



CLOUD **FOUNDRY**

# Cloud Foundry Developer

Hands-On Workshop

Application Deployment using Pivotal Cloud

Version 2.0.a

Pivotal

# Copyright Notice

Copyright © 2018 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

- Course Orientation
- CF Motivations
- Basic Definitions
- Tech Overview
- Core Themes
- Logging & Metrics
- Resiliency and High Availability
- Services

1

# Course Agenda: Day 2

- 12 Factor Applications
- Log Drain Review
- App Execution & Security Groups
- Staging & Running
- Microservices
- Route Services
- Deploying Docker Containers to PAS



# Course Agenda: Day 3

- TCP Routing
- UAA & OAuth2
- Cloud Native
- Avoiding Cascading Failure
- Distributed Tracing
- App Lifecycle and Spaces
- Avoiding Non-Breaking Changes
- CI/CD Automation

3

# 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 Dell-EMC, VMware and General Electric
  - Formed in 2013, public company since 2018
- Several products supporting cloud deployment
  - Cloud Foundry itself
  - Spring Projects
  - Gemfire Distributed Data Grid
    - Pivotal Cloud Cache
  - RabbitMQ Messaging (AMQP)

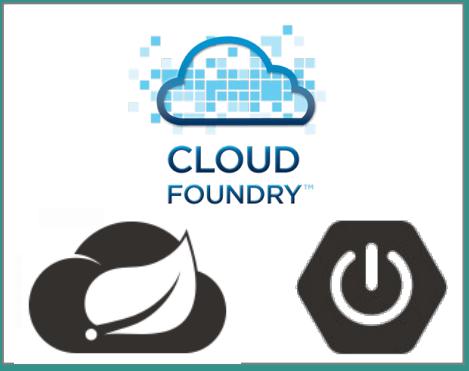


Pivotal™

# 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**

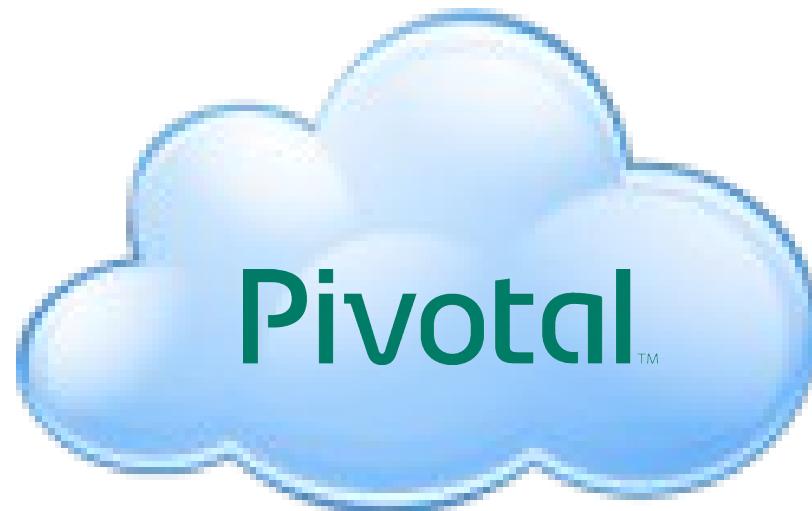
*Promoting agile development*

Pivotal™

# Covered in this section

- Agenda
- About Pivotal

Let's get on with the course..!



Pivotal™



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™

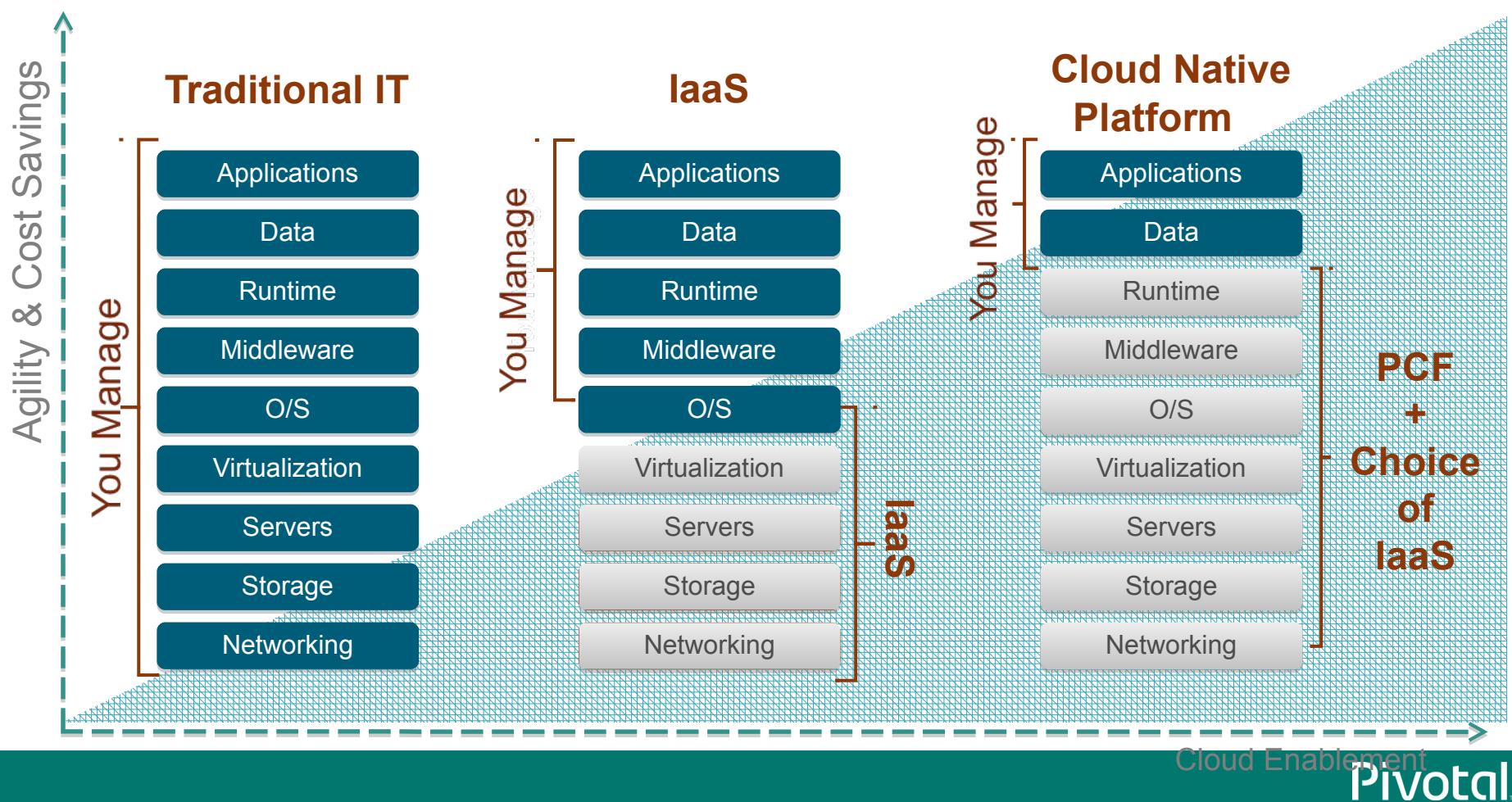
# Types of Cloud Computing: SaaS

- 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 Disrupting Industries



Square



U B E R



NETFLIX



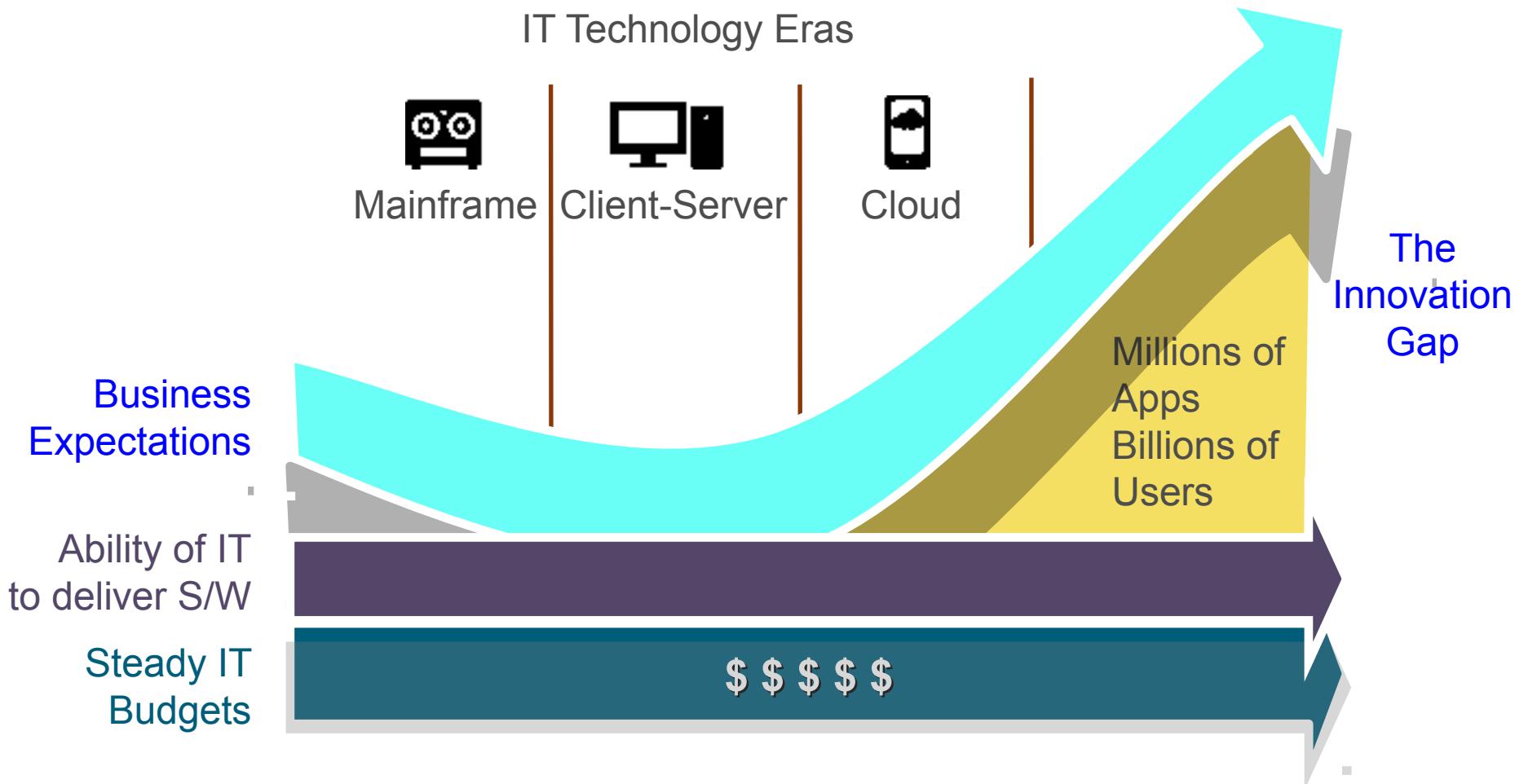
TESLA

Pivotal™

# Industry Trends

- **Decentralized I.T.**
  - Workloads in more clouds, localized app ownership
  - Self-service, focus on business value.
- **Accelerating transformation**
  - Companies moving faster towards software-driven future
  - Need new skills and more diverse runtimes
- **Security as a CEO priority**
  - Threats, new and old, have major business impact
  - Passive security isn't enough.
- **Reactive architecture takes hold**
  - Event-driven applications drive real-time insight & behavior

# Business Needs Exceed IT Expectations



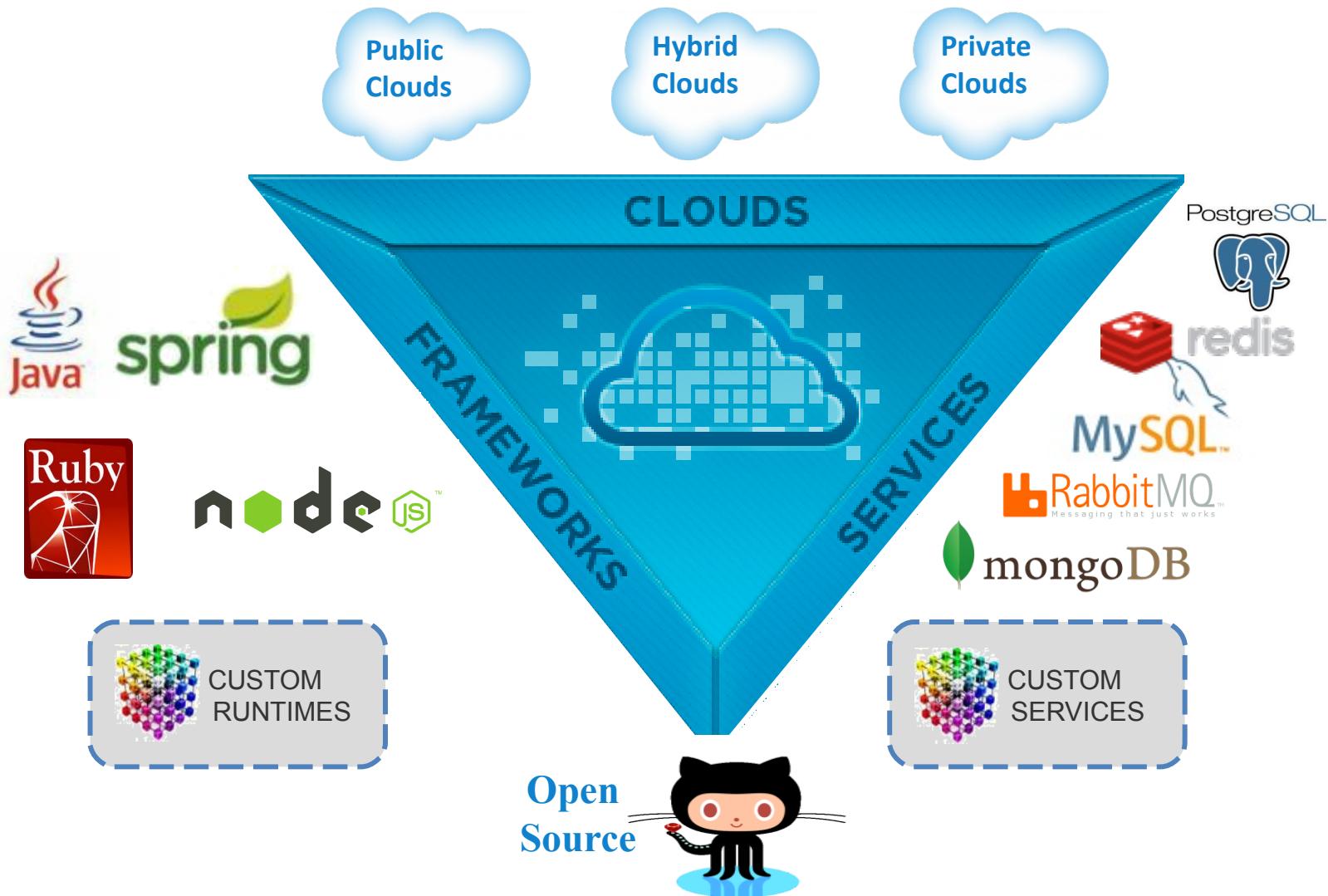
Gartner, 2013: "Hunting and Harvesting in a Digital World: The 2013 CIO Agenda"

Pivotal™

# What is Cloud Foundry?

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

# Cloud Foundry – The Open PaaS



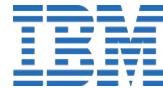
# Why Cloud Foundry?

- 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

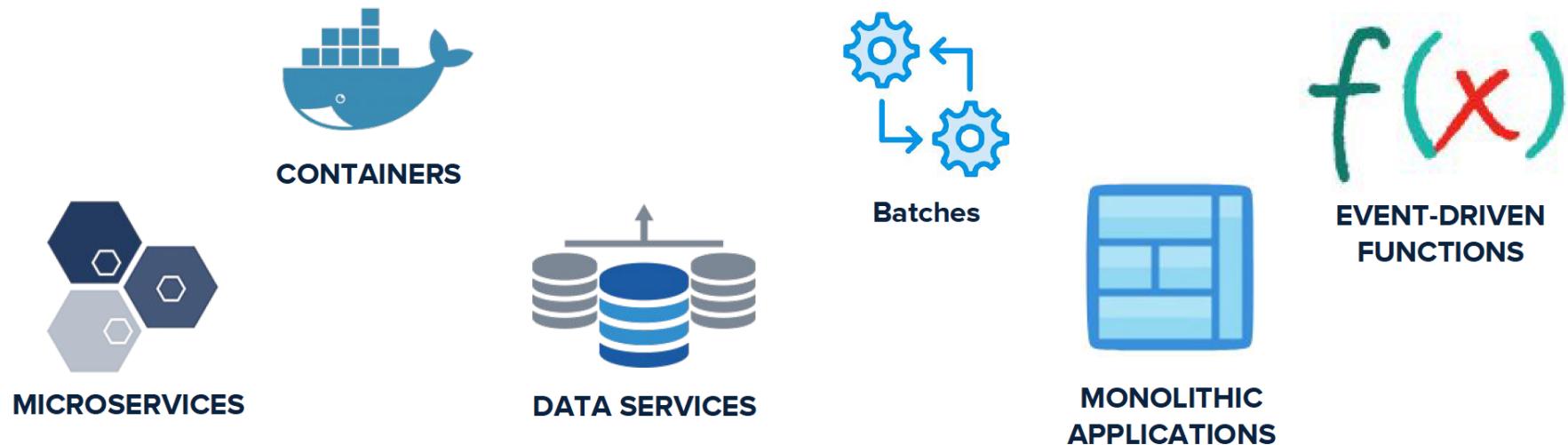
- Largest Chinese Internet search site
  - One billion page views per day
  - Ad-Words facility powered by Cloud Foundry



# What is Pivotal Cloud Foundry?

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

# PCF V2 – Supporting Multiple Workloads



Container  
Orchestrator  
(CaaS)

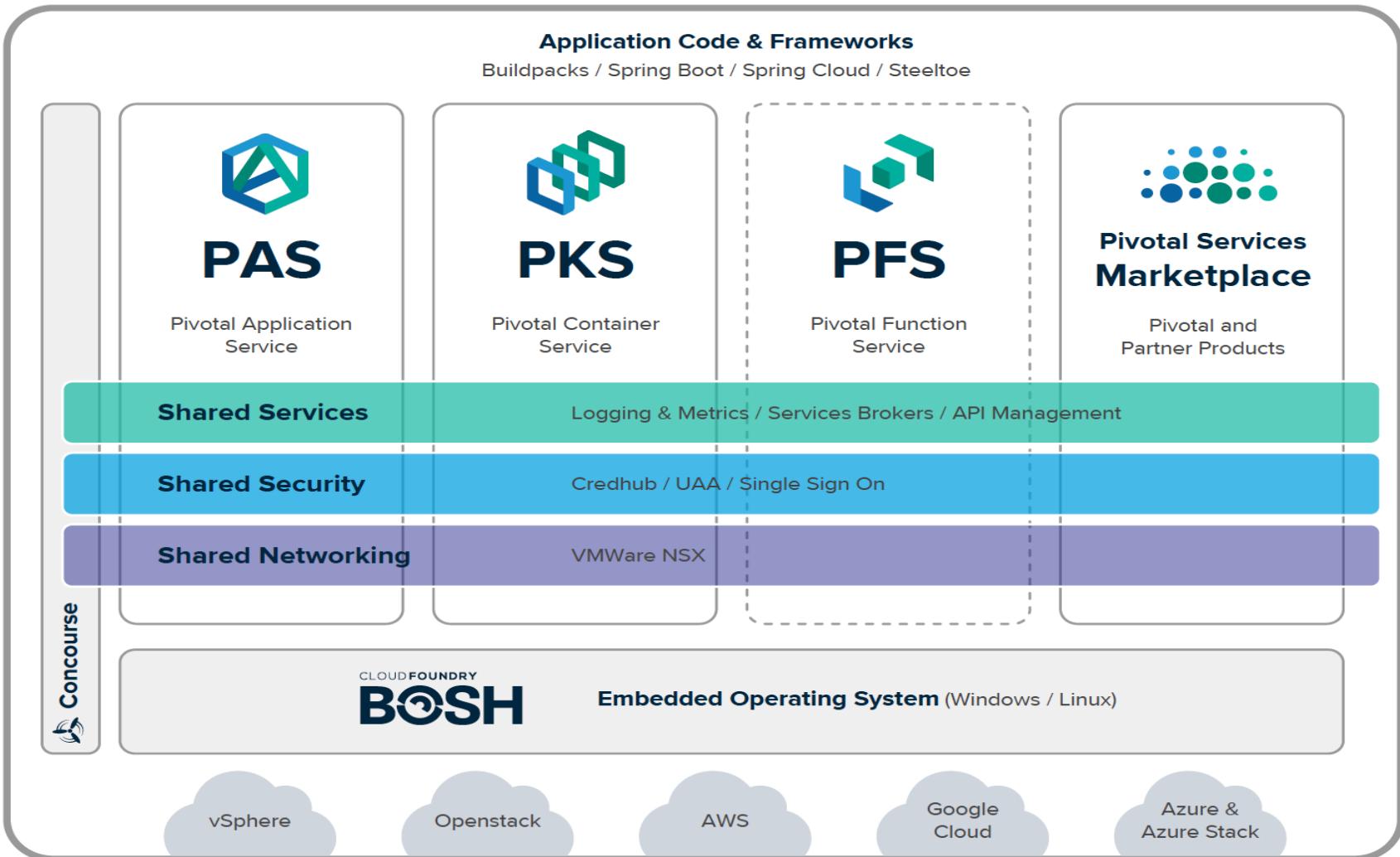
Application  
Platform  
(PaaS)

Serverless  
Functions  
(FaaS)

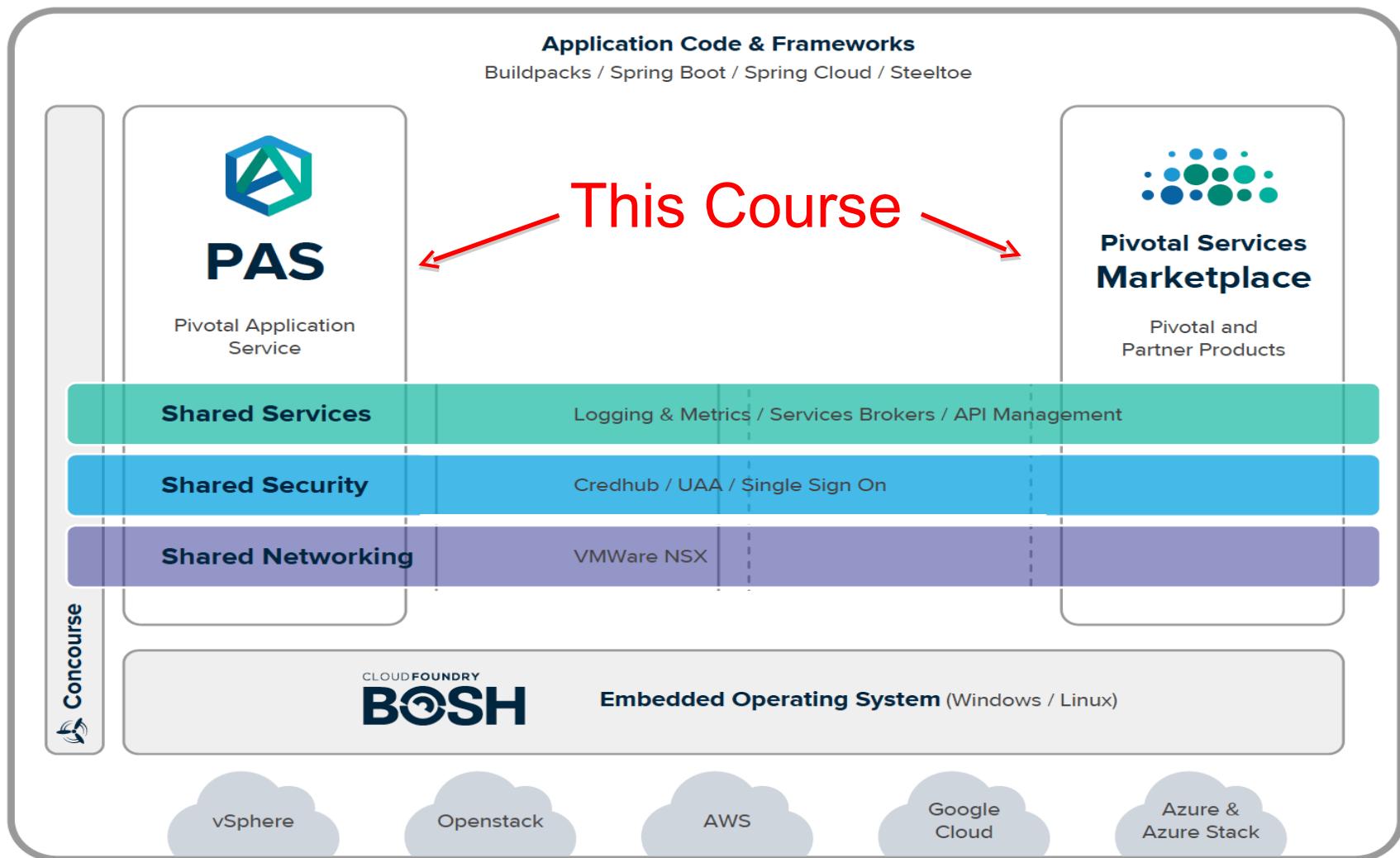
IaaS

Pivotal™

# Pivotal Cloud Foundry V2

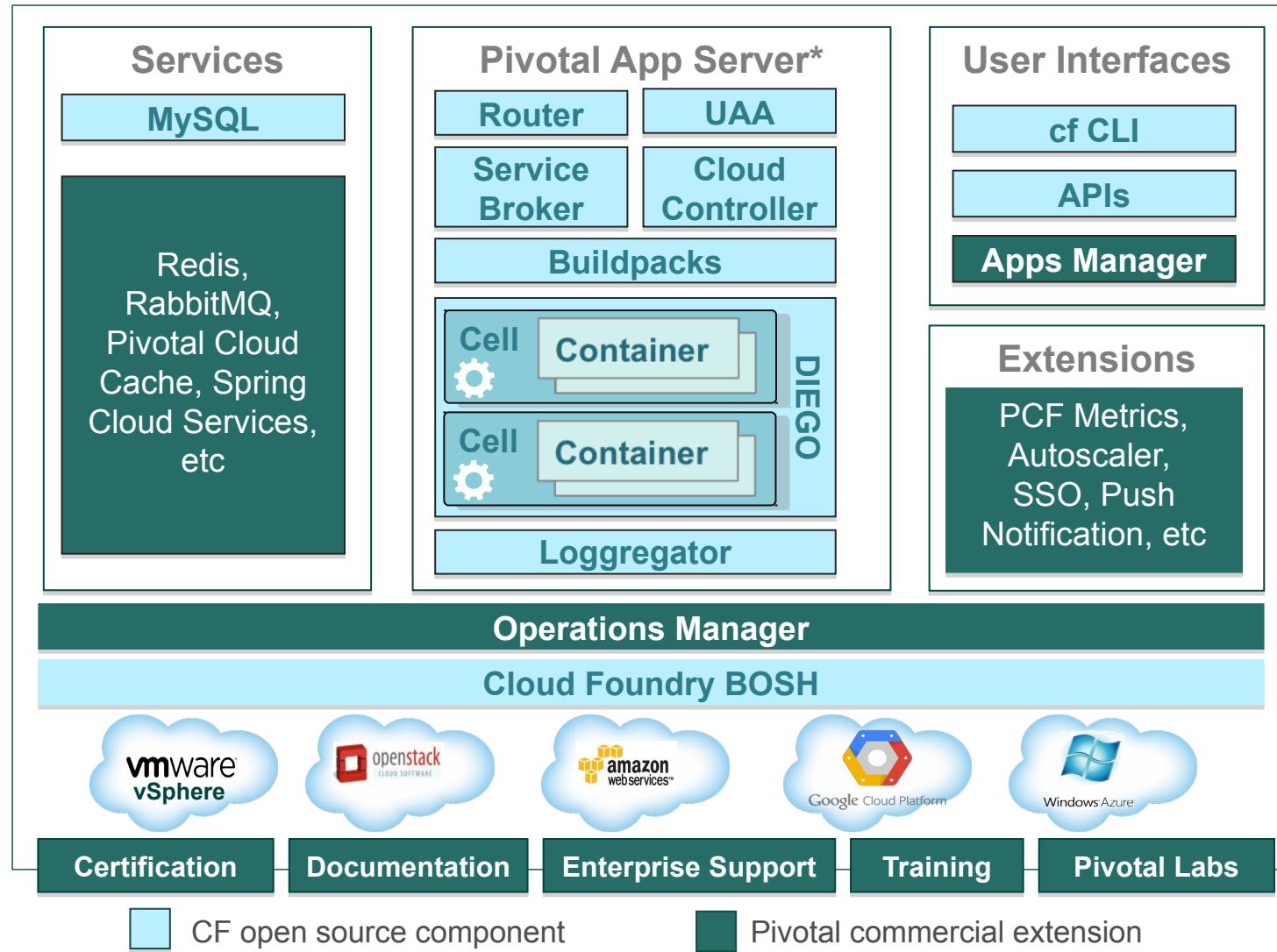


# Pivotal Cloud Foundry V2



\* Formerly Elastic Runtime

# App Deployment 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

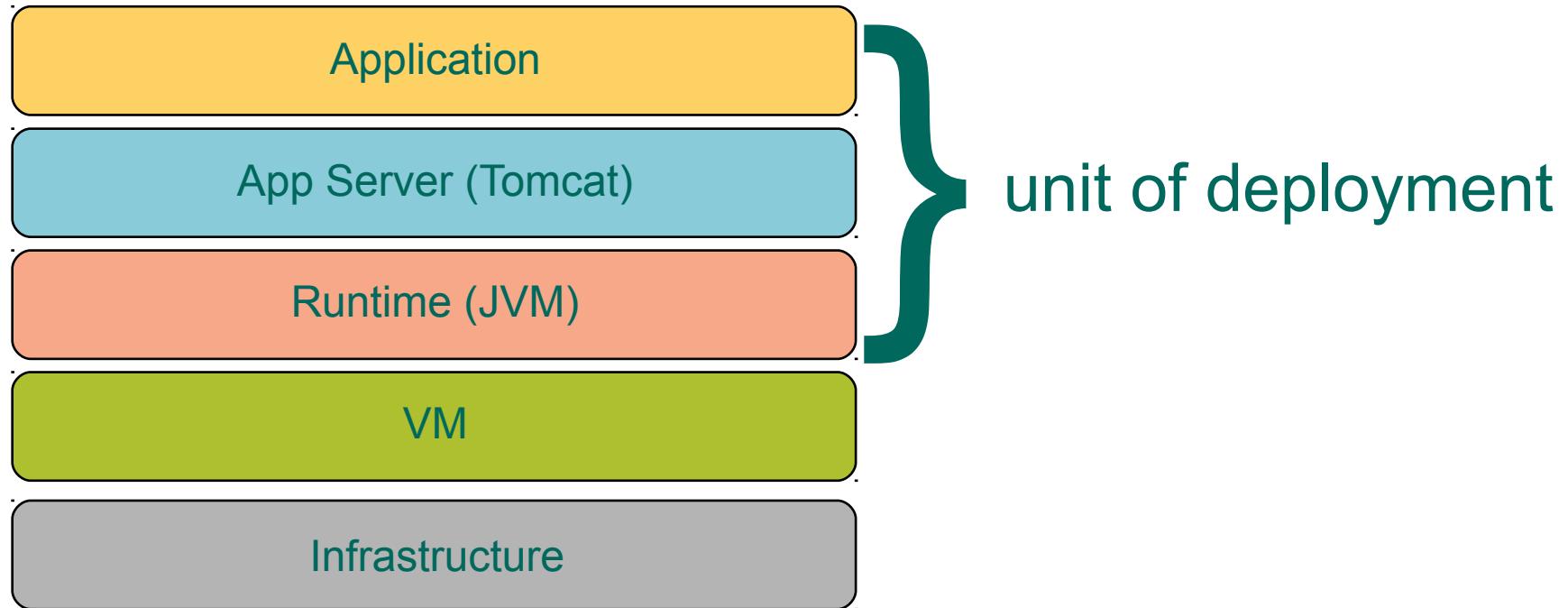
- ◆ Commercial product
- ◆ Run private PaaS in-house, or
- ◆ Create public PaaS (ISP managed)
- ◆ Sophisticated Web Console (App Mgr)
- ◆ Additional Tools (Ops Mgr)
- ◆ Less hassle
- ◆ Support offered

# Pivotal Application Service (PaaS)

- **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

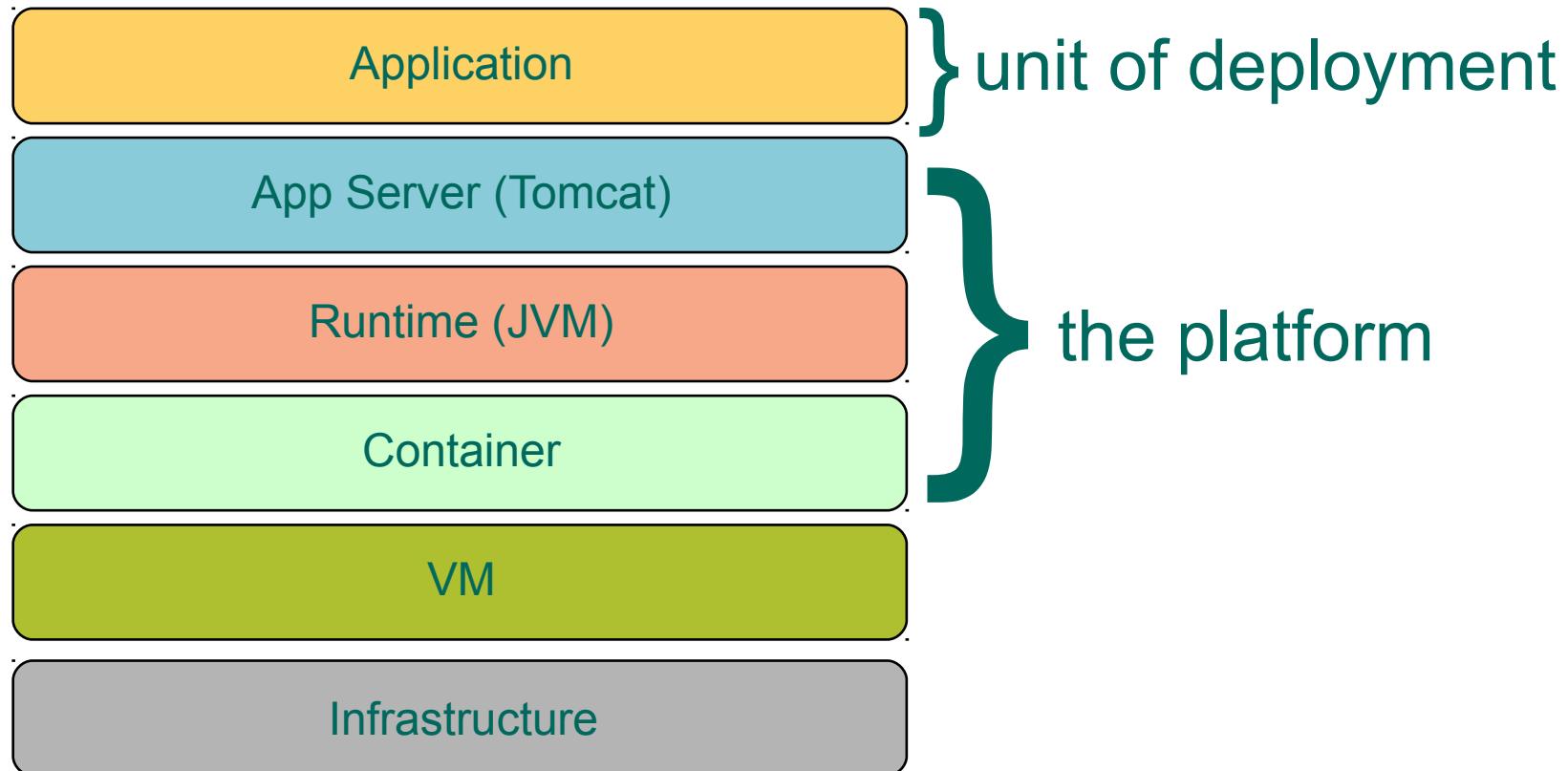


Scaling = creating VMs from blueprints/templates



CLOUD FOUNDRY

# PaaS: App-centric Deployment

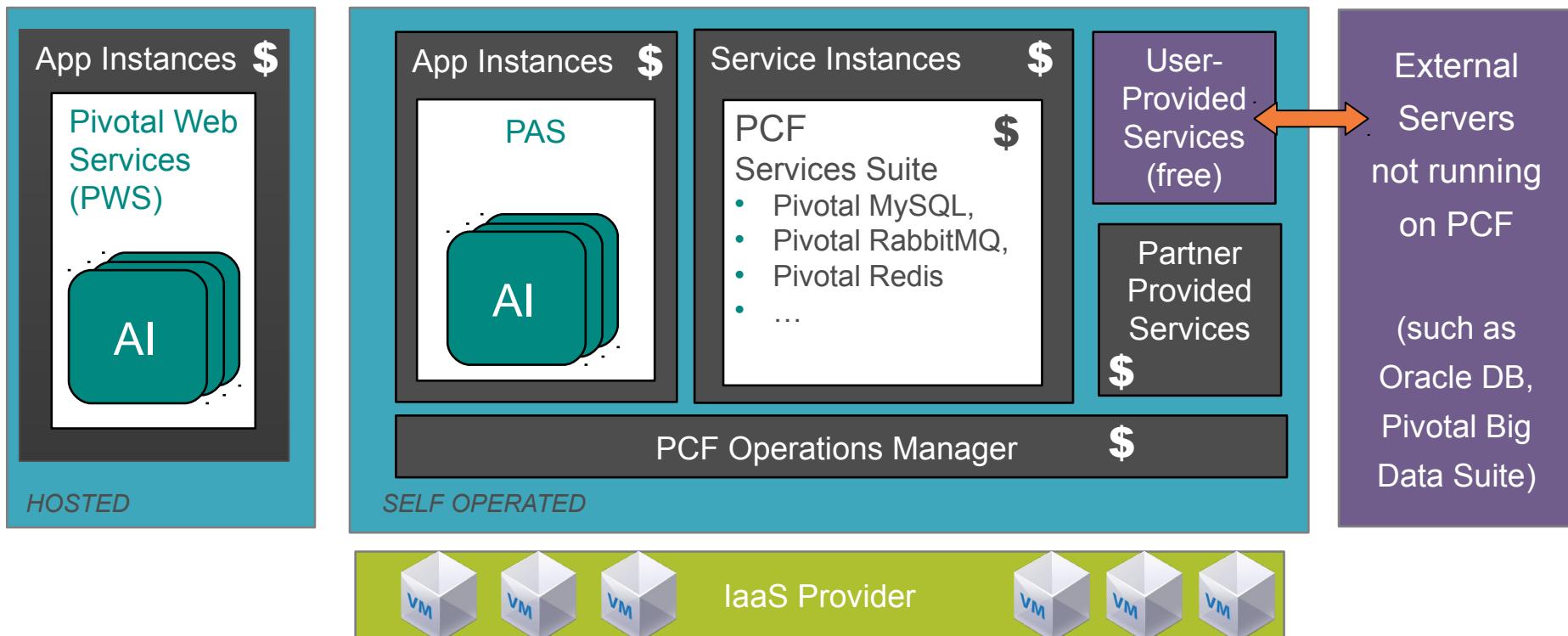


Scaling = creating containers in a VM pool

Pivotal™

# What 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

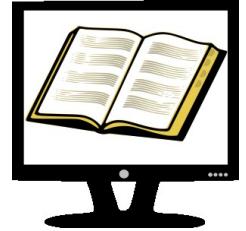
- Highly scalable
- Self healing
- Logging & audit trail
- Existing enterprise policies – user access & authorization
- Application monitoring
- Operational metrics
- App uptime SLAs

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



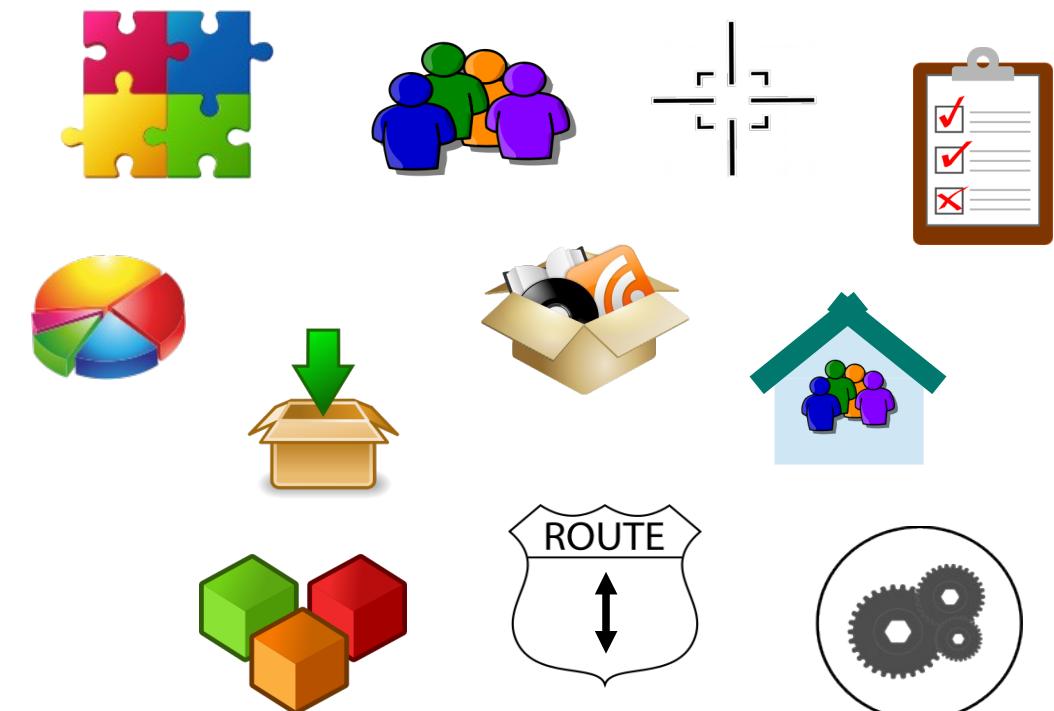
# Cloud Foundry Concepts

## 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: .
```

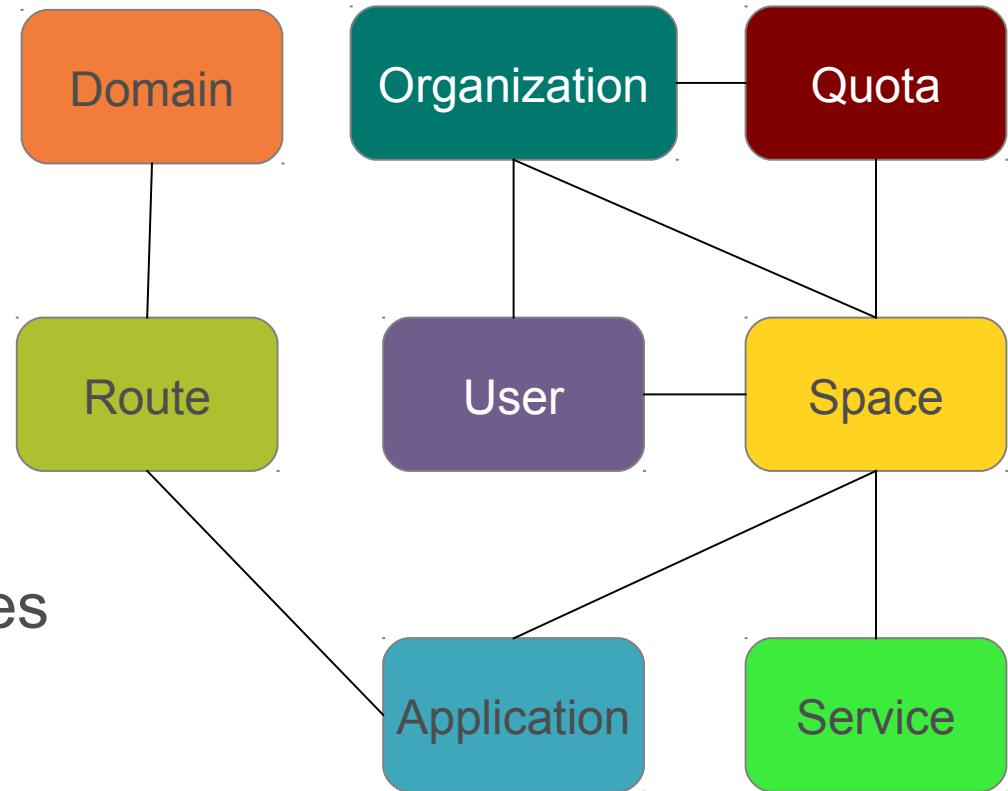
# Roadmap

- 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

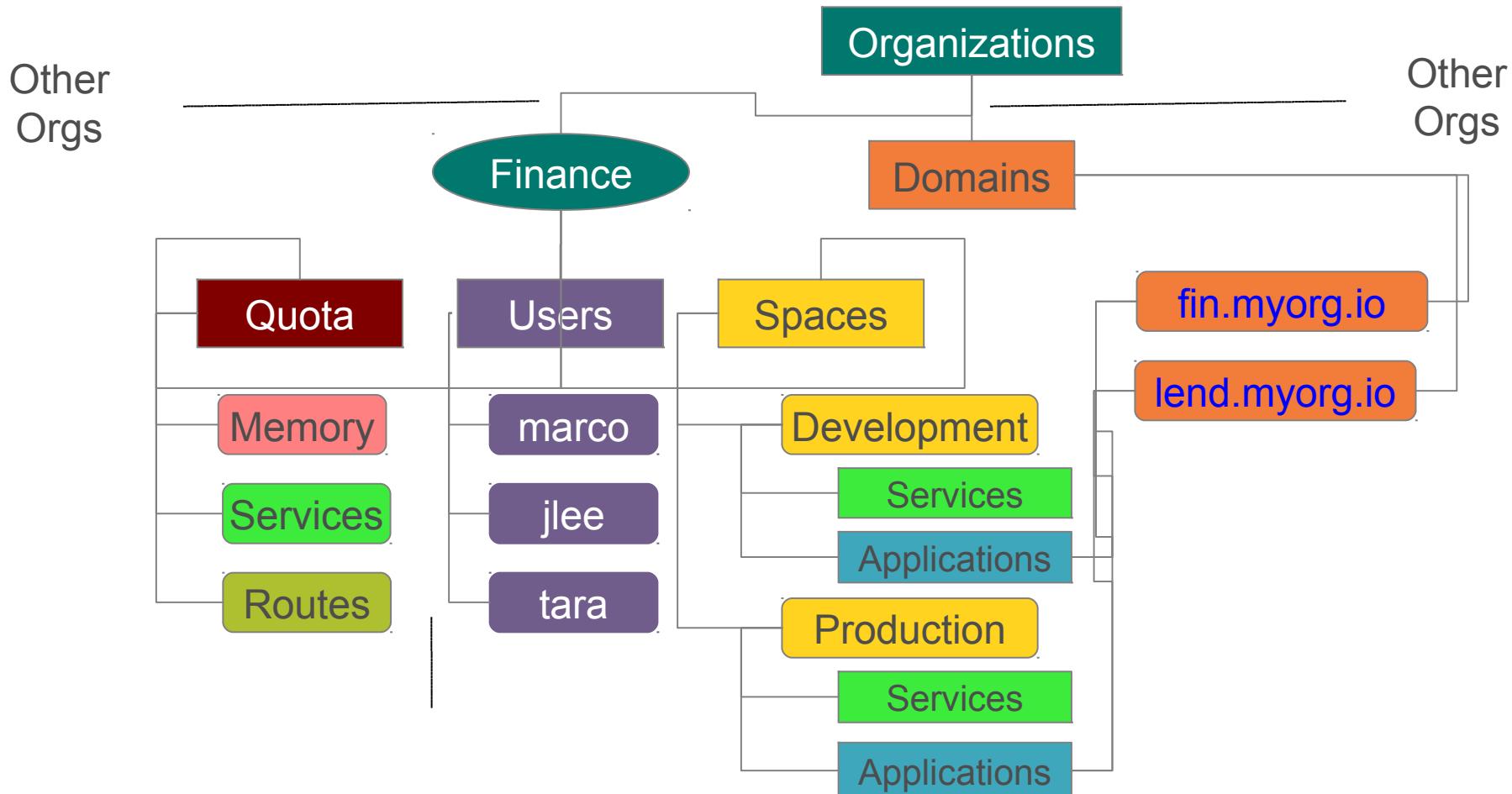




# Organizations

- 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

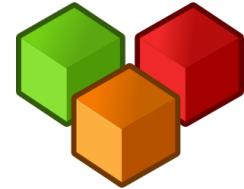
Online Banking, Mobile Banking, International Payments

By application suite

Office Tools, Branch Advisor, Mortgage Assist

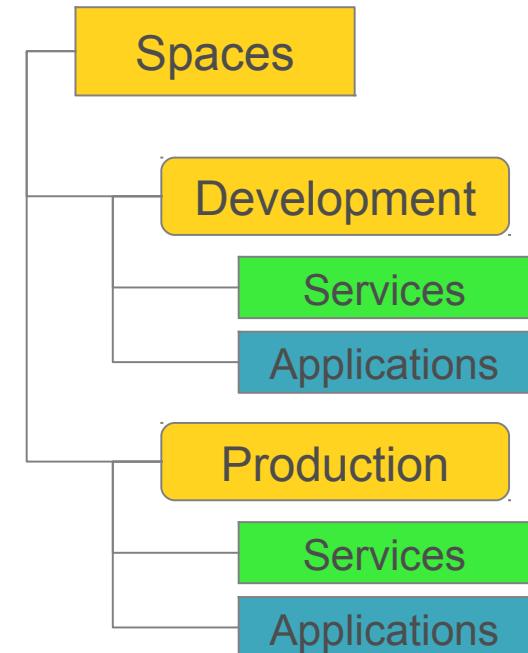
Each would have same/similar spaces

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

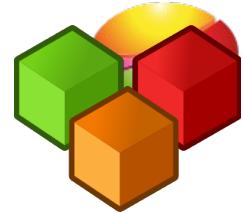
- Space Manager
  - Can invite/manage users, enable features for a given space
- Space Developer
  - Can create, delete, manage applications and services, full access to all usage reports and logs
- Space Auditor
  - View only access to all space information, settings, reports, logs



# 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
```

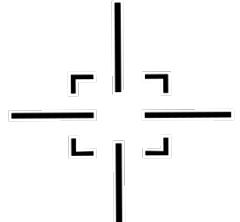


# Quotas

- 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**

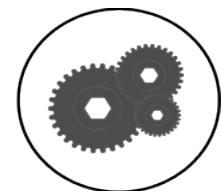
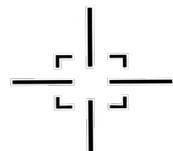
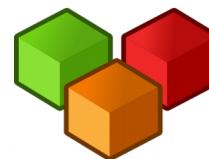


# Services

- 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

# Overview

- After 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](https://console.run.pivotal.io)
    - Click “**SIGN UP FOR FREE**” to setup a new account and your username/password
  - API URL: [api.run.pivotal.io](https://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

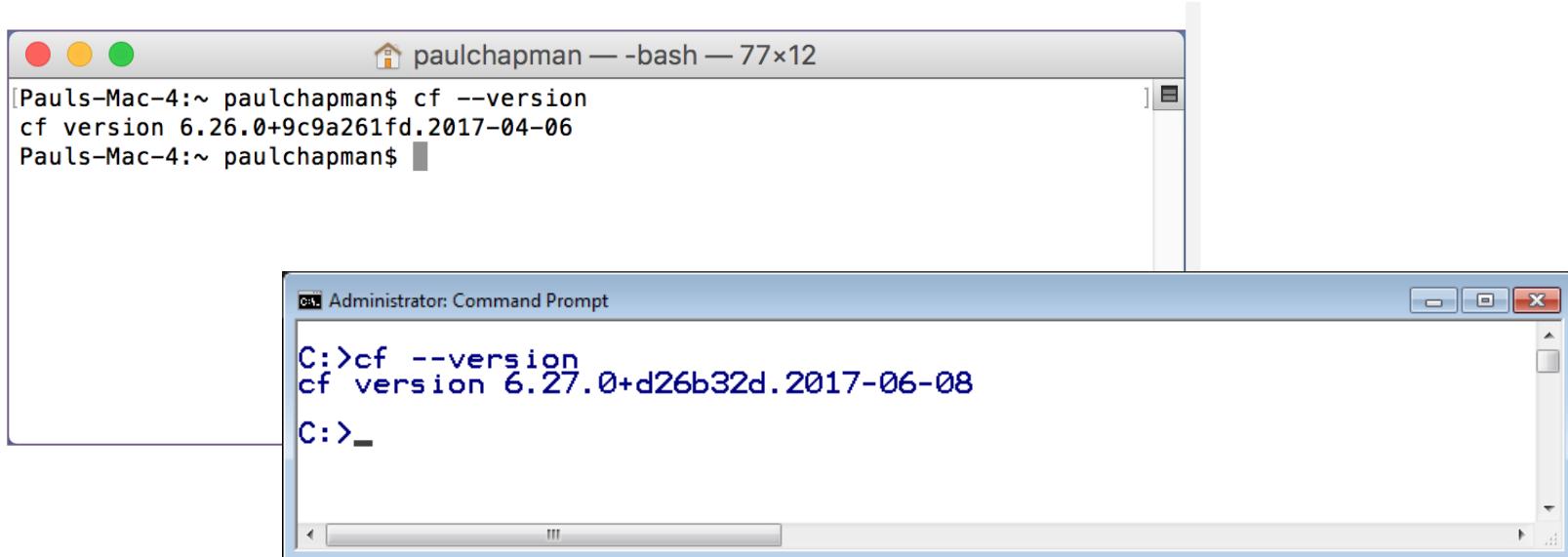
- **Getting Started with the Command Line Interface**
- Login
- Deploying an Application
- Managing Application Instances

# 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

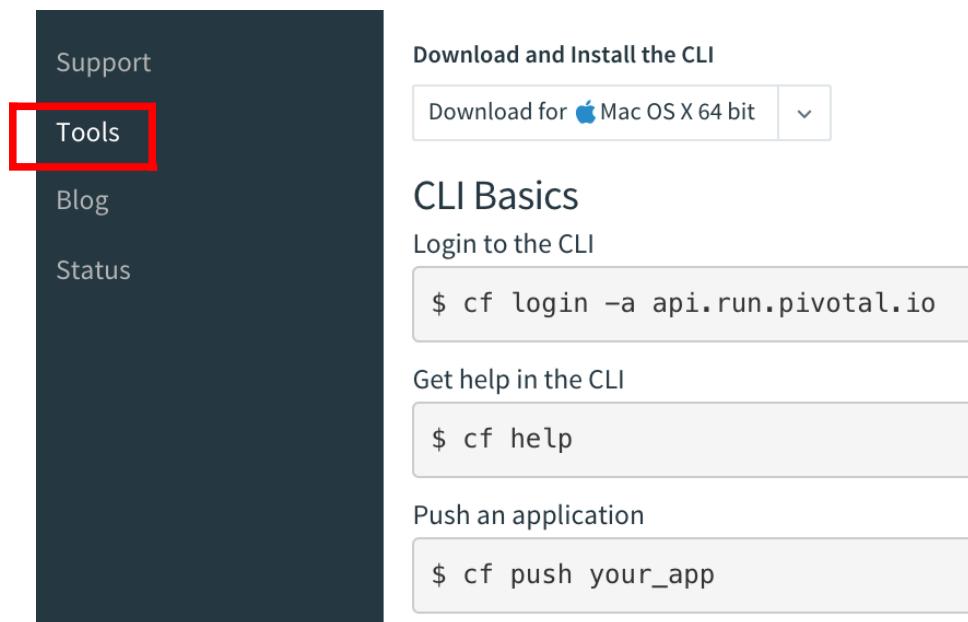
For simplicity, most examples in this section show PWS URLs

- 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

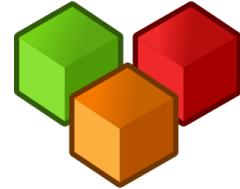


# DO NOW – Viewing Organization(s)

- 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:
```



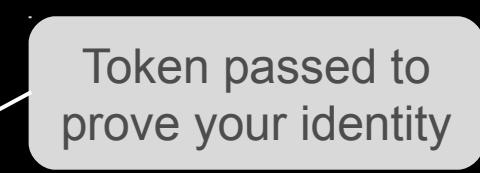
# Managing Spaces

- 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

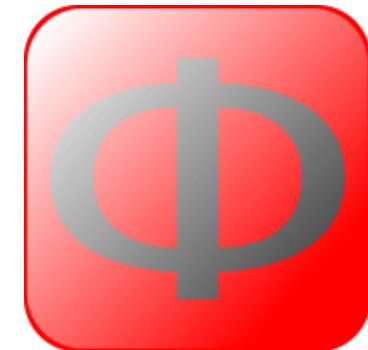
# Roadmap

- 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 750M  
                  -n spring-music-678  
                  -p build/libs/spring-music.war
```

- Deploys war file (specify path if needed)
- 1 instance, 750M 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 750M  
-p pre-built/spring-music.war
```

```
> cf push spring-music -n spring-music-678 -p build/libs/spring-music.war -i 1 -m 750M
```

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

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

Updates CF metadata  
(app name, instances, memory)

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

Establish & bind route

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

Uploads war

```
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 2.0.2_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  
\(0.2s\)  
----> 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  
\(0.0s\)  
...  
----> 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: 750M 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 750M	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

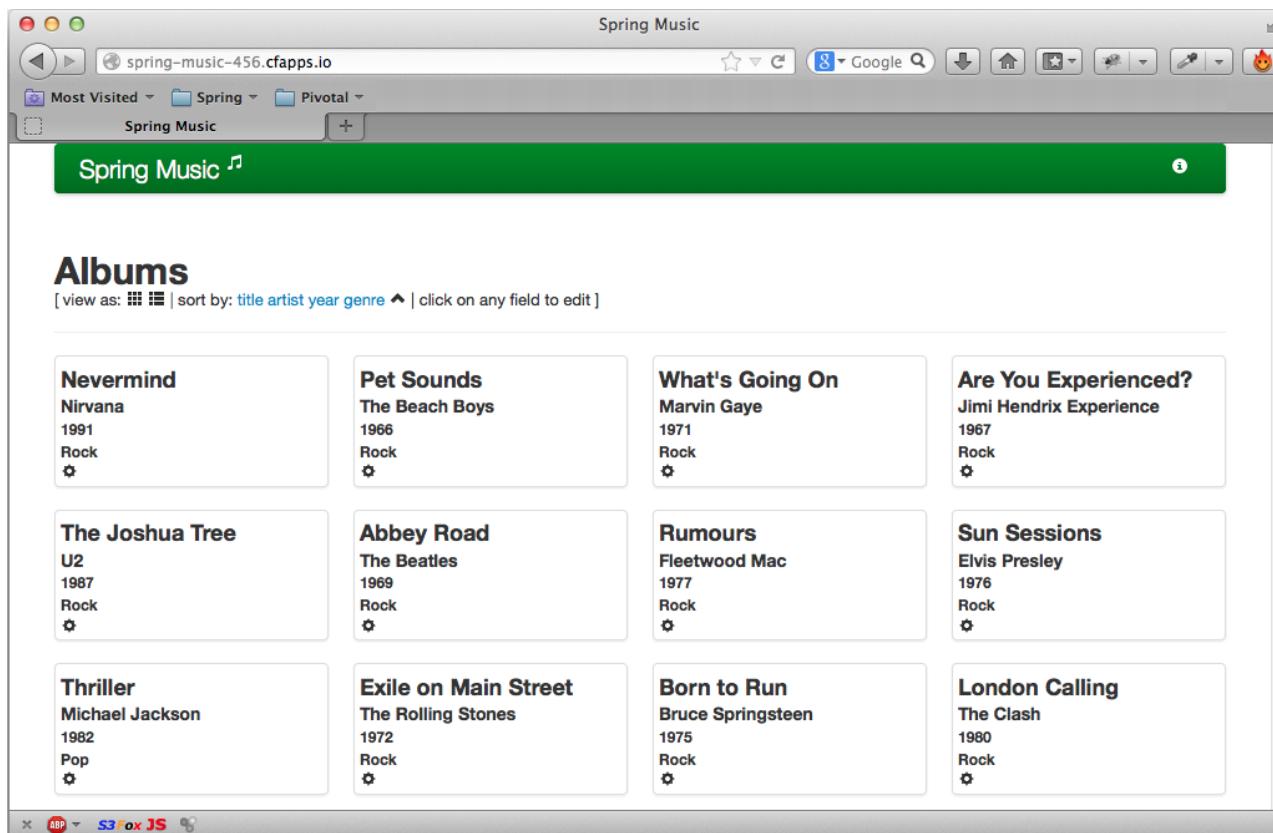
name          requested state    instances   memory   disk      urls
spring-music  started           1/1        750M     1G       spring-music-678.cfapps.io
```

- `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, an 'ORG' section, and a dropdown menu set to 'pivotaledu'. Below that is a 'SPACES' section with four items: 'development' (which is selected and highlighted in green), 'production', 'staging', and 'Marketplace'. A red arrow points from the text 'Click to select different space' to the 'development' item in the sidebar. The main content area shows the 'pivotaledu > development' space. It has a summary bar indicating '1 Running', '0 Stopped', and '0 Crashed'. Below this is a navigation bar with tabs: 'App (1)' (which is selected and highlighted in blue), 'Services', 'Security', and 'Settings'. The 'App (1)' tab is active. The main table below is titled 'Apps' and has columns: NAME, INSTANCES, MEMORY, LAST PUSH, and ROUTE. There is one row for 'spring-music' which is 'Running' with 1 instance, 512MB memory, pushed 17 hours ago, and the URL <http://spring-music-pc.cfapps.io>. A red arrow points from the text 'URL of your application' to the URL in the table. Another red arrow points from the text 'Click application name to see its dashboard (next slide)' to the 'spring-music' application name.

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: **Running**

Stop | Restart | Status

APP: **spring-music**

Logs

Go to App Home Page | VIEW APP

Overview Services Route (1) Logs Tasks Settings Buildpack: N/A

Events Last Push: 02:19 PM 04/30/18 App Summary

Scaled app instances to 1  
pchapman@pivotal.io 04/30/2018 at 02:19:58 PM

Started app  
pchapman@pivotal.io 04/30/2018 at 02:19:30 PM

Select tabs for more details

Instances / Allocated: 1 / 1 | Memory / Allocated: 0.34 / 0.50 GB | Disk / Allocated: 0.15 / 1.00 GB

Application Information

Processes and Instances

Get runtime info | View in PCF Metrics

Scaling control

web

Instances: 1 | Memory Allocated: 512 MB | Disk Allocated: 1 GB | SCALE

Autoscaling

#	CPU	Memory	Disk	Uptime
0	0%	344.8 MB	153.01 MB	1 min

Recent events

# Scaling Control

 Scale app

Click Scale button to reveal scaling side-panel

web

Instances

Memory Limit

Disk Limit

1



512 MB



1 GB



Usage Total

Instances

Memory Limit

Disk Limit

1

0.50 GB

1.00 GB

APPLY CHANGES

# Managing Routes

APP  
spring-music ● Running [VIEW APP](#)

Overview Services **Route (1)** Logs Tasks Settings Buildpack: N/A

Routes [MAP A ROUTE](#)

<https://spring-music-pc.cfapps.io>

**Map a Route**

**Map a New Route**

Hostname / Path (optional)

**Map an Existing Route**

[select a route]

Enter new domain (2) – make it *unique*. Click **MAP** (3)

**CANCEL** **MAP**

Delete a mapping (1) click red cross at far right

1

2

3

# 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 *PAS\** Architecture



\* Formerly *Elastic Runtime*

# Agenda

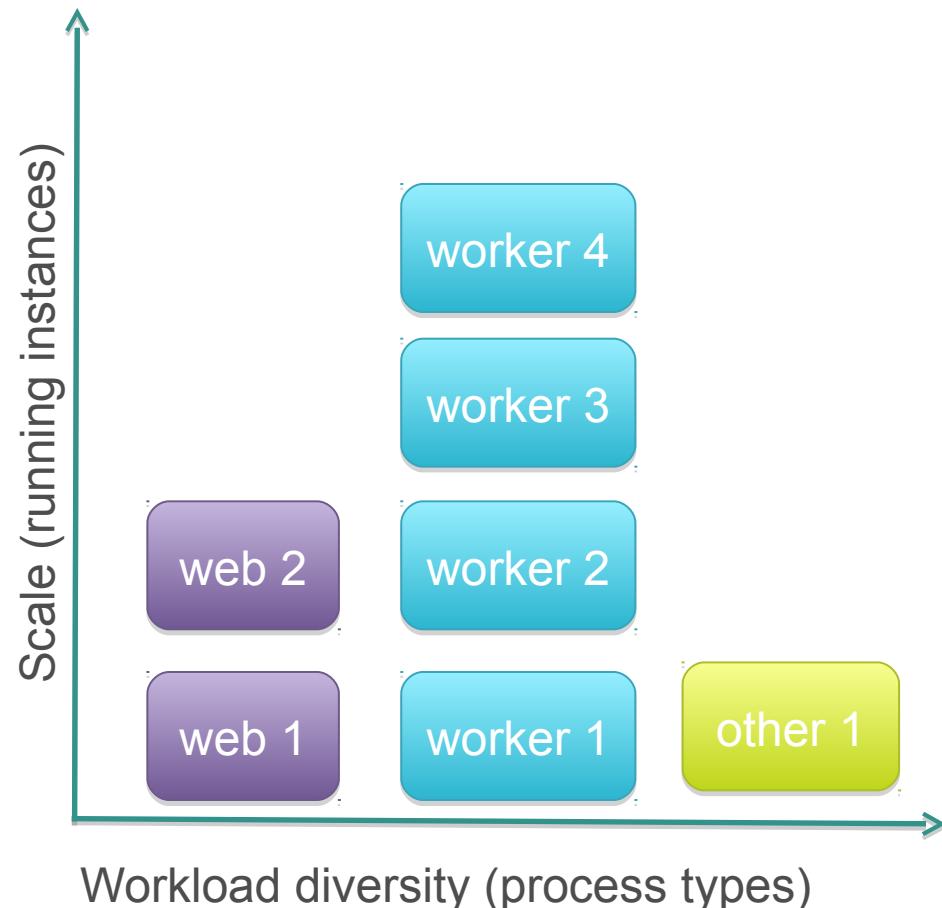
- **Cloud Native Apps**
- Elastic Runtime Architecture

# Fundamental Changes

*Cloud Native applications operate in a different environment*

# Distributed System

- Distributed systems are hard to build, test, manage, and scale
  - Cloud Foundry empathizes scaling out
    - Run multiple instances not bigger machines
  - Applications run in “containers” for safety



# “Ephemeral” (Temporary) Infrastructure

- Virtual machines and containers are temporary
  - When an application dies or is scaled down, its container is destroyed
    - Any in-memory state or local files (such as logs) are lost
  - Do not assume the file-system is always there – *it is not*
    - Nor is it shared by multiple instances

# Immutable Infrastructure

- Updates to systems and applications are *not* done in-place
  - New, updated instances are created *instead*
  - Existing instances are lost, just like scaling down

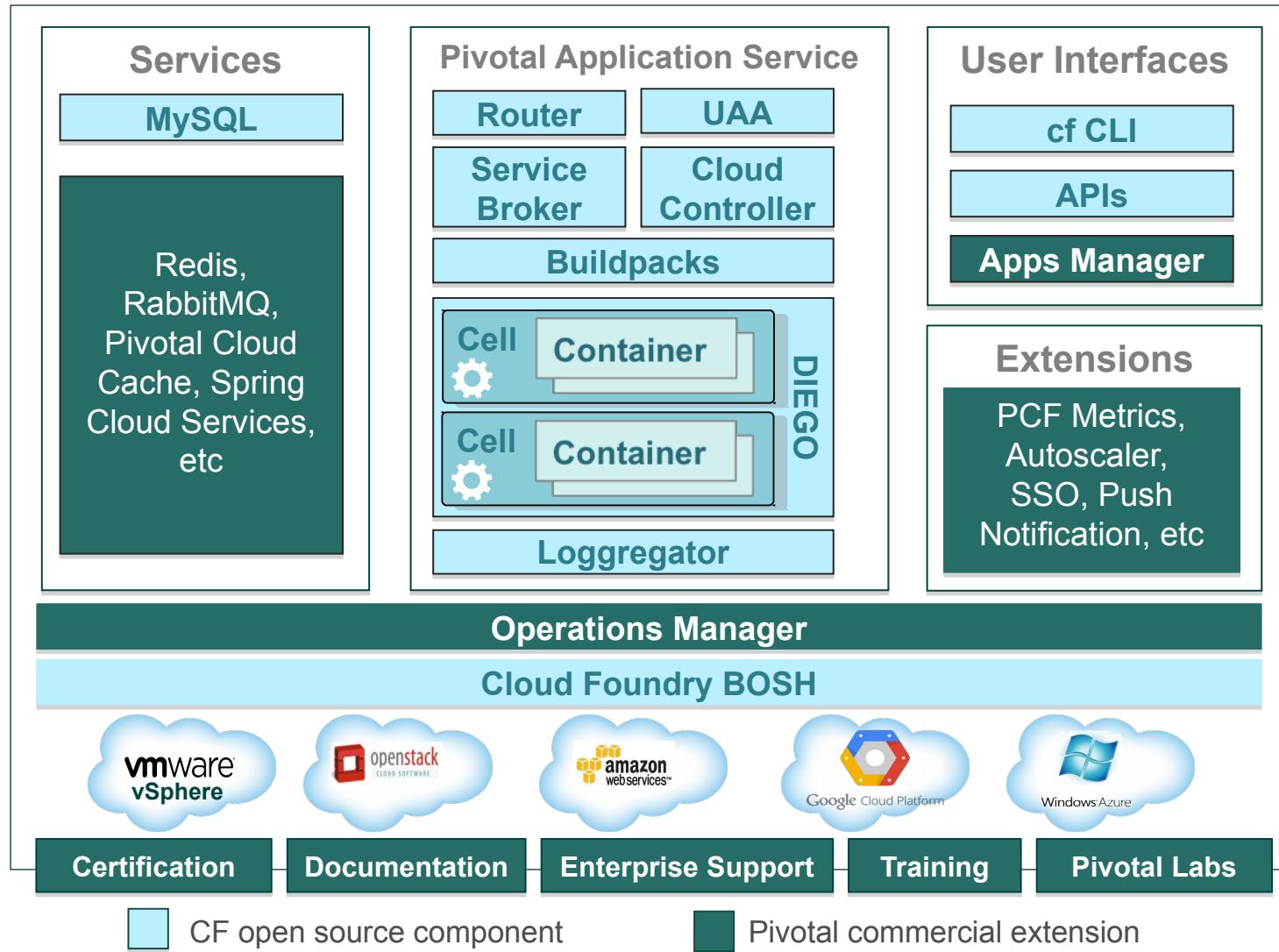
# Agenda

- Cloud Native Apps
- **Pivotal Application Server**
  - Formerly the Elastic Runtime

# Pivotal Cloud Foundry Overview

- Since PCF 2.0 three runtimes are available
  - Pivotal Application Server (PAS) – runs applications
  - Pivotal Kontainer Service (PKS) – runs Docker containers using Kubernetes
  - Pivotal Function Service (PFS) – coming soon
- What is the PAS?
  - *The PAS (formerly Elastic Runtime) is Cloud Foundry*
  - The subject of this course

# PCF Architecture



# Pivotal Application Server (PAS)

- PAS Subsystems
  - Diego (container manager)
  - Loggregator
  - Cloud Controller API
  - Routing

Pivotal Application  
Server  
=

Cloud Foundry  
OSS

~ 30 VMs

# Diego

*PAS's internal architecture*

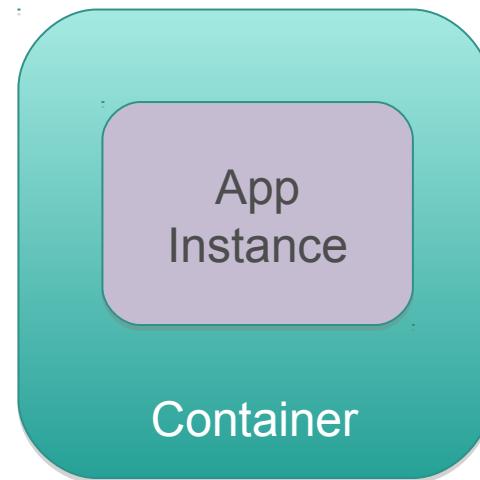
<https://docs.pivotal.io/pivotalcf/concepts/diego/diego-architecture.html>

# 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

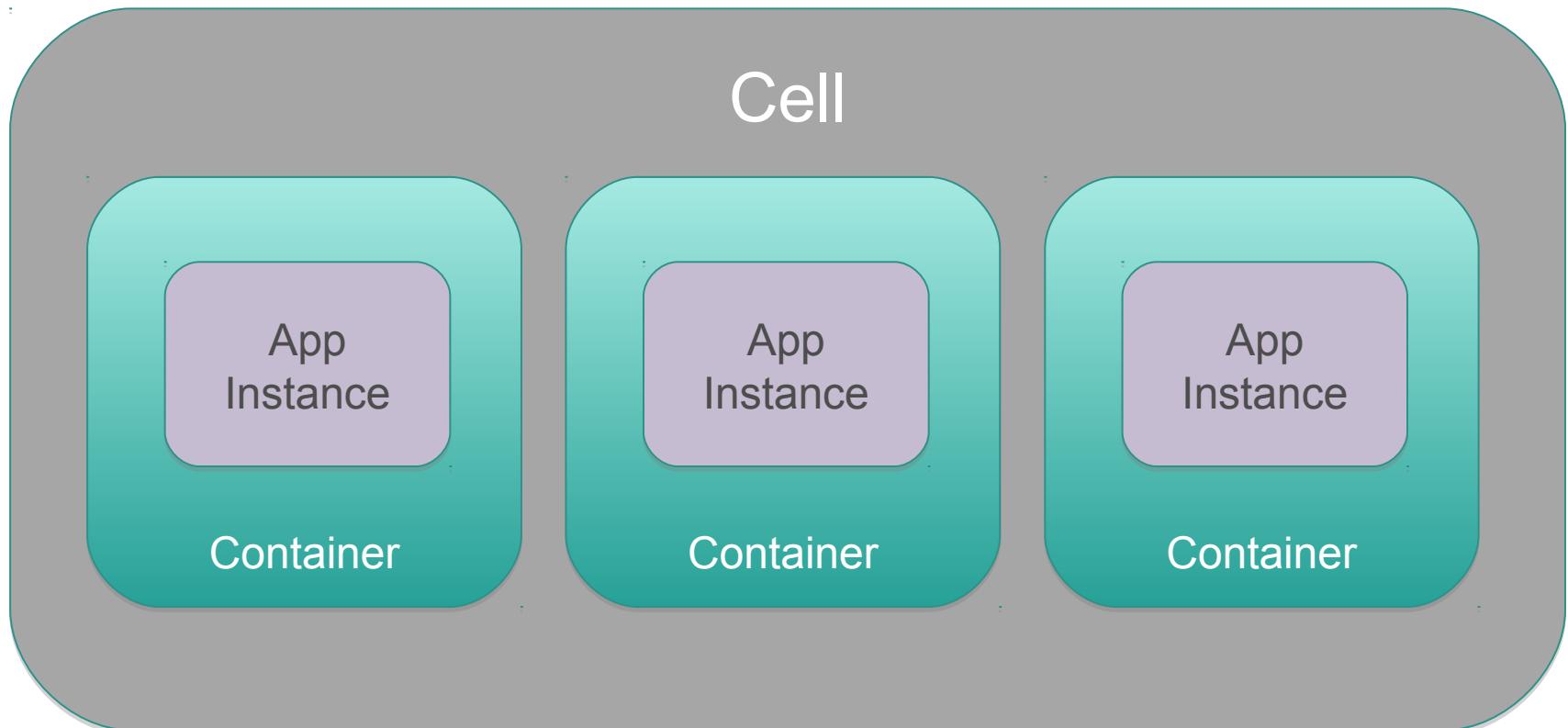
# Diego: Container

- An application instance is run within an immutable container



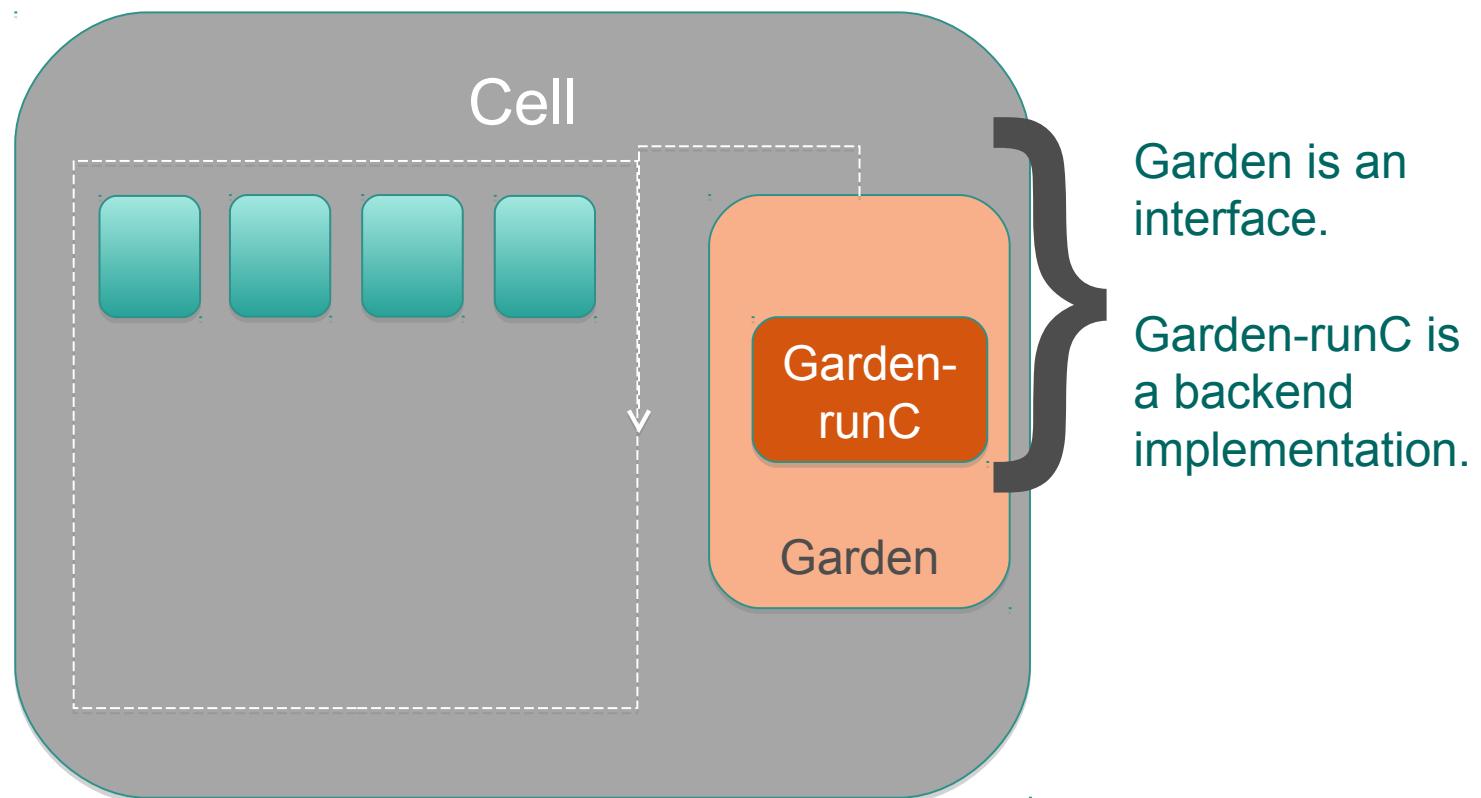
# Diego: Cell

- Containers are run within a Cell (Virtual Machine)



# Diego: Garden

- Containers are managed by Garden

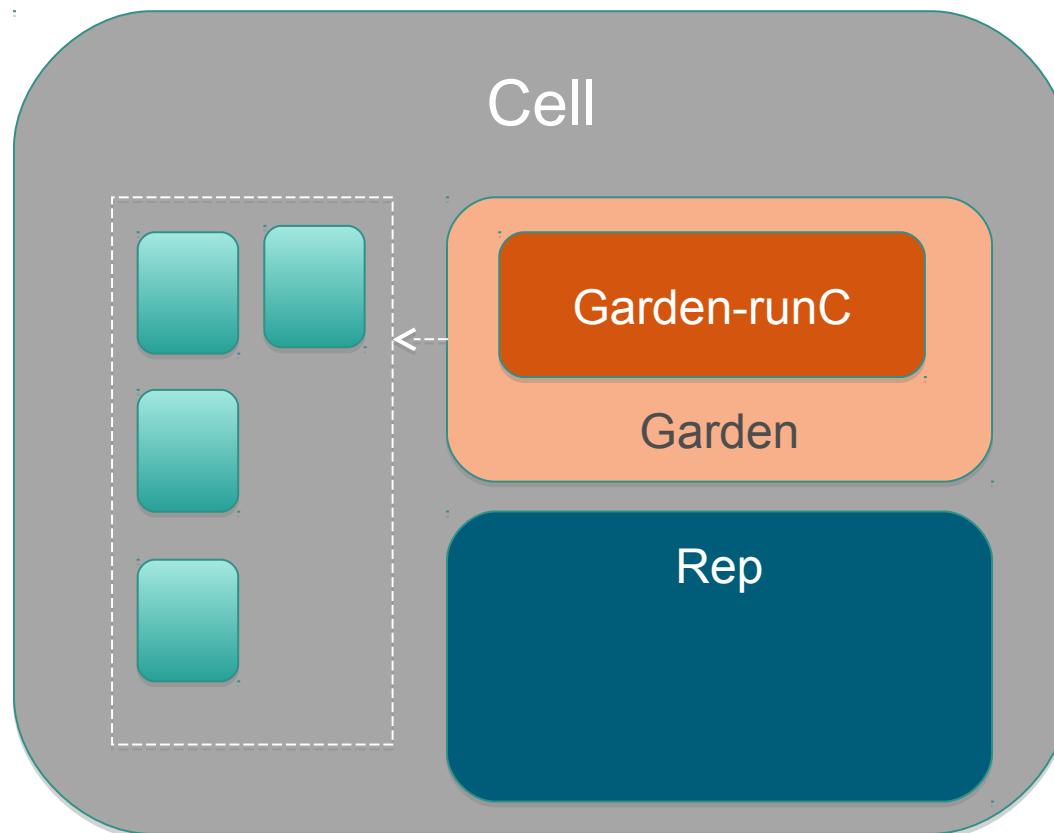


# Diego: Auction

- 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

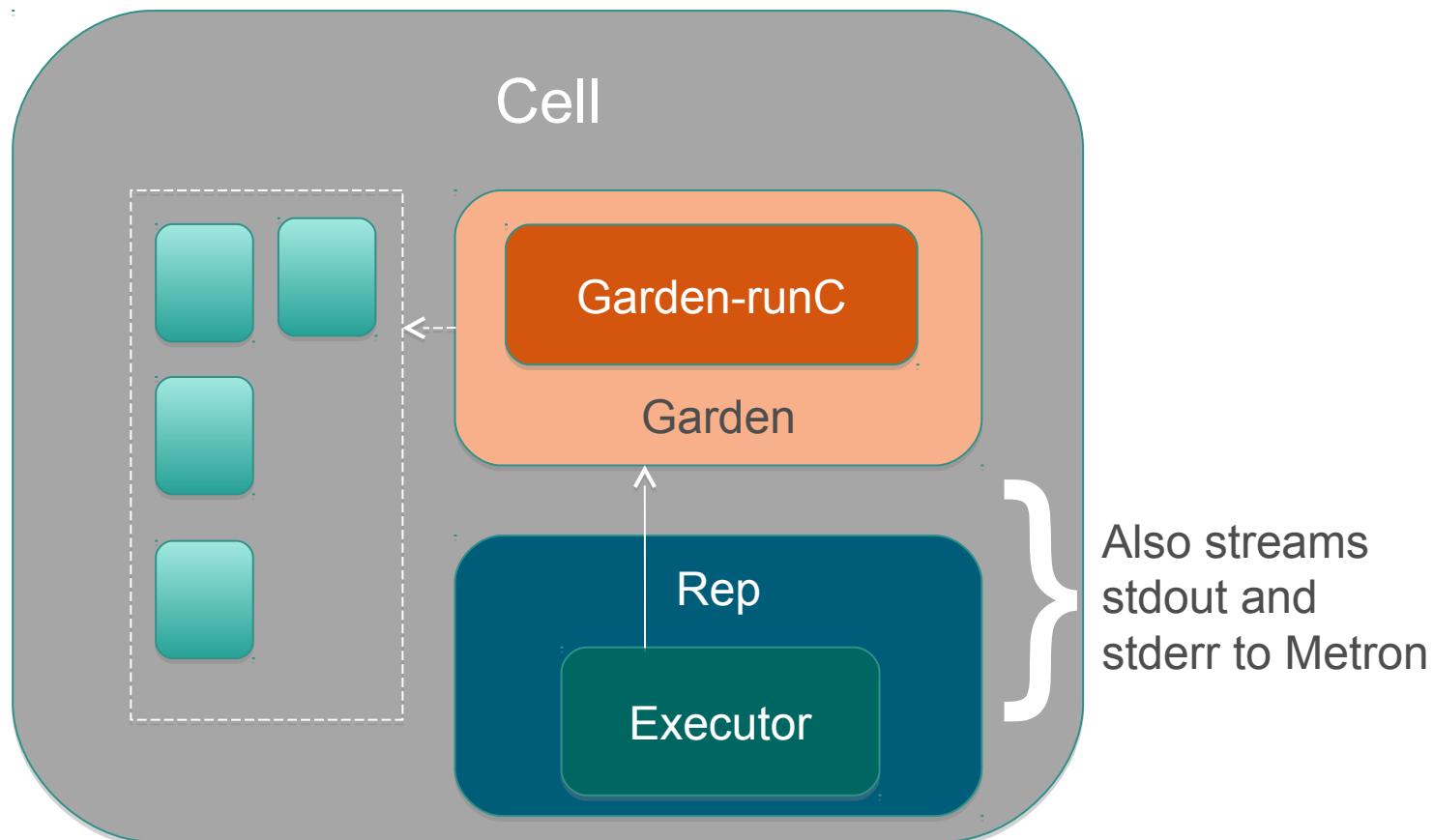
# Diego: Rep

- Represents the Cell in the BBS/auctions



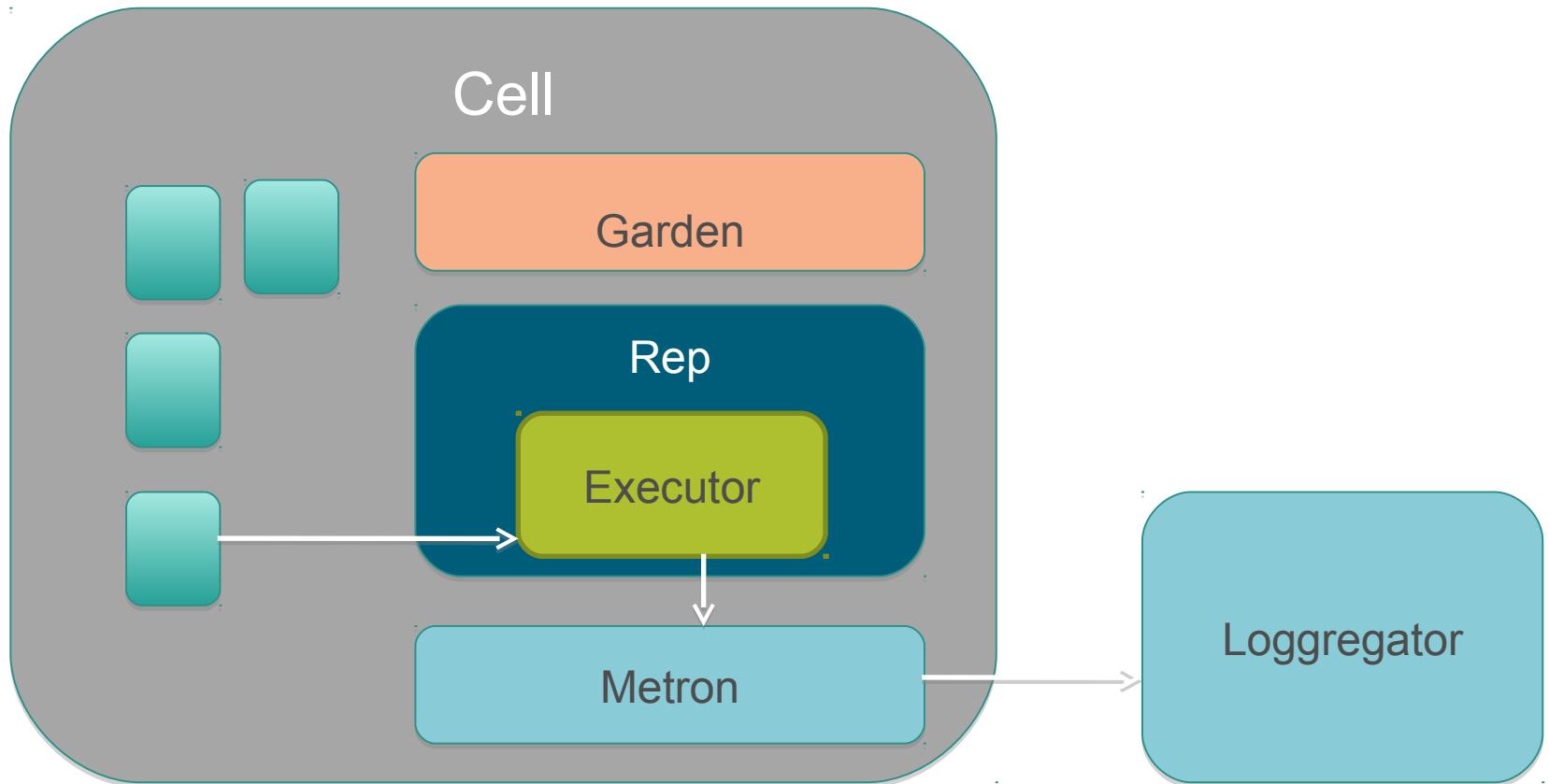
# Diego: Executor

- Manages container allocations on the cell



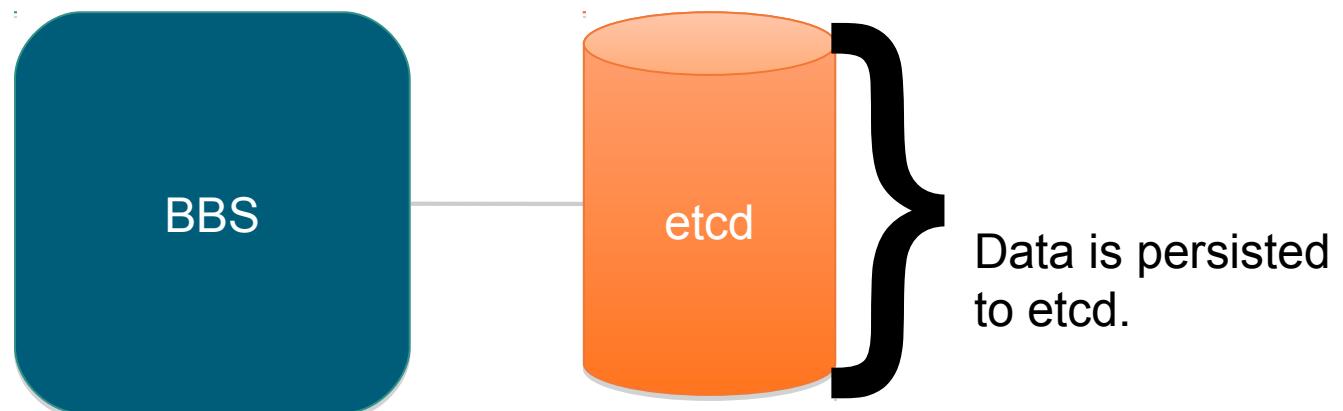
# Diego: Metron

- Forwards logs to the Loggregator subsystem



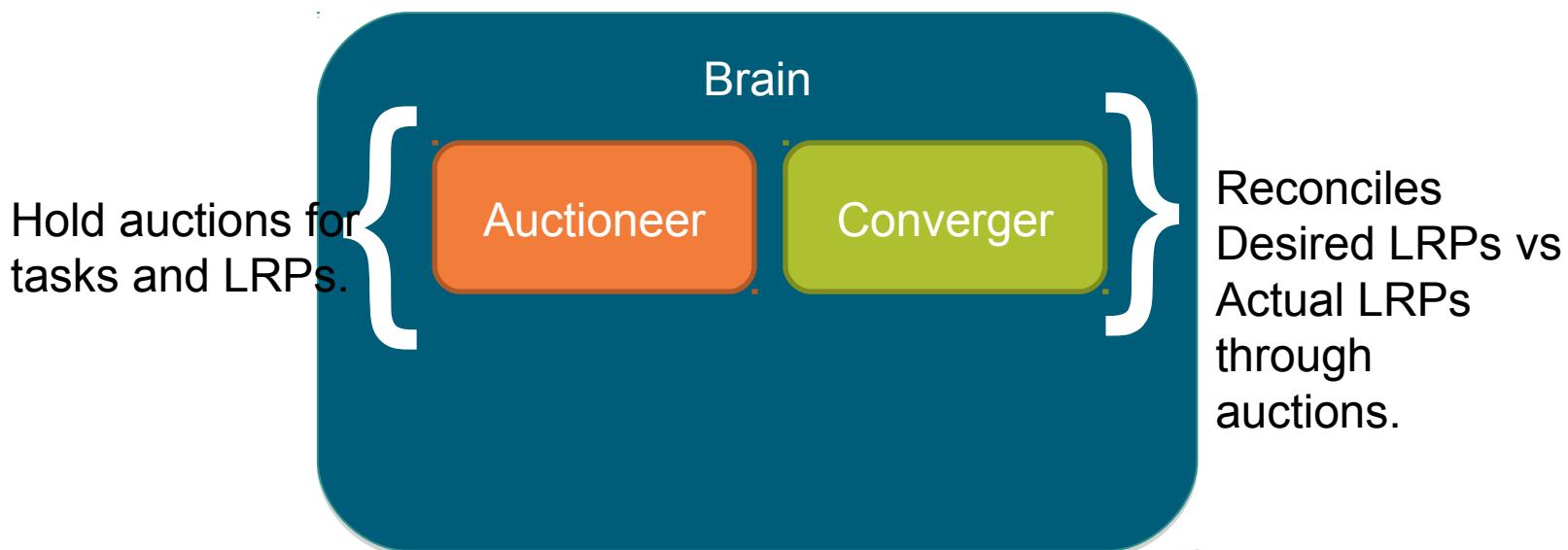
# Diego: BBS

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



# Diego: Brain

- The Brain is composed of two components



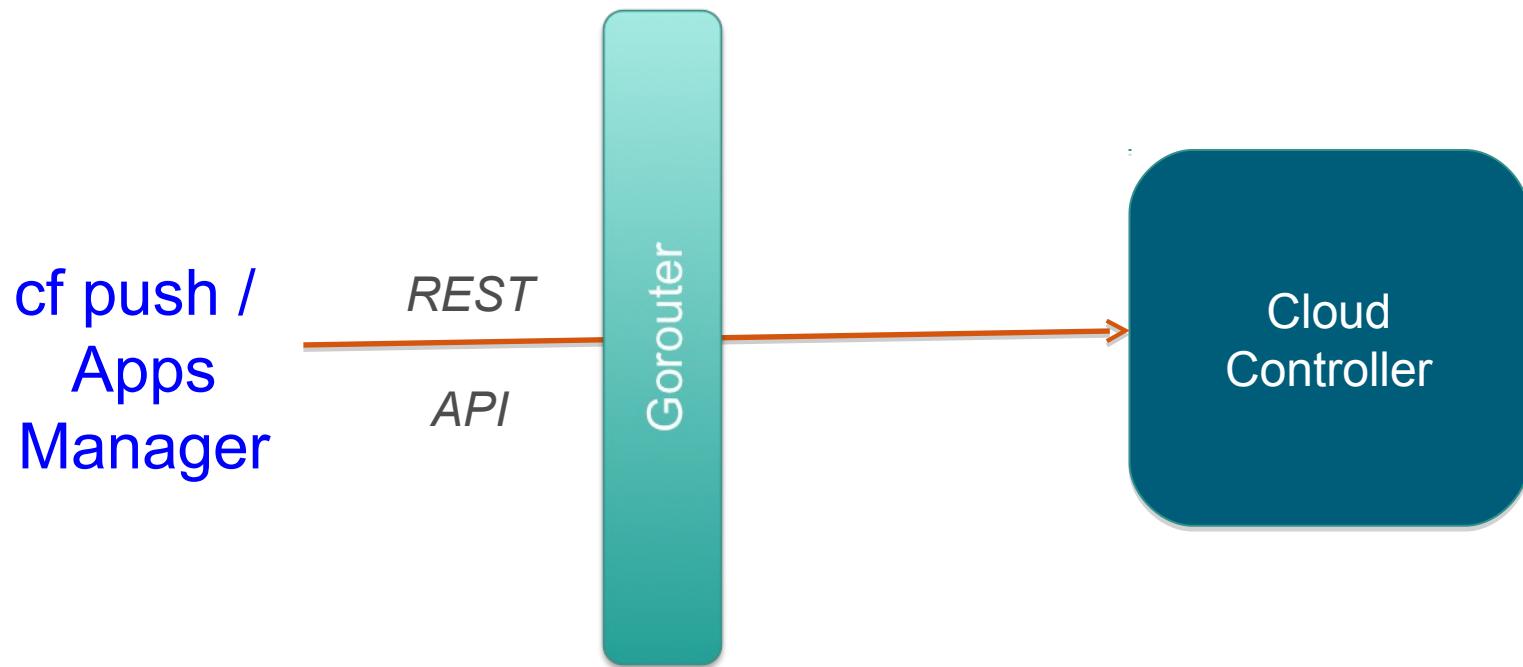
# Cloud Controller API

*Your interface to Cloud Foundry*

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

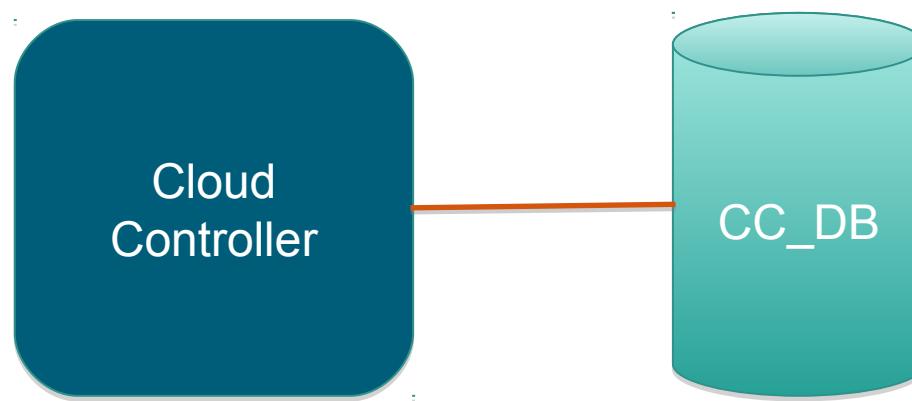
# CCAPI: Cloud Controller

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



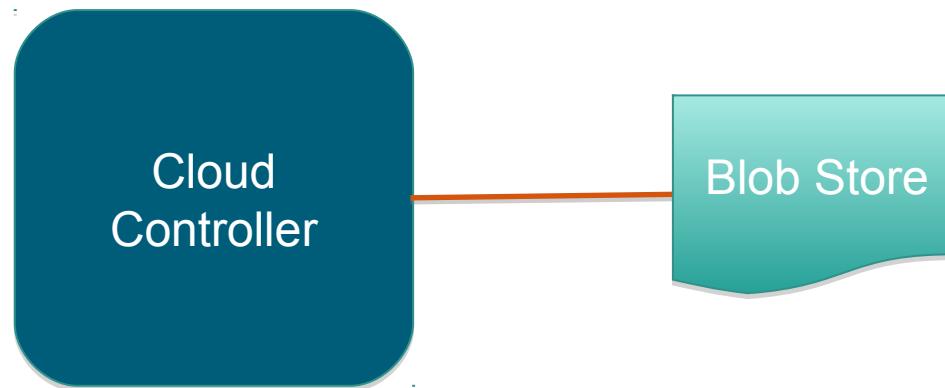
# CCAPI: Cloud Controller Database

- The Cloud Controller persists Org/Space/App data in the Cloud Controller Database
  - 3 node MySQL cluster (for redundancy)



# CCAPI: Blob Store

- The Cloud Controller persists app packages and droplets to the blob store
  - Default: Network Addressable Storage (NAS) device
  - Or use Amazon S3 if on AWS cloud



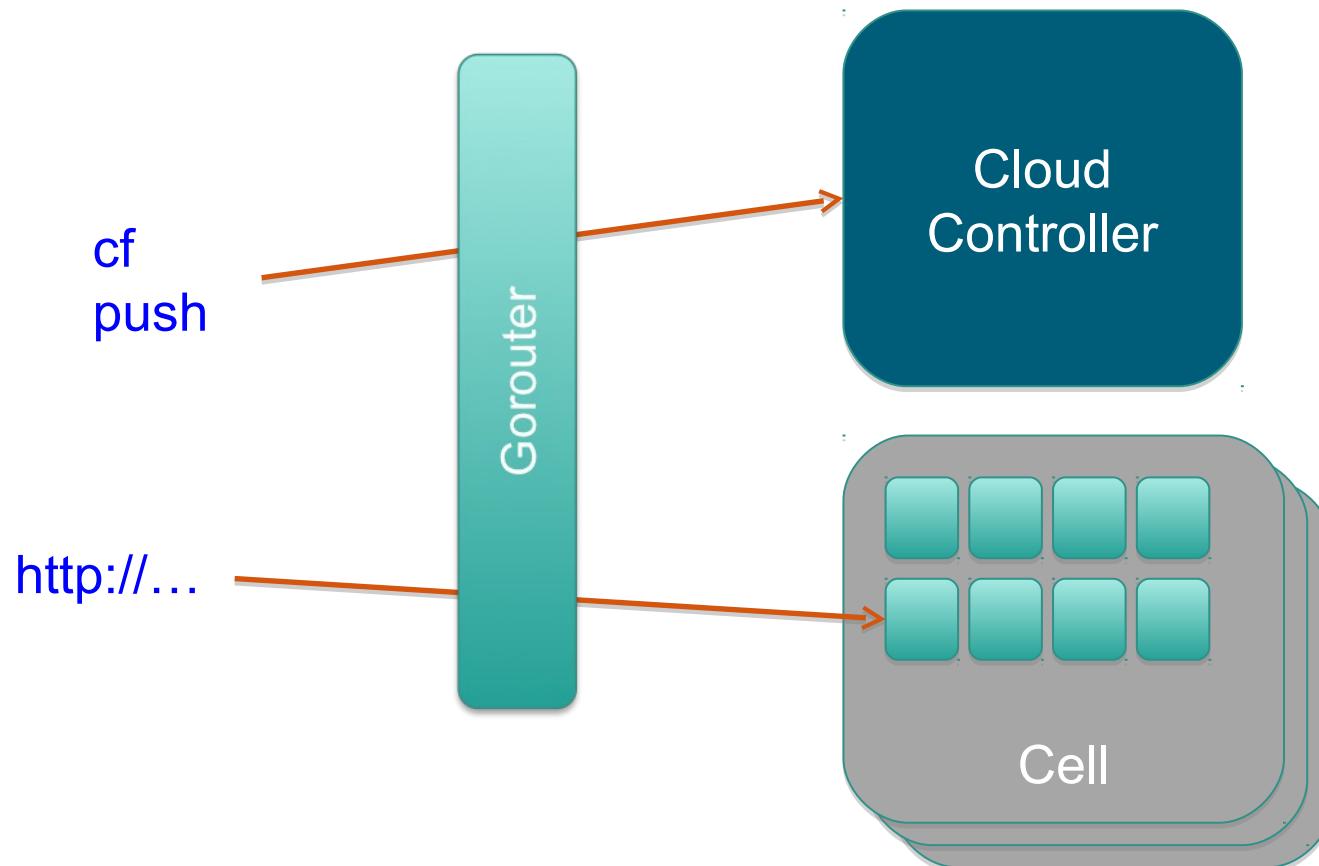
# Routing

*Allowing the outside to talk to the inside*

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

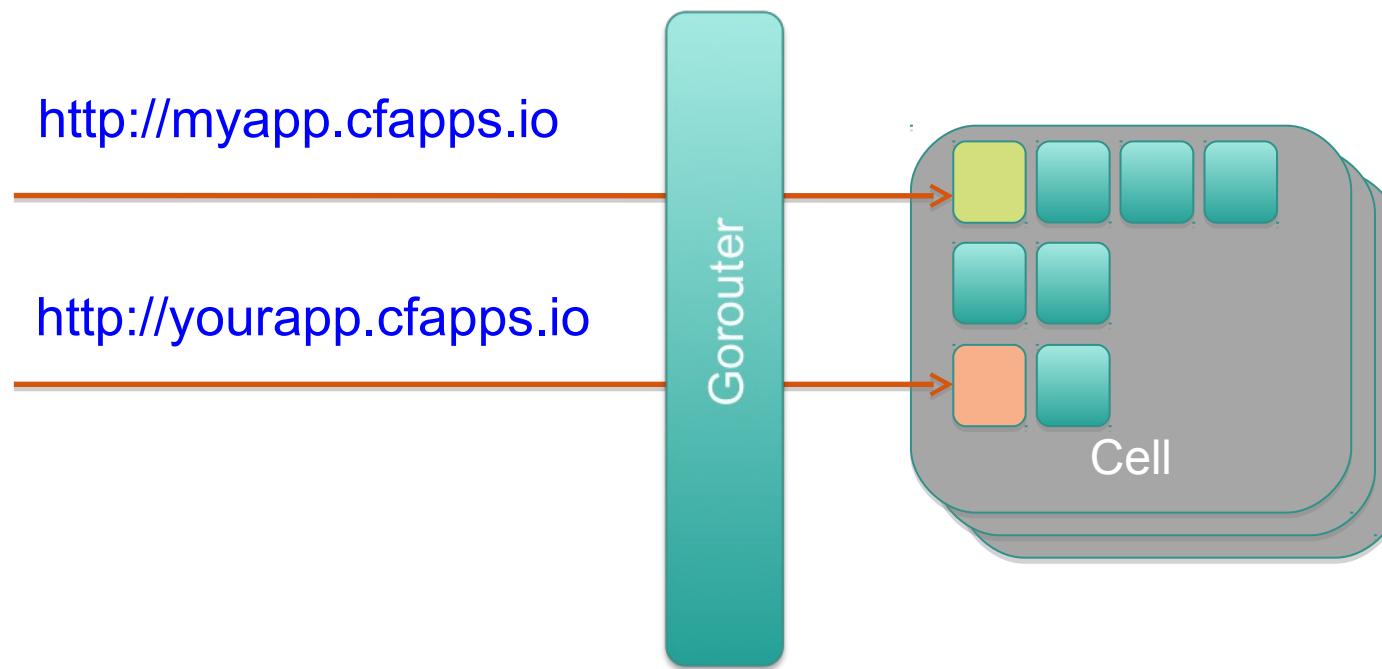
# Router

- The router directs traffic to appropriate component.



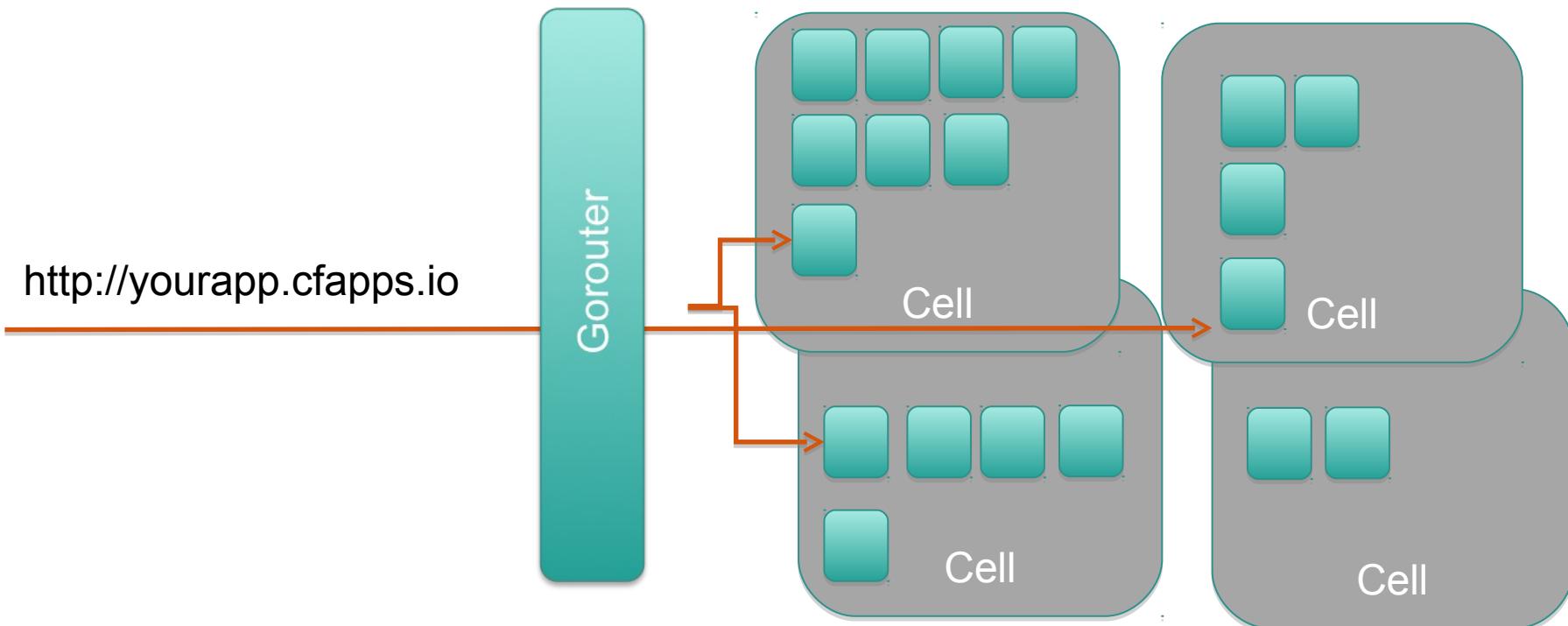
# Router

- The router routes requests to all app instances.

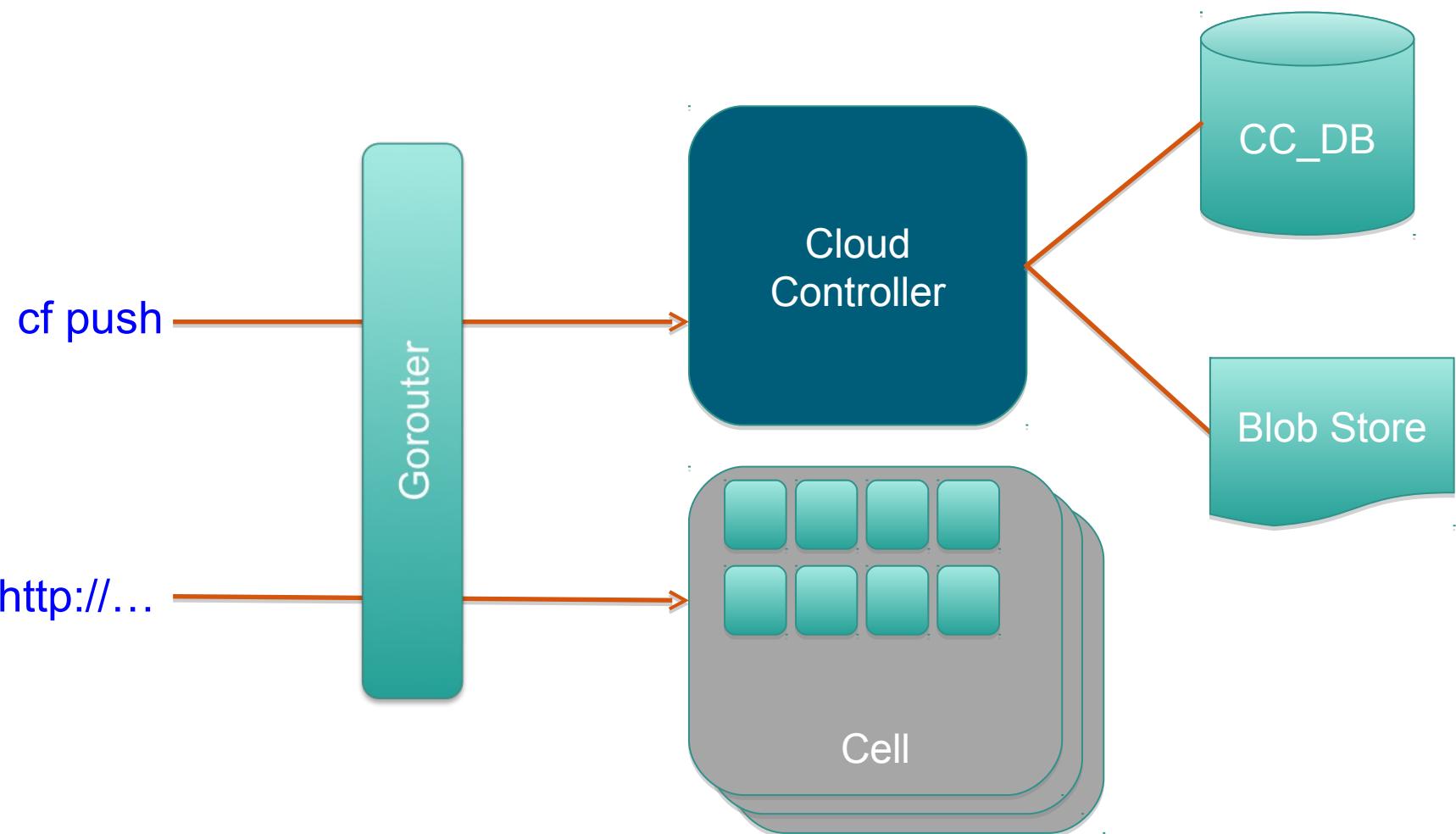


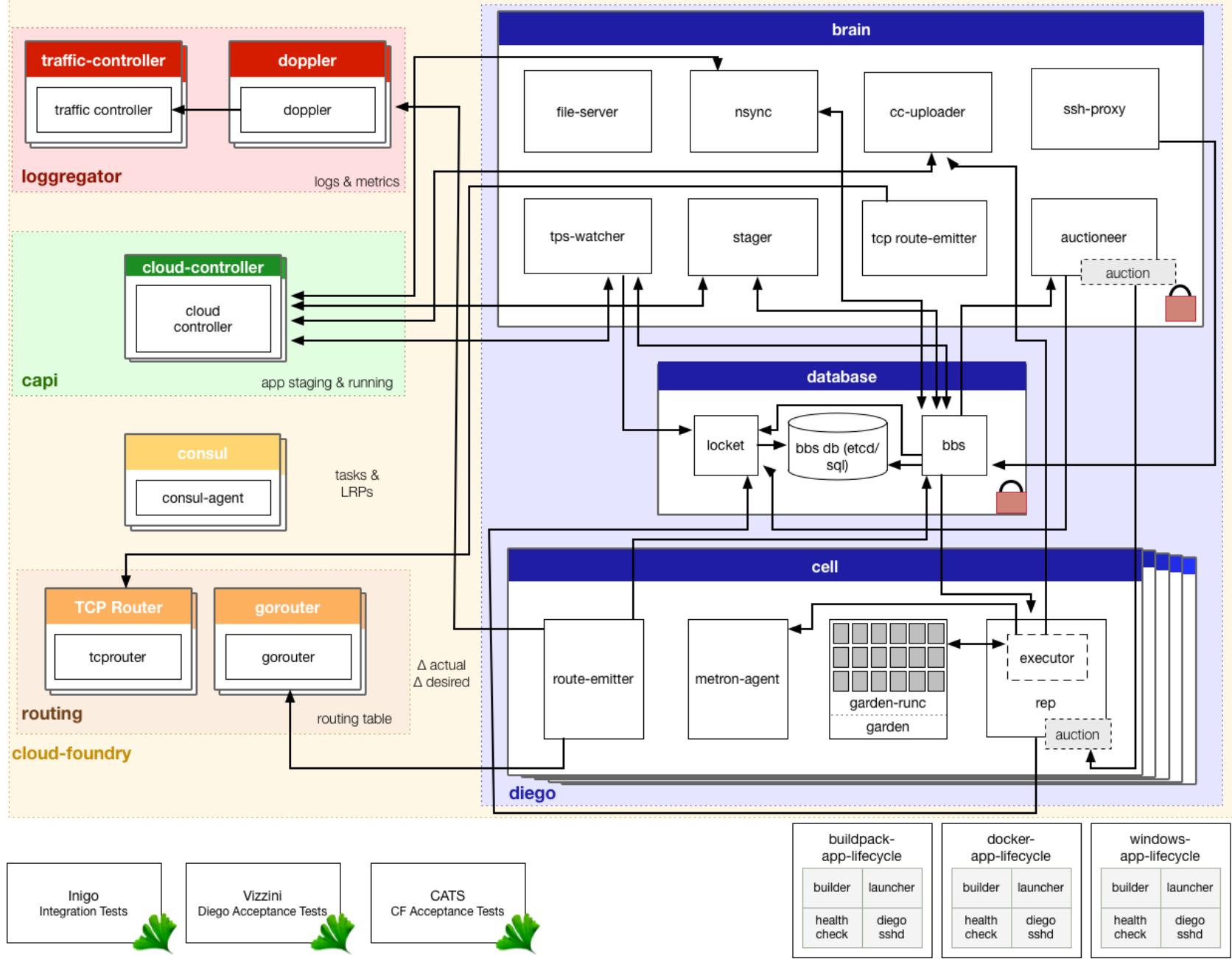
# Router

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



# PAS Simplified Architecture





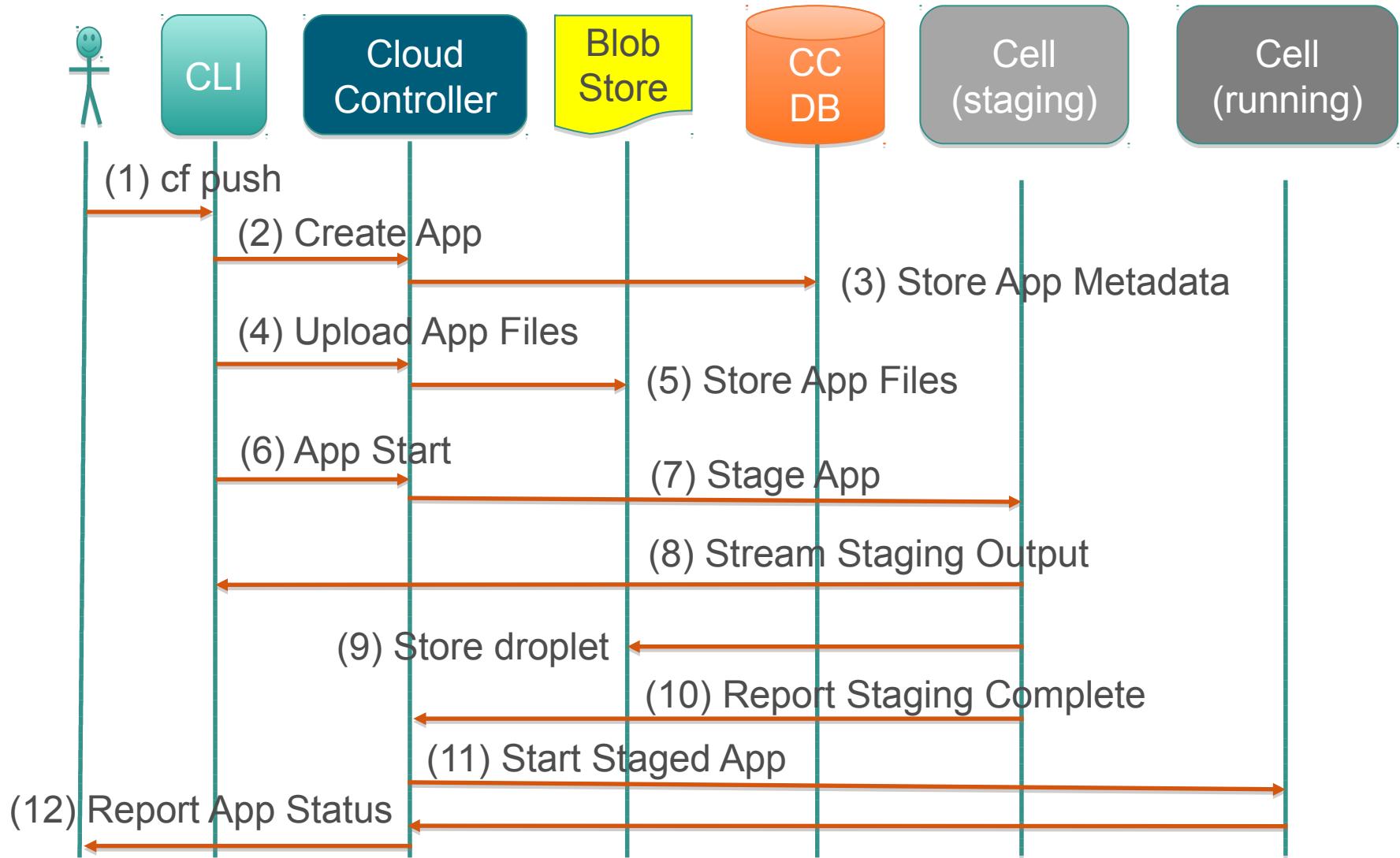
# Key Flows

*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 the Pivotal Application Service (*formerly Elastic Runtime*)
- How an application is Staged and Deployed

# Application Logging

Production features

The Loggregator

# Learning Objectives

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



# Agenda

- **Logging Architecture**
- Application Logging

# Loggregator

*Logging subsystem*

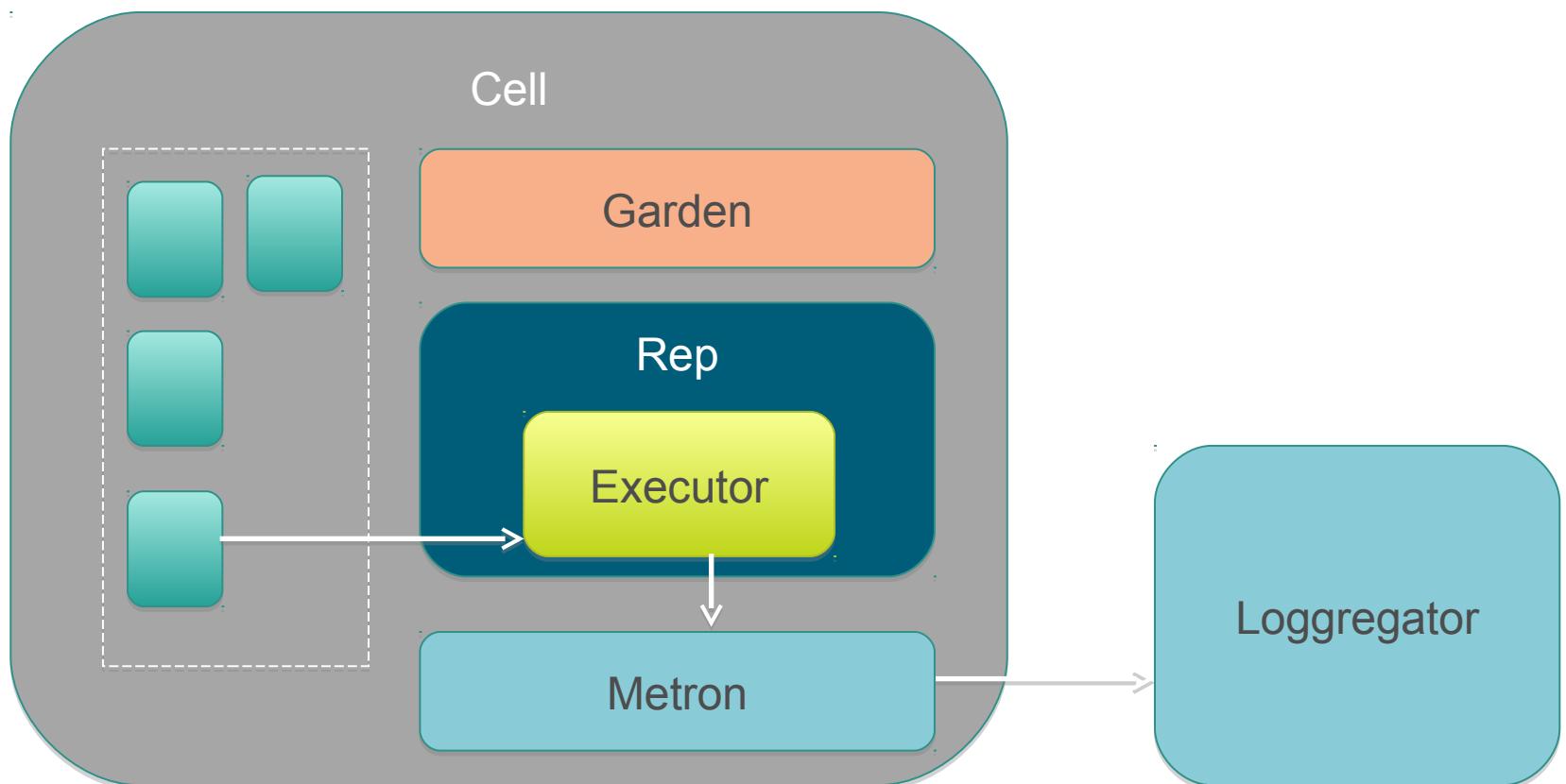
<https://docs.pivotal.io/pivotalcf/loggregator/architecture.html>

# 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

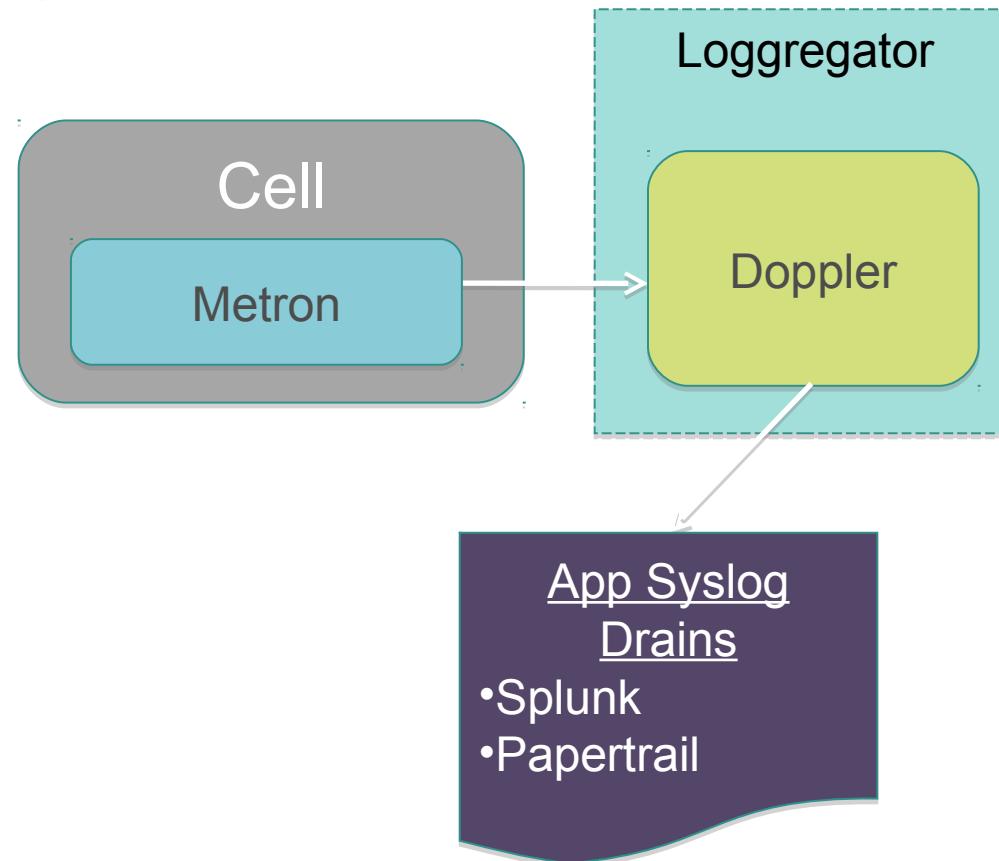
# Loggregator: Metron

- Forwards logs to the Loggregator subsystem



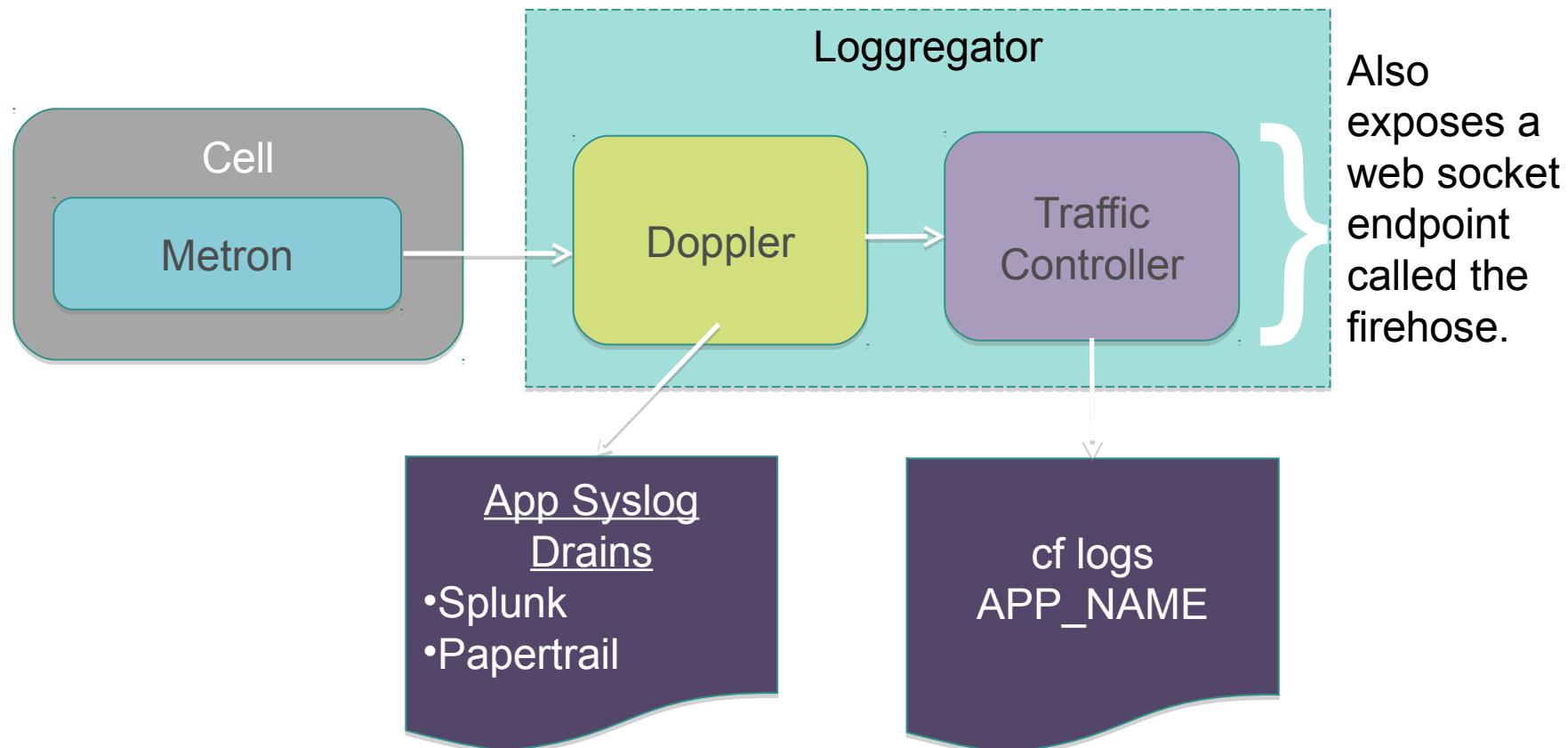
# Loggregator: Doppler

- Gathers logs from Metron



# Loggregator: Traffic Controller

- Handles client requests for logs

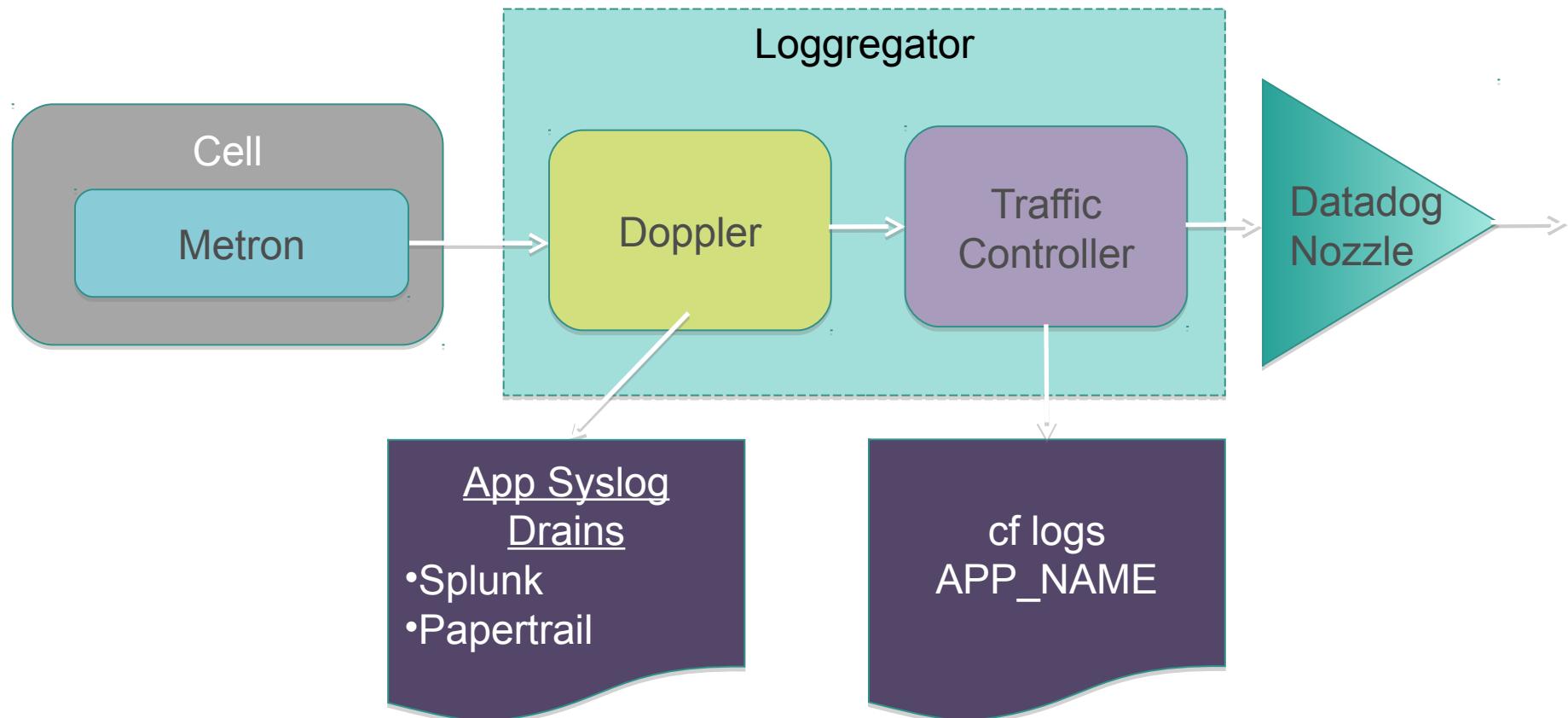


# Loggregator: Firehose

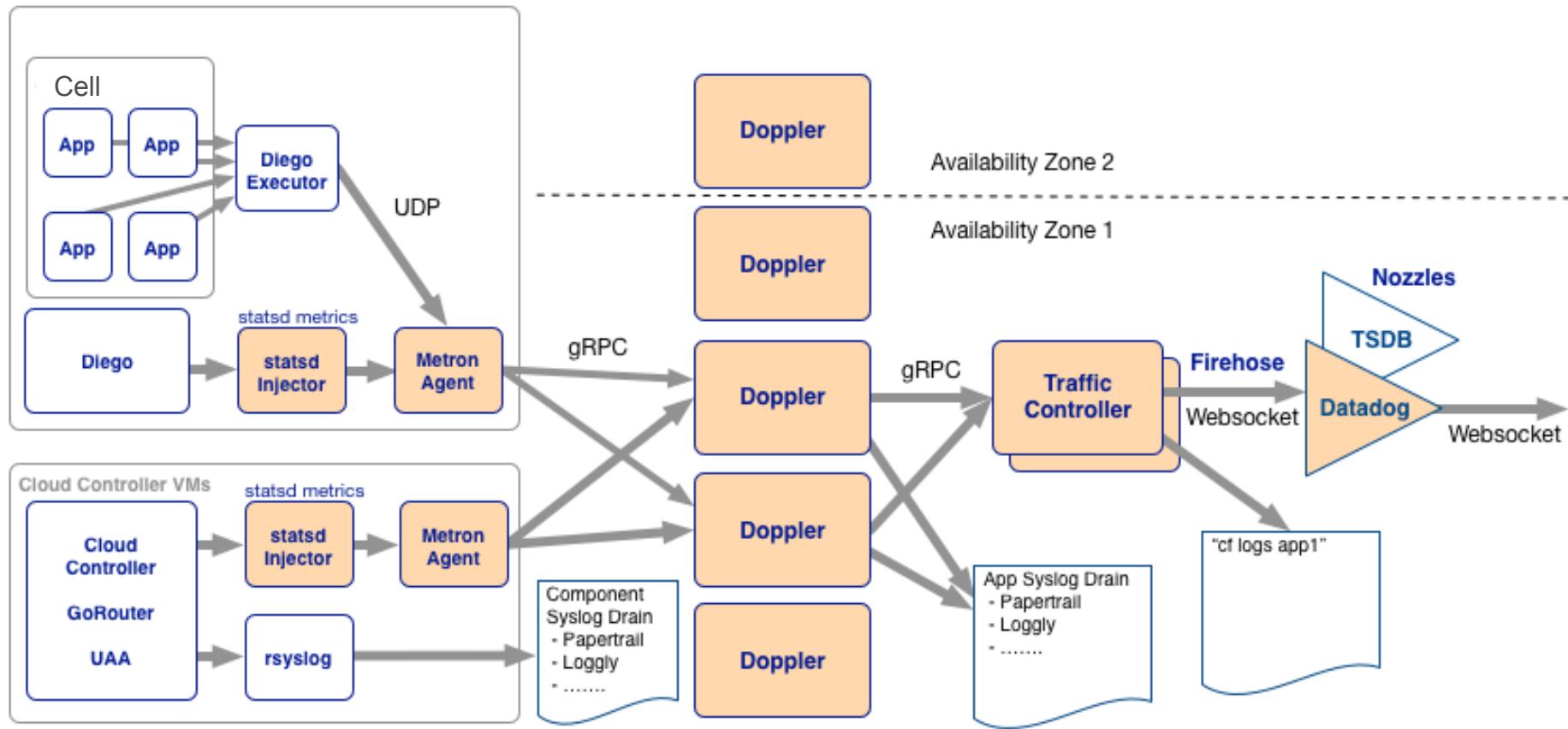
- A websocket endpoint that exposes app logs, container metrics and PAS component metrics
- *Does not include PAS component logs*

# Loggregator: Nozzles

- Consume the firehose output



# Full System Diagram

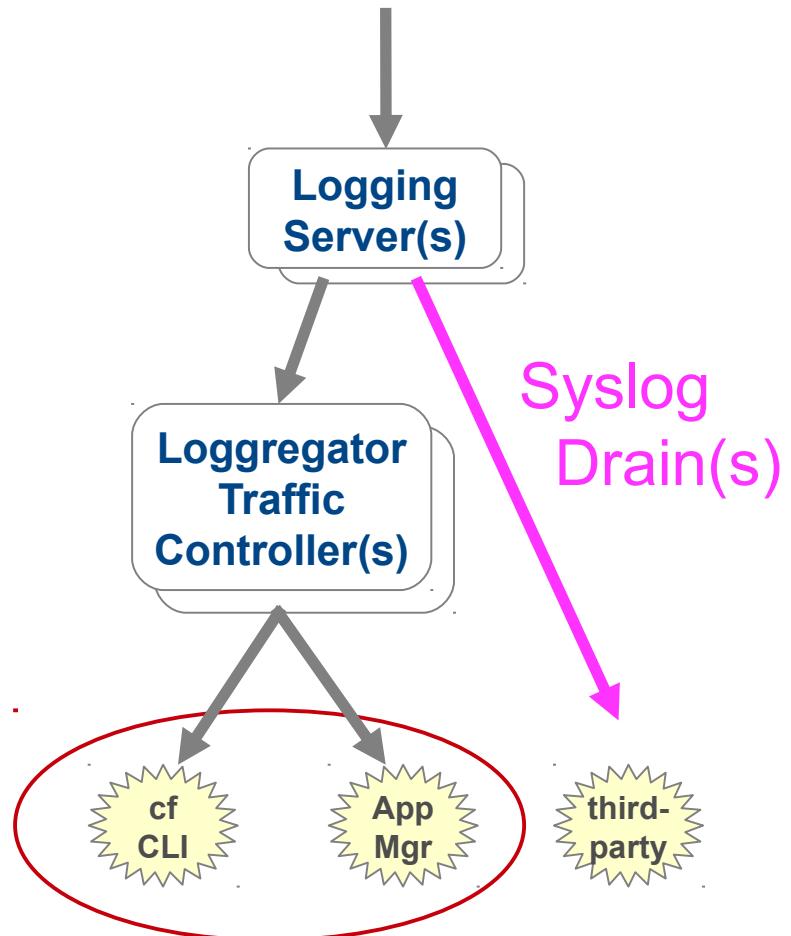


# Agenda

- Logging Architecture
- **Application Logging**

# 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

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

## Log Type / Origin Code

- API – Cloud Controller
  - A change in application state
- APP/PROC|TASK/?/#
  - LRP: PROC, Task: TASK
  - LRP: WEB, Task: task name
  - Application Instance no.
- RTR – Router
- CELL – Execution Agent
- etc.

# 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 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-8080"]
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

# Lab

Logging and scaling an application

# 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

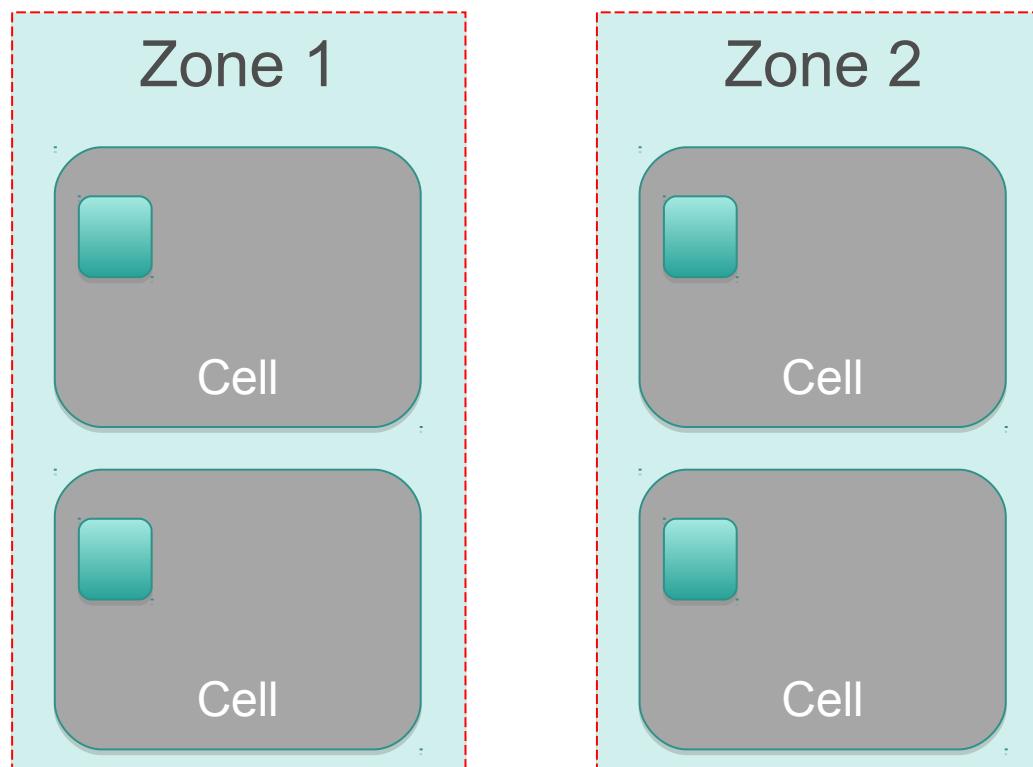
- High Availability in Cloud Foundry

# 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

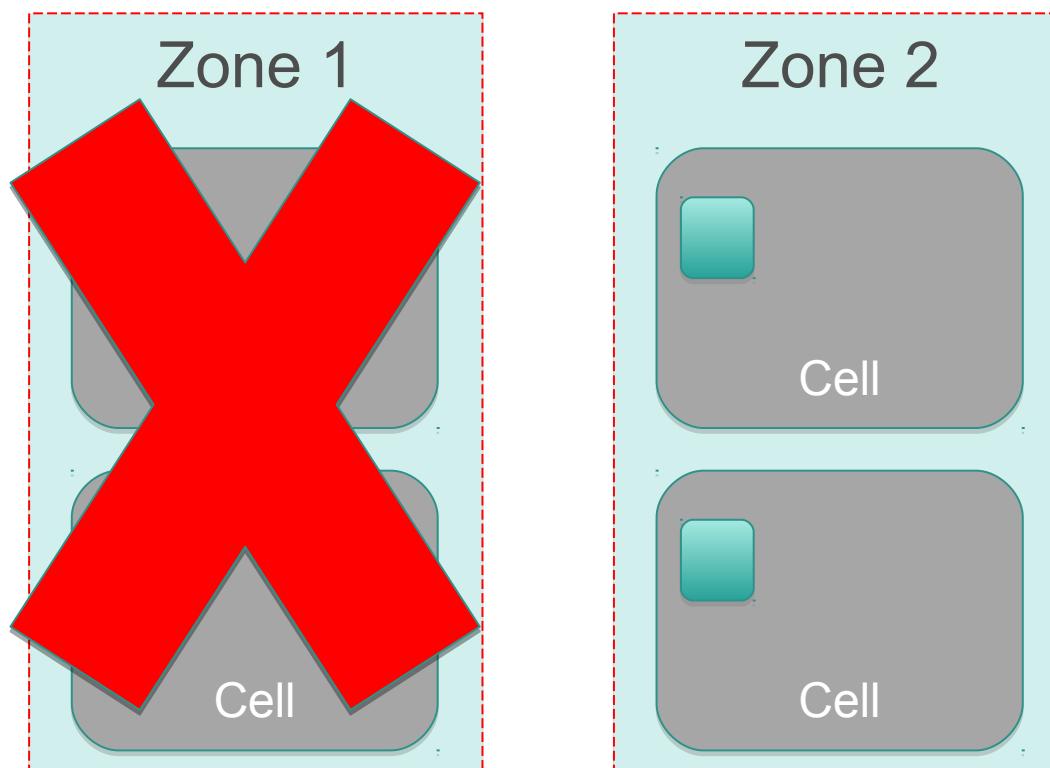
# Availability Zones

- Application instances are evenly distributed across availability zones.



# Availability Zones

- Application stays up despite losing an AZ

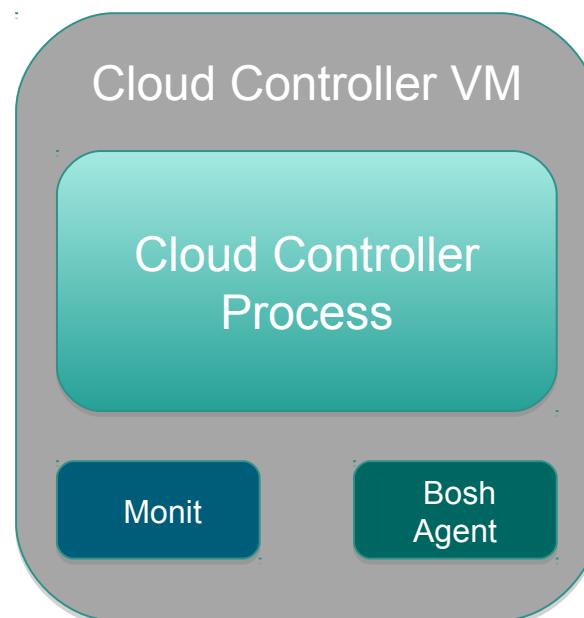


# Availability Zones and the PAS

- Internal components *also* spread across AZs
  - PAS 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

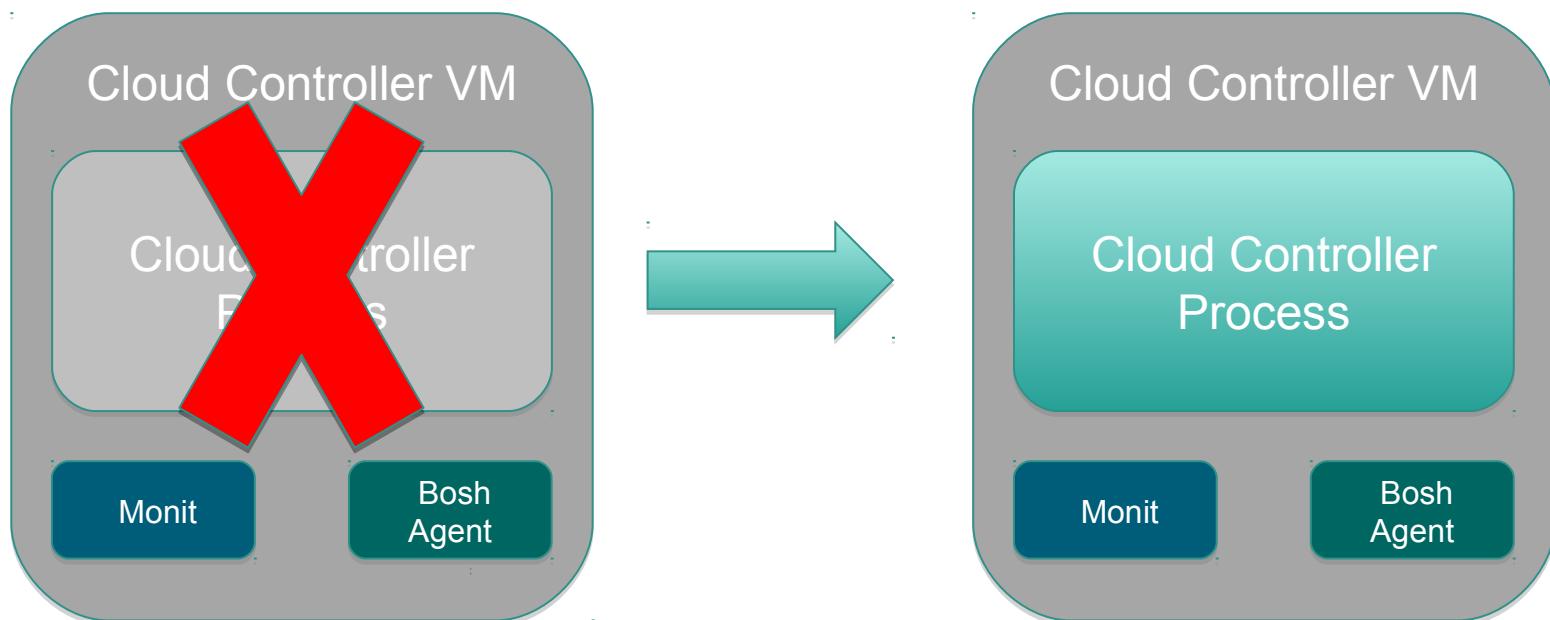
# BOSH Managed Processes

- Key processes are monitored and automatically restarted
  - For example the Cloud Controller



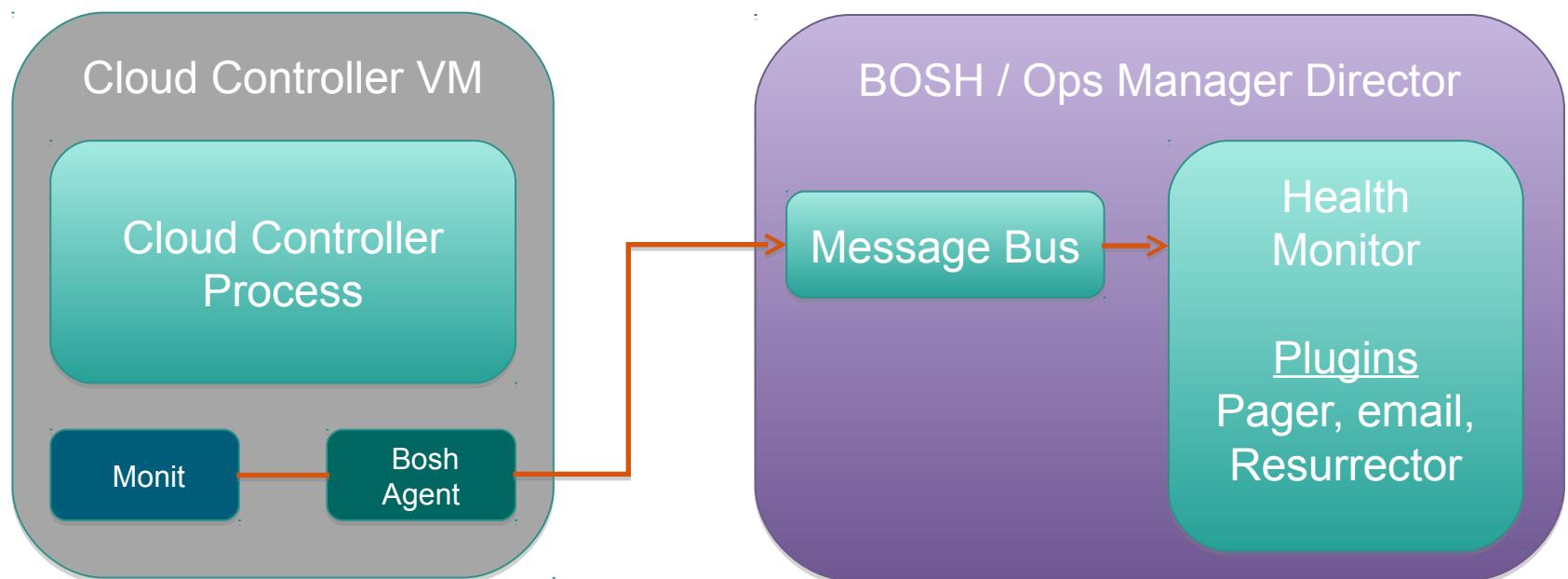
# BOSH Managed Processes

- Key processes are monitored and automatically restarted



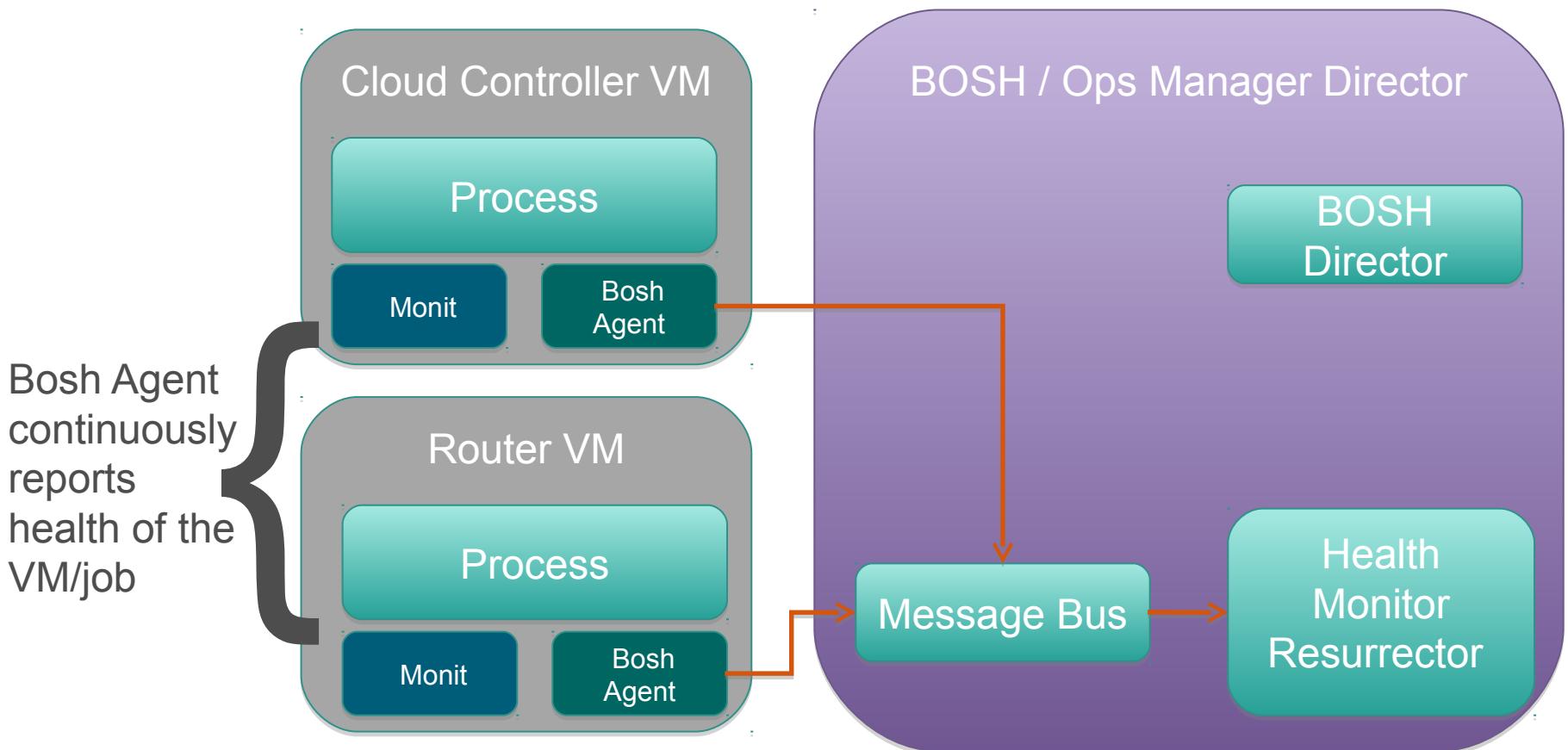
# BOSH Managed Processes

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



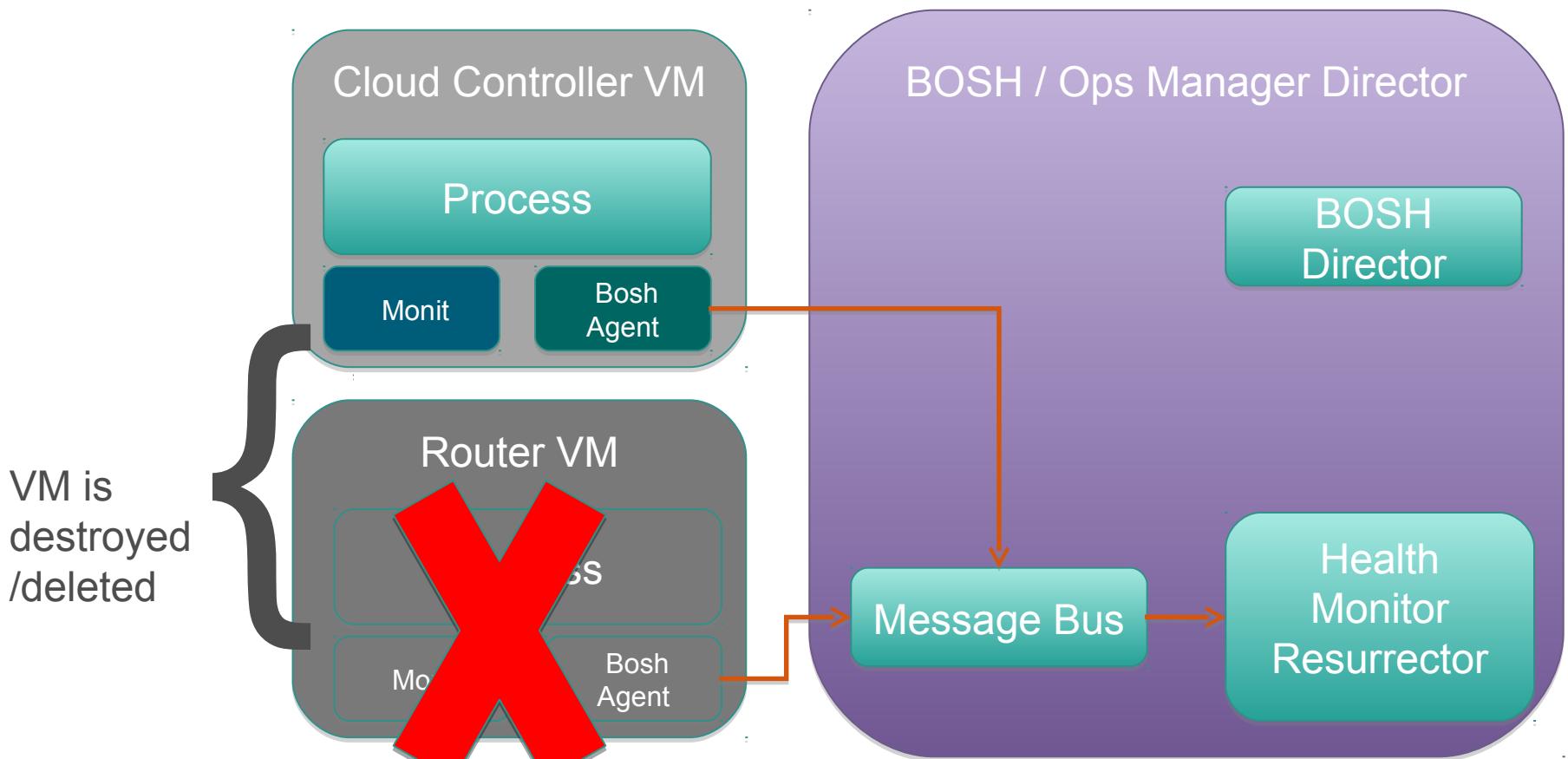
# Failed VMs

- Failed VMs will be recreated automatically.



# Failed VMs

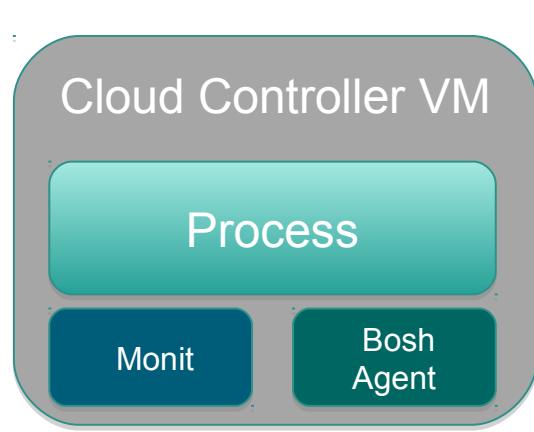
- Failed VMs will be recreated automatically.



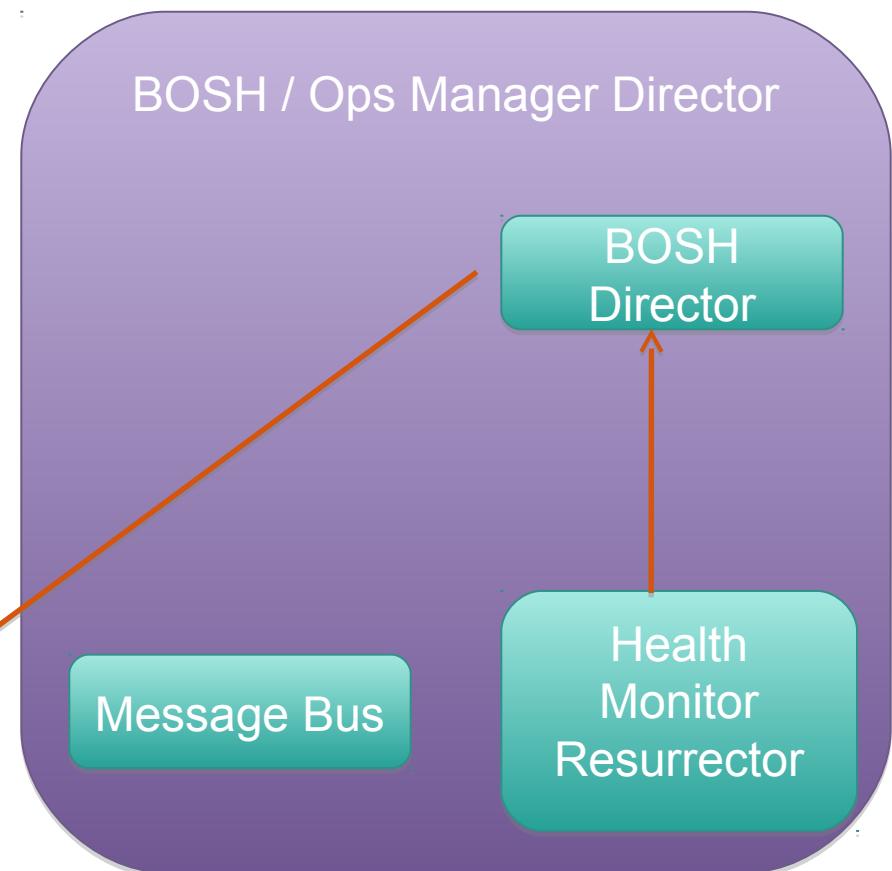
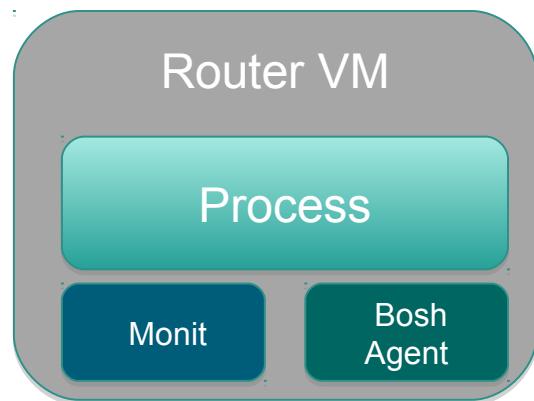
# Failed VMs

- Failed VMs will be recreated automatically

2) Director recreates VM/Job

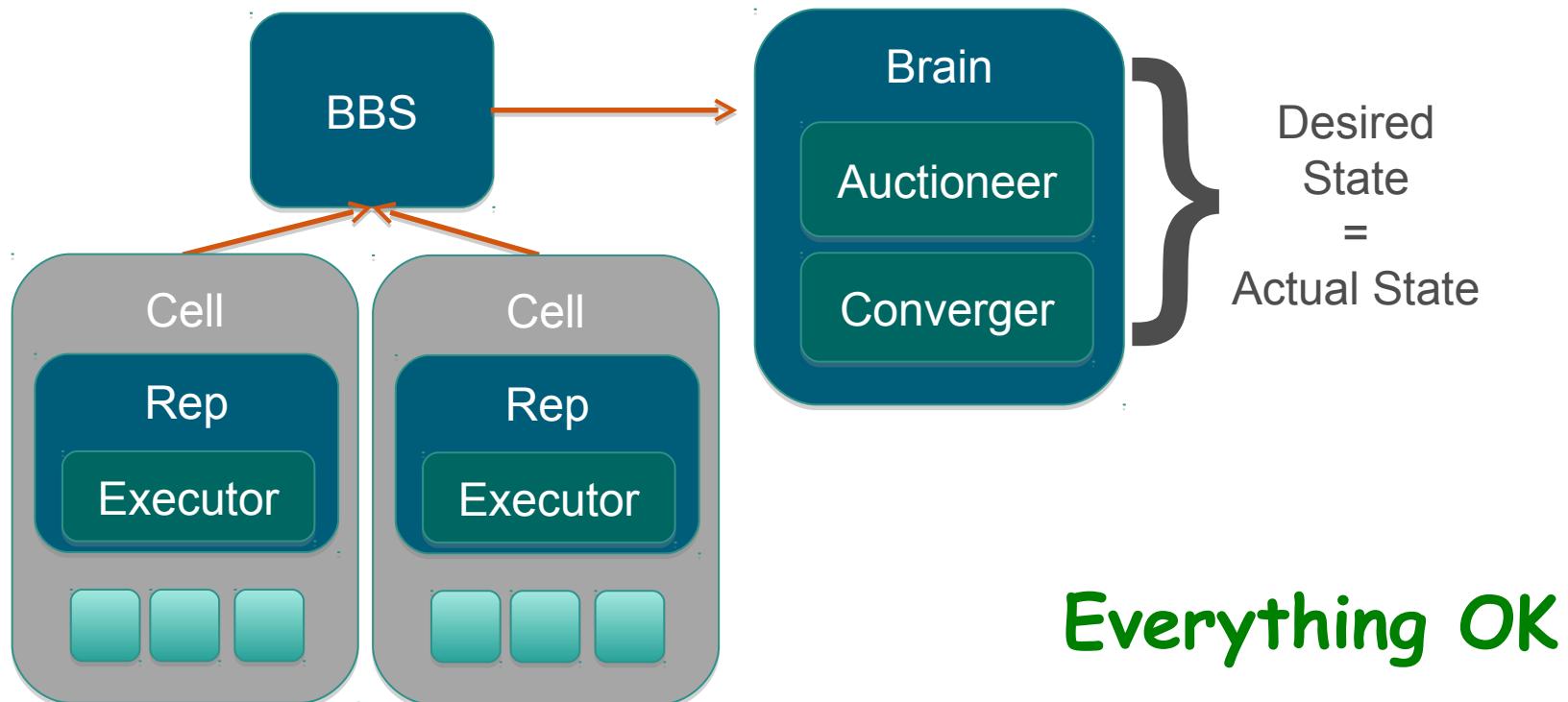


1) Resurrector determines there is a missing VM



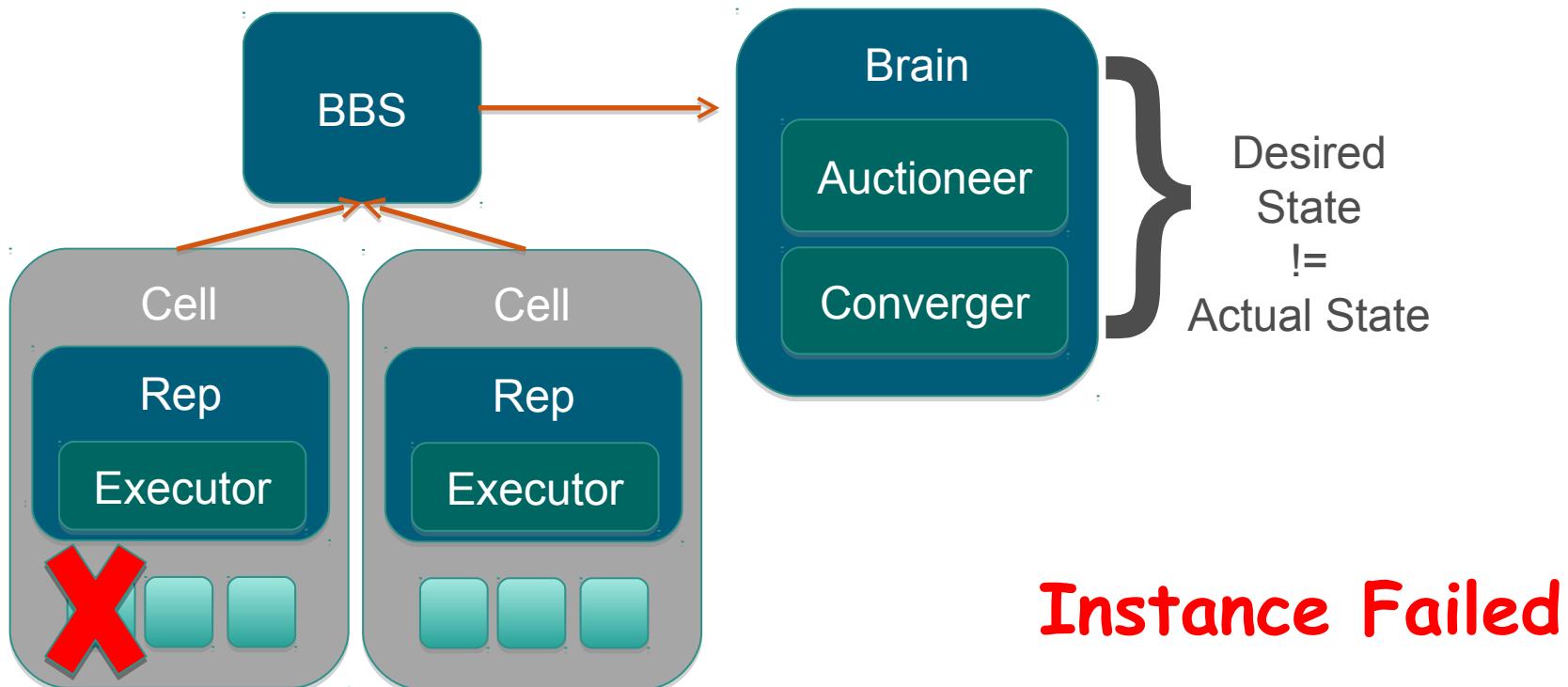
# Self Healing Application Instances

- Once running, failed application instances will be recreated



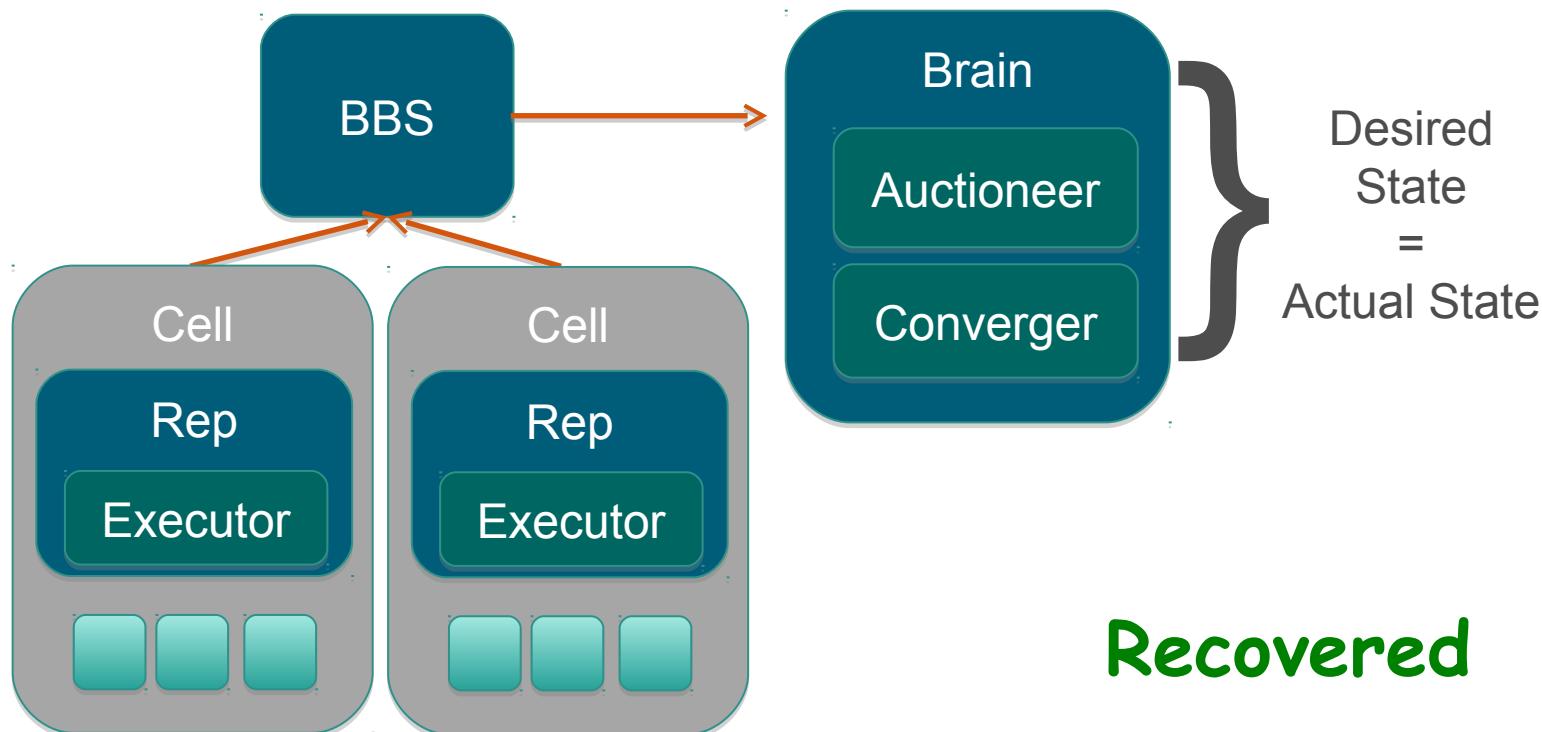
# Self Healing Application Instances

- 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

# Services

What are they?

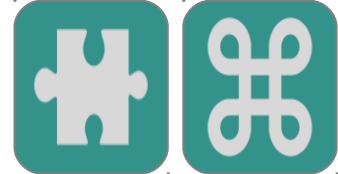
Creating and binding

# 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

# 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



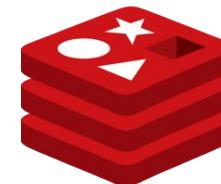
# Roadmap

- 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

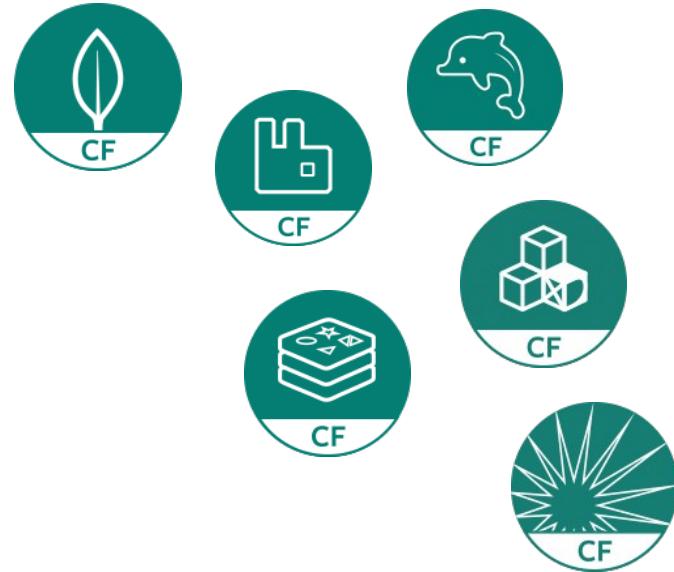
- Typically installed by PCF Operators as “Tiles”
  - Each tile sets up service by deploying its service broker

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. Below this is a "Import a Product" button and links to download PCF compatible products from Pivotal Network. The main area is the "Installation Dashboard" which displays several service tiles:

- Ops Manager Director for VMware vSphere (v1.5.2.0)
- Pivotal Application Service (v2.0.0)
- 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 within the App Manager Console. The left sidebar displays the organization 'krueger-net' and spaces 'development', 'production', 'QA', 'staging', and 'Marketplace'. The main content area is titled 'Marketplace' and includes a search bar. Below it, a section titled 'Services' lists several services:

- 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
```

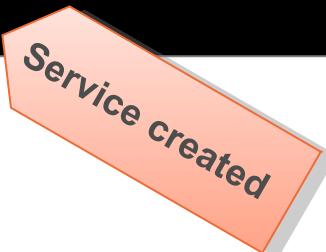
# Finding Existing Service Instances

## 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. On the left, a sidebar menu includes links for Org, Spaces (development, production, staging), Marketplace, Docs, Support, Tools, Blog, and Status. The main area displays the 'pivotaledu > development' space. At the top, it shows the space name 'development' with metrics: RUNNING (1), STOPPED (3), and CRASHED (0). Below this, tabs for Apps (4), Service (1), Routes (6), Members (3), and Settings are visible. The 'Service (1)' tab is selected, showing a table with one row:

Service	Name	Bound Apps	Plan
ClearDB MySQL Database	mydb	0	free - Spark DB

Two orange callout boxes are overlaid on the interface: one pointing to the 'Marketplace' link in the sidebar with the text 'Marketplace', and another pointing to the 'ADD A SERVICE' button with the text 'Services'.

# Finding Available Services

## Service Selection

The screenshot shows the Pivotal Web Services Marketplace interface. On the left, there's a sidebar with navigation links: ORG (pivotedu), SPACES (development, production, staging), Marketplace (which is selected and highlighted in blue), Docs, Support, Tools, Blog, and Status. The main content area has a header "pivotaledu > Marketplace" and a sub-header "Marketplace". It says "Get started with our free marketplace services. Upgrade select plans to gain access to premium service plans." Below this is a search bar with the placeholder "Search" and the letter "l". A section titled "Services ^" lists several services with their icons and descriptions:

- App Autoscaler**  
Scales bound applications in response to load
- ClearDB MySQL Database**  
Highly available MySQL for your Apps.
- ElephantsQL**  
PostgreSQL as a Service
- Gluon CloudLink**  
Mobile Synchronization and Cloud Integration
- MongoDB**  
A simple MongoDB service broker implementation
- PubNub**  
Build Realtime Apps that Scale
- Scheduler for PCF**  
Scheduler service

A large orange arrow points diagonally across the service list, containing the text "Choose to check available plans".

# 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), and 'Marketplace' (selected). The main content area shows the 'ElephantSQL' service page under 'pivotaledu > Marketplace > ElephantSQL'. The service details include a logo of an elephant, the name 'ElephantSQL', the description 'PostgreSQL as a Service', and links to 'Docs' and 'Support'. To the right, there's an 'ABOUT THIS SERVICE' section stating 'The most advanced open-source database, hosted in the cloud.' and a 'COMPANY' section for '84codes AB'. Below this, a list of plans is shown: 'Tiny Turtle' (free), 'Pretty Panda' (\$19.00/MONTH), 'Happy Hippo' (\$99.00/MONTH), and 'Enormous Elephant' (\$499.00/MONTH). A red callout box with the text 'Choose the plan' points to the 'SELECT THIS PLAN' button next to the 'Tiny Turtle free' plan.

Plan	Cost	Features
Tiny Turtle	free	• Shared high performance cluster • 20 MB data • 4 concurrent connections
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. On the left, the sidebar includes sections for ORG (pivotaledu), SPACES (development, production, staging), Marketplace (selected), Docs, Support, Tools, Blog, and Status. The main content area displays the Marketplace page for ElephantSQL, which is described as "PostgreSQL as a Service". It features a "Tiny Turtle" plan, which is free and includes a shared high-performance cluster, 20 MB data, and 4 concurrent connections. The "Instance Configuration" form on the right allows creating a new instance named "mypg" in the "development" space, with the option to bind it to an app (set to "[do not bind]"). Buttons for "SHOW ADVANCED OPTIONS", "CANCEL", and "CREATE" are present. A large orange callout bubble in the bottom right corner contains the text "Create the service".

# Provisioning a new Service Instance Complete

The screenshot shows the Pivotal Web Services dashboard. On the left, a sidebar menu includes links for ORG, pivotaledu, SPACES (development, production, staging), Marketplace, Docs, Support, Tools, Blog, and Status. The main content area is titled "pivotaledu > development". It displays a success message: "Service instance 'mypyg' was successfully created". Below this, it shows the state of the "development" space: 1 RUNNING, 3 STOPPED, 0 CRASHED. The "Services" tab is selected, showing two services: ElephantSQL (mypyg) and ClearDB MySQL Database (mydb). A large orange arrow points from the top right towards the "Success" message. Another orange arrow points from the bottom right towards the "Service available" status in the "Services" table.

Service	Name	Bound Apps	Plan
ElephantSQL	mypyg	0	free - Tiny Turtle
ClearDB MySQL Database	mydb	0	free - Spark DB

# 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 interface for creating a new service instance. On the left, the sidebar includes sections for Org (pivotededu), Spaces (development, production, staging), Marketplace (selected), Docs, Support, Tools, Blog, and Status. The main content area displays the ElephantSQL service details: SERVICE ElephantSQL (PostgreSQL as a Service), ABOUT THIS SERVICE (The most advanced open-source database, hosted in the cloud.), and COMPANY (84codes AB). A sidebar on the right lists service plans: Tiny Turtle (free) with options: Shared high performance cluster, 20 MB data, 4 concurrent connections. The central form is titled 'Instance Configuration' and contains fields for 'Instance Name' (set to mypg), 'Add To Space' (development), and 'Bind To App (Optional)' (set to [do not bind]). Two orange callout boxes provide instructions: one pointing to the 'Instance Name' field with the text 'Specify application name', and another pointing to the 'CREATE' 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

- More information may available
  - Open Service details page in App Manager
  - Try clicking on Support and/or Manage links

mydb  
SERVICE: ClearDB MySQL Database PLAN: Spark DB

Overview Plan Settings Docs Support Manage

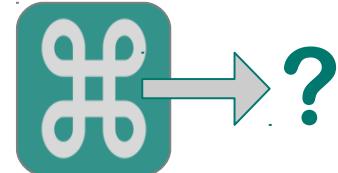
Bound Apps **BIND APP**

No Apps have been bound to this Service

# 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

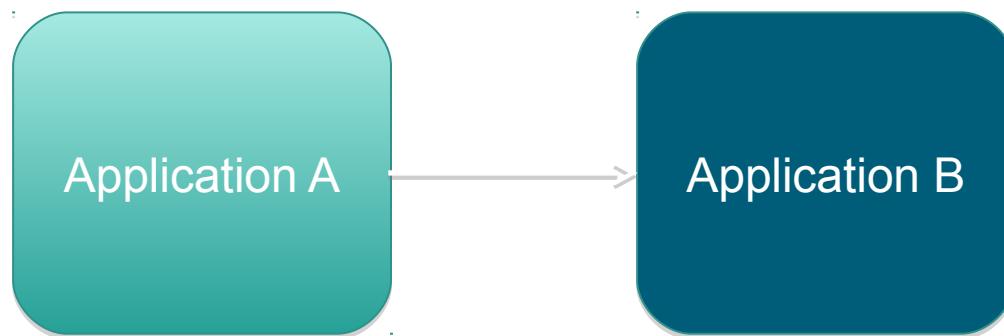
# User Provided Services – 1

- **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 fill in the corresponding fields when prompted.

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



# Designing Applications for Cloud Foundry

## Writing Applications that Scale

Getting it right *By Design*

Pivotal

# Overview

- After completing this lesson, you should be able to:
  - Consider design considerations for Cloud Foundry

# Roadmap

- 12-Factor Applications

# Developer Topics / Application Architecture

- Applications may require adjustments to run successfully in Cloud environment
  - See *Developer Topics*

<http://docs.cloudfoundry.org/devguide/deploy-apps/prepare-to-deploy.html>

# 12-Factor Application

- <http://12factor.net>
- Outlines architectural principles for modern apps
  - Focus on scaling, continuous delivery, portable, and cloud ready

# 12-Factor Application

## I. Codebase

One codebase tracked in SCM, many deploys

## II. Dependencies

Explicitly declare and isolate dependencies

## III. Configuration

Store config in the environment

## IV. Backing Services

Treat backing services as attached resources

## V. Build, Release, Run

Strictly separate build and run stages

## VI. Processes

Execute app as stateless processes

## VII. Port binding

Export services via port binding

## VIII. Concurrency

Scale out via the process model

## IX. Disposability

Maximize robustness with fast startup and graceful shutdown

## X. Dev/prod parity

Keep dev, staging, prod as similar as possible

## XI. Logs

Treat logs as event streams

## XII. Admin processes

Run admin / mgmt tasks as one-off processes

# 12-Factor Application

## I. Codebase

One codebase tracked in SCM, many deploys

## II. Dependencies

Explicitly declare and isolate dependencies

## III. Configuration

Store config in the environment

- **Codebase**
  - An application has a single codebase
    - Multiple codebase = distributed system (not an app)
      - Each component in a codebase can (should) be an app
  - Tracked in version control
    - Git, Subversion, Mercurial, etc.
  - Multiple Deployments
    - i.e. development, testing, staging, production, etc.

# 12-Factor Application

## I. Codebase

One codebase tracked in SCM, many deploys

## II. Dependencies

Explicitly declare and isolate dependencies

## III. Configuration

Store config in the environment

- Dependencies
  - Packaged as jars (Java), RubyGems, CPAN (Perl)
  - Declared in a Manifest
    - Maven POM, Gemfile / bundle exec, etc.
  - No reliance on specific system tools
    - i.e. Linux tool not available on Windows

# 12-Factor Application

## I. Codebase

One codebase tracked in SCM, many deploys

## II. Dependencies

Explicitly declare and isolate dependencies

## III. Configuration

Store config in the environment

- Configuration
  - Separate from the code
  - Also separate from the application
    - i.e. DB credentials, hostnames, passwords
    - Acid Test – could the codebase be made open source?
    - Internal wiring (i.e. Spring configuration) considered part of codebase.
  - Environment Variables recommended.

# 12-Factor Application

## IV. Backing Services

Treat backing services as attached resources

## V. Build, Release, Run

Strictly separate build and run stages

## VI. Processes

Execute app as stateless processes

- Backing Services
  - Service consumed by app as part of normal operations
    - DB, Message Queues, SMTP servers
    - May be locally managed or third-party managed
  - Services should be treated as resources
    - Connected to via URL / configuration
    - Swappable (change in-memory DB for MySQL)

# 12-Factor Application

## IV. Backing Services

Treat backing services as attached resources

## V. Build, Release, Run

Strictly separate build and run stages

## VI. Processes

Execute app as stateless processes

- Build, Release, Run
  - Build stage – converts codebase into build (version)
    - Including managed dependencies
  - Release stage – build + config = release
    - Ready to run
  - Run – Runs app in execution environment

# 12-Factor Application

## IV. Backing Services

Treat backing services as attached resources

## V. Build, Release, Run

Strictly separate build and run stages

## VI. Processes

Execute app as stateless processes

- Processes
  - One or more discrete running processes
  - Stateless
    - Processes should not store internal state (HTTP Sessions)
  - Shared Nothing
    - Data needing to be shared should be persisted
  - Memory / local tmp storage considered volatile
  - Processes may intercommunicate via messaging / persistent storage

# 12-Factor Application

## VII. Port binding

Export services via port binding

## VIII. Concurrency

Scale out via the process model

## IX. Disposability

Maximize robustness with fast startup and graceful shutdown

- Port Binding
  - App should not need a “container”
    - Java App Server, Apache HTTPD for PHP ...
    - PaaS now takes that role
  - Apps should export HTTP as a service
    - Define as a dependency (#2)
      - Tornado (Python), Thin (Ruby), embedded Jetty/Tomcat (Java)
    - Execute at runtime
  - One App can become another App's service (#4, #6)

# 12-Factor Application

## VII. Port binding

Export services via port binding

## VIII. Concurrency

Scale out via the process model

## IX. Disposability

Maximize robustness with fast startup and graceful shutdown

- Concurrency
  - Processes are first class citizens
    - Like Unix service daemons
    - Unlike Java threads
  - Individual processes are free to multithread
    - BUT a VM can only get so large (vertical scaling).
    - Must be able to span multiple machines (horizontal scaling)

# 12-Factor Application

## VII. Port binding

Export services via port binding

## VIII. Concurrency

Scale out via the process model

## IX. Disposability

Maximize robustness with fast startup and graceful shutdown

- **Disposability**
  - Processes should be disposable
    - Remember, they're stateless!
  - Should be quick to start and stop
    - Should exit gracefully / finish current requests.
    - Or should be idempotent / reentrant
  - Enhances scalability and fault tolerance
  - Design *crash-only* software

# 12-Factor Application

## X. Dev/prod parity

Keep dev, staging, prod as similar as possible

## XI. Logs

Treat logs as event streams

## XII. Admin processes

Run admin / mgmt tasks as one-off processes

- Development, Staging, Production should be similar
  - Dev / Prod environments often different
    - Tool gap – devs use SQLite/Nginx, prod uses Apache/Oracle
    - Personnel gap – developers develop, admins deploy
    - Time gap - (development over weeks / months)
  - Keep differences minor
    - Reduce tool gap – use same software
    - Reduce time gap - small changes & continuous deployment
    - Reduce personnel gap - involve developers in deployment and monitoring

# 12-Factor Application

## X. Dev/prod parity

Keep dev, staging, prod  
as similar as possible

## XI. Logs

Treat logs as event  
streams

## XII. Admin processes

Run admin / mgmt tasks  
as one-off processes

- Logs are streams of aggregated, time-ordered events
  - Apps are not concerned with log management
    - Just write to sysout.
  - Separate log managers handle management
    - Logging as a service
- Can be managed via tools like Papertrail, Splunk ...
  - Log indexing and analysis

# 12-Factor Application

## X. Dev/prod parity

Keep dev, staging, prod  
as similar as possible

## XI. Logs

Treat logs as event  
streams

## XII. Admin processes

Run admin / mgmt tasks  
as one-off processes

- Admin Processes / Management Tasks Run as One-Off Processes.
  - DB Migrations, one time scripts, etc.
  - Use same environment, tools, language as application processes
    - REPL

*Read–Eval–Print Loop = command-shell  
for running non-interactive shell scripts*

# 12-Factor Application

## I. Codebase

One codebase tracked in SCM, many deploys

## II. Dependencies

Explicitly declare and isolate dependencies

## III. Configuration

Store config in the environment

## IV. Backing Services

Treat backing services as attached resources

## V. Build, Release, Run

Strictly separate build and run stages

## VI. Processes

Execute app as stateless processes

## VII. Port binding

Export services via port binding

## VIII. Concurrency

Scale out via the process model

## IX. Disposability

Maximize robustness with fast startup and graceful shutdown

## X. Dev/prod parity

Keep dev, staging, prod as similar as possible

## XI. Logs

Treat logs as event streams

## XII. Admin processes

Run admin / mgmt tasks as one-off processes

# Summary

- After completing this lesson, you should have learned:
  - Architectural design factors for building scalable applications in Cloud Foundry



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



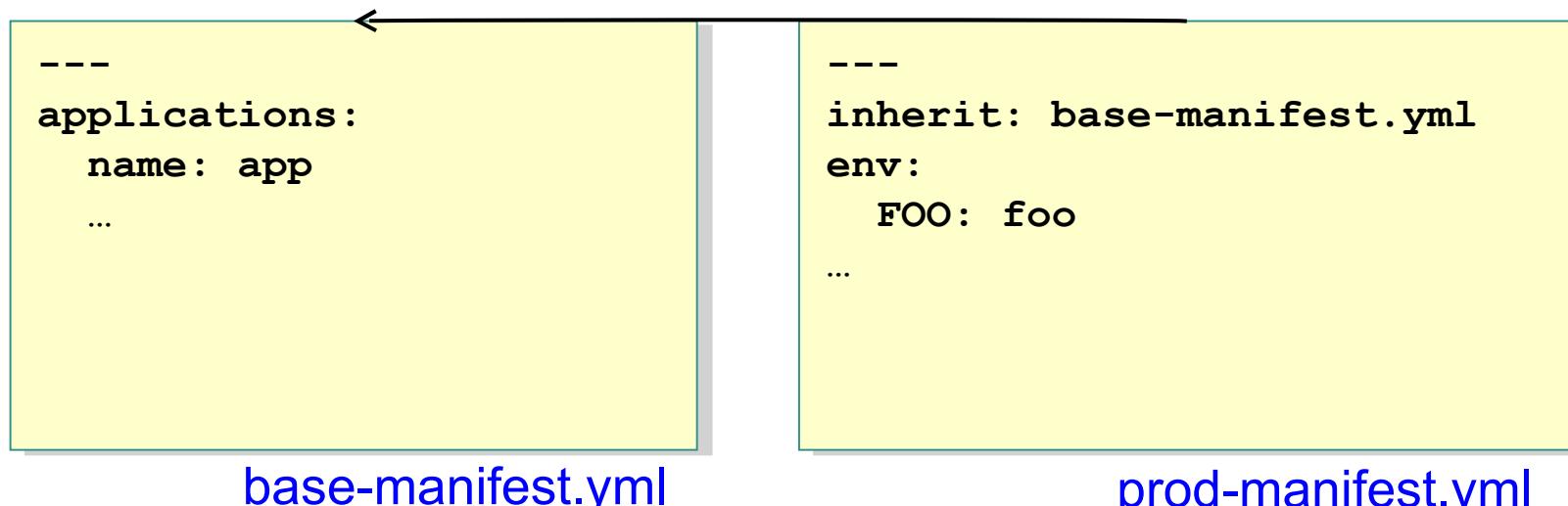
```
---  
applications:  
- name: cf-node-demo  
  memory: 128M  
  instances: 1  
  host: demo-${random-word}  
  domain: cfapps.io  
  path: .
```

- 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



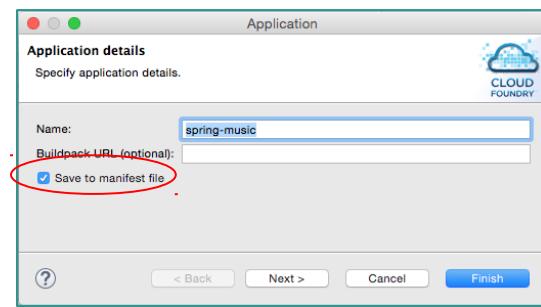
# Manifest Inheritance

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

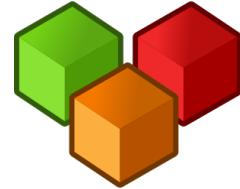


# Manifests and STS

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



The screenshot shows the Pivotal Tooling interface for managing applications. On the left, the 'Applications' panel lists the deployed application 'spring-music'. Below it, the 'Services' panel shows various service instances: elephantsql, feedbackdb-mysql2, logdrain, new-relic, sendmail-dev, and session-replication. The 'General' panel on the right displays application details: Name (spring-music), Mapped URLs (spring-music-pc1.cfapps.io), Instances (1), and Manifest (highlighted with a red circle). The 'General' section also includes Memory Limit (512 MB) and Environment Variables settings. The 'Application Operations' panel contains buttons for Start, Stop, Update and Restart, Push, and Debug. The 'Application Services' and 'Instances' panels are partially visible at the bottom.



# Note on Spaces

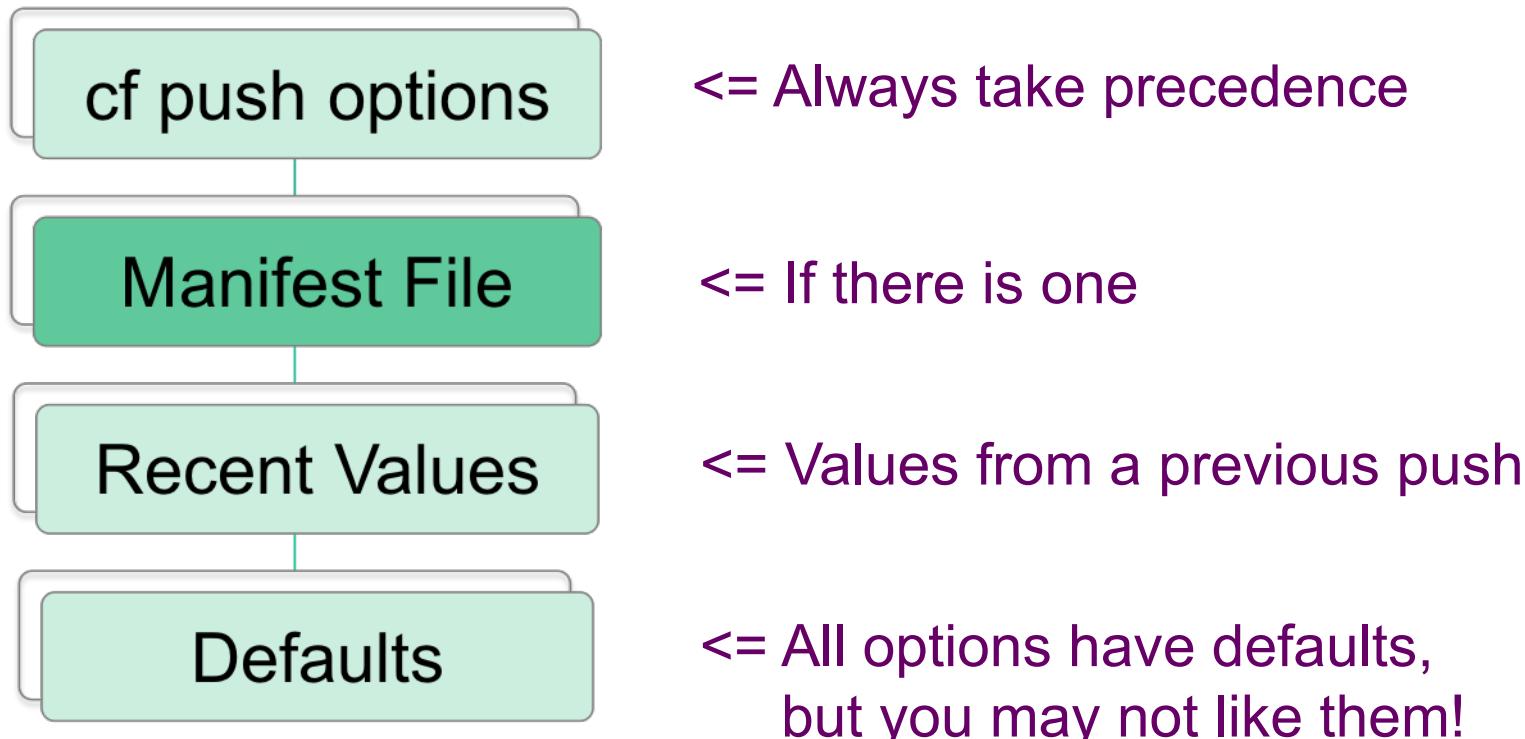
- A Space *cannot* be specified in a manifest file
  - Set first: `cf target -s development`
- To determine the current space
  - Just run: `cf target`



# 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



# 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

# Log Draining

Using a third-party log manager

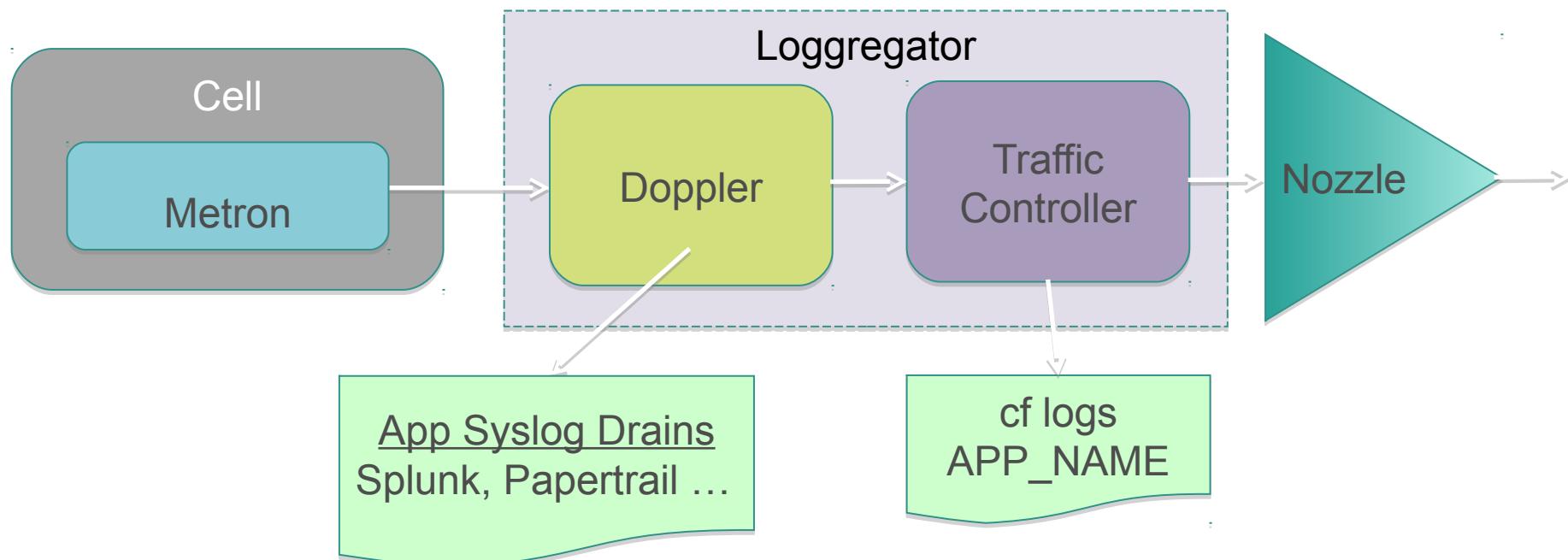
Setting up a syslog drain

# 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™

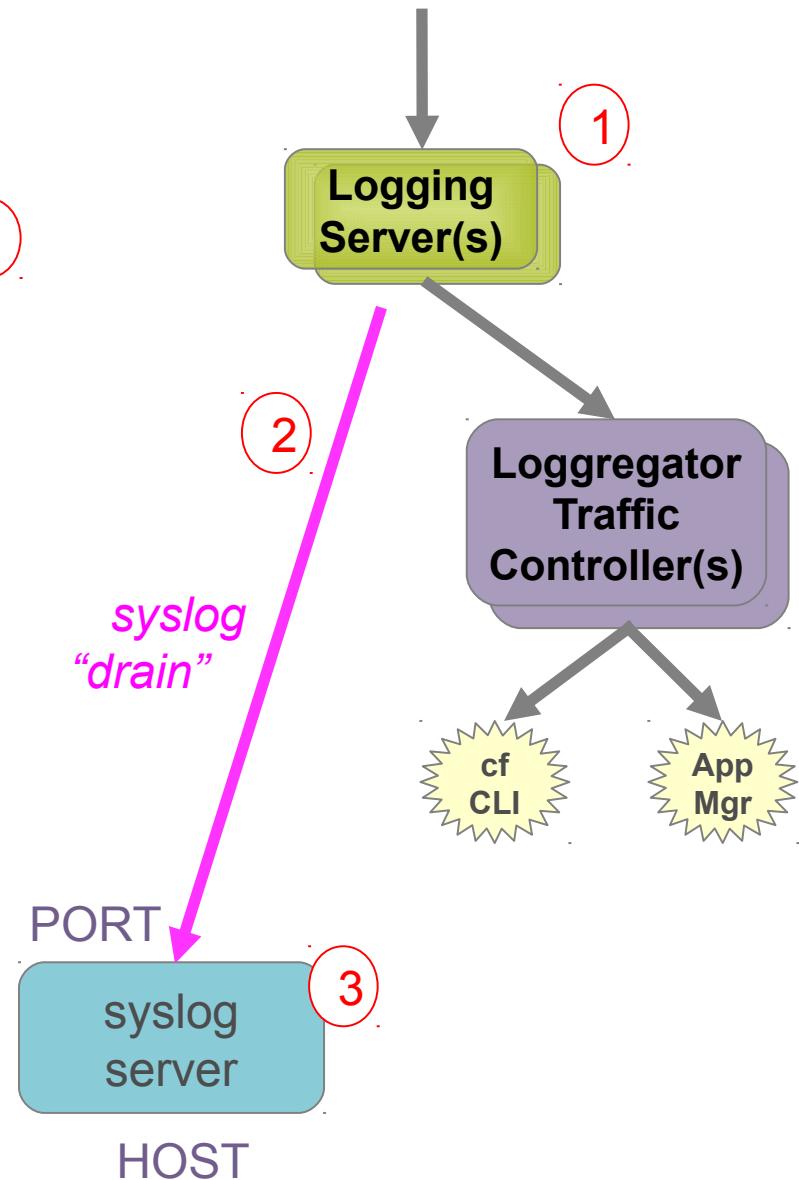
Pivotal™

# 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.

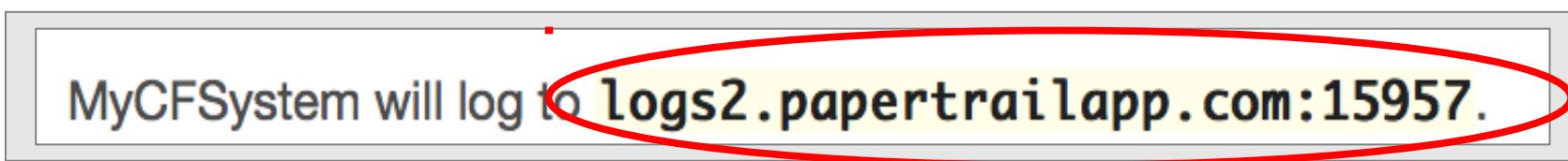
What should we call it?

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

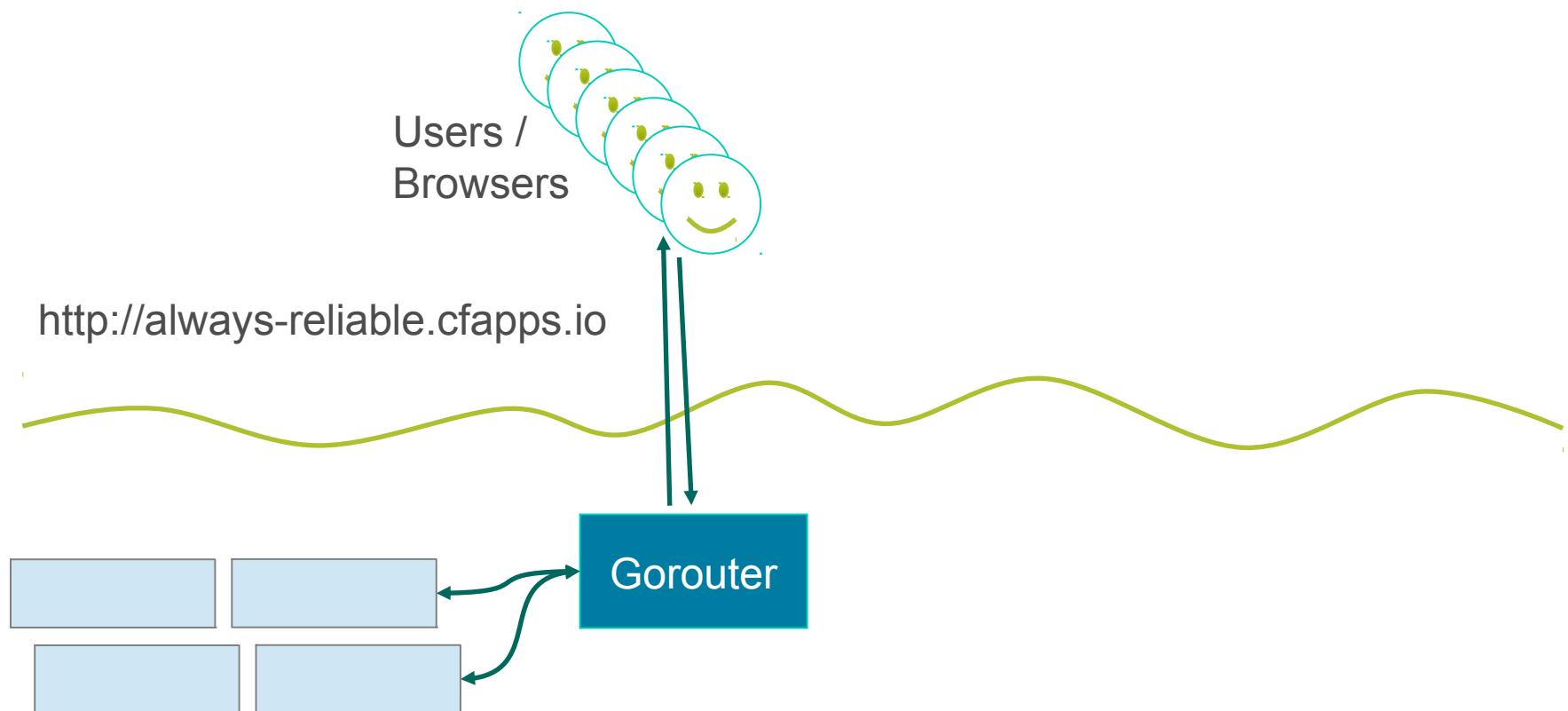
- **Zero-Downtime Deployments**
- Implications for Application & Data Model Design

# 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.

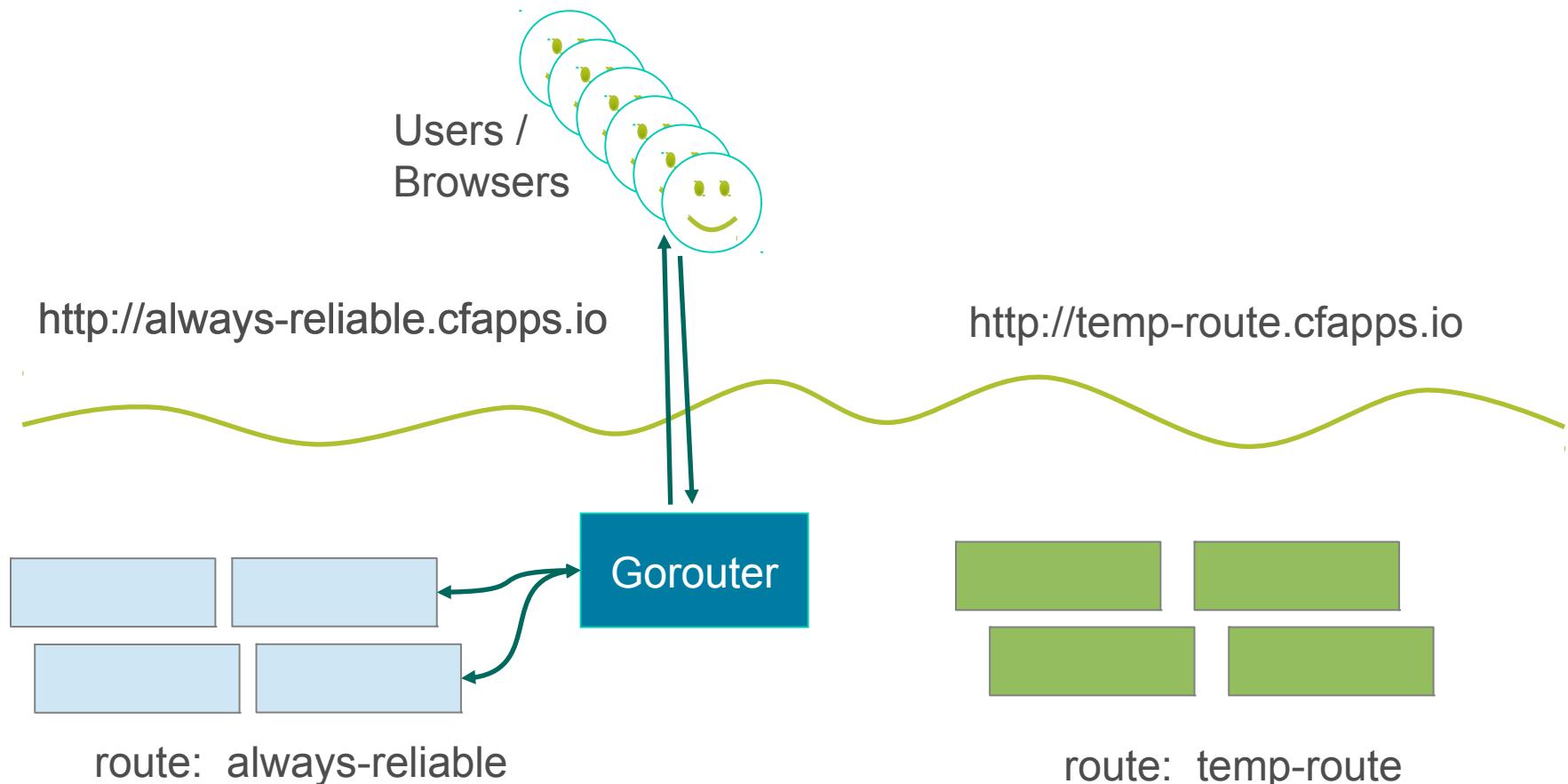
# Blue Green Deployment – Existing App

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



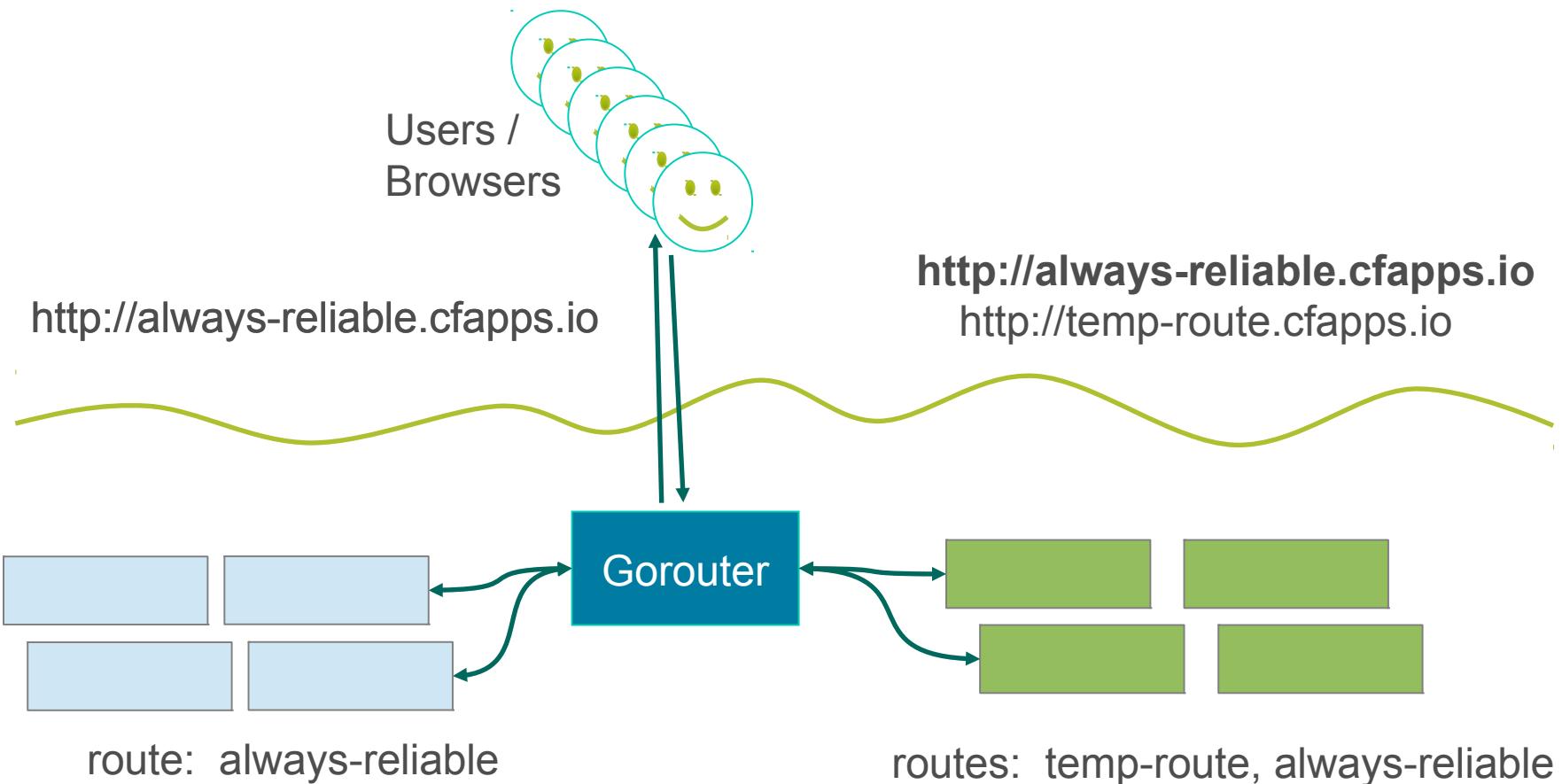
# Blue Green Deployment – New Version

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



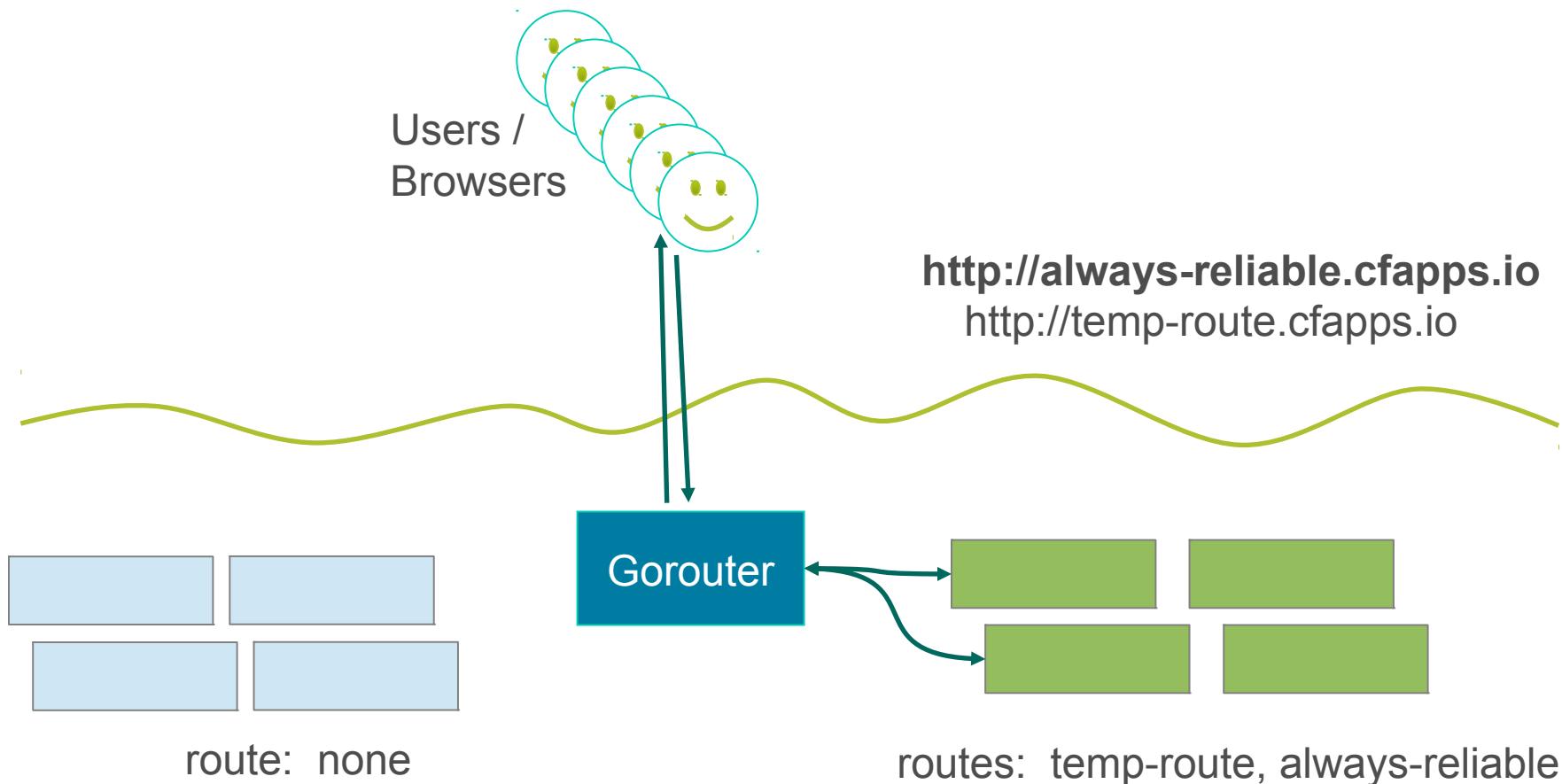
# Blue Green Deployment – Duplicate Route

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



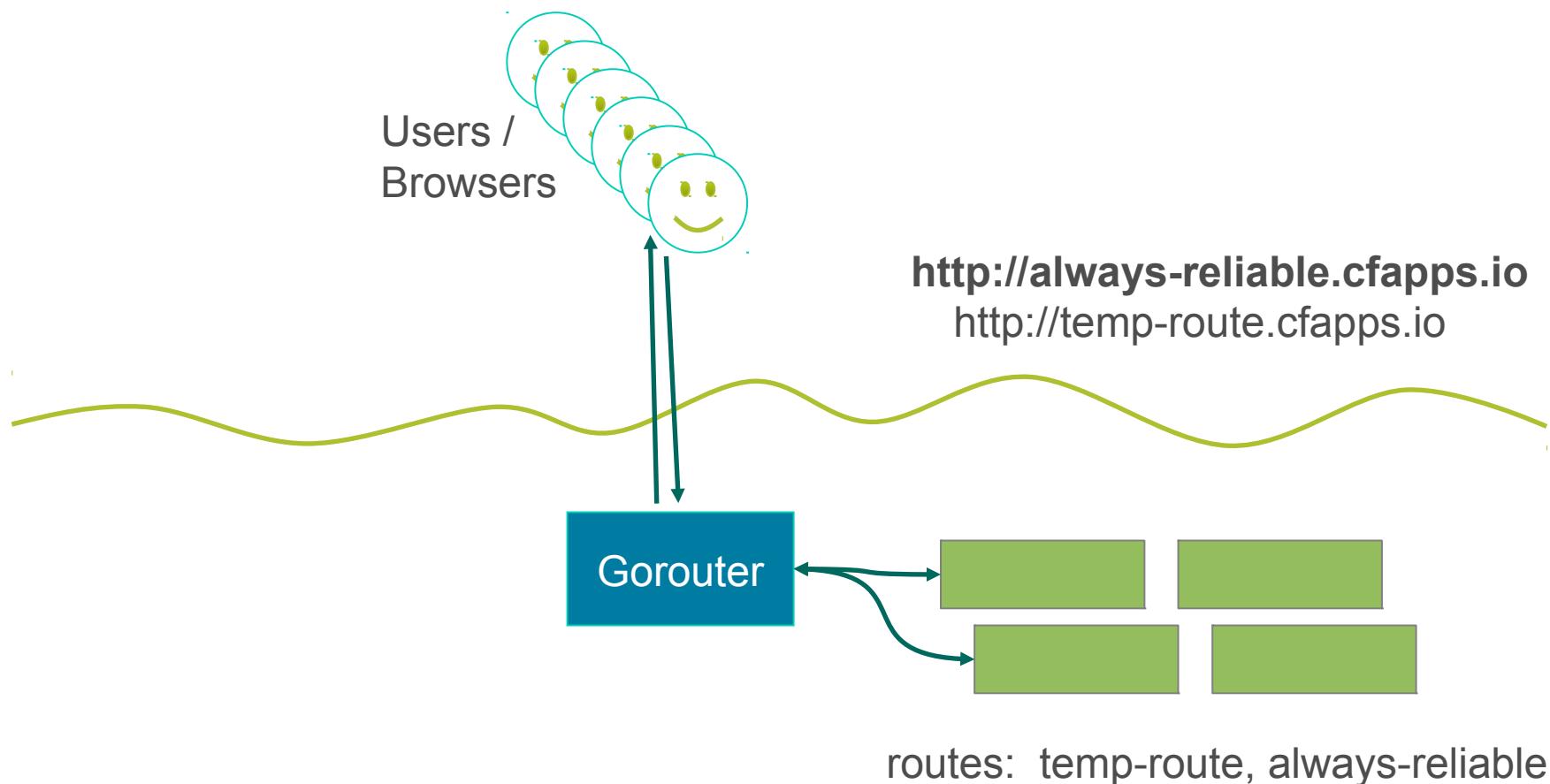
# Blue Green Deployment – Disconnect Blue

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



# Blue Green Deployment – Remove Blue

`cf delete blue`

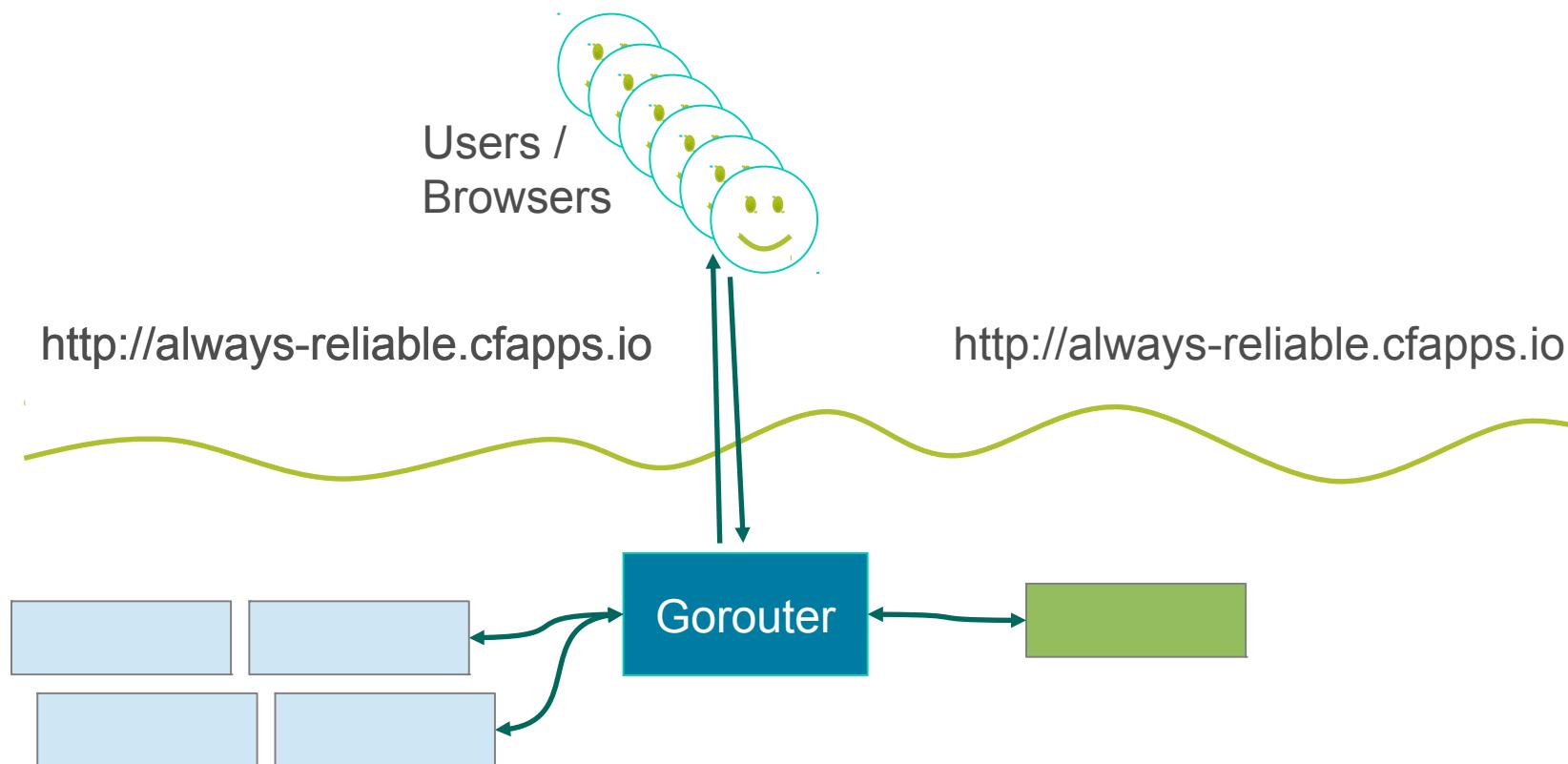


# Canary Deployments

- 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

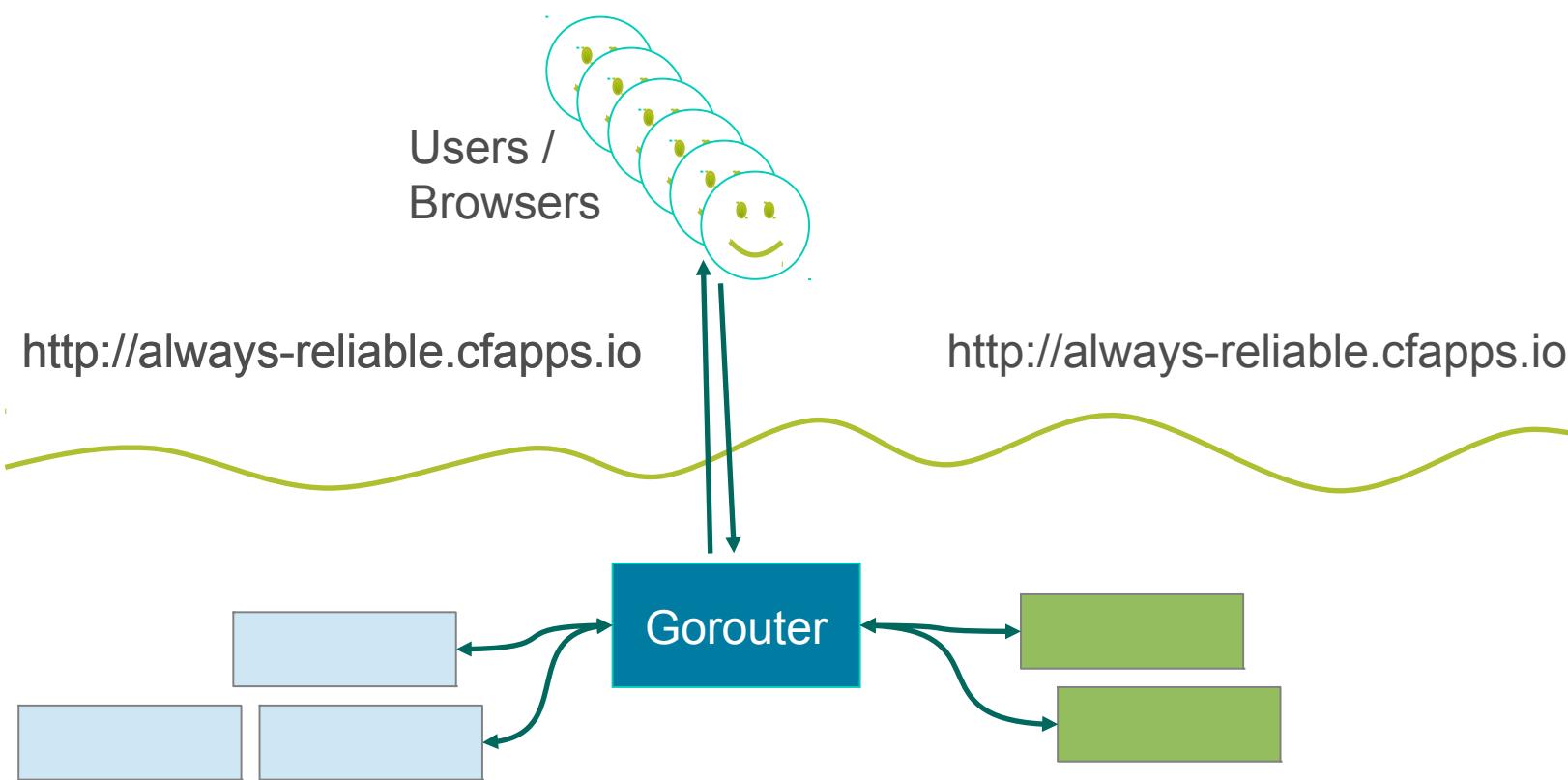
```
cf push green -p app.war -n always-reliable -i 1
```



# Canary Deployment – Scale Traffic

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

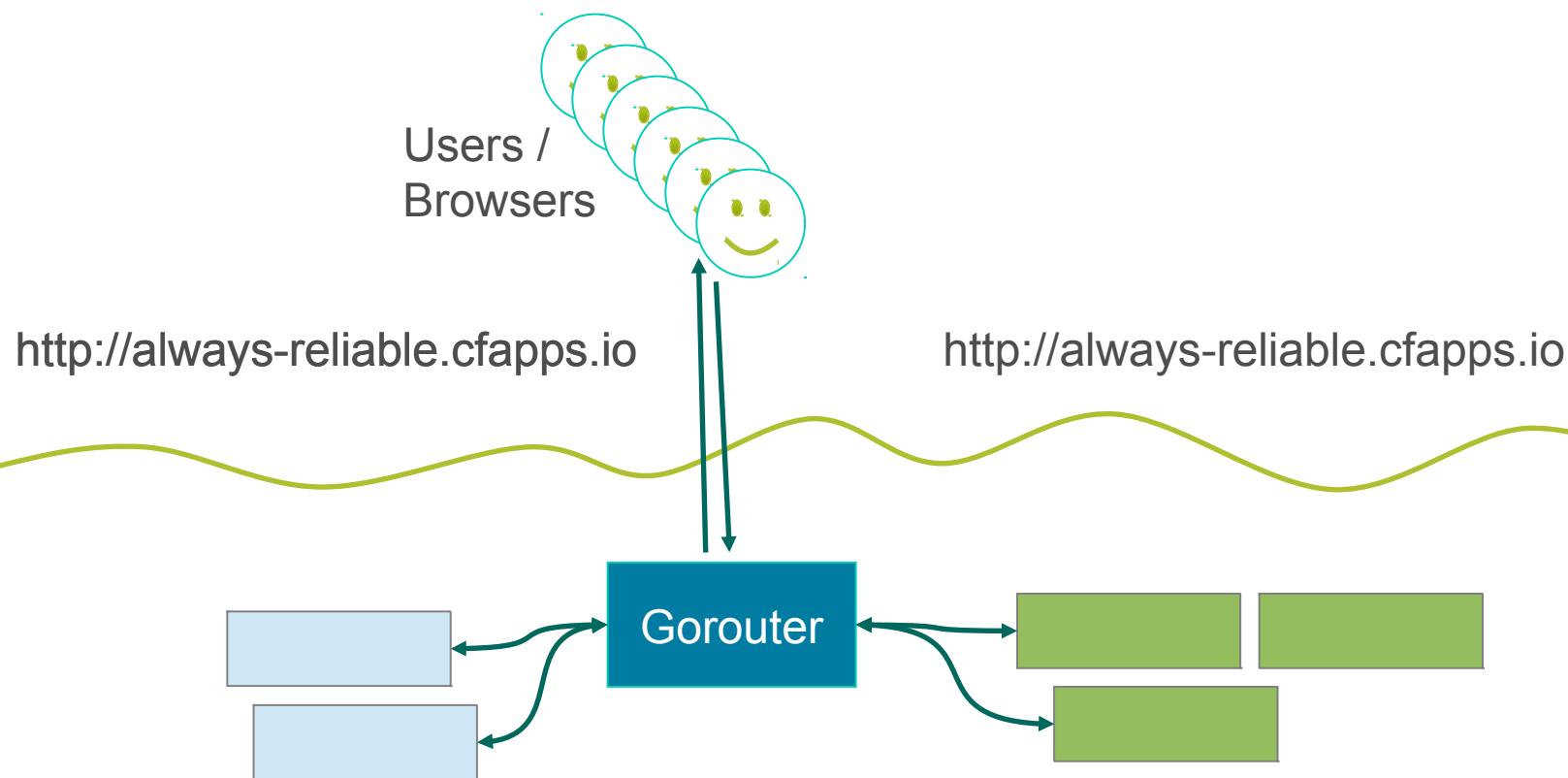
```
cf scale blue -i 3
```



# Canary Deployment – Scale Traffic

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

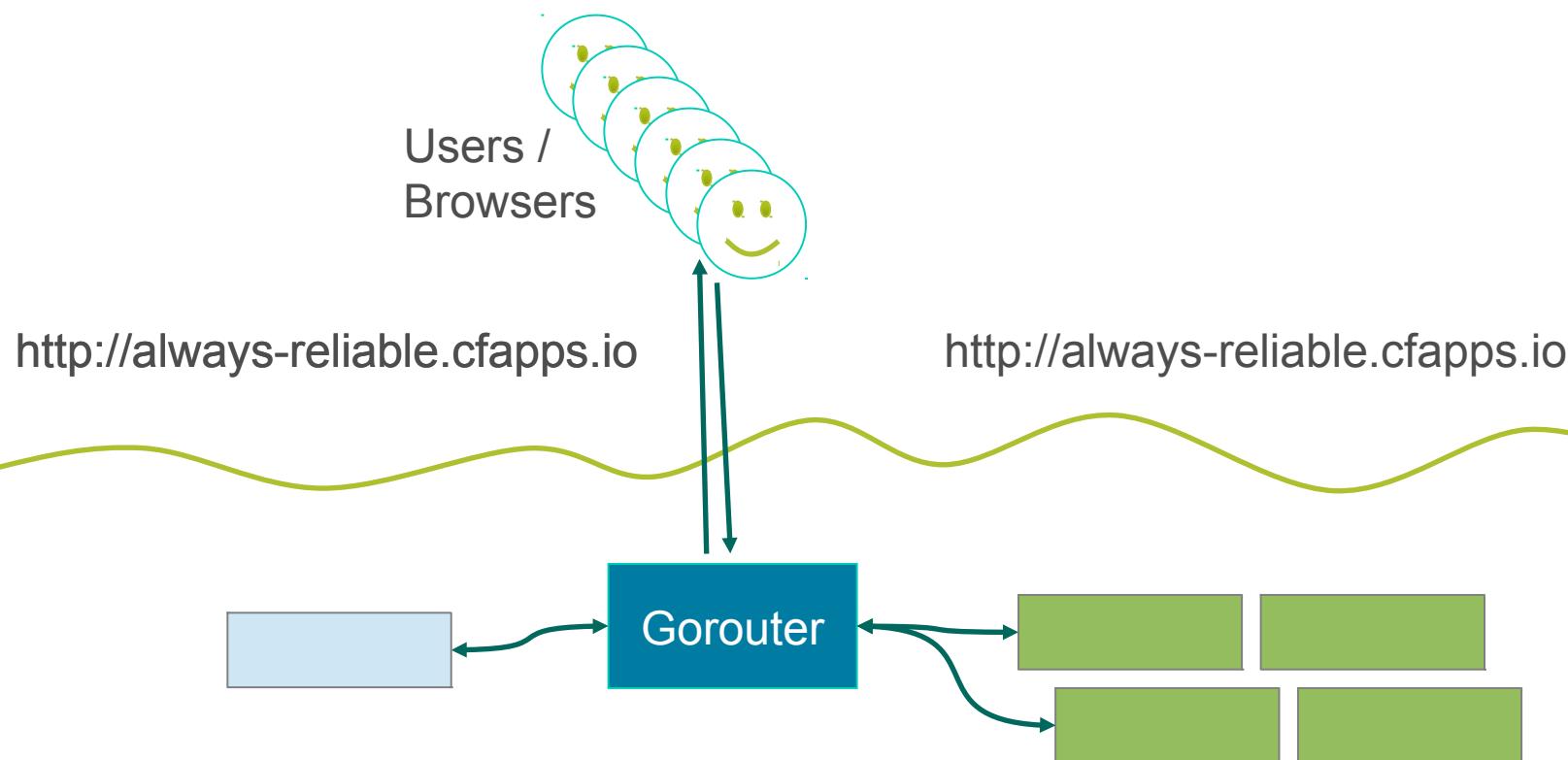
```
cf scale blue -i 2
```



# Canary Deployment – Scale Traffic

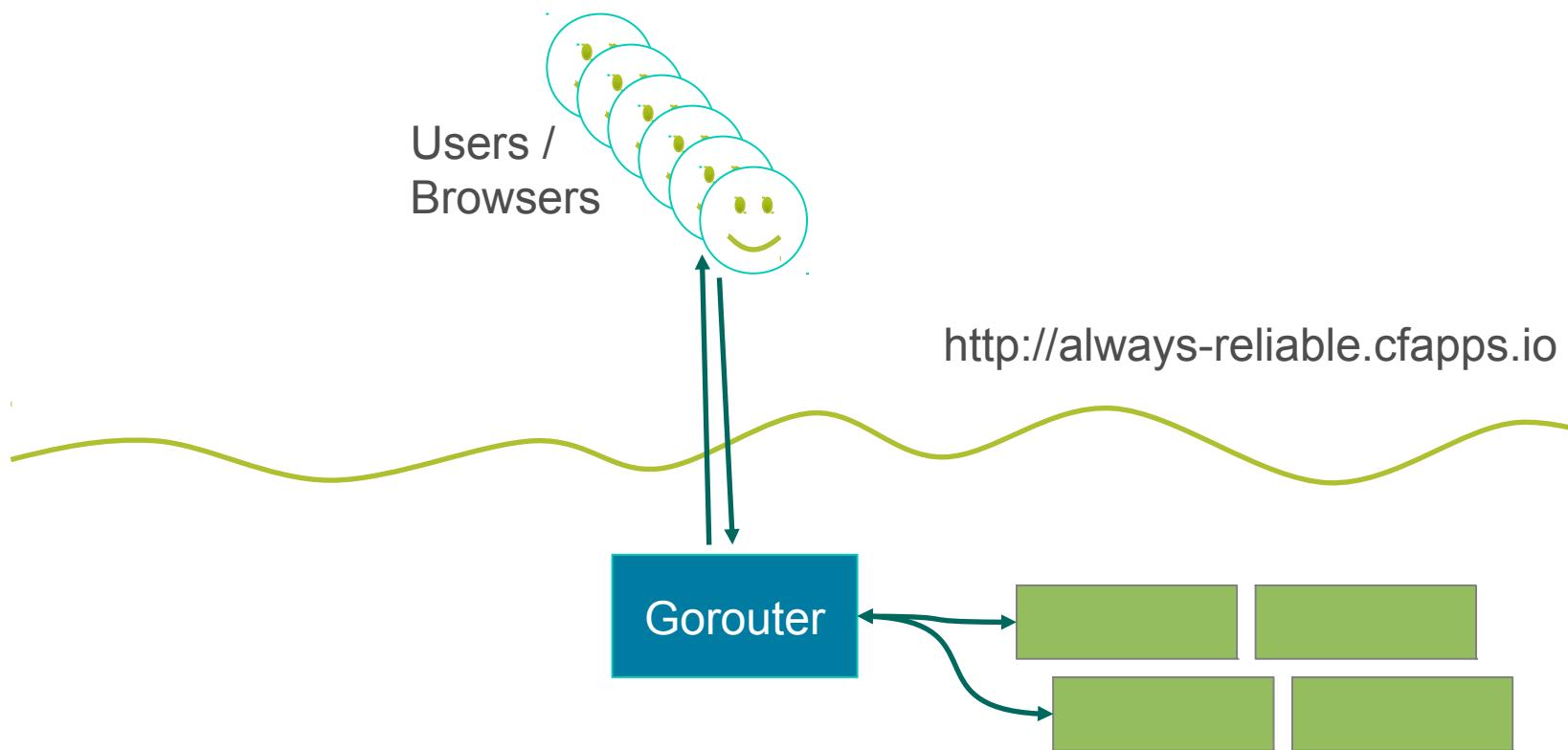
```
cf scale green -i 4
```

```
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

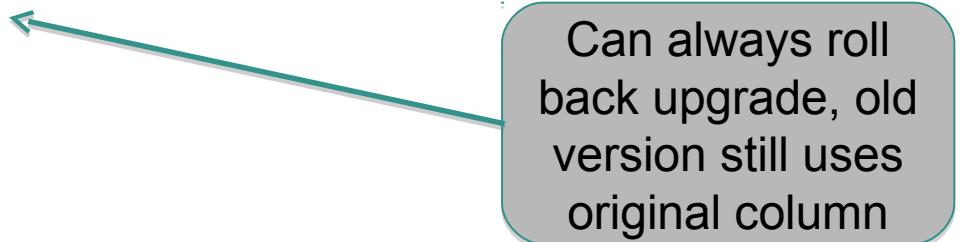
# Database

- 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



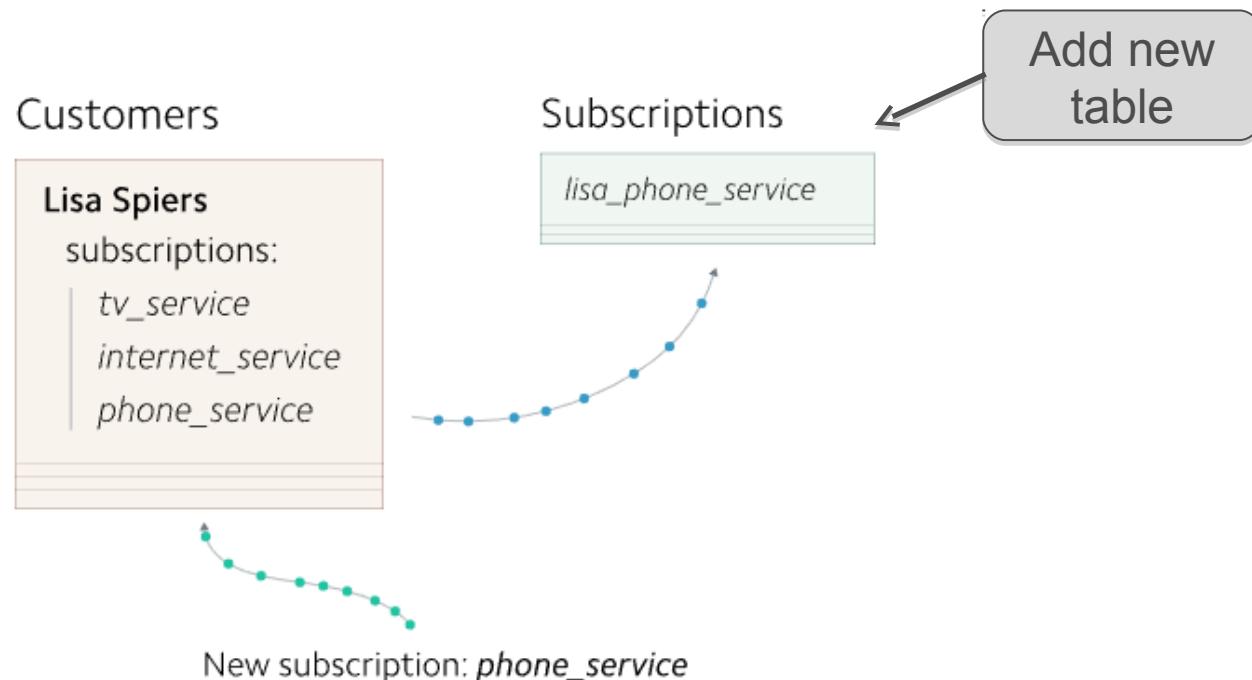
Can always roll  
back upgrade, old  
version still uses  
original column

# 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

# Application Security Groups

Controlling Outgoing Requests

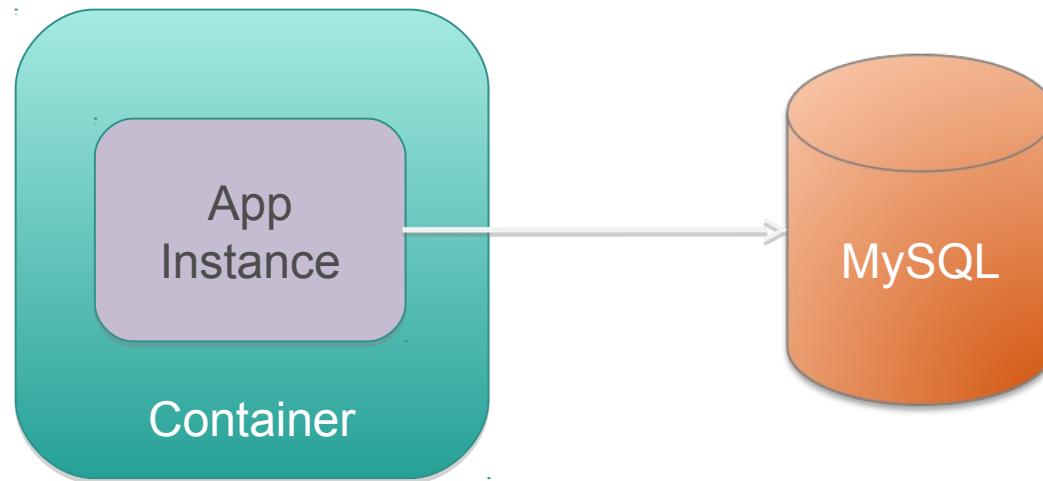
*Administration Facility*

# Agenda

- Application Security Groups Overview

# Application Security Groups

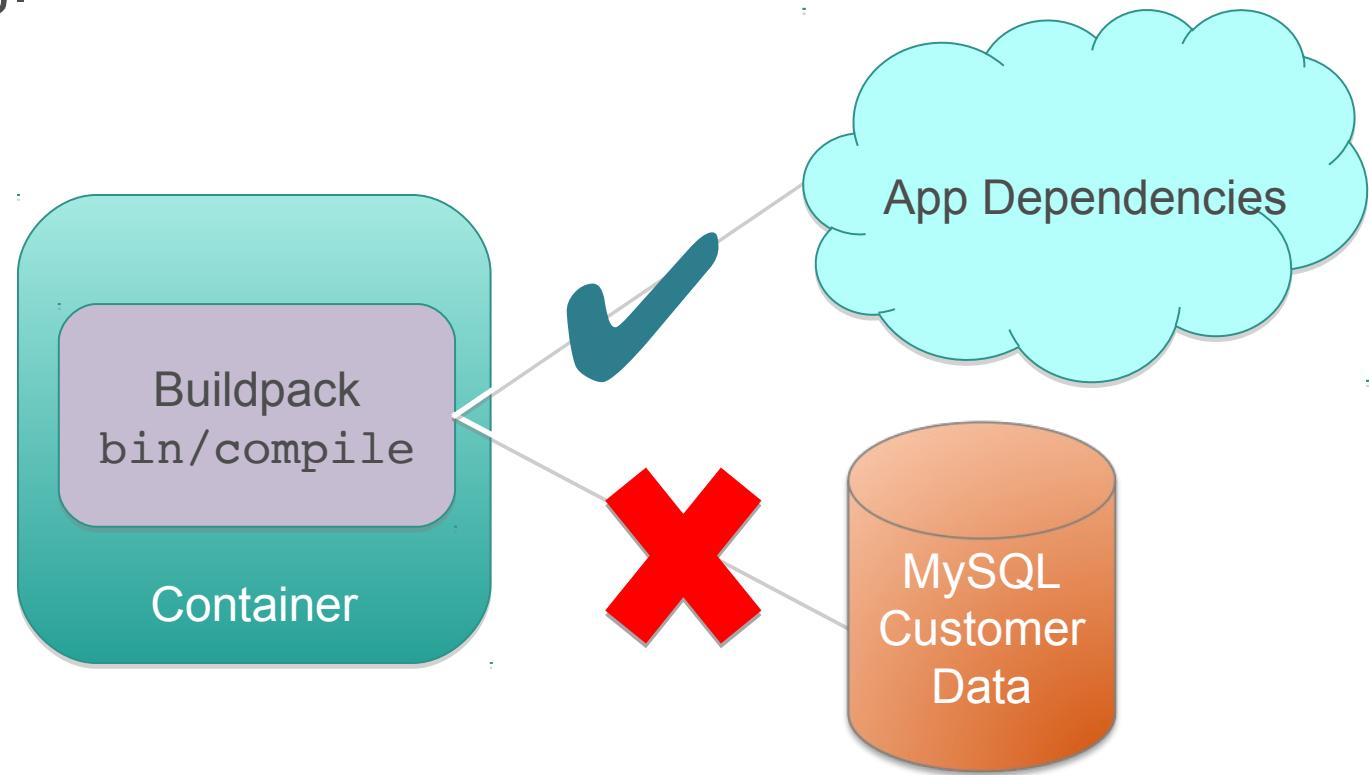
- Are virtual firewalls that control egress/outbound traffic for applications.



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

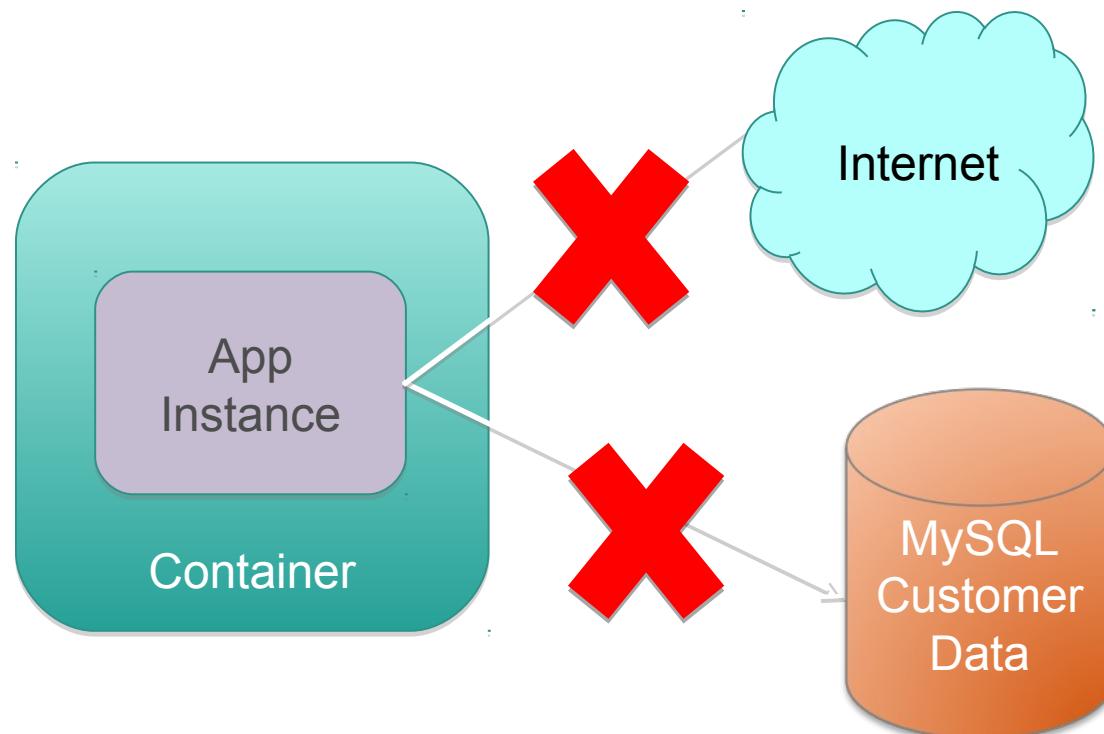
# Staging Security Groups

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



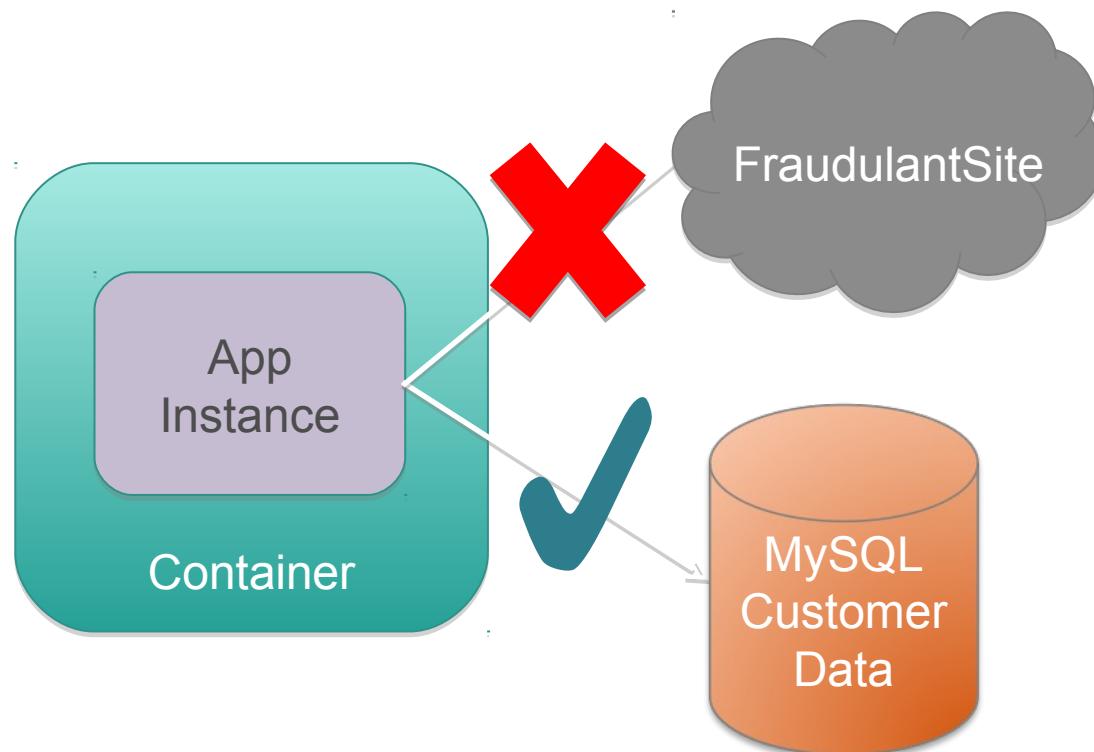
# Running Security Groups

- 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

# 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 PCF Metrics application console for a node.js application named "node". The top navigation bar includes tabs for APP, Overview, Services, Route (1), Env Variables, Logs, and Settings. The Buildpack is listed as node.js 1.5.22. On the left, there's a sidebar with "Events" showing a "Started app" log entry from pchapman@pivotal.io on 11/08/2016 at 07:38:47 AM UTC, and an "Updated app" log entry from the same user on 11/08/2016 at 07:38:03 AM UTC. The main area has sections for "Scaling" (with "Instances" set to 1, "Memory Limit" at 128 MB, and "Disk Limit" at 1 GB) and "Status" (showing 0 instances running, 0% CPU usage, 17.89 MB memory usage, 39.72 MB disk usage, and an uptime of 1 d 5 hr 32 min). A red oval highlights the "View in PCF Metrics" link in the Status section.

APP

node ■  Running

[View App](#)

Overview Services Route (1) Env Variables Logs Settings Buildpack: node.js 1.5.22

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

Cancel [Scale App](#)

Instances	Memory Limit	Disk Limit
1	128 MB	1 GB

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 is a navigation bar with tabs: APM, BROWSER, MOBILE, PLATFORM, SERVERS, and INSIGHTS. Below the navigation bar, the title "PCF Ops Manager" is displayed next to a green square icon containing a white letter "P". On the right side of the header, there is a dropdown menu labeled "pivotaled".

The main content area is titled "Available Products". It lists several items:

- Ops Manager Director (No upgrades available)
- Pivotal Service Broker (No upgrades available)
- New Relic Service Broker Alpha v-0.0.2** (No upgrades available) - This item is circled with a blue oval.
- AppDirectress v1.0.0 Experimental (No upgrades available)
- Custom Autoscaler Experimental (No upgrades available)
- MySQL for Pivotal Cloud Foundry (No upgrades available)

Below the product list is a button labeled "Import a Product" with a blue arrow pointing towards it from the bottom left.

The central part of the screen is the "Installation Dashboard". It features three tiles:

- Ops Manager Director for VMware vSphere®** (v1.0.0)
- 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 towards the "New Relic" tile. Another blue arrow points from the "Import a Product" button towards the "New Relic" tile.

On the right side of the dashboard, there is a section titled "Recent Install Logs" with a "No updates" message and a "Apply changes" button. At the bottom of the dashboard, there is a link to "Download PCF compatible products at" followed by a URL.

At the very bottom of the page, there is a footer with the text "PCF Ops Manager v1.4.0.0 © 2015 Pivotal Software, Inc. All Rights Reserved." and a "End User License Agreement".

Pivotal™

# 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 ▾

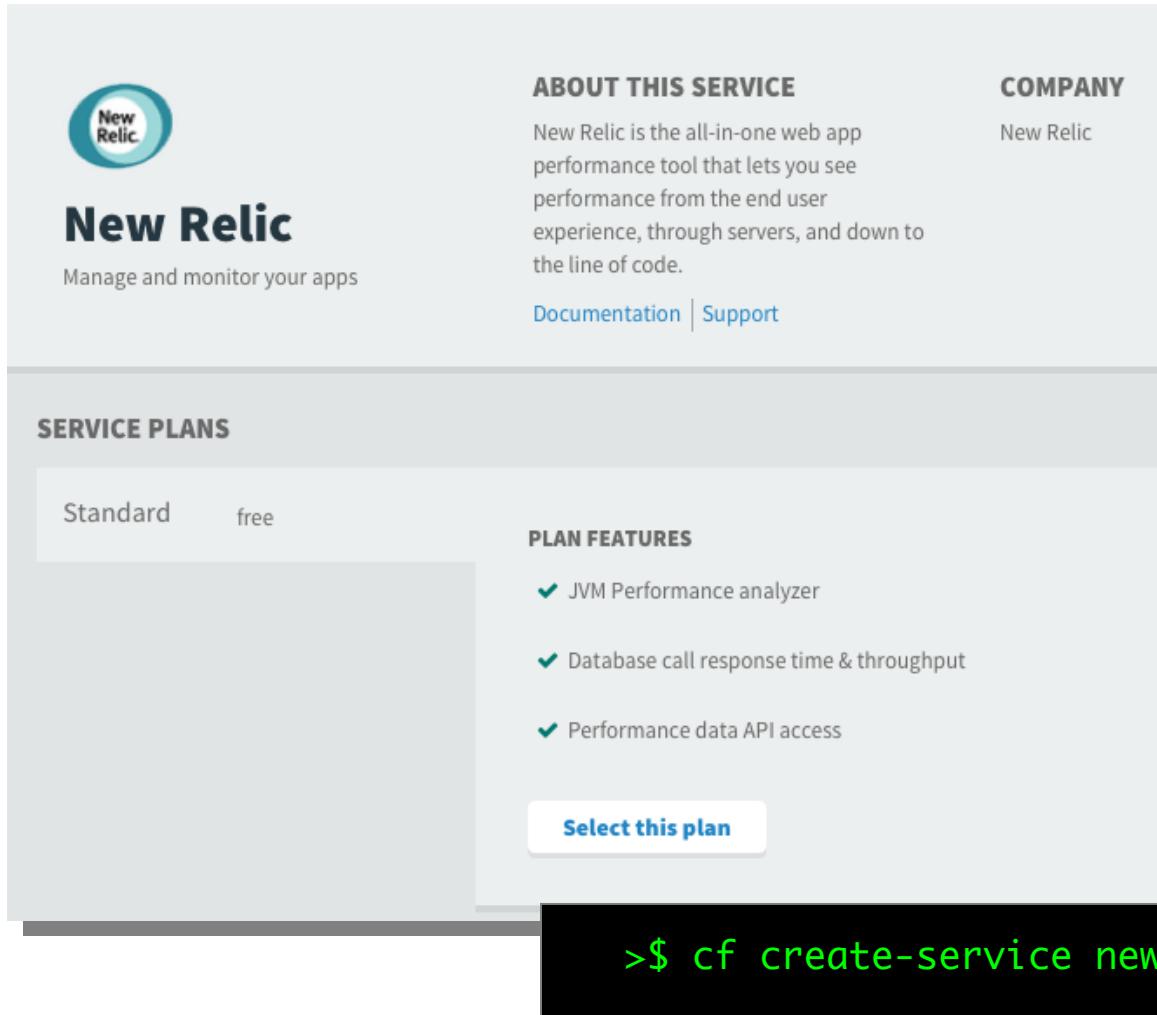
-  **3scale API Management**  
API Management Platform >
-  **App Autoscaler**  
Scales bound applications in response to load (beta) >
-  **BlazeMeter**  
Performance Testing Platform >
-  **MySQL for Pivotal Cloud Foundry**  
MySQL databases on demand >
-  **New Relic**  
Manage and monitor your apps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



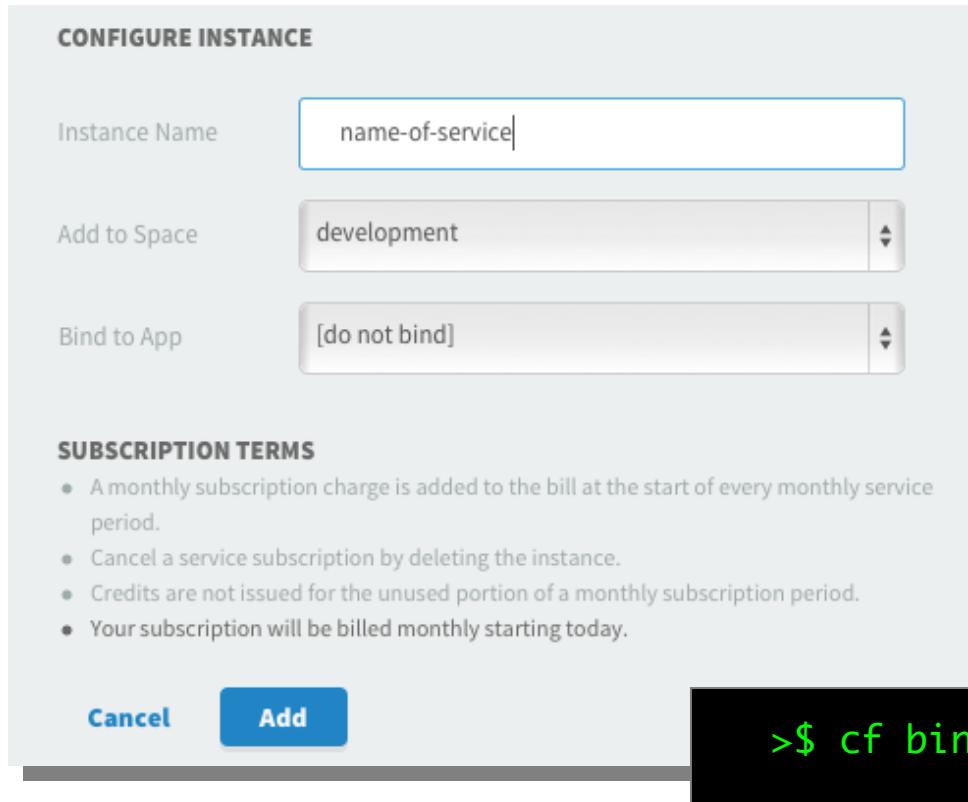
The screenshot shows the New Relic service creation interface. At the top left is the New Relic logo and the text "New Relic". Below it is the subtext "Manage and monitor your apps". To the right is the "ABOUT THIS SERVICE" section, which describes New Relic as an all-in-one web app performance tool. 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 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 a bulleted list of terms. At the bottom are 'Cancel' and 'Add' buttons, with the 'Add' button being highlighted. To the right of the dialog is a black bar containing the command: `>$ cf bind-service some-app bpm`.

CONFIGURE INSTANCE

Instance Name: name-of-service

Add to Space: development

Bind to App: [do not bind]

SUBSCRIPTION TERMS

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

Cancel Add

>\$ cf bind-service some-app bpm

# Access via *Manage* Link in App Manager



The screenshot shows the New Relic App Manager interface. At the top, there is a header with the application name "my-apm", the plan "New Relic Standard", and a count of "1". Below the header, there are links for "Manage", "Documentation", "Support", and "Delete". A red circle highlights the "Manage" link. An arrow points from this highlighted link down to the main content area.

The main content area features the New Relic logo and the "INSIGHTS™" banner. The navigation bar includes "Applications", "cf-session-demo" (selected), "Monitoring", "Events", "Reports", and "Settings". The left sidebar lists "Applications", "Browser", "Transactions", "Mobile", "Servers", "Dashboards", "Plugins", and "Tools". The "Dashboards" and "Tools" items have a dropdown arrow icon.

The central dashboard displays a chart titled "Web transactions response time". The Y-axis ranges from 0 ms to 250 ms, and the X-axis shows times from 11:00 to 11:20. A single transaction spike is highlighted in light blue, reaching a peak of 13 ms at approximately 11:18. The label "App server" is shown near the peak of the spike.

# 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

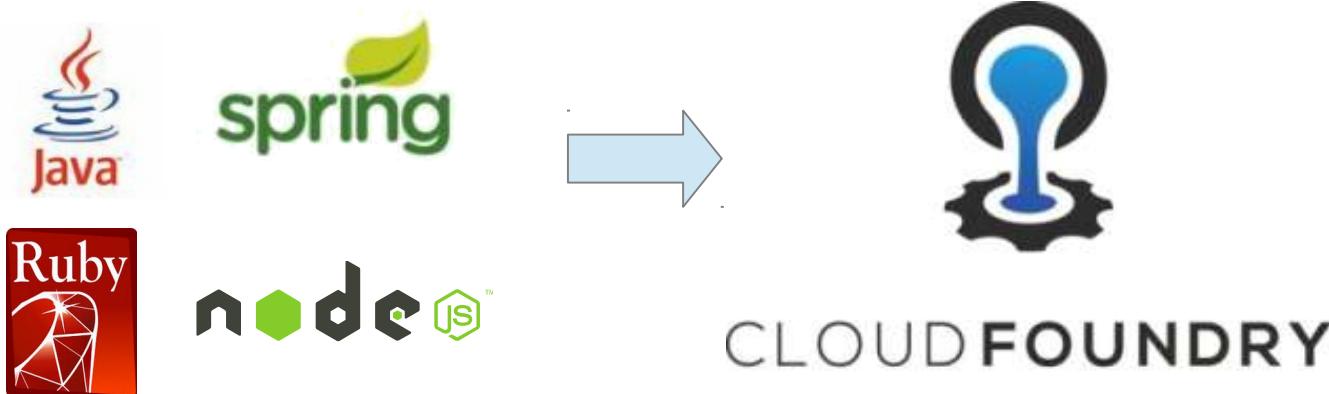
- After completing this lesson, you should be able to:
  - Explain what a buildpack is
  - Deploy using a buildpack
  - Describe the basic Buildpack API
  - Describe the behavior of the Java Buildpack
  - Configure / extend a Buildpack
  - Use Docker image in PCF

# Roadmap

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

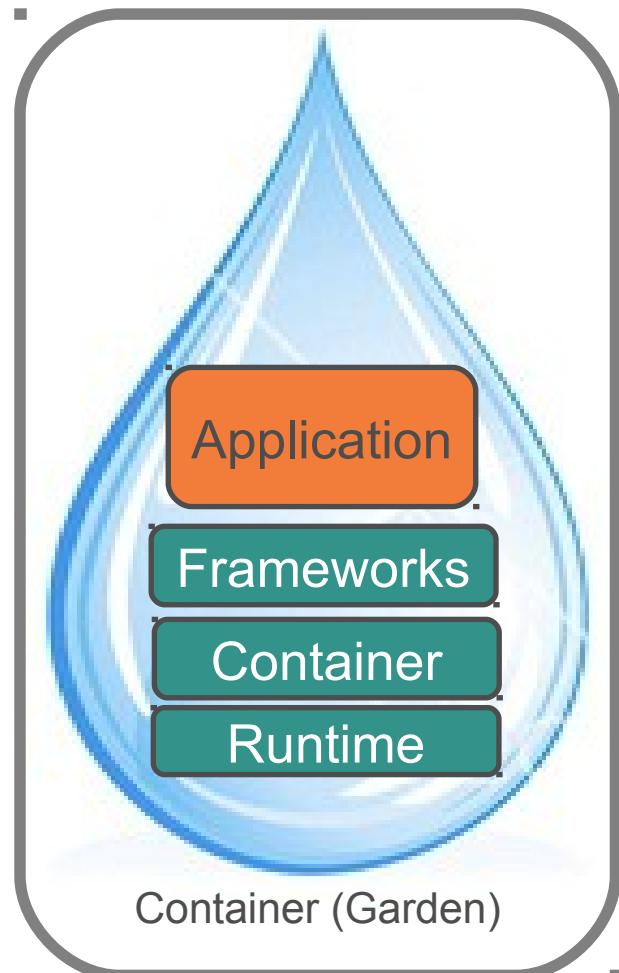


# Buildpack Does the Same Thing

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)



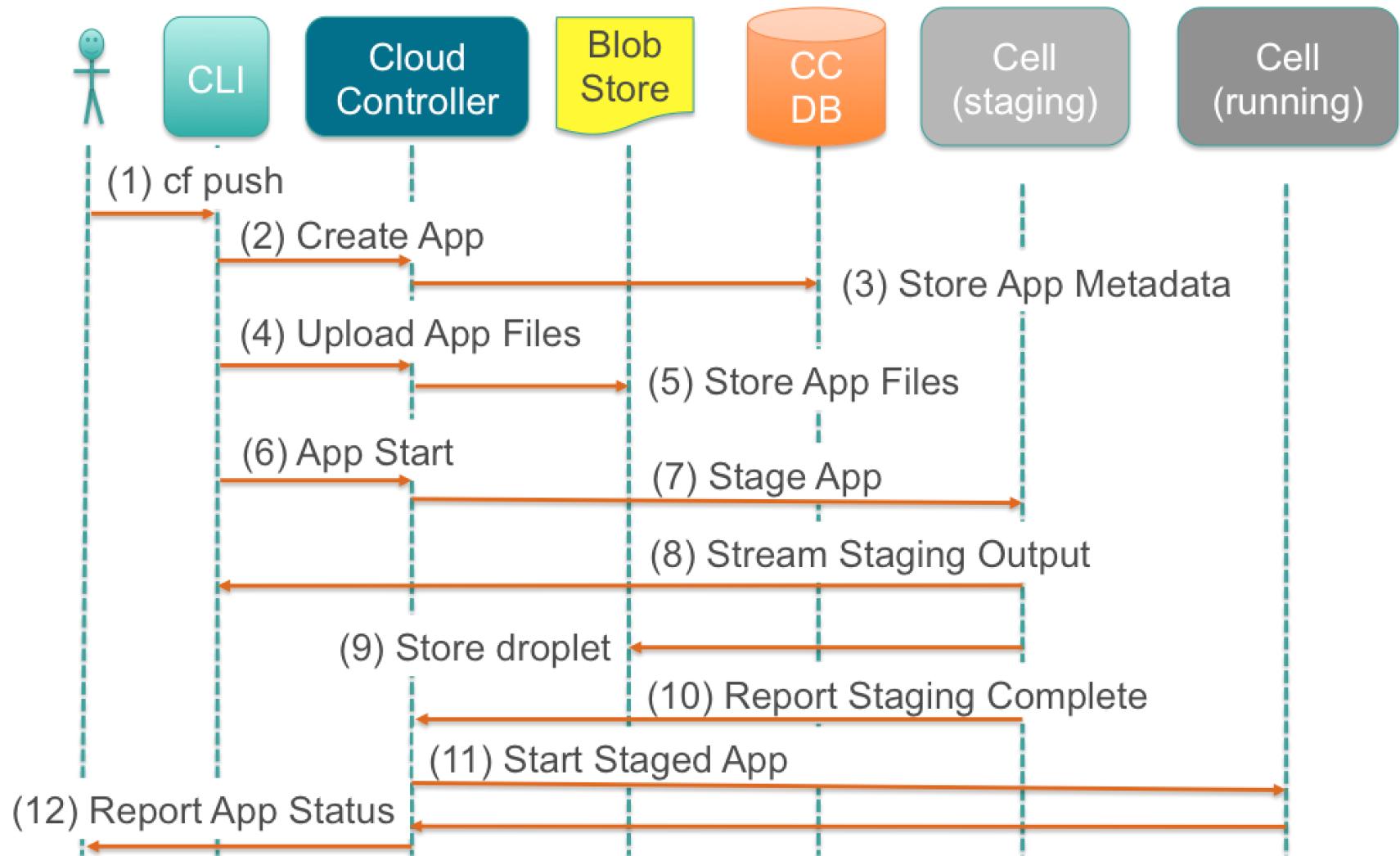
# Buildpacks are Not...

- 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

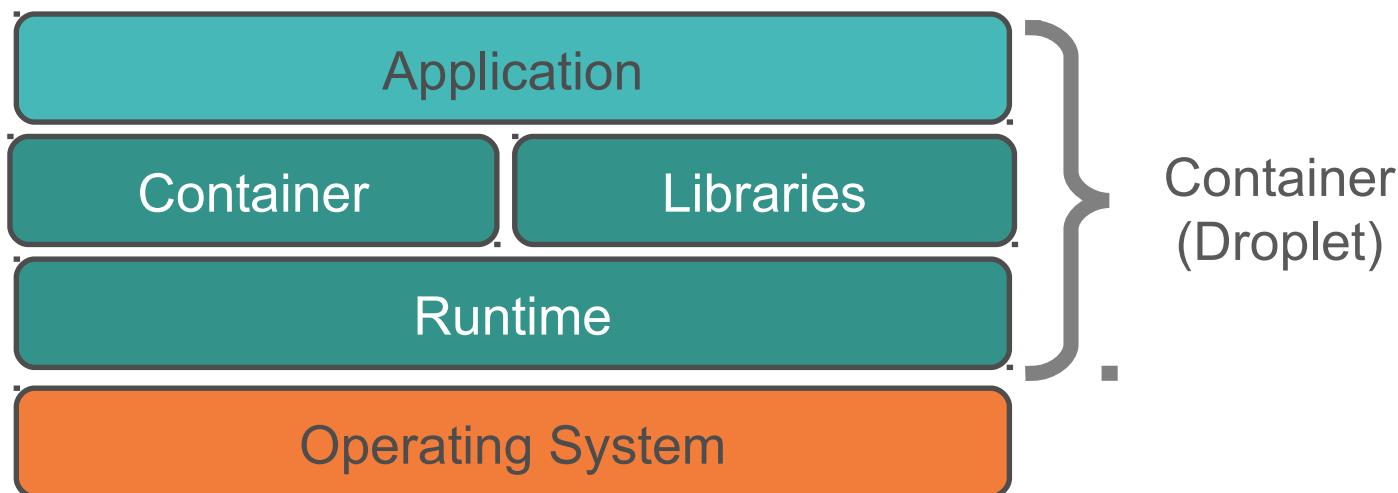
- What are Buildpacks?
- **Deploying to Cloud Foundry**
- Using Buildpacks
- Buildpack API
- Java Buildpack
- Configure / Extend Java Buildpack
- Customization without Forking
- Using Docker in PCF

# Deploying to CF



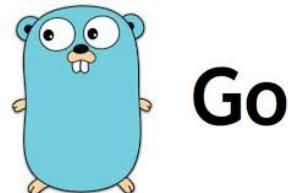
# Staging and Buildpacks

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

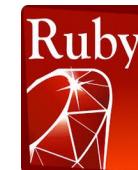


# Available Buildpacks

- 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>



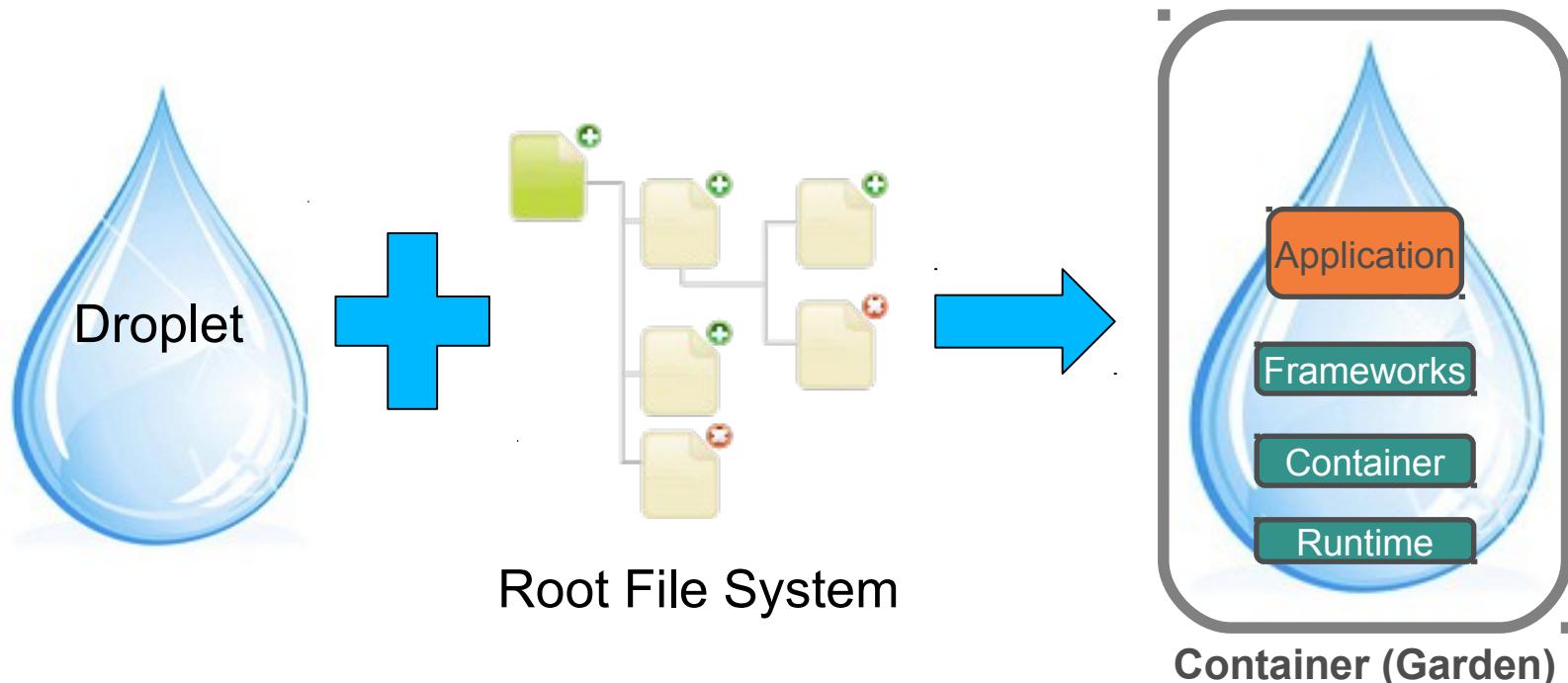
# Compatibility

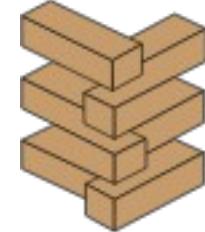
- Buildpacks can be compatible with multiple PaaS offerings
- CF buildpacks follow the Heroku buildpack design
  - CF and Heroku buildpacks are compatible (if you care to make them compatible)
  - Other PaaS offerings adopting the buildpack design



# Stacks

- Provide the underlying root file system for the container
- Two stacks available
  - Ubuntu Linux or Windows 10 Server





# Why Stacks?

- Operational Control
  - Standardization of underlying platform
  - Restrict users to approved/tested stacks
  - Easy to upgrade and patch
    - *For example:* to overcome malware or a virus
- Operators can upgrade buildpacks and stacks
  - Without developers being aware, unless they want to be

# Roadmap

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

# 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) ← From Blobstore  
-----> Downloading Open Jdk JRE 1.8.0_20 from  
http://download.run.pivotal.io/openjdk/lucid/x86_64/open  
jdk-1.8.0_20.tar.gz (1.0s) ← From Amazon S3  
          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) ← To Blobstore  
          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
- **Configure / Extend Java Buildpack**
- Customization without Forking
- Using Docker in PCF

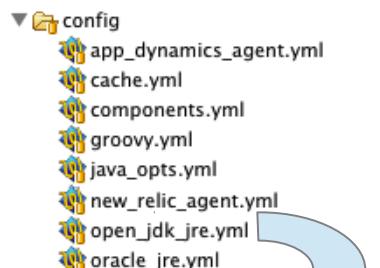
# Customization

- 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

# Roadmap

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

# 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

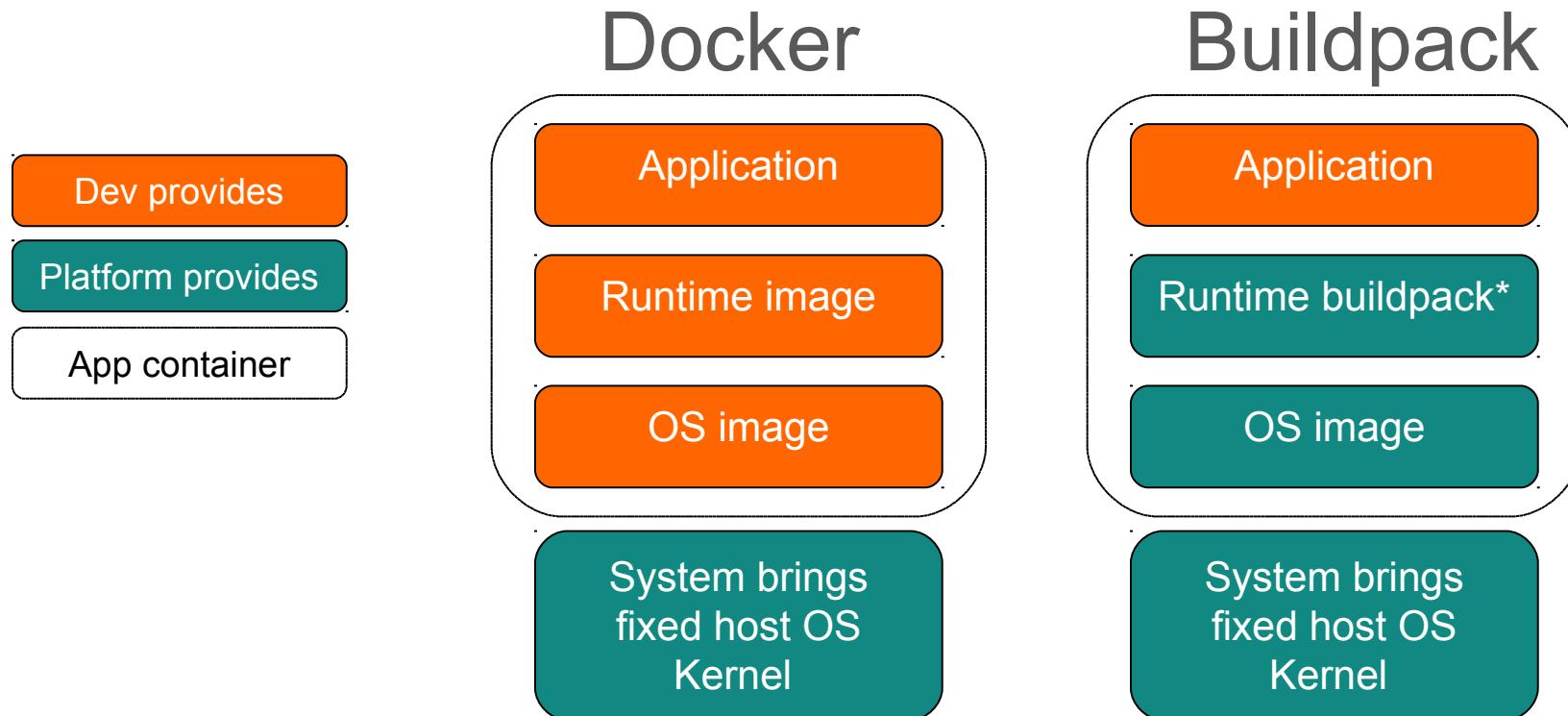
- Use environment variable to override a buildpack configuration file
  - Naming convention used:
    - `my_file.yml` → `JBP_CONFIG_MY_FILE`
  - Variable must be set to valid *inline* YAML syntax
- To change default version of Java to 7
  - Override `open_jdk_jre.yml`

```
>$ 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
- Configure / Extend Java Buildpack
- Customization without Forking
- **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

# Introduction to Microservices

Monolith vs Microservices

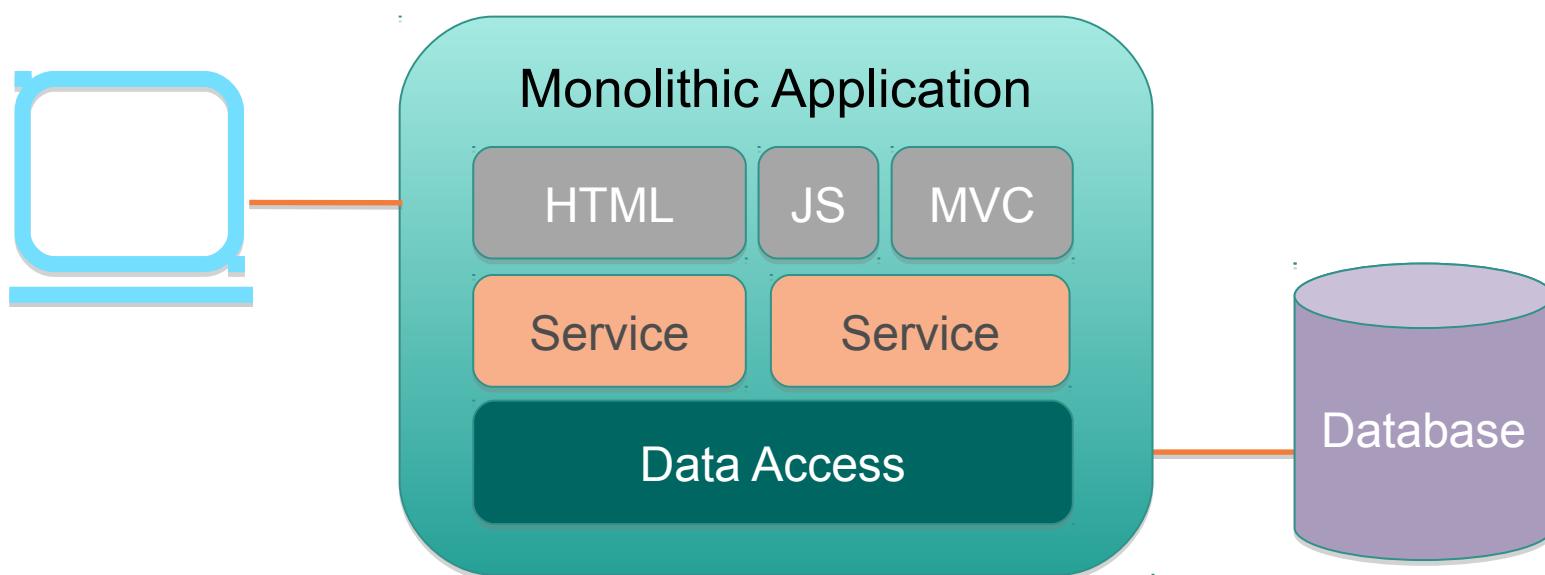
Enabling Microservices with PCF

# Agenda

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

# Monolith

- A three tiered monolith.



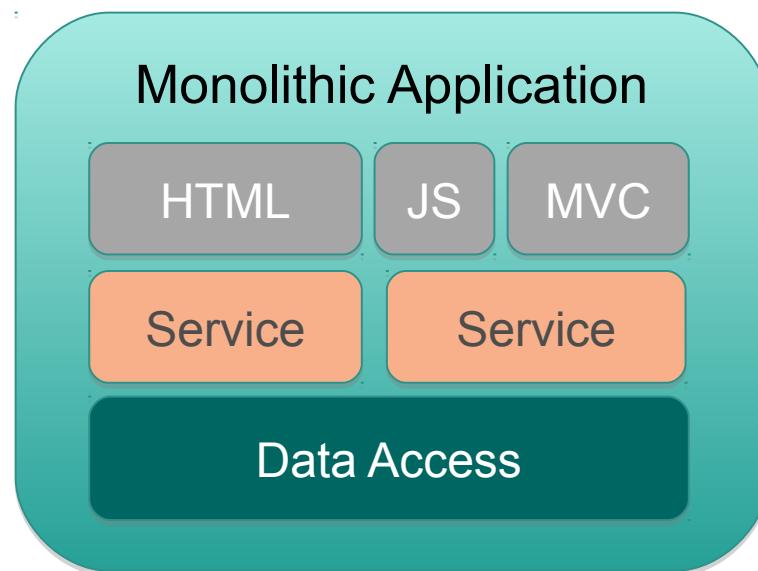
# Monolith challenges

# 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

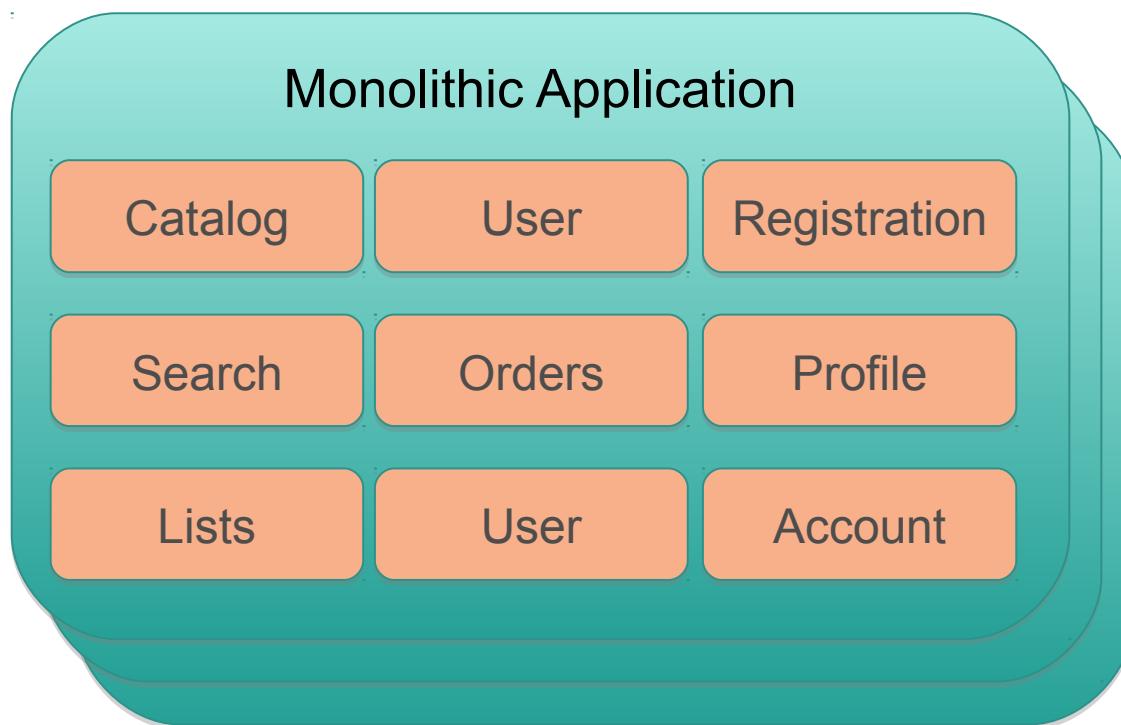
# Monolith Change Cycle

- 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

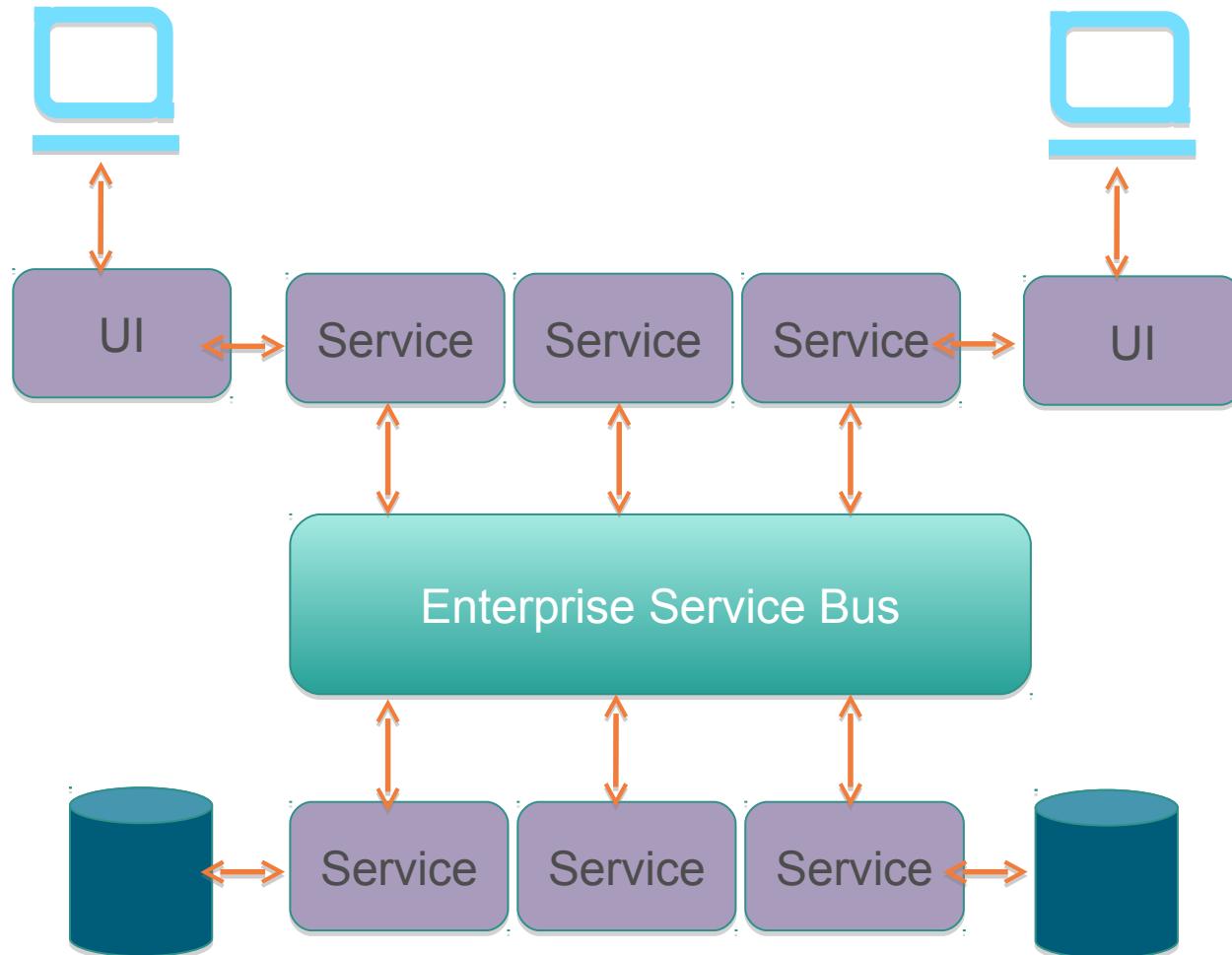
- Compose an application using a set of small services
  - each running as independent processes
  - intercommunicating via open protocols
  - separately written, deployed, scaled & maintained
  - encapsulating business capability

# 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



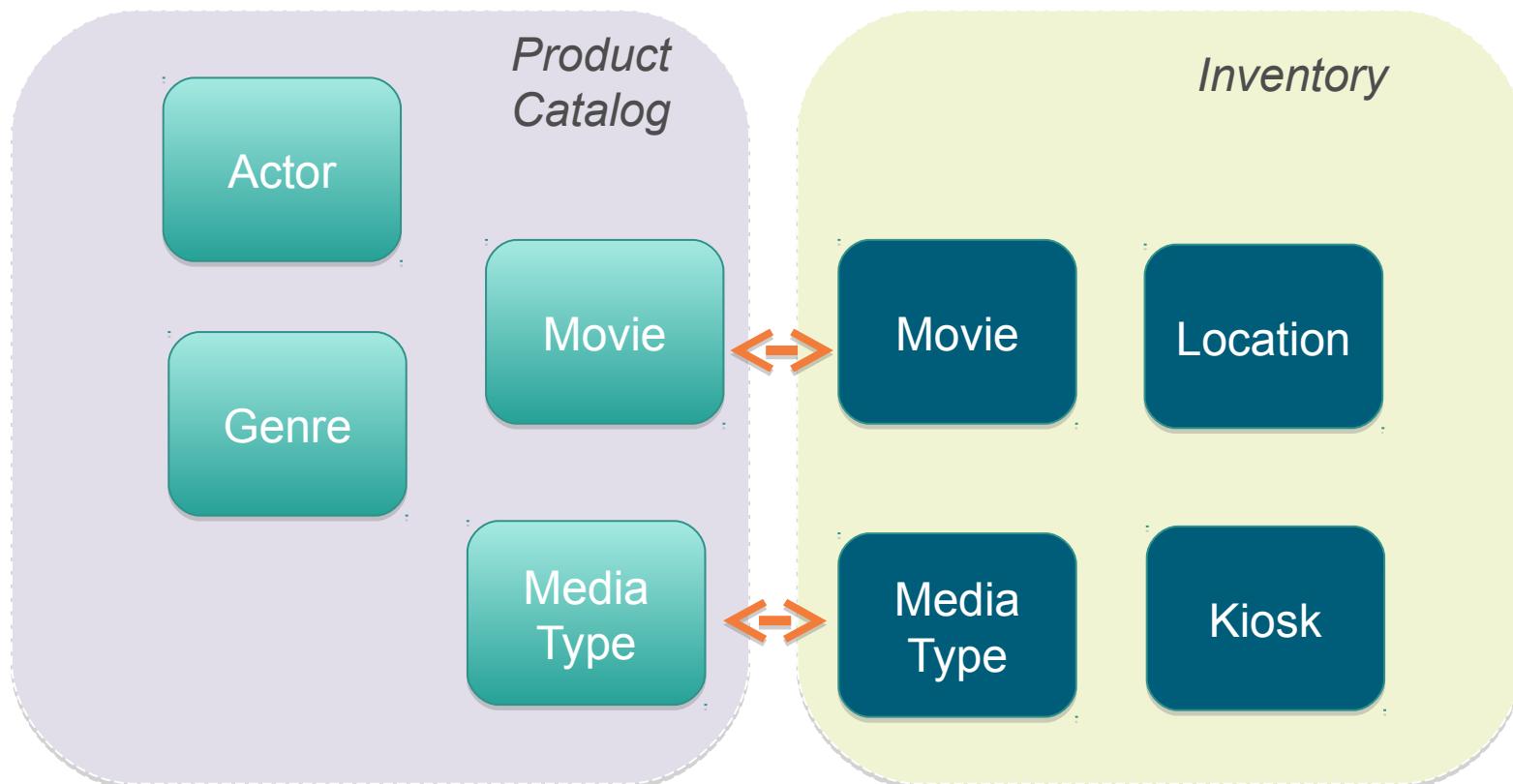
# Unix Pipes and Filters

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

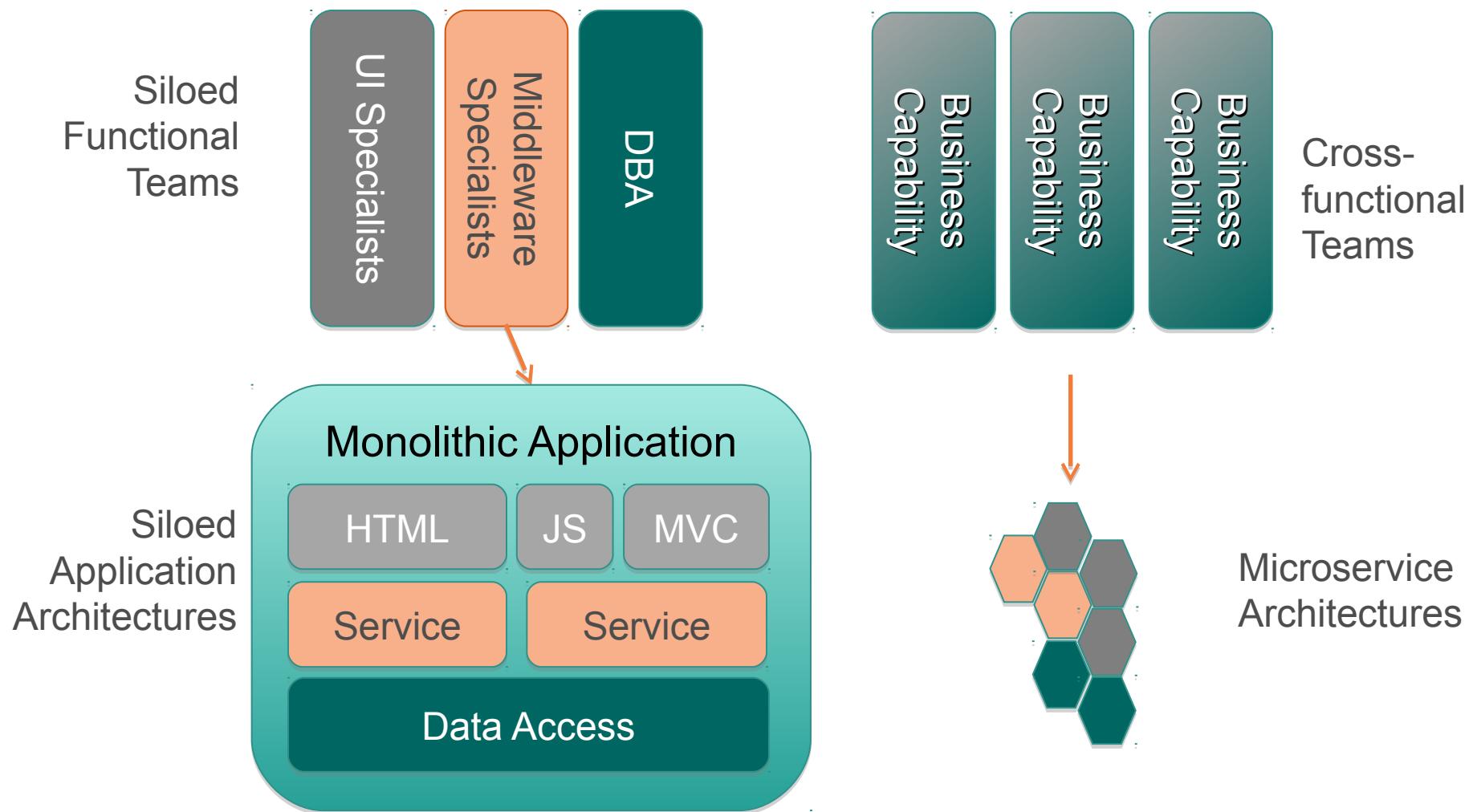
```
cut -d" " -f1 < access.log | sort | uniq -c | sort -rn | less
```

# Bounded Context

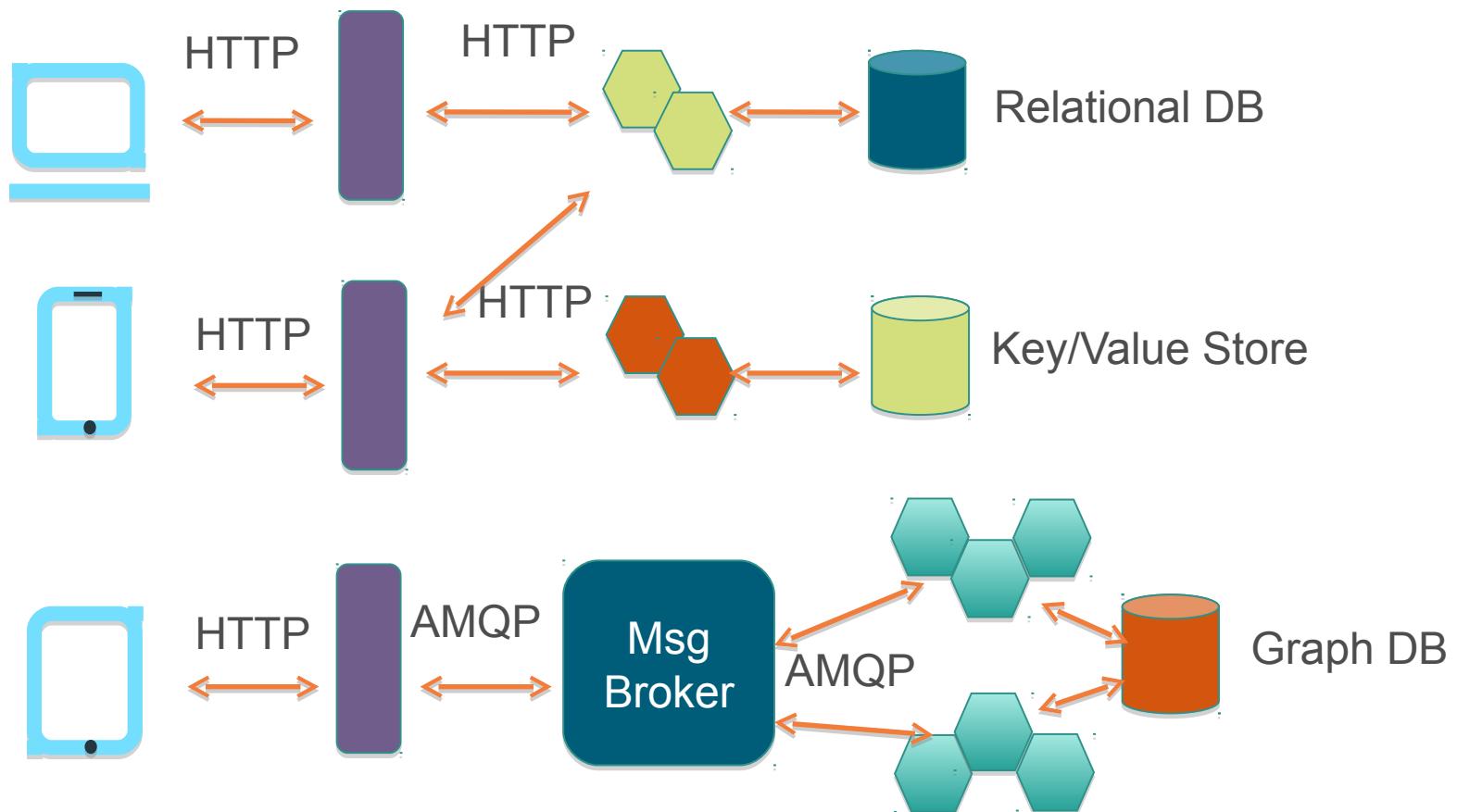
- Microservices use a *bounded context*.



# Organize Around Business Capabilities

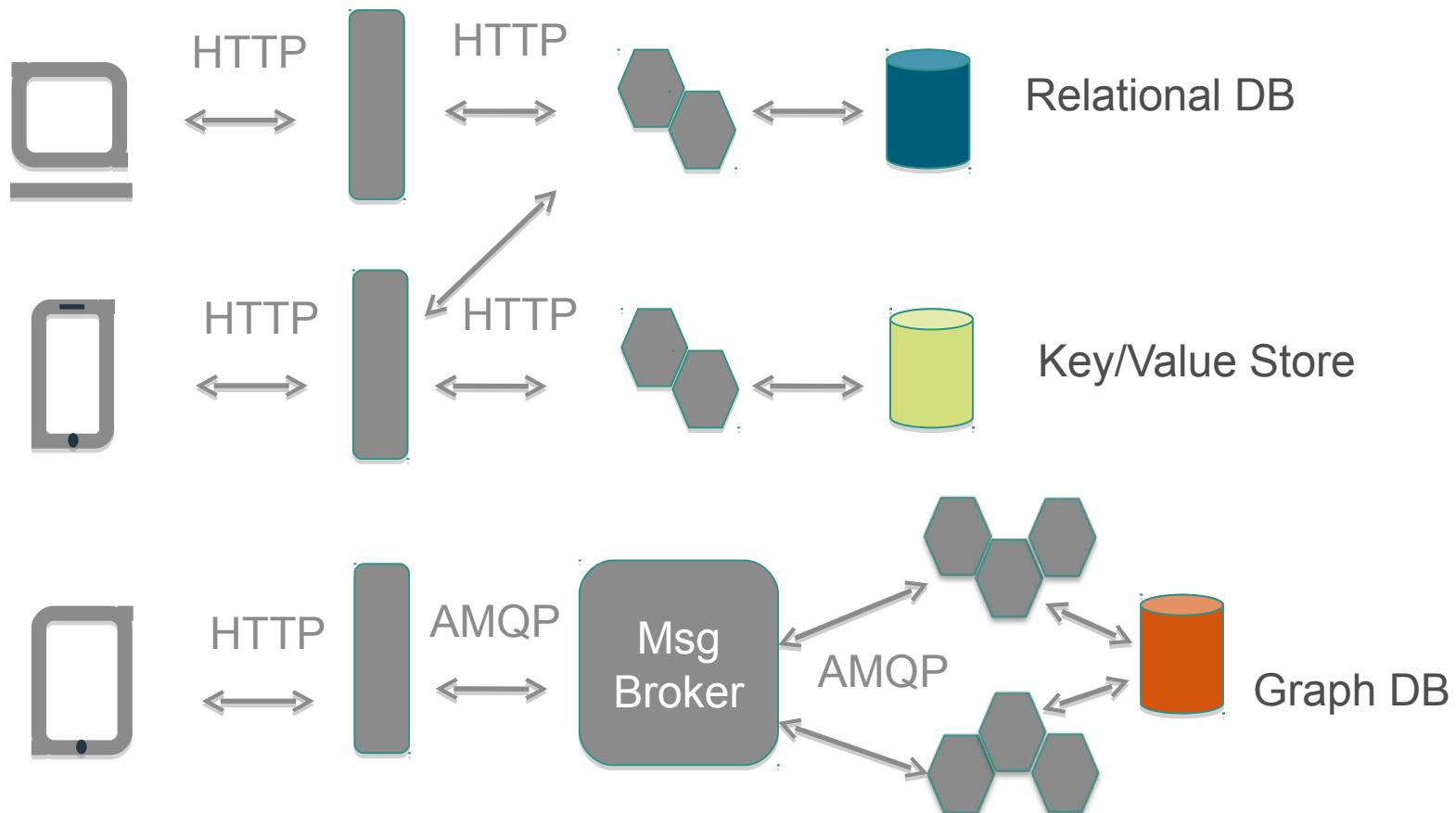


# Microservice Architecture (Simplified)



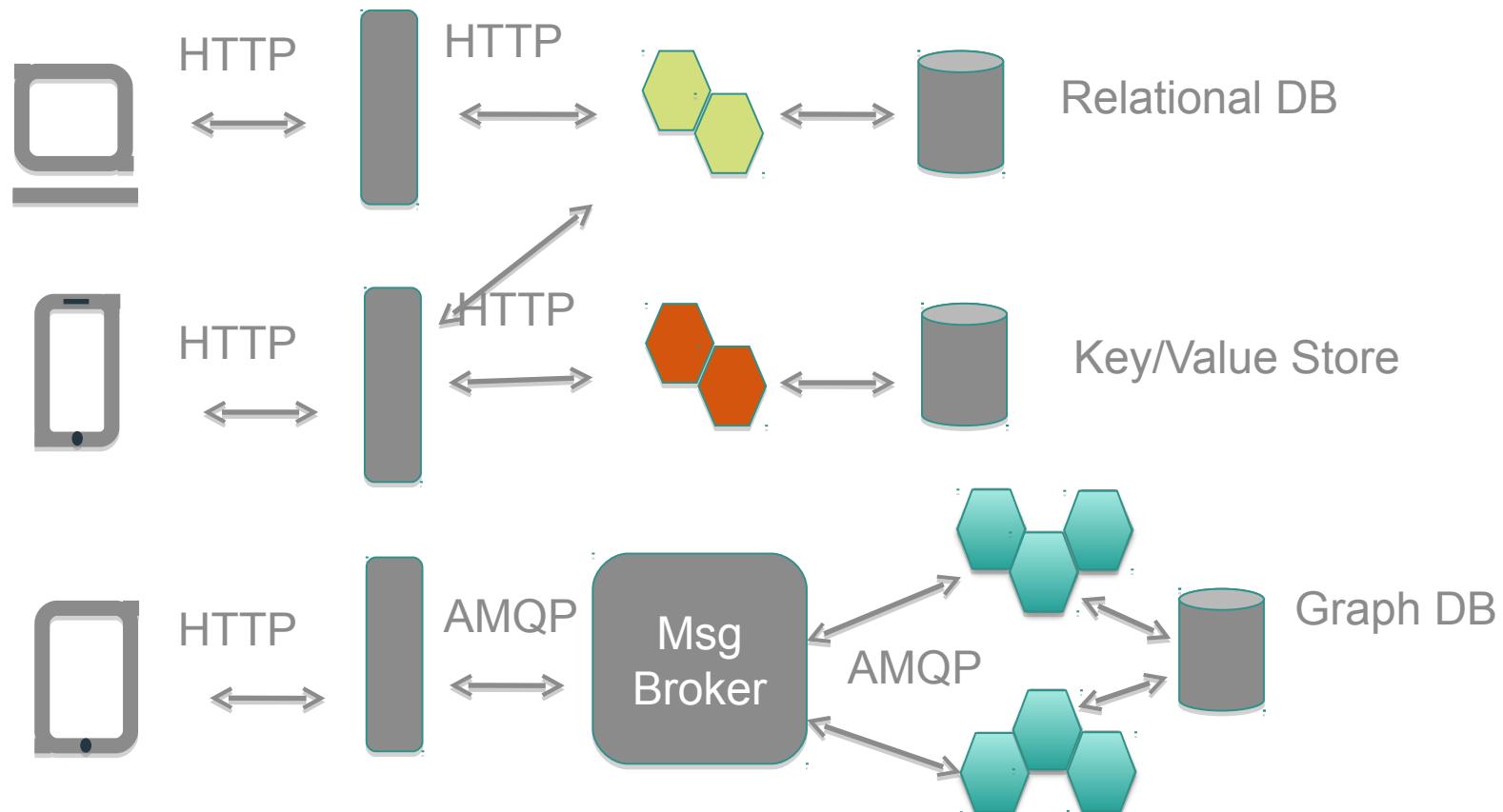
# Polyglot Persistence

- Freedom to pick the persistence solution.



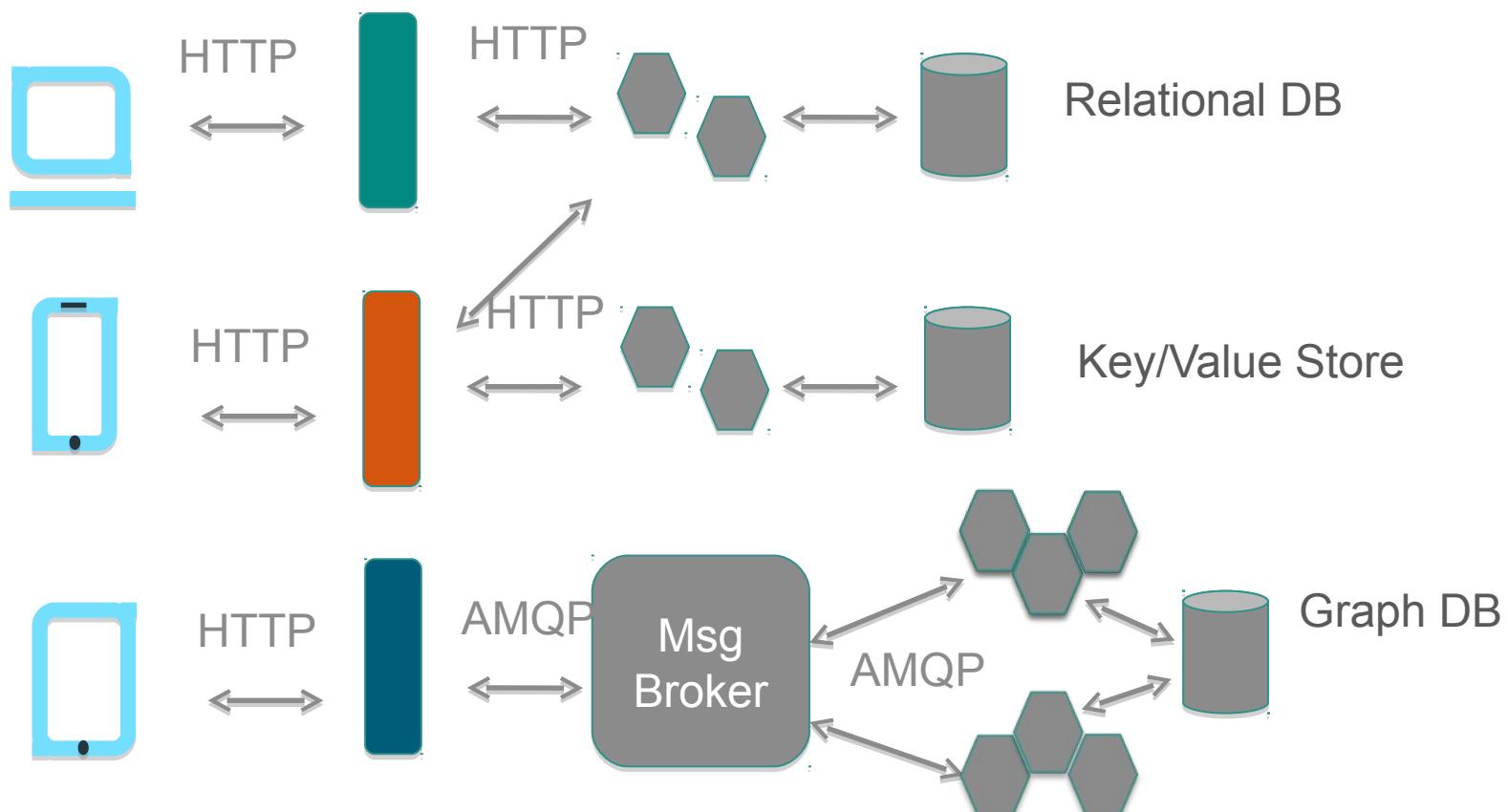
# Polyglot Apps

- Choice of language when developing apps.



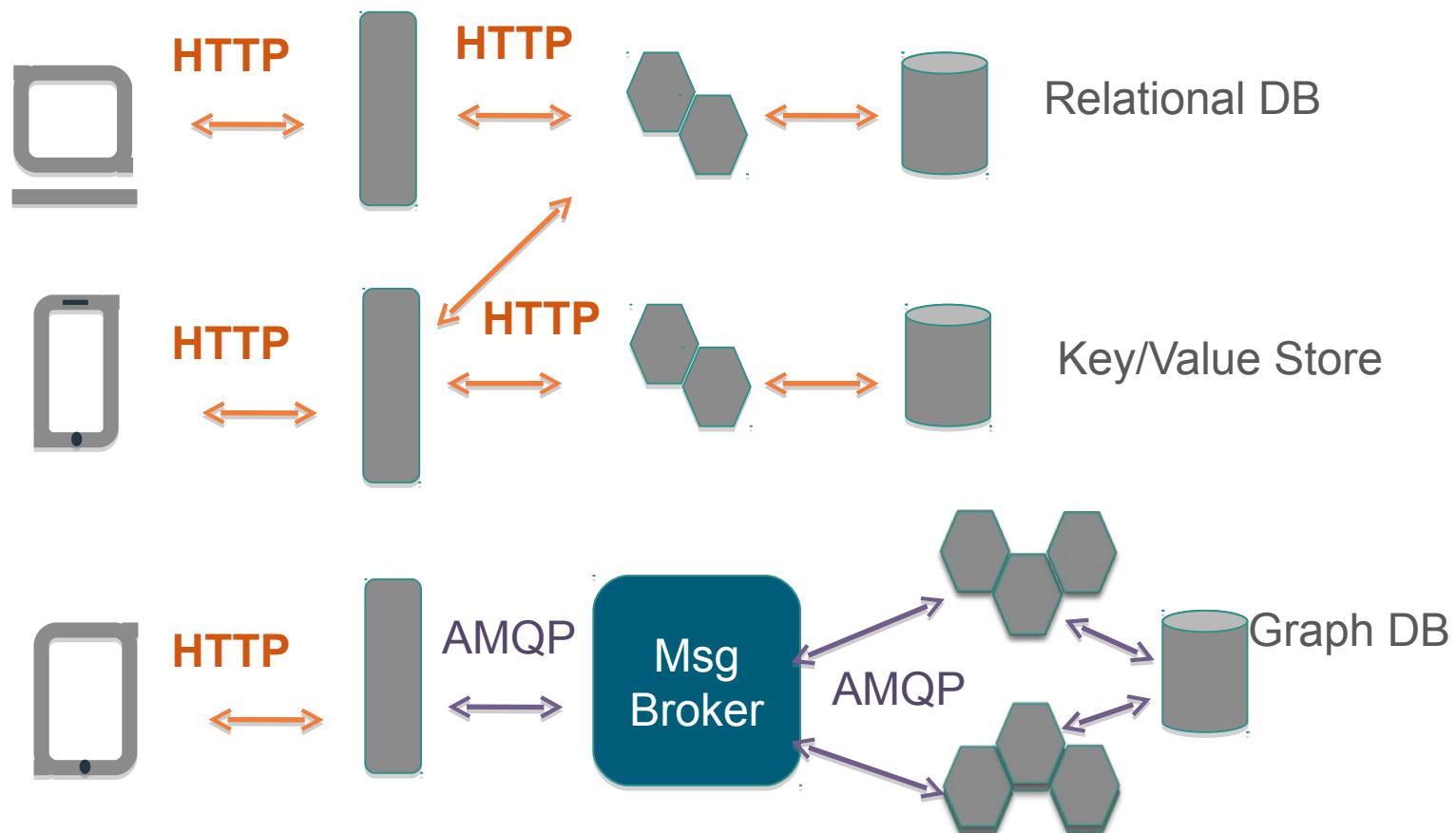
# API Gateway

- Device specific gateways.



# Cloud Protocols

- Use cloud friendly protocols.



# Microservice Benefits

# Microservice Advantages – 1

## Change Cycle



- Change cycles are decoupled
- Enables more frequent deploys

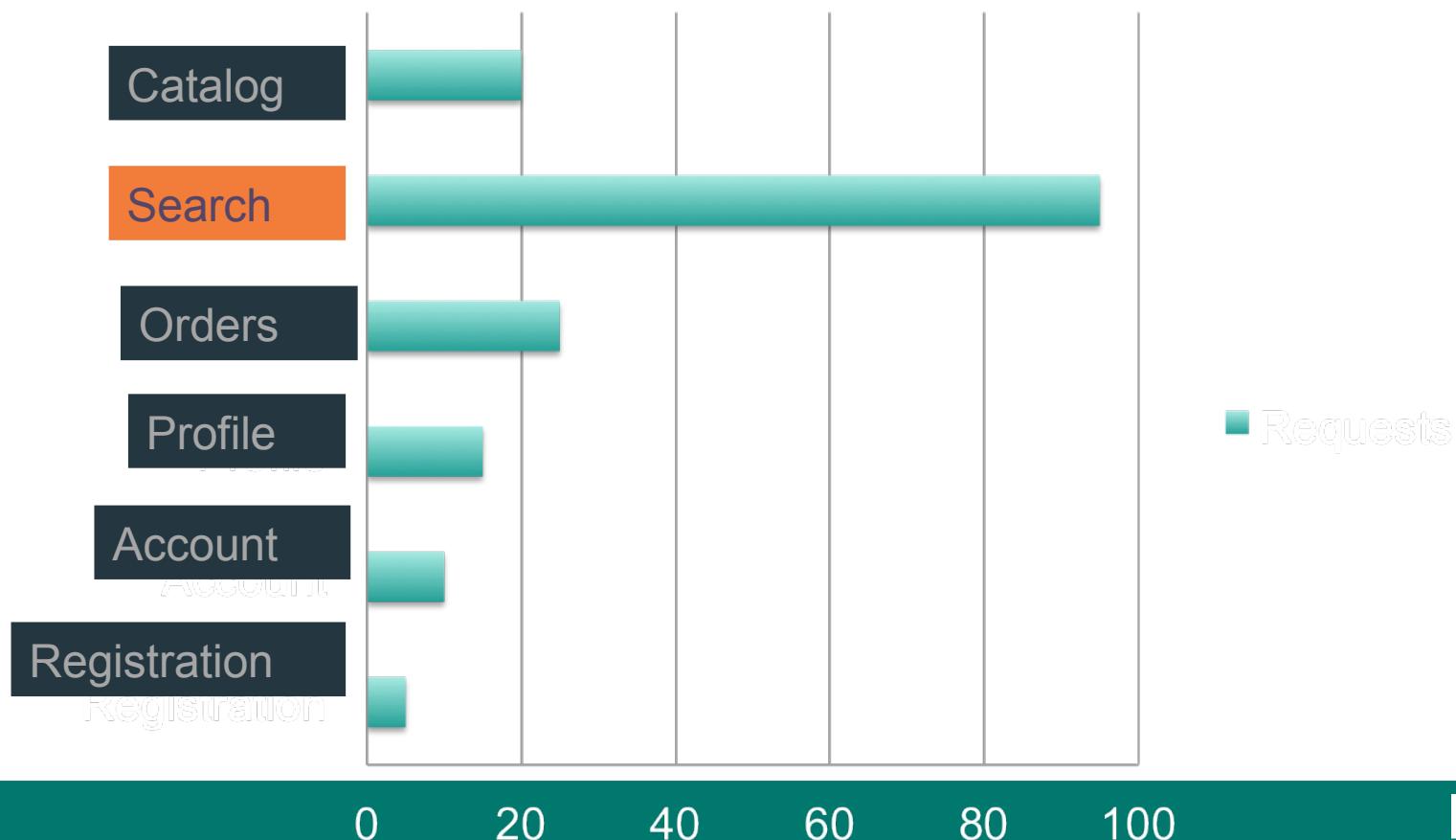
# Microservice Advantages – 2

## 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

# Advanced Topics

## Useful Features

Tasks, Volumes, More on routing

# Agenda

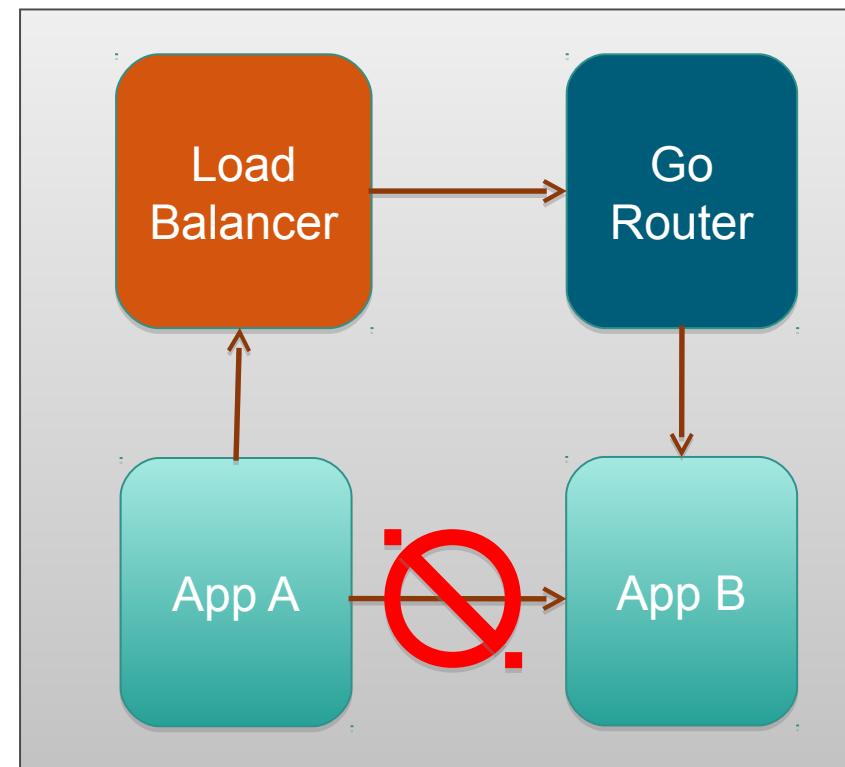
- Tasks
- File System as a Service
- Container to Container Routing
- TCP Routing

# Agenda

- **Container to Container Routing**
- TCP Routing

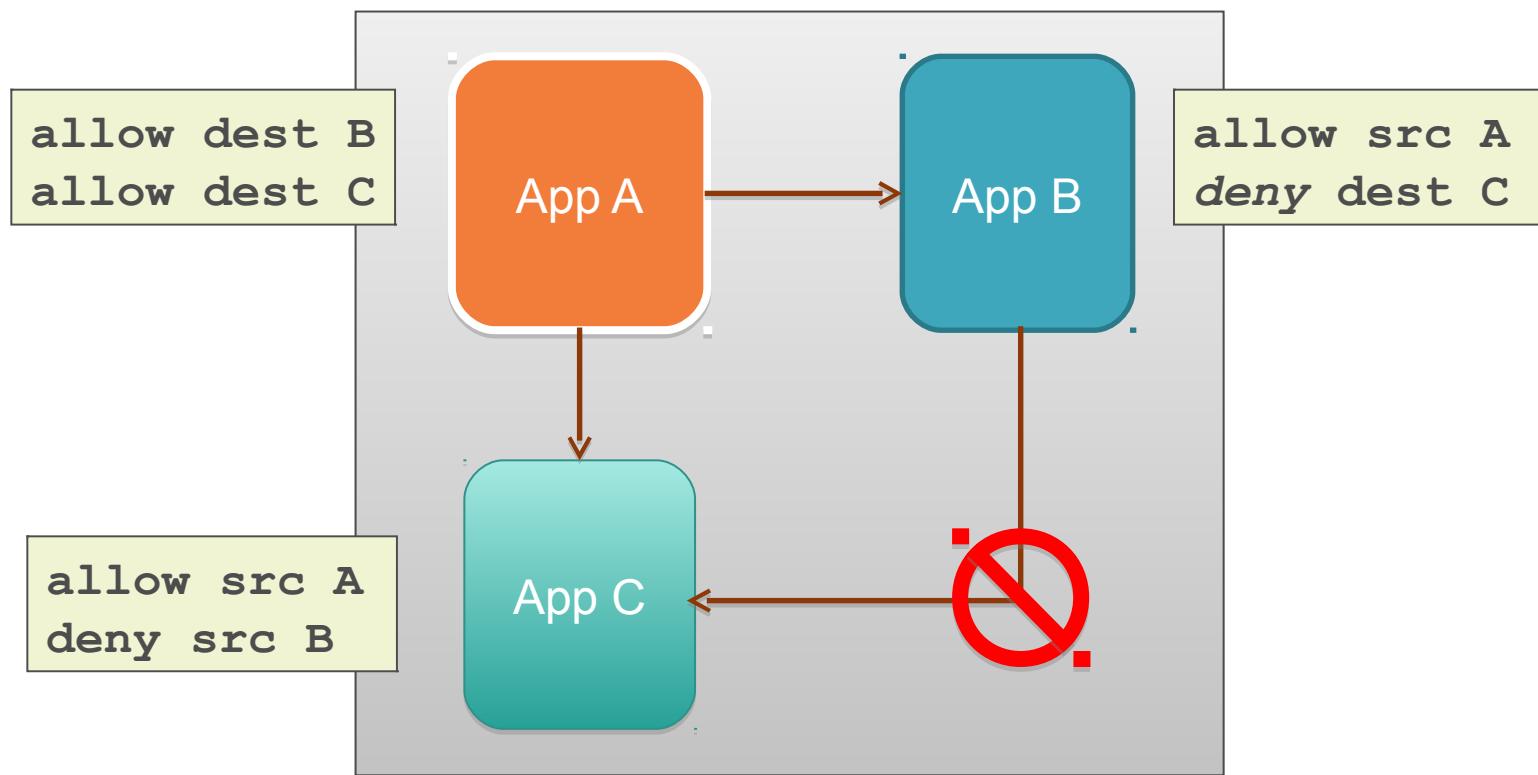
# Without C-C Routing

- 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

- 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
```



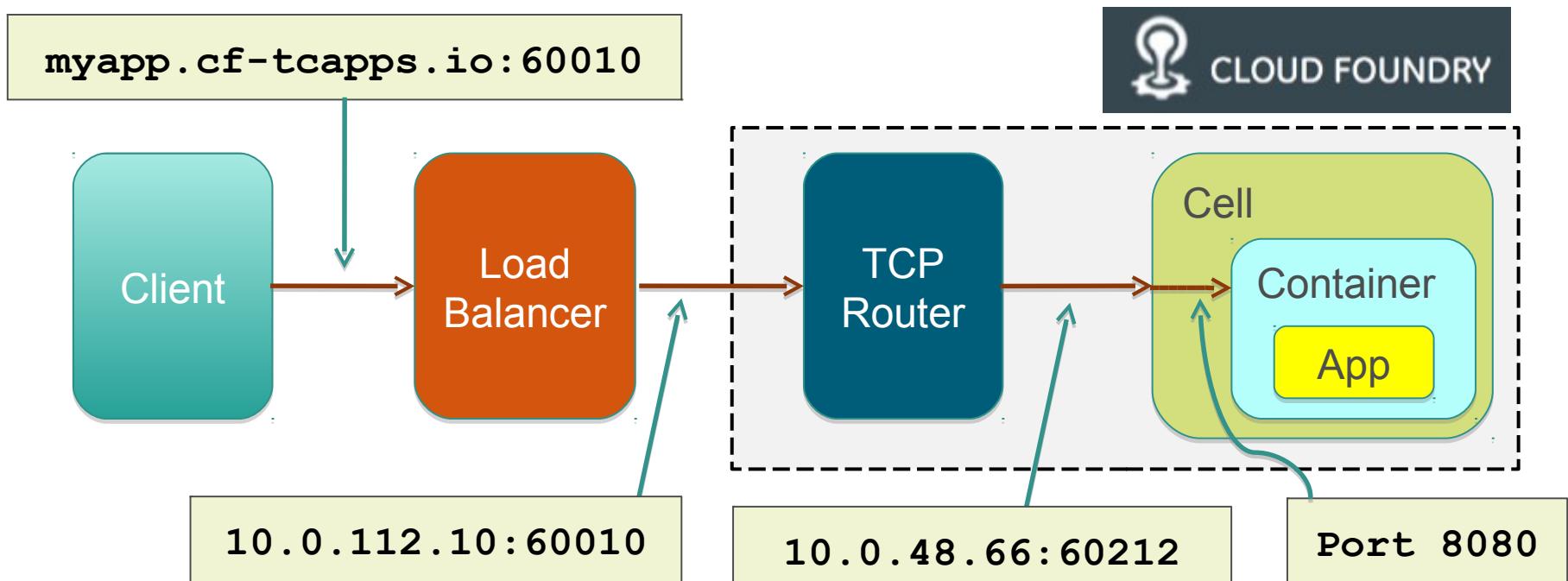
*random port*

- 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
  - Explain Container to Container Routing
  - Push an application using TCP Routing

# Lab

## Running a Task in Cloud Foundry

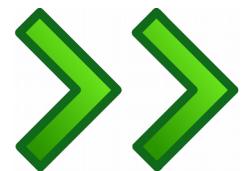
# Finishing Up

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



- Topics Covered
  - Cloud Foundry Basics
  - Cloud-Native Architectural Principles
  - Troubleshooting Applications on Cloud Foundry
  - Cloud-Native Application Security
  - Working with Services in Cloud Foundry
  - Application Management on Cloud Foundry
  - Container Management within Cloud Foundry

<https://www.cloudfoundry.org/certification>



# Certification

- Performance-based exam
  - You actually use CF in the exam
  - Virtual – can take the exam anywhere in the world
  - Cost is \$300 (April 2018)
  - Log into the [CFCD portal](#) – you will need a Linux Foundation ID
- Before taking the exam
  - Review all the slides
  - Redo the labs

# Other courses



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

# Cloud Foundry Operations



CLOUD FOUNDRY

- 4 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 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