Pivotal

Intro To Cloud Foundry - Desert Code Camp 2016

October 2016

Caleb Washburn

About me….

- Advisory Solutions Architect @ Pivotal
- 18+ years in IT, mostly working within an enterprise
- Working with Spring based applications for 10+ year. Yes…when XML was the cool new thing
- Contributor/Maintainer of a few open source projects (mainly written in GO)
- Experience with various cloud technologies and IaaS providers
- My Github - https://github.com/calebwashburn/
- Disclaimer: Work at Pivotal but my opinions are my own

What we plan to go over…

- What is Cloud Foundry?
- What Cloud Foundry isn't
- 12-Factor
- Buildpacks
- Orgs/Spaces
- Roles
- cf push/manifests
- Logging
- Scaling
- Blue/Green Deployment
- Services
- Service Brokers

Cloud Foundry (https://docs.cloudfoundry.org/)

- Open Source Platform as a Service (PaaS)
- Several Distributions based on OSS (Pivotal Cloud Foundry, IBM Blue Mix, others). Of course use Pivotal Cloud Foundry :)
- Runs on several IaaS platforms (AWS, vSphere, Azure, GCP, OpenStack, Photon) even on your local machine (https://network.pivotal.io/products/pcfdev)
- Helps abstract IaaS specifics away from your application. Portability!
- Supports multiple application languages (Java, Node, Ruby, Python, PHP, Go, .NET core, Docker)
- Goal is to reduce operational overhead of hosting applications for both operators and developers
- Put developers in control to allow rapid delivery and spend less time setting up servers to host you apps
- Optimized to run applications that adhere to the 12 Factors
- Great for running microservices
- Applications are run in containers. Each app gets own dedicated container

What Cloud Foundry Isn’t

- A VM provider - No “Just give me a VM” - Use IaaS for that
- Overkill for hosting 1 application (lost of VM’s make up cloud foundry)
- Not a good fit if application has file persistence needs (coming soon disk as a service)
- Silver Bullet that makes crappy apps work

12 Factor App Characteristics (https://12factor.net/)

I. Codebase
One codebase tracked in revision control, many deploys
II. Dependencies
Explicitly declare and isolate dependencies
III. Config
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 the app as one or more 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 development, staging, and production as similar as possible
XI. Logs
Treat logs as event streams
XII. Admin processes
Run admin/management tasks as one-off processes

Build Packs (https://docs.cloudfoundry.org/buildpacks/)

Name	Supported Languages, Frameworks, and Technologies	GitHub Repo
Java	Grails, Play, Spring, or any other JVM-based language or framework	Java source
Ruby	Ruby, JRuby, Rack, Rails, or Sinatra	Ruby source
Node.js	Node or JavaScript	Node.js source
Binary	*n/a*	Binary source
Go	Go	Go source
PHP	Cake, Symfony, Zend, Nginx, or HTTPD	PHP source
Python	Django or Flask	Python source
Staticfile	HTML, CSS, JavaScript, or Nginx	Staticfile source

Pivotal

Orgs/Spaces and Roles

Organizations are logical segmentation within Cloud Foundry.
- An Organization can contain 1 - many spaces.
- Organizations has 3 roles (OrgManager, OrgAuditor, OrgBillingManager)

Spaces are used to allow hosting multiple instances of same apps and services in the same Cloud Foundry. Think Dev -> QA -> Prod as spaces
- Each space gets own app instances of both apps and services.
- Space has 3 roles (SpaceManager, SpaceDeveloper, SpaceAuditor)

CF Push / Manifests

No matter what language you are using to deploy your app use the same command.

- cf push -f manifest.yml (or if manifest in current working dir you can ignore -f)

Manifests articulate characteristics of your app.
- Memory needs
- Buildpack to use
- Domain to bind to
- Host name of app
- Number of instances
- Environment variables
- path to route on (hello.test.com/v1)

Logging

Along with the 12 factors.
- No longer log to files.
- Log to system out
- Cloud Foundry will consolidate all your application logs into the firehose for you.
- Operator of platform can also pipe your logs to syslog for ability to look at them over time in Splunk, Kibana, etc
- Use cf logs <appname> to see your logs

Scaling

Since your applications are all deployed as containers. Cloud Foundry can easily scale up/down your application with zero downtime

- cf scale <appname> -i <intance count>

Pivotal Provides Auto Scaling Broker that can be used to configure policies to do this for you based on cpu/memory usage. Autonomic Scaling to ensure your application stays up and can meet the demands.

Blue/Green Deployment

Blue/Green Deployment is the ability to update your application with a new version with Zero Downtime.

Auto Pilot cf cli plugin enables this to be done for you.

cf zero-downtime-push <appname> -f <manifest>

Services/Service Brokers

To see what services have been provided by operators of your platform use the marketplace

cf marketplace

This will list out what services are available in your environment.

cf create-service <options>

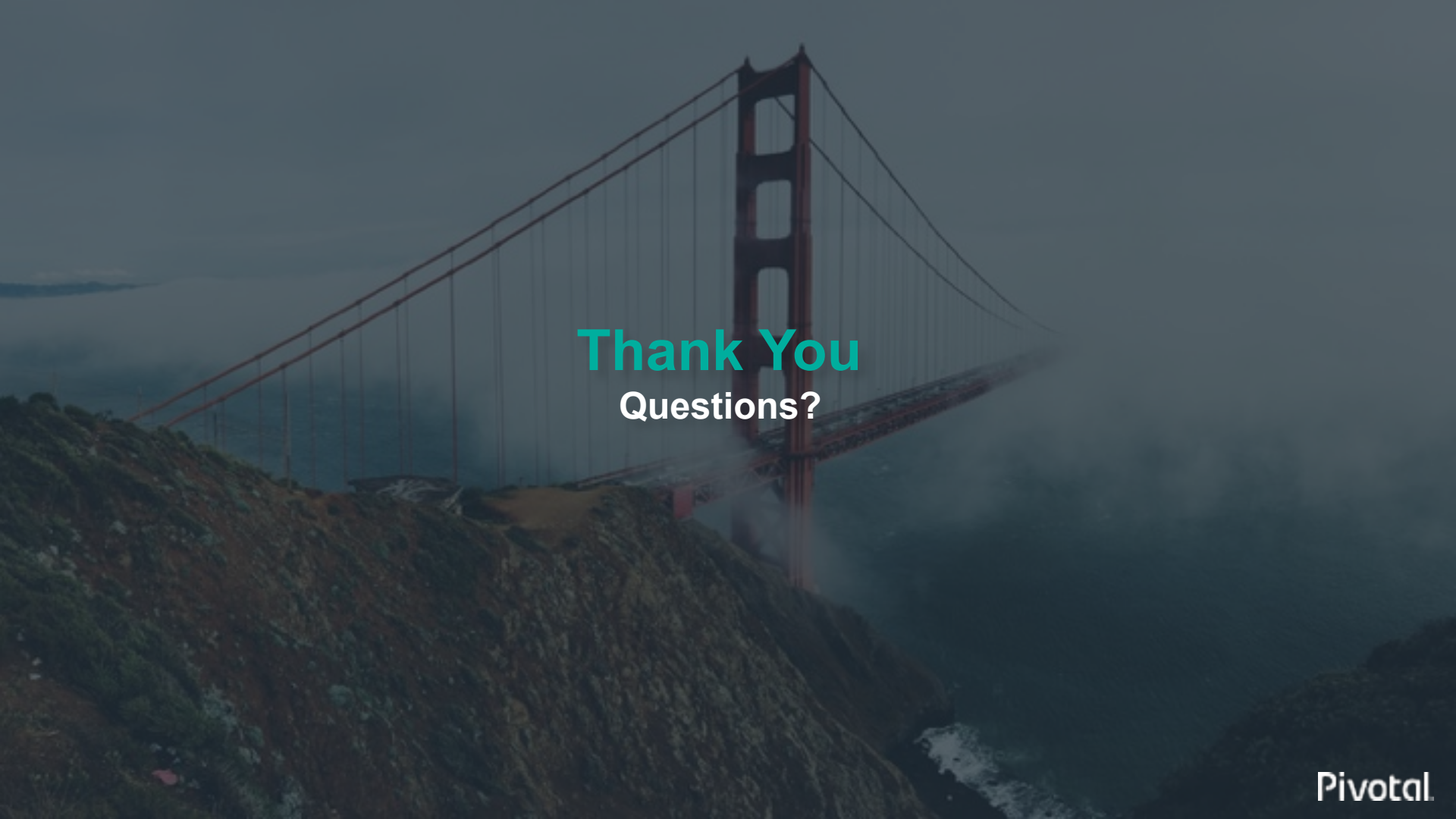
You can also create your own service brokers as they are applications deployed to cloud foundry that implement a specific interface exposed as REST endpoints.

Optimized for Spring Boot

If using Java/Spring Cloud Foundry has been optimized to make running Spring Boot apps really simple.

If we have time here's a demo using start.spring.io


Pivotal
Transforming How The World Builds Software


Thank You
Questions?
Pivotal

References

Sample Apps Repo
- (https://github.com/cloudfoundry/cf-acceptance-tests/tree/master/assets)

Spring Boot Starter
- (https://start.spring.io)