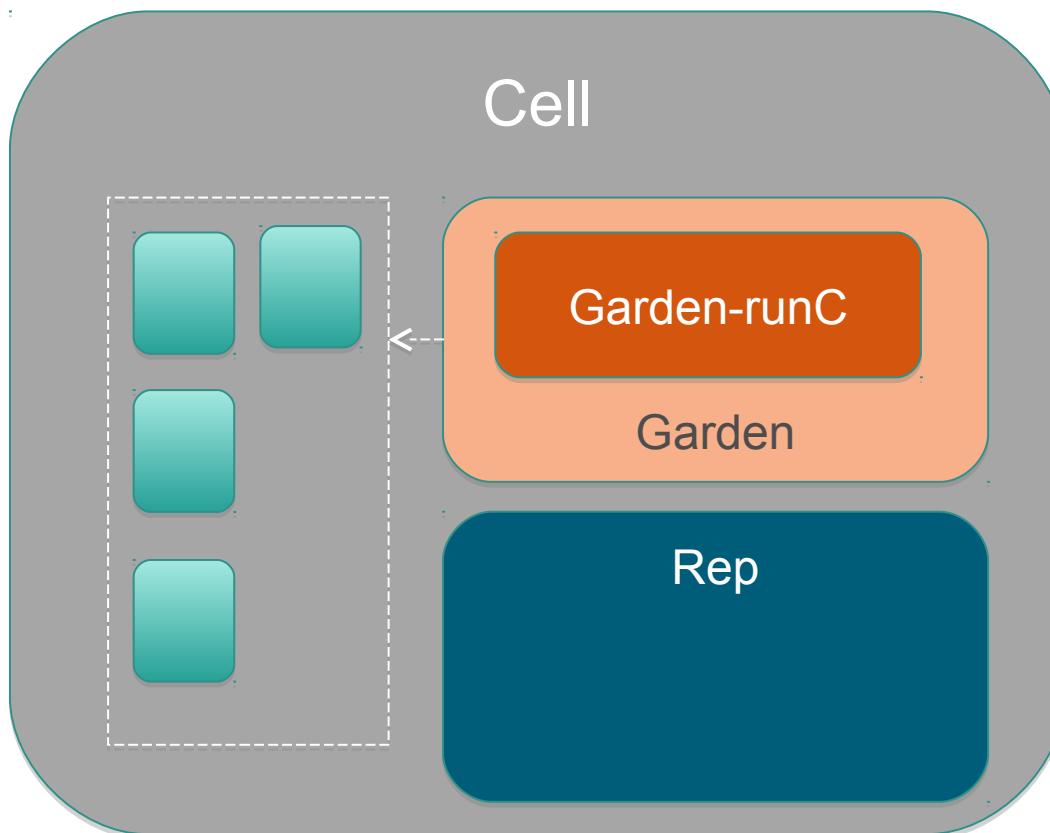


# Diego: Auction

- An auction is held to bid on executing a task or an LRP
- Why?
  - To ensure load spread evenly across Cells
  - Make best use of available resources

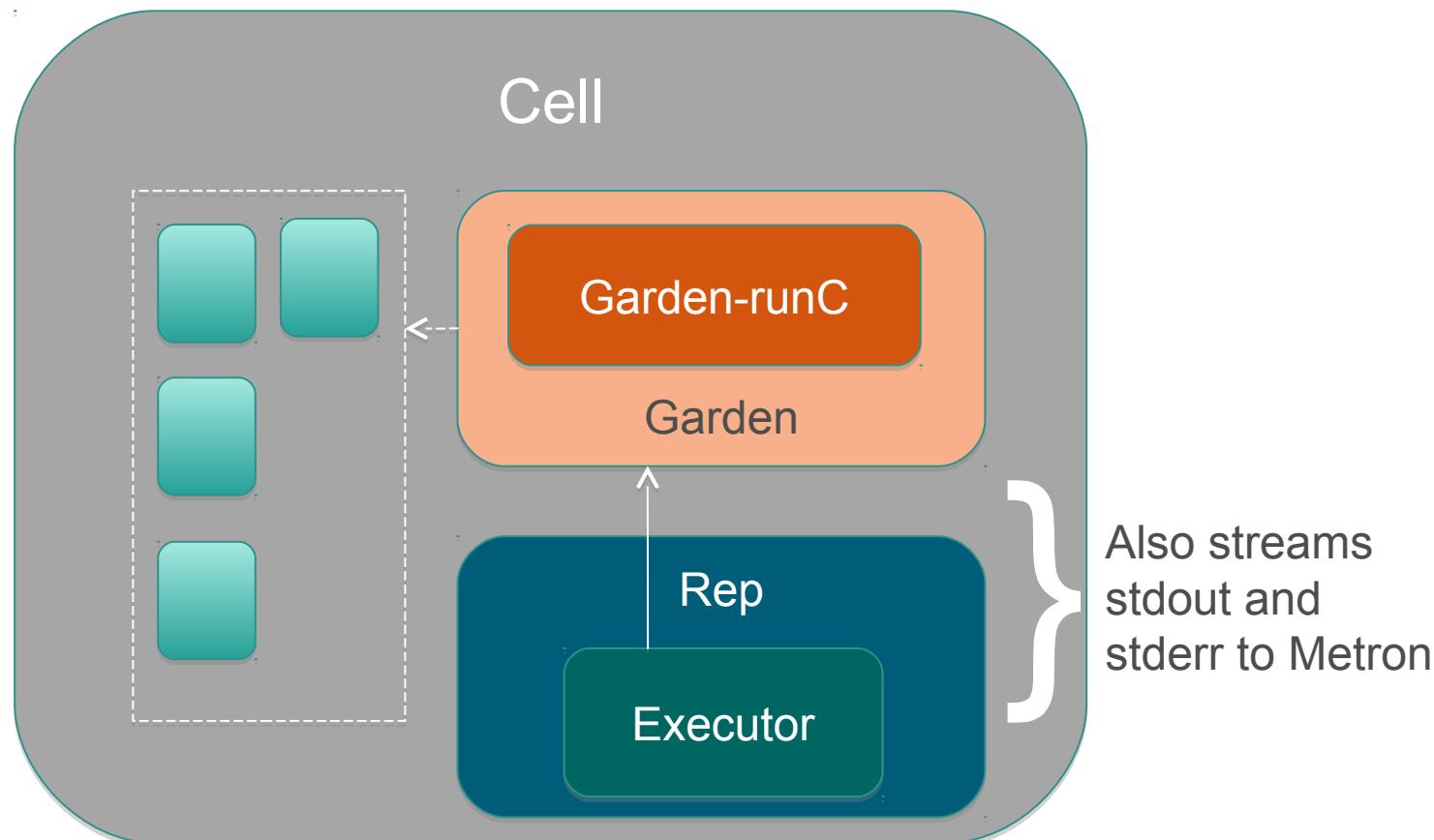
# Diego: Rep

- Represents the Cell in the BBS/auctions



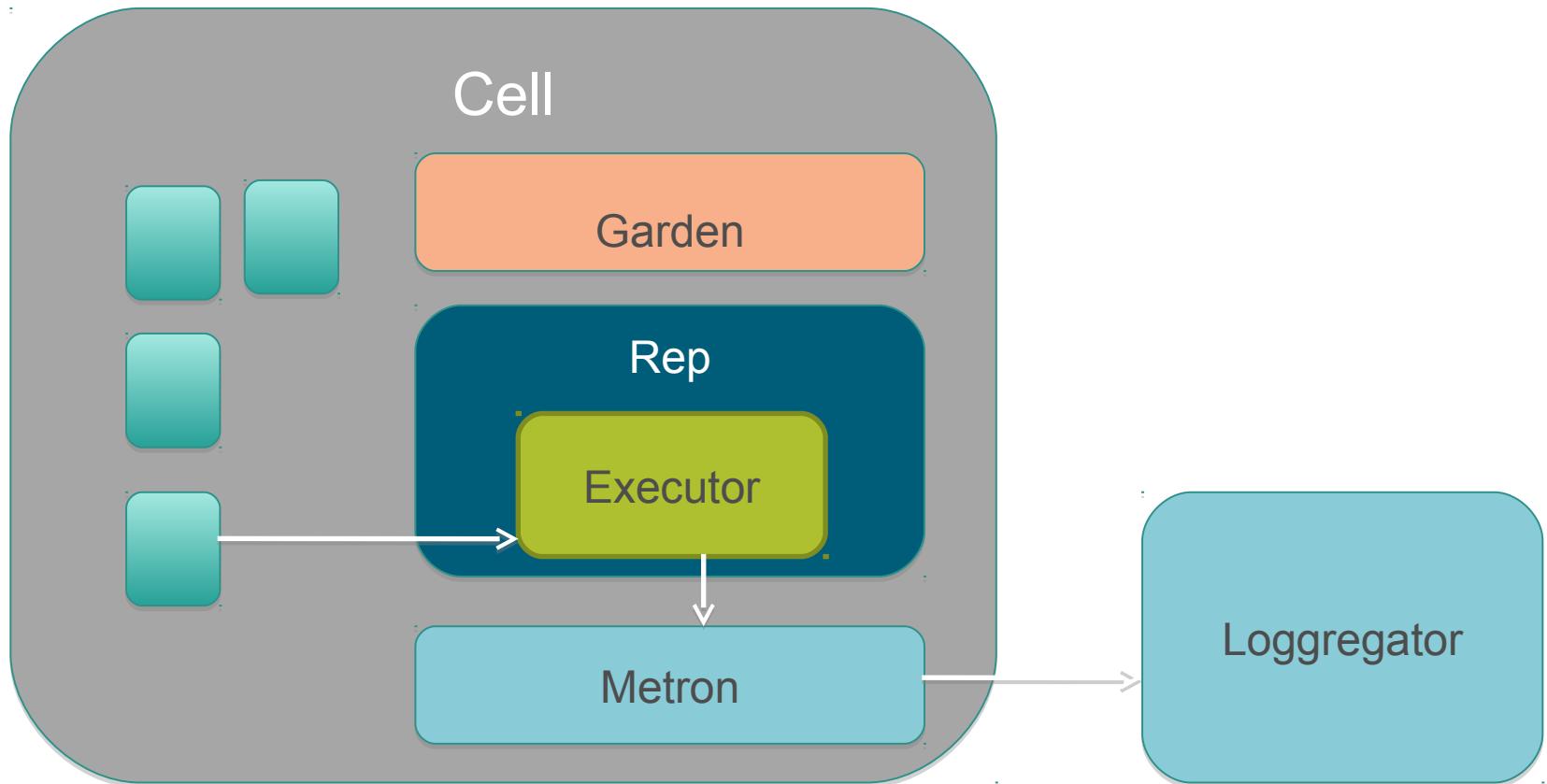
# Diego: Executor

- Manages container allocations on the cell



# Diego: Metron

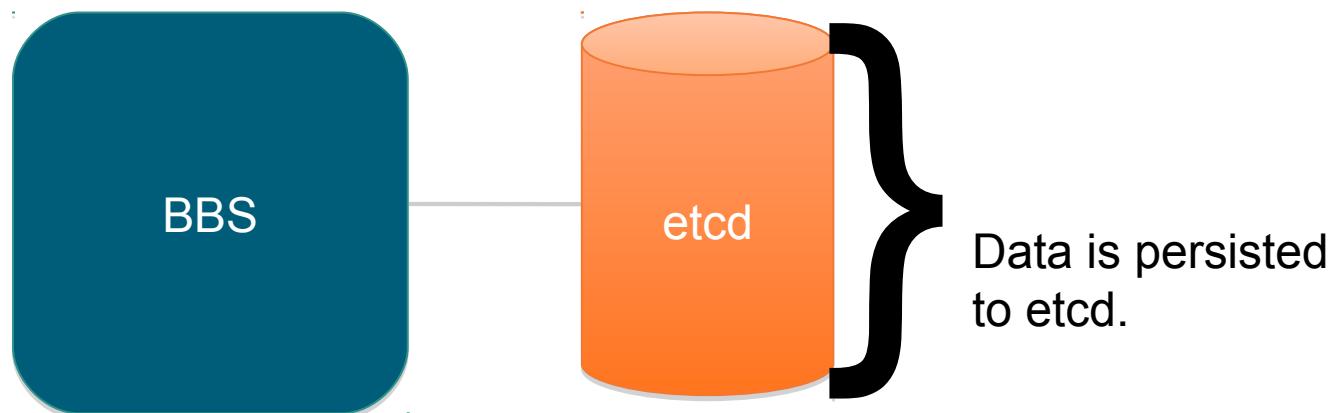
- Forwards logs to the Loggregator subsystem



Pivotal™

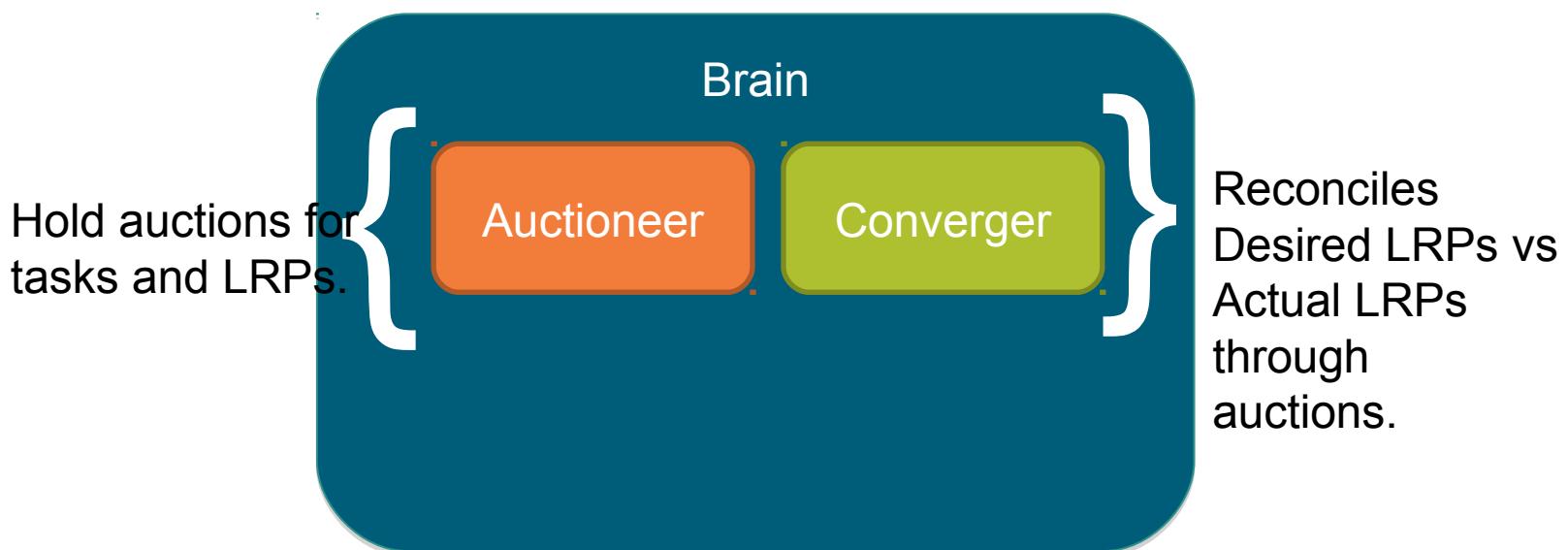
# Diego: BBS

- BBS (Bulletin Board System) is the API to access the Diego database for tasks and LRPCs



# Diego: Brain

- The Brain is composed of two components



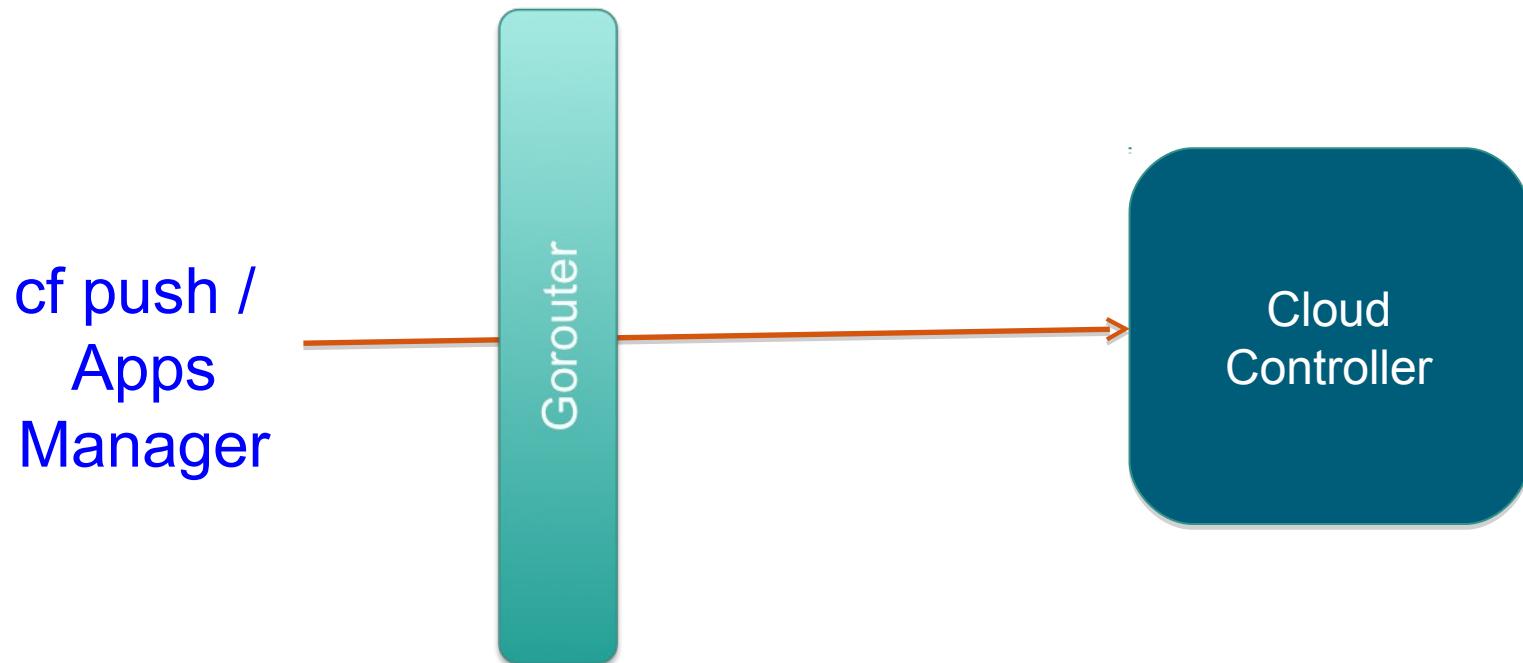
# Cloud Controller API

*Your interface to Cloud Foundry*

<https://docs.pivotal.io/pivotalcf/concepts/architecture/cloud-controller.html>

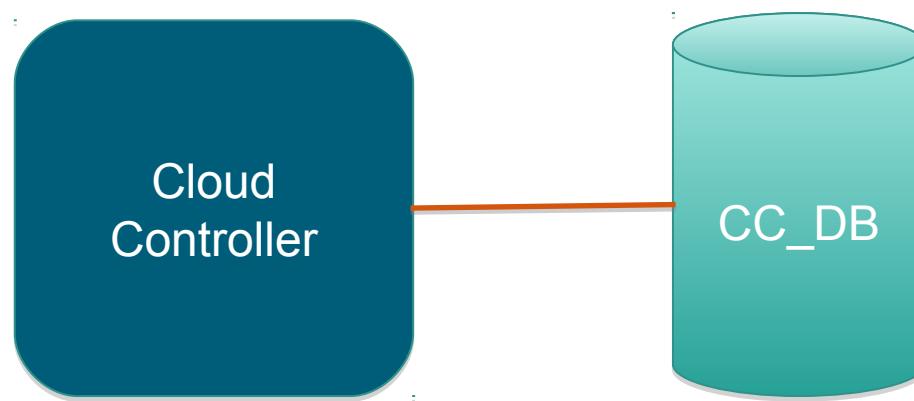
# CCAPI: Cloud Controller

- Cloud Controller exposes an API
  - For using and managing the Elastic Runtime



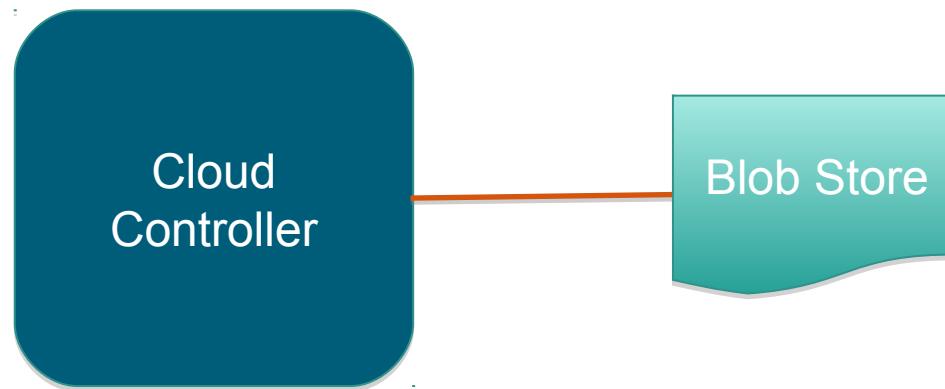
# CCAPI: Cloud Controller Database

- The Cloud Controller persists Org/Space/App data in the Cloud Controller Database.



# CCAPI: Blob Store

- The Cloud Controller persists app packages and droplets to the blob store.



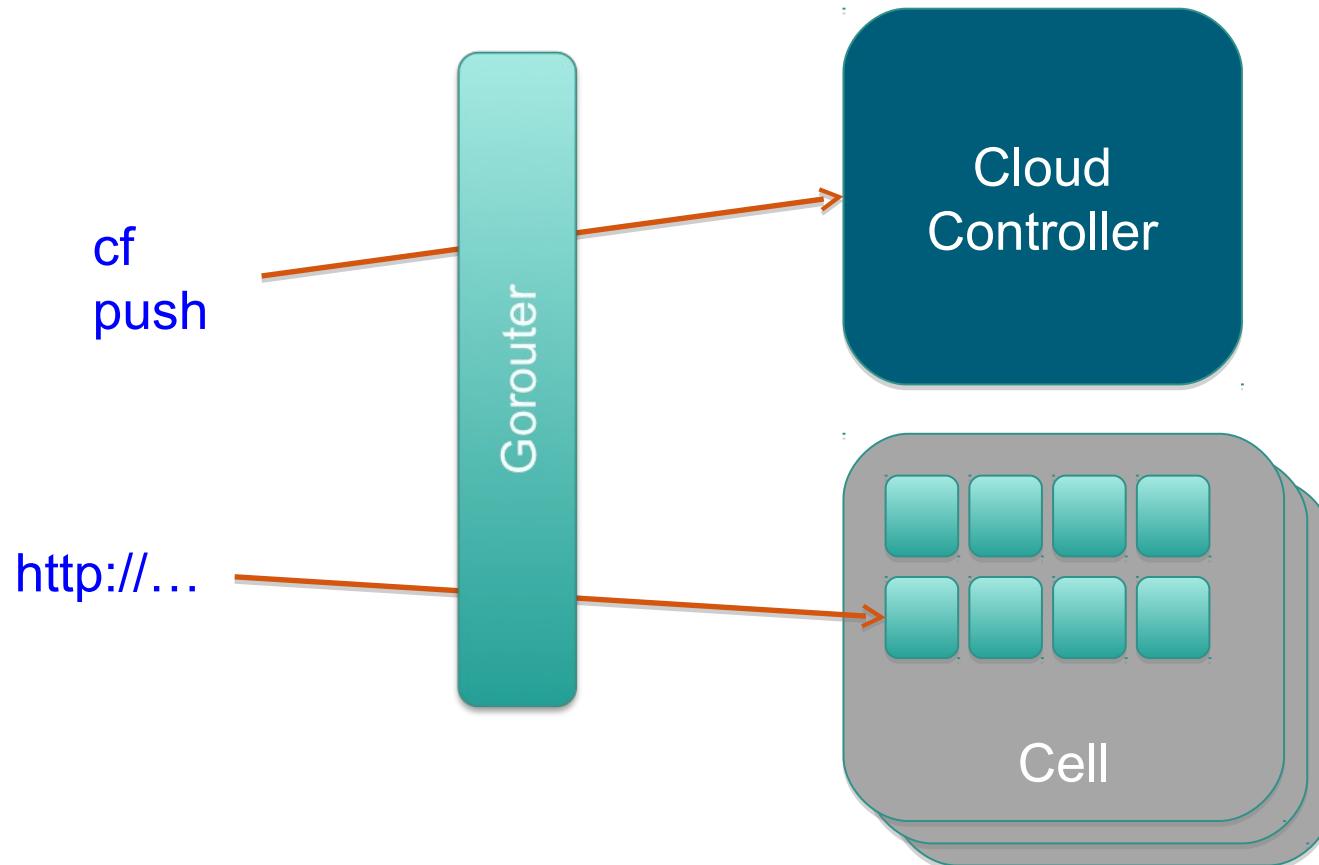
# Routing

*Allowing the outside to talk to the inside*

<https://docs.pivotal.io/pivotalcf/concepts/architecture/router.html>

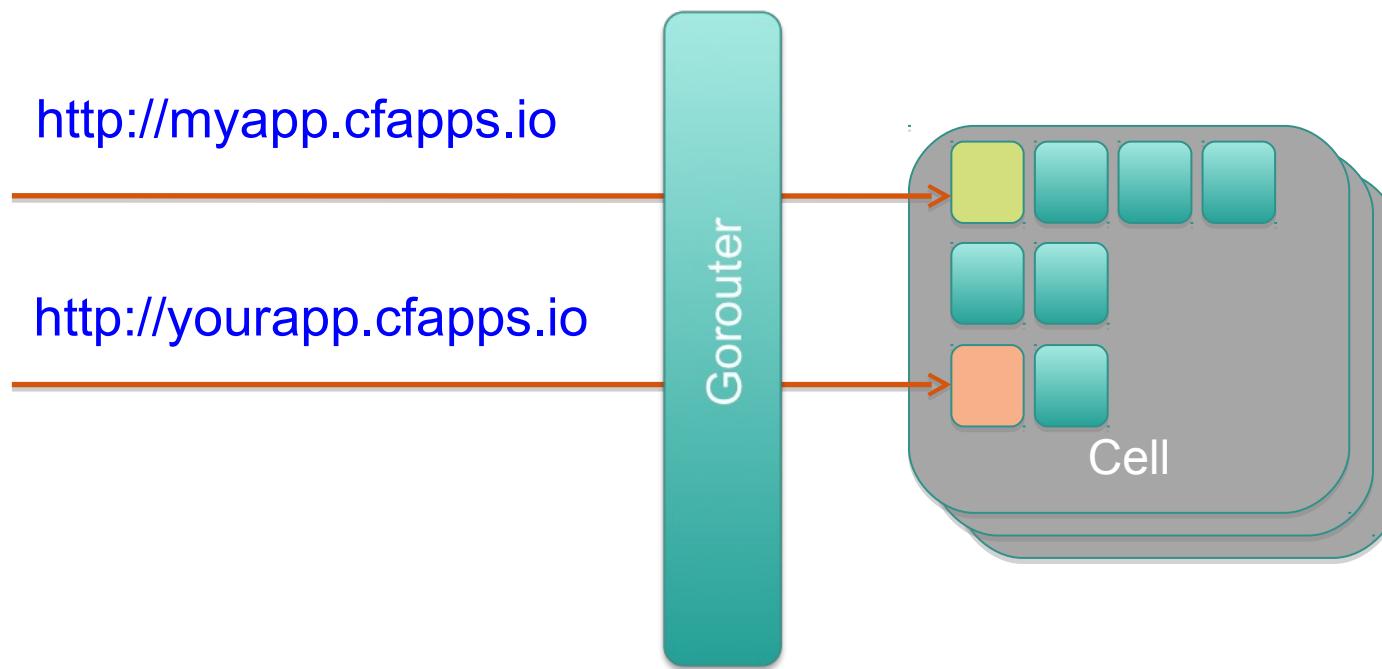
# Router

- The router directs traffic to appropriate component.



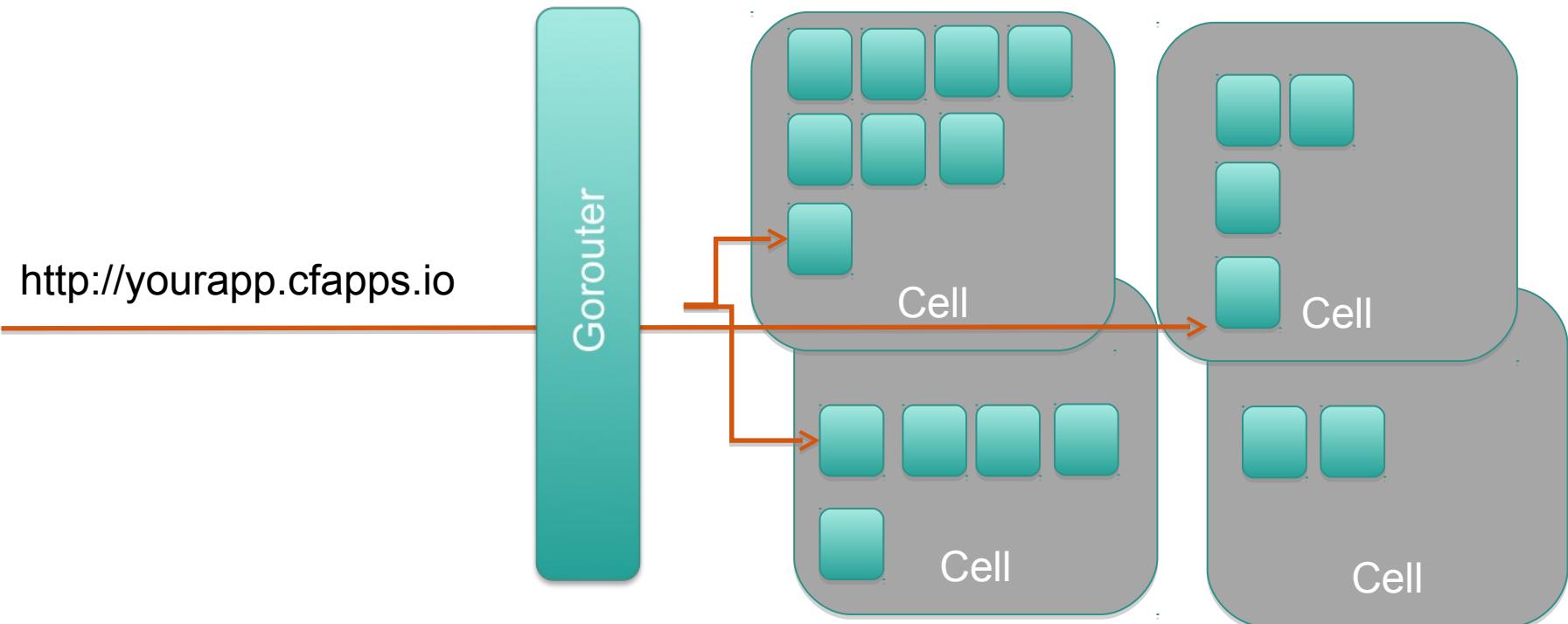
# Router

- The router routes requests to all app instances.

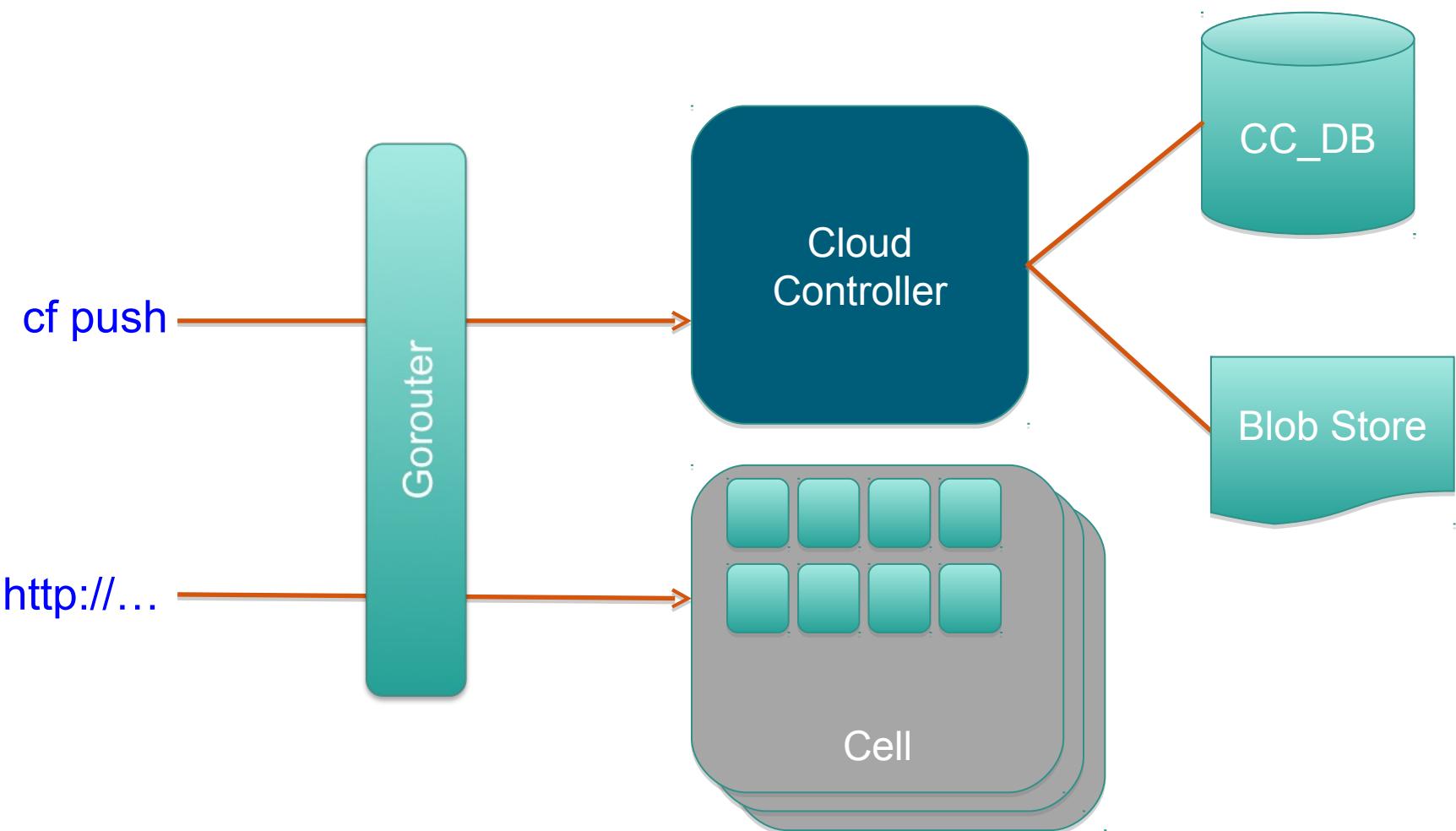


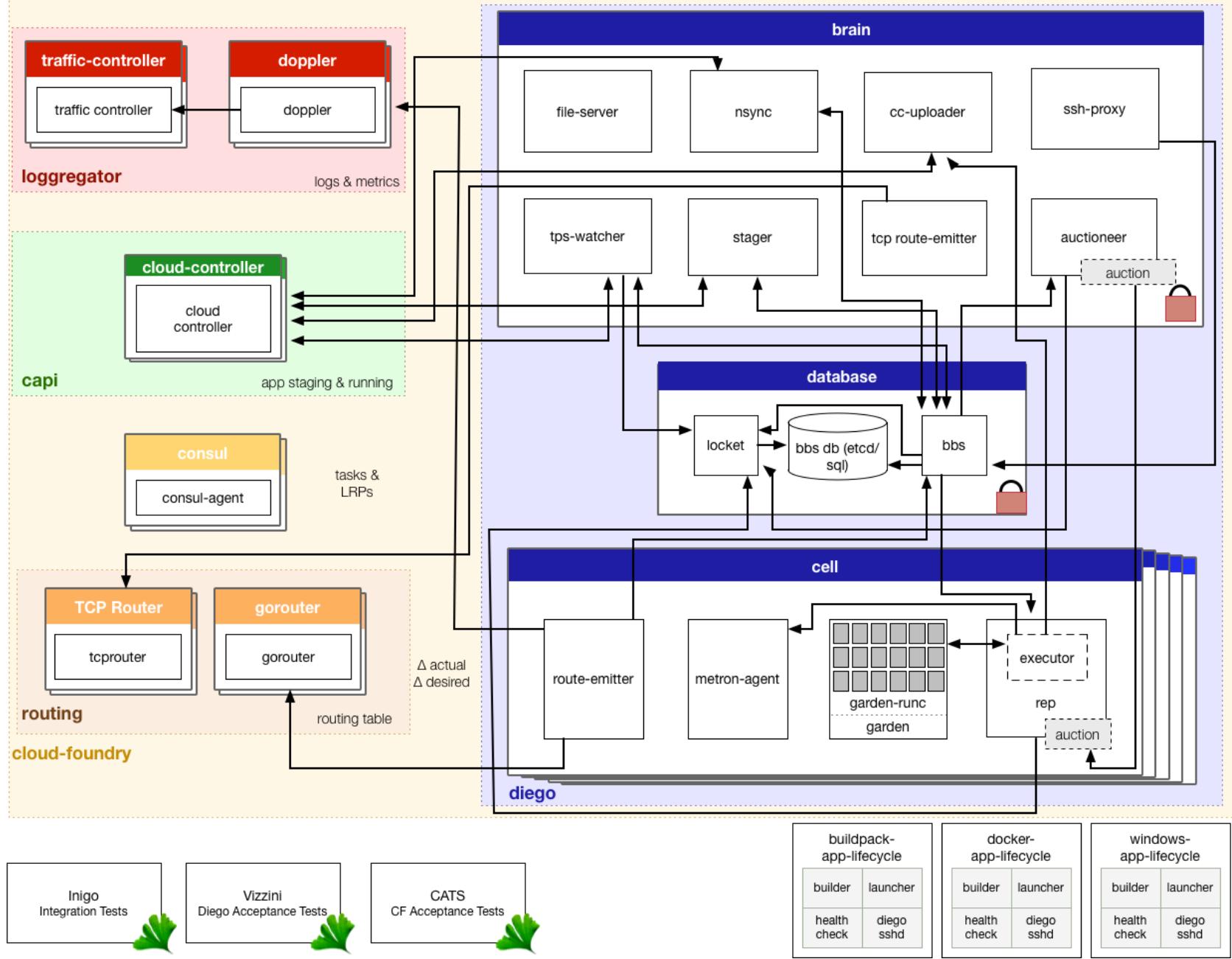
# Router

- The router round-robs between application instances
  - Typically on different Cells



# Elastic Runtime Simplified Architecture





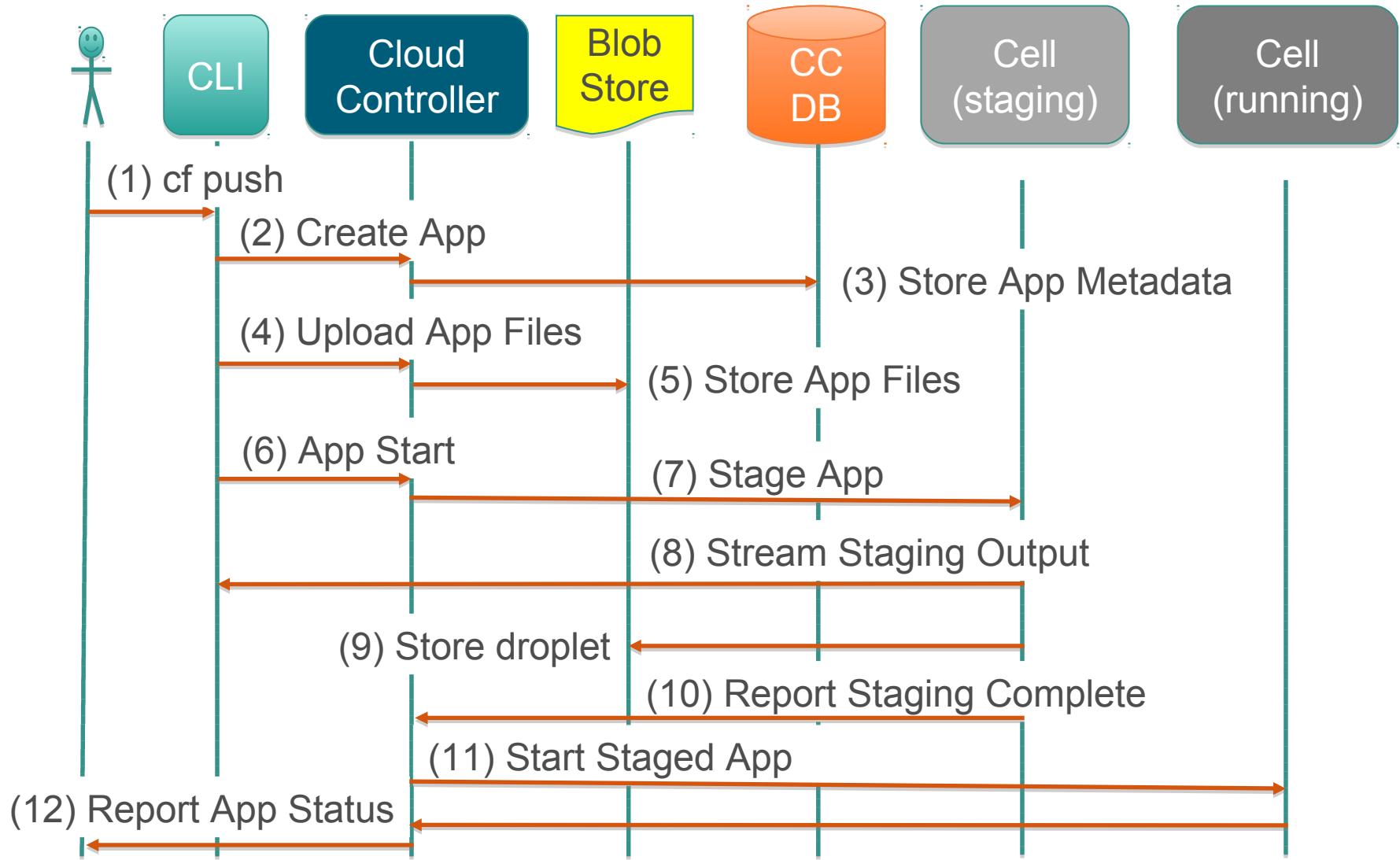
# Key Flows

*How things happen*

<https://docs.pivotal.io/pivotalcf/concepts/diego/diego-architecture.html>

<https://docs.pivotal.io/pivotalcf/concepts/how-applications-are-staged.html>

# Review Pushing An App



# Topics Covered

- The challenges of developing Cloud Native Apps
- The Internal Architecture of Cloud Foundry's Elastic Runtime
- How an application is Staged and Deployed

# Logging, Scaling and High Availability

Production features

PCF Internal Architecture

# Learning Objectives

- After completing this lesson, you should be able to:
  - Describe Cloud Foundry's logging architecture
  - Interpret logging output
  - Scale application instances



# Agenda

- **Logging Architecture**
- Application Logging
- Scaling

# Loggregator

*Logging subsystem*

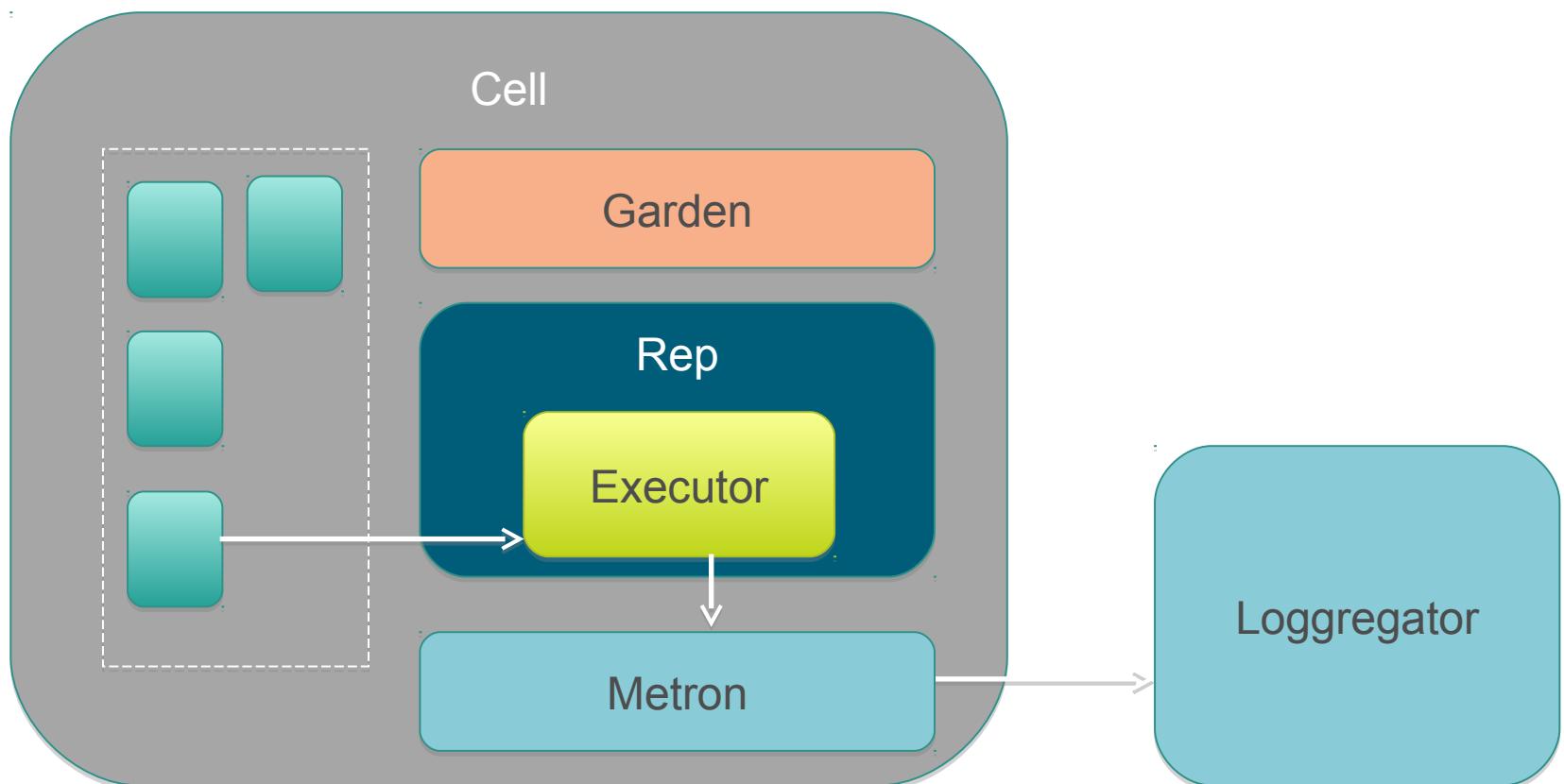
<https://docs.pivotal.io/pivotalcf/loggregator/architecture.html>

# Loggregator

- Streams logs to terminal (**stdout**, **stderr**), not files
- Gathers logs from apps and CF system components
- Stores limited amount of logs in buffer
- Drains logs to third-party log management service

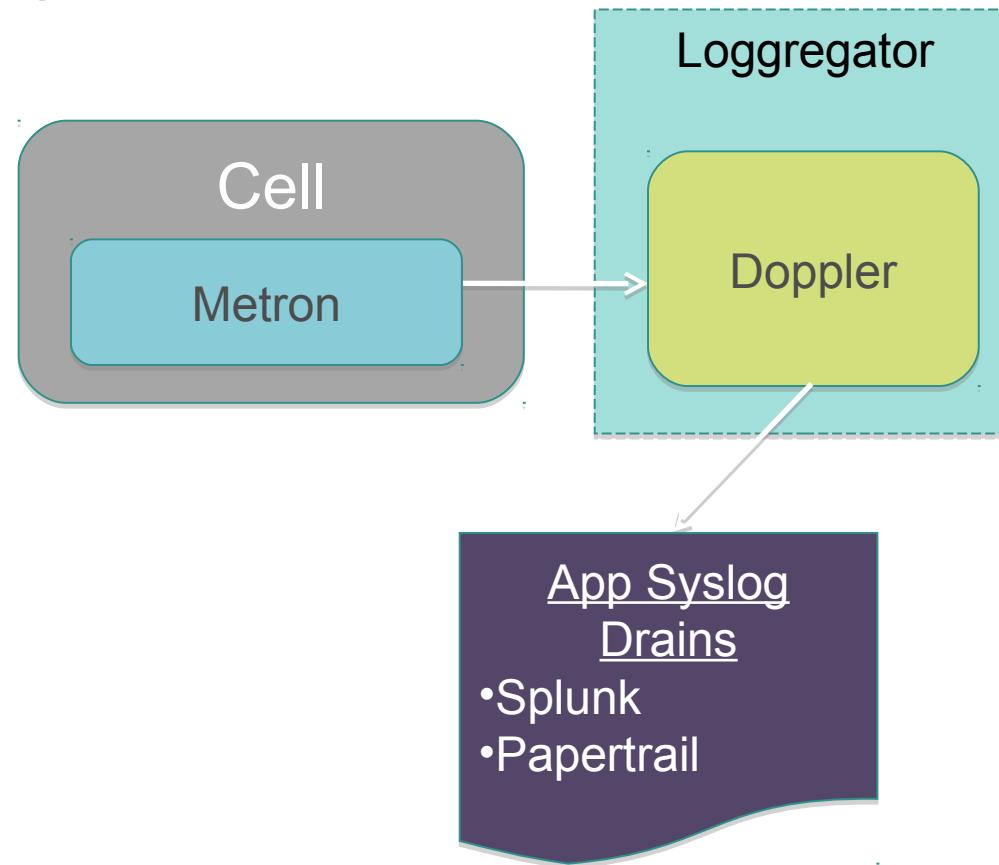
# Loggregator: Metron

- Forwards logs to the Loggregator subsystem



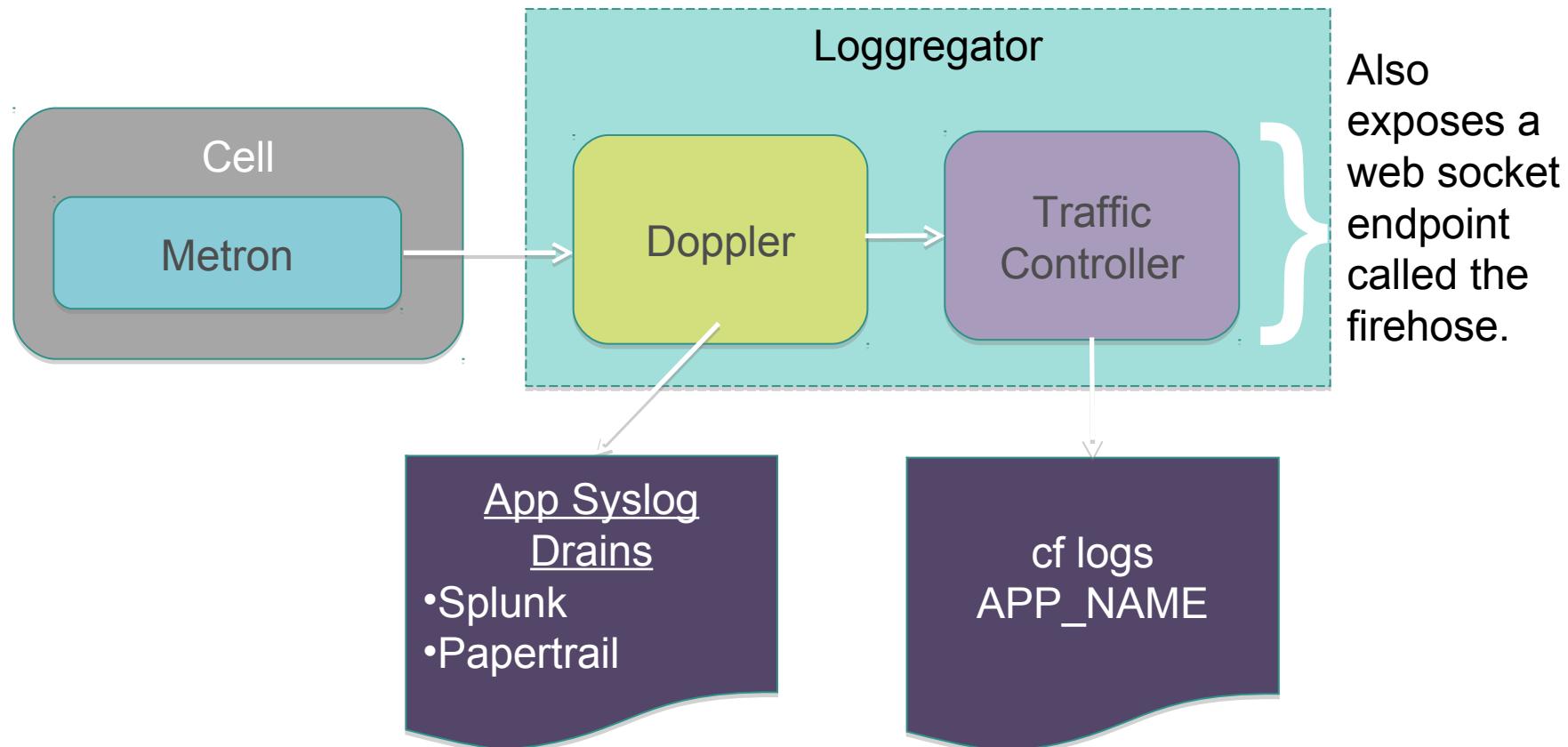
# Loggregator: Doppler

- Gathers logs from Metron



# Loggregator: Traffic Controller

- Handles client requests for logs

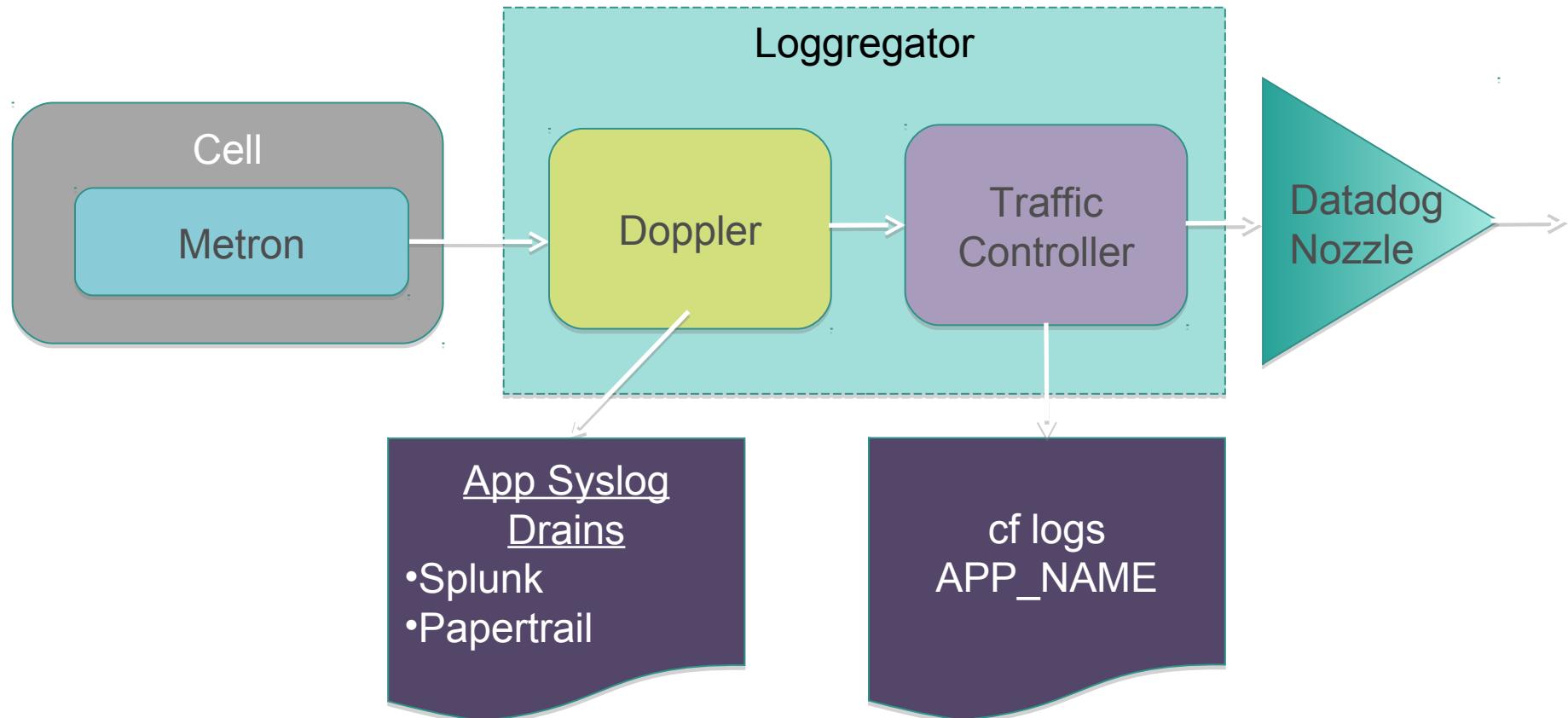


# Loggregator: Firehose

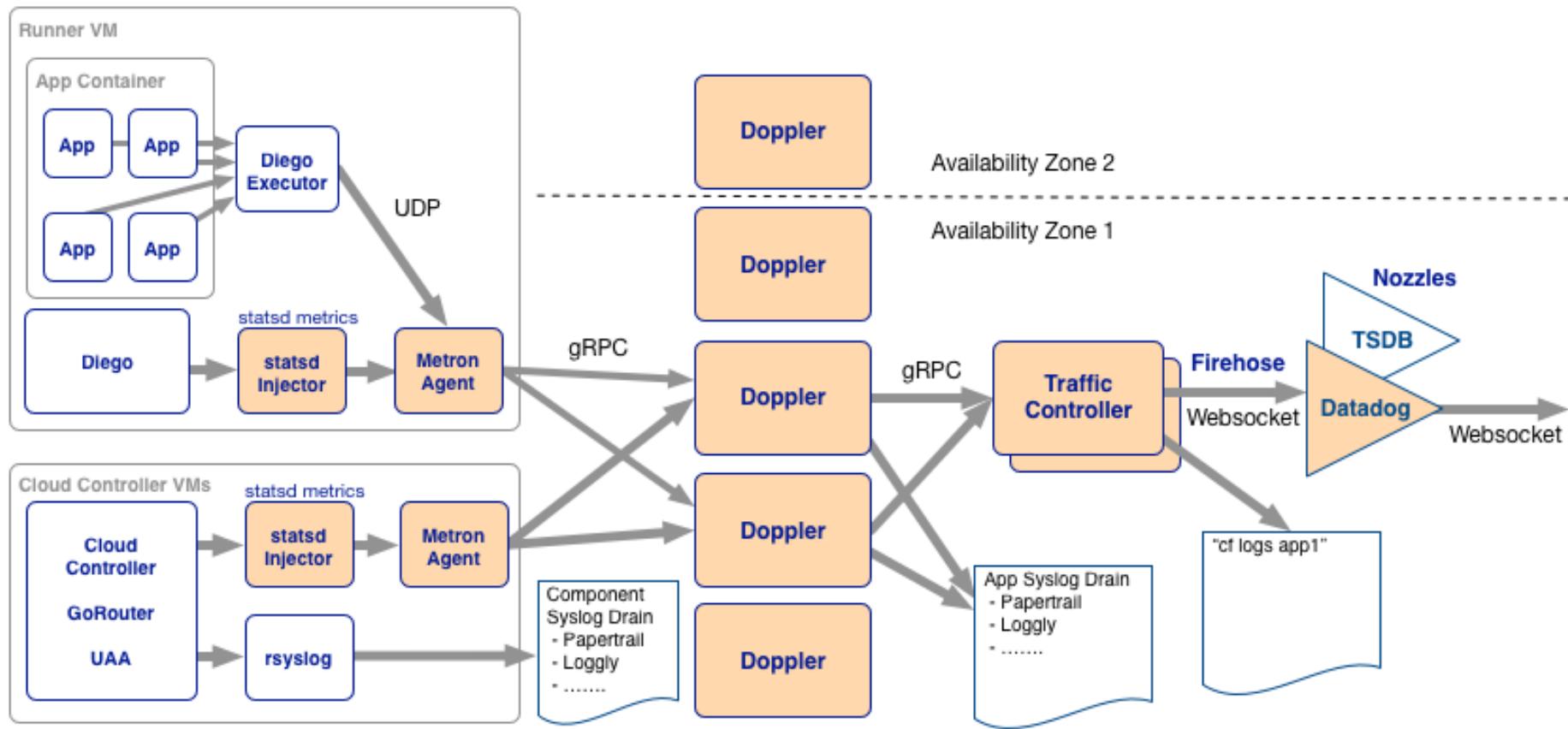
- A websocket endpoint that exposes app logs, container metrics and ER component metrics
- *Does not include ER component logs*

# Loggregator: Nozzles

- Consume the firehose output



# Full System Diagram

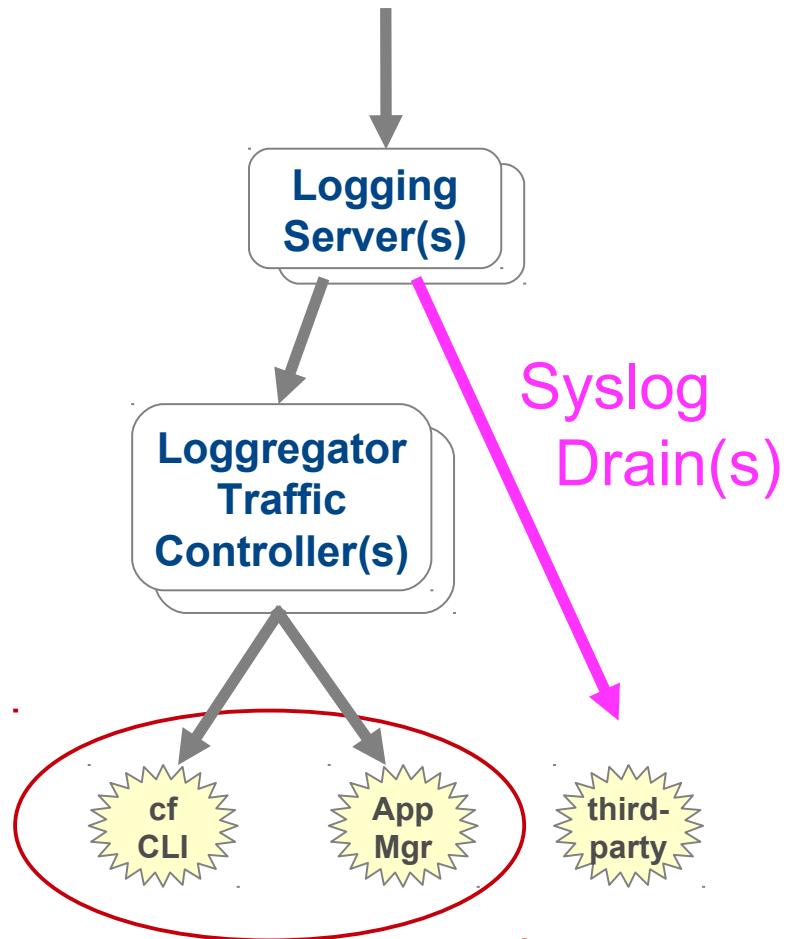


# Agenda

- Logging Architecture
- **Application Logging**
- Scaling

# Logging - Sinks

- CLI – Tail
  - Quick way to obtain output
  - `cf logs <appname>`
  - Use `<ctrl>-c` to exit
- Recent activity
  - `cf logs <appname> --recent`
- Third-party log managers
  - Splunk, logstash, Papertrail, etc.



# A Tour of Log Output

2017-04-11T14:58:41.66	[API]	OUT Updated app with guid 018f9a20-...
2017-04-11T14:58:53.32	[APP/PROC/WEB/0]	OUT 18:58:53,323 INFO WebApplication...
2017-04-11T14:58:53.32	[APP/PROC/WEB/0]	OUT 18:58:53,324 INFO WebApplication...
2017-04-11T14:58:53.52	[APP/PROC/WEB/0]	OUT 18:58:53,527 INFO XmlBeanReader:316 - Loading definitions from class path ...
2017-04-11T14:58:54.47	[APP/PROC/WEB/0]	OUT [...]
2017-04-11T14:58:56.24	[APP/PROC/WEB/0]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.25	[APP/PROC/WEB/0]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.38	[APP/PROC/WEB/0]	ERR INFO: Initializing Spring for Cloud Foundry
2017-04-11T14:58:57.44	[APP/PROC/WEB/0]	ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start
2017-04-11T14:58:57.44	[APP/PROC/WEB/0]	ERR INFO: Starting ProtocolHandler ["http-bio-62773"]
2017-04-11T14:58:57.45	[APP/PROC/WEB/0]	ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start
2017-04-11T14:58:57.45	[APP/PROC/WEB/0]	ERR INFO: Server startup in 8713 ms
2017-04-11T15:01:09.44	[RTR]	OUT spring-music-567.cfapps.io - [19:01:09 +0000] "GET /info HTTP/1.1" 200 48
2017-04-11T15:01:09.46	[RTR]	OUT spring-music-567.cfapps.io - [19:01:09 +0000] "GET /albums HTTP/1.1" 200 4669
2017-04-11T15:03:35.16	[CELL]	OUT Starting app instance (index 1) with guid 018.70d-e84f05375859
2017-04-11T15:03:40.43	[APP/PROC/WEB/1]	ERR INFO: Starting Servlet Engine: Apache Tomcat/7.0.52
2017-04-11T14:58:54.47	[APP/PROC/WEB/1]	OUT [...]
2017-04-11T14:58:56.26	[APP/PROC/WEB/1]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.26	[APP/PROC/WEB/1]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.38	[APP/PROC/WEB/1]	ERR INFO: Initializing Spring Environment for Cloud Foundry
2017-04-11T14:58:57.44	[APP/PROC/WEB/1]	ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start
2017-04-11T14:58:57.44	[APP/PROC/WEB/1]	ERR INFO: Starting ProtocolHandler ["http-bio-62773"]
2017-04-11T14:58:57.45	[APP/PROC/WEB/1]	ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start
2017-04-11T14:58:57.45	[APP/PROC/WEB/1]	ERR INFO: Server startup in 8713 ms

## Timestamp

- added by Loggregator

# A Tour of Log Output

Log Line	Log Type / Origin Code	Notes
2017-04-11T14:58:41.66 [API]	OUT Updated app with guid 018f9a20-f0c9.5375859 {"state"=>"STARTED"}	
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT 18:58:53,323	
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT 18:58:53,324	
2017-04-11T14:58:53.52 [APP/PROC/WEB/0]	OUT 18:58:53,527	
2017-04-11T14:58:54.47 [APP/PROC/WEB/0]	OUT [...]	
2017-04-11T14:58:56.24 [APP/PROC/WEB/0]	OUT Hibernate: in	
2017-04-11T14:58:56.25 [APP/PROC/WEB/0]	OUT Hibernate: in	
2017-04-11T14:58:56.38 [APP/PROC/WEB/0]	ERR INFO: Initiali	
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR Apr 11, 2014	
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR INFO: Startin	
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR Apr 11, 2014	
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR INFO: Server	
2017-04-11T15:01:09.44 [RTR]	OUT spring-music	
2017-04-11T15:01:09.46 [RTR]	OUT spring-music	
2017-04-11T15:03:35.16 [CELL]	OUT Starting app	
2017-04-11T15:03:40.43 [APP/PROC/WEB/1]	ERR INFO: Startin	
2017-04-11T14:58:54.47 [APP/PROC/WEB/1]	OUT [...]	
2017-04-11T14:58:56.26 [APP/PROC/WEB/1]	OUT Hibernate: in	
2017-04-11T14:58:56.26 [APP/PROC/WEB/1]	OUT Hibernate: in	
2017-04-11T14:58:56.38 [APP/PROC/WEB/1]	ERR INFO: Initiali	
2017-04-11T14:58:57.44 [APP/PROC/WEB/1]	ERR Apr 11, 2014	
2017-04-11T14:58:57.44 [APP/PROC/WEB/1]	ERR INFO: Startin	
2017-04-11T14:58:57.45 [APP/PROC/WEB/1]	ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start	
2017-04-11T14:58:57.45 [APP/PROC/WEB/1]	ERR INFO: Server startup in 8713 ms	

# A Tour of Log Output

<p>2017-04-11T14:58:41.66 [API]</p> <p>2017-04-11T14:58:53.32 [APP/PROC/WEB/0]</p> <p>2017-04-11T14:58:53.32 [APP/PROC/WEB/0]</p> <p>2017-04-11T14:58:53.52 [APP/PROC/WEB/0]</p> <p>2017-04-11T14:58:54.47 [APP/PROC/WEB/0]</p> <p>2017-04-11T14:58:56.24 [APP/PROC/WEB/0]</p> <p>2017-04-11T14:58:56.25 [APP/PROC/WEB/0]</p> <p>2017-04-11T14:58:56.38 [APP/PROC/WEB/0]</p> <p>2017-04-11T14:58:57.44 [APP/PROC/WEB/0]</p> <p>2017-04-11T14:58:57.44 [APP/PROC/WEB/0]</p> <p>2017-04-11T14:58:57.45 [APP/PROC/WEB/0]</p> <p>2017-04-11T14:58:57.45 [APP/PROC/WEB/0]</p> <p>2017-04-11T15:01:09.44 [RTR]</p> <p>2017-04-11T15:01:09.46 [RTR]</p> <p>2017-04-11T15:03:35.16 [CELL]</p> <p>2017-04-11T15:03:40.43 [APP/PROC/WEB/1]</p> <p>2017-04-11T14:58:54.47 [APP/PROC/WEB/1]</p> <p>2017-04-11T14:58:56.26 [APP/PROC/WEB/1]</p> <p>2017-04-11T14:58:56.26 [APP/PROC/WEB/1]</p> <p>2017-04-11T14:58:56.38 [APP/PROC/WEB/1]</p> <p>2017-04-11T14:58:57.44 [APP/PROC/WEB/1]</p> <p>2017-04-11T14:58:57.44 [APP/PROC/WEB/1]</p> <p>2017-04-11T14:58:57.45 [APP/PROC/WEB/1]</p> <p>2017-04-11T14:58:57.45 [APP/PROC/WEB/1]</p>	<p>OUT Updated app with guid 018f9a20-f0c9.5375859 {"state"=&gt;"STARTED"})</p> <p>OUT 18:58:53,323 INFO WebApplicationContext:244 - Found 1 ...</p> <p>OUT 18:58:53,324 INFO WebApplicationContext:229 - Successfully resolved ...</p> <p>OUT 18:58:53,527 INFO XmlBeanReader:316 - Loading definitions from class path ...</p> <p>OUT ...]</p> <p>OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)</p> <p>OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)</p> <p>ERR INFO: Initializing Spring for Cloud Foundry</p> <p>ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start</p> <p>ERR INFO: Starting ProtocolHandler ["http-bio-62773"]</p> <p>ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start</p> <p>ERR INFO: Server startup in 8713 ms</p> <p>OUT spring-music-567.cfapps.io</p> <p>OUT spring-music-567.cfapps.io</p> <p>OUT Starting app instance (index 0)</p> <p>ERR INFO: Starting Servlet Engine</p> <p>OUT ...]</p> <p>OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)</p> <p>OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)</p> <p>ERR INFO: Initializing Spring Environment for Cloud Foundry</p> <p>ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start</p> <p>ERR INFO: Starting ProtocolHandler ["http-bio-62773"]</p> <p>ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start</p> <p>ERR INFO: Server startup in 8713 ms</p>
--	--

## Channel

- whether this output came from SYSOUT or SYSERR

# A Tour of Log Output

2017-04-11T14:58:41.66 [API]	OUT	Updated app with guid 018f9a20-f0c9.5375859 {"state"=>"STARTED"}
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT	18:58:53,323 INFO WebApplicationContext:244 - Found 1 ...
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT	18:58:53,324 INFO WebApplicationContext:229 - Successfully resolved ...
2017-04-11T14:58:53.52 [APP/PROC/WEB/0]	OUT	18:58:53,527 INFO XmlBeanReader:316 - Loading definitions from class path ...
2017-04-11T14:58:54.47 [APP/PROC/WEB/0]	OUT	[...]
2017-04-11T14:58:56.24 [APP/PROC/WEB/0]	OUT	Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.25 [APP/PROC/WEB/0]	OUT	Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.38 [APP/PROC/WEB/0]	ERR	INFO: Initializing Spring for Cloud Foundry
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR	Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR	INFO: Starting ProtocolHandler ["http-bio-62773"]
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR	Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR	INFO: Server startup in 8713 ms
2017-04-11T15:01:09.44 [RTR]	OUT	spring-music-567.cfapps.io - [19:01:09 +0000] "GET /info HTTP/1.1" 200 48
2017-04-11T15:01:09.46 [RTR]	OUT	spring-music-567.cfapps.io - [19:01:09 +0000] "GET /albums HTTP/1.1" 200 4669
2017-04-11T15:03:35.16 [CELL]	OUT	Starting app instance (index 1) with guid 018.70d-e84f05375859
2017-04-11T15:03:40.43 [APP/PROC/WEB/1]	ERR	INFO: Starting Servlet Engine: Apache Tomcat/7.0.52
2017-04-11T14:58:54.47 [APP/PROC/WEB/1]	OUT	[...]
2017-04-11T14:58:56.26 [APP/PROC/WEB/1]	OUT	Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?) sert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)

## Message

- From the application / component

2017-04-11T14:58:57.45 [APP/PROC/WEB/1] ERR 6:58:57 PM org.apache.coyote.AbstractProtocol start  
6:58:57 PM org.apache.catalina.startup.Catalina start  
6:58:57 PM org.apache.catalina.startup.Catalina start

2017-04-11T14:58:57.45 [APP/PROC/WEB/1] ERR INFO: Server startup in 8713 ms

# A Tour of Log Output

2017-04-11T14:58:41.66 [API]	OUT Updated app with guid 018f9a20-f0c9.5375859 {"state"=>"STARTED"}
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT 18:58:53,323 INFO WebApplicationContext:244 - Found 1 ...
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT 18:58:53,324 INFO WebApplicationContext:229 - Successfully resolved ...
2017-04-11T14:58:53.52 [APP/PROC/WEB/0]	OUT 18:58:53,527 INFO XmlBeanReader:316 - Loading definitions from class path ...
2017-04-11T14:58:54.47 [APP/PROC/WEB/0]	OUT [...]
2017-04-11T14:58:56.24 [APP/PROC/WEB/0]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.25 [APP/PROC/WEB/0]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.38 [APP/PROC/WEB/0]	ERR INFO: Initializing Spring for Cloud Foundry
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR INFO: Starting ProtocolHandler ["http-bio-62773"]
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR INFO: Server startup in 8713 ms
2017-04-11T15:01:09.44 [RTR]	OUT spring-music-567.cfapps.io - [19:01:09 +0000] "GET /info HTTP/1.1" 200 48
2017-04-11T15:01:09.46 [RTR]	OUT spring-music-567.cfapps.io - [19:01:09 +0000] "GET /albums HTTP/1.1" 200 1660
2017-04-11T15:03:35.16 [CELL]	OUT Starting app instance (index 1) with guid 018f9a20-f0c9.5375859
2017-04-11T15:03:40.43 [APP/PROC/WEB/1]	ERR INFO: Starting Servlet Engine: Apache Tomcat/8.0.29
2017-04-11T14:58:54.47 [APP/PROC/WEB/1]	OUT [...]
2017-04-11T14:58:56.26 [APP/PROC/WEB/1]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.26 [APP/PROC/WEB/1]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.38 [APP/PROC/WEB/1]	ERR INFO: Initializing Spring Environment for Cloud Foundry
2017-04-11T14:58:57.44 [APP/PROC/WEB/1]	ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start
2017-04-11T14:58:57.44 [APP/PROC/WEB/1]	ERR INFO: Starting ProtocolHandler ["http-bio-62773"]
2017-04-11T14:58:57.45 [APP/PROC/WEB/1]	ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start
2017-04-11T14:58:57.45 [APP/PROC/WEB/1]	ERR INFO: Server startup in 8713 ms

Instance 0 startup activity

# A Tour of Log Output

2017-04-11T14:58:41.66 [API]	OUT Updated app with guid 018f9a20-f0c9.5375859 {"state"=>"STARTED"})
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT 18:58:53,323 INFO WebApplicationContext:244 - Found 1 ...
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT 18:58:53,324 INFO WebApplicationContext:229 - Successfully resolved ...
2017-04-11T14:58:53.52 [APP/PROC/WEB/0]	OUT 18:58:53,527 INFO XmlBeanReader:316 - Loading definitions from class path ...
2017-04-11T14:58:54.47 [APP/PROC/WEB/0]	OUT [...]
2017-04-11T14:58:56.24 [APP/PROC/WEB/0]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.25 [APP/PROC/WEB/0]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.38 [APP/PROC/WEB/0]	ERR INFO: Initializing Spring for Cloud Foundry
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR INFO: Starting ProtocolHandler ["http-bio-62773"]
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR INFO: Server startup in 8713 ms
2017-04-11T15:01:09.44 [RTR]	OUT spring-music-567.cfapps.io - [19:01:09 +0000] "GET /info HTTP/1.1" 200 48
2017-04-11T15:01:09.46 [RTR]	OUT spring-music-567.cfapps.io - [19:01:09 +0000] "GET /albums HTTP/1.1" 200 4669
2017-04-11T15:03:35.16 [CELL]	OUT Starting app instance 0 (index 1) with guid 018.70d-e84f05375859
2017-04-11T15:03:40.43 [APP/PROC/WEB/1]	ERR INFO: Starting Servlet Engine: Apache Tomcat/7.0.52
2017-04-11T14:58:54.47 [APP/PROC/WEB/1]	OUT [...]
2017-04-11T14:58:56.26 [APP/PROC/WEB/1]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.26 [APP/PROC/WEB/1]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.38 [APP/PROC/WEB/1]	ERR INFO: Initializing Spring Environment for Cloud Foundry
2017-04-11T14:58:57.44 [APP/PROC/WEB/1]	ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start
2017-04-11T14:58:57.44 [APP/PROC/WEB/1]	ERR INFO: Starting ProtocolHandler ["http-bio-62774"]
2017-04-11T14:58:57.45 [APP/PROC/WEB/1]	ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start
2017-04-11T14:58:57.45 [APP/PROC/WEB/1]	ERR INFO: Server startup in 8713 ms

User requests for a web page on instance 0

# A Tour of Log Output

2017-04-11T14:58:41.66 [API]	OUT Updated app with guid 018f9a20-f0c9.5375859 {"state"=>"STARTED"}
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT 18:58:53,323 INFO WebApplicationContext:244 - Found 1 ...
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT 18:58:53,324 INFO WebApplicationContext:229 - Successfully resolved ...
2017-04-11T14:58:53.52 [APP/PROC/WEB/0]	OUT 18:58:53,527 INFO XmlBeanReader:316 - Loading definitions from class path ...
2017-04-11T14:58:54.47 [APP/PROC/WEB/0]	OUT [...]
2017-04-11T14:58:56.24 [APP/PROC/WEB/0]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.25 [APP/PROC/WEB/0]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.38 [APP/PROC/WEB/0]	ERR INFO: Initializing Spring for Cloud Foundry
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR INFO: Starting ProtocolHandler ["http-bio-62773"]
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR INFO: Server startup in 8713 ms
2017-04-11T15:01:09.44 [RTR]	OUT spring-music-567.cfapps.io - [19:01:09 +0000] "GET /info HTTP/1.1" 200 48
2017-04-11T15:01:09.46 [RTR]	OUT spring-music-567.cfapps.io - [19:01:09 +0000] "GET /albums HTTP/1.1" 200 4669
2017-04-11T15:03:35.16 [CELL]	OUT Starting app instance (index 1) with guid 018.70d-e84f05375859
2017-04-11T15:03:40.43 [APP/PROC/WEB/1]	ERR INFO: Starting Server Engine: Apache Tomcat/7.0.52
2017-04-11T14:58:54.47 [APP/PROC/WEB/1]	OUT [...]
2017-04-11T14:58:56.26 [APP/PROC/WEB/1]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.26 [APP/PROC/WEB/1]	OUT Hibernate: insert into Alb
2017-04-11T14:58:56.38 [APP/PROC/WEB/1]	ERR INFO: Initializing Spring F
2017-04-11T14:58:57.44 [APP/PROC/WEB/1]	ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start
2017-04-11T14:58:57.44 [APP/PROC/WEB/1]	ERR INFO: Starting ProtocolHandler ["http-bio-62774"]
2017-04-11T14:58:57.45 [APP/PROC/WEB/1]	ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start
2017-04-11T14:58:57.45 [APP/PROC/WEB/1]	ERR INFO: Server startup in 8713 ms

Request to scale application

- cf scale <appname> -i 2

# A Tour of Log Output

2017-04-11T14:58:41.66 [API]	OUT Updated app with guid 018f9a20-f0c9.5375859 {"state"=>"STARTED"}
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT 18:58:53,323 INFO WebApplicationContext:244 - Found 1 ...
2017-04-11T14:58:53.32 [APP/PROC/WEB/0]	OUT 18:58:53,324 INFO WebApplicationContext:229 - Successfully resolved ...
2017-04-11T14:58:53.52 [APP/PROC/WEB/0]	OUT 18:58:53,527 INFO XmlBeanReader:316 - Loading definitions from class path ...
2017-04-11T14:58:54.47 [APP/PROC/WEB/0]	OUT [...]
2017-04-11T14:58:56.24 [APP/PROC/WEB/0]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.25 [APP/PROC/WEB/0]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.38 [APP/PROC/WEB/0]	ERR INFO: Initializing Spring for Cloud Foundry
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start
2017-04-11T14:58:57.44 [APP/PROC/WEB/0]	ERR INFO: Starting ProtocolHandler ["http-bio-62773"]
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start
2017-04-11T14:58:57.45 [APP/PROC/WEB/0]	ERR INFO: Server startup in 8713 ms
2017-04-11T15:01:09.44 [RTR]	OUT spring-music-567.cfapps.io - [19:01:09 +0000] "GET /info HTTP/1.1" 200 48
2017-04-11T15:01:09.46 [RTR]	OUT spring-music-567.cfapps.io - [19:01:09 +0000] "GET /albums HTTP/1.1" 200 4669
2017-04-11T15:03:35.16 [CELL]	OUT Starting app instance (index 1) with guid 018.70d-e84f05375859
2017-04-11T15:03:40.43 [APP/PROC/WEB/1]	ERR INFO: Starting Servlet Engine: Apache Tomcat/7.0.52
2017-04-11T14:58:54.47 [APP/PROC/WEB/1]	OUT [...]
2017-04-11T14:58:56.26 [APP/PROC/WEB/1]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.26 [APP/PROC/WEB/1]	OUT Hibernate: insert into Album (albumId, artist, trackCount, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2017-04-11T14:58:56.38 [APP/PROC/WEB/1]	ERR INFO: Initializing Spring Environment for Cloud Foundry
2017-04-11T14:58:57.44 [APP/PROC/WEB/1]	ERR Apr 11, 2014 6:58:57 PM org.apache.coyote.AbstractProtocol start
2017-04-11T14:58:57.44 [APP/PROC/WEB/1]	ERR INFO: Starting ProtocolHandler ["http-bio-62773"]
2017-04-11T14:58:57.45 [APP/PROC/WEB/1]	ERR Apr 11, 2014 6:58:57 PM org.apache.catalina.startup.Catalina start
2017-04-11T14:58:57.45 [APP/PROC/WEB/1]	ERR INFO: Server startup in 8713 ms

Instance #1 startup activity

# Agenda

- Logging Architecture
- Application Logging
- **Scaling**

# Configuring a Deployed Application

- Change the number of instances
  - `cf scale <app> -i <new-value>`
  - Two instances: `cf scale spring-music -i 2`
  - New instances added, or some existing instances stopped

# Configuring a Deployed Application

- Change the memory allocation
  - `cf scale <app> -m <new-value>`
  - 1024M: `cf scale spring-music -m 1024M`
  - Requires a restart to take effect

# Configuring a Deployed Application

- Change the disk space allocation
  - `cf scale <app> -k <new-value>`
  - 512M: `cf scale spring-music -k 512M`
  - Requires a restart to take effect

# Topics Covered

- Cloud Foundry's Logging Architecture
- Application Logging Output
- Scaling Application Instances

# Lab

Logging and scaling an application

# Services

What are they?

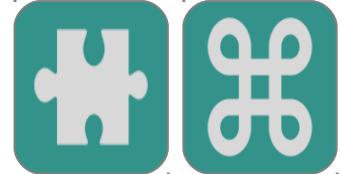
Creating and binding

# Agenda

- **Services**
- Managed (Marketplace) Services
- Provisioning Services
- Binding to a Service
- User Provided Services

# Cloud Foundry Services

## What is a service?



- **Service** is an **external** application **dependency** or **component** such as
  - Database
  - Message Queue
  - Monitoring App
  - Security
  - Hadoop
  - Other dependent applications



# Cloud Foundry Services

## Features and Functionality

- Provide *functionality* to your applications
- *External* to your applications
  - Add-on, provisioned *alongside* an application.
- May be *shared* among many applications
  - Example – Relational DB, Messaging system
- Are *bound* to (associated with) an application
  - Using a “Service Broker”



**Service-Broker** allows Cloud Controller to interface with *any* service, *without* needing to know anything about it.



# Cloud Foundry Services

## Why use a service?

- Applications are the deployment unit
  - Must be self-contained
  - Anything else they need is provided by the PaaS
    - *By a service*
- Services in a PaaS are
  - One of the main possible charging units / elements
    - Instead of hardware resources like an IaaS
  - Make commercial PaaS possible
  - Enable charge-back in your organization

# 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

# Twelve Factor Design Patterns

Codebase

Depend-  
encies

Config

Backing  
Services

Build,  
release, run

Processes

Port binding

Concurrency

Disposability

Dev/prod  
parity

Logs

Admin  
Processes

# Services – Important Cloud Design Pattern

- **Configuration**
  - *Store configuration in the environment*
  - Service information passed via environment
    - **VCAP\_SERVICES**
    - Connection details, credentials, ...
- **Backing Services**
  - *Treat backing services as attached resources*
  - Services separate from the applications that use them

# Services Summary

- Question: Within a Cloud Foundry App, How do I...
  - Connect to a relational database?
  - Connect to a messaging system?
  - Connect to an email system?
  - Utilize NoSQL databases?
  - Read and write files
  - Save and retrieve sessions
  - Access anything *not* coded in my application
- **Answer: Via Services**



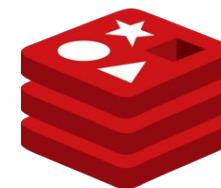
# Roadmap

- Services
- **Managed (Marketplace) Services**
- Provisioning Services
- Binding to a Service
- User Provided Services

# Pivotal CF Services

## What Services are Available

- Whatever your company chooses
  - Services added after Pivotal CF installation by Ops
  - Available as .pivotal files from Pivotal Network
    - See: <https://network.pivotal.io>
- PCF provides many services
  - They may or may not be available to your private cloud



Pivotal™

# Pivotal CF Services

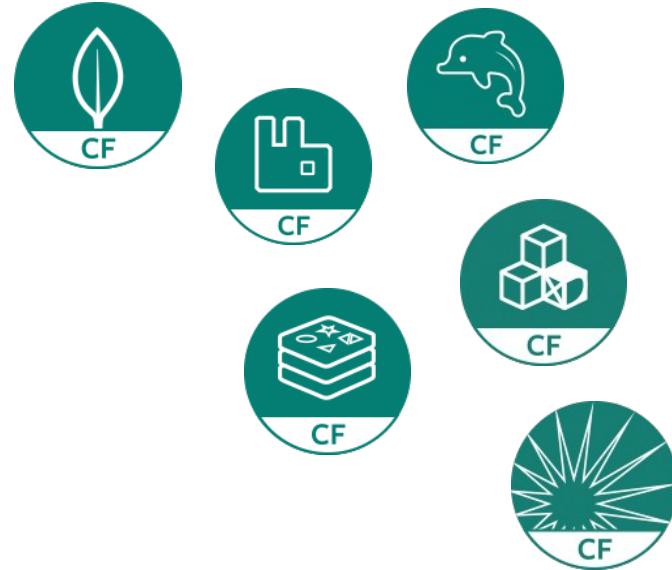
- Typically installed by PCF Operators as “Tiles”
  - Each tile sets up service by deploying its service broker

The screenshot shows the Pivotal CF Ops Manager interface. On the left, a sidebar lists "Available Products" including Ops Manager Director, Pivotal Elastic Runtime, RabbitMQ, AppDynamics Service Broker, Pivotal JMX Bridge, and MySQL for Pivotal Cloud Foundry. At the bottom of this sidebar are buttons for "Import a Product" and download links for PCF compatible products from the Pivotal Network. The main area is titled "Installation Dashboard" and displays six service tiles:

- Ops Manager Director for VMware vSphere (v1.5.2.0)
- Pivotal Elastic Runtime (v1.5.2.0)
- RabbitMQ (v1.5.1)
- AppDynamics Service Broker (v1.0)
- MySQL for Pivotal Cloud Foundry (v1.7.0.4)
- Pivotal JMX Bridge (Ops Metrics) (v1.4.1.0)

On the right side of the dashboard, there is a "Recent Install Logs" section and a large blue "Apply changes" button. A callout bubble with an arrow points to the RabbitMQ tile, containing the text: "Once installed each ‘Tile’ service appears in the Marketplace".

# Pivotal Services



- PCF supplies many service “Tiles”
  - MySQL (MariaDB), Redis,
  - Rabbit/MQ, Cloud Cache
  - Single Sign-On, ClamAV
  - ...
- Only available in Marketplace if *Tile* previously deployed by PCF Operators
  - Developers can only create instances of services already in the marketplace

# Pivotal Web Services (PWS)



- Public Cloud Foundry instance
  - Hosted on AWS
  - Provides extensive marketplace of services via *App Direct*
    - Some free, some pay-per use
    - Examples
      - Postgres DB, MySQL, MongoDB, Redis
      - Rabbit MQ
      - Blazemeter monitoring
      - Many, many more ...
  - Plus some services from Pivotal
    - Autoscaler, MySQL for PCF, Spring Cloud Services ...

# Pivotal Web Services (PWS)

## Marketplace Home Page in App Manager Console

The screenshot shows the Pivotal Web Services (PWS) Marketplace Home Page in the App Manager Console. The left sidebar displays the organization 'krueger-net' and the selected space 'Marketplace'. The main content area is titled 'Marketplace' and contains a search bar. Below the search bar, there is a section titled 'Services' with a dropdown arrow. A list of services is shown, each with an icon and a brief description:

- 3scale API Management: API Management Platform
- App Autoscaler: Scales bound applications in response to load
- BlazeMeter: Performance Testing Platform
- Cedexis Openmix: Openmix Global Cloud and Data Center Load Balancer
- Cedexis Radar: Free Website and Mobile App Performance Reports
- Circuit Breaker: Circuit Breaker Dashboard for Spring Cloud Applications
- ClearDB MySQL Database: Highly available MySQL for your Apps.

Two services, 'App Autoscaler' and 'Circuit Breaker', are highlighted with arrows pointing to a callout box labeled 'PCF Service'.

Pivotal™

# Roadmap

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



# Service Lifecycle

- Each service has well-defined *lifecycle*
  - **Create Service**
    - Service must exist in Marketplace
    - Really “provisioning”
  - **Bind Service**
    - Make available to an application
  - **Unbind Service**
  - **Delete Service**

# Finding Available Services

## Command Line Interface

- Check marketplace for available services
  - Essentially a service catalog
- Example – PWS marketplace
  - Your PCF will be different

```
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
cleardb      spark, boost, amp, shock
cloudamqp    lemur, tiger, bunny, rabbit, panda
cloudforge   free, standard, pro
elephantsql  turtle, panda, hippo, elephant
ironmq       pro_platinum, pro_gold, large, medium, small, pro_silver
ironworker   large, pro_gold, pro_platinum, pro_silver, small, medium
...
...
```

# 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, must be in correct space
  - [cf target -s \[space-name\]](#)

# 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
- 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
```

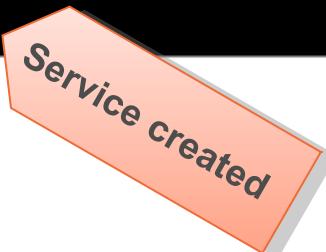
# Finding Existing Service Instances

## 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
```



# Roadmap

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

# Provisioning Service Instances GUI

The screenshot shows the Pivotal Web Services (PWS) interface for the 'development' space. On the left, a sidebar menu includes links for Org (pivotaledu), Spaces (development, production, staging), Marketplace, Docs, Support, Tools, Blog, and Status. A large orange arrow points from the 'Marketplace' link towards the 'Services' section on the right. The main content area displays the 'development' space with the following details:

**APPLICATIONS**

STATUS	APP	INSTANCES	MEMORY
STOPPED	booking-app-123 booking-app-123.cfapp....	1	1GB 54

**SERVICES**

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

**Services**

Pivotal

# Finding Available Services

## Service Selection

The screenshot shows the Pivotal Web Services Marketplace interface. On the left, a sidebar menu includes 'ORG' (pivotaledu), 'SPACES' (development, production, staging), and 'Marketplace' (selected). Below these are links for 'Docs', 'Support', 'Tools', 'Blog', and 'Status'. The main content area is titled 'Services Marketplace' and features a message: 'Get started with our free marketplace services. Upgrade to gain access to premium service plans.' It lists several services with their icons and descriptions:

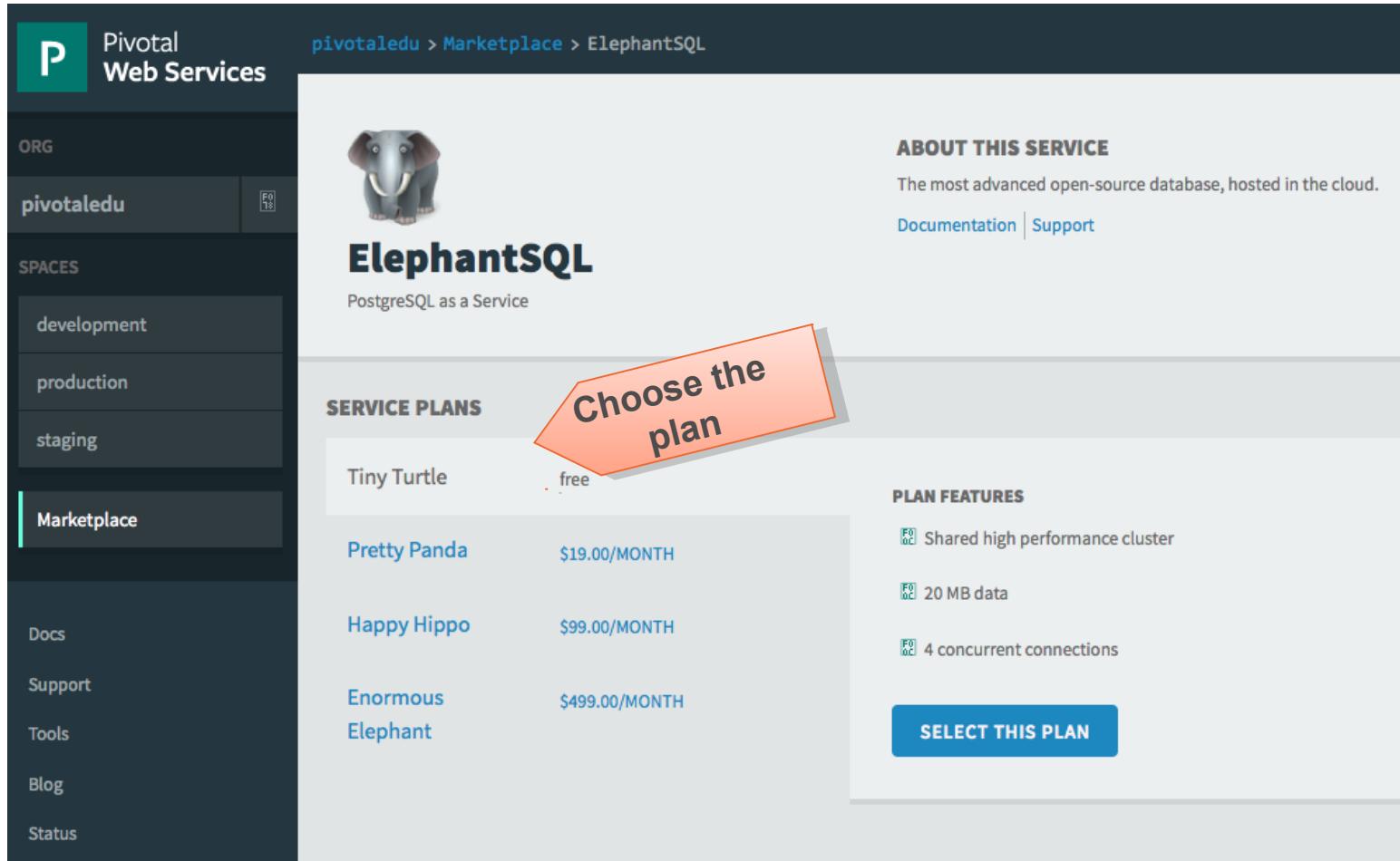
- BlazeMeter**: The JMeter Load Testing Cloud
- CloudAMQP**: Managed HA RabbitMQ servers in the cloud
- ElephantSQL**: PostgreSQL as a Service
- IronWorker**: Scalable Background and Async Processing
- ClearDB MySQL Database**: Highly available MySQL for your Apps.
- CloudForge**: Development Tools In The Cloud
- IronMQ**: Powerful Durable Message Queueing Service

A large orange callout box with a diagonal shadow points from the bottom right towards the 'VIEW PLAN OPTIONS' button on the ElephantSQL card, containing the text: "Choose to check available plans".

Pivotal™

# Provisioning a new Service Instance

## Pick a Plan



The screenshot shows the Pivotal Web Services interface. On the left, a sidebar lists 'ORG' (pivotaledu), 'SPACES' (development, production, staging), 'Marketplace' (selected), 'Docs', 'Support', 'Tools', 'Blog', and 'Status'. The main content area shows the 'pivotaledu > Marketplace > ElephantSQL' page. It features a large elephant icon, the title 'ElephantSQL PostgreSQL as a Service', and a 'ABOUT THIS SERVICE' section stating 'The most advanced open-source database, hosted in the cloud.' with links to 'Documentation' and 'Support'. A prominent orange callout box with the text 'Choose the plan' points to the 'SERVICE PLANS' section. This section lists four plans: 'Tiny Turtle' (free), 'Pretty Panda' (\$19.00/MONTH), 'Happy Hippo' (\$99.00/MONTH), and 'Enormous Elephant' (\$499.00/MONTH). To the right, a 'PLAN FEATURES' section lists 'Shared high performance cluster', '20 MB data', and '4 concurrent connections', each with an icon. A blue 'SELECT THIS PLAN' button is at the bottom.

Service Plan	Cost
Tiny Turtle	free
Pretty Panda	\$19.00/MONTH
Happy Hippo	\$99.00/MONTH
Enormous Elephant	\$499.00/MONTH

# Provisioning a new Service Instance

## Provision (Create) Service

The screenshot shows the Pivotal Web Services interface for provisioning a new service instance. On the left, the navigation bar includes 'ORG' (pivotededu), 'SPACES' (development, production, staging), 'Marketplace' (selected), 'Docs', 'Support', 'Tools', 'Blog', and 'Status'. The main content area displays the 'ElephantSQL' service details: 'ABOUT THIS SERVICE' (PostgreSQL as a Service, hosted in the cloud), 'COMPANY' (84codes AB), and a 'SERVICE PLAN' (Tiny Turtle, free). The 'CONFIGURE INSTANCE' section allows specifying the 'Instance Name' (mypyg), 'Add to Space' (development), and 'Bind to App' ([do not bind]). A large orange callout points to the 'Instance Name' field with the text 'Specify instance name'. Another orange callout points to the 'ADD' button with the text 'Create the service'.

# Provisioning a new Service Instance Complete

The screenshot shows the Pivotal Web Services dashboard for the organization 'pivotaledu' and space 'development'. A green banner at the top indicates 'Service instance mypg created.' An orange callout box labeled 'Success' is overlaid on the banner. Below the banner, the space name 'development' is displayed. The 'APPLICATIONS' section lists one application named 'booking-app-123' which is currently 'STOPPED'. The 'SERVICES' section lists two service instances: 'mysql' (bound to 'spark') and 'mypg' (bound to 'turtle'). An orange callout box labeled 'Service available' is overlaid on the 'mypg' row.

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

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

Pivotal™

# Roadmap

- Services
- Managed (Marketplace) Services
- Provisioning Services
- **Binding to a Service**
- User Provided Services

# Using a Service

## Binding using the CLI

- **Binding** associates an application to a service instance.
  - Use `cf bind-service`
  - `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
```

- Once application staged
  - `cf env [app-name]`
    - Look for **VCAP\_SERVICES** in the output

# Using a Service Binding Using App Manager – During *Creation*

The screenshot shows the Pivotal Web Services App Manager interface. On the left, the sidebar lists 'ORG' (pivotaledu), 'SPACES' (development, production, staging), and 'Marketplace'. The 'Marketplace' item is selected and highlighted with a teal bar at the bottom. The main content area shows the 'ElephantSQL' service details: 'ABOUT THIS SERVICE' (PostgreSQL as a Service) and 'COMPANY' (84codes AB). Below this is the 'SERVICE PLAN' section, which shows 'Tiny Turtle' as the plan and 'free' as the cost. A large modal window titled 'CONFIGURE INSTANCE' is open, prompting for 'Instance Name' (mypyg), 'Add to Space' (development), and 'Bind to App' ([do not bind]). At the bottom of the modal are 'CANCEL' and 'ADD' buttons. Two orange callout boxes are overlaid on the modal: one pointing to the 'Instance Name' field with the text 'Specify application name', and another pointing to the 'ADD' button with the text 'Create the service'.

# Accessing a Bound Service

- Bind the service instance to application(s)
  - 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` environment variable
- Changes (host/port/credentials) are *external* to app
  - Rerun application to pick up *new* service information

# 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

ClearDB is MySQL service offered through App Direct

# 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



# Using a Service – Application View

## 2. Library Support

- Avoid manual parsing using a cloud-aware library
  - Cloud foundry aware helper code
    - Parses **VCAP\_SERVICES** for you
  - JVM: use Spring Cloud Connectors
  - Node.js: use *cfruntime* object
  - Ruby: **cf-apps-utils** gem

Derived from  
**VCAP\_SERVICES**

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

Java Example

# Viewing Connection Information

- Connection information also available via App Manager:

**BOUND SERVICES**

**+ Bind a Service**

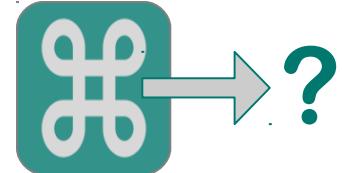
mydb	Manage	Support	Docs
ClearDB MySQL Database Spark DB			
<a href="#">▼ Hide credentials</a>			<a href="#">Unbind</a>
JDBCURL	jdbc:mysql://b4eb8a837a231d:c3b28eec@us-cdbr-east-06.cleardb.net:3306/ad_d691ab771b6397e		
URI	mysql://b4eb8a837a231d:c3b28eec@us-cdbr-east-06.cleardb.net:3306/ad_d691ab771b6397e?reco		
NAME	ad_d691ab771b6397e		
HOSTNAME	us-cdbr-east-06.cleardb.net		
PORT	3306		
USERNAME	b4eb8a837a231d		
PASSWORD	c3b28eec		

[View JSON](#)

# Roadmap

- Services
- Managed (Marketplace) Services
- Provisioning Services
- Binding to a Service
- **User Provided Services**

# Accessing External Services



- Typically these exist already
  - Corporate Databases: Oracle, DB2, SQL Server
  - ERP and CRM systems (Oracle, SAP ...)
  - Messaging: IBM MQ Series, Tibco
  - Existing JEE or .NET applications
  - Mail Server
  - Your Mainframe and other legacy applications
  - Cloud-based services such as Sales, CRM or Payroll

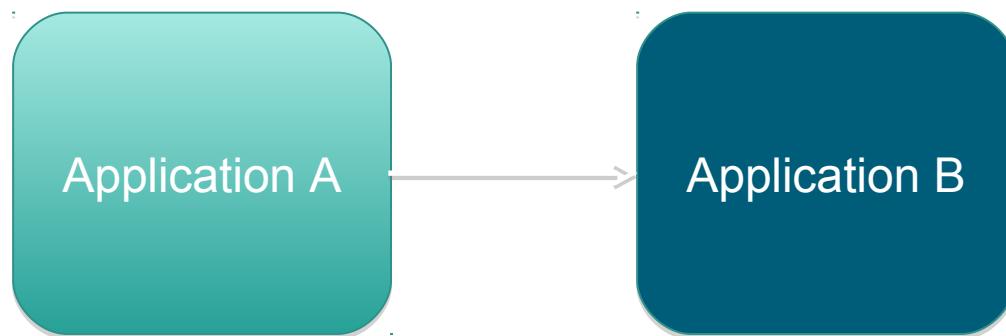
# User Provided Services – 1

- **User-Provided Service Instances**
  - Access services *outside* of Cloud Foundry
  - Look like other service instances once provisioned
  - 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
  - Can inject same service info into multiple applications

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

# User Provided Services – 2

- Also used for application to application binding
  - One application offers “services” to the other



# Defining User Provided Services

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

Using alias: `cf cups`

```
$ 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
```

The diagram illustrates the interaction between the command line and the terminal prompts. An arrow points from the parameter `-p "hostname, port, username, password, name"` in the command line to the first prompt `hostname>`. Another arrow points from the same parameter to the second prompt `port>`. A third arrow points from the parameter to the third prompt `username>`. A fourth arrow points from the parameter to the fifth prompt `password>`. A fifth arrow points from the parameter to the sixth prompt `name>`. This visualizes how the user specifies a list of parameters at the command line, which are then used to generate the individual prompts for each value.

Specify any list of parameters here

Prompts for parameters values

# User Provided Services - Accessing

- Bound service info also available in **VCAP\_SERVICES** environment variable
- In your code
  - Access variable
  - Parse JSON
  - Use to connect
- Or use a library such as *Spring Cloud Connectors*

```
{  
  "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": []
  }
]
```

# Summary – Two Types of Service

- Managed Services (a.k.a. “Marketplace” Services)
  - Available 'out-of-the-box'
  - Select from Marketplace 'catalog'
  - Instances provisioned *for you*
- User Defined Services
  - Services running external to Cloud Foundry
  - Connection information stored and used to connect
  - PaaS does not provision resources, only supplies connection information
  - Provisioned and managed *by you*



# What you have learned

- What is a service?
- Provisioning Services
  - Using the CLI
  - Using the Pivotal CF App Manager Console
- Binding to a Service
- Accessing Service details via **VCAP\_SERVICES**
- User Provided Services

# Lab

## Using services



CLOUD **FOUNDRY**

# Manifests and Environment Variables

A closer look at practical Cloud Foundry usage

Configuring your pushed application

Pivotal

# Overview

- After completing this lesson, you should be able to:
  - Define a Manifest
  - Set Environment Variables

# Roadmap

- **Manifest Files**
- Environment Variables



# Cloud Foundry Manifest file – 1

- Describes the application deployment options
  - Automates subsequent deployments
  - Same options as `cf push` command
    - Plus options *only* available in the manifest
- Create using your favorite text-editor
- Or run
  - `cf create-app-manifest <app-name>`
  - Will create a manifest reflecting how app was pushed



# Cloud Foundry Manifest file – 2

- Default name: `manifest.yml`
  - YAML\* format
  - Human friendly data serialization standard
  - Supported by many programming languages
  - Less verbose than XML, similar to JSON
  - <http://www.yaml.org>

\* *YAML Ain't Markup Language*



# Using a Manifest with Push

- `cf push` automatically detects manifest
  - In current directory or parent directories
  - Expects file named `manifest.yml`
  - Override with `-f` option
    - `cf push -f dev-manifest.yml`
  - Or ignore with `--no-manifest` option.
- No manifest found?
  - `cf push` will default all deployment options
    - Not the best choices
    - *Different* to previous version of `cf` (which prompted)



# YAML Format

- 3 dashes
  - Indicate start of document
- Indent with spaces, not tabs!
  - Determines hierarchy
  - Each indent 2 spaces
  - “-” defines “group”
- Syntax: `property: value` pairs
- `#` starts a one line comment
- See <http://www.yaml.org/spec/1.2/spec.html>

```
---
```

```
applications:
```

```
- name: nodetestdh01
```

```
  memory: 64M
```

```
  instances: 2
```

```
  host: crn    # comment
```

```
  domain: cfapps.io
```

```
  path: .
```

```
  # comment
```

```
- name: nextapp    # group 2
```

```
  memory: 256M
```

```
  ...
```

# manifest.yml Example



```
---
```

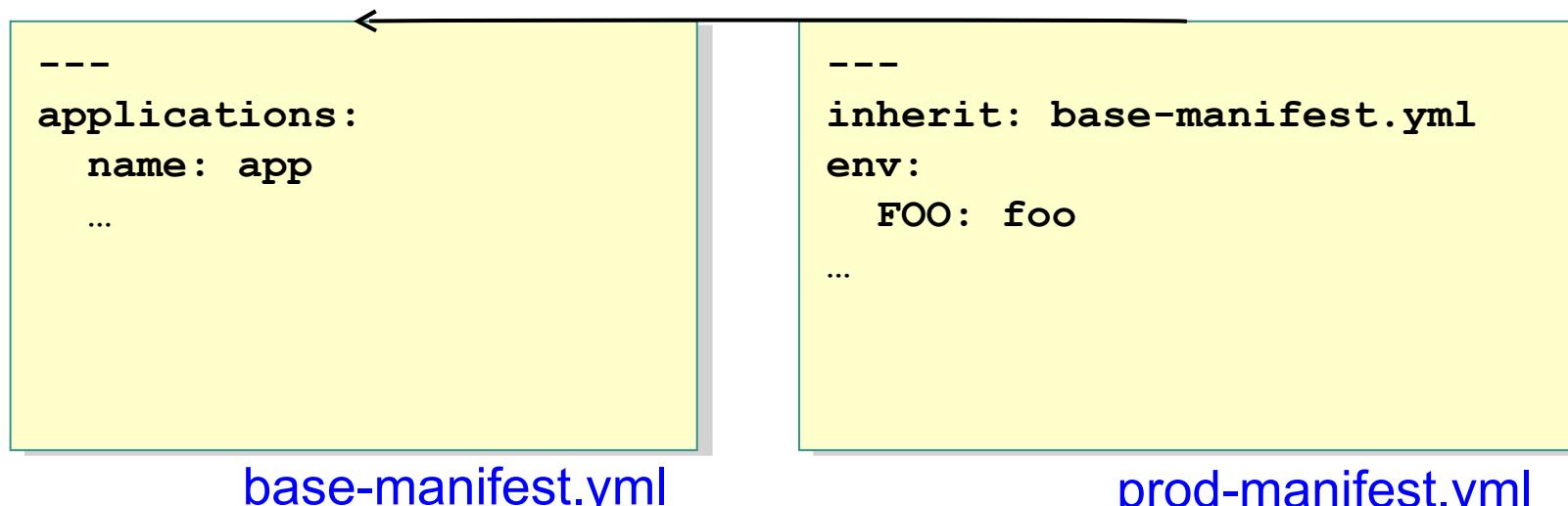
```
applications:
- name: cf-node-demo
- memory: 128M
  instances: 1
  host: demo-${random-word}
  domain: cfapps.io
  path: .
```

- Applications: can describe one or more applications
- Name: of the app – used in commands
- Command: command to run (optional)
- Memory ceiling / instances to run
- Host: your choice, must be unique (within domain)
  - Tip: \${random-word}
- Path: to executable



# Manifest Inheritance

- You may wish to have multiple manifests for an App
  - Different manifests for each space
- One manifest can “inherit” from another

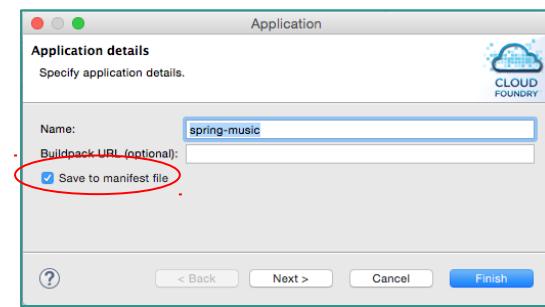


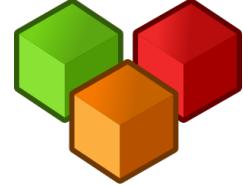
base-manifest.yml

prod-manifest.yml

# Manifests and STS

- STS users can create a manifest
  - Checkbox during push
  - Or after an application has been pushed
    - Applications tab of Server properties

A screenshot of the Pivotal STS interface. On the left, the 'Applications' tab shows a deployed application named 'spring-music'. In the center, the 'Services' tab lists various services: elephantsql, feedbackdb-mysql2, logdrain, new-relic, sendmail-dev, and session-replication, each with its vendor and provider information. On the right, the 'General' panel shows the application's name as 'spring-music', mapped URLs as 'spring-music-**pc1**.cfapps.io', instances as 1, and a manifest file. A large red circle highlights the 'Save' button in the general panel. Other sections include 'Application Operations' with buttons for Start, Stop, Update and Restart, Push, and Debug; 'Application Services'; and 'Instances'.



# Note on Spaces

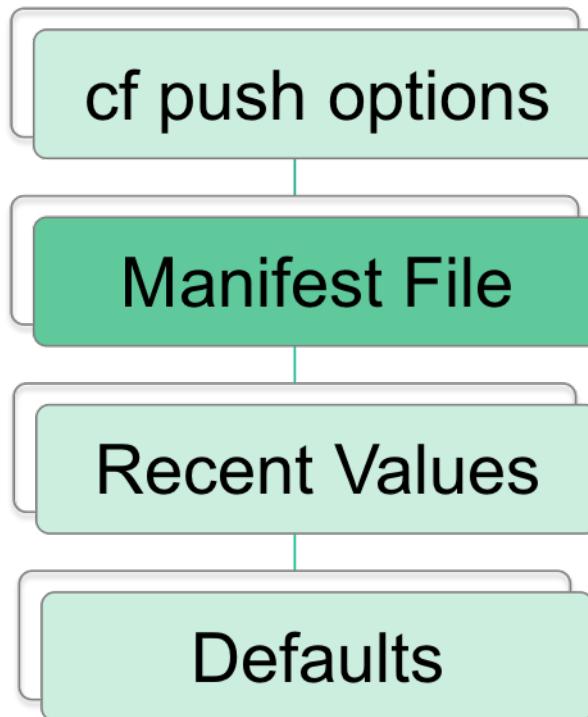
- A Space *cannot* be specified in a manifest file
  - Set first: `cf target -s development`
- To determine the current space
  - Just run: `cf target`



# Manifest vs CLI

- A manifest reduces the amount of typing when deploying via CLI
  - Purpose is to make deployment easily *repeatable*
- Options specified via CLI override options specified via manifest
  - Example: **cf push my-app -i 8 -m 1024M**
  - Deploys 8 instances with 1024 M limit each, regardless of manifest settings

# Precedence Rules



<= Always take precedence

<= If there is one

<= Values from a previous push

<= All options have defaults,  
but you may not like them!

# Defining Services in Manifest

- Add required services to manifest

```
---  
applications:  
- name: spring-music  
  memory: 512M  
  instances: 1  
  host: spring-music  
  domain: cfapps.io  
  path: build/libs/spring-music.war  
# services, one per line  
services:  
- mypg  
- mydb
```

# Defining User Provided Services in Manifest

- Service credentials in manifest

```
---
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
```

# Roadmap

- Manifest Files
- **Environment Variables**

# Environment Variables

- Key / value pairs
  - Used for anything you like
- Set via command line
  - `cf set-env <app-name> <env-var-name> [<value>]`
  - Requires re-staging (i.e. `cf restage` or `cf push`) to take effect
- Or use App Manager or Eclipse/STS plug-in

# Environment Variables – Manifest

- Or specify via manifest

```
---
```

```
env:    # global, all apps
  spring_profiles_active: dev
  another_variable: foo
applications:
  ...
```

```
---
```

```
applications:
- name: myapp
  memory: 256M
  instances: 1
  host: crn
  domain: cfapps.io
  env:    # this app only
    spring_profiles_active: dev
    another_variable: foo
```

# Environment Variables - Precedence

- Environment variables via manifest take precedent over CLI
  - Opposite from the push options defined earlier!
- Example:
  - `cf set-env app FOO fromCLI`
  - `cf push`
  - Result? 'fromManifest'!
- Use `cf push app --no-manifest` to bypass manifest values.

```
---  
env:  
  FOO: fromManifest  
applications:  
  name: app
```

# Environment Variables - Persistence

- Environment variables retain their values
  - Whether application is running or not.
- To view use: **cf env <app>**
- Use **cf unset-env <app> <var>** to remove
- If changed while app is running
  - Use **cf restage <app>** to make change take effect

# Environment Variables - Accessing

- CF environment variables are available to applications
  - Appear like any other environment variable
- Access via...
  - Java: `System.getenv("some_variable");`
  - Ruby: `ENV['some_variable']`
  - Node.js: `process.env.some_variable`

# Environment Variables - Groups

- **System-Provided** env vars
  - e.g. VCAP\_APPLICATION, VCAP\_SERVICES
- **User-Defined** env vars
  - e.g. cf set-env app foo bar
- **Running** env vars
  - system-wide variables, apply to all running apps (Ops only)
- **Staging** env vars
  - system-wide variables, apply to all staging apps (Ops only)

# Environment Variables - VCAP\_APPLICATION

- Information on memory, instances
  - JSON formatted object (described later):

```
{  
  "instance_id": "451f045fd16427bb99c895a2649b7b2a",  
  "instance_index": 0,  
  "host": "0.0.0.0",  
  "port": 61857,  
  "started_at": "2013-08-12 00:05:29 +0000",  
  "started_at_timestamp": 1376265929,  
  "start": "2013-08-12 00:05:29 +0000",  
  "state_timestamp": 1376265929,  
  "limits": {"mem": 512, "disk": 1024, "fds": 16384},  
  ...  
}
```

# Environment Variables - VCAP\_SERVICES

- Information on all bound services
  - JSON formatted object

```
{ "elephantsql": [ { "name": "elephantsql-c6c60", "label": "elephantsql-n/a", "tags": [ "postgres", "postgresql", "relational" ], "plan": "turtle", "credentials": { "uri": "postgres://PHxTPn@babar.elephantsql.com:5432/se1mbd" } }, "sendgrid": [ { "name": "mysendgrid", "credentials": { "username": "QvsXMBJ3rK", "password": "HCHMOYluTv" } } ] }
```

# Environment Variables - Management

- Not all environment variables can be changed
  - Those set by CF runtime cannot
    - Such as **VCAP\_SERVICES** (set by service binding)
    - See URL below for list of variables set by runtime
- To view environment variables:
  - **cf env [app-name]**
    - Displays user defined and system defined variables
    - Some variables only available to running instances

<http://docs.pivotal.io/pivotalcf/devguide/deploy-apps/environment-variable.html>

# Summary

- After completing this lesson, you should have learned:
  - Use of environment variables
  - All about application manifests

# Lab

## Using a Manifest

# Application Security Groups

Controlling Outgoing Requests

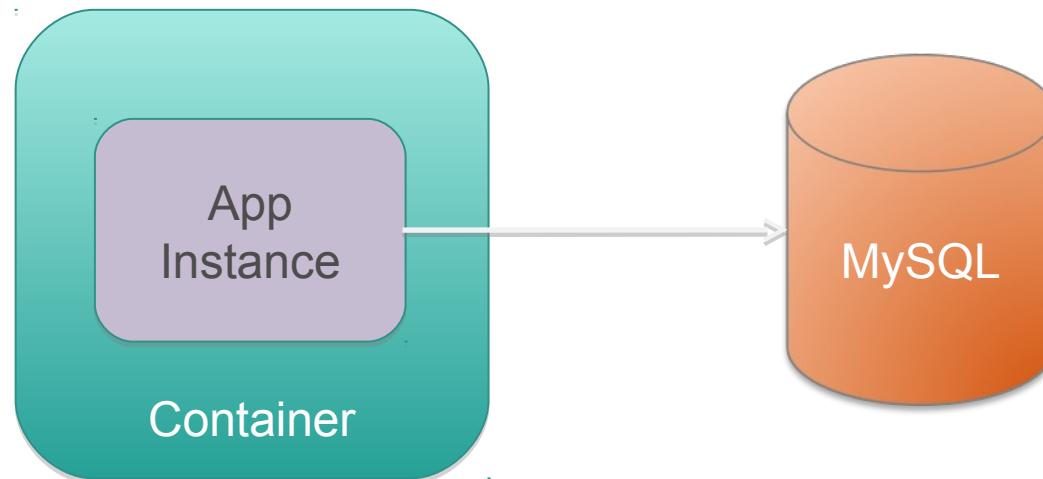
*Administration Facility*

# Agenda

- Application Security Groups Overview

# Application Security Groups

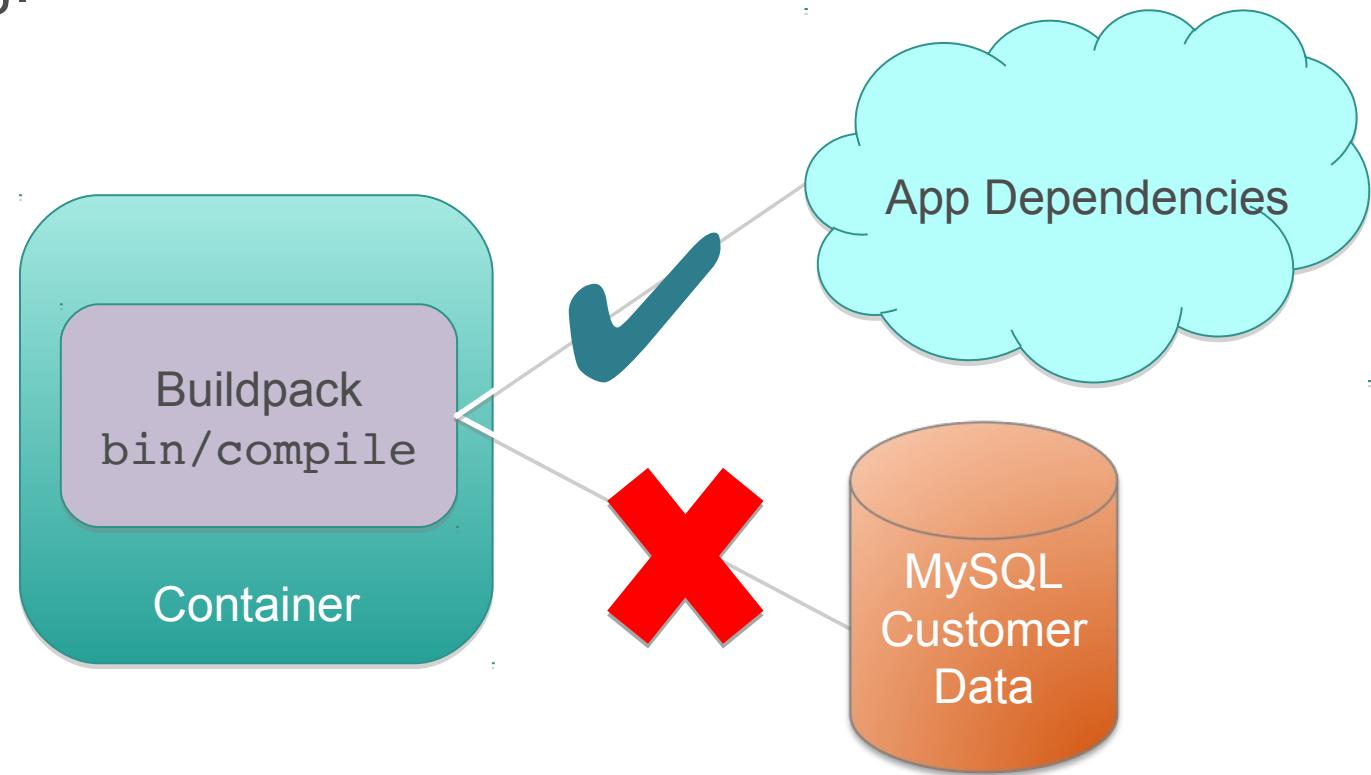
- Are virtual firewalls that control egress/outbound traffic for applications.



<https://docs.pivotal.io/pivotalcf/adminguide/app-sec-groups.html>

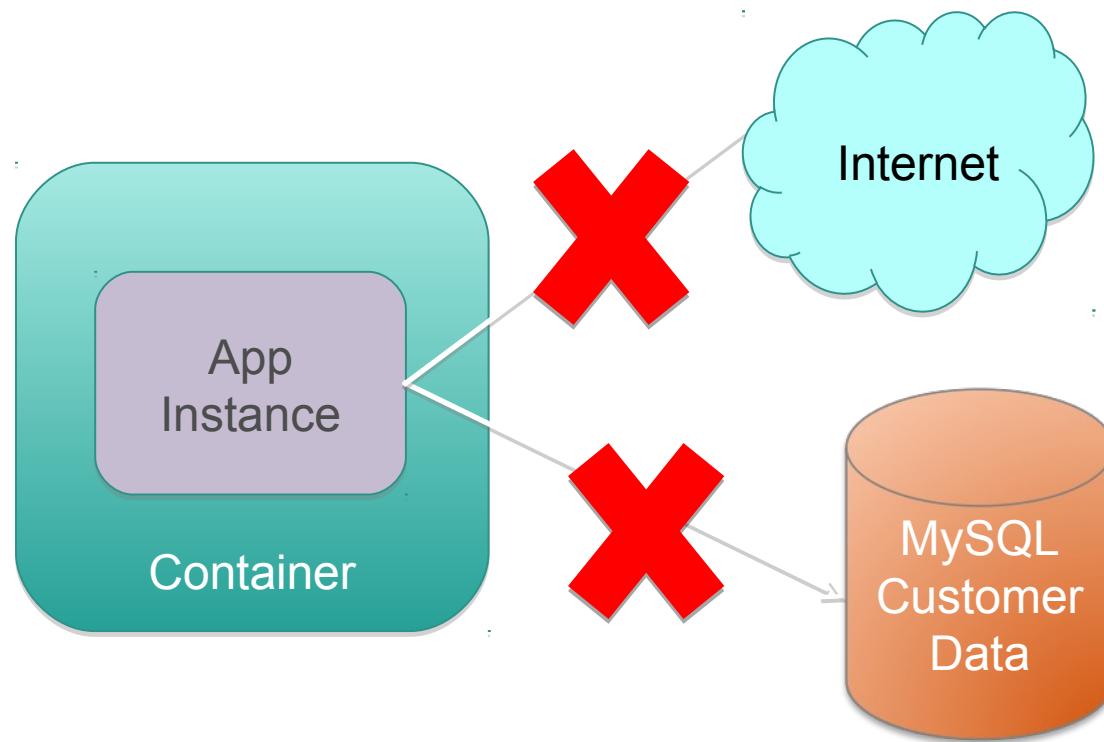
# Staging Security Groups

- Whitelist what the app should have access to when staging.



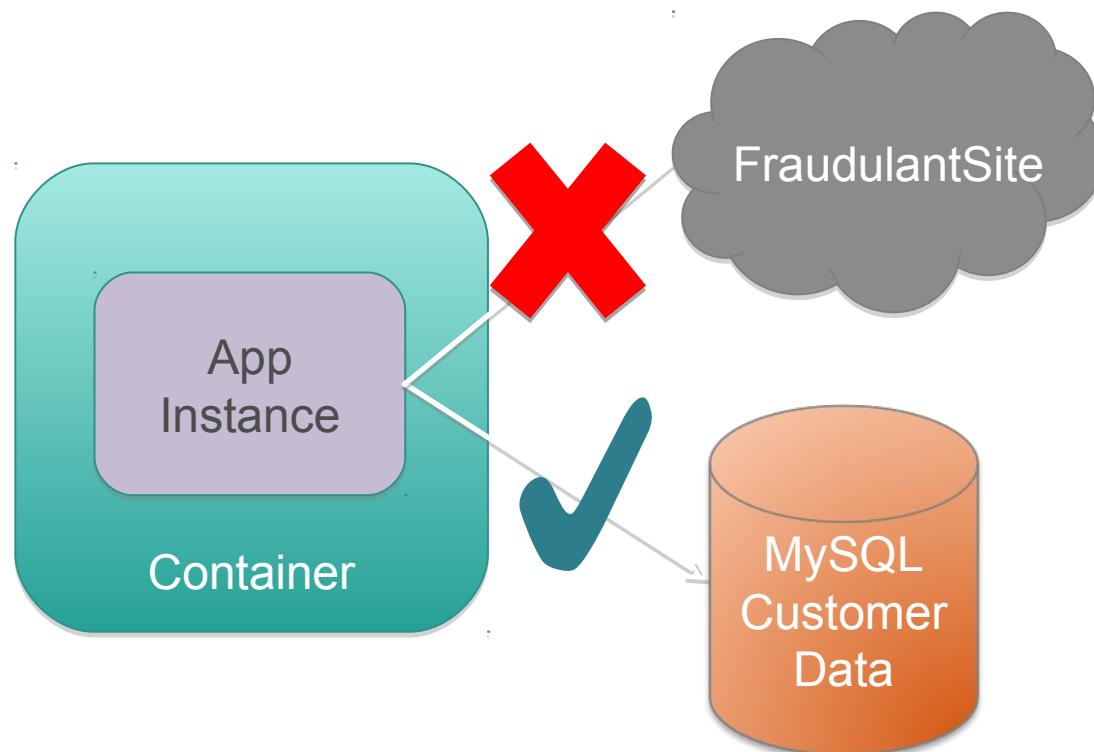
# Running Security Groups

- Whitelist what all apps should have access to when running.



# Space Security Groups

- Whitelist what the apps in a space should have access to when running.



# Default Security Group

- PCF

```
$> cf security-groups
Getting security groups as user@domain
OK

      Name          Organization     Space
#0    default_security_group
```

- PWS

```
$> cf security-groups
Getting security groups as user@domain
OK

      Name          Organization     Space
#0    public_networks
```

# View Default Security Group (PCF)

```
$> cf security-group default_security_group
Getting info for security group default_security_group as ...
OK

Name      default_security_group
Rules
[
  {
    "destination": "0.0.0.0-169.253.255.255",
    "protocol": "all"
  },
  {
    "destination": "169.255.0.0-255.255.255.255",
    "protocol": "all"
  }
]
No spaces assigned
```

# View Default Security Group (PWS)

```
$> cf security-group public_networks
Getting info for security group public_networks as user@domain
OK

Name      public_networks
Rules
[
  {
    "destination": "0.0.0.0-9.255.255.255",
    "protocol": "all"
  },
  ... several more ...
]
No spaces assigned
```

# Example JSON File

- Only the protocols, IP address and ports listed are allowed
  - TCP to ip-foo:3306
  - TCP in IP range some-ip to another-ip on port 55882

```
[  
  { "protocol": "tcp",  
    "destination": "10.0.11.0/24",  
    "ports": "1-65535" },  
  
  { "protocol": "udp",  
    "destination": "10.0.11.0/24",  
    "ports": "1-65535" }  
]
```

# Usage

Administrator only

- Create a group and then bind to a space or globally

## SECURITY GROUP:

`security-group`

`security-groups`

`create-security-group`

`update-security-group`

`delete-security-group`

`bind-security-group`

`unbind-security-group`

Show a single security group

List all security groups

Create a security group

Update a security group

Deletes a security group

Bind a security group to a space

Unbind a security group from a space

`bind-staging-security-group`

Bind a security group to the list of security groups  
to be used for staging applications

`staging-security-groups`

List security groups in the staging set for  
applications

`unbind-staging-security-group`

Unbind a security group from the set of security  
groups for staging applications

`bind-running-security-group`

Bind a security group to the list of security groups  
to be used for running applications

`running-security-groups`

List security groups in the set of security groups for  
running applications

`unbind-running-security-group`

Unbind a security group from the set of security  
groups for running applications

# Log Draining

Using a third-party log manager

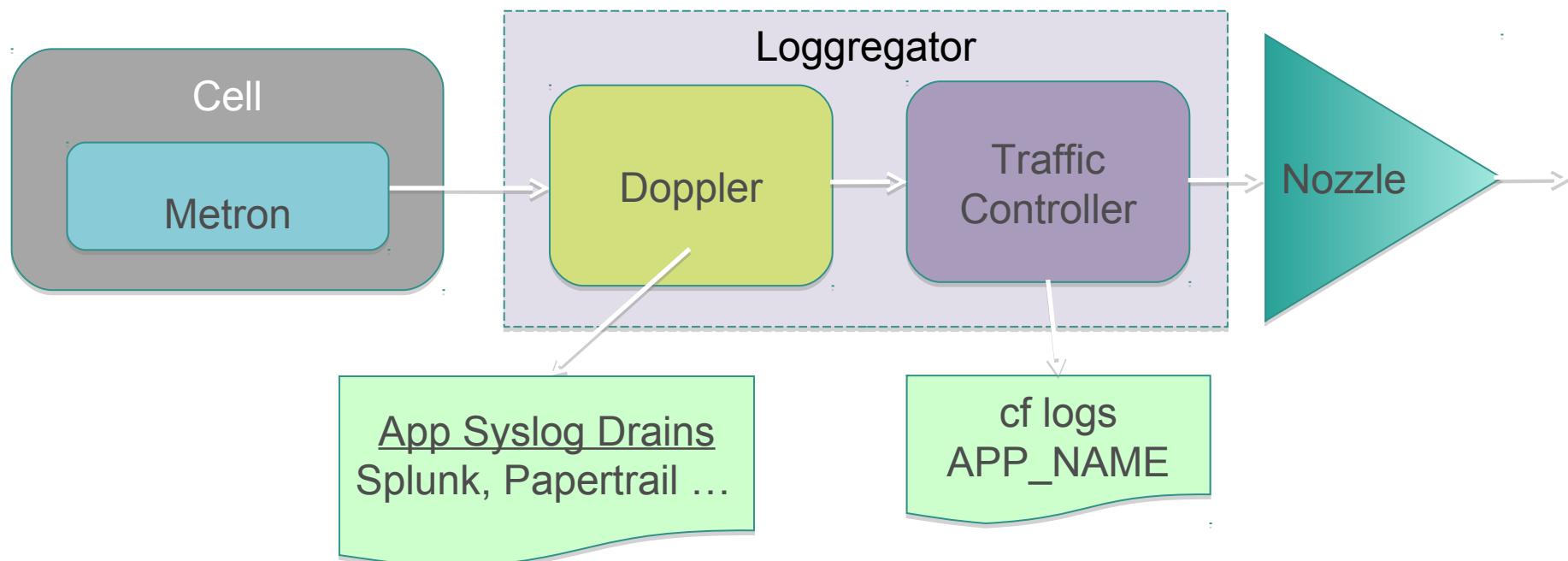
Setting up a syslog drain

# Roadmap

- Log Management

# Recall: Log Aggregation Architecture

- Collects log output from app instances, CF components
- Aggregates into a consolidated log
- Sinks to cf logs, App Mgr, *third-party log managers*



# Why Third-Party Log Managers?

- Recommended approach
  - Can store far more logging information than CF
  - Allow for persistence, storage, *searching, analyzing, metrics*
- Variety of third-party log managers supported:



papertrail



splunk>storm™

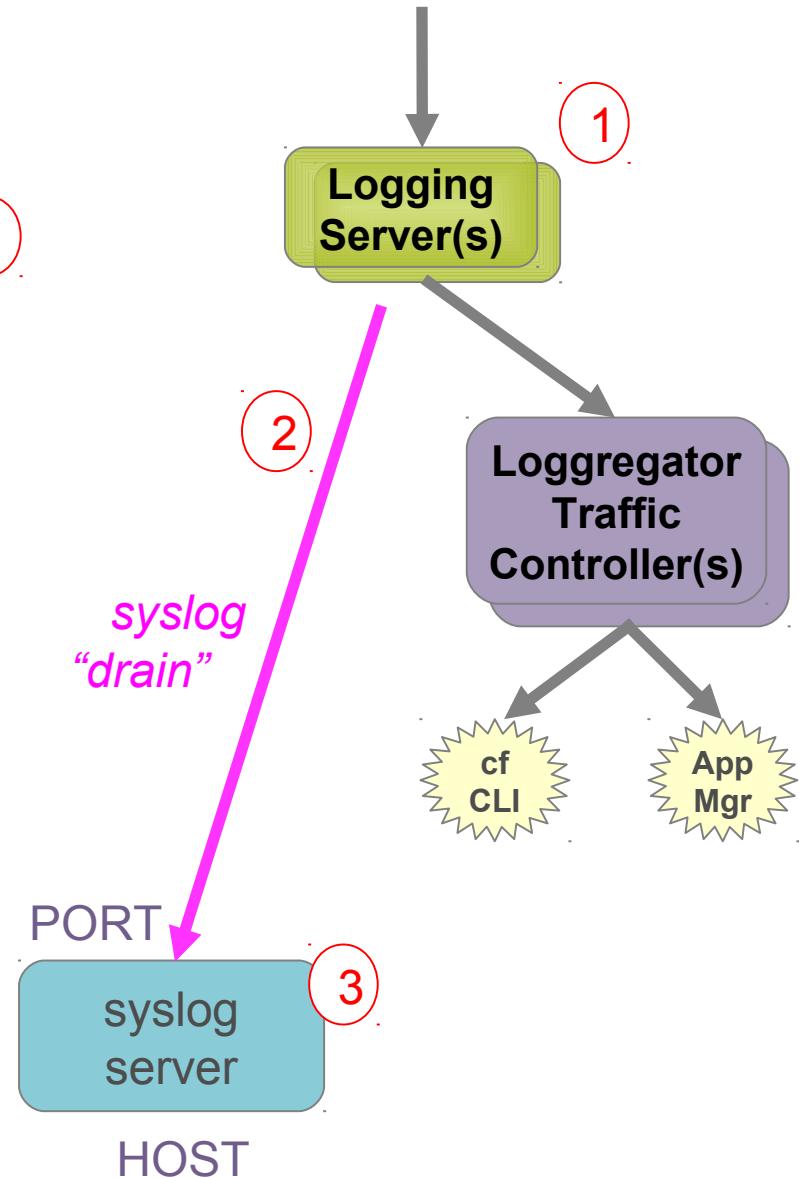
Pivotal™

# Connecting to Third-Party Log Managers

- Setup Log Manager, determine **HOST** and **PORT**.
  - Process varies according to vendor
- Create User Provided Service with a Syslog drain:
  - `cf cups <SERVICE> -l syslog://<HOST>:<PORT>`
- Bind to application
  - Cloud Foundry sinks loggregator output to this drain for this application
  - See *next slide* ...

# How It Works

- All output for app collected by Logging (Doppler) server
- Loggregator opens socket to HOST:PORT
  - Sends all log info for app to socket in syslog format
- Received by third-party syslog server



# Example: PWS and PaperTrail

- **PaperTrail:** Cloud-based Log Manager
  - a) Create account at <https://papertrailapp.com>
  - b) Use the “Add System” button
    - Papertrail will provide you the URL to use for your syslog drain
    - Example: logs2.papertrailapp.com:41846

# Example: PWS and Papertrail

c) Click the “Alternatives” link

d) Select “*Cloud Foundry*” option

e) Name your system

Choose your situation:



**A My syslogd only uses the default port**

GNU syslogd and some embedded devices will only log to port 514. A few old Linux distro versions use GNU syslogd (mostly CentOS and Gentoo).



**B I use Cloud Foundry**

Register each app separately. Use Heroku? [Here's how.](#)



**C My system's hostname changes**

In rare cases, one system may change hostnames frequently. For example, a roaming laptop which sets its hostname based on DHCP (and roams across networks).

We'll provide an app-specific syslog drain and step-by-step setup for [Cloud Foundry](#).

Let's create a destination for this app.

What should we call it?

Alphanumeric. Does not need to match app name.

Save →

# Example: PWS and PaperTrail

- f) Setup user defined service using Papertrail's URL:

MyCFSSystem will log to **logs2.papertrailapp.com:15957**.

- g) Create User Provided Service with a Syslog drain:

```
cf cups the-drain -l  
syslog://logs2.papertrailapp.com:15957
```

- h) Bind to application:

```
cf bind-service the-app the-drain
```

# About Syslog

- De facto standard for logging on Unix/Linux
  - Can log to a file or a *syslogd* server (via a protocol)
  - Splunk, Papertrail and others provide syslog servers
- To log to syslog
  - Generate a TCP or UDP message in the right format
  - Open a socket to your syslog server and send

# Lab

Configuring Third-Party Log  
Management Tools with  
Cloud Foundry

# Blue-Green

Performing Rolling Deployments

Upgrading with zero downtime

# Roadmap

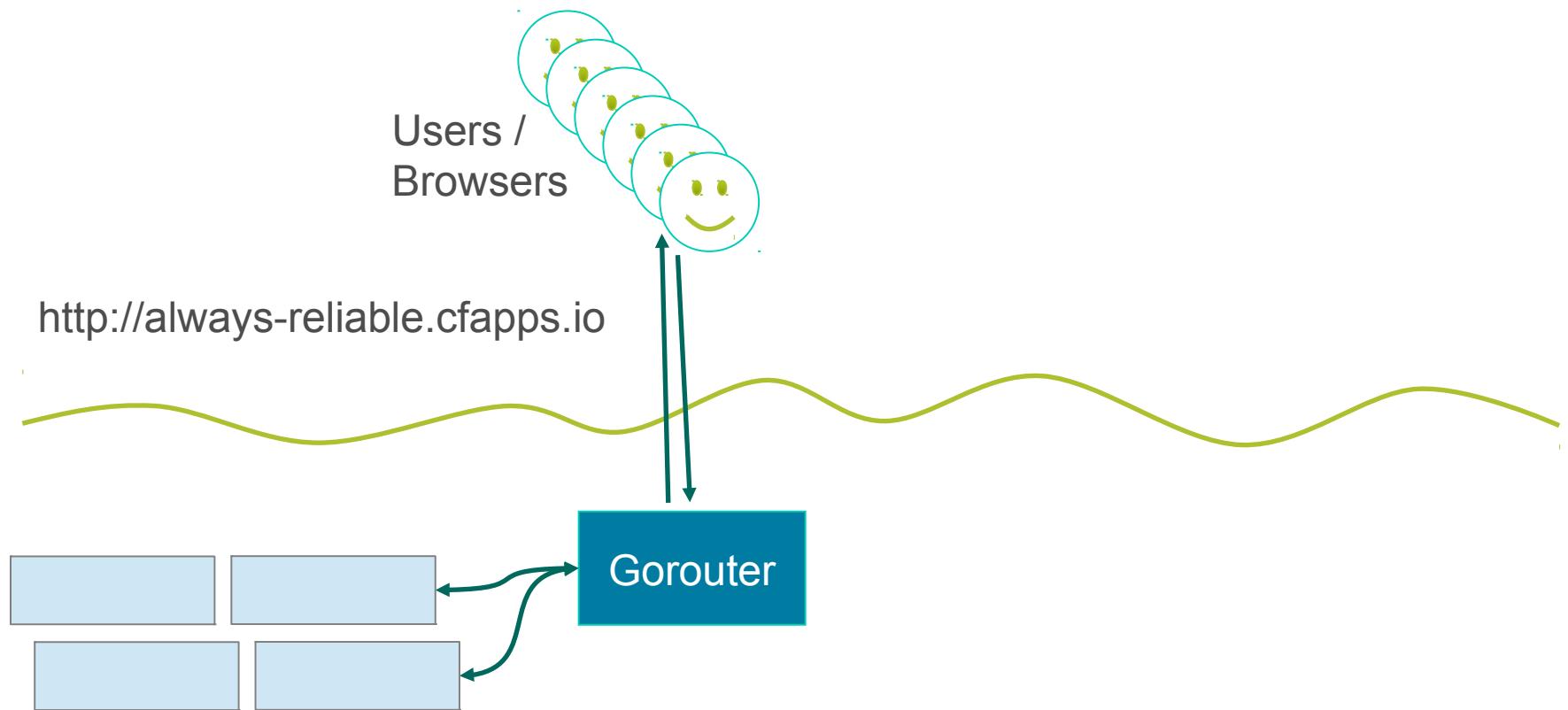
- **Zero-Downtime Deployments**
- Implications for Application & Data Model Design

# Blue / Green Deployments

- **cf push** causes CF to stop old instances, then start new
  - Bad news if you are a user
- Blue / Green Deployment eliminates user downtime
  - Also known as “zero-downtime” or “A / B” Deployment
  - Avoids “*Site Temporarily Down for Maintenance*”
- How it works:
  - Run 2 versions of an application (new / old)
    - NOT merely multiple instances.
  - Alter routes for applications to transfer traffic.
  - **Note:** Users can still experience session loss.

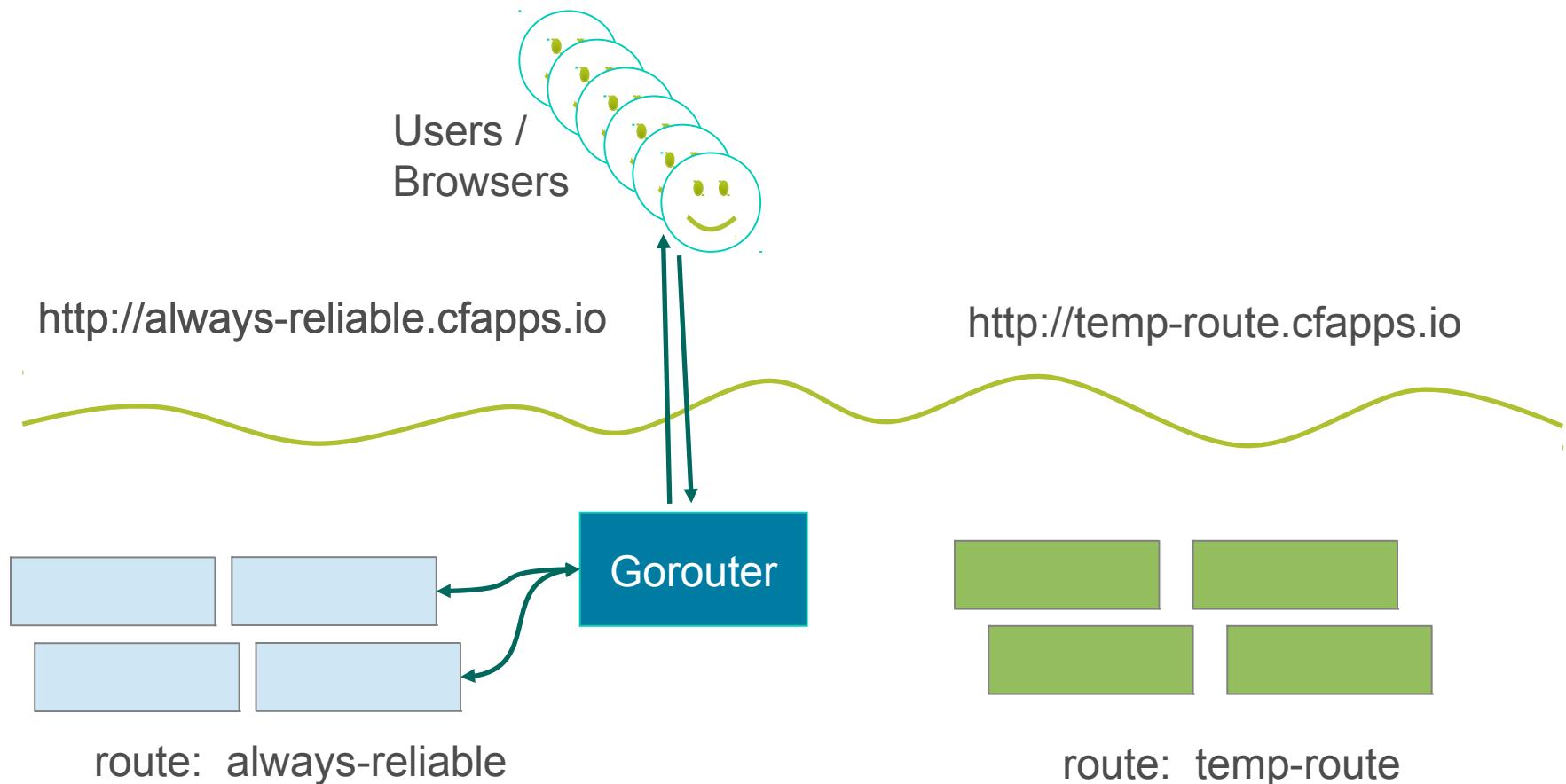
# Blue Green Deployment – Existing App

```
cf push blue -p app.war -n always-reliable -i 4
```



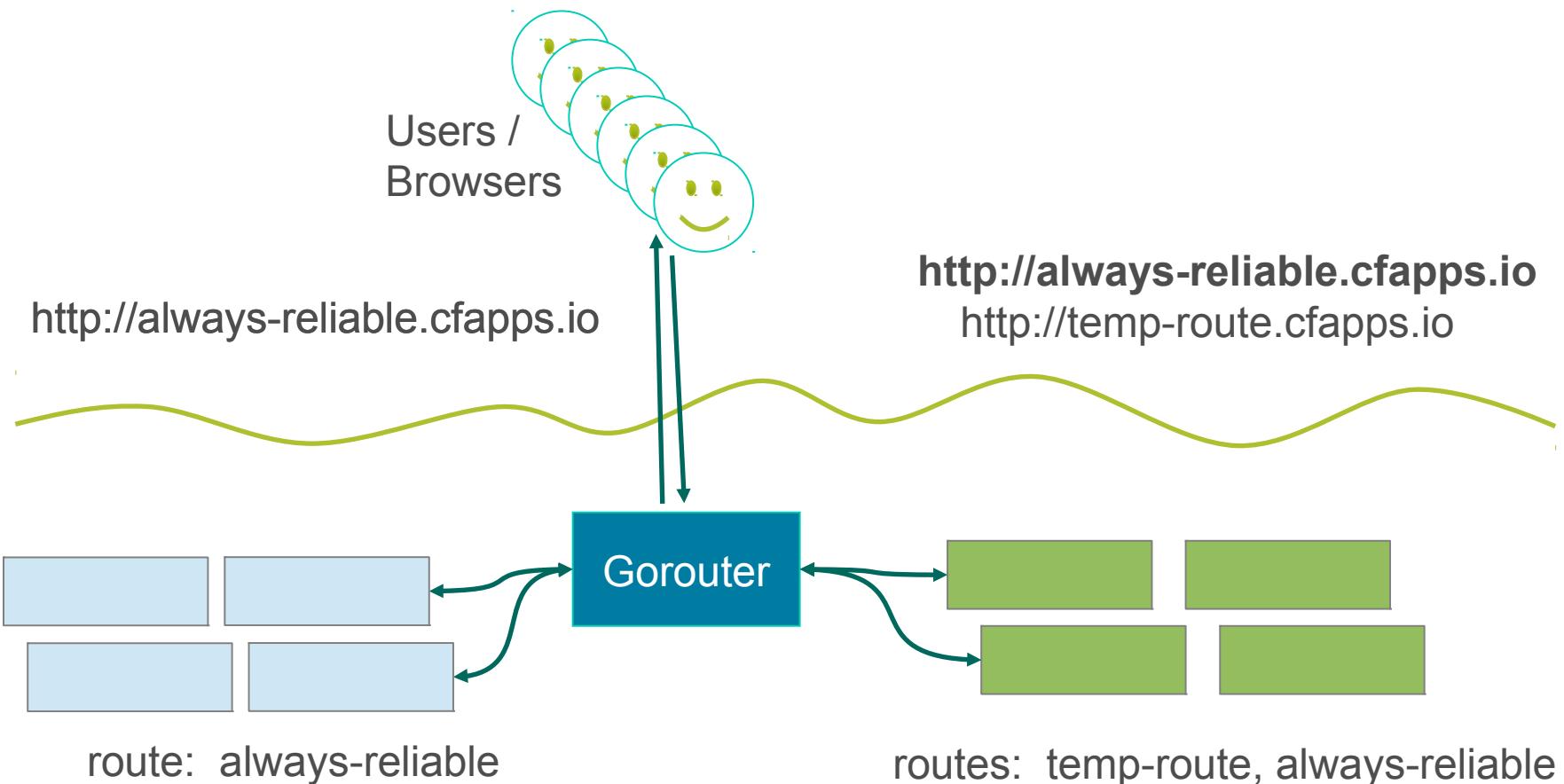
# Blue Green Deployment – New Version

```
cf push green -p app.war -n temp-route -i 4
```



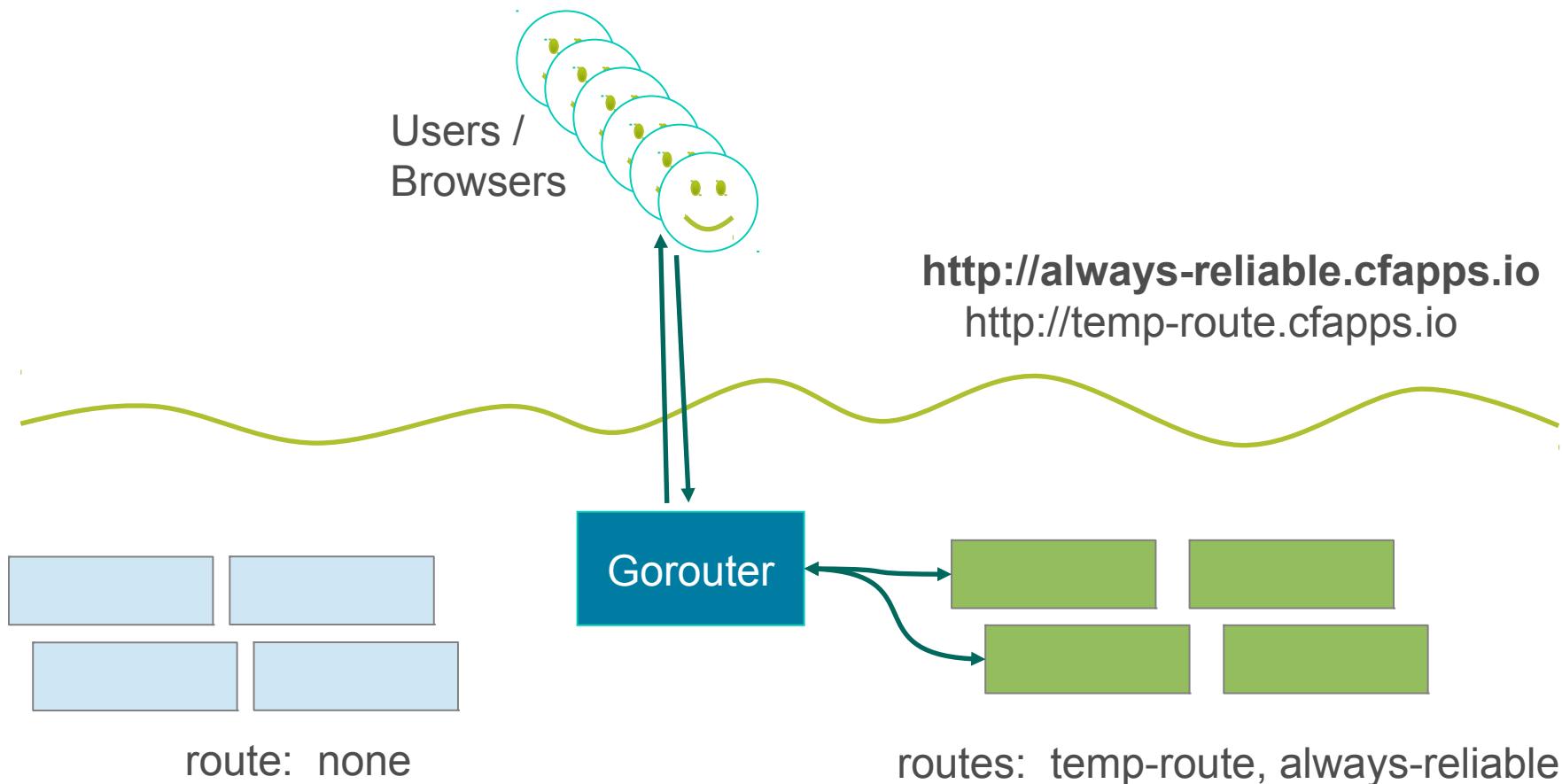
# Blue Green Deployment – Duplicate Route

```
cf map-route green cfapps.io -n always-reliable
```



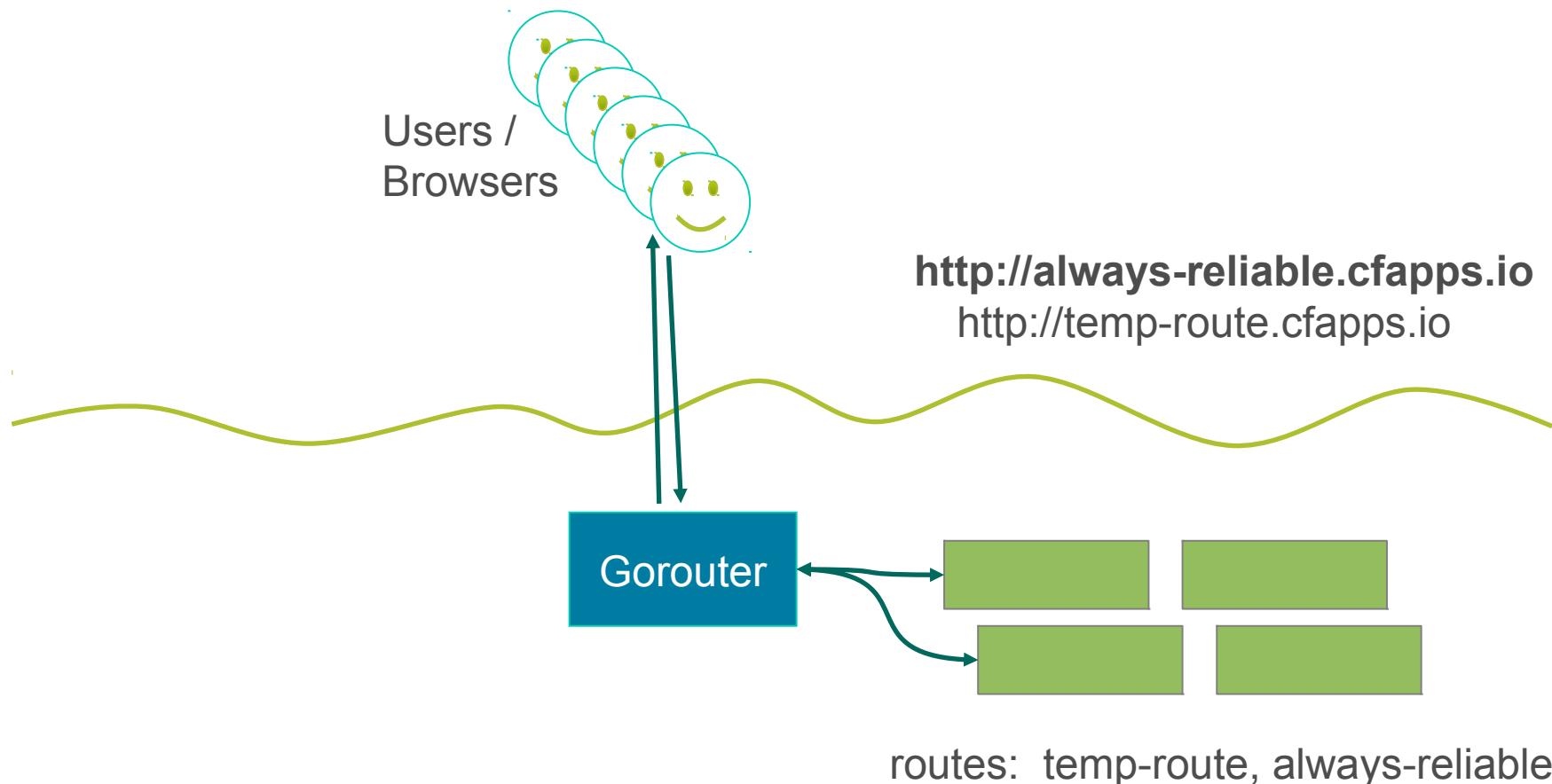
# Blue Green Deployment – Disconnect Blue

```
cf unmap-route blue cfapps.io -n always-reliable
```



# Blue Green Deployment – Remove Blue

`cf delete blue`

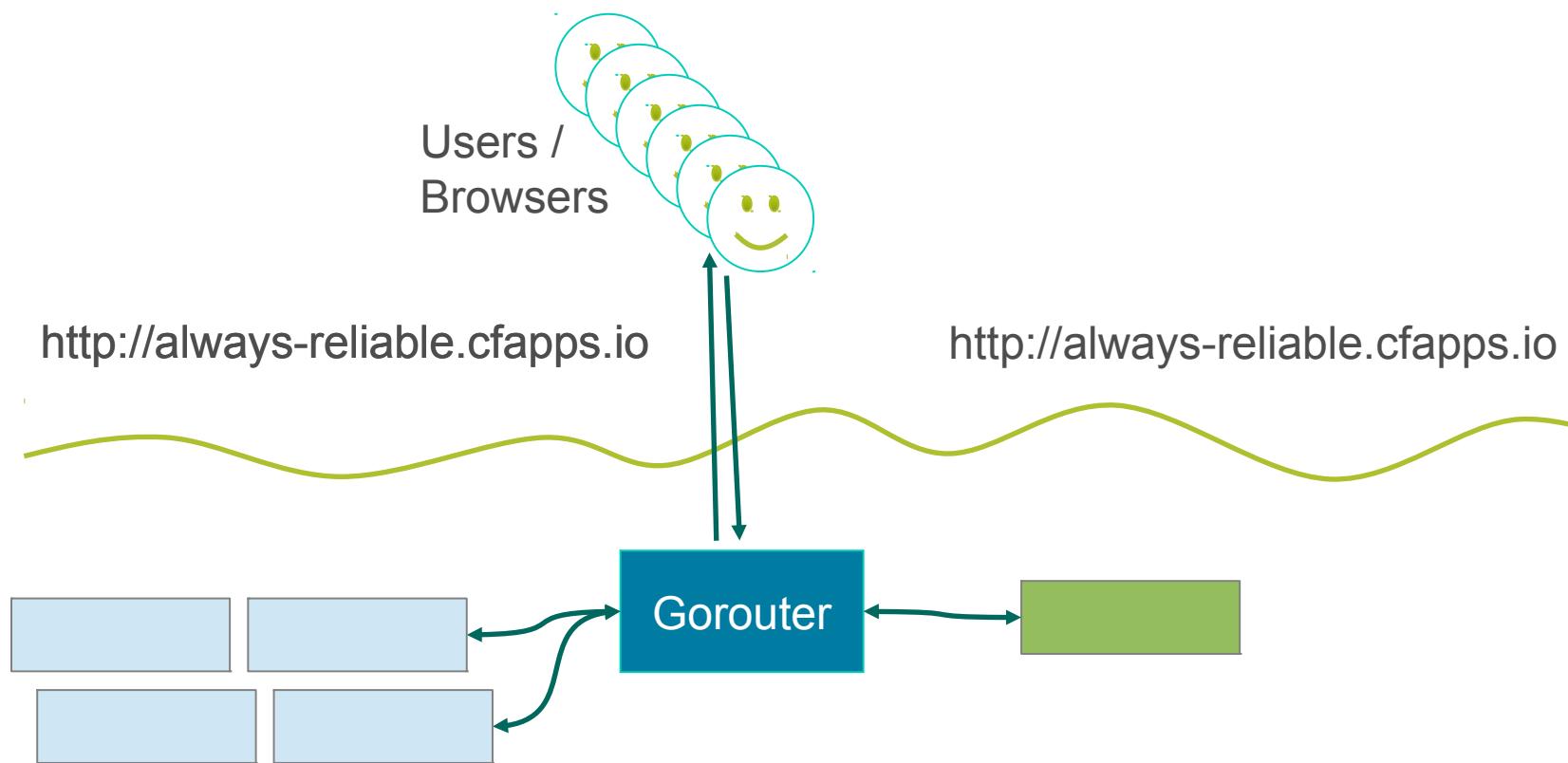


# Canary Deployments

- Variation on the Blue/Green Deployment.
    - “Canary in a coal mine”
1. Start with many 'blue' instances
  2. Start a single 'green' instance, route traffic to both
    - Green instance is the 'Canary'
  3. Watch the Canary
    - If it behaves, scale 'green' up / scale 'blue' down.
  4. Continue monitoring and scaling until zero blue instances

# Canary Deployment – Push The Canary

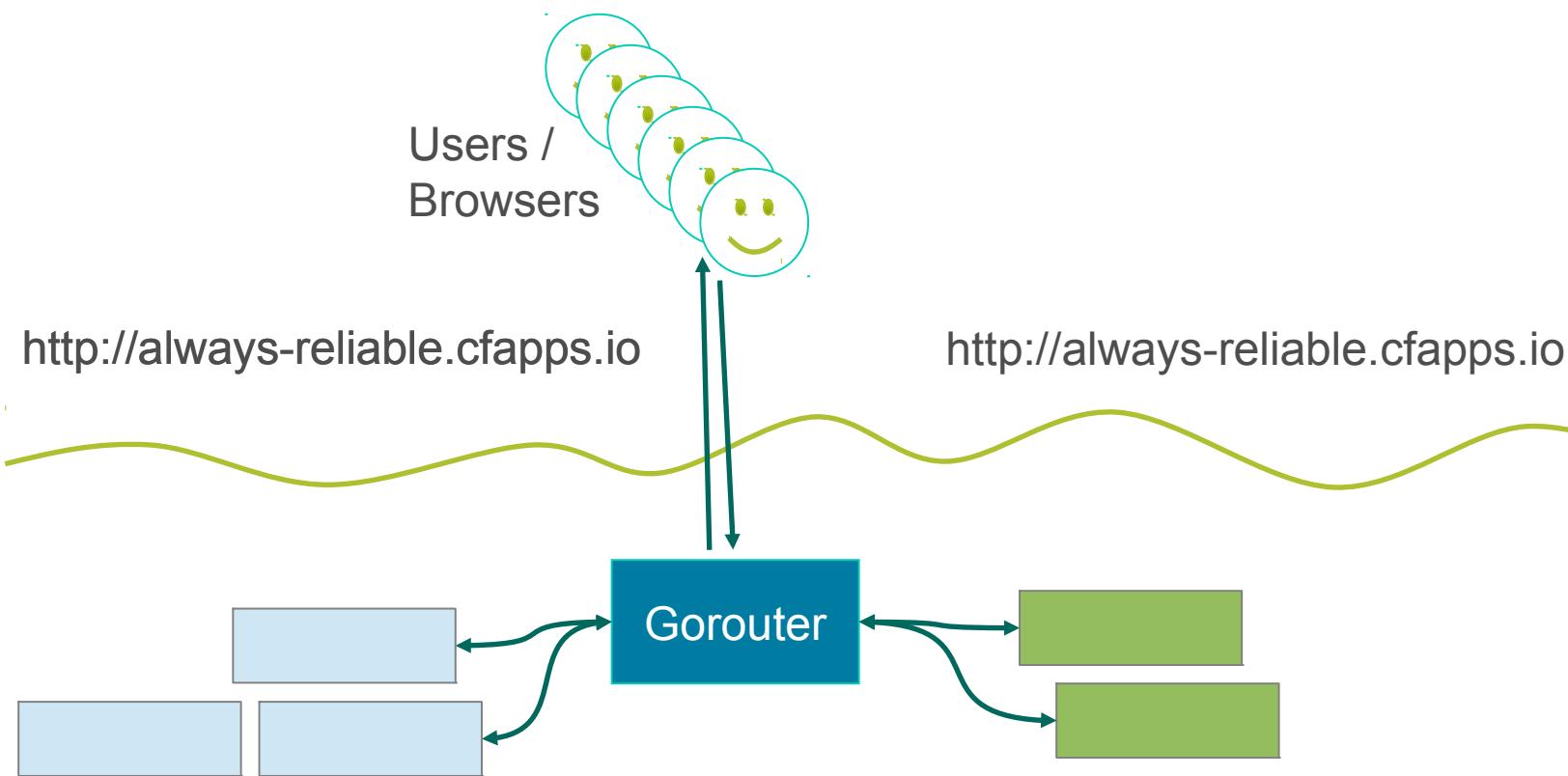
```
cf push green -p app.war -n always-reliable -i 1
```



# Canary Deployment – Scale Traffic

```
cf scale green -i 2
```

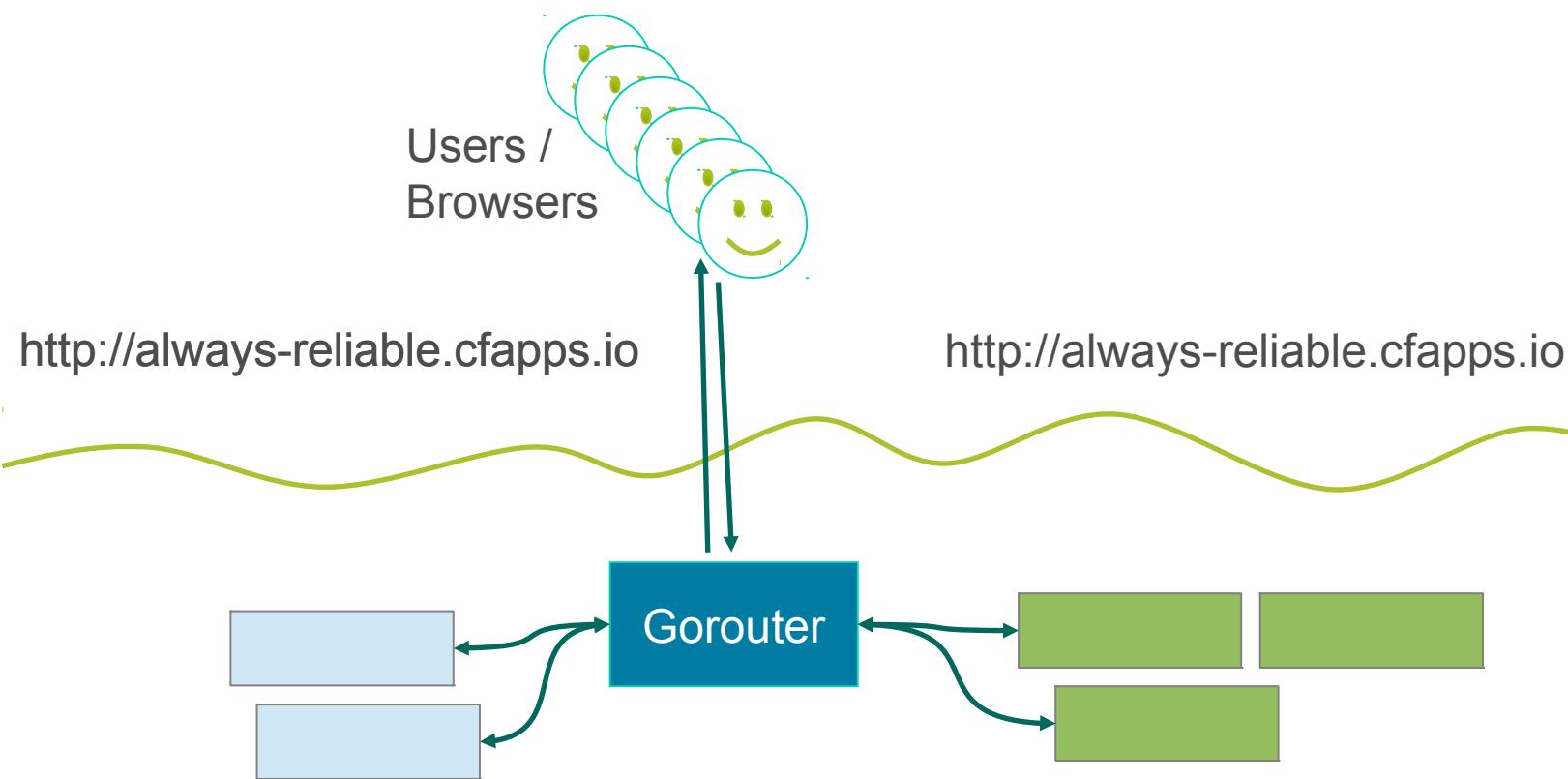
```
cf scale blue -i 3
```



# Canary Deployment – Scale Traffic

```
cf scale green -i 3
```

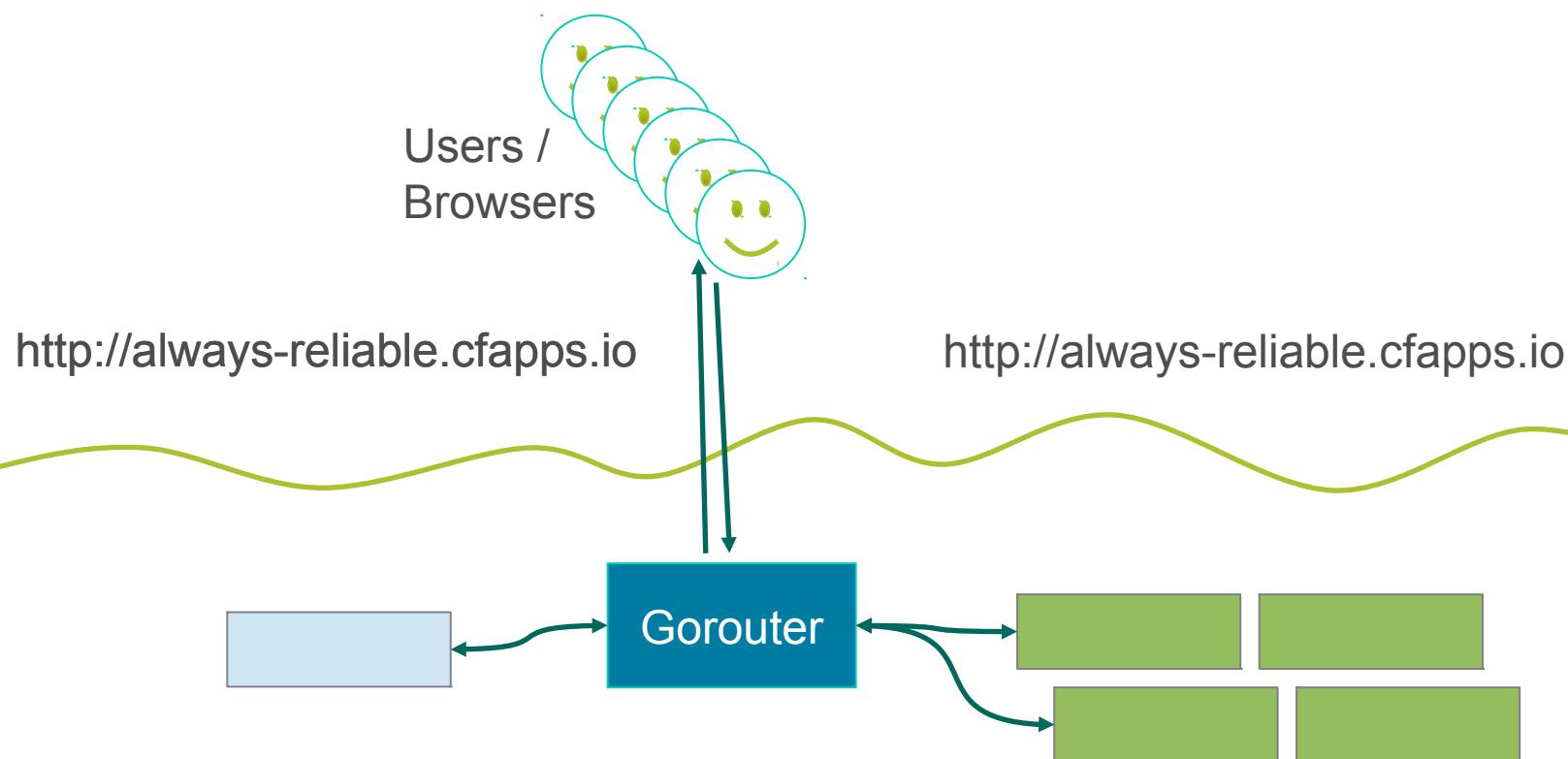
```
cf scale blue -i 2
```



# Canary Deployment – Scale Traffic

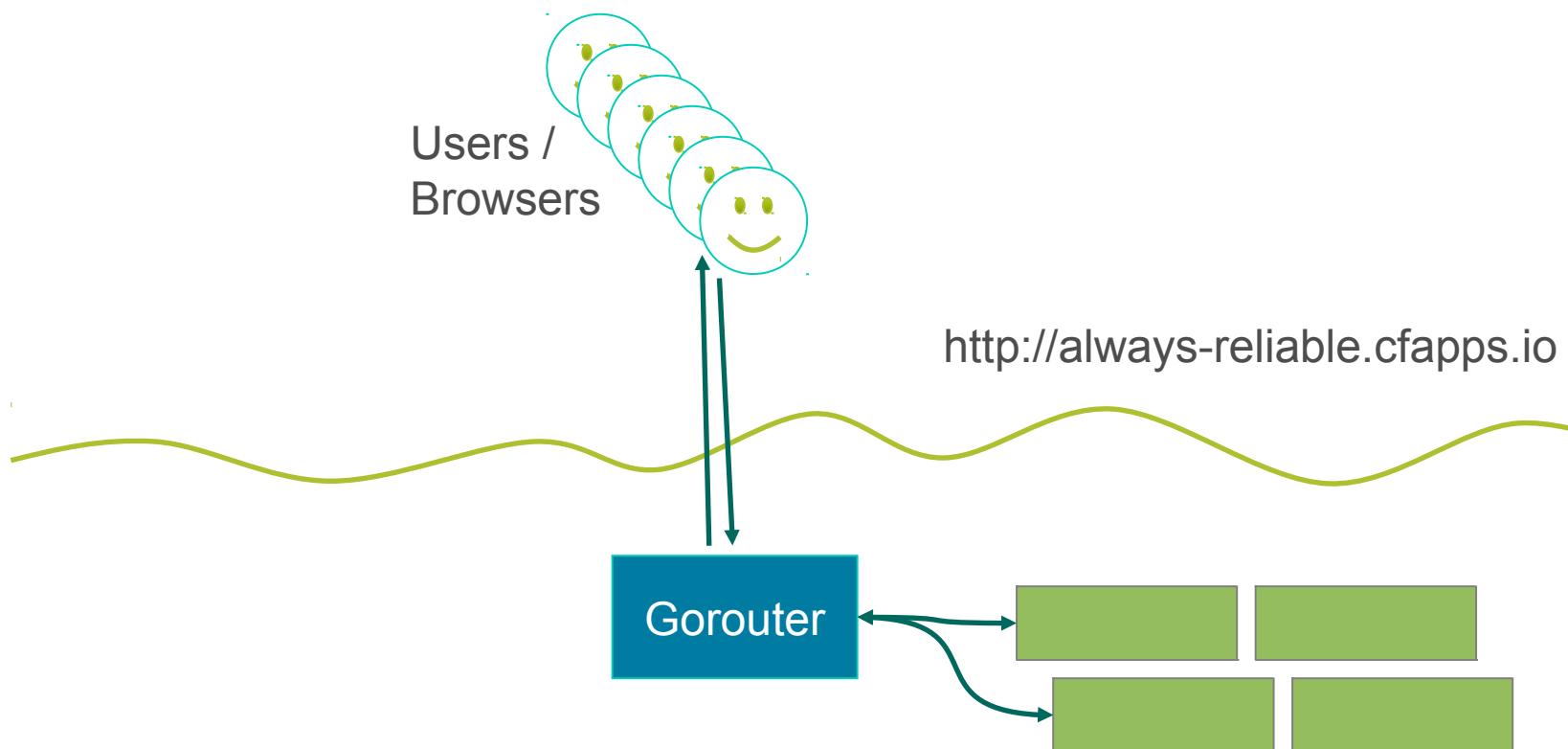
```
cf scale green -i 4
```

```
cf scale blue -i 1
```



# Canary Deployment – Scale Traffic

`cf delete blue`



# blue-green-deploy plugin

```
cf blue-green-deploy <app> --smoke-test <script>
```

- Push new version of app with a new name
- Optionally runs smoke tests against new app:
  - Fail:
    - New version of app marked as failed
    - Left around for investigation
  - Pass:
    - Remap routes from previous app to new app
    - Clean up previous version of app

# Roadmap

- Zero-Downtime Deployments
- **Implications for Application & Data Model Design**

# Ensure Compatible Changes

- Application upgrade often involves database changes
- **BUT:** if you need to rollback, system must still work

# Admin Processes

- Run admin/mgmt tasks as one-off processes.
  - Migrating data

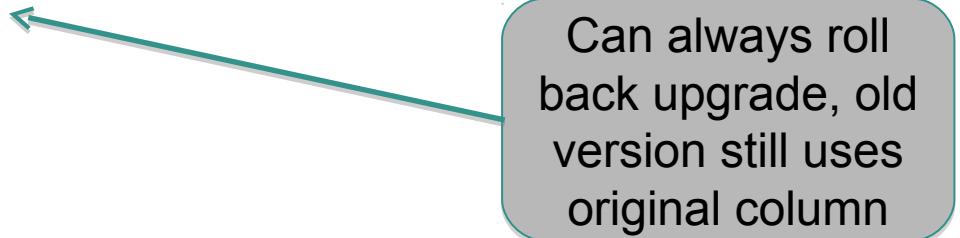
# Database

- Do make changes *idempotent*
  - Example: copy data to a new field (don't delete data)
- Procedure: *Rename Column*
  - Run script to
    - Add new column to database
    - Populate it from original column
  - Upgrade application to new version
    - Ensure it is no longer using the old column
  - Delete original column
- Alternative: use database views

Can always roll back upgrade, old version uses original column

# Database

- No destructive database changes allowed.
  - **Example:** don't drop a column
- *Procedure: Delete Column*
  - New application version no longer requires a column
  - Upgrade application
  - If upgrade goes OK, *and* the app is ignoring the column,  
*then* delete column



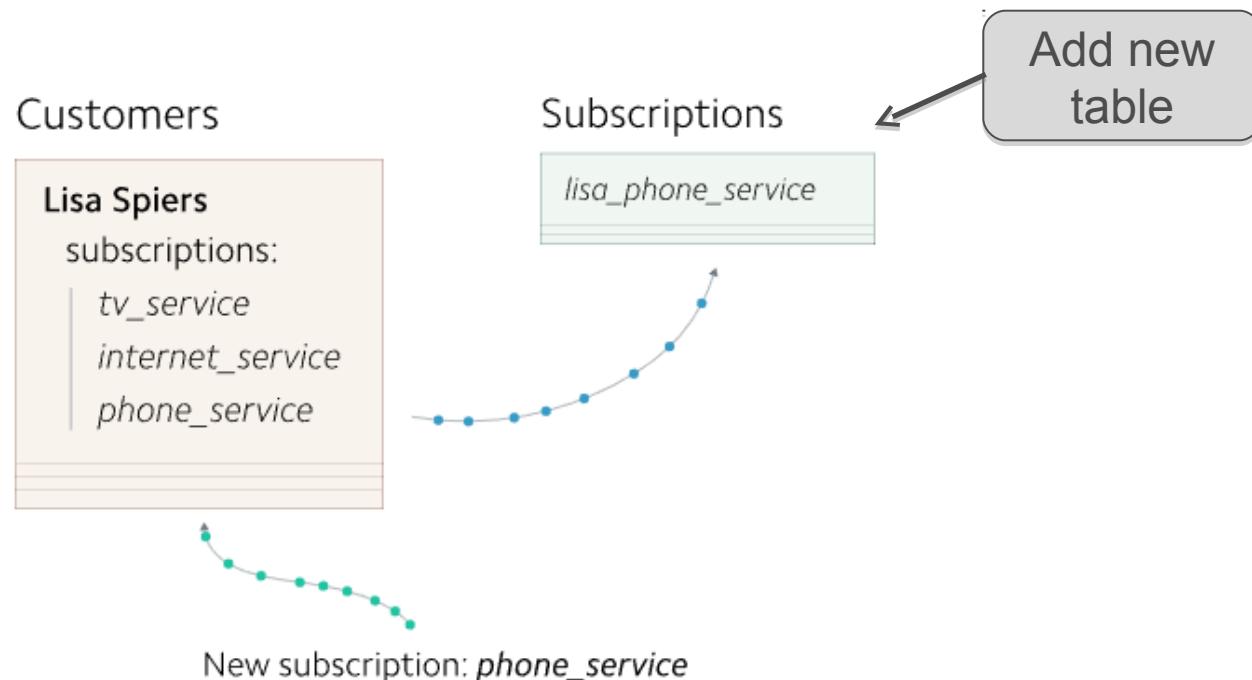
Can always roll  
back upgrade, old  
version still uses  
original column

# Database

- Do have backwards compatible changes.
    - **Example:** nullable fields
  - *Procedure: Add Column*
    - Run script to add new column to database
      - Allow nulls since it has no values yet
    - Upgrade application to new version
      - Over time application puts data in new column
    - Eventually add the not-null (or any other) constraint to the database
- Can always roll back upgrade, old version ignores new column
- 

# Online Migrations at Scale

- Article from Stripe
  - Migration of a 1-1 relationship to a 1-many
    - <https://stripe.com/blog/online-migrations>



# Summary

- Zero-Downtime Deployments
- Implications for Application & Data Model Design

# Lab

## Blue-Green Deployment

# Introduction to Microservices

Monolith vs Microservices

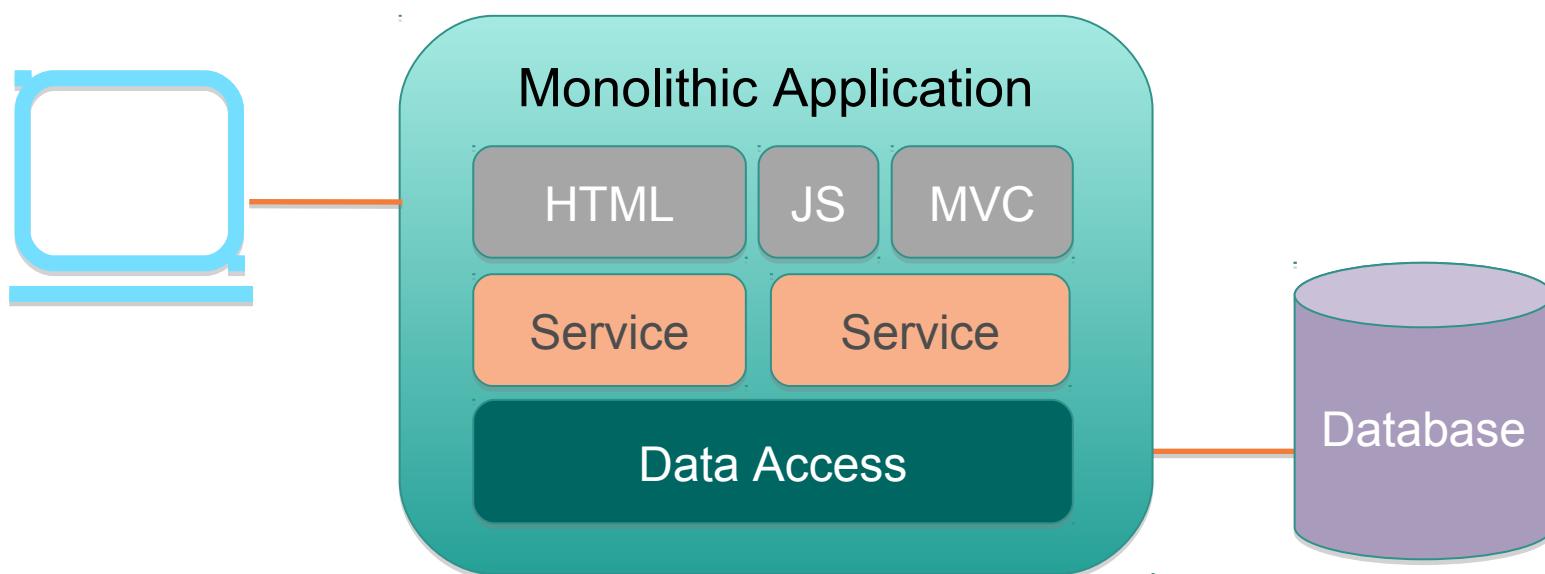
Enabling Microservices with PCF

# Agenda

- **The Monolith**
- Microservices
- Microservices and Pivotal Cloud Foundry

# Monolith

- A three tiered monolith.



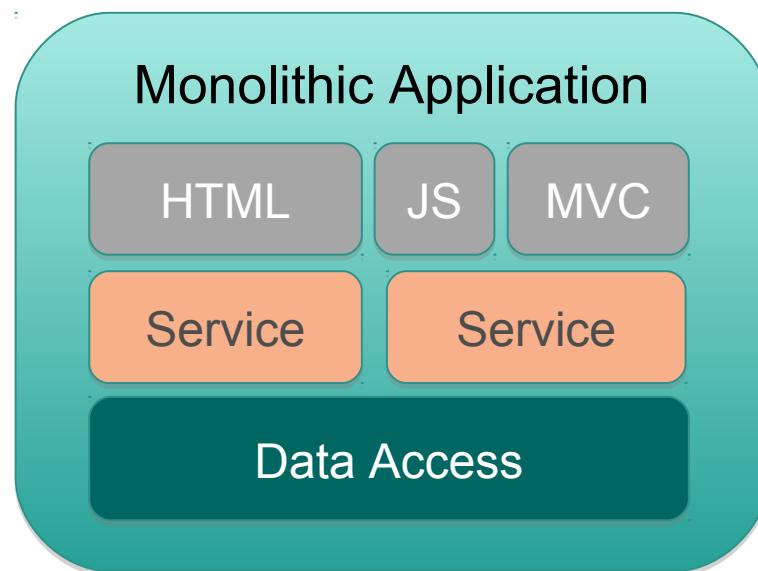
# Monolith challenges

# Monolith Design Patterns

- Monoliths may face challenges in a cloud environment
  - Session state in application tier (traditional cluster).
  - Writing to the file system as if it were a persistent file system
  - Relying on application server or the OS to provide dependencies for your application to run
  - Long startup times

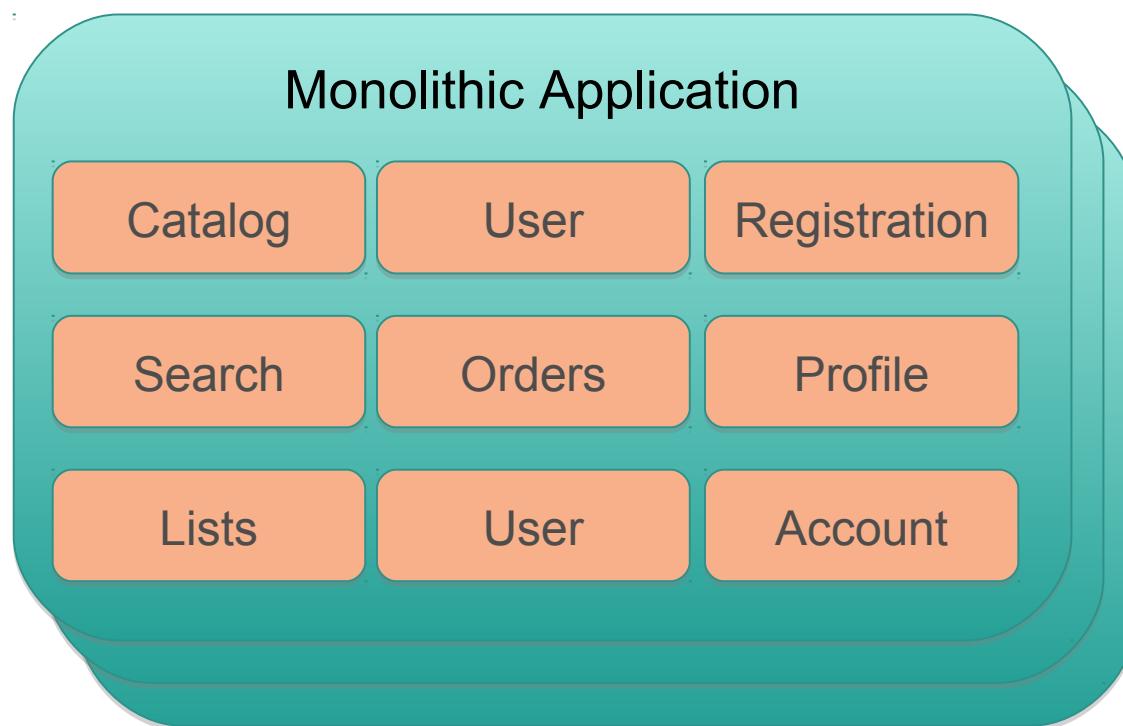
# Monolith Change Cycle

- Monoliths couple change cycles together
  - Upgrade everything at same time



# Monolith Scaling

- Monoliths services can't be scaled independently



# Monolith Challenges

- Too many developers in one code base
- Developers struggle to understand a large codebase
- Long term commitment to the technology stack

# Agenda

- The Monolith
- **Microservices**
- Microservices and Pivotal Cloud Foundry

# Microservices Defined

*Microservices are a loosely coupled Service-Oriented Architecture (SOA) with bounded contexts*

— Adrian Cockcroft, Netflix

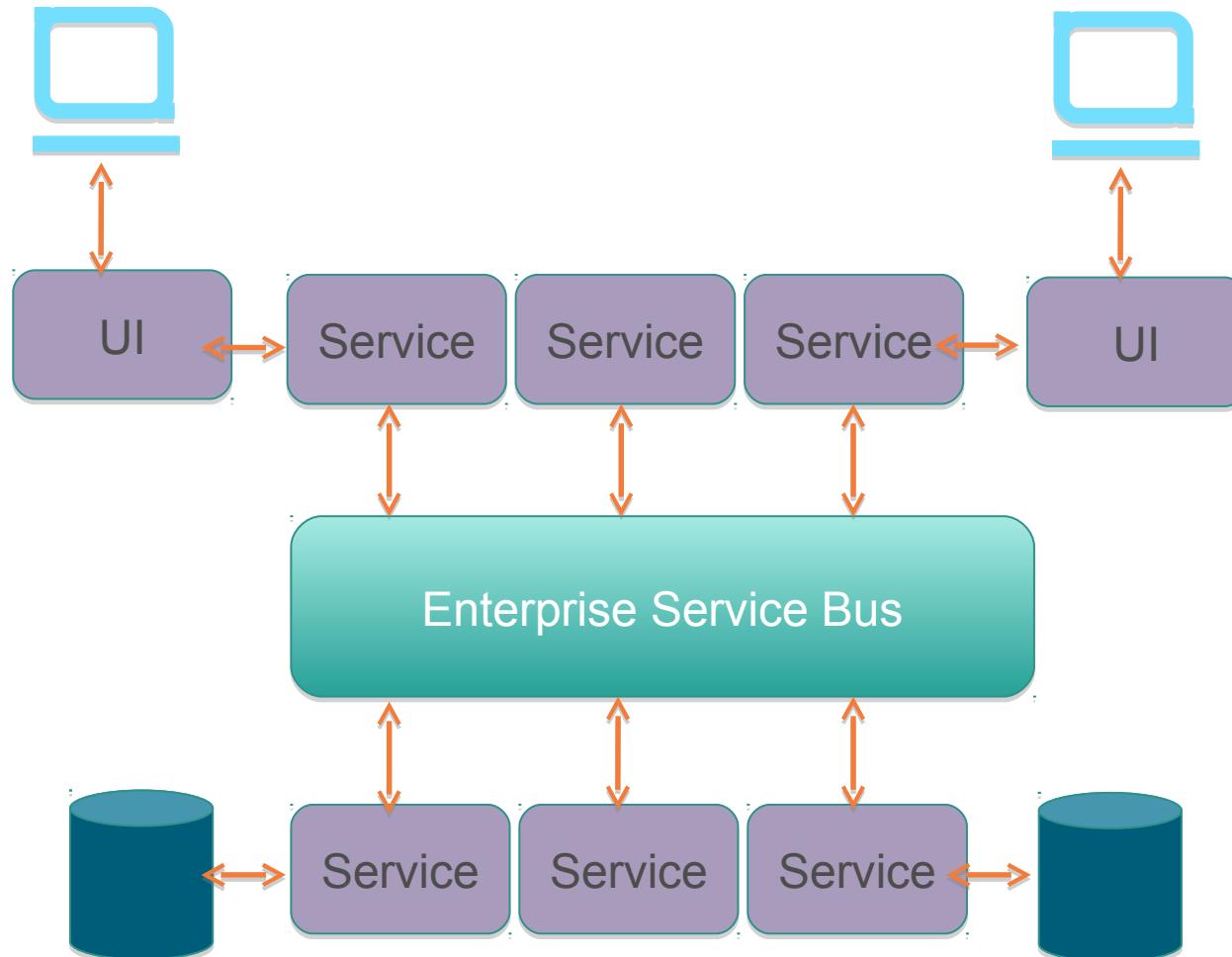
- Compose an application using a set of small services
  - each running as independent processes
  - intercommunicating via open protocols
  - separately written, deployed, scaled & maintained
  - encapsulating business capability

# The Importance of APIs

All teams will henceforth expose  
their data and functionality  
through service interfaces

— Jeff Bezos, Amazon (2002)

# Traditional ESB / SOA



# Orchestration vs Choreography

- Dancers don't need a conductor
  - Distributed, no centralized control



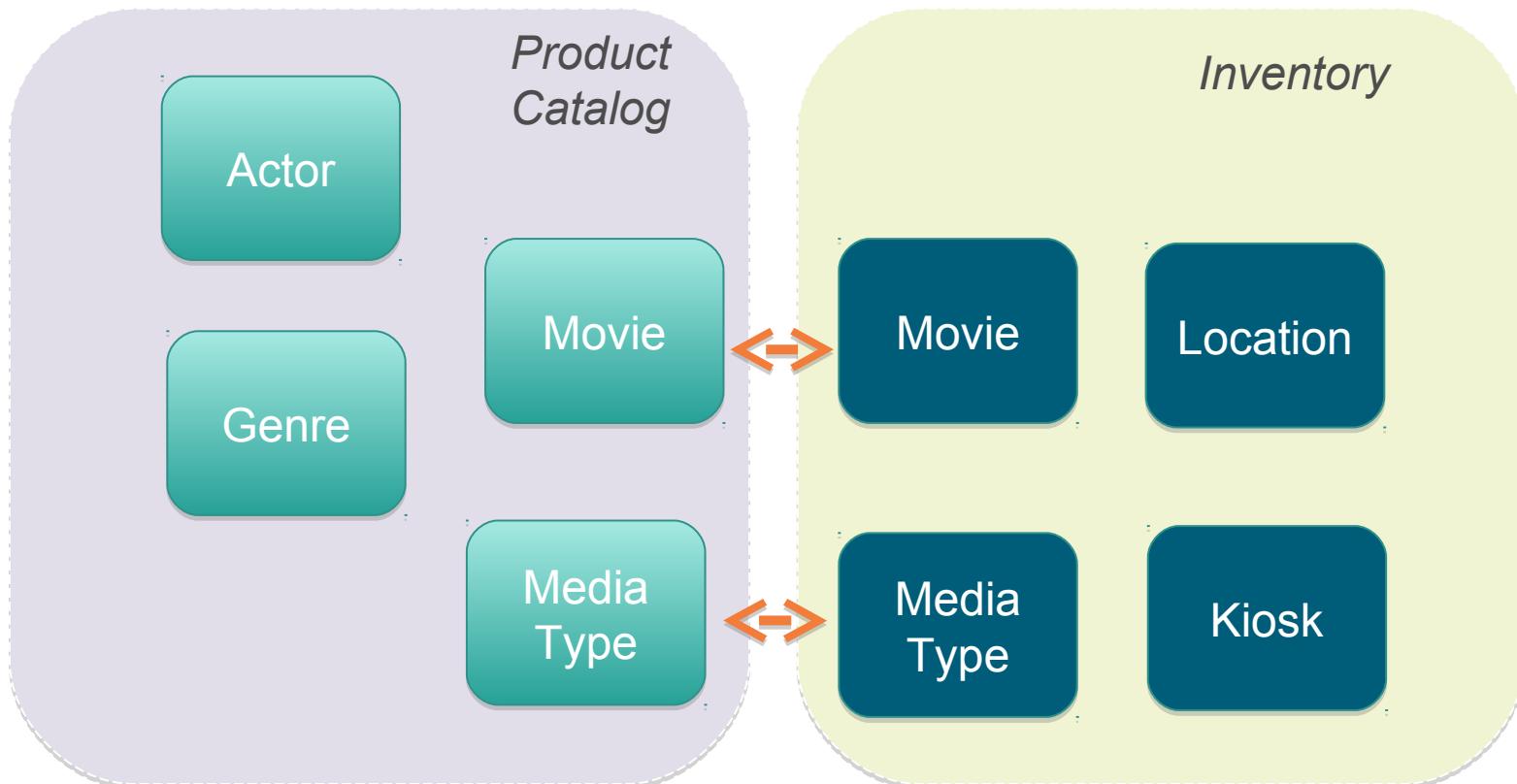
# Unix Pipes and Filters

- A microservice has a single responsibility
- Compare to Unix pipe commands:

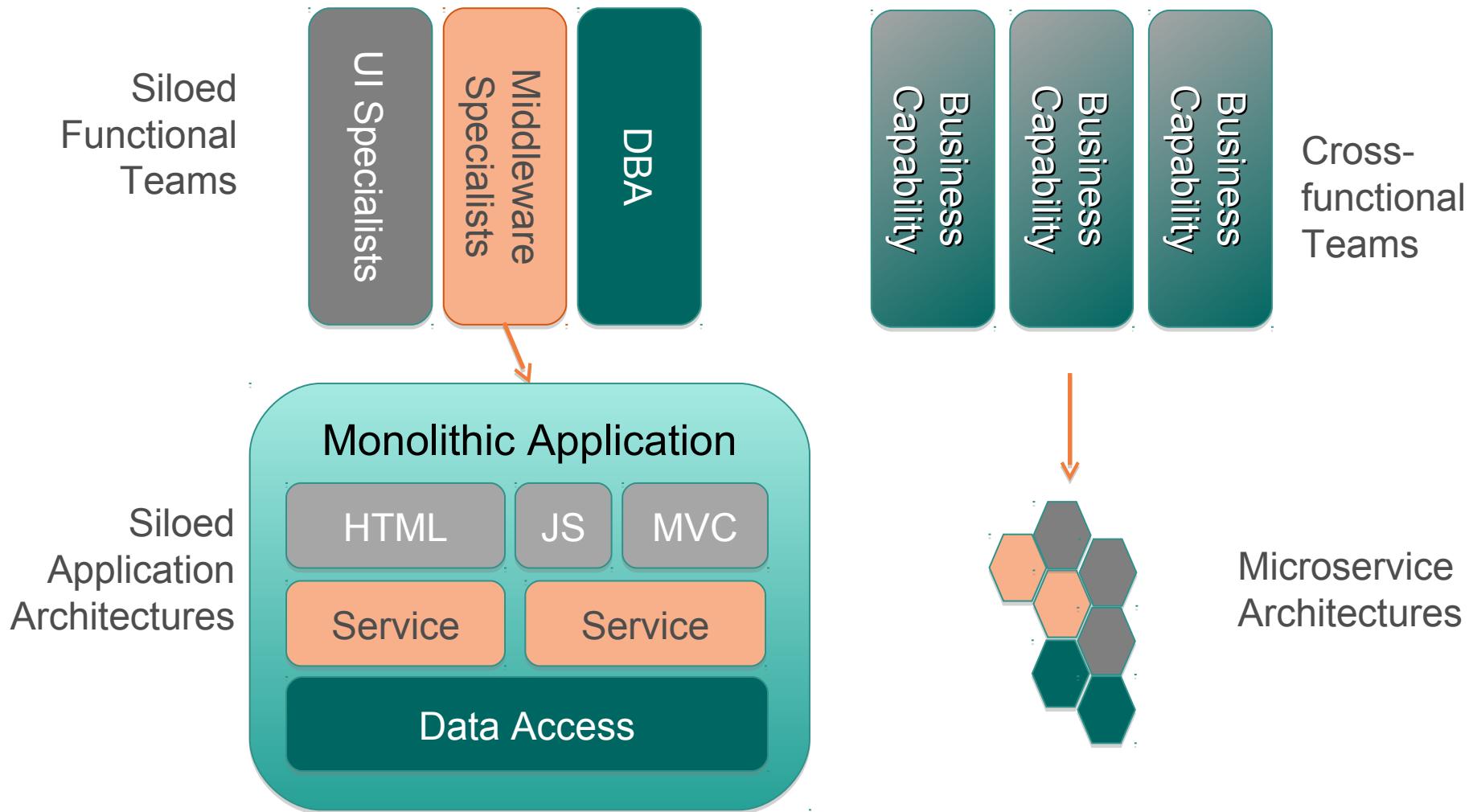
```
cut -d" " -f1 < access.log | sort | uniq -c | sort -rn | less
```

# Bounded Context

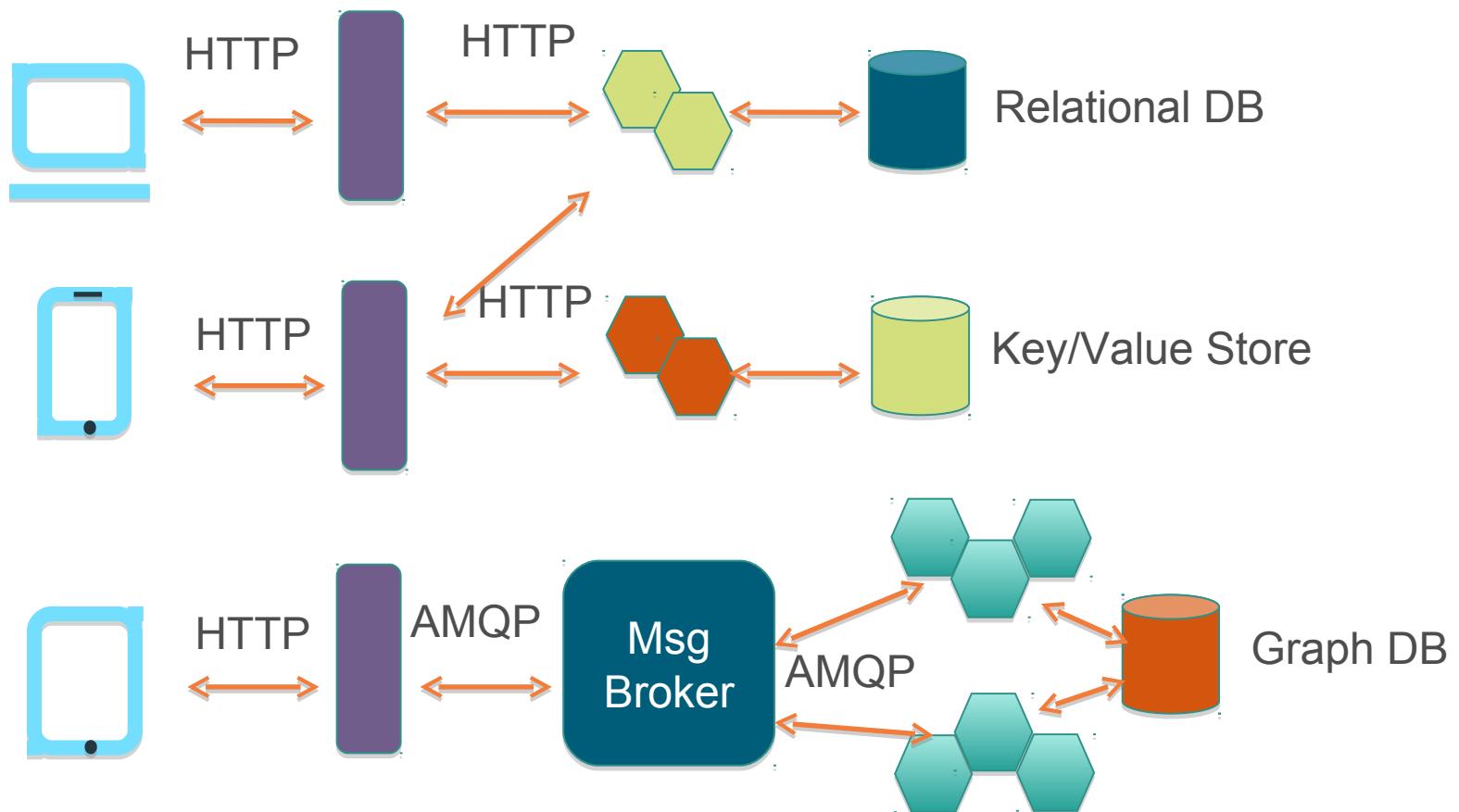
- Microservices use a *bounded context*.



# Organize Around Business Capabilities

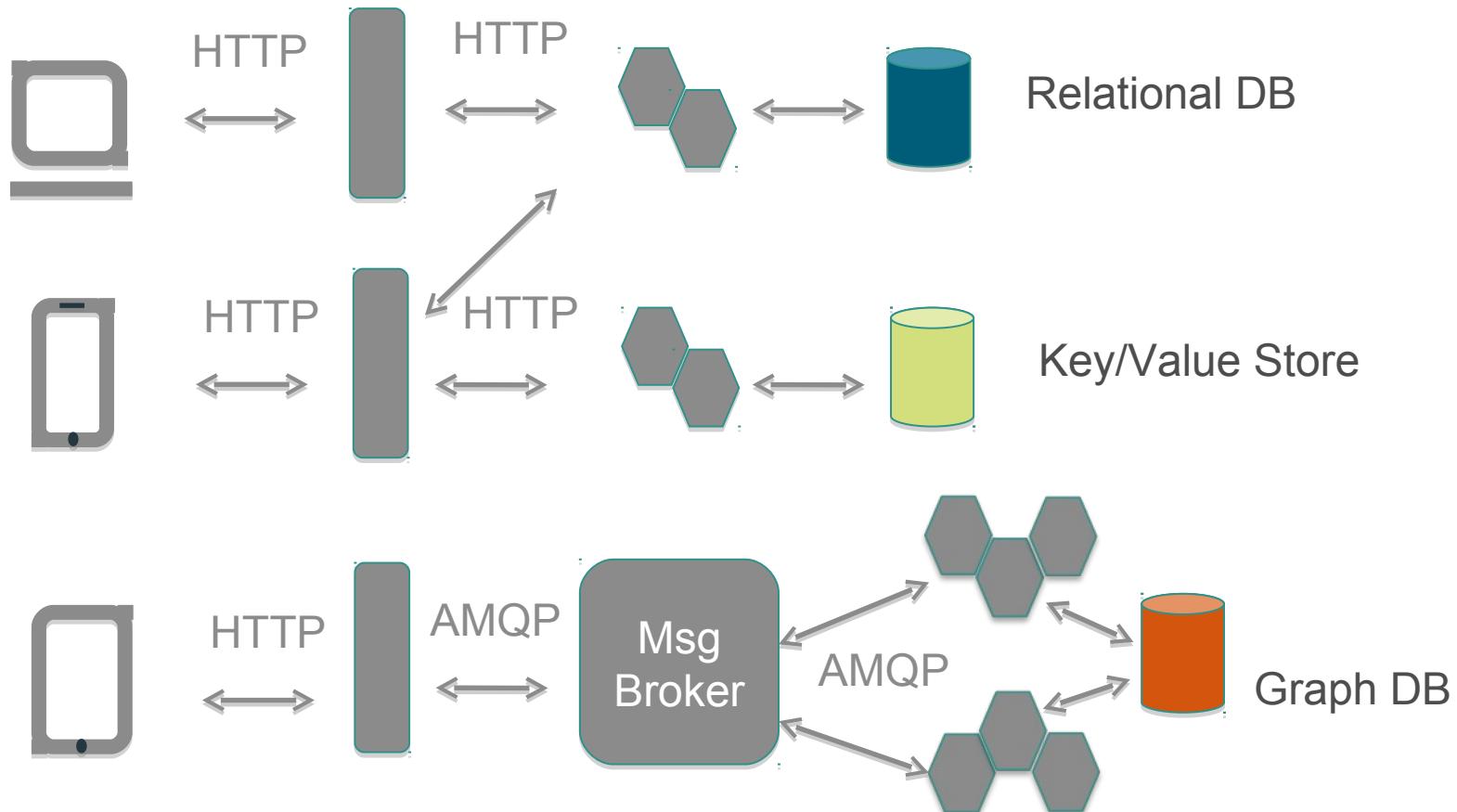


# Microservice Architecture (Simplified)



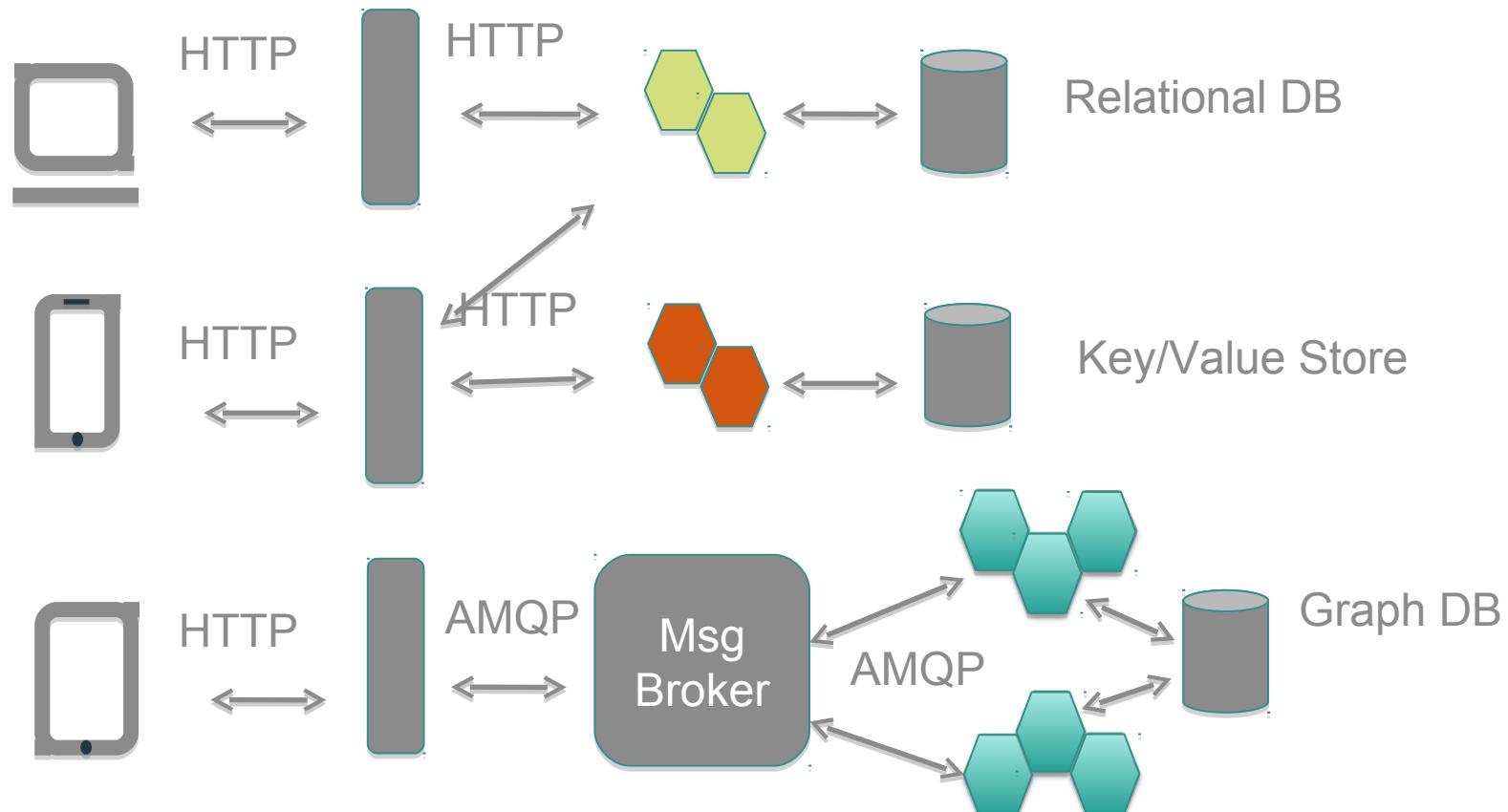
# Polyglot Persistence

- Freedom to pick the persistence solution.



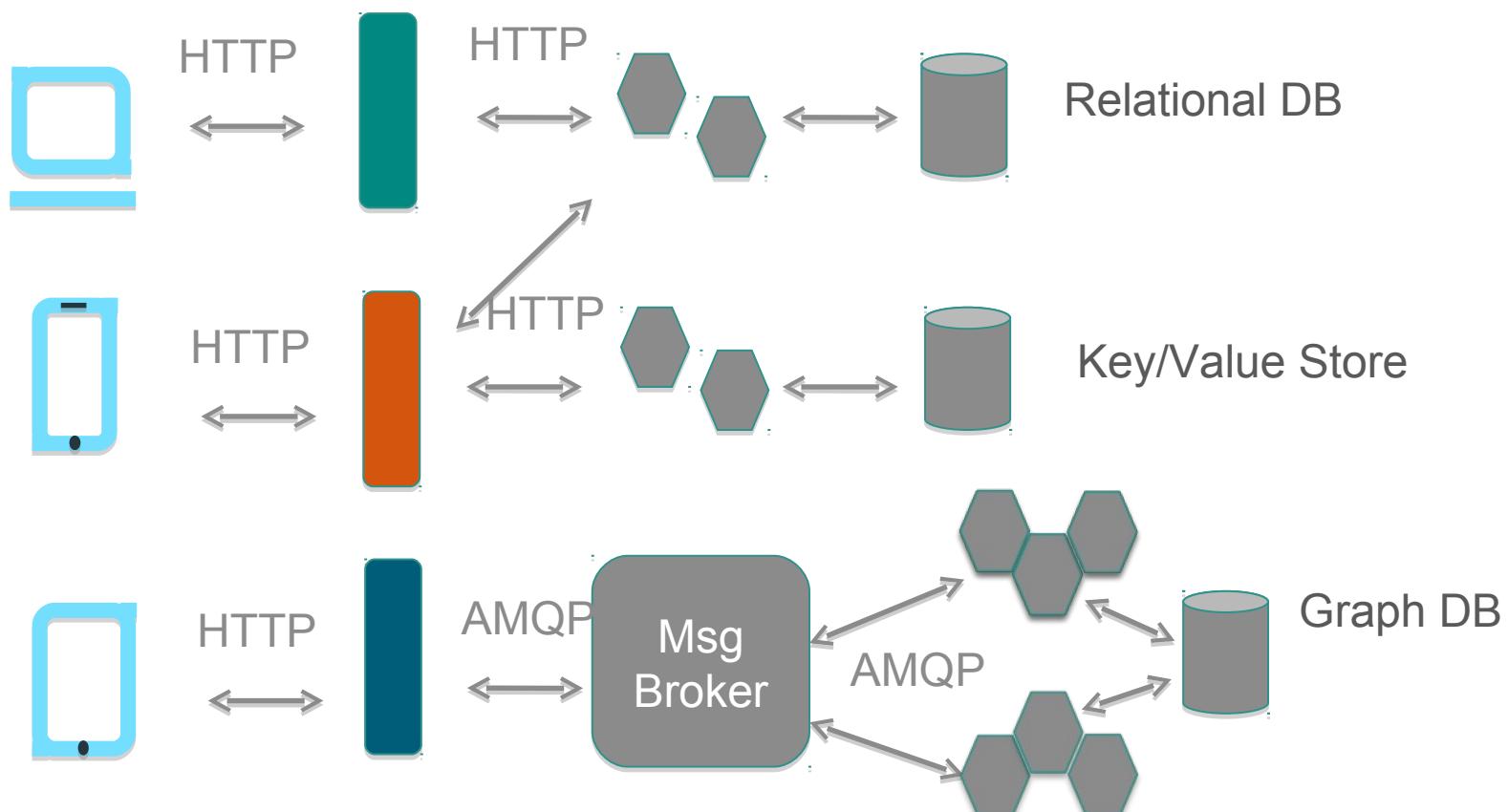
# Polyglot Apps

- Choice of language when developing apps.



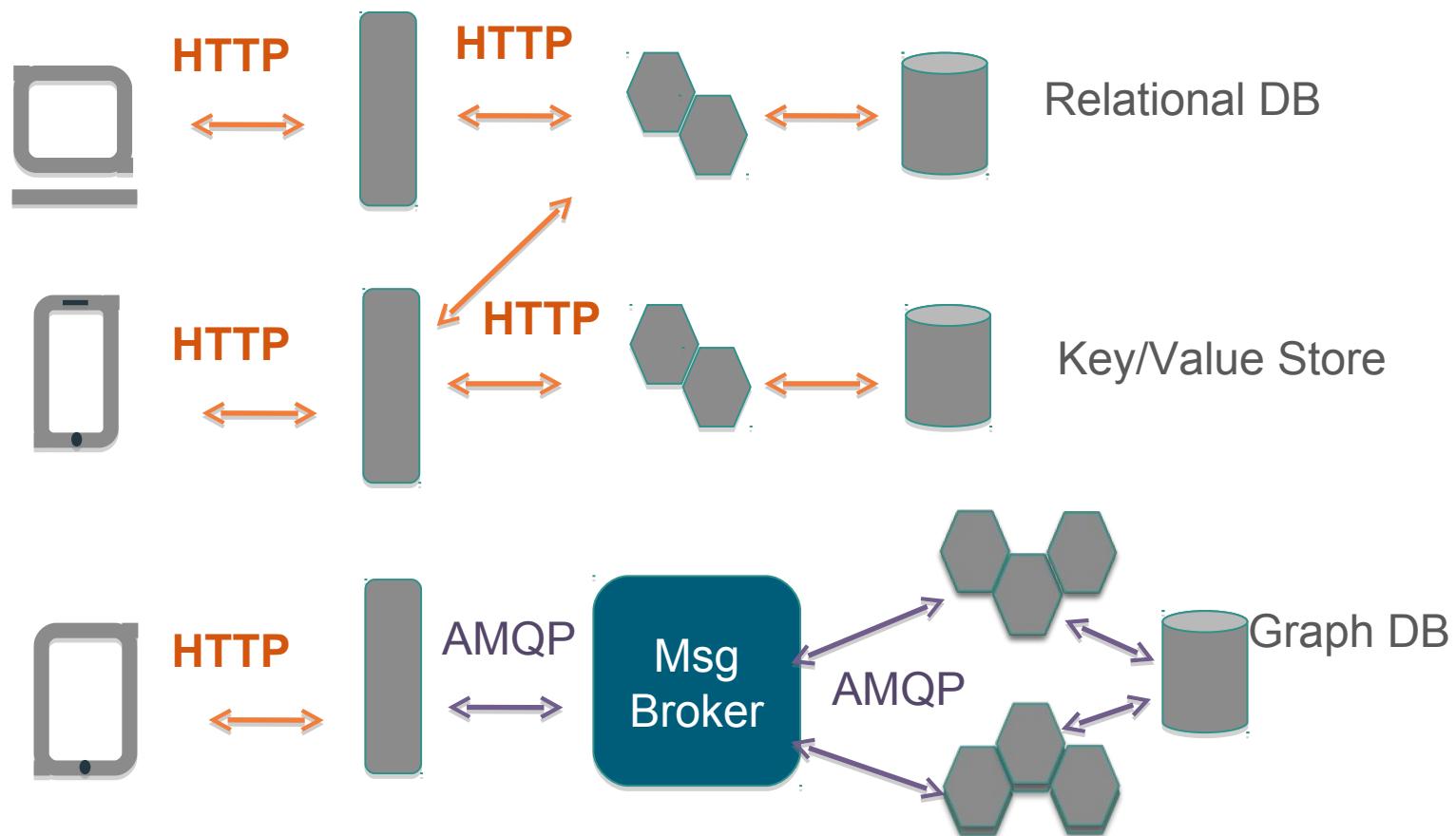
# API Gateway

- Device specific gateways.



# Cloud Protocols

- Use cloud friendly protocols.



# Microservice Benefits

# Microservice Advantages – 1

## Change Cycle



- Change cycles are decoupled
- Enables more frequent deploys

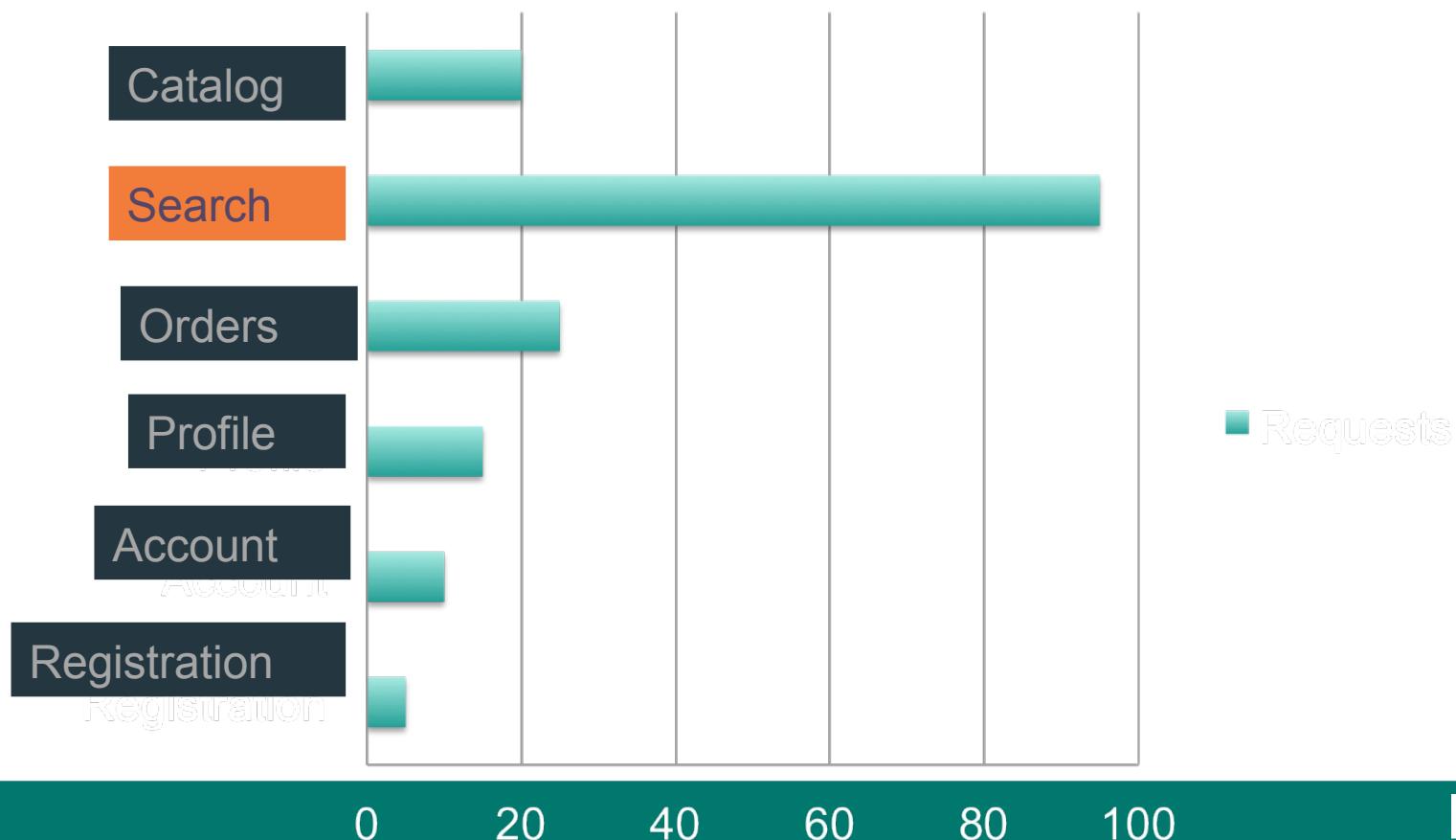
# Microservice Advantages – 2

## Scaling



- Allow for efficient scaling.

Requests Per Minute





# Microservice Advantages – 3

- Developers learn a smaller codebase faster
- Fewer developers in each code base
- Each microservice can use its own Technology stack

# Agenda

- The Monolith
- Microservices
- **Microservices and Pivotal Cloud Foundry**

# Microservice Challenges

- Microservices have their challenges too!
  - Cloud Foundry can help

<http://martinfowler.com/bliki/MicroservicePrerequisites.html>

<http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html>

# Operations Overhead



- Microservices have operations overhead
  - Multiple processes, load-balancing, redundancy, HA
- Agreed, but this is mitigated with PCF

# Operations Overhead

Consider:

- Dynamic Routing
- Scaling
- Monitoring
- Services
- Health Management
- Buildpacks

# Substantial DevOps Skills Required



- Typically requires “cultural” change
  - How applications are developed and deployed
- Substantial *DevOps* skills are required to run Microservices
- Agreed. This is a good thing

# Substantial DevOps Skills Required

Consider:

- Polyglot Persistence via Service Brokers
- Space Parity & Immutable Infrastructure
- Buildpacks
- Health Management

# Development Challenges



- Difficult to achieve strong consistency across services
  - Eventual consistency, Compensating transactions
- Distributed system
  - Harder to debug/trace
  - Greater need for end-to-end testing
  - Expect, test for and handle failure of any process
- Platform can help with *some* of these too

# Development Challenges

Consider:

- Continuous Integration and Deployment
- Performance Management
- Cloud Services

# Summary

- The Monolith
- Microservices
- Microservices and Pivotal Cloud Foundry



# Autoscaling

Using the Autoscaling service

Pivotal

# Roadmap

- **Autoscaling**

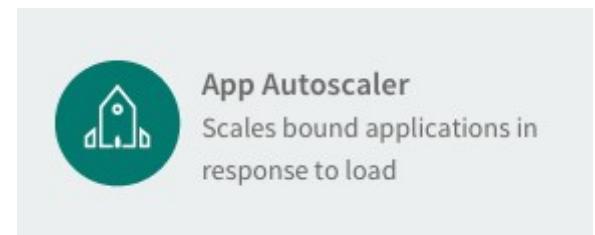
# Scaling Options



- CF allows horizontal scaling
  - Controlling the # of instances of an application running
  - All behind a common router (load balancer)
  - Controllable via the manifest, **cf** command line, or App Manager console
- All options require manual intervention

# AutoScaling

- CF can allow applications to be automatically scaled
  - “AutoScaling”
- System load can be used as a trigger in place of manual interaction.
- Autoscaling Service
  - Must be installed by administrator



# AutoScaling Service – Steps

1) Create the service

2) Bind to Application

1) Set desired scaling parameters

1) Instance Limits

2) Scaling Rules

- CPU Utilization
- HTTP Throughput
- HTTP Latency

3) Scheduled Limits

attendee-service

ENABLED



INSTANCE LIMITS

edit

Currently in use: 1

Minimum: 1

Maximum: 3

SCALING RULES

edit

Metric	Low	High	Status
CPU Utilization	20%	80%	Active

SCHEDULED LIMITS

edit

No Scheduled Limit Changes

EVENT HISTORY

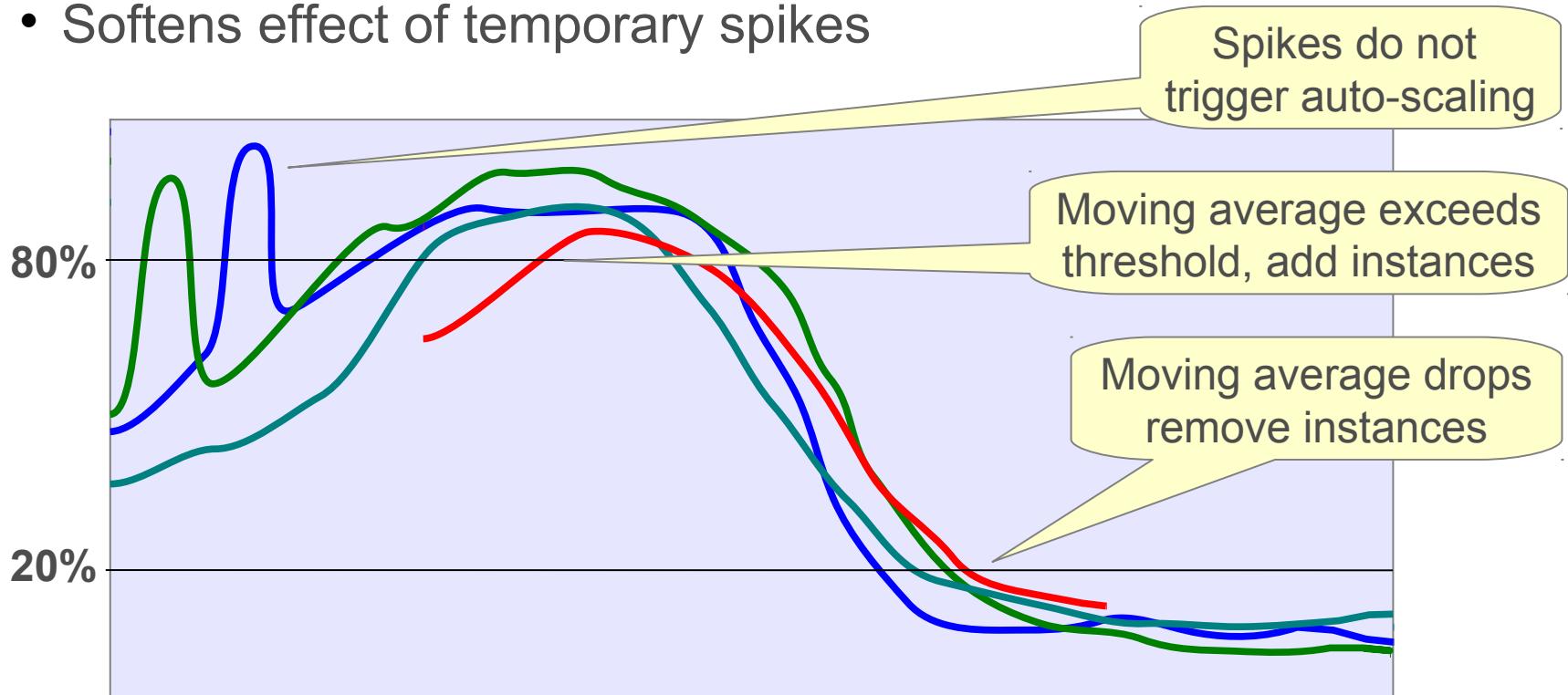
No events recorded.

view more

Pivotal™

# AutoScaling – Moving Average

- Scaling activity based on moving averages
  - Softens effect of temporary spikes



- Manual scaling disables AutoScaling

- Re-enable:

attendee-service

**ENABLED**



# AutoScaling – Scheduling

- Autoscaling events can be scheduled
- Changes autoscaling behavior on the given date / time.
- May be single event or recurring

Scheduled Limit Changes: Attendee-Service Times Shown Are Local X

---

**Configure**

Date	Time	Repeat (optional)	Min	Max
<input type="button" value="Month ▲"/> <input type="button" value="Day ▲"/> <input type="button" value="Year ▲"/>	<input type="button" value="HH ▲"/> : <input type="button" value="MM ▲"/> <input type="button" value="AM ▲"/>	<input type="checkbox"/> Su <input checked="" type="checkbox"/> Mo <input checked="" type="checkbox"/> Tu <input checked="" type="checkbox"/> We <input checked="" type="checkbox"/> Th <input checked="" type="checkbox"/> Fr <input type="checkbox"/> Sa	<input type="text" value="2"/>	<input type="text" value="5"/> <input type="button" value="SAVE"/>

---

**Scheduled** + ADD NEW

Date	Time	Repeat	Min	Max	Edit	Delete
			0	0	<span>✎</span>	<span>✖</span>

# Lab

## Autoscaler

# Application Performance Monitoring

Going beyond logging

Using PCF Metrics and third-party monitoring tools

# Roadmap

- PCF Metrics
- 3<sup>rd</sup> Party APM Tool - New Relic



# Application Performance Monitoring

- Logs and analysis only takes you so far
- Important to have real-time monitoring of applications
  - Uptime, performance, etc.
- Application Performance Monitoring (APM) Tools
  - Monitor your application while running
  - Several choices available in Cloud Foundry
    - New Relic, AppDynamics, Dynatrace ...

# PCF Metrics

- Pivotal Cloud Foundry comes with some metrics information built-in
  - **Container Metrics:** CPU, memory, disk percentages
  - **HTTP Metrics:** requests per second, HTTP errors per second, request latency
  - **App Events:** create, update, start, stop, and crash

# PCF Metrics

- In the Application Console
  - Go to dashboard page for your application

The screenshot shows the PAS Application Console for a node.js application named "node".

**Header:** APP node (status: Running) View App ↗ Buildpack: node.js 1.5.22

**Navigation:** Overview Services Route (1) Env Variables Logs Settings

**Events:** Last Push: 01:08 PM 11/08/16

- Started app pchapman@pivotal.io 11/08/2016 at 07:38:47 AM UTC
- Updated app pchapman@pivotal.io 11/08/2016 at 07:38:03 AM UTC

**Scaling:** Instances: 1, Memory Limit: 128 MB, Disk Limit: 1 GB. Scale App button.

**Status:** View in PCF Metrics ↗

# ▲	STATUS	CPU	MEMORY	DISK	UPTIME
0	● Running	0%	17.89 MB	39.72 MB	1 d 5 hr 32 min

# PCF and New Relic\*\*

- Available as Marketplace Service
  - Install from a PCF Tile
  - PWS offers it too
- What it does:
  - Tracks different instances of application
  - Monitors down to the line of code



**New Relic Service Broker for PCF**

\*\* "New Relic"  
– anagram of the  
name of founder  
*Lew Cirne*

# New Relic Tile in Ops Manager

The screenshot shows the PCF Ops Manager interface. At the top, there are tabs for APM, BROWSER, MOBILE, PLATFORM, SERVERS, and INSIGHTS. Below the tabs, the main area is titled "Available Products". On the left, a list of products is shown with their versions and upgrade status:

- Ops Manager Director v1.4.0.0 No upgrades available
- Pivotal Service Broker Alpha v-0.0.2 No upgrades available
- NewRelic Service Broker Alpha v-0.0.2 No upgrades available
- AppDirectress v1.0.0-alpha Experimental No upgrades available
- Custom Autoscaler Experimental No upgrades available
- MySQL for Pivotal Cloud Foundry No upgrades available

On the right, the "Installation Dashboard" is displayed. It features three tiles:

- "Ops Manager Director for VMware vSphere" v1.4.0.0
- "Pivotal Elastic Runtime" v1.4.0.0-alpha.629.2dd8c81
- "New Relic" v1.3.0

A blue arrow points from the "Import a Product" button at the bottom left to the "New Relic" tile. Another blue arrow points from the "New Relic" tile towards the "Apply changes" button on the right.

# New Relic in PWS Marketplace

Pivotal Web Services

pchapman@pivotal.io ▾

ORG  
Pivotal-Education... ▾  
SPACES  
dgadiraju@gmail.com  
matthias.eschhold@alli...

Marketplace  
Docs  
Support  
Tools  
Blog  
Status

## Marketplace

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

Search the Marketplace

Services ▾

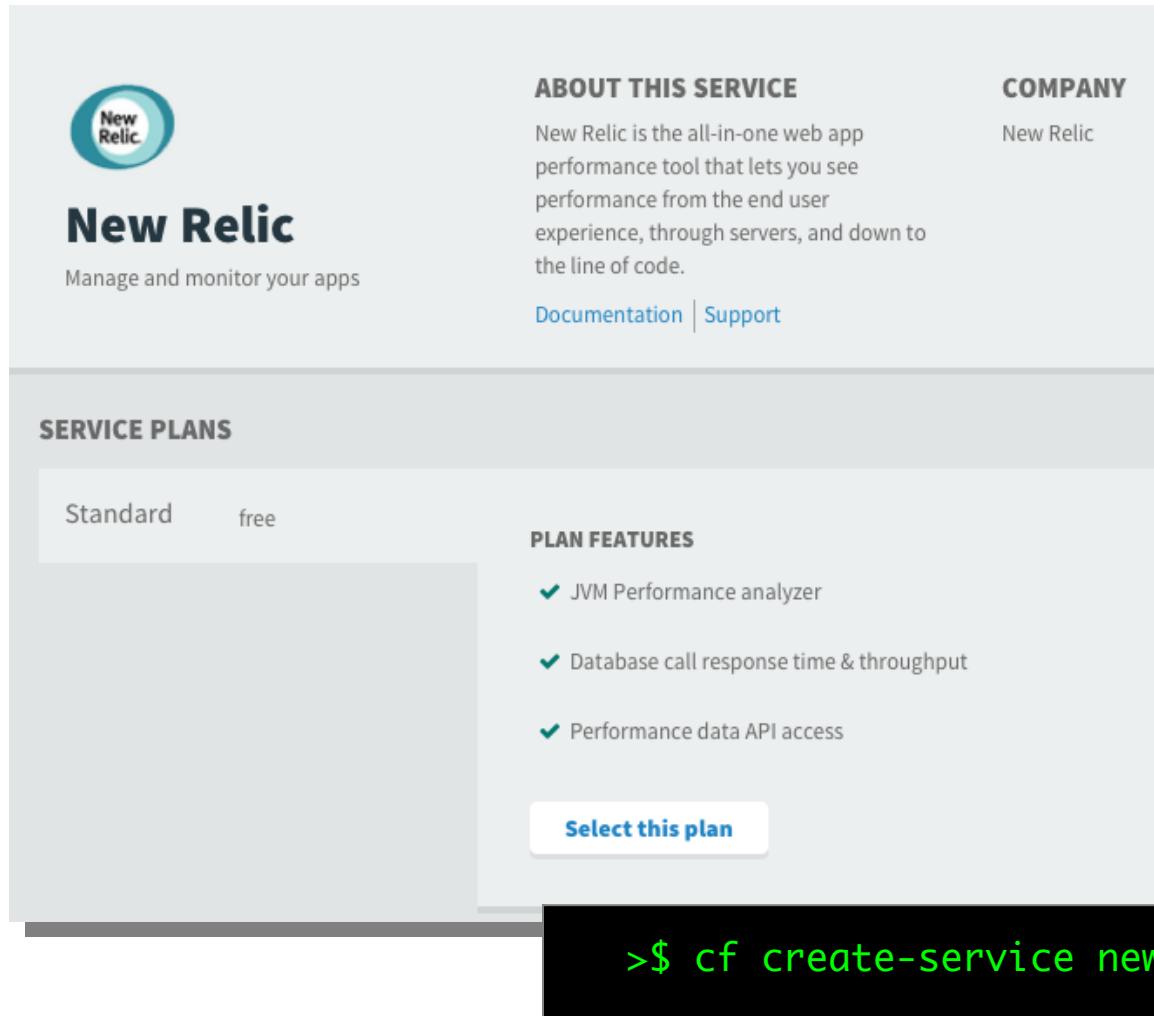
-  **3scale API Management**  
API Management Platform >
-  **App Autoscaler**  
Scales bound applications in response to load (beta) >
-  **BlazeMeter**  
Performance Testing Platform >
-  **MySQL for Pivotal Cloud Foundry**  
MySQL databases on demand >
-  **New Relic**  
Manage and monitor your apps > 
-  **Pivotal SSL**  
Upload your SSL certificate for your app(s) on your custom domain >

Pivotal™

# PCF and New Relic – Usage

- How To Use:
  - Create New Relic service in desired space
  - Bind to desired Application(s)
  - Increase app memory if needed (extra footprint)
  - Re-stage application
    - *Java Buildpack includes New Relic Agent, others may not*
  - APM available as a link from within PWS

# Creating the New Relic Service



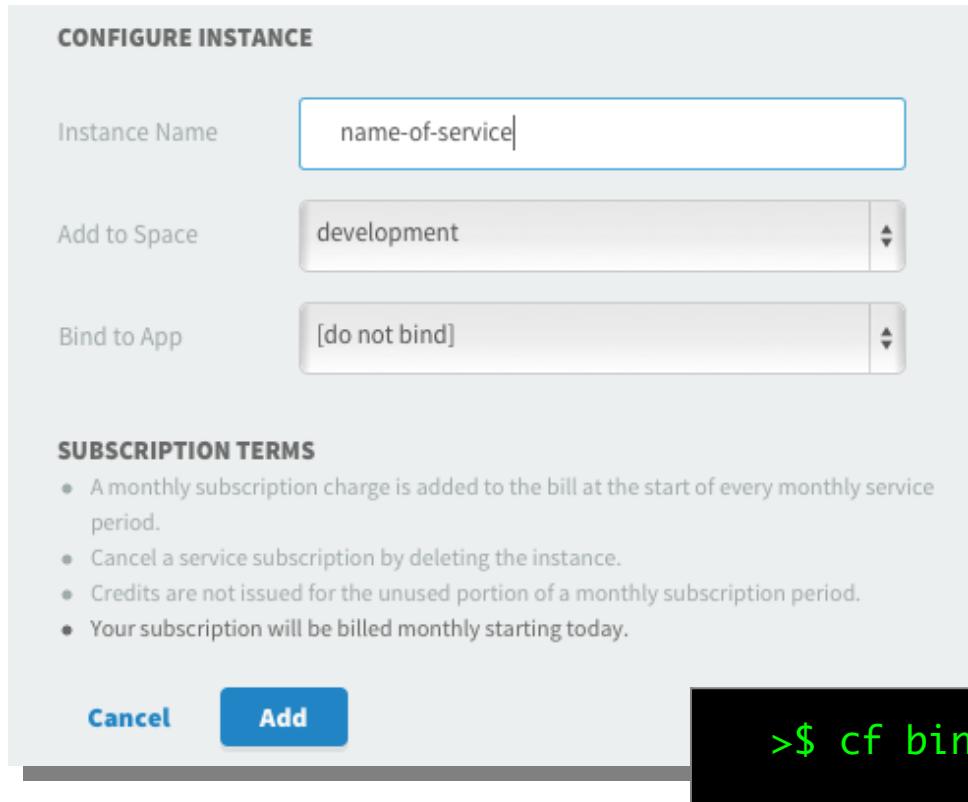
The screenshot shows the New Relic service creation interface. At the top left is the New Relic logo and the text "New Relic". Below it is the subtext "Manage and monitor your apps". To the right is the "ABOUT THIS SERVICE" section, which describes New Relic as an all-in-one web app performance tool. It includes links to "Documentation" and "Support". Further right is the "COMPANY" section, which lists "New Relic". Below these sections is the "SERVICE PLANS" section. It shows a single plan: "Standard" with "free" next to it. Underneath the plan is the "PLAN FEATURES" section, which lists three items with checkmarks: "JVM Performance analyzer", "Database call response time & throughput", and "Performance data API access". A blue button labeled "Select this plan" is located at the bottom of this section. At the very bottom of the interface is a black bar containing the green text ">\$ cf create-service newrelic standard apt".

- Use App Manager Console

- Use cf CLI

# Create Service / Bind Application

- Use cf CLI or App Manager Console:

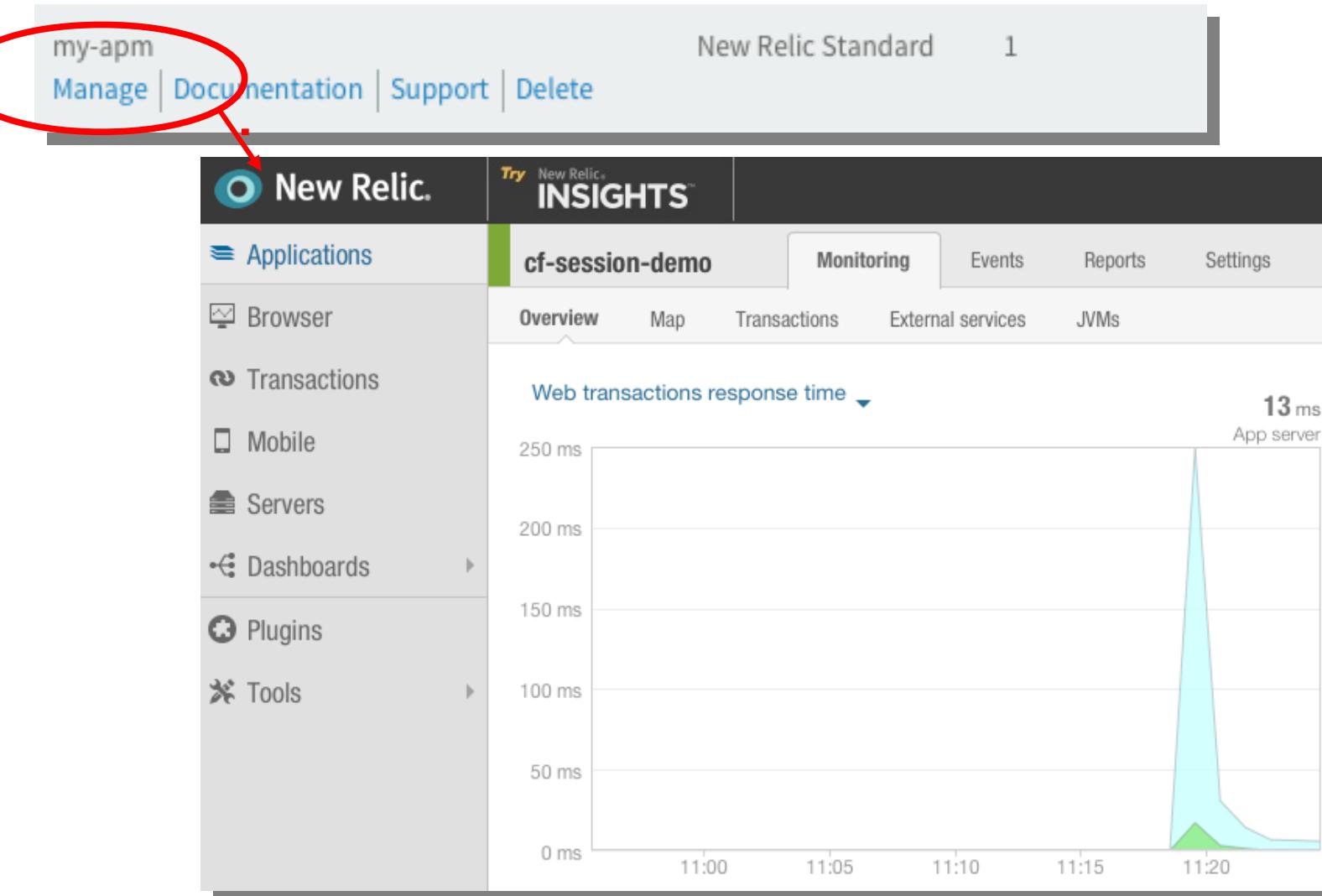


The screenshot shows the 'CONFIGURE INSTANCE' dialog. It has three input fields: 'Instance Name' containing 'name-of-service', 'Add to Space' containing 'development', and 'Bind to App' containing '[do not bind]'. Below these is a section titled 'SUBSCRIPTION TERMS' with the following bullet points:

- 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.

At the bottom are 'Cancel' and 'Add' buttons. To the right of the dialog is a black bar containing the command: `>$ cf bind-service some-app apt`.

# Access via *Manage* Link in App Manager



The screenshot shows the New Relic Insights interface. In the top left, the application name "my-apm" is displayed, with a red circle highlighting it and a red arrow pointing from the "Manage" link in the top navigation bar down to the application name. The top navigation bar also includes links for "Documentation", "Support", and "Delete". The main dashboard shows the "cf-session-demo" application selected. The monitoring section displays a chart of "Web transactions response time" over time, with a single sharp spike reaching up to 250 ms at approximately 11:18. The chart is labeled "App server" for the source of the transaction.

# Summary

- PCF metrics provides basic monitoring.
- Utilize 3<sup>rd</sup>-party APM tool for full monitoring capability.

# Two Labs

1. Application Performance Monitoring
2. Metrics (optional)



CLOUD **FOUNDRY**

# Introduction to BuildPacks

Deploying applications written in various languages

Java, .NET, Ruby, Groovy, JavaScript ... + Cloud

Pivotal

# Overview

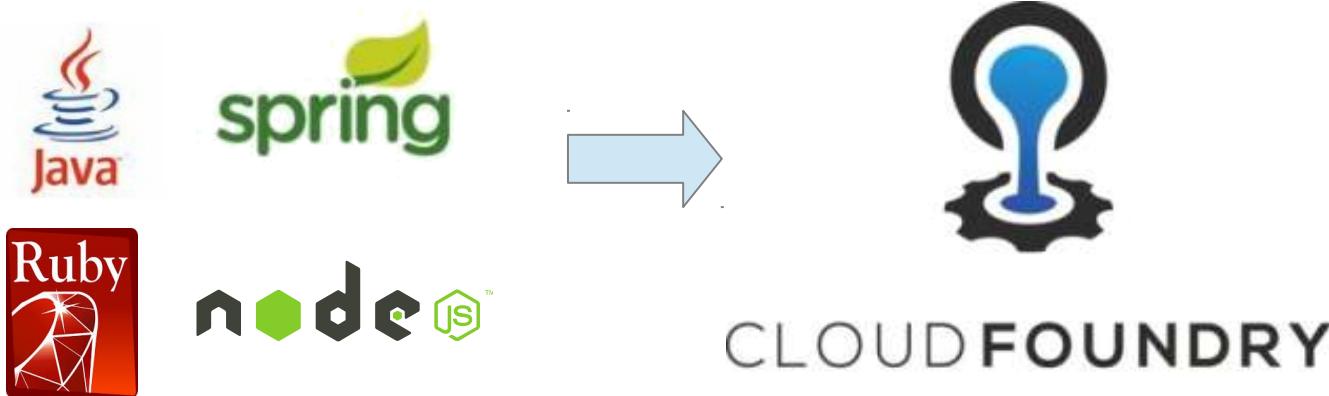
- After completing this lesson, you should be able to:
  - Explain what a buildpack is
  - Deploy using a buildpack
  - Describe the basic Buildpack API
  - Describe the behavior of the Java Buildpack
  - Configure / extend a Buildpack
  - Use Docker image in PCF

# Roadmap

- **What are Buildpacks?**
- Deploying to Cloud Foundry
- Using Buildpacks
- Buildpack API
- Java Buildpack
- Customization without Forking
- Configure / Extend Java Buildpack
- Using Docker in PCF

# The Question:

- Applications can be written in many languages / frameworks:



- ...and yet each type can run in Cloud Foundry
- How is this possible?

# Configuring a Server from Scratch

- If you were configuring a new server to run an application, what would you include / install?
  - Operating system
  - Runtimes for your software (Java, Ruby, Python, etc.)
  - Containers as needed (e.g. Tomcat for Java, Apache HTTPD for PHP)
  - Frameworks as needed (APM tools)
  - Application binaries
- Good idea to write a *script* for this

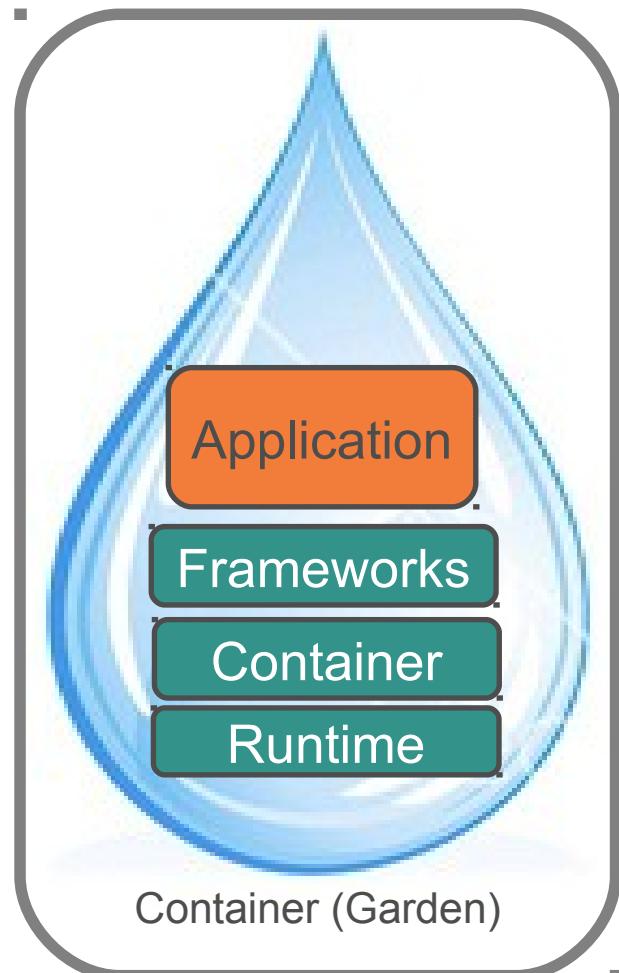


# Buildpack Does the Same Thing

Except the goal is to run on Cloud Foundry

**Buildpack** – a combination of *scripts* that assembles runtimes, containers, frameworks, and your application into a *droplet*

**Droplets** – run inside PCF Containers  
● Which run inside Cells (VMs)



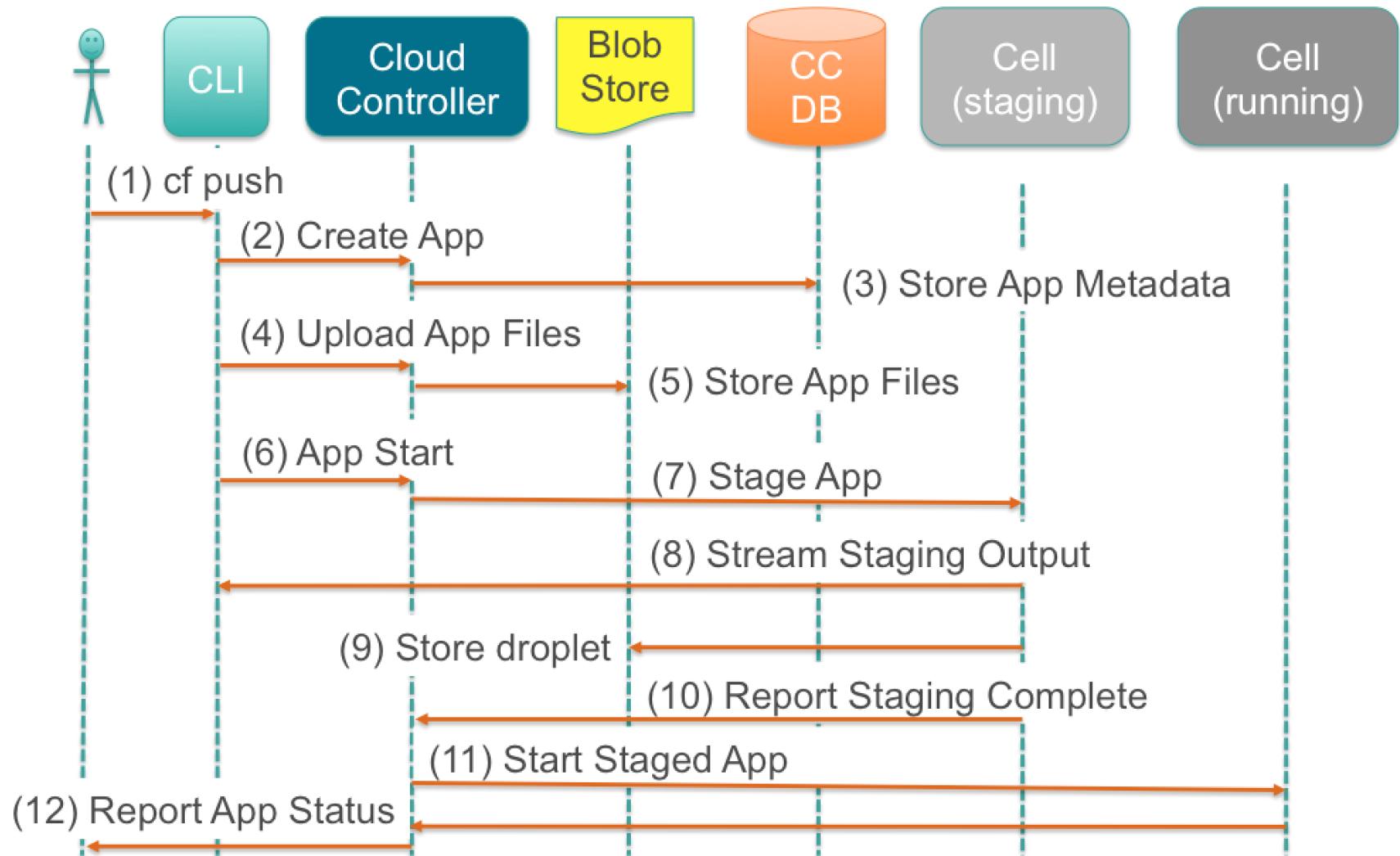
# Buildpacks are Not...

- Buildpacks...
  - Are *not* a special build process for your application
    - Buildpacks build *droplets*
  - Do *not* run on your local machine
    - Buildpacks run on CF during the staging process

# Roadmap

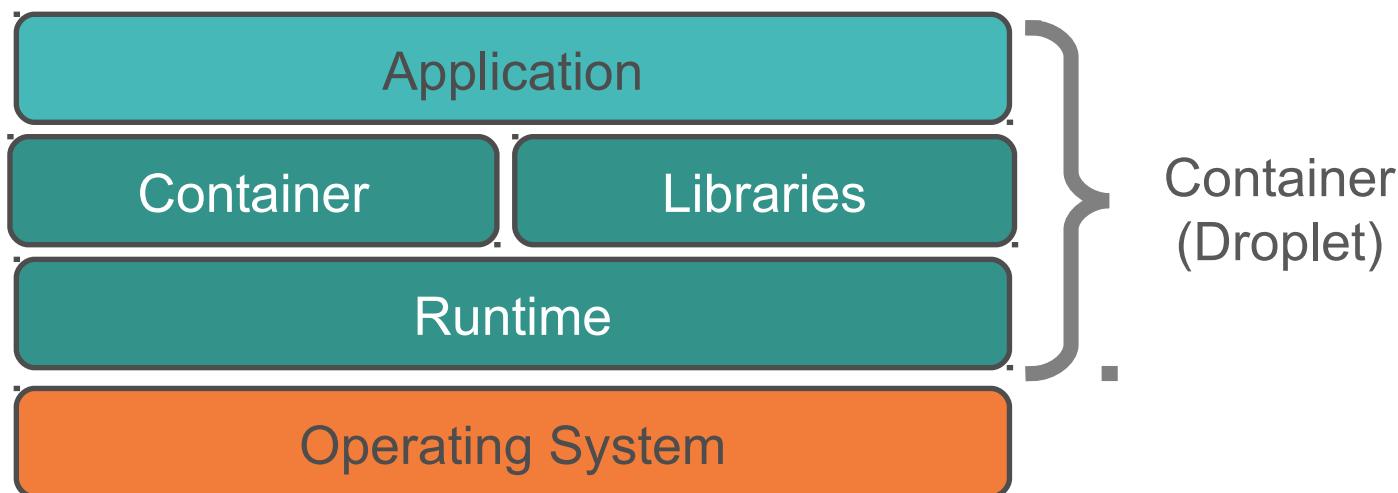
- What are Buildpacks?
- **Deploying to Cloud Foundry**
- Using Buildpacks
- Buildpack API
- Java Buildpack
- Customization without Forking
- Configure / Extend Java Buildpack
- Using Docker in PCF

# Deploying to CF



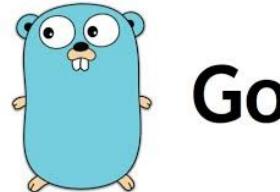
# Staging and Buildpacks

- Build packs are responsible for preparing the machine image for an application

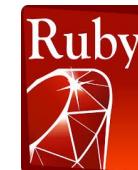


# Available Buildpacks

- Buildpacks are either
  - Installed into a cloud foundry instance or
  - Loaded from an external location at push time
- Buildpacks provided by public Cloud Foundry
  - Note: This list expands over time!



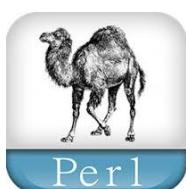
*static*



Pivotal

# Custom Buildpacks

- CF Community provides buildpacks for other languages
- Or write your own
  - Usually by forking / adapting an existing buildpack
- For list of CF Community Buildpacks
  - <https://github.com/cloudfoundry-community/cf-docs-contrib/wiki/Buildpacks>



Pivotal

# Compatibility

- Buildpacks can be compatible with multiple PaaS offerings
- CF buildpacks follow the Heroku buildpack design
  - CF and Heroku buildpacks are compatible (if you care to make them compatible)
  - Other PaaS offerings adopting the buildpack design



# Roadmap

- What are Buildpacks?
- Deploying to Cloud Foundry
- **Using Buildpacks**
- Buildpack API
- Java Buildpack
- Customization without Forking
- Configure / Extend Java Buildpack
- Using Docker in PCF

# Built-In Buildpacks

- Use **cf buildpacks** to determine installed buildpacks

```
> cf buildpacks  
Getting buildpacks...
```

buildpack	position	enabled	locked	filename
staticfile_buildpack	1	true	true	staticfile_buildpack-cached-v1.4.7.zip
java_buildpack	2	true	false	java-buildpack-offline-v3.15.zip
ruby_buildpack	3	true	true	ruby_buildpack-cached-v1.6.40.zip
nodejs_buildpack	4	true	false	nodejs_buildpack-cached-v1.5.35.zip
go_buildpack	5	true	true	go_buildpack-cached-v1.8.4.zip
python_buildpack	6	true	false	python_buildpack-cached-v1.5.18.zip
php_buildpack	7	true	false	php_buildpack-cached-v4.3.33.zip
dotnet_core_buildpack	8	true	true	dotnet-core_buildpack-cached-v1.0.19.zip
dotnet_core_buildpack_beta	9	true	false	dotnet-core_buildpack-cached-v1.0.0.zip
binary_buildpack	10	true	true	binary_buildpack-cached-v1.0.13.zip

# Managing Built-In Buildpacks

```
$> cf create-buildpack <name> <path> <order>
```

- <path> – local directory / zip file / URL / URL to zip file
- <order> – relative order in buildpack list
- **--enable** / **--disable**

- Commands for update, delete, rename available
- Administrator permissions required

# Automatic Detection / Explicit Reference

**\$> cf push**

- Application checked against pre-defined buildpacks
- Matching buildpack invoked automatically

**\$> cf push -b <buildpack-name>**

- Desired buildpack specified (installed buildpack)

**\$> cf push -b <url>**

- The desired buildpack is referenced by a Git URL
  - Note: “disable custom buildpacks” disables this option

# Specify within manifest

- Use buildpack element
  - Specify name or URL

```
---
applications:
- name: cf-my-app
  host: cf-my-app
  domain: cfapps.io
  path: target/my-war.war
  buildpack: https://github.com/cloudfoundry/java-buildpack
```

- Remember precedence
  - Options specified in push command *override* manifest

# Roadmap

- What are Buildpacks?
- Deploying to Cloud Foundry
- Using Buildpacks
- **Buildpack API**
- Java Buildpack
- Customization without Forking
- Configure / Extend Java Buildpack
- Using Docker in PCF

# Buildpack API

- Three scripts, typically written in Bash script or Ruby
  - **/bin/detect app\_directory**
    - Inspect application to see if the buildpack should be applied
  - **/bin/compile app\_directory cache\_directory**
    - Creates Droplet by combining application with runtime, container, packages, libraries (
    - Downloading them if necessary, kept in cache thereafter
  - **/bin/release app\_directory**
    - Build application start command

**NOTE:** *Assemble* or *Pack* would be a better name than *Compile*  
**No** code compilation is happening

# /bin/detect

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

 <b>Ruby</b> <i>A Programmer's Best Friend</i>	<b>Gemfile</b> exists?
 <b>node.js</b>	<b>package.json</b> exists?
 <b>python</b> <sup>TM</sup>	<b>setup.py</b> exists?

# **bin/detect**

```
$ cf push
```

Staging task iterates over all system buildpacks calling:

**bin/detect** scripts

until one returns exit code 0

```
$ cf push -b <url|name>
```

**bin/detect**

is *not* called

# /bin/compile

- Actually 'Builds' Container (Droplet)
- Provides any of the following if needed

Runtime	Java VM, Ruby Interpreter, JavaScript Interpreter, .NET ...
App Server	Apache HTTPD, Tomcat, Nginx, WEBrick, Rails, NodeJS ...
Support Libraries	Ruby gems, NPM packages, Java Jars, APM agents ...

- Resulting container (droplet) “uploaded” to Blobstore

# /bin/compile caching

- Buildpack needs many dependencies
  - Runtime, container, and support packages
  - Often downloaded from sources external to Cloud Foundry
    - Depending on the buildpack
    - Pivotal uses Amazon S3 by default
- Cloud Foundry provides a cache
  - Stores all downloaded artifacts
  - Reduces downloads
  - Subsequent staging operations faster

# /bin/release

- Builds a YAML-formatted hash with three possible keys
- On Cloud Foundry (currently) only the *web:* value is used to get the start command for the app

```
config_vars:  
  name: value  
default_process_types:  
  web: <start command>
```

# Roadmap

- What are Buildpacks?
- Deploying to Cloud Foundry
- Using Buildpacks
- Buildpack API
- **Java Buildpack**
- Customization without Forking
- Configure / Extend Java Buildpack

# Example: Java Buildpack

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



# Java Buildpack Concepts

## Containers

How an application is run

## Frameworks

Additional application transformations

## JREs Java Runtimes

# Java Buildpack Concepts

## Containers

Executable JARs, Groovy, Play,  
Servlet 2 & 3, Spring Boot CLI

## Frameworks

AppDynamics, New Relic,  
Spring Auto-reconfiguration

## JREs

OpenJDK, Oracle JDK

# Container Detection Criteria

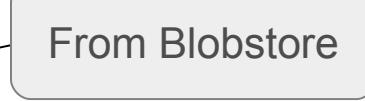
Language	Criteria
Java <code>main()</code>	<code>META-INF/MANIFEST.MF</code> exists with <code>Main-class</code> attribute set
Tomcat	<code>WEB-INF</code> directory exists
Groovy	<code>.groovy</code> file with a <code>main()</code> method, or <code>.groovy</code> file with no classes, or <code>.groovy</code> file with a shebang ( <code>#!</code> ) declaration
Spring Boot CLI	one or more POGO <code>.groovy</code> files with no <code>main()</code> method, and no <code>WEB-INF</code> directory
Play	<code>start</code> and <code>lib/play.play_*.jar</code> exist

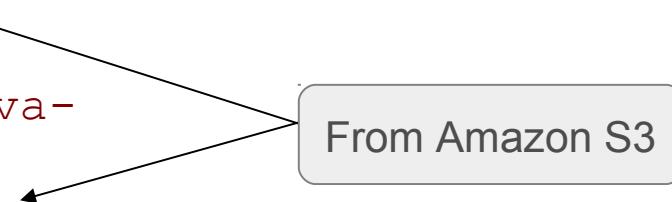
# Framework Detection Criteria

Framework	Criteria
App Dynamics	App Dynamics service bound to app
New Relic	New Relic service bound to app
Spring AutoConfiguration	spring-core*.jar in the application directory

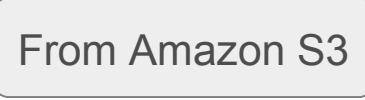
# /bin/compile Output Example

- Output example:

-----> Downloaded app package (11M) 

-----> Downloading Open Jdk JRE 1.8.0\_20 from  
[http://download.run.pivotal.io/openjdk/lucid/x86\\_64/openjdk-1.8.0\\_20.tar.gz](http://download.run.pivotal.io/openjdk/lucid/x86_64/openjdk-1.8.0_20.tar.gz) (1.0s) 

Expanding Open Jdk JRE to .java-buildpack/open\_jdk\_jre (1.1s)

-----> Downloading Tomcat Instance 7.0.53 from  
<http://download.run.pivotal.io/tomcat/tomcat-7.0.53.tar.gz> (0.5s) 

Expanding Tomcat to .java-buildpack/tomcat (0.1s)

-----> Uploading droplet (49M) 

# Roadmap

- What are Buildpacks?
- Deploying to Cloud Foundry
- Using Buildpacks
- Buildpack API
- Java Buildpack
- **Customization without Forking**
- Configure / Extend Java Buildpack

# Customization without Forking

- Simple customization of properties can be done without forking the buildpack
  - Set environment variables instead
    - Either using `cf set-env` or in the `env:` section of manifest
- Two options:
  - `JAVA_OPTS` variable
  - `JBP_CONFIG` variables

# Change JVM Runtime Options – I

- The `JAVA_OPTS` variable is recognized when app runs:

```
$> cf set-env spring-music JAVA_OPTS -showversion
Setting env variable JAVA_OPTS -showversion spring-music myorg
development jlee@pivotal.io
OK
TIP: Use 'cf restage' to ensure your env variable changes take effect

$> cf restage spring-music
... usual push output ...
2015-04-10T16:45:11.88 [App/0] ERR openjdk version "1.8.0_40-"
2015-04-10T16:45:11.88 [App/0] ERR OpenJDK Runtime Environment (build
1.8.0_40--vagrant_2015_03_26_09_03-b25)
2015-04-10T16:45:11.88 [App/0] ERR OpenJDK 64-Bit Server VM (build
25.40-b25, mixed mode)
...
$>
```

# Change JVM Runtime Options – II

- Most JVM options can be specified this way
  - Except some that govern memory sizing
    - Such as **-Xms**, **-Xmx**, **-Xss**, **-XX:MaxPermSize**,  
**-XX:MaxMetaspaceSize**, **-XX:MetaspaceSize**,  
**-XX:PermSize**
    - Most other **-XX** options can be used
    - For full details see:

[https://github.com/cloudfoundry/java-buildpack/blob/master/docs/framework-java\\_opts.md](https://github.com/cloudfoundry/java-buildpack/blob/master/docs/framework-java_opts.md)

# JBP\_CONFIG variables

- Use environment variable to override a buildpack configuration file
  - Naming convention used:
    - `my_file.yml` → `JBP_CONFIG_MY_FILE`
  - Variable must be set to valid *inline* YAML syntax
- To change default version of Java to 7
  - Override `open_jdk_jre.yml`

```
>$ cf set-env my-application JBP_CONFIG_OPEN_JDK_JRE '[version: 1.7.0_+, memory_heuristics: {heap: 85, stack: 10}]'
```

# Roadmap

- What are Buildpacks?
- Deploying to Cloud Foundry
- Using Buildpacks
- Buildpack API
- Java Buildpack
- Customization without Forking
- **Configure / Extend Java Buildpack**
- Using Docker in PCF

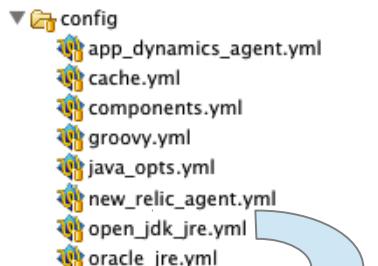
# Customization

- You may alter 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
  - Code at [github.com](https://github.com)
- ...Or simply downloading, modifying, and zipping.



# Customizing Configuration

- Most configuration options found in /config
  - determine behavior of a JRE, Container, or Framework



```
---
repository_root: "{default.repository.root}/openjdk/{platform}/{architecture}"
version: 1.8.0_+
memory_sizes:
  metaspace: 64m..
memory_heuristics:
  heap: 75
  metaspace: 10
  stack: 5
  native: 10
```

*repository\_root* and  
*version* typically at the  
start of each file.

# Locating Downloads

- URLs derived from repository root
  - `{default.repository.root}/openjdk/{platform}/{architecture}`
  - `download.pivotal.io.s3.amazonaws.com/openjdk/lucid/x86_64`
  - `index.yml` holds location of each version

```
# http://download.pivotal.io.s3.amazonaws.com/openjdk/lucid/x86_64/index.yml
---
1.8.0_25: https://download.run.pivotal.io/.../x86_64/openjdk-1.8.0_25.tar.gz
1.7.0_71: https://download.run.pivotal.io/.../x86_64/openjdk-1.7.0_71.tar.gz
1.8.0_31: https://download.run.pivotal.io/.../x86_64/openjdk-1.8.0_31.tar.gz
1.7.0_75: https://download.run.pivotal.io/.../x86_64/openjdk-1.7.0_75.tar.gz
1.8.0_40: https://download.run.pivotal.io/.../x86_64/openjdk-1.8.0_40.tar.gz
...
```

# Customization by Configuration: Tomcat

- Example: customizing the Tomcat artifact for download

```
# cloudfoundry/java-buildpack/config/tomcat.yml
---
tomcat:
  version: 8.0.+
  repository_root: "{default.repository.root}/tomcat"
...
```

```
# http://files.example.com/tomcat-custom/index.yml
---
8.0.18: https://download.run.pivotal.io/tomcat/tomcat-8.0.18.tar.gz
8.0.17: https://download.run.pivotal.io/tomcat/tomcat-8.0.17.tar.gz
7.0.59: https://download.run.pivotal.io/tomcat/tomcat-7.0.59.tar.gz
8.0.20: https://download.run.pivotal.io/tomcat/tomcat-8.0.20.tar.gz
```

# Resource Configuration

- Tomcat container supports simple customization of **context.xml** and **server.xml**
  - Files will *overlay* sandbox provided values.

```
resources/tomcat/conf
  └── context.xml
  └── server.xml
```

- Not just for Tomcat
  - JDK, New Relic, ...

# Extending the Buildpack - 1

- You can *extend* the Java Buildpack
  - To add different JRE, Container, or Framework
- Implement support class (Ruby) in the appropriate directory
  - with additional support classes as necessary

```
lib/java_buildpack
  └── jre
      └── container
          └── framework
```

# Extending the Buildpack - 2

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

```
# Return String or an Array<String> that identifies the component to be
# used in staging, or nil.
def detect

# Modifies the application's file system. Component is expected to
# transform the application's file system in whatever way is necessary
# (e.g. downloading files or creating symbolic links) to support the function
# of the component. Status output written to STDOUT is expected.
def compile

# Modifies the application's runtime configuration to support the function
# of the component. Create the command required to run the application,
# taking context values into account when creating the command. Container
# components are expected to return the command required to run the application.
def release
```

# Extending the Buildpack - 3

- Add new support class to **config/components.yml**

```
# Configuration for components to use in the buildpack
---
containers:
  - "JavaBuildpack::Container::DistZip"
  - "JavaBuildpack::Container::Groovy"
  - "JavaBuildpack::Container::JavaMain"
  - "JavaBuildpack::Container::PlayFramework"
  - "JavaBuildpack::Container::Ratpack"
  - "JavaBuildpack::Container::SpringBoot"
  - "JavaBuildpack::Container::SpringBootCLI"
  - "JavaBuildpack::Container::Tomcat"
  - "JavaBuildpack::Container::YOUR-CONTAINER-HERE"

jres:
  - "JavaBuildpack::jre::OpenJdkJRE"
#  - "JavaBuildpack::jre::OracleJRE" ... or here if JRE...
frameworks:
  - "JavaBuildpack::Framework::AppDynamicsAgent" ... or here if framework"
```

# More on Customization

- Much more information and documentation included in the GitHub repository

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

- Your Cloud Foundry installation may not allow custom buildpacks
  - Administrator option to enable/disable



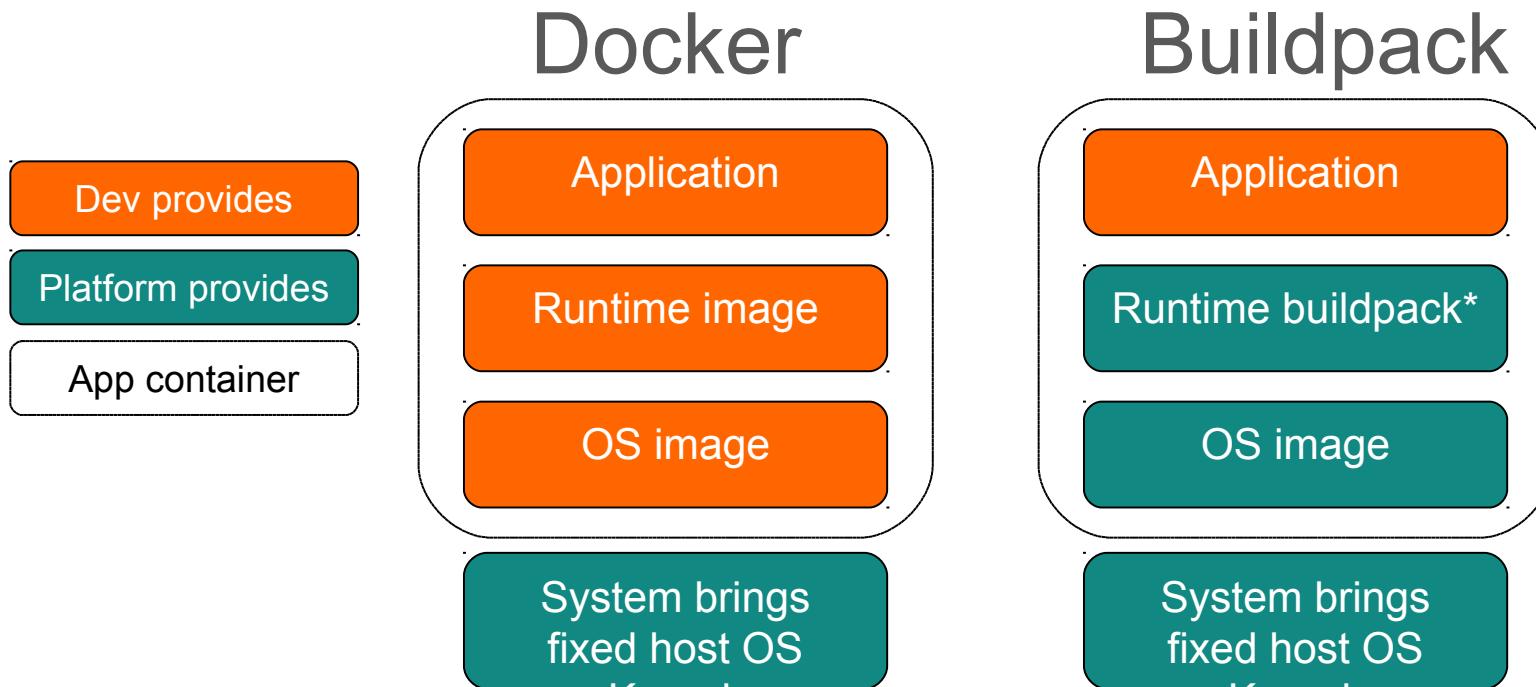
# Offline Buildpacks

- Wish to avoid downloading buildpacks from Internet
- Java Buildpack can be packaged as *Offline* Buildpack
  - Builds droplets *without* internet connection
- One-time build process
  - Internally packages latest version of each dependency within the buildpack
  - Disables remote downloads
  - About 180M in size
  - Install using `cf create-buildpack/update-buildpack`
  - <https://github.com/cloudfoundry/java-buildpack/blob/master/docs/buildpack-modes.md>
- **Note:** Pivotal CF ships with offline buildpacks!

# Roadmap

- What are Buildpacks?
- Deploying to Cloud Foundry
- Using Buildpacks
- Buildpack API
- Java Buildpack
- Customization without Forking
- Configure / Extend Java Buildpack
- **Using Docker in PCF**

# Docker vs Buildpack



\*Devs may also provide their own buildpack

# Docker in PCF

- Container:
  - Both Docker & Garden-runC use OCI libraries
  - Fetch & cache layers associated with a Docker image
  - Combine & mount the layers as container's root file system
  - Diego runs & monitors the processes inside the container
- Cloud Controller:
  - Stores the metadata associated with the Docker image
  - Runs the start command specified in Docker image
  - Instructs Diego & Gorouter to route traffic to lowest-numbered port exposed in Docker image (default 8080)

# Docker – deploying application

- Admin to enable Docker-based apps to run:

```
cf enable-feature-flag diego_docker
```

- Push a Docker image from Docker Hub:

```
cf push APP-NAME --docker-image REPO/IMAGE:TAG
```

- Push a Docker image from another registry:

```
cf push APP-NAME --docker-image  
MY-REGISTRY.DOMAIN:PORT/REPO/IMAGE:TAG
```

Note: PWS currently disables Docker feature (*cf feature-flags*)

# Summary

- After completing this lesson, you should have learned:
  - What a buildpack is and how to use it
  - The basic API of a Buildpack
  - The behavior of the Java Buildpack
  - How to configure / extend a Buildpack
  - How to use Docker image in PCF

# Lab

## Using Buildpacks

# Service Brokers

Implementing a Managed Service

# Roadmap

- **Custom Services using Service Brokers**
- Writing a Custom Service Broker

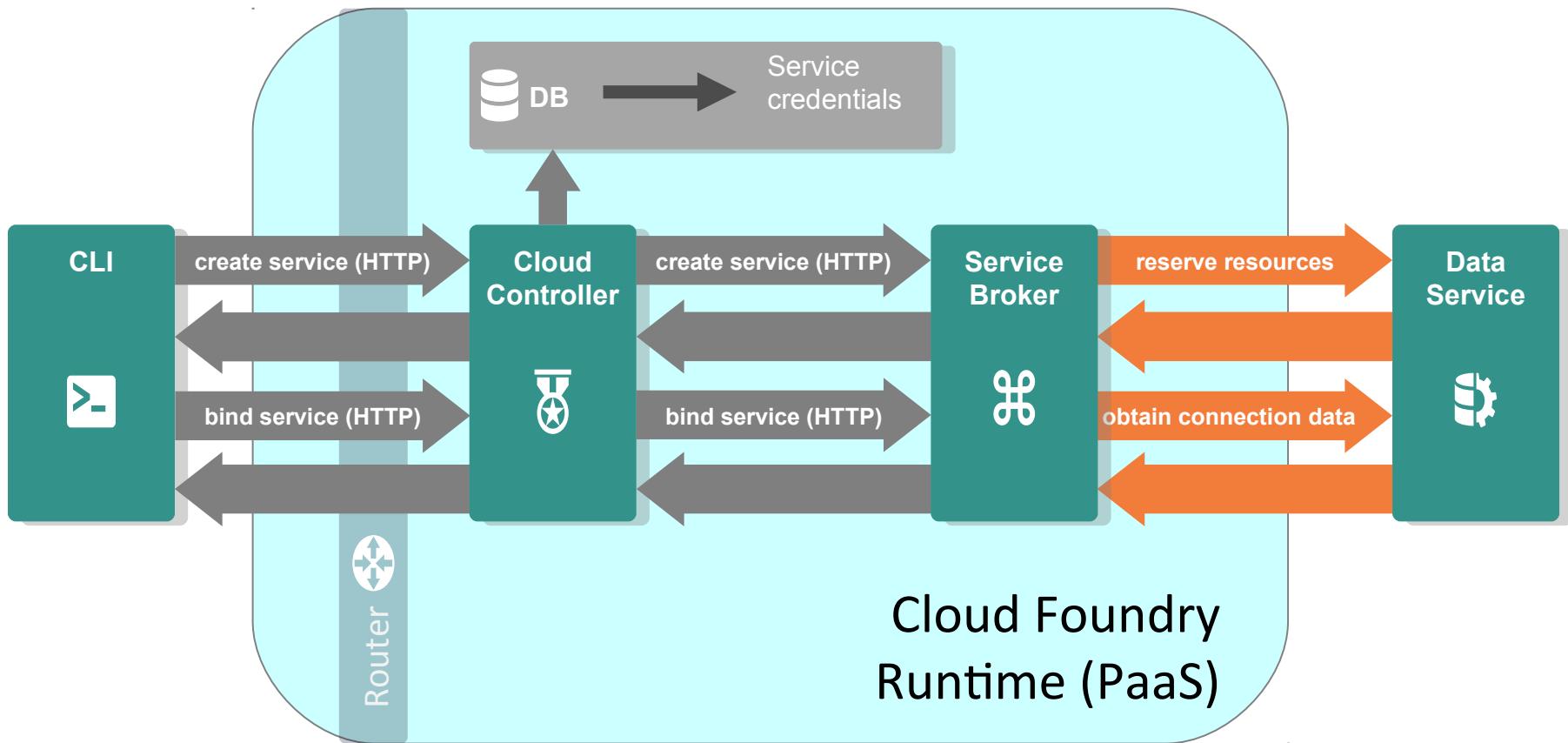




# Managed Services

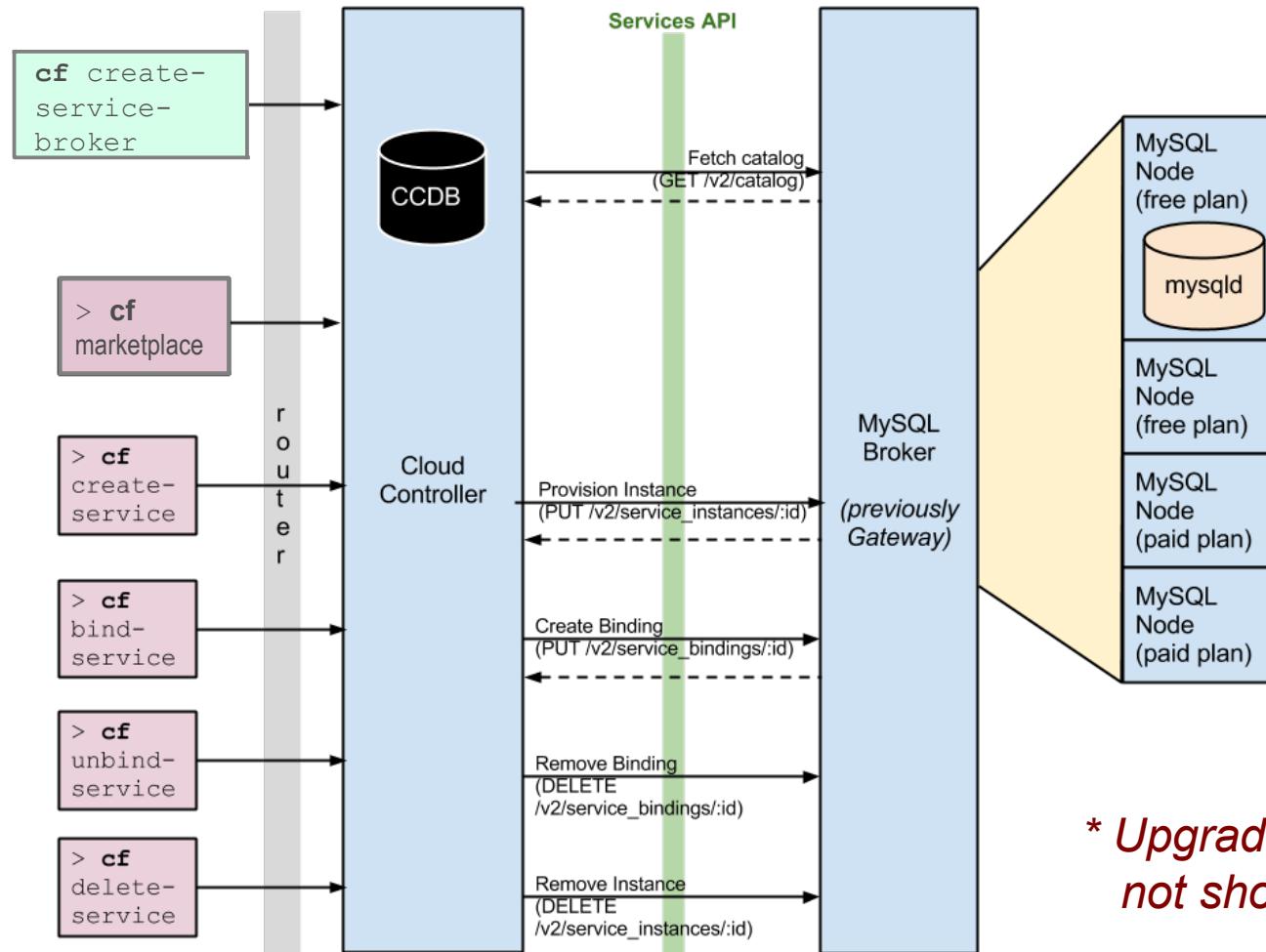
- Managed services must implement the Service Broker API
  - Appear in the “*Marketplace*”
- The service-broker can do *anything*
  - Whether a service is actually managed by PCF depends on the implementation

# Creating and Binding a Service





# Service Broker Example (MySQL v1)



# Service Broker API

- REST based API over HTTP
  - So Service Brokers can run *anywhere*
  - Although they are often PCF-deployed applications
- <https://github.com/openservicebrokerapi/servicebroker/>

# Service Broker API Summary

- API Endpoints are \*
  - **Catalog Mgmt** – describe the plans offered by the service
  - **Provision** – create a service instance
  - **Deprovision** – delete a service instance
  - **Bind** – create a binding between an app and service instance
  - **Unbind** – delete binding
  - **Change plan** – upgrade the plan of an existing service

\* Not a complete list

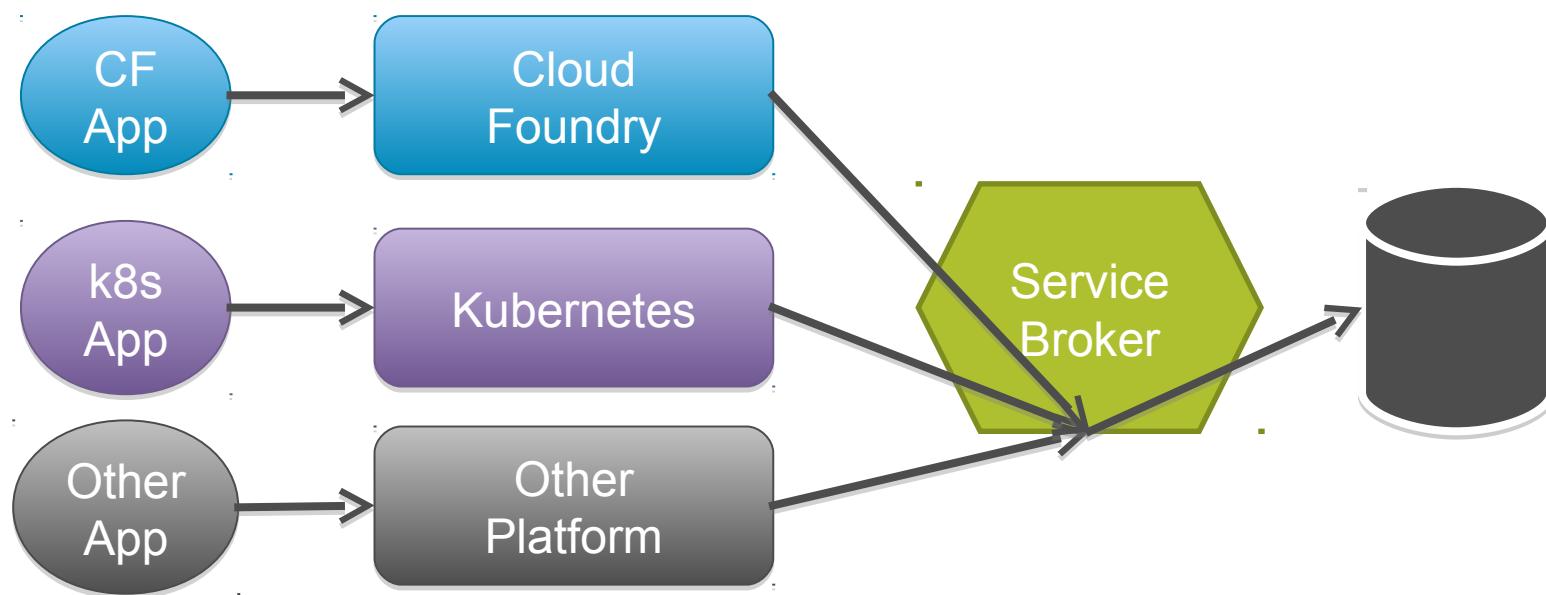
# Server Broker API



- [`GET /v2/catalog`](#)
  - List services and plans available from this broker.
- [`PUT /v2/service\_instances/:instance\_id`](#)
  - Create a new service instance.
- [`PUT /v2/service\_instances/:instance\_id/service\_bindings/:id`](#)
  - Create a new binding to a service instance.
- [`DELETE /v2/service\_instances/:instance\_id/service\_bindings/:id`](#)
  - Unbind from a service instance.
- [`DELETE /v2/service\_instances/:instance\_id`](#)
  - Delete a service instance
- [`PATCH /v2/service\_instances/:instance\_id`](#)
  - Update service to different plan

# Open Service Broker

- OSS Project to extend CF's Service Broker abstraction to other platforms
  - Such as Kubernetes

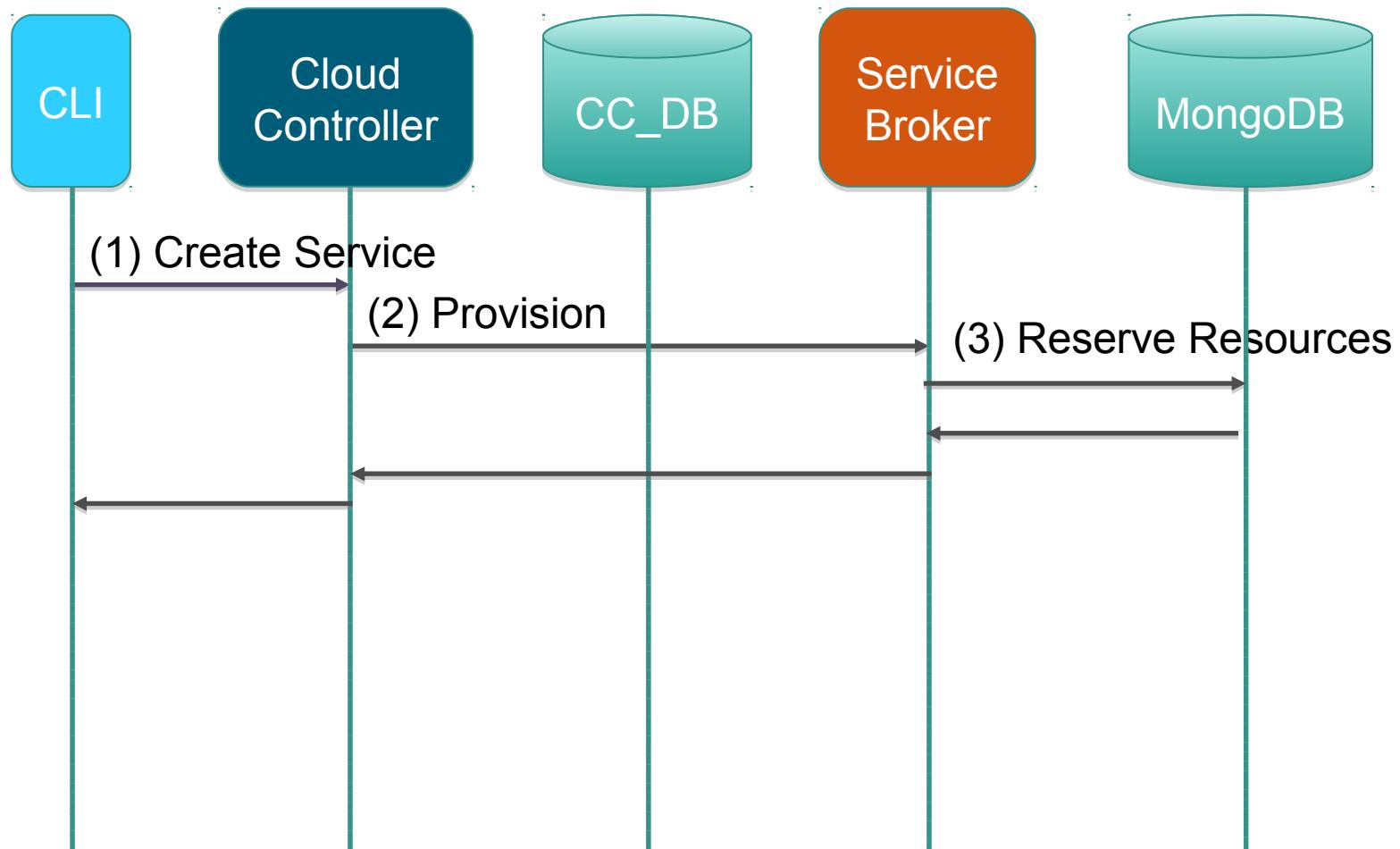


[https://github.com/openservicebrokerapi/servicebroker/blob/master/\\_spec.md](https://github.com/openservicebrokerapi/servicebroker/blob/master/_spec.md)

# What should I provision?

- Provision anything you want
- Provisioning, updating, and de-provisioning can be done synchronously *or* asynchronously
  - Asynchronous provisioning is used when the provisioning process takes a significant amount of time eg. creating new VM's for the service

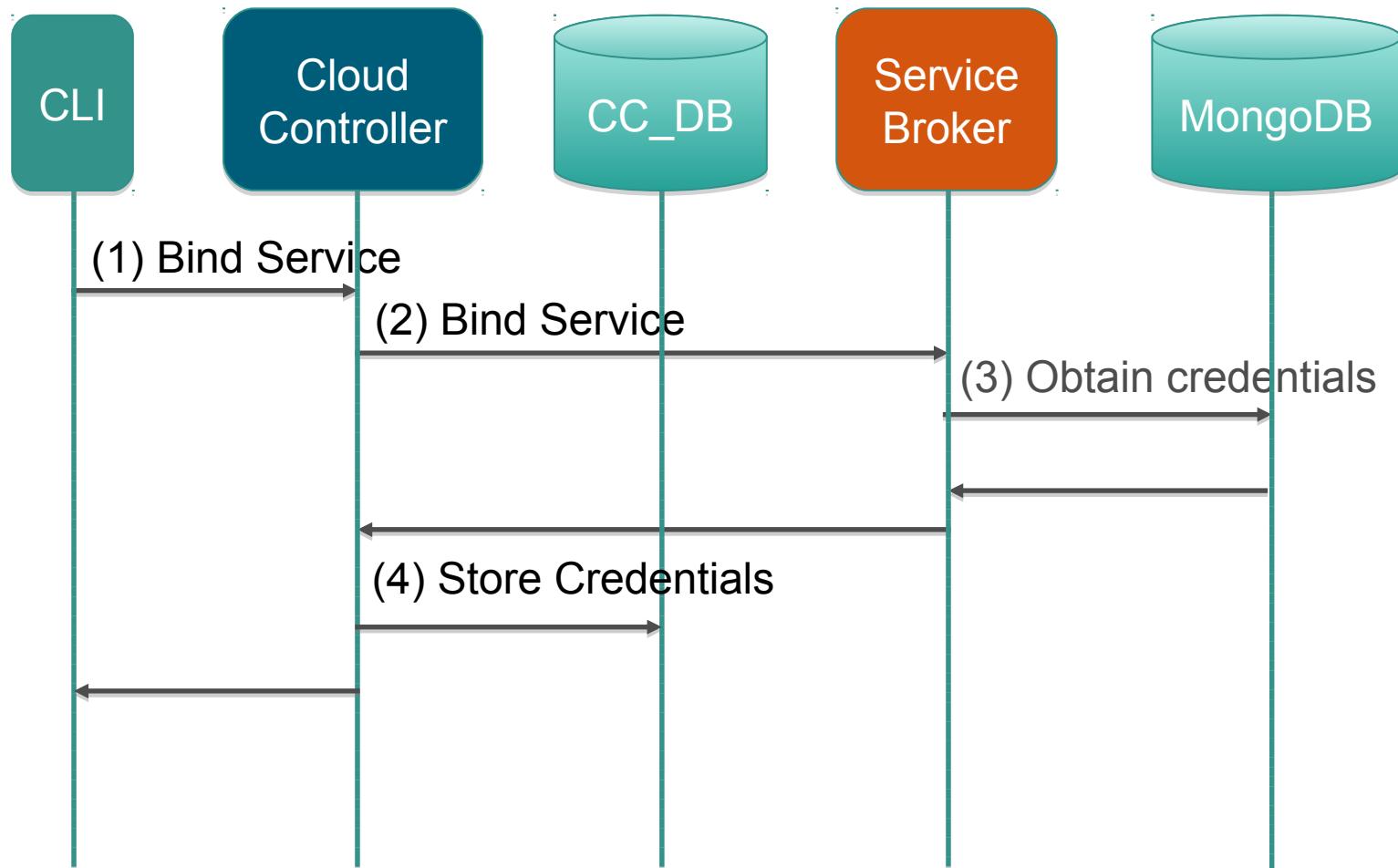
# Provision Sequence



# Binding

- Provide unique credentials per binding
  - Made available to application via **VCAP\_SERVICES**
  - Stored in CCDB for future use

# Bind Sequence



# Roadmap

- Custom Services using Service Brokers
- **Writing a Custom Service Broker**



# 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
- Broker can be implemented
  - As a separate application
  - By adding required http endpoints to an existing service
- Deploy using `cf create-service-broker`
  - Need administrator privileges to make generally available
  - A developer *can* enable a broker for a specific space

# Service Instance Provisioning Examples

- 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
- For any of the above provisioning could also provide a
  - A database with business schema already there
  - A copy of a full database, *for example* a QA database that is a facsimile of the production database
- For non-data services
  - Provisioning could just mean getting an account on an existing system

# Service Broker Deployment Models

- Many deployment models are possible:
  - Entire service packaged and deployed by BOSH alongside Cloud Foundry (Ops Manager Tile)
  - Broker packaged and deployed by BOSH alongside Cloud Foundry
    - rest of the service deployed and maintained by other means
  - Broker (and optionally service) pushed as an application to Cloud Foundry user space
  - Entire service, including broker, deployed and maintained outside of Cloud Foundry by other means

# Custom Service Broker

- Spring Cloud Service Broker project
  - <http://cloud.spring.io/spring-cloud-cloudfoundry-service-broker>



# Register New Service

- Register your new service-broker
  - `cf create-service-broker`
    - and other similar commands
- For more information see
  - <http://docs.cloudfoundry.org/services/managing-service-brokers.html>



# Summary

We have learned about

- Custom Services using Service Brokers
- Writing a Custom Service Broker



# Lab

## Service Brokers

# Continuous Delivery

Ready to deploy *any* time

Natural fit with Cloud Foundry

# Agenda

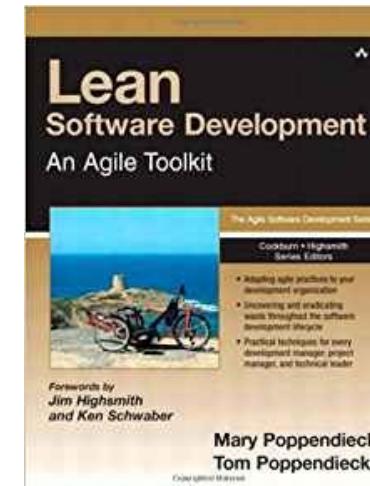
- **Define Continuous Delivery**
- Strategies for Success
- How Pivotal Cloud Foundry Enables Continuous Delivery

# What is Continuous Delivery?

“How long would it take in your environment to push a single line of code to production?”

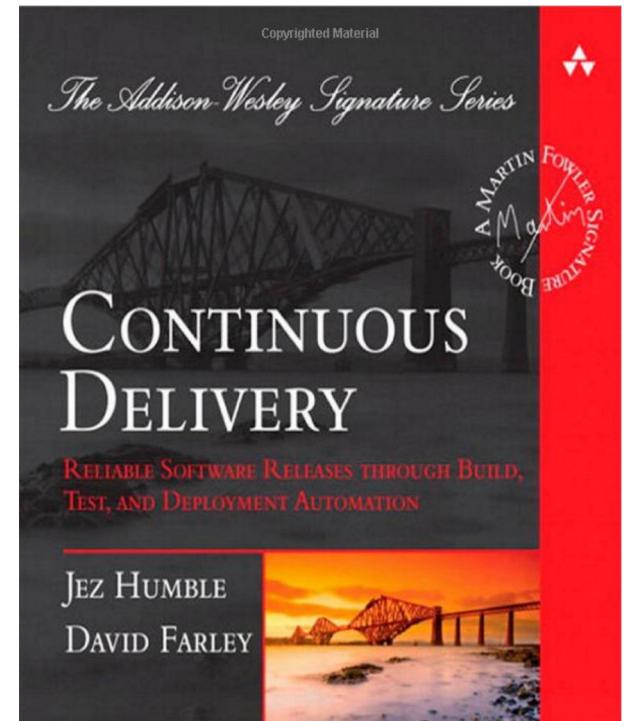


— *Mary Poppendieck*



# Continuous Delivery

- Build software so that it can be released to production *at any time*



# Continuous Deployment

Continuous Deployment

!=

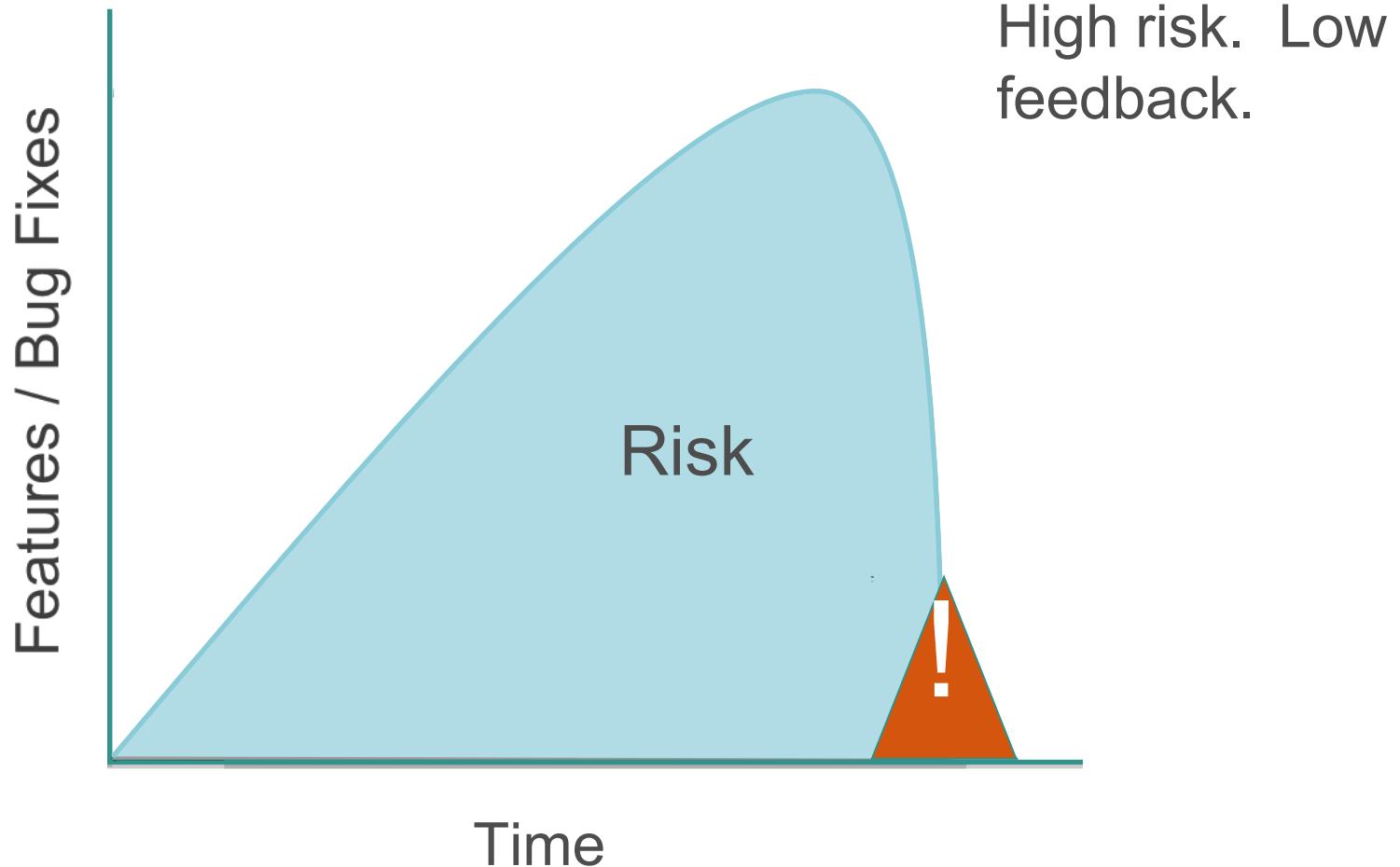
Continuous Delivery

# Who Decides When to go to Production?

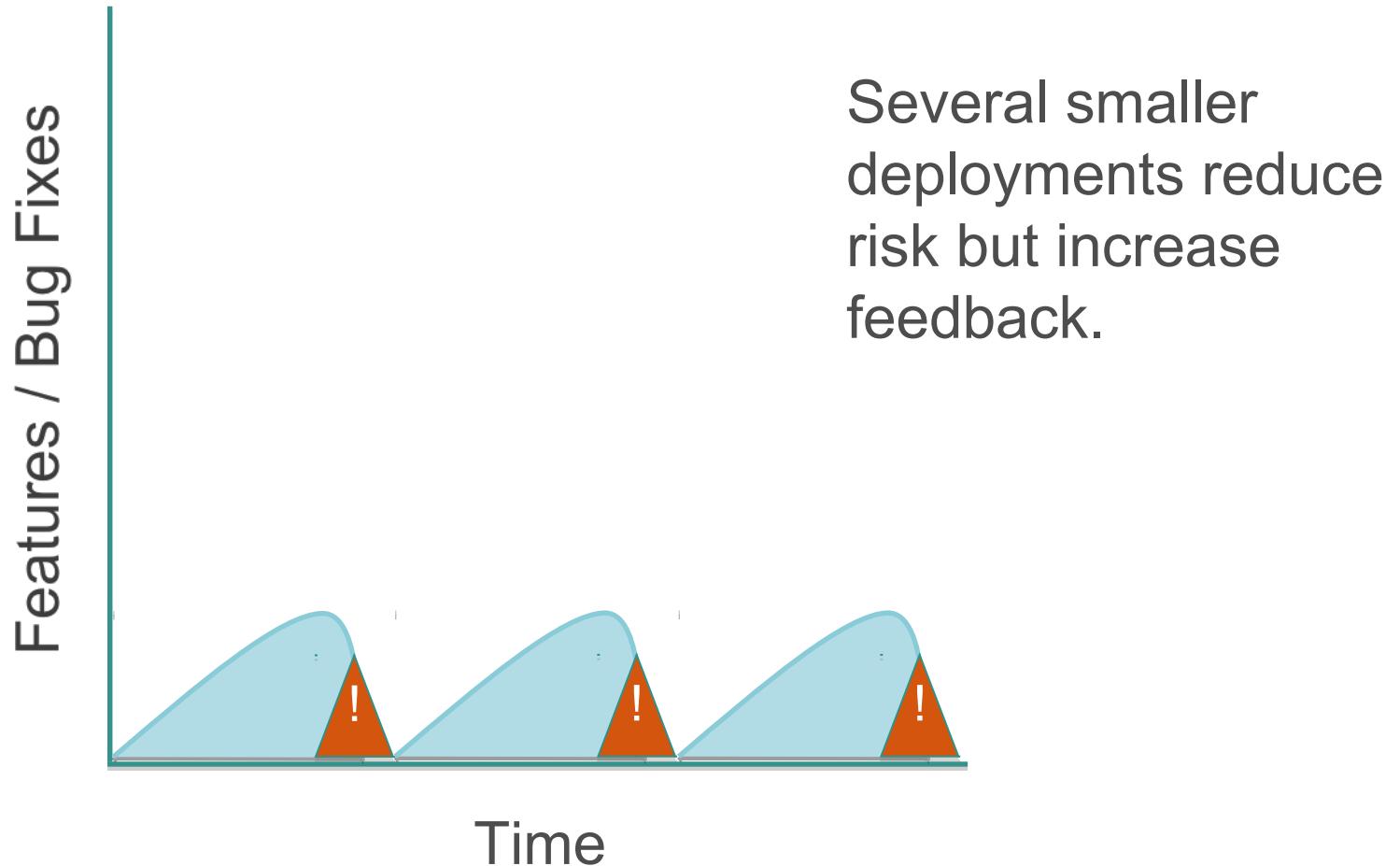
- The business
  - **BUT:** Technical teams should be ready!



# Waterfall Application Delivery



# Application Delivery with Continuous Delivery



# Agenda

- Define Continuous Delivery
- **Strategies For Success**
- Pivotal Cloud Foundry Enables Continuous Delivery

# Continuous Delivery Strategies – DOs

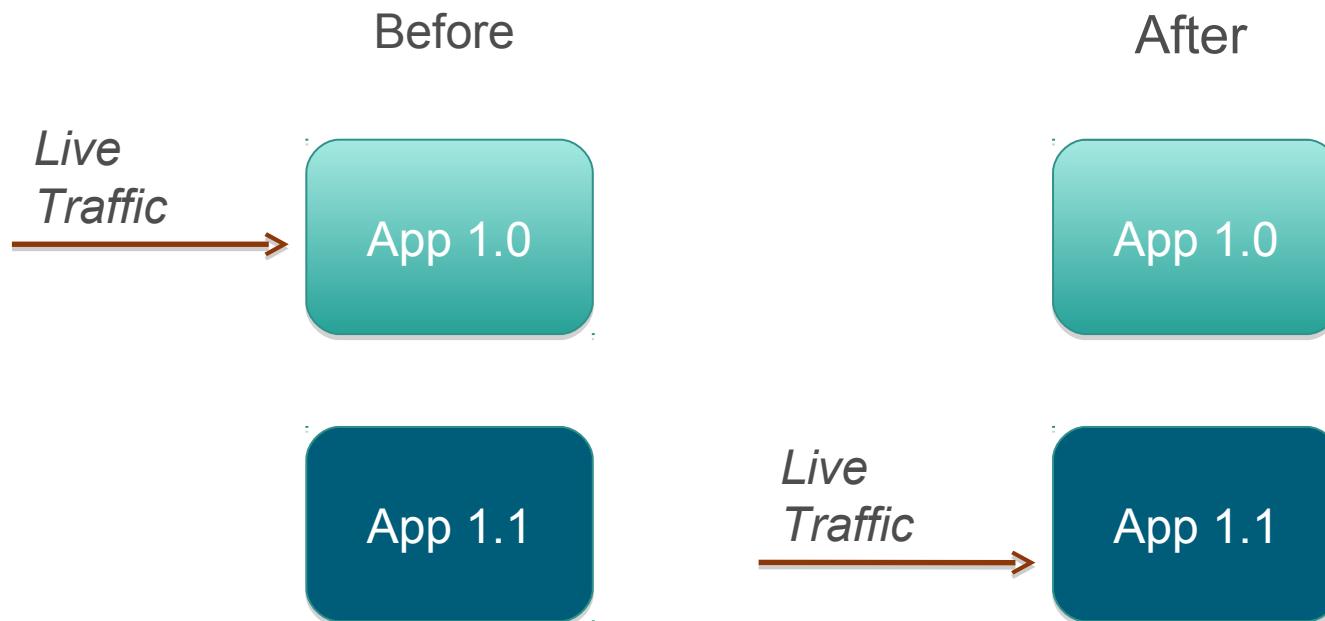
- Do implement continuous integration
- Do externalize environment specific configuration
- Do recreate app environments frequently
- Do ensure that database changes are automated
- Do deploy less more frequently
- Do automate all testing where test failures would prevent a production release from occurring
- Do try to use tools that support the process

# Continuous Delivery Strategies – DON'Ts

- Do *not* assume existing processes are right
- Do *not* create environment specific packages
- Do *not* use a different process for different environments

# Continuous Delivery Strategies

- Blue-Green deployments can be used with a continuous delivery pipeline.

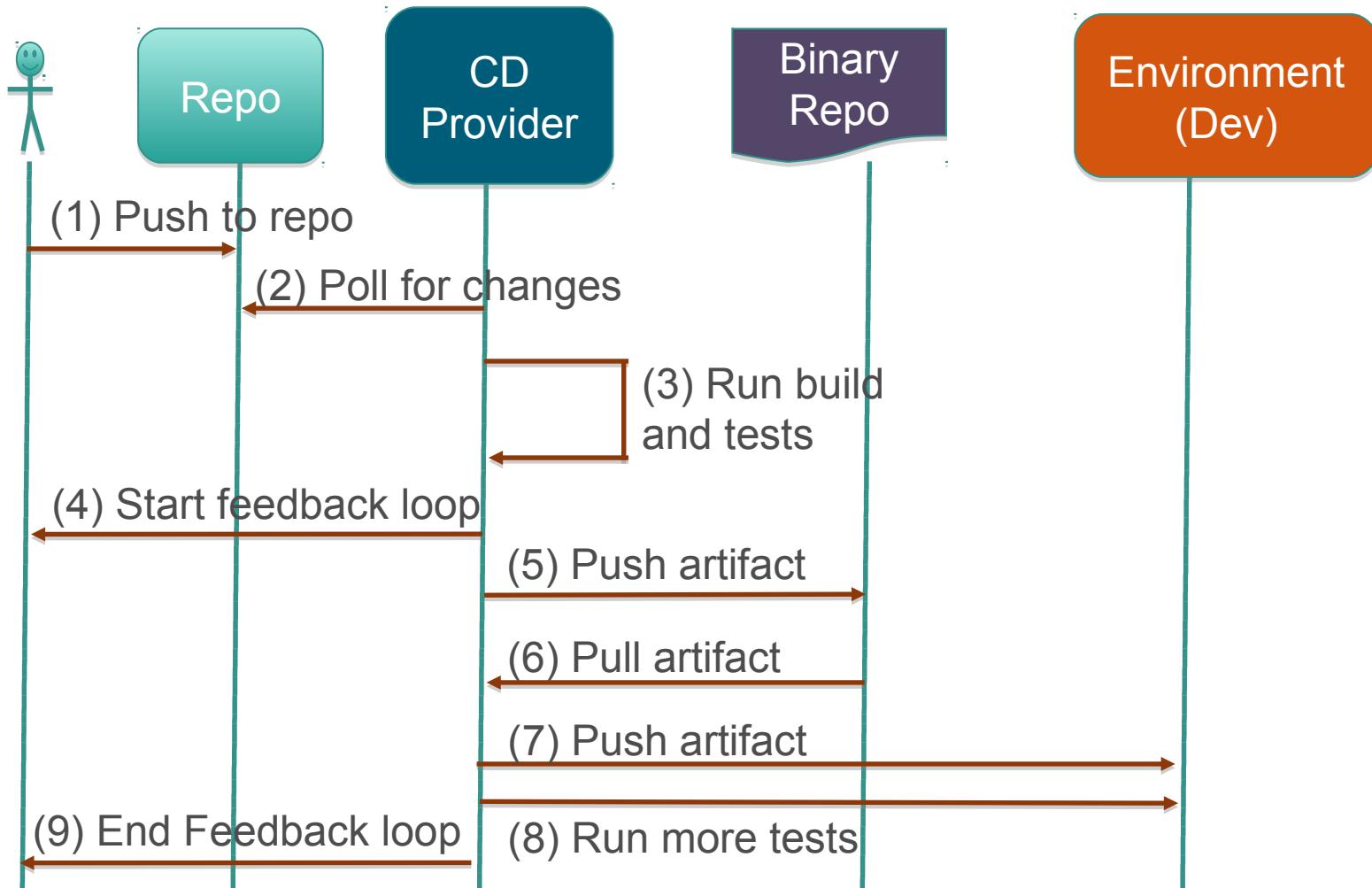


# Implications of Continuous Delivery

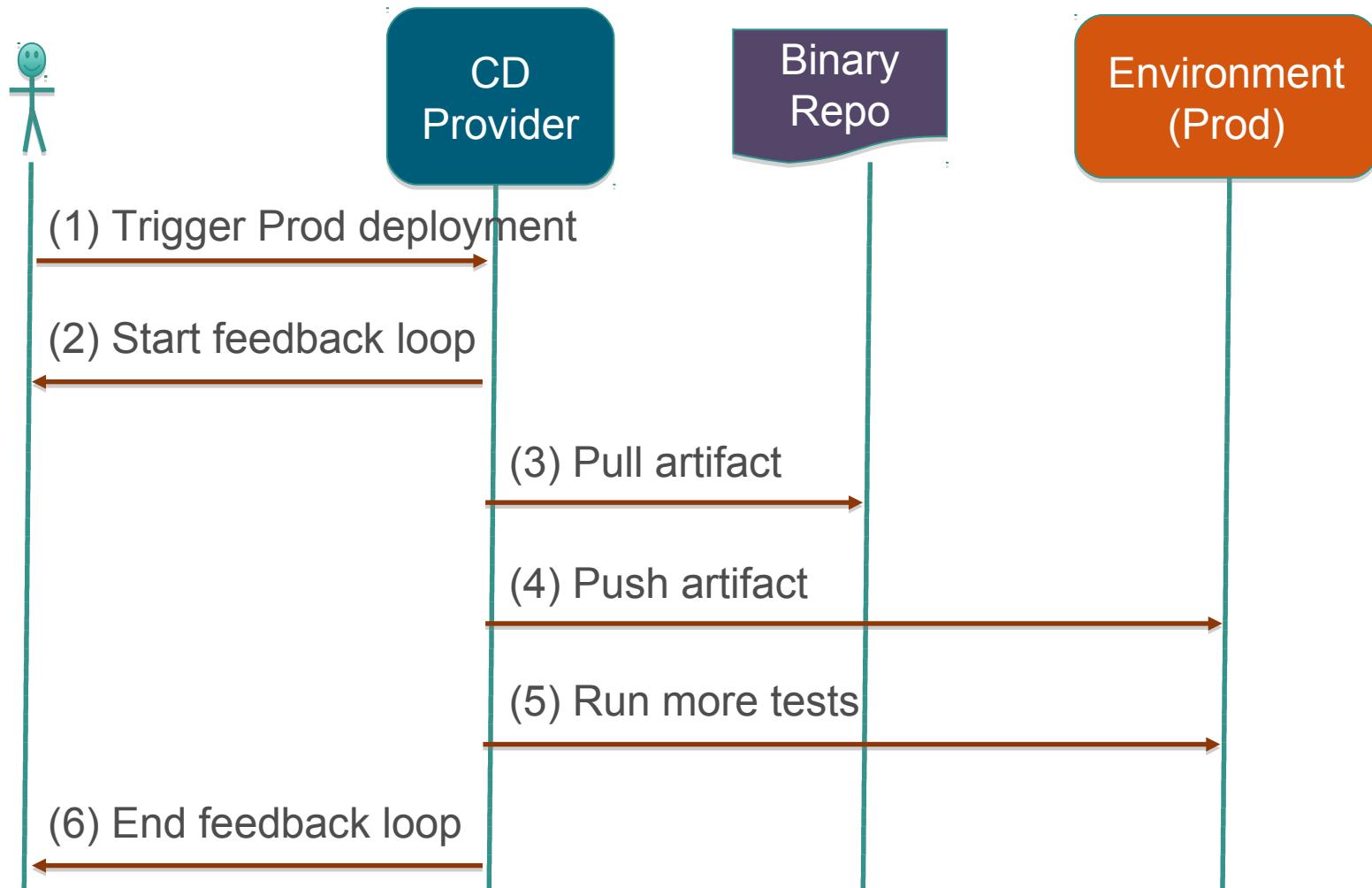
- Software is deployable *throughout* its lifecycle
- Prioritize keeping the software deployable over working on new features
- Can get fast, automated feedback on the production readiness of a system *whenever* it is changed
- Can perform “push-button” deployments of any version of the software to any environment on demand

<http://martinfowler.com/bliki/ContinuousDelivery.html>

# Continuous Delivery Sequence (Development)



# Continuous Delivery Sequence (Production)



# Agenda

- Define Continuous Delivery
- Strategies
- **Pivotal Cloud Foundry Enables Continuous Delivery**

# PCF Enabling Continuous Delivery

- Get a new application environment in seconds
- Enables complete consistency between environments
- A consistent API to automate deployments
- Inject environment specific configuration
- Promote applications through environments with the same process



CLOUD FOUNDRY

Pivotal™

# Lab

## Continuous Delivery using Jenkins

# Routing Service

Intercepting incoming requests

# Agenda

- **Purpose**
- Request Flow

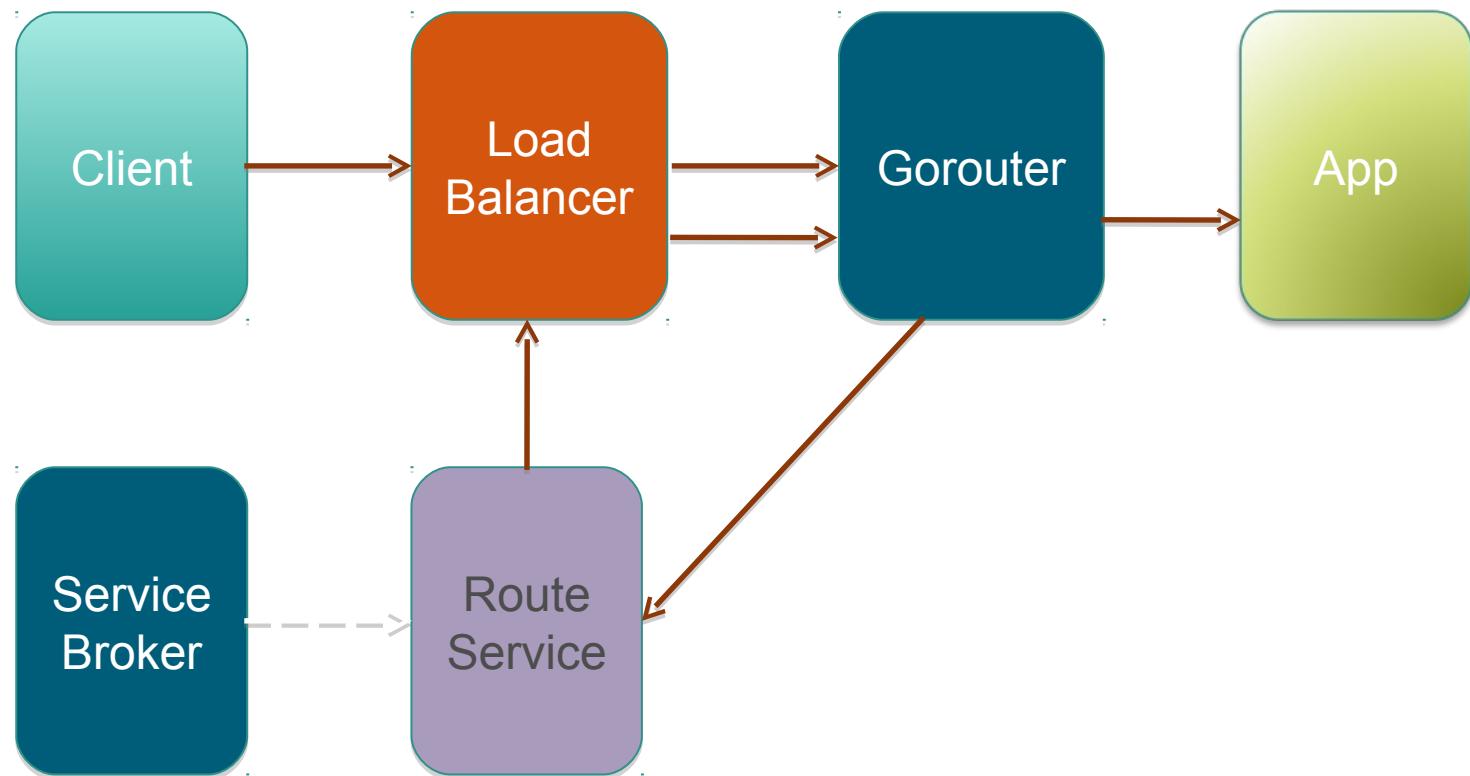
# Route Services - Purpose

- Provide transformation or processing to requests before/after they reach an application
- Examples:
  - Authentication
  - Rate Limiting
  - Logging
  - Caching

# Agenda

- Purpose
- Request Flow

# Route Service Request Flow



# Route Service – Headers

- **X-CF-Forwarded-Url**
- Contains the URL of the application route

# Route Service – Headers

- **X-CF-Proxy-Signature**
- **X-CF-Proxy-Metadata**
- Used by the Gorouter to validate the request and pass through to the application

# Lab

## Using a Route Service

# High Availability in CF

Production features

4 Levels of High Availability

# Learning Objectives

- After completing this lesson, you should be able to:
  - Explain the Four Levels of High Availability



# Agenda

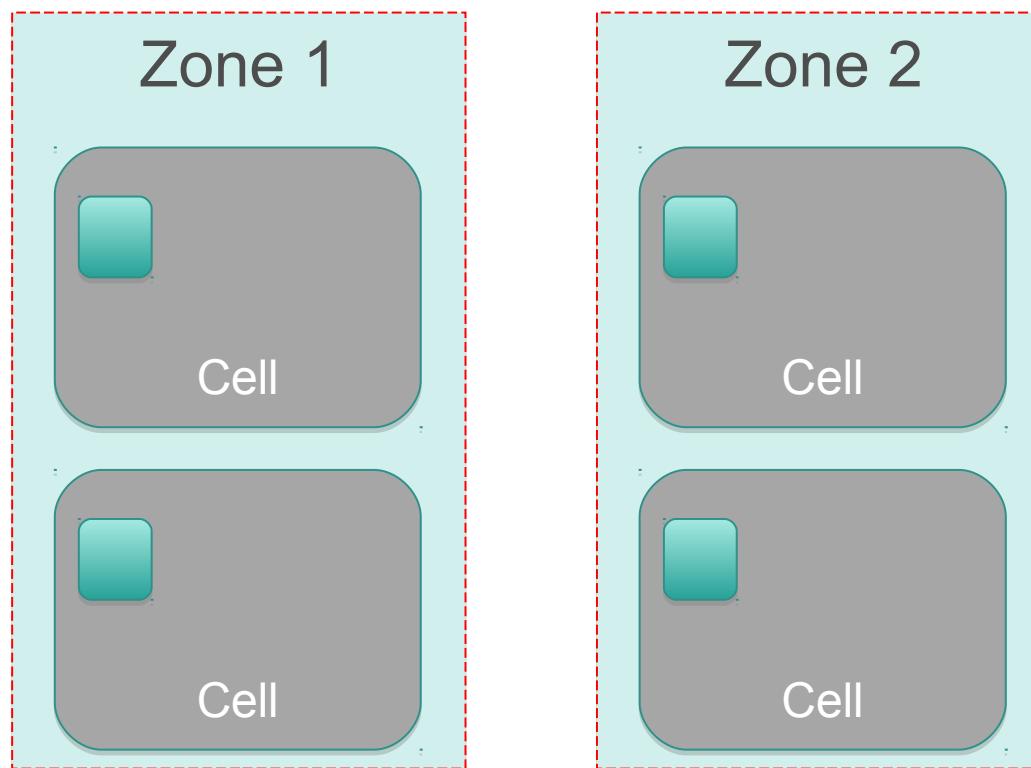
- High Availability in Cloud Foundry

# Four Levels of High Availability

1. BOSH detects and recreates failed VMs
2. BOSH also detects and restarts failed processes
3. Application instances are automatically spread
  - Across different Cells
  - Across Cells in different Availability Zones
4. Cloud Foundry *Converger* detects if an application instance has failed and restarts it

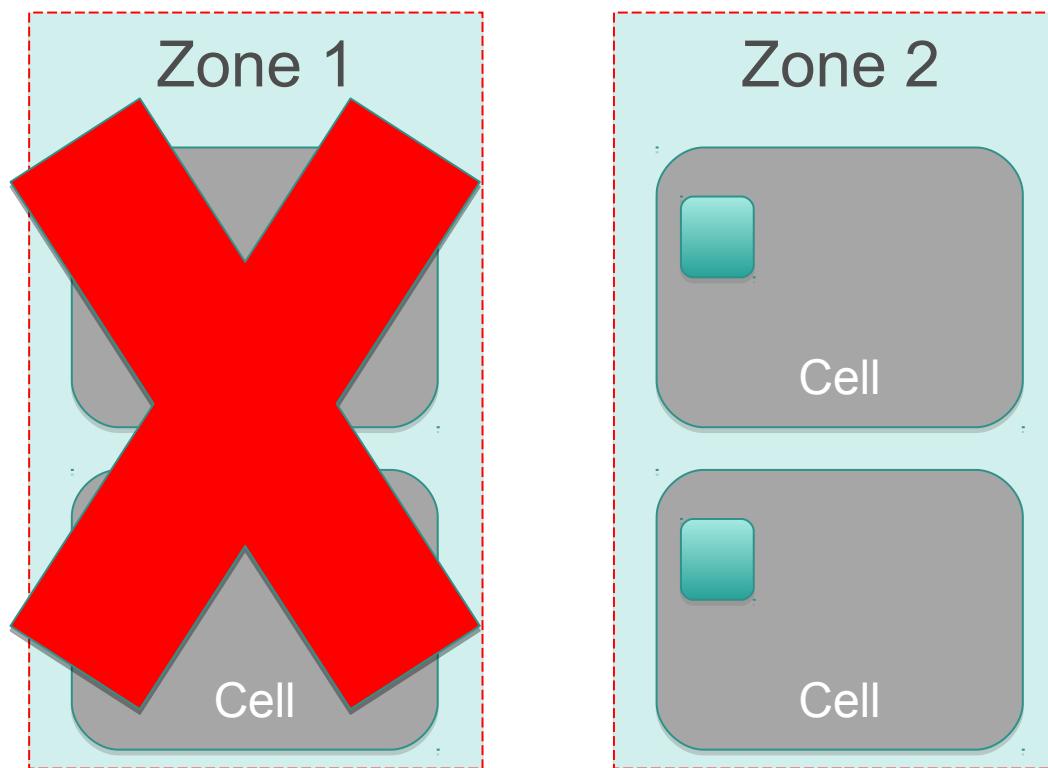
# Availability Zones

- Application instances are evenly distributed across availability zones.



# Availability Zones

- Application stays up despite losing an AZ

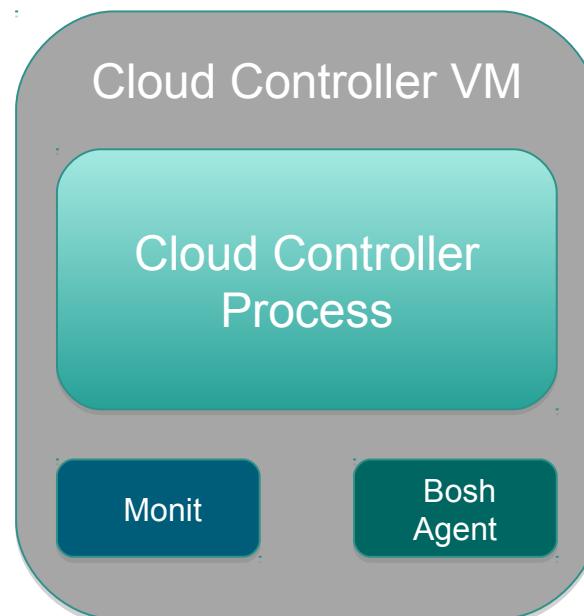


# Availability Zones and the Elastic Runtime

- Internal components *also* spread across AZs
  - ER components like Cells, Routers
  - Loggregator components (Doppler, Traffic Controller)
  - Database servers
    - CCDB: Clustered MySQL servers
    - BBS: Uses multiple *etcd* servers
- PCF Administrators determine the number of instances of these components to setup

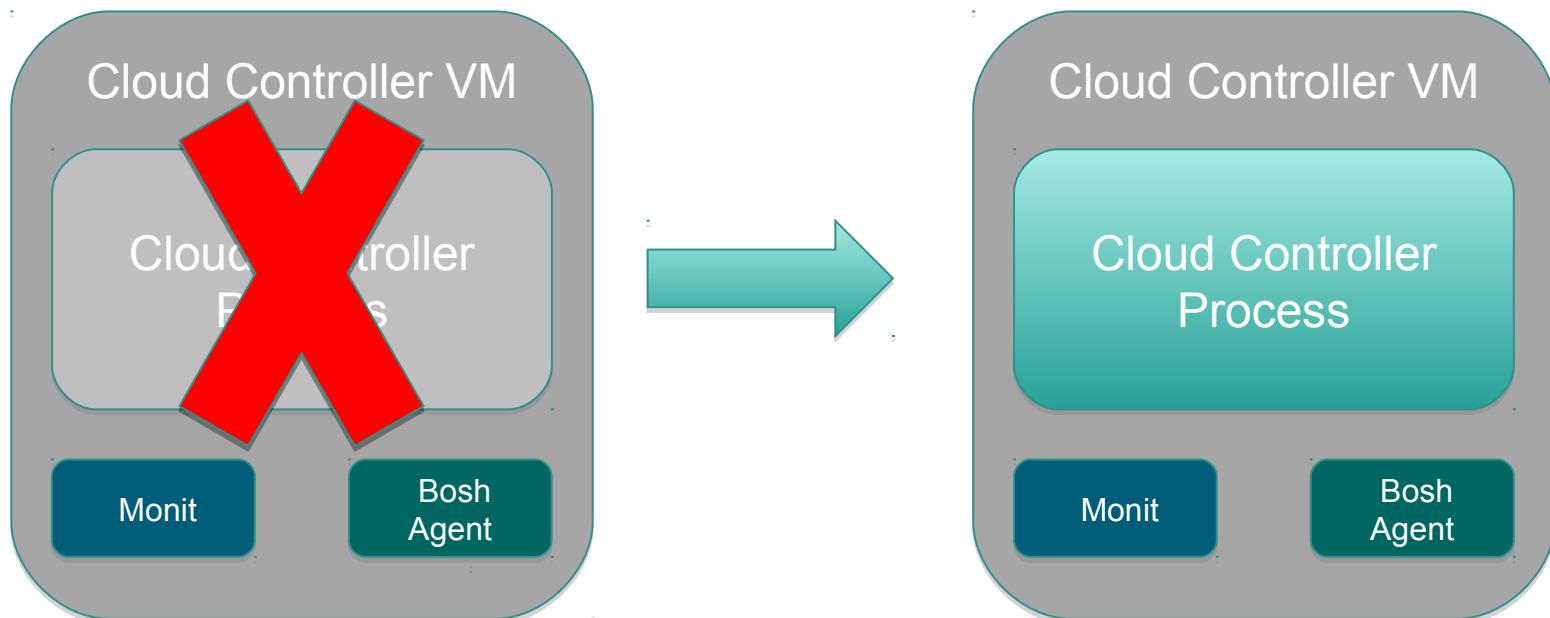
# BOSH Managed Processes

- Elastic runtime processes are monitored and automatically restarted



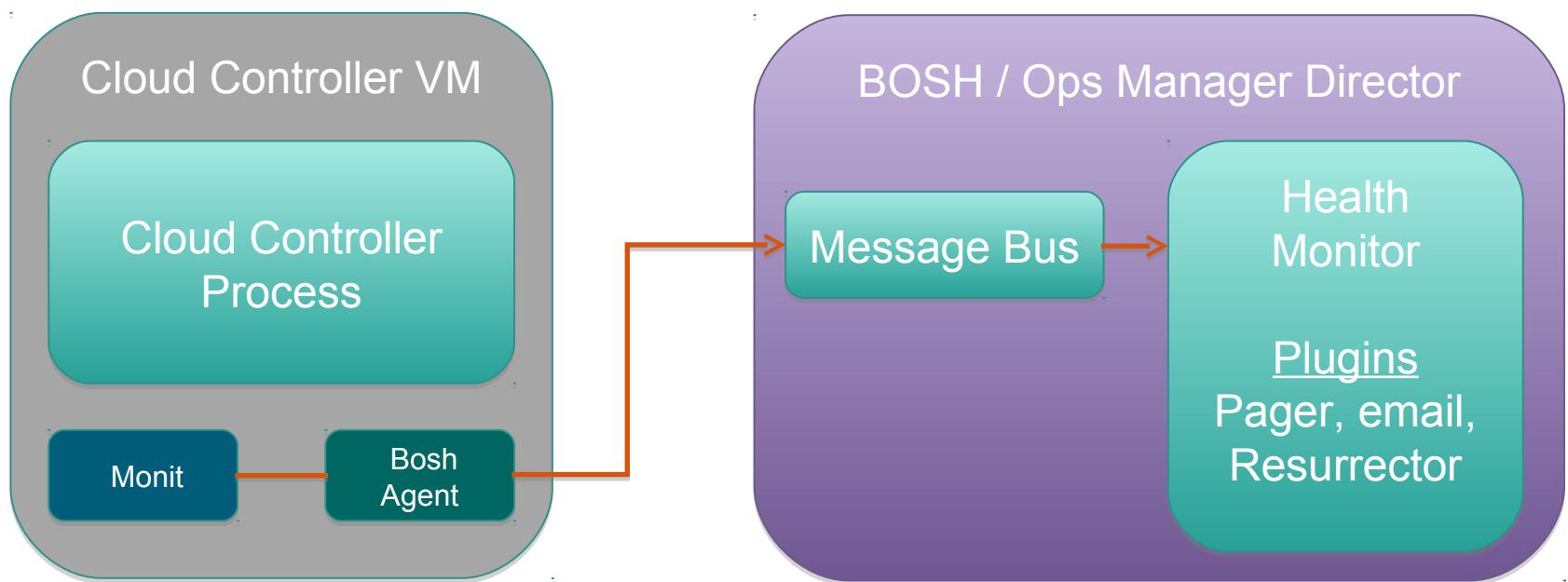
# BOSH Managed Processes

- Elastic runtime processes are monitored and automatically restarted



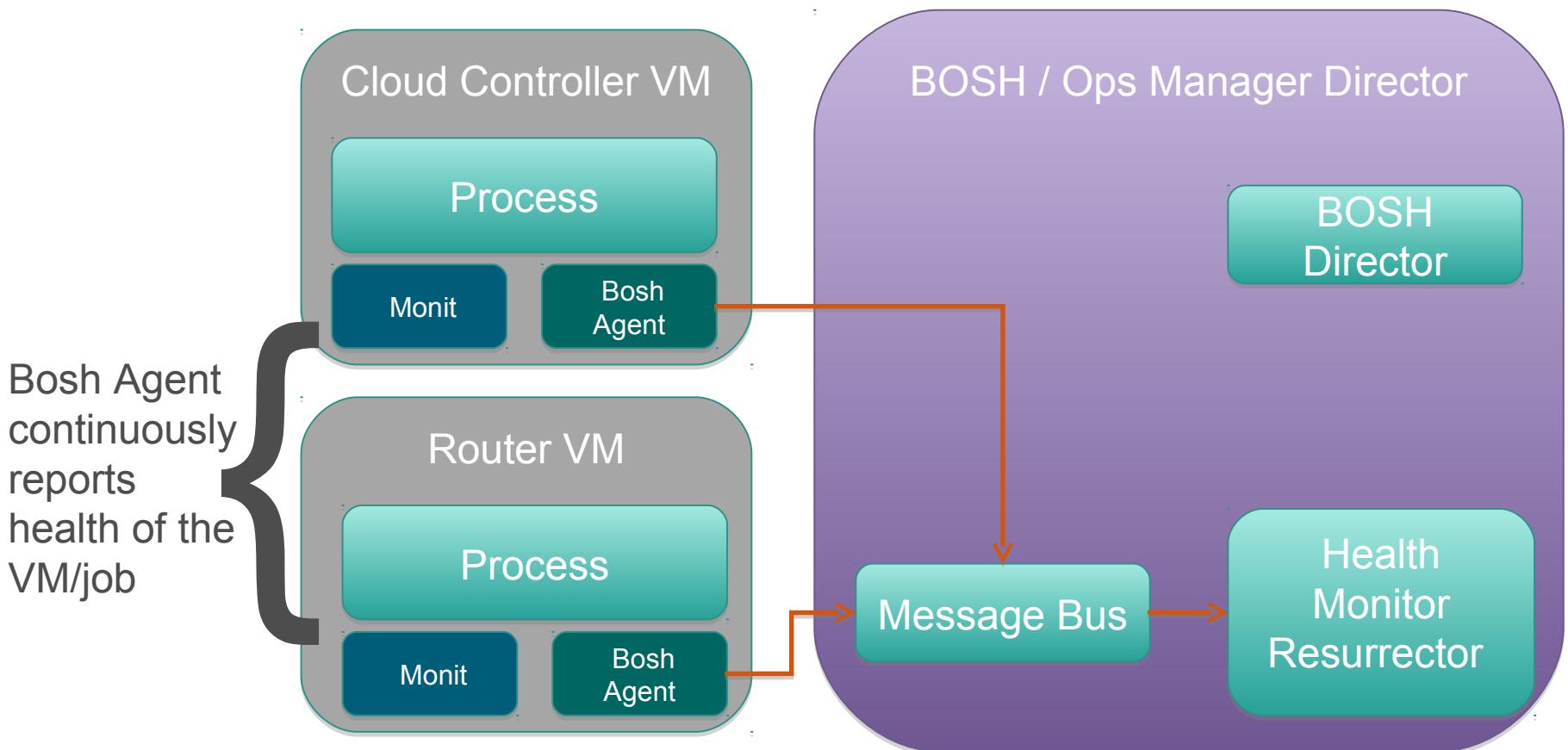
# BOSH Managed Processes

- Restart event is reported back to the Health Monitor for further investigation



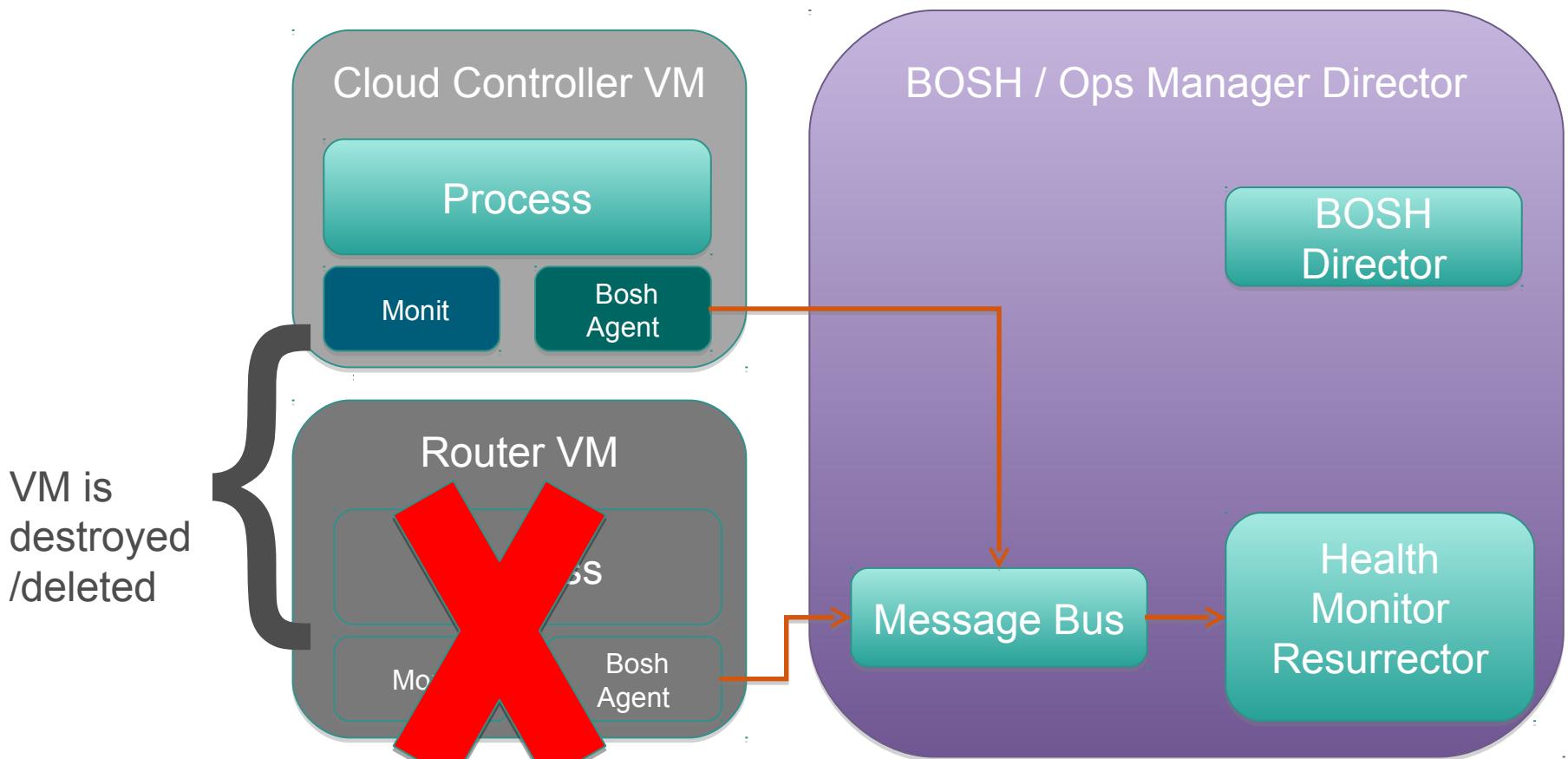
# Failed VMs

- Failed VMs will be recreated automatically.



# Failed VMs

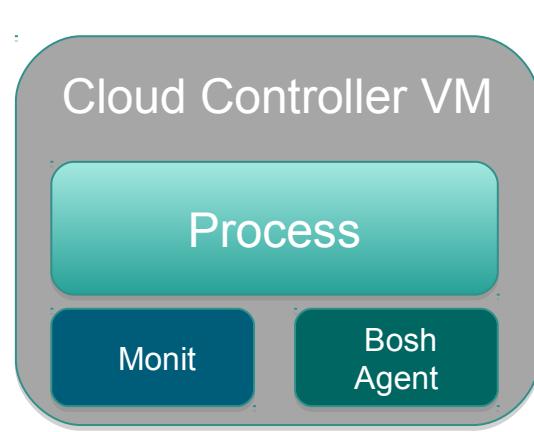
- Failed VMs will be recreated automatically.



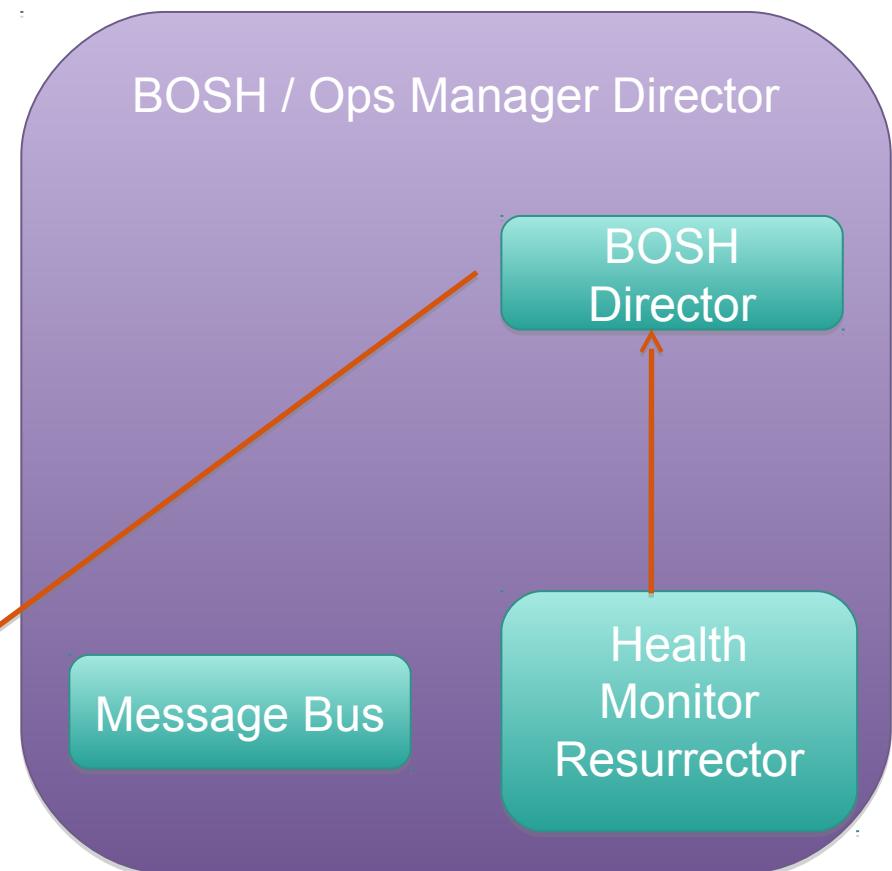
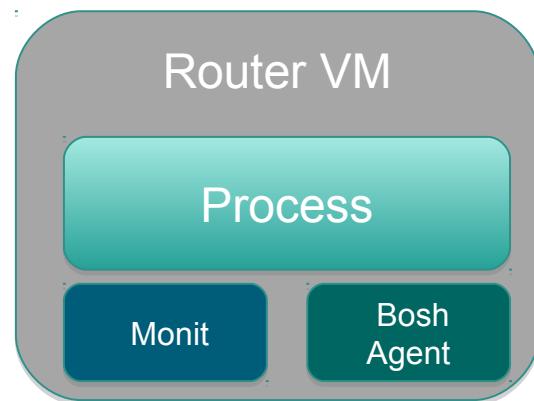
# Failed VMs

- Failed VMs will be recreated automatically

2) Director recreates VM/Job

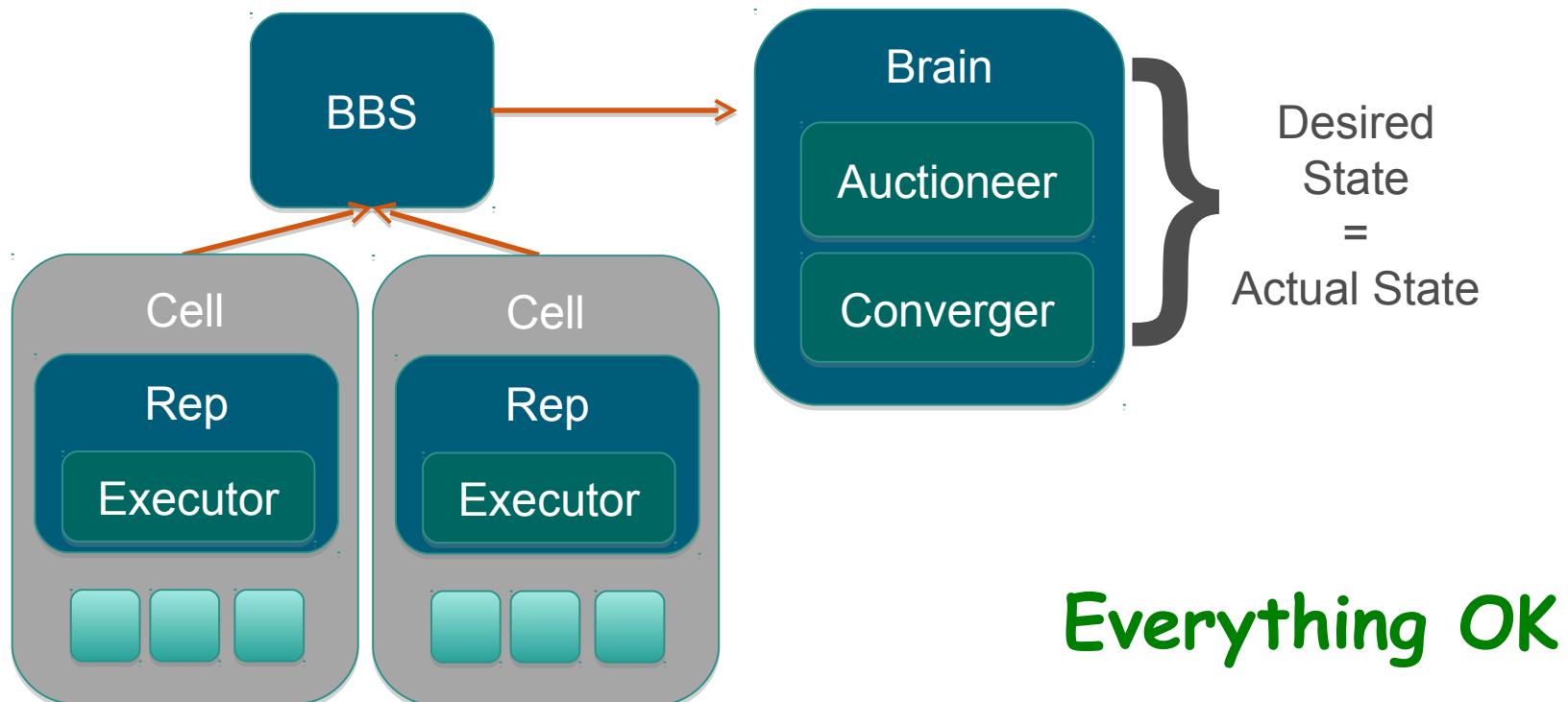


1) Resurrector determines there is a missing VM



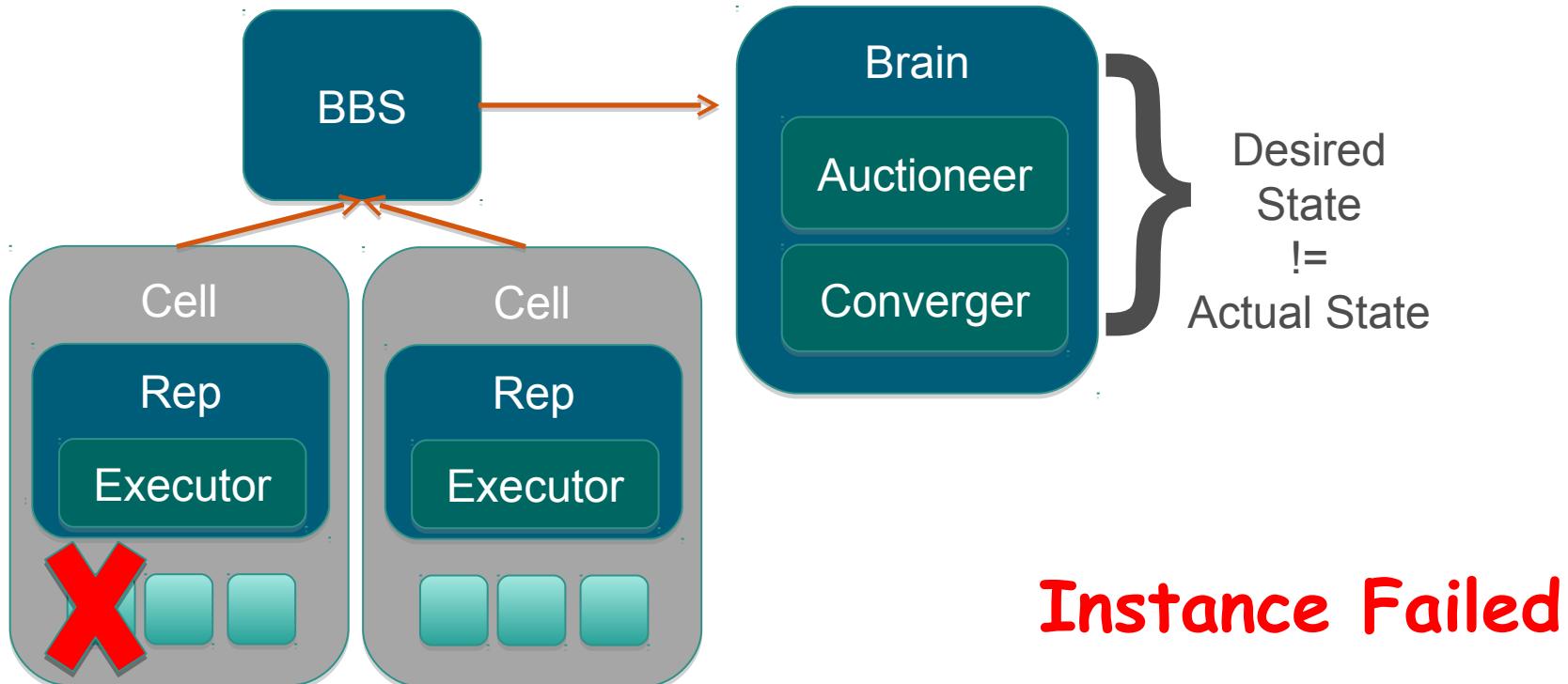
# Self Healing Application Instances

- Once running, failed application instances will be recreated



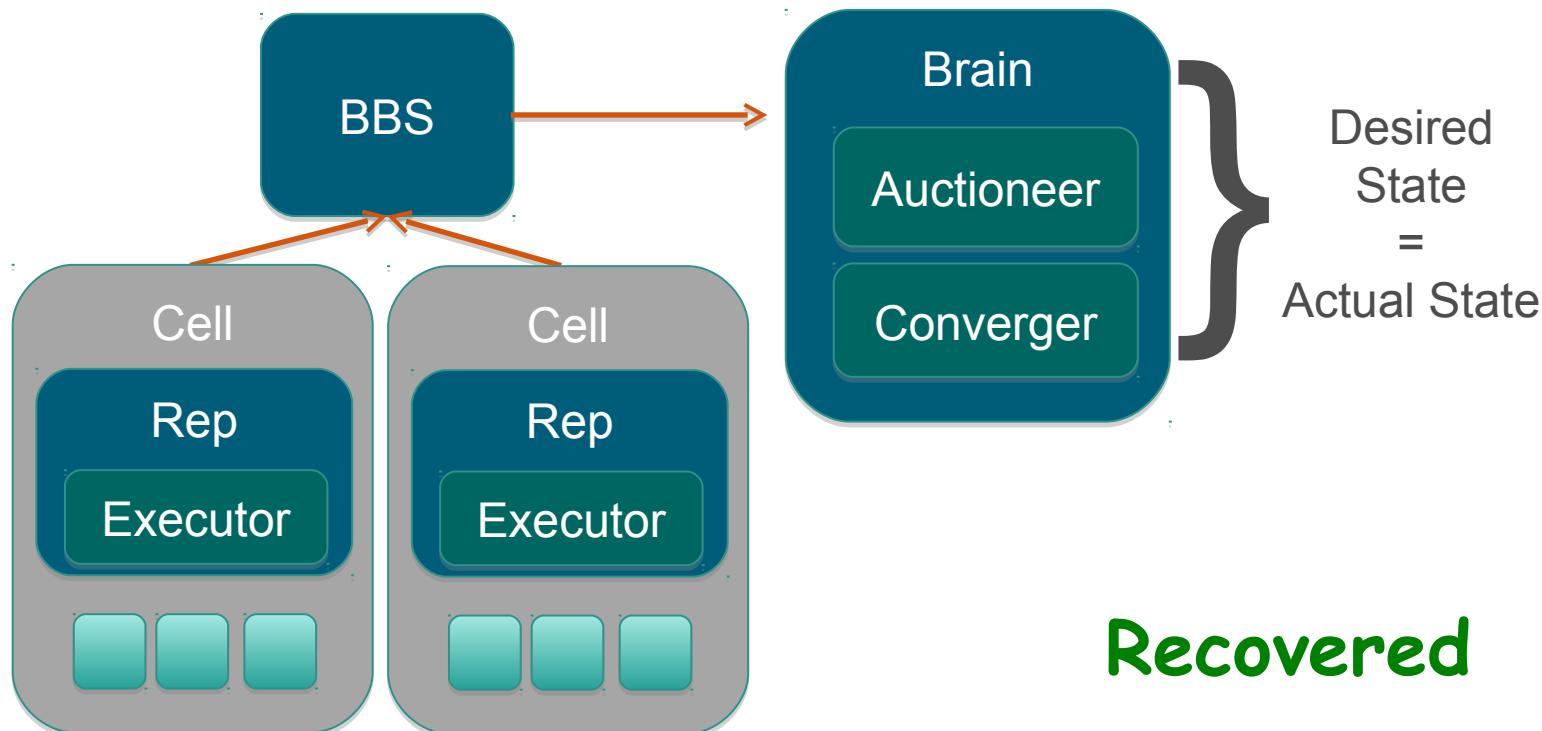
# Self Healing Application Instances

- Once running, failed application instances will be recreated



# Self Healing Application Instances

- Once running, failed application instances will be recreated



# Summary

- High Availability in Cloud Foundry
  - Availability Zones
  - BOSH VM monitoring
  - BOSH process monitoring
  - Cloud Foundry Application monitoring

# Advanced Topics

## Useful Features

Tasks, Volumes, More on routing

# Agenda

- Tasks
- File System as a Service
- Container to Container Routing
- TCP Routing

# What is a Task?

- Unlike a long-running process (LRP), tasks are not expected to run forever
  - A task runs once and is allowed to terminate
  - Application staging runs as a task

# What is a Task?

- Examples
  - Migrating a database
  - Sending an email
  - Running a batch job
  - Running a data processing script
  - Processing images
  - Optimizing a search index
  - Uploading data
  - Backing-up data
  - Downloading content

# Running a Task

- A tasks runs associated with an app
  - `cf run-task <app> <command> --name <task>`

```
$> cf run-task my-app "bin/rails db:migrate" --name my-task
Creating task for app my-app in org myorg / space development
as user@my.com...
OK
Task 1 has been submitted successfully for execution.
```

```
$> cf logs my-app --recent
2017-01-03T15:58:06.57-0800 [APP/TASK/my-task/0] OUT Creating
container
2017-01-03T15:58:08.45-0800 [APP/TASK/my-task/0] OUT
Successfully created container
2017-01-03T15:58:13.32-0800 ... CREATE TABLE ...
...
```

# Agenda

- Tasks
- **File System as a Service**
- Container to Container Routing
- TCP Routing

# Container File System is Temporary

- Many applications rely on the file system
  - Makes porting existing applications to PCF hard
- File System as a Service
  - A file-system independent of any container
  - Out-lives any container
  - NFS mounted storage

# Marketplace Service: nfs

- If configured for your PCF installation
  - *Volume Services*
  - nfs service in marketplace
- Create a service in usual way
- Use **VCAP\_SERVICES** to find volume

```
$ cf marketplace
service    plans          description
nfs        Existing      Service for connecting to NFS volumes
```

# Create and Bind Service

- Create
  - Use `-c` option to specify volume to use

```
$> cf create-service nfs Existing nfs_service_instance  
      -c '{"share": "10.10.10.10/export/myshare"}'
```

- Bind
  - Use `-c` option to specify GID and UID

```
$> cf bind-service my-app nfs_service_instance  
      -c '{"uid": "1000", "gid": "1000", "mount": "/var/volumel"}'  
...  
$> cf restage my-app  
...
```

# Example VCAP\_SERVICES

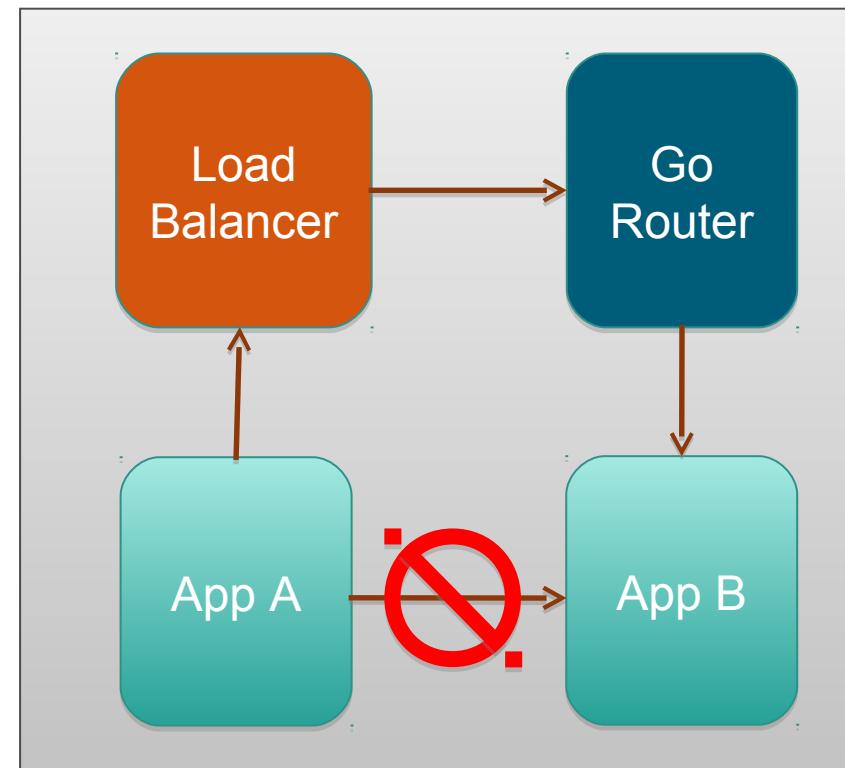
```
"nfs": [ {  
    "credentials": {},  
    "label": "nfs",  
    "name": "nfs_service_instance",  
    "plan": "Existing",  
    "provider": null,  
    "syslog_drain_url": null,  
    "tags": [ "nfs" ],  
    "volume_mounts": [  
        {  
            "container_dir":  
                "/var/vcap/data/153e3c4b-1151-4cf7-b311-948dd77fce64",  
            "device_type": "shared",  
            "mode": "rw"  
        }  
    ]  
} ]
```

# Agenda

- Tasks
- File System as a Service
- **Container to Container Routing**
- TCP Routing

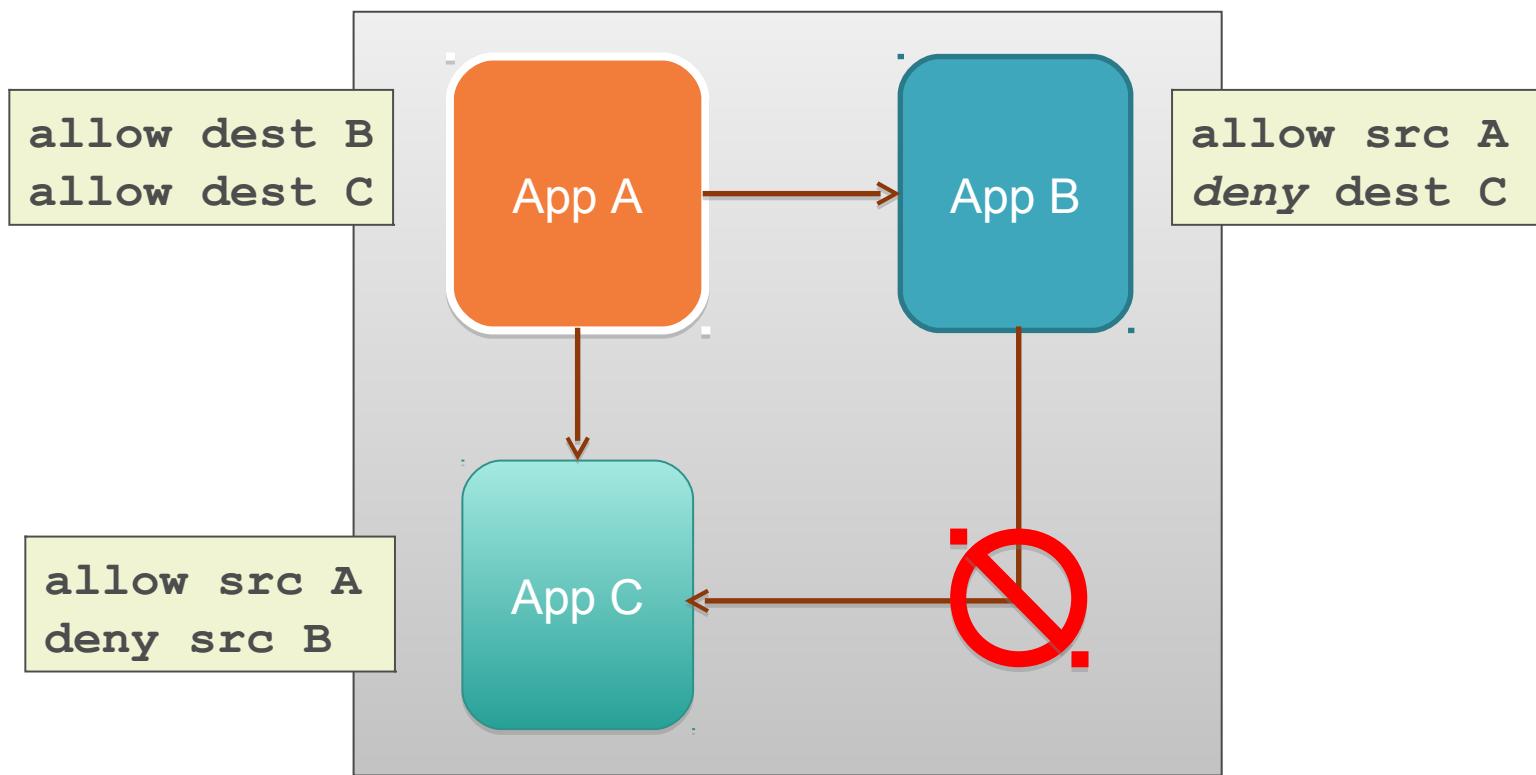
# Without C-C Routing

- Applications in PCF may need to talk to each other
  - Microservices are a common example
- The problem
  - Applications can only use public URL
  - Requests go via PCF's *GoRouter*



# With C-C Routing – 1

- Applications talk directly, once allowed



# With C-C Routing – 2

- Benefits
  - Internal services no longer need public routes
  - Direct communication faster
- Implication
  - Client-side load-balancing
    - Calling app will need to load-balance if multiple instances of called app
  - *Example:* Java applications
    - Netflix *Ribbon*, supported by Spring Cloud
  - *Example:* node.js applications
    - Resilient: <https://www.npmjs.com/package/resilient>

# Enabling C-C Routing

- Must be enabled globally by your PCF Ops first
- Download and install *network-policy-plugin* for cf
- Specify desired access

```
$> cf install-plugin ~/Downloads/network-policy-plugin  
...  
$> cf allow-access frontend backend -protocol tcp -port 8080  
...
```

# Agenda

- Tasks
- File System as a Service
- Container to Container Routing
- **TCP Routing**

# TCP Routing

- Support non-HTTP protocols
  - Internet of Things (such as MQTT, AMQP)
  - Legacy workloads
  - Non-persistent TCP services (*example*: Redis for caching)
  - BYO Container!
- New use cases for *existing* workloads
  - Terminate TLS at your app
  - Ensure incoming request is only decrypted when it reaches your application

# What Domains Are Available?

- Use `cf domains`
  - Look for domain type `tcp`

```
>$ cf domains
Getting domains in org pivotaledu as user@my.com...
name          status    type
cfapps.io    shared
cf-tcpapps.io  shared   tcp  ←
```

# Assigning a Route to an Application

- Push your app and specify its TCP domain

```
>$ cf domains
Getting domains in org pivotaledu as user@my.com...
name          status   type
cfapps.io     shared
cf-tcpapps.io shared   tcp

>$ cf push myapp -d cf-tcpapps.io --random-route
Binding ...cf-tcpapps.io:60010 to myapp...
OK
```



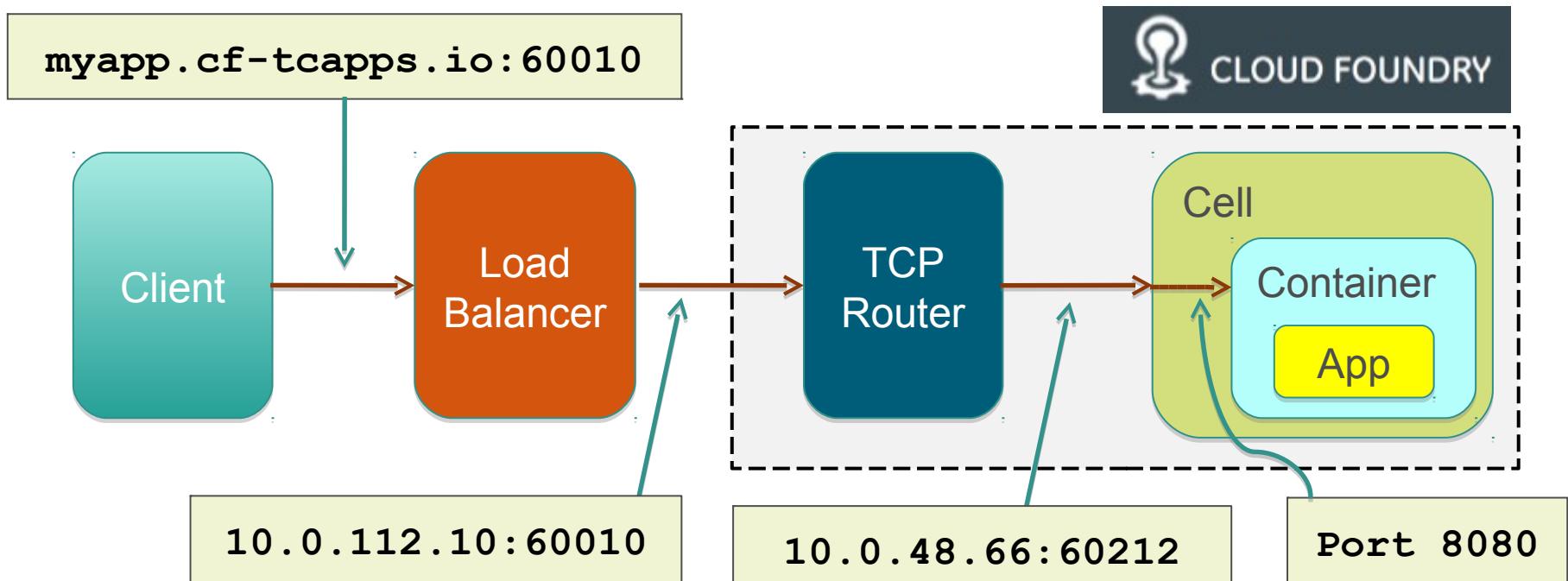
*random port*

- Or use map-route

```
$> cf map-route myapp cf-tcpapps.io --port 60010
Creating route cf-tcpapps.io:60600 for org...
```

# How it Works

- TCP Router makes routing decision based on port request is received on
  - Your app *must* listen on port 8080



# What We Have Learned

- You should now be able to
  - Run a Task
  - Describe File System as a Service
  - Explain Container to Container Routing
  - Push an application using TCP Routing

# Lab

## Running a Task in Cloud Foundry

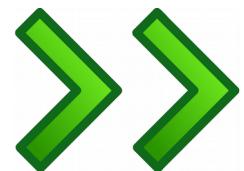
# Finishing Up

Course Completed

What's next?

# What's Next

- Congratulations, we've finished the course
- What to do next?
  - Certification
  - Other courses
  - Resources
  - Evaluation
- Check-out optional sections on ...



# Certification

- Computer-based exam
  - 50 multiple-choice questions
  - 90 minutes
  - Passing score: 76% (38 questions answered successfully)
- Preparation
  - Review all the slides
  - Redo the labs

# Certification: Questions

## Sample question

- Statements
  - a. Cloud Foundry is open-source and freely available
  - b. Pivotal Cloud Foundry is a commercial product
  - c. Cloud Foundry runs on a variety of Infrastructures
- Pick the correct response:
  - 1. Only a. is correct
  - 2. Both a. and c. are correct
  - 3. All are correct
  - 4. None are correct

# Certification: Logistics

- Where?
  - Online at PSI (Innovative Exams)
    - <https://www.examslocal.com>
- How?
  - Purchase a certification voucher *when ready*
    - <https://pivotal.io/training/certification>
  - Register/sign-in and book an exam using the voucher
    - <http://it.psionline.com/exam-faqs/pivotal-faq>
  - Take the test from *any* location
- For more information: [education@pivotal.io](mailto:education@pivotal.io)



Voucher is valid for 3 months  
– *use it or lose it!*

# Other courses



- Many courses available
  - PCF Administration
  - Core Spring
  - Web Applications with Spring
  - Spring Boot Developer
  - Spring Cloud Services
  - Spring Cloud Data Flow
  - Gemfire, Rabbit MQ ...
- More details here:
  - <http://www.pivotal.io/training>

# Cloud Foundry Administrator



CLOUD FOUNDRY

- 5 day course covering
  - Application deployment to Cloud Foundry
    - Logging, scaling, blue-green deployments
  - “Day 1 Operations”
    - Installation of PCF Ops Manager and Elastic Runtime
    - Configuring users, roles, and quotas
    - Capturing and reading logs
  - “Day 2 Operations”
    - Backing up and restoring an installation
    - Using BOSH
    - Upgrading Ops Manager and tiles.

Pivotal™

# Core Spring



- Four day course covering
  - Application configuration using Java Configuration, XML and/or Annotations
  - How Spring works internally and makes use of Aspect Oriented Programming
  - Data persistence using JDBC and JPA
  - Declarative Transaction Management
  - Introduction to web-applications and Spring MVC
  - Building RESTful Servers
  - Spring Boot, Spring Cloud and Microservices

# Spring Web

- 4-day workshop
- Making the most of Spring in the web layer
  - Spring MVC using Spring Boot
  - REST using MVC and AJAX, CORS
  - Security of Web applications
  - Mock MVC testing framework
  - Spring Web Sockets
- Spring Web Application Developer certification

# Enterprise Spring



- Building loosely coupled event-driven architectures
  - Separate processing, communications & integration
- 4 day course covering
  - Tasks, Scheduling and Concurrency
  - Advanced transaction management
  - REST Web Services with Spring MVC
  - Spring Batch
  - Spring Integration
  - Data Ingestion, Transformation and Extractions

# Spring Boot Developer



- 2 day workshop
  - Getting started with Spring Boot
  - Spring Boot CLI
  - Configuration, auto-configuration and profiles
  - Web development and REST
  - Data Access: JDBC, JPA, Spring Data, NoSQL
  - Testing
  - Security, Messaging
  - Deployment, Metrics, Actuator
  - Microservices

# Spring Cloud Services

## Microservices With Spring



- 2 day course
  - Introduction to Spring Boot
    - Underpins all Spring Cloud projects
  - Pushing Applications to a PaaS
    - Using Pivotal Cloud Foundry
  - What are Microservices?
    - Architecting a microservices solution
  - Cloud infrastructure services and Netflix OSS
    - Service Configuration
    - Service Registration
    - Load-balancing and fault tolerance

# Pivotal Support Offerings

- Global organization provides 24x7 support
  - How to Register: <http://tinyurl.com/piv-support>
- Premium and Developer support offerings:
  - <http://www.pivotal.io/support/offering>
  - <http://www.pivotal.io/support/oss>
  - Both Pivotal App Suite *and* Open Source products
- Support Portal: <https://support.pivotal.io>
  - Community forums, Knowledge Base, Product documents



# Pivotal Consulting

- Custom consulting engagement?
  - Contact us to arrange it
    - <http://www.pivotal.io/contact/spring-support>
    - Even if you don't have a support contract!
- Pivotal Labs
  - Agile development experts
  - Assist with design, development and product management
    - <http://www.pivotal.io/agile>
    - <http://pivotallabs.com>



Pivotal™

# Resources



- CF docs can be found on three different sites, depending on the context:
  - <http://docs.cloudfoundry.org>
    - Open-source CF project docs
  - <http://docs.pivotal.io/pivotalcf>
    - Setting up and using Pivotal Cloud Foundry
  - <http://docs.run.pivotal.io>
    - Information about running on Pivotal Web Services
- Stack Overflow – Active Forums
  - <http://stackoverflow.com>

# Thank You!



- We hope you enjoyed the course
- Your course is registered with the Pivotal Academy
  - Please fill out the *evaluation form*
- Once you've done, login to *Pivotal Academy*
  - You can download your Attendance Certificate



CLOUD **FOUNDRY**

# Installation Overview

Basics of PCF Installation

Or take the *PCF Administration* course

Pivotal

# Overview

- After completing this lesson, you should:
  - Understand how **PCF** is installed
    - But not necessarily do it yourself
  - Be familiar with the *Ops Manager Console*
  - Understand how the Elastic Runtime and CF Services are installed as “*Tiles*”

# Roadmap

- Prerequisites
- Configuration
  - Installing Pivotal CF Ops Manager VM
  - Configuring Ops Manager Director
  - Installing Elastic Runtime Tile
  - Installing Service Tile(s)

# Prerequisites

- A supported IaaS
  - Private Cloud (on-premise)
    - RedHat Open Stack
    - VMware ESXi
  - Public Cloud
    - Amazon Web Services (AWS)
    - Google Cloud
    - Microsoft Azure





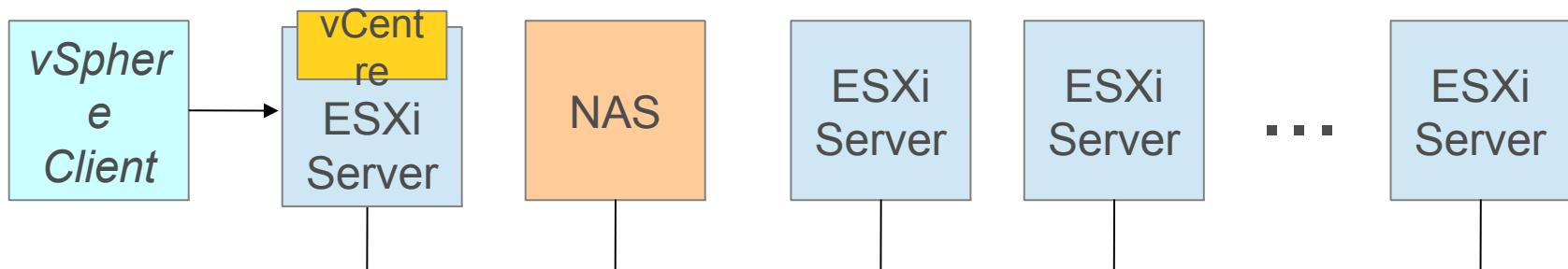
# Prerequisites - VMware

- A cluster running VMware virtualization software
  - Most common on-premise scenario
  - Alternative on-premise option
    - Red Hat Open Stack
    - Not covered here
- VMware experience is *very useful*
  - PCF setup via a prepackaged VM (appliance)



# Typical VMware Cluster

- The cluster needs the following nodes
  - One vCentre Server to co-ordinate the cluster
  - A vSphere client to configure the cluster (MS Windows)
    - Sends commands to vCentre
  - Network Storage (NAS)
  - ESXi servers (where vCenter & CF VMs actually run)



# Cluster Typical Resources

- vCentre
  - 6 CPUs, 6G RAM, 125G disk
- ESXi
  - 4+ CPUs, 16+ G RAM, 120+G
  - Scale proportionally
- vSphere
  - Windows only
  - 2-4 CPUs, 2G RAM, 20G disk
- NAS
  - 4 CPUs, 2G RAM, 1T disk

*Guidelines  
Only*

# Prerequisites – AWS



- An AWS account
- A key-pair for security (and the corresponding PEM file)
- A Cloud Formation script
  - Sets up most of the AWS resources you need
  - One ships with Pivotal's Elastic Runtime
    - See <https://network.pivotal.io>
- The ability to create more than the default limit of VMs
  - You just have to ask Amazon
  - Installing CF runs up over 20 new VMs
- A “Stack” in which to run the cloud formation script

# Installation Steps

- Five steps
  1. Get the software from the *Pivotal Network*
  2. Install Pivotal Ops Manager appliance (VM)
    - Only Pivotal CF has this
  3. Configure vSphere Tile - in Ops Manager Director
  4. Configure Elastic Runtime Tile (= CF)
  5. Add Services Tiles

These instructions show VMware, AWS and OpenStack are similar

# Pivotal Network for all Pivotal Software



**Buildpacks**



**Elastic Runtime**



**Ops Manager**



**Ops Metrics**

Release Download Files



Pivotal Cloud Foundry Ops Manager for AWS - 1.6.12.0  
38.4 KB 1.6.12.0



Pivotal Cloud Foundry Ops Manager for vSphere - 1.6.12.0  
2.02 GB 1.6.12.0



Pivotal Cloud Foundry Ops Manager for vCloud Air & vCD - 1.6.12.0  
2.09 GB 1.6.12.0



Pivotal Cloud Foundry Ops Manager for OpenStack - 1.6.12.0  
3.91 GB 1.6.12.0



Release Download Files

PCF 1.6 Clouformation script for AWS  
12.6 KB 1.0

PCF 1.6 Documentation  
23.3 MB 1.6

PCF Elastic Runtime  
3.43 GB 1.6.18

DiegoWindows 1.6.15  
6 Files

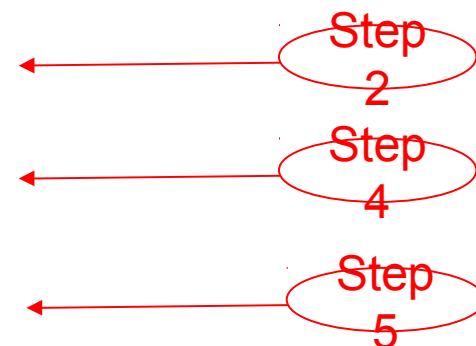
Pivotal™

# Step 1. Download Software



Pivotal Network

- Download from <http://network.pivotal.io>
  - See also <http://docs.pivotal.io>
- Typical files you might download
  - pcf-xxx.ova = Ops Manager VM
  - cf-xxx.pivotal = CF Elastic Runtime
  - p-mysql-xxx.pivotal = MySQL service
  - Documentation

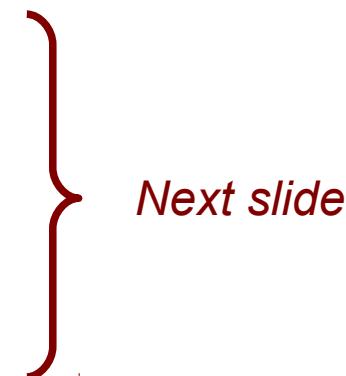


# Roadmap

- Prerequisites
- Configuration
  - **Installing Pivotal CF Ops Manager VM**
  - Configuring Ops Manager Director
  - Installing Elastic Runtime Tile
  - Installing Service Tile(s)

# Step 2. Install Ops Manager VM

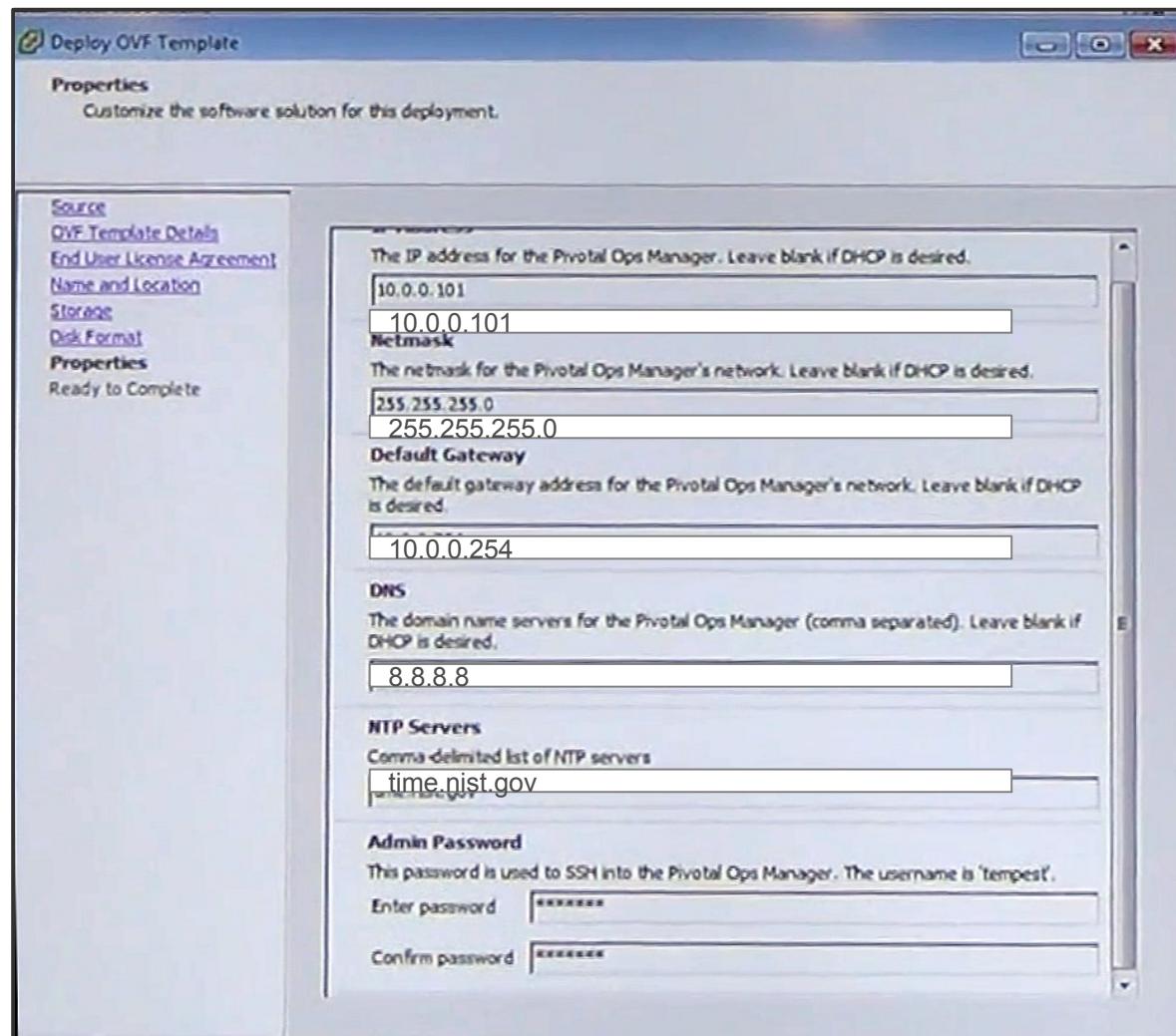
- Deploy OVA template
  - Use vSphere ... as you would to deploy any other VM
    - *Do not* use DHCP – requires a fixed IP address
  - Most important step, specify
    - IP Address of OPS Manager VM
    - Default gateway, DNS, NTP server
    - A username and password
      - allows *ssh* access to VM
      - for example to run BOSH commands
  - Deploy and start up the VM



AWS – Deploy a similar VM onto your Amazon cluster, configuration is similar

## 2b. Main Configuration Step

- A sys-admin/ops would *know* these values
  - IP address for Ops Mgr VM
  - Username and password to allow *ssh* into Ops Mgr VM



# Roadmap

- Prerequisites
- Configuration
  - Installing Pivotal CF Ops Manager VM
  - **Configuring Ops Manager Director**
  - Installing Elastic Runtime Tile
  - Installing Service Tile(s)

# What is the *Ops Manager*?

- A Ruby on Rails web-application
  - Pre-installed on the Ops Mgr VM
  - Starts up whenever the VM is started
- Provides a GUI for configuring a Pivotal CF “foundation”
  - Behind the scenes it configures and runs *BOSH*

## BOSH



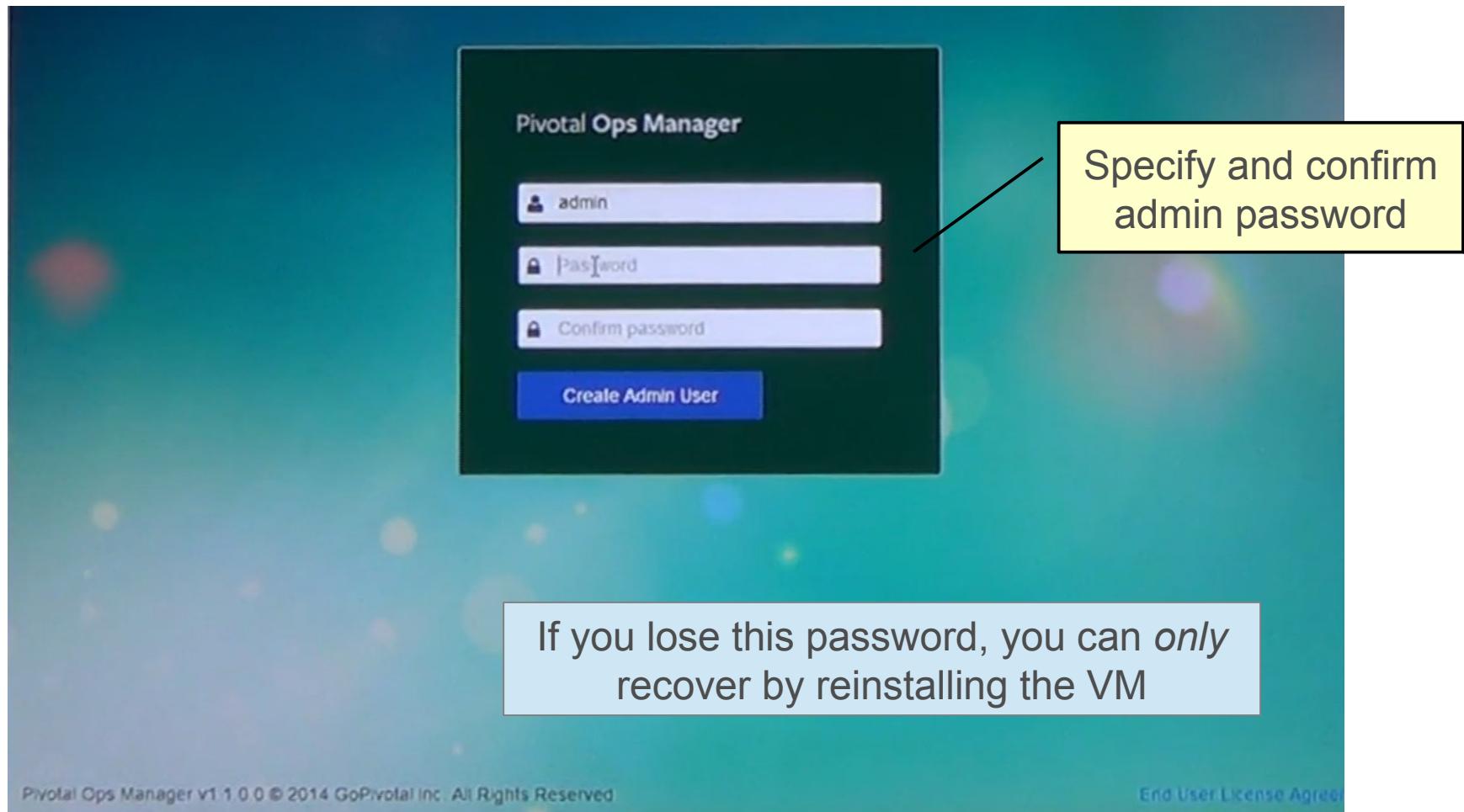
BOSH

- An open-source deployment tool used to run up the VMs and processes that make up CF
- IaaS independent: runs on vCenter, AWS ...
- Without Ops Mgr, OSS CF requires *manual* use of BOSH

# Step 3. Configure vSphere “Tile”

- Use browser to connect to IP address specified during OVA installation
  - Connects to Ops Manager Director (*a Rails web app*)
  - Asks you to setup an admin account: specify username and password you *won't forget*
- Most Pivotal CF Admin handled through Ops Manager
  - Not much need to use underlying infrastructure
    - Except to provide more underlying resources such as physical machines (vSphere) or storage

## 3a. Open URL in Browser



# Step 3. Configure vSphere “Tile”

- Installed features shown as “Tiles” in the main window
  - Color coding: **Orange**: not configured, **Blue**: configured, **Pivotal Green**: running
  - If anything goes wrong - click on Recent Install Logs to see what happened
  - There is a Revert option to return last good configuration
- First tile is always to configure Ops Mgr Director to use the underling IaaS
  - Here vSphere (Tiles exist for AWS, Open Stack ...)

# 3b. Browser View

The screenshot shows the Pivotal CF Ops Manager Installation Dashboard. At the top, there's a header with a logo, the text "Pivotal CF Ops Manager", and a user dropdown. A yellow callout box labeled "Revert option" points to a "Revert" link in the Pending Changes section. Another yellow callout box labeled "IaaS Tile" points to a large yellow rectangular area containing the text "IaaS Tile". A third yellow callout box labeled "Click here to see logs in case of errors" points to a link within the same yellow area. The dashboard itself has sections for Available Products (listing Ops Manager Director, Pivotal Elastic Runtime, and Pivotal CF Ops Metrics), Installation Dashboard (showing a tile for Ops Manager Director for VMware vSphere v1.3.4.0), Pending Changes (with an "Install" button and a "Recent Install Logs" link), and a footer with copyright and license information.

Pivotal CF Ops Manager

admin ▾

Deletion in progress.

Available Products

- Ops Manager Director  
No upgrades available
- Pivotal Elastic Runtime  
1.3.4.0 ▾
- Pivotal CF Ops Metrics  
1.3.3.0 ▾

Import a Product

Download Pivotal CF compatible products at [Pivotal Network](#)

Installation Dashboard

Ops Manager Director for  
**vmware**  
vSphere®  
v1.3.4.0

Pending Changes

INSTALL Ops Manager Director

Install

IaaS Tile

Recent Install Logs ▾

Click here to see logs in case of errors

Pivotal Ops Manager v1.3.4.0 © 2015 Pivotal Software, Inc. All Rights Reserved.

End User License Agreement

## 3c. Configure Ops Manager for vSphere

- vSphere knows about the Ops Manager VM
  - vSphere just installed and ran it
- But, *now* we need to tell Ops Manager about vSphere
  - Some configuration is green – defaults OK
  - Just need to setup the rest (orange)
- Also installs a single BOSH instance
  - Termed a *Micro-BOSH*
  - Enables Ops Manager to drive vSphere and create VMs
    - Uses BOSH to do all the hard work
    - Without Ops Manager, you'd have to use BOSH manually



# 3d. vCenter Credentials

The screenshot shows the Pivotal CF Ops Manager interface. The top navigation bar includes a logo, the title "Pivotal CF Ops Manager", and a user dropdown for "admin". Below the header, the "Installation Dashboard" is visible, with "Ops Manager Director" selected. A sidebar on the left lists several configuration steps: "vCenter Config" (selected), "Director Config", "Create Availability Zones", "Assign Availability Zones", "Create Networks", "Assign Networks", and "Resource Config" (checked). A yellow callout box labeled "orange = yet to be done" points to the sidebar items. The main content area is titled "vCenter Config" and contains fields for "vCenter IP Address\*" (set to "10.0.0.102") and "vCenter Credentials\*" (username "root" and password "\*\*\*\*\*"). It also includes fields for "Datacenter Name\*" and "Datastore Names (comma delimited)\*". A blue "Save" button is at the bottom. A yellow callout box labeled "vCenter IP Address" highlights the IP address field. A large yellow callout box on the right contains the text: "Real deployment: setup vCenter admin user with required access. For demos, just use root!".

Pivotal CF Ops Manager

admin

Installation Dashboard

Ops Manager Director

Settings Status Credentials Logs

vCenter Config

Director Config

Create Availability Zones

Assign Availability Zones

Create Networks

Assign Networks

Resource Config

orange = yet to be done

vCenter IP Address

10.0.0.102

vCenter Credentials\*

root

\*\*\*\*\*

Datacenter Name\*

Datastore Names (comma delimited)\*

Save

Real deployment: setup vCenter admin user with required access.  
For demos, just use root!

green = defaults OK

# 3e. Network Configuration

Installation Dashboard  
Ops Manager Director

Settings Status Credentials

vCenter Config Director Config Create Availability Zones Assign Availability Zones Create Networks Assign Networks Resource Config

green – now configured OK

Create Networks

Networks  
One or many IP ranges upon which your products will be deployed

LAN

Name\* LAN

vSphere Network Name\* LAN

Subnet (CIDR Range)\* 10.0.0.0/24

Excluded IP Ranges 10.0.0.1-10.0.0.102

DNS\* 10.111.111.222

Gateway\* 10.0.0.1

The network gateway IP to be used by VMs

Pivotal CF uses *all* addresses in subnet – *except* those excluded (already used by vSphere)

An ops person would know these values – because they setup The network originally

**No DHCP** is used – VMs given *explicit* addresses by BOSH

## 3g. Other Settings

- Availability Zones
  - Correspond to clusters in vCenter, supported directly in AWS, OpenStack
  - Provide redundancy for high-availability
- Networks
  - Multiple networks possible
  - Define IP addresses available for use by CF VMs
- Resource sizes
  - Amount of resources available to Ops Manager
  - Increase in big system to deploy faster

# 3h. Do the Install!

- Click Install – takes 15-20 mins to run

The screenshot shows the PCF Ops Manager interface. On the left, there's a sidebar with 'Available Products' and a 'Import a Product' button. The main area is the 'Installation Dashboard'. It features a card for 'Ops Manager Director' which includes the text 'Ops Manager Director for VMware vSphere® v1.4.0.0'. To the right of the card is a large blue button labeled 'Apply changes'. A red arrow points from a callout box at the bottom right towards this button. The callout box contains the text 'green – now configured OK'. In the top right corner of the dashboard, there's a 'Pending Changes' section with a single item: 'INSTALL Ops Manager Director'.

# Other IaaS

- An Ops Manager VM exists for
  - VMware, AWS, Microsoft Azure, Red Hat Open Stack and Google Cloud
  - Similar setup
- CF OSS *can* also be installed on any supported IaaS
  - But is harder to do: No *Ops Mgr*
    - You must configure it manually (BOSH scripting)



Windows Azure®



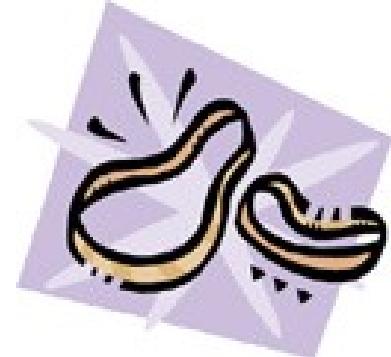
Google Cloud Platform

Pivotal™

# Roadmap

- Prerequisites
- Configuration
  - Installing Pivotal CF Ops Manager VM
  - Configuring Ops Manager Director
  - **Installing Elastic Runtime Tile**
  - Installing Service Tile(s)

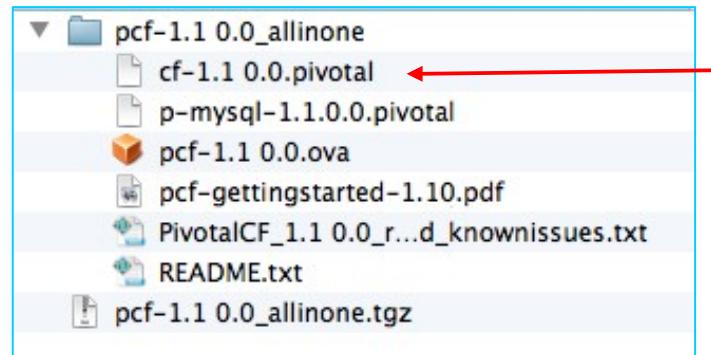
# Elastic Runtime



- A scalable runtime environment
  - This is what you actually deploy to
    - *To a developer, it **is** Cloud Foundry*
  - Uses the micro-BOSH to control the underlying IAAS
    - In our case: vSphere
- Operational control of Cloud Foundry is *really* management and configuration of the **Elastic Runtime**

# Step 4: Elastic Runtime Installation

- To install:
  - Click on “Import a Product” ①
  - Select “Elastic Runtime” = cf-xxx.pivotal
  - Click Add – Tile appears ②
  - Click on the Elastic Runtime tile to configure it



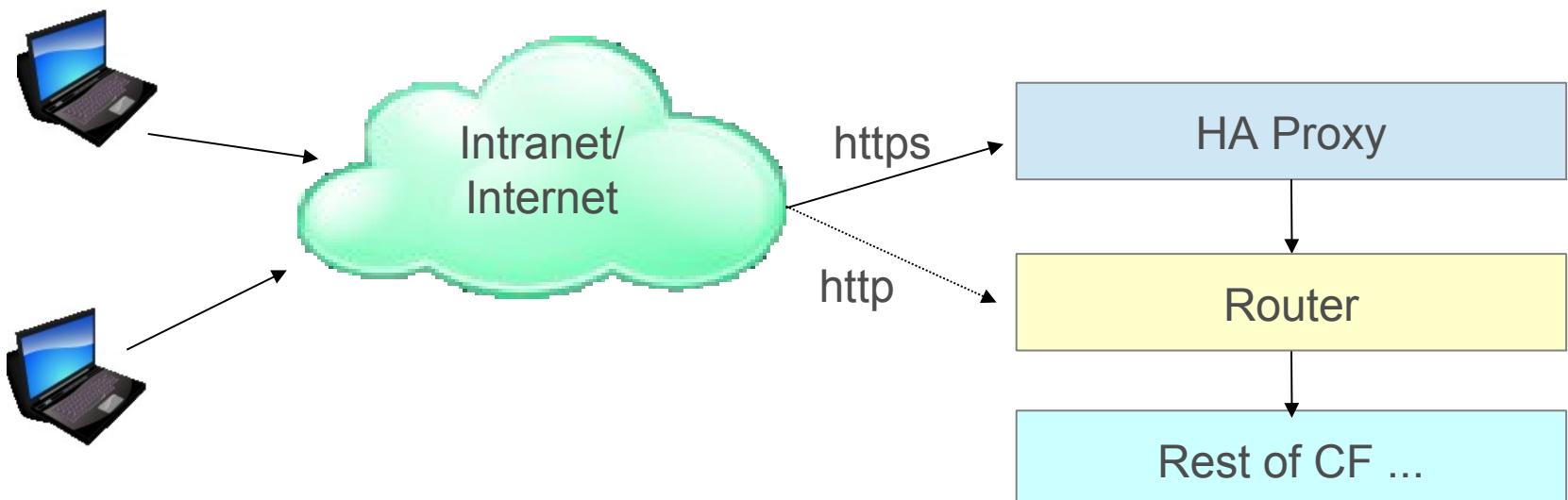
The screenshot shows the 'Available Products' section on the left and the 'Installation Dashboard' on the right. In the 'Available Products' section, the 'Pivotal Elastic Runtime' tile is highlighted with a red box and circled with a red number 1. In the 'Installation Dashboard', a tile for 'Operations Manager Director for VMware vSphere' is shown, and its 'Add' button is highlighted with a red box and circled with a red number 2.

1  
All services are added like this:  
MySQL, Rabbit/MQ ...  
– download .pivotal files from PivNet

Orange – not configured

# Accessing Your CF Instance

- Rest of company/outside world needs IP address to use
  - If public, setup DNS to point to that address
  - CF includes HA proxy load-balancer (or use your own)
    - HTTPS support (optional) via HA Proxy



# 4a. IPs and Ports – 1

PCF Ops Manager admin ▾

◀ Installation Dashboard

Pivotal Elastic Runtime

Settings Status Credentials Logs

Most options are green – ok defaults

Assign Networks

Assign Availability Zones

Root Filesystem

System Database Config

File Storage Config

IPs and Ports

MySQL Proxy Config

Cloud Controller

Router IPs

HAProxy IPs

Loggregator Port

10.0.0.121

Static address, visible to rest of company and/or outside world

Form continues on next slide ...

## 4a. IPs and Ports – 2

- Use real certificates if you have them, or generate self-signed

*... page continues from previous slide*

The screenshot shows the 'SSL Certificate' configuration section. On the left, a sidebar lists configuration items: External Endpoints (checked), SSO Config, LDAP Config, SMTP Config, Errands, Resource Config, and Stemcell. The main area contains two large text input fields for certificate and private key content, both starting with '-----BEGIN' and ending with '-----'. Below these fields is a button labeled 'Generate Self-Signed RSA Certificate'. To the right, a yellow callout box contains the text 'Click to auto-generate self-signed certs' with a cursor icon pointing towards the generate button. At the bottom, there is a checked checkbox for 'Trust Self-Signed Certificates' and a blue 'Save' button.

External Endpoints

SSO Config

LDAP Config

SMTP Config

Errands

Resource Config

Stemcell

SSL Certificate \*

```
-----BEGIN CERTIFICATE-----  
MIIDLjCCAhgAwIBAgIVAKYKJ8qMc5kyh3SEnqA  
Iet//ZeuMA0GCSqGSIb3DQEBr  
BQUAMD4XCZAJBgNVBAYTAiVTMRAwDgYDVQQK  
DAdQaXZvdGFsMR0wGwYDVQQDDBQa  
LJMweC5lZHUUucGl2b3Rhbc5pbzAeFw0xNTA2Mj  
AxMDUyMDMwEjEuOvNzA3MTkvMDUuA  
-----BEGIN RSA PRIVATE KEY-----  
MIIEpQIBAAKCAQEao+eEt2eakfyrlhurMgy3gre2a  
twwRDhi3Jur199/Q/PHqkvC  
nwuMMnzHSQDYzhBfxRZaaCvkRL2DDNp8-RJs1  
IA7AOcqKrkky-iBeTSBhDKP9tB  
BWZ0qGO0S5MCJ01Yb1C8ELLT1mzNPujA1VnK/  
A6cndrPUcRqavQwulQWIMM5V6O  
Generate Self-Signed RSA Certificate
```

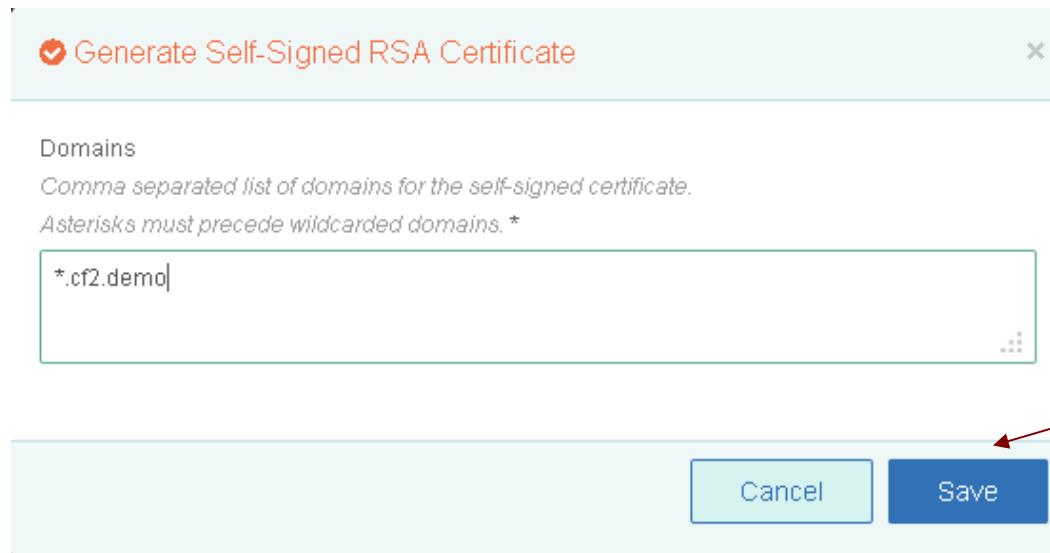
Trust Self-Signed Certificates

**Save**

**Click to auto-generate self-signed certs**

## 4b. Self-signed Certificates

- In popup window ...
  - Certificates should be generated for \*.[your-domain]



If using self-signed certificates:

```
cf login --skip-ssl-validation . . .
```

# 4c. Specify Domains

- Cloud Controller Tab
  - System and App domains
  - Same used for the HAProxy certificates, without the \* wildcard

- ✓ Assign Networks
- ✓ Assign Availability Zones
- ✓ Root Filesystem
- ✓ System Database Config
- ✓ File Storage Config
- ✓ IPs and Ports
- ✓ MySQL Proxy Config
- Cloud Controller

Coordinates Pivotal CF Elastic Runtime ap

---

System Domain \*

Apps Domain \*

Cloud Controller DB Encryption Key

Maximum File Upload Size (MB) ( min: 1024, max: 2048 ) \*

- System domain is used to target and push apps to Elastic Runtime
  - In this case: `cf login -a api.cf2demo`
- Application domain is used to serve applications
  - `cf push spring-music` would create app at *spring-music.cf2.demo*

## 4d. Many Other Options

- Additional items
  - Networks, High Availability Zones
    - If you have a choice, tell ER which ones to use
  - SSO, LDAP
    - Used with SSO appliances and Active Directory integration
  - Mail Setup
    - Optional SMTP/email configuration to send email to users
      - for example: invites to use system, error/system messages
  - Errands
    - Tasks that run at end of Install – including creating the App Manager console

## 4e. Resource Sizing

- Configure how many instances of each component should be deployed
  - Can normally accept the defaults
- **Important:**
  - Ops Manager console is still available *after* CF is installed
    - Normal way for ops to expand/configure system
    - Next slide ...

## Resource Config

JOB	INSTANCES	PERSISTENT DISK TYPE	VM TYPE
Consul	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
NATS	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
etcd	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Diego BBS	Automatic: 1	Automatic: 1 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
File Storage	Automatic: 1	Automatic: 100 GB	Automatic: medium.mem (cpu: 1, ram: 8 GB, disk: 100 GB)
MySQL Proxy	Automatic: 1	None	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)
MySQL Server	Automatic: 1	Automatic: 100 GB	Automatic: large.disk (cpu: 2, ram: 8 GB, disk: 100 GB)
Backup Prepare Node	0	Automatic: 200 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)
Cloud Controller Database (Postgres)	Automatic: 0	Automatic: 2 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
UAA Database (Postgres)	Automatic: 0	Automatic: 10 GB	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
UAA	Automatic: 1	None	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 10 GB)
Cloud Controller	Automatic: 1	Automatic: 1 GB	Automatic: medium.disk (cpu: 2, ram: 4 GB, disk: 10 GB)
HAProxy	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Clock Global	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Cloud Controller Worker	Automatic: 1	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Collector	Automatic: 0	None	Automatic: micro (cpu: 1, ram: 1 GB, disk: 2 GB)
Diego Brain	Automatic: 1	Automatic: 1 GB	Automatic: small (cpu: 1, ram: 2 GB, disk: 4 GB)
Diego Cell	Automatic: 3	None	Automatic: xlarge.disk (cpu: 4, ram: 16 GB, disk: 100 GB)

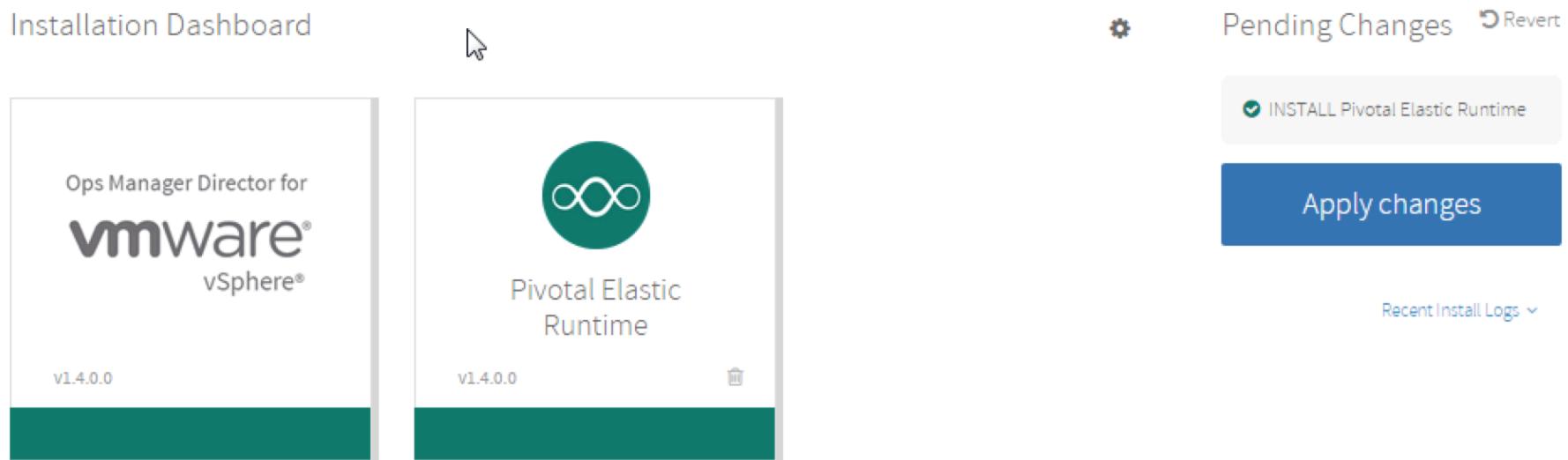
More Cells  
=  
More App Instances

Diego Cell

Pivotal

## 4f. Install Elastic Runtime

- Click “Apply Changes”
  - This takes a while, up to 3 hours



# 4f. Installation Takes Time!

PIVOTAL  
Operations Manager

Installation Dashboard

## Installation in Progress

4%

- Setting Micro BOSH deployment manifest
- Installing Micro BOSH**
- Checking Micro BOSH status
- Logging into director
- Creating director user
- Removing director bootstrap user
- Targeting director for Pivotal Elastic Runtime
- Logging into director for Pivotal Elastic Runtime
- Uploading release for Pivotal Elastic Runtime
- Uploading stemcell for Pivotal Elastic Runtime
- Setting installation manifest for Pivotal Elastic Runtime

- Many steps
- Many VMs created

- Installing Pivotal Elastic Runtime
- Checking Pivotal Elastic Runtime
- Verifying app push
- Pushing Console app
- Verifying Console app
- Targeting director for Pivotal MySQL Dev
- Logging into director for Pivotal MySQL Dev
- Uploading release for Pivotal MySQL Dev
- Uploading stemcell for Pivotal MySQL Dev
- Setting installation manifest for Pivotal MySQL Dev
- Installing Pivotal MySQL Dev
- Checking Pivotal MySQL Dev
- Verifying MySQL service
- Storing settings in file system

Each item listed is a *BOSH task*

# Checking Installation

## Login to Pivotal CF App Manager Console

- Navigate to [http://console.<your\\_domain>](http://console.<your_domain>)

The screenshot shows the Pivotal CF App Manager Console interface. The top navigation bar includes the Pivotal CF logo, the organization name "system", and a user dropdown set to "admin". The left sidebar lists organizational units: "ORG" (selected), "SPACES", "Docs", "Support", and "Tools". Under "SPACES", three spaces are listed: "app-usage-service", "apps\_manager", and "autoscaling". The main content area displays the "ORG" dashboard for "system". It shows a quota of 5% (5 GB of 100 GB Limit). Below this, it indicates 3 Spaces, 1 Domain, and 2 Members. Three cards provide detailed information about each space:

- SPACE app-usage-service:** 3% of Org Quota. Contains 3 Apps and 0 Services.
- SPACE apps\_manager:** 1% of Org Quota. Contains 1 App and 0 Services.
- SPACE autoscaling:** Contains 1 App and 0 Services.

# Installing Services

- Many pre-packaged services for Pivotal CF
  - Supplied as **.pivotal** files at Pivotal Network
    - See <https://network.pivotal.io>

 <p><b>MongoDB for Pivotal CF</b> MongoDB Data Store</p>	 <p><b>MySQL for Pivotal CF</b> MySQL database-as-a-service for Cloud Foundry applications</p>	 <p><b>Pivotal CF</b> Rapidly deploy and scale applications on private clouds W...</p>
 <p><b>Pivotal CF RabbitMQ Service</b> Give your applications a common platform to safely</p>	 <p><b>Pivotal HD for Pivotal CF®</b> Gain insight from the massive data captured by apps, syst...</p>	 <p><b>Redis for Pivotal CF</b> Cloud Foundry Redis service for application development a...</p>

# Installing Services – MySQL



- For example: to install MySQL
  - Go back to Operations Manager home-page
  - Click “Add New Product”
  - Select MySQL
    - `p-mysql-1.10.12.pivotal` (your version may be different)
  - Appears as new Tile
  - Review options and install
    - Just like we did for Elastic Runtime
  - MySQL now appears as a service in the Pivotal CF Marketplace



# Typical Ops Manager View After Installation

**PCF Ops Manager**

Available Products < Installation Dashboard 

Ops Manager Director No upgrades available	Ops Manager Director for <b>vmware</b> vSphere® v1.5.2.0	Pivotal Elastic Runtime v1.5.2.0	RabbitMQ v1.5.1
Pivotal Elastic Runtime No upgrades available			
RabbitMQ No upgrades available			
AppDynamics Service Broker No upgrades available	AppDynamics Service Broker v1.0	MySQL for Pivotal Cloud Foundry v1.7.0.4	Pivotal JMX Bridge (Ops Metrics) v1.4.1.0
Pivotal JMX Bridge No upgrades available			
MySQL for Pivotal Cloud Foundry No upgrades available			

**Import a Product**

Download PCF compatible products at [Pivotal Network](#)

Pivotal

# Summary

- After completing this lesson, you should have learnt:
  - How to setup Operations Manager VM
  - How to Install the CF Elastic Runtime
  - How to install other services