Run .NET and SQL Server natively on Linux with OpenShift

Hit us up @OSS_Advocate @HaroldWong

John Osborne
Senior Developer Advocate
Red Hat

Harold Wong
Cloud Architect
Microsoft

Jason Dudash
Specialist Solutions Architect
Red Hat

Agenda

- Intro to .NET Core
  - Differences from .NET*
  - Development Tooling
  - Container implications for .NET apps
- In Action
  - .NET Core RHEL -> Containers -> OpenShift
  - Source to Image
  - SQL Server
    - Persistent Storage
    - Initializing Databases
  - Advanced
  - Tooling

Exciting times ahead for .NET

.NET FRAMEWORK

Multi-purpose, comprehensive framework for desktop and web applications

MODERN DEVICE EXPERIENCES

UNIVERSAL WINDOWS PLATFORM
Unified development across Windows devices

XAMARIN
Extend your reach to any device with .NET

MODERN CLOUD EXPERIENCES

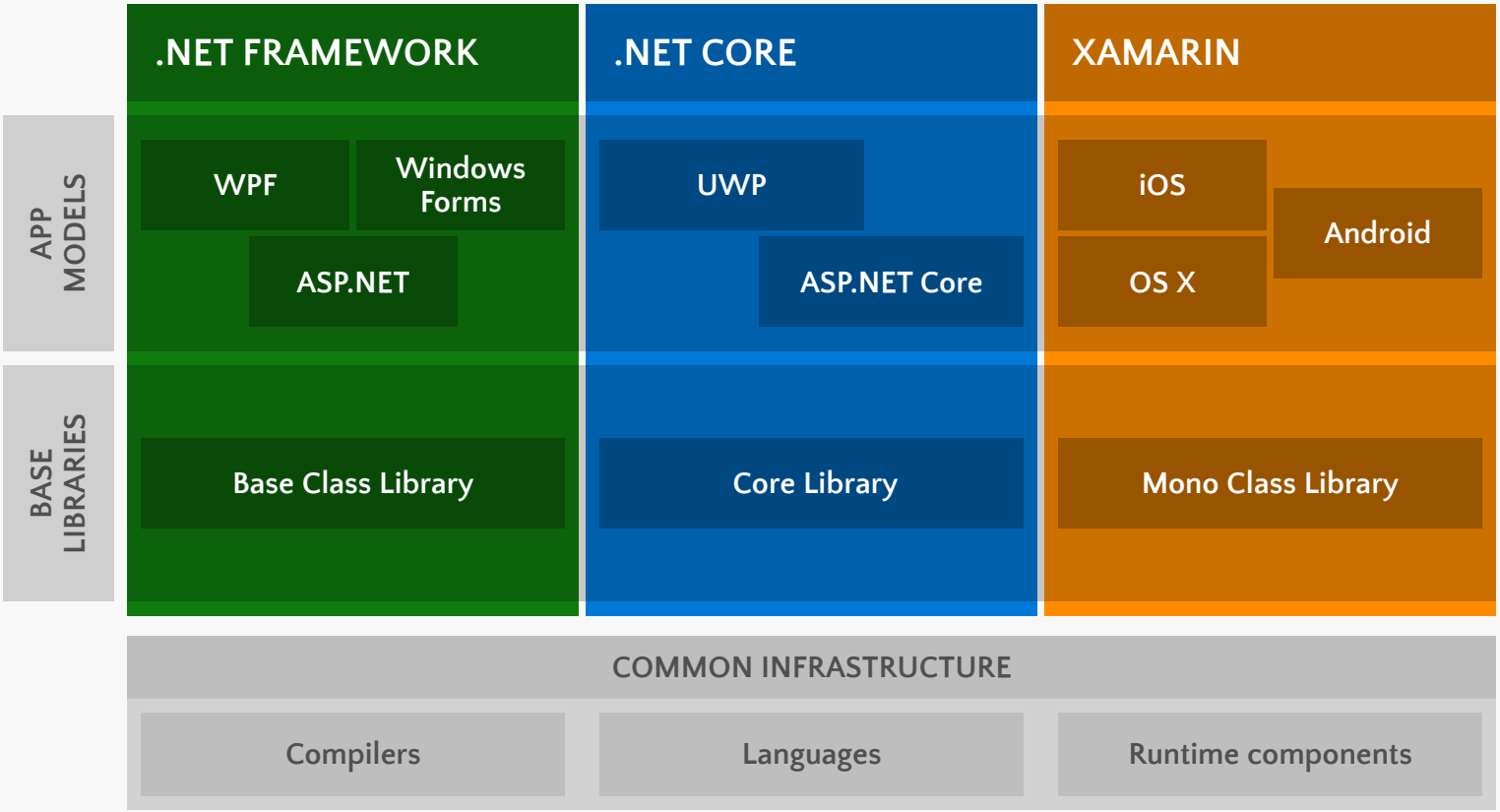
.NET CORE
Cross-platform, high performance .NET

ASP.NET CORE
Cloud optimized framework for micro services

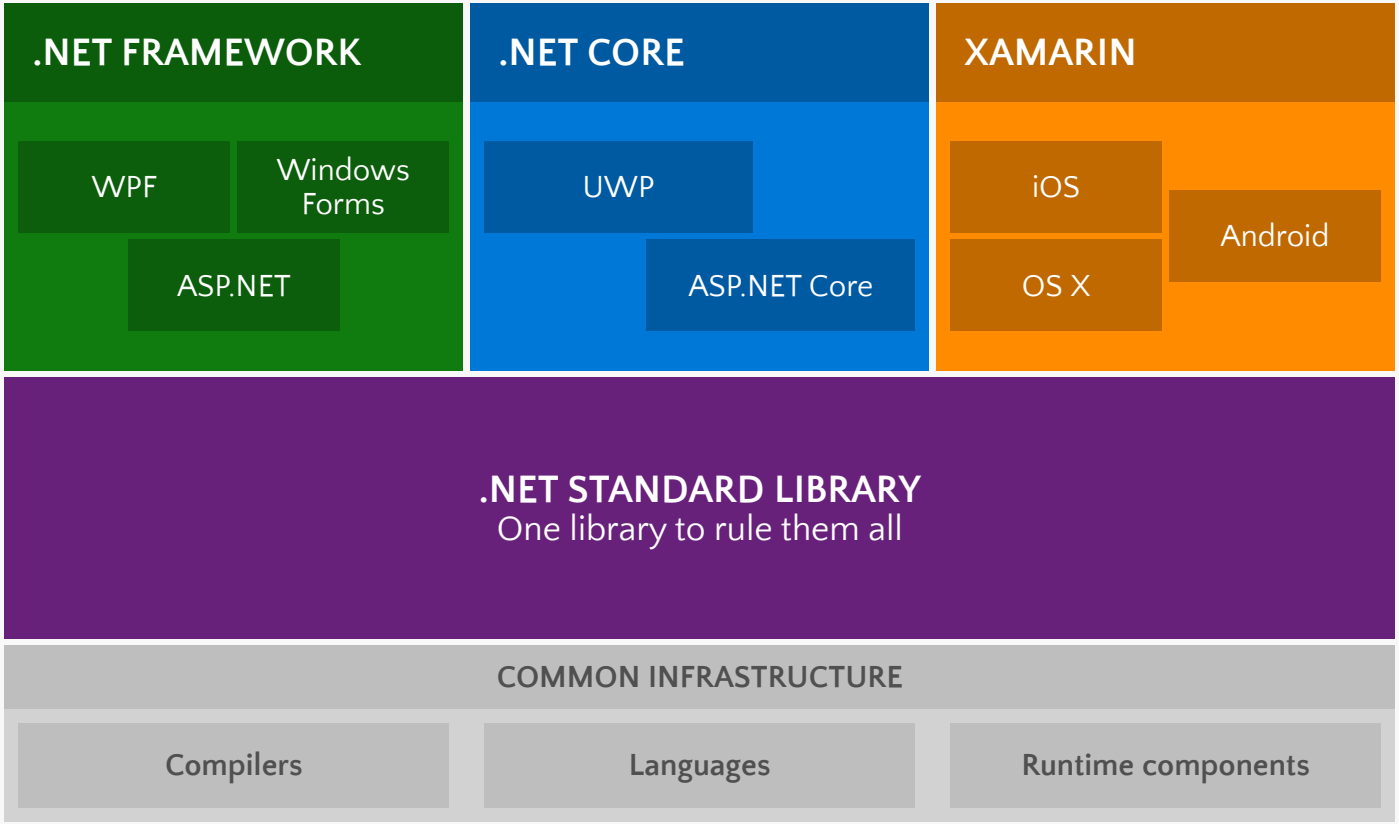
.NET today—the family gets bigger

.NET FRAMEWORK	.NET CORE	XAMARIN
Platform for .NET applications on Windows	Cross-platform and open source framework optimized for modern app needs and developer workflows	Cross-platform and open source Mono-based runtime for iOS, OS X, and Android devices
Distributed with Windows	Distributed with app	Distributed with app

.NET today—app models and libraries



.NET tomorrow



Languages

C#

Flexible, powerful, multi-purpose

- Investments in C# 7: Evolve for modern patterns in distributed applications: tuples, pattern matching, others.

VB

Simple, easy to use

- Investments in VB 15: Bring key language features and platforms while keeping VB simplicity principles.

F#

Elegant functional programming

- Investments in vNext: Language updates for C# 7 / VB 15 interop, platform support and tooling updates.

.NET innovation

.NET CORE

.NET CORE

- New open source and cross-platform .NET runtime and library stack
- High performance, including native compilation
- New set of command-line tools—".NET Core CLI"—for compiling and publishing apps
- Supports app local, "shared framework," and Docker deployment

ASP.NET CORE

- ASP.NET Core builds on top of .NET Core
- Single framework for web pages, services, and microservices
- Introduces concept of middleware pipeline, enabling you to inject as little or much functionality as needed
- Fully integrates with CLI tooling and uses the shared framework
- Takes advantage of .NET Core performance and includes a very high performance web server, built on LibUV

Tools for any developer and any app


Visual Studio


Visual Studio Code


Xamarin Studio


Command Line Interface

OPEN

- Any App: Desktop, Mobile, Server, Cloud
- Any Developer: IDE, Code editor, CLI
- Any OS: Windows, OS X, Linux

POWERFUL

- Easy and quick installation
- Better productivity with reimagined inner loop
- First-class .NET experiences

ASP.NET

Improved tooling and frameworks

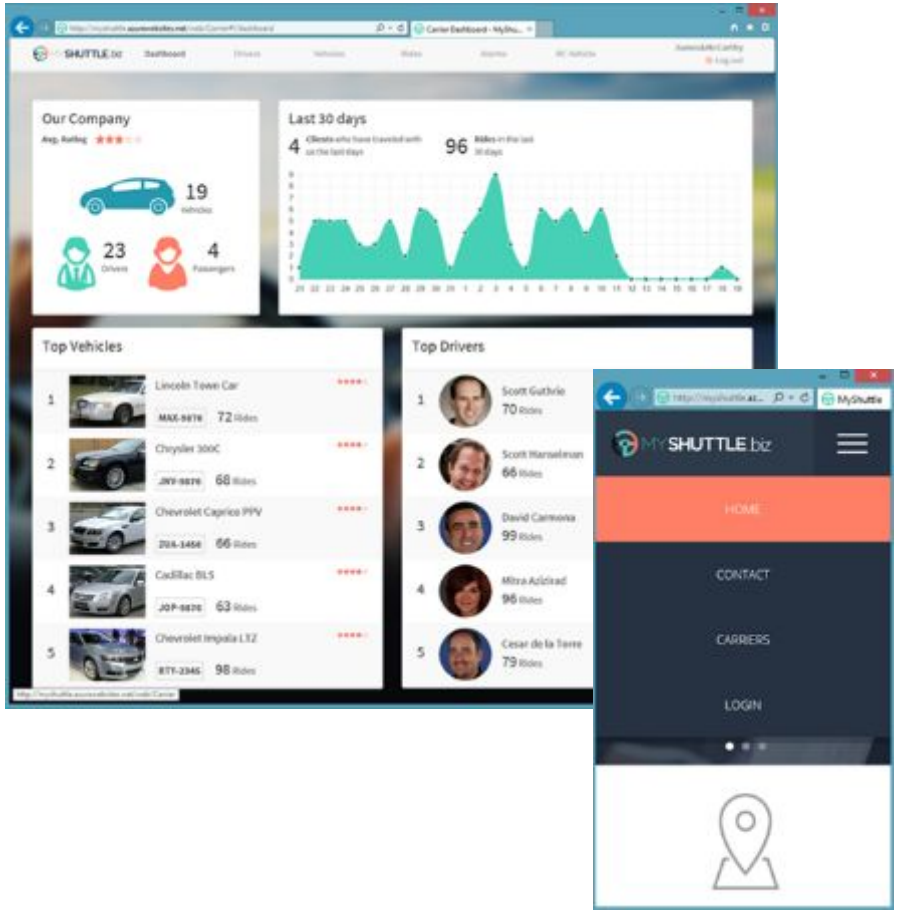
- Deliver value faster with improved tooling and frameworks

ASP.NET Core 1.0

- Smaller footprint
- Modular
- Faster
- Any Platform

Cloud-Ready

- Tools and frameworks ready for seamless transition to cloud.
- Remote diagnostics on the cloud.
- Container support via Docker.



.NET Core

Cross Platform 

You can create .NET Core apps that run on Windows, Linux and Mac.

Open Source 

Runtime, libraries, compiler, languages and tools are all open source on GitHub where contributions are accepted, tested and fully supported

Lightweight 

No impact deployment and a modular development model where you only take dependencies on the minimal set of packages you need

Modern 

Multiple language support with C#, VB, F# and modern constructs like generics, Language Integrated Query (LINQ), async support and more

Flexible 

.NET Core supports multiple editors and development environments with a simple set of command line tools available across operating systems

Familiar 

Reuse code and skills using the same languages, compilers and libraries across the multiple .NET platforms

Fast 

Native compilation across platforms and high performance ASP.NET Core is 8x faster than Node.js and 3x faster than Go

Open source .NET

Platforms

- General purpose **.NET Core** runtime, compilers and libraries
- **ASP.NET Core** web server stack
- **Xamarin** SDK (runtime, libraries, command line tools)

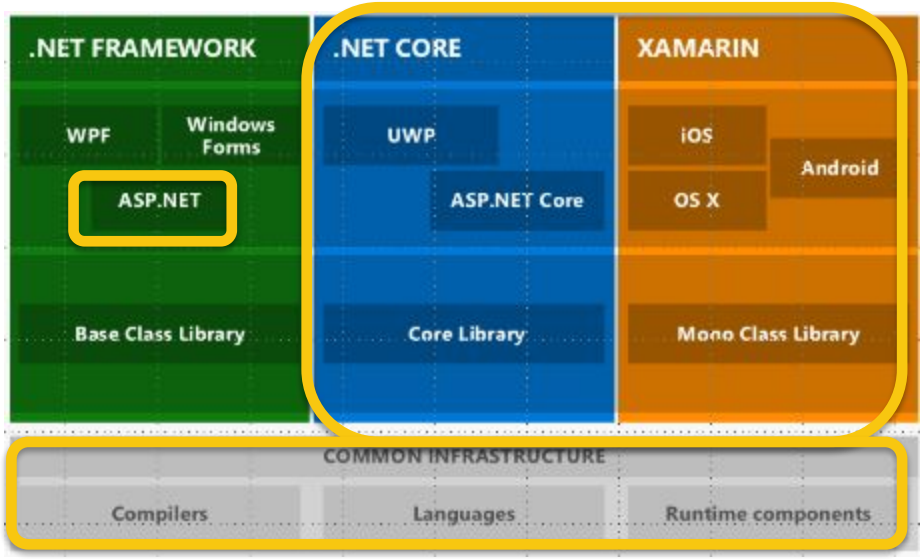
Fully supported cross-platform

- Windows, Linux and OS X
- Visual Studio tooling support (e.g. debugging and deploying to Docker in Linux)
- Omnisharp extensions to cross-plat IDEs (Sublime, Emacs...)

Open Source

- .NET Core and ASP.NET Core source being developed on GitHub
- Contributions accepted, tested and fully supported
- Close collaboration with Mono community

What is open source?



Get started from:

github.com/microsoft/dotnet

.NET Core and SQL Server In Action

.NET Core on RHEL without OpenShift

Installing and testing a simple app in Red Hat Enterprise Linux without containers

- Enable the proper .NET Core channel so you can access rpms
  - e.g. `subscription-manager repos --enable=rhel-7-server-dotnet-rpms`
- `yum install rh-dotnetcore11`
- `yum install scl-utils` and then `scl enable rh-dotnetcore11 bash`
- `dotnet new`
- `dotnet restore`
  - *What's restore do? … NuGet packages*
- `dotnet run`

Developing .NET Apps on OpenShift

How about .NET Microservices running in a polyglot container platform?

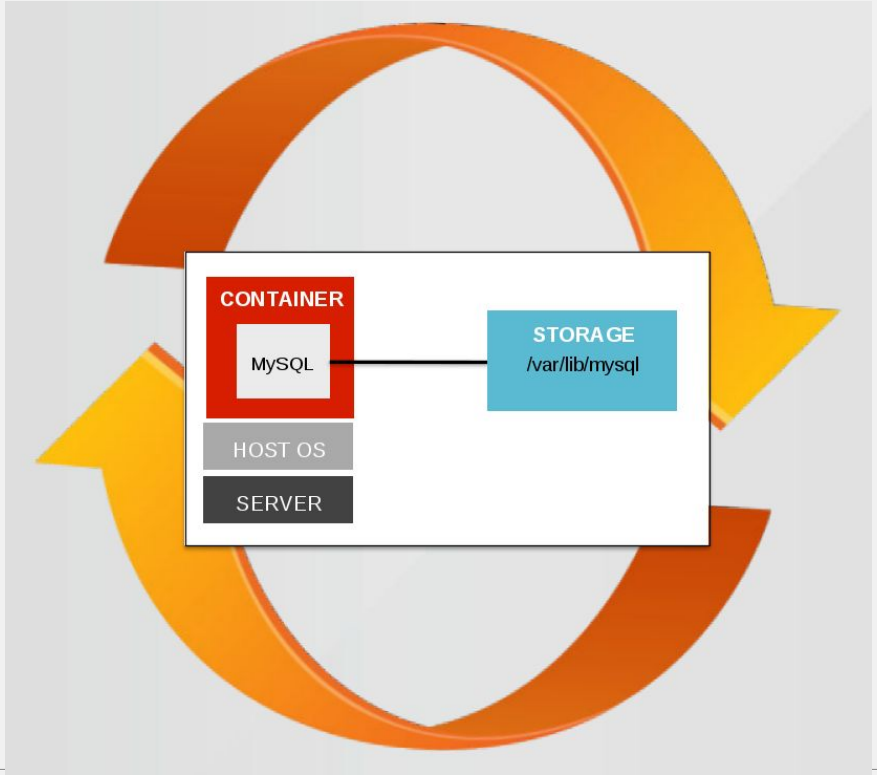
- .NET app development on OpenShift
  - .NET Core in containers - packaged with everything it needs to run
  - Option 1 - use the image available in Red Hat's registry:
    - `FROM registry.access.redhat.com/dotnet/dotnetcore-11-rhel7`
  - Option 2 - use Source 2 Image (S2I)
- Why S2I?
  - Leverage pre-made and supported base container images
  - Layers just your code on top
  - Developers focus on the code, not the Dockerfiles and container building
    - You can customize the S2I using scripts and env variables

Deploying .NET Apps via S2I

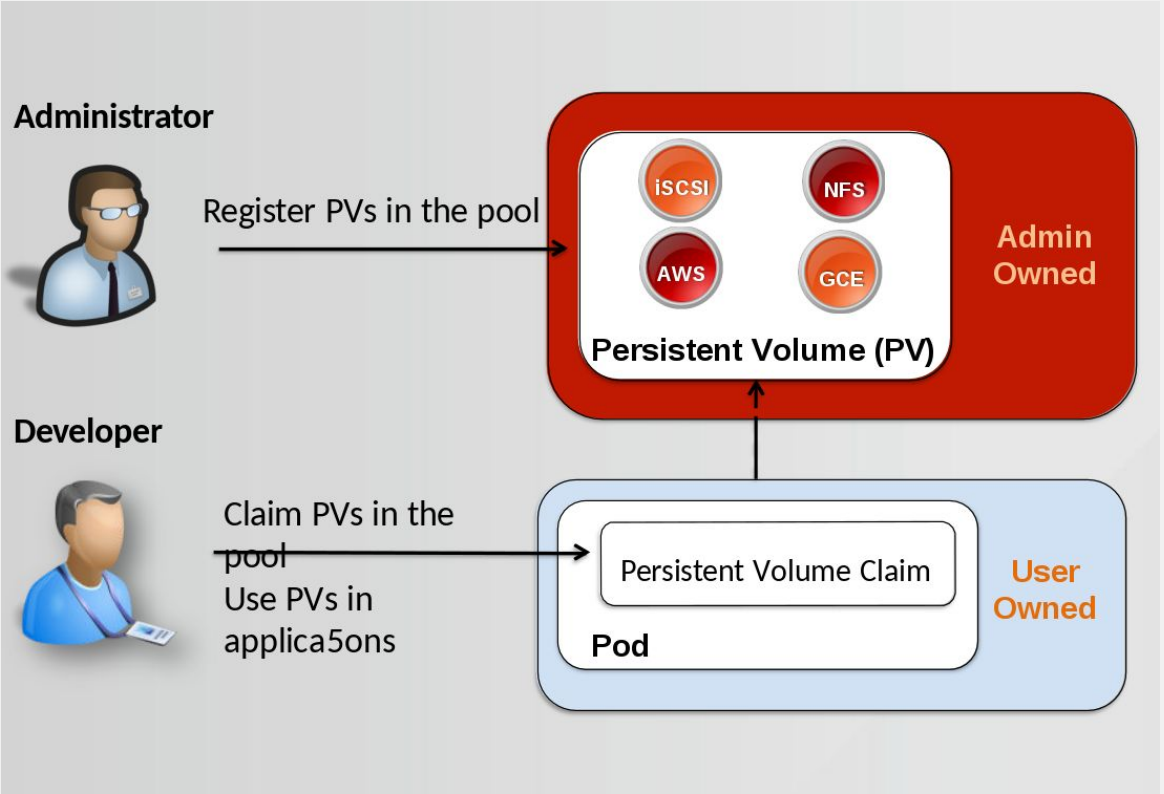
Let’s see a demo of deploying a .NET web service with S2I

SQL Server

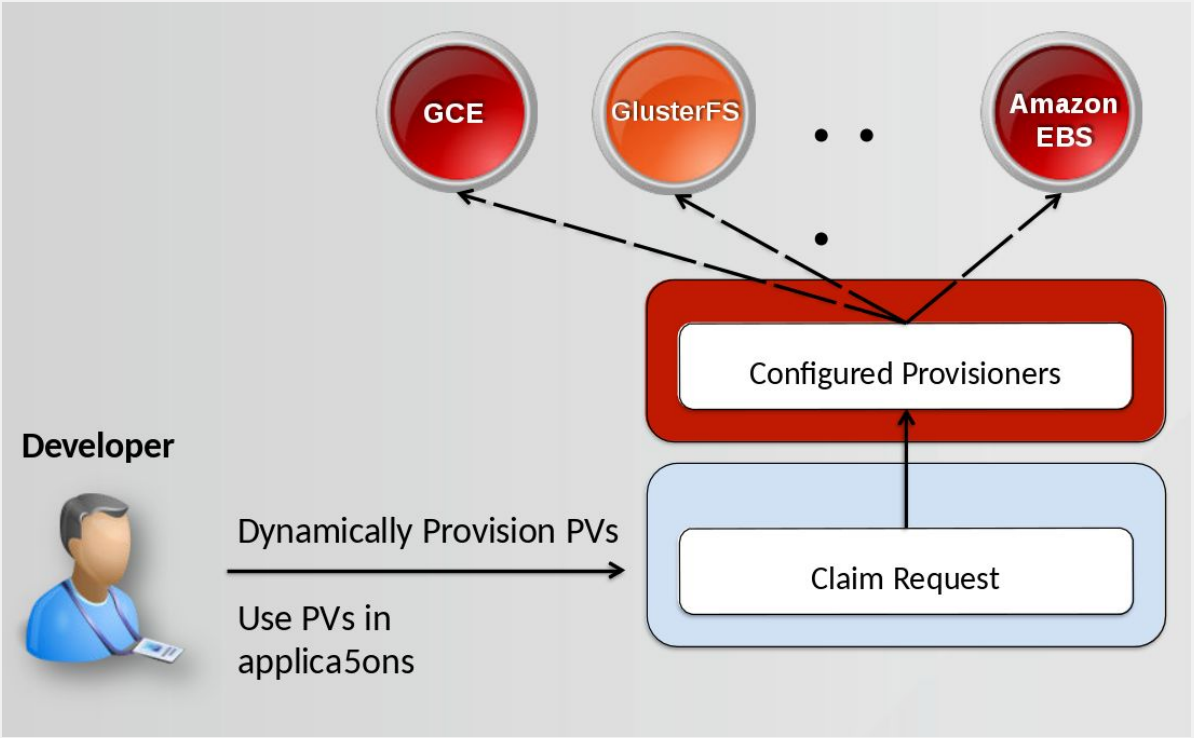
Persistent Storage



Persistent Storage (Static Method)



Persistent Storage (Dynamic Method)



Initializing Databases

- Create custom image with already loaded
- Automate some process that lives outside of OCP (i.e. Jenkins)
- Use a container
  - Pod Lifecycle Hooks
  - InitContainers
  - Co-located container in the pod
  - Job (hint: use templates!)
  - Bake it into the app
    - Consider (Extended builds or Multi-State Builds)

References

Things we've referenced and more!

- Open Shift .NET Core S2I reference docs
- Red Hat developers .NET page
- Demo source code
- .Net Core containers in the Red Hat container registry
- Red Hat support page for .NET Core

THANK YOU


plus.google.com/+RedHat


facebook.com/redhatinc


linkedin.com/company/red-hat


twitter.com/RedHatNews

youtube.com/user/RedHatVideos