

Pivotal.

Open.
Agile.
Cloud-Ready.



Agenda

- Pivotal and Microsoft Partnership
- Stories from our Customers
- .NET on Pivotal Cloud Foundry
- More on .NET Apps
- More on windows diego cells

Pivotal and Microsoft Partnership

- Containerization of Windows
 - 2012 R2 approach
 - Namespace Isolation is less than Linux. Not recommended to run hostile code in windows. They will manage cpu, memory, disk.
 - 2016 approach & HyperV/Nano containers
- Bosh CPI for Azure
 - Msft built the CPI for Bosh
 - Aug 2016 (~PCF 1.8) OpsManager supports Azure

Pivotal and Microsoft Partnership

- Open Source Philosophy
 - .NET, etc - Open Sourced
 - Linux, etc. on Azure
- Support for Linux
 - SQL server
 - .NET
- Go To Market coordination
 - PCF on Msft price list

Types of Apps supported

- ASP.NET MVC apps
- ASP.NET Webform apps that do not use VM local session state
- OWIN apps
- Background processes (e.g. .NET command line apps)

Versions of .NET supported

- All versions of the .NET SDK supported by Windows Server 2012 R2 are supported on PCF
 - 3.5
 - 4.5
 - 4.5.x
 - 4.6

Reliability

- Container –level failover
- Support HA across Availability Zones (Diego)
- The Router will hold requests until an application instance is available to receive request.

Roadmap for ~2016

- Steel Toe (Spring Boot for .NET)
- Windows Bosh Agent
- SSH Support
- Windows 2016 Support, including Nano
- Windows 2016 Native Container Support
- Windows Virtual Networking
- CoreCLR & ASP.NET 5.0

.NET deployment

A few comments of what customers have told us about deploying on premise

Deploying a new app

- Provision a VM, IP, DNS, etc
- Windows updates...reboot...update...reboot
- Install IIS
- Create Website, configure app pool
- Configure SSL
- Configure Load Balancer

...now scale up

- get another VM
- template might have IIS installed
- duplicate your IIS config
- rework the load balancer

After all this, do you ever scale down?

Phew...now I can deploy my code

- manually doing one of:
 - Package your webapp as an MSI and coordinate pulling nodes out of the cluster to install
 - Login to the machine, download a zip file and copy it to c:\inetpub (or maybe run a .bat file)
 - The really modern teams are using NuGet for bits and most of the rest is the same

And when you have a problem

- Ops log in to each machine in the cluster and download log files and/or event log
 - A few hours (or days) later you get them and actually get to start diagnosing issues

Question

You want to do an isolated test of new code and need a place for it to run. Do you:

- A. Build a new vm and install everything?
- B. Clone an existing VM and hope you don't break anything? If so, how long do you wait?
- C. Find a "roomy" dev server and add it there?

Bonus Question

After that new environment gets created:

- A. It lives forever because no one will delete it?
- B. I gave a developer admin rights on the box,
they'll surely clean it up when they're done.
- C. Here is the URL for my ticketing system.

Pivotal.

.NET on Pivotal Cloud Foundry



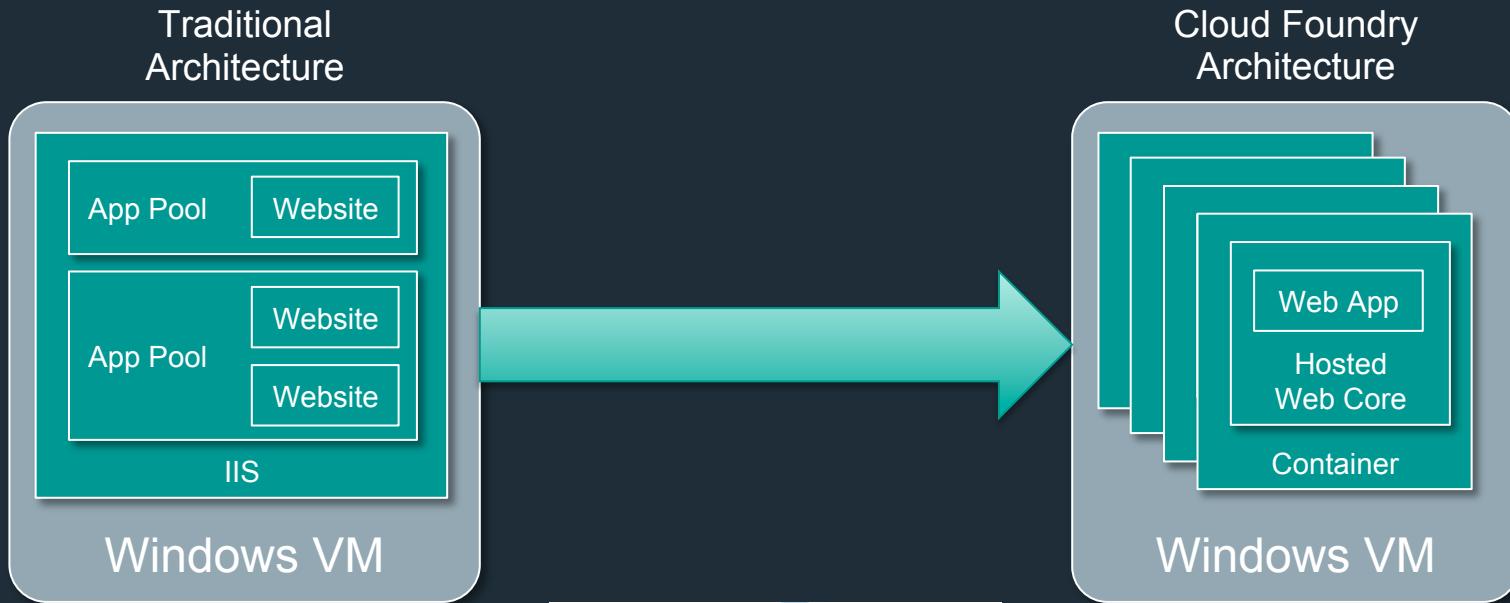
...with Cloud Foundry

- Deploy: cf push
 - deploy app, configure SSL/load balancing
- Scale: cf scale
 - add/remove app instances, updates load balancer
- View Logs: cf logs
 - Developers can see logs in real time!

Pushing a .NET Application

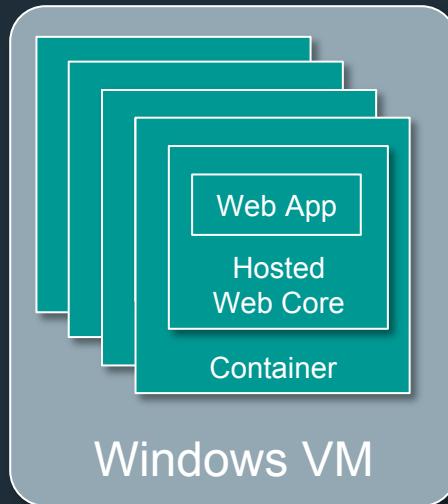
- To push:
 - binary buildpack
 - windows2012R2 stack
- Either publish or push from your project folder

App Pools, IIS, Containers, VM's, oh my.



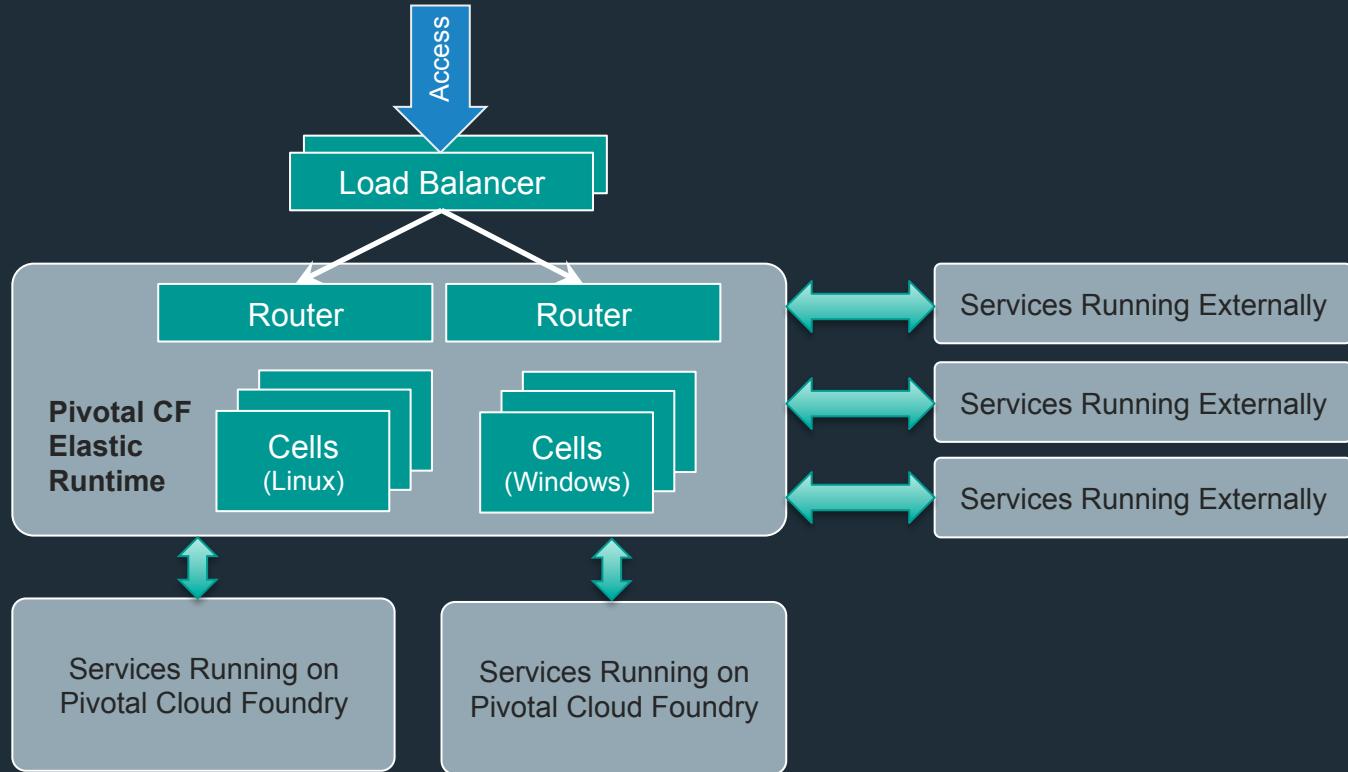
Isolation – Work in progress

Cloud Foundry
Architecture



Pivotal™

Cloud Foundry Architecture



Key Features

- High Availability
- Scaling – Manual and Autoscaling
- Notifications
- Security Groups
- Aggregated Logs & Metrics
- Environment Variables
- Marketplace and User-Provided Services

Integrating with Services

- Marketplace Services
 - Extract connection details from environment variables
- Connection Strings in Web.config
 - via User Provided Services
 - Spliced into Web.config during staging

Pivotal.

.NET apps



Writing 12 Factor .NET Apps

- Use an external session/cache provider
 - RDBMS, Redis or GemFire
- Override MachineKey in Web.config
- Use environment variables for environment config
- Use OAuth instead of Integrated Windows Auth
- Avoid the GAC and custom ISAPI handlers

Pivotal.

Operating .NET apps on Pivotal Cloud Foundry



Operations with .NET

- Build a pool of Windows VMs and connect to runtime
- Use a separate subnet or OpsManager exclusion list
- Monitor capacity using standard monitoring tools
(SCOM, vROps, etc)

Pivotal.

Open.
Agile.
Cloud-Ready.



Pivotal.

Supporting .NET apps on Pivotal Cloud Foundry



Diego on Windows - Preparation

- Windows Server 2012 R2
 - Install Windows Updates
 - Needs to be on a different network than BOSH
 - Alternatively assign IP from exclusion range
 - Using RDP? Disable CPU Fair Share
<http://bit.ly/cpufairshare>
- Download bits for network.pivotal.io
 - Setup.ps1
 - Generate.exe

Diego on Windows - Configuration

- Run `setup.ps1`
 - Configures Windows features
- Run `generate.exe` in cmd prompt
 - Specify bosh url (<https://un:pwd@director.sys.pcf.com>)
 - Provide a Windows username – must be part of Admin group
 - Provide password for the username
 - Specify output directory

Diego on Windows - Install

- Download bits from network.pivotal.io
 - DiegoWindows.msi
 - GardenWindows.msi
 - Save/Move them to the output folder
- Run install.bat in cmd prompt
 - Generated in the output folder

Diego on Windows - Verification

- Download bits from network.pivotal.io
 - hakim.exe
 - Creates a container to make sure everything works
- Check Task Manager | Services for running processes
 - GardenWindowsService
 - ContainerizeService
 - MetronService
 - RepService
 - ConsulService
- Run Smoke Tests
 - <https://github.com/cloudfoundry/cf-smoke-tests>
 - Be sure you enable windows tests in the JSON config file

Diego on Windows - Uninstall

- Evacuate Cells
 - Use the script here: <http://bit.ly/evacuatecell>
- Uninstall from Control Panel | Uninstall a program
 - In the following order
 - DiegoWindows
 - GardenWindows