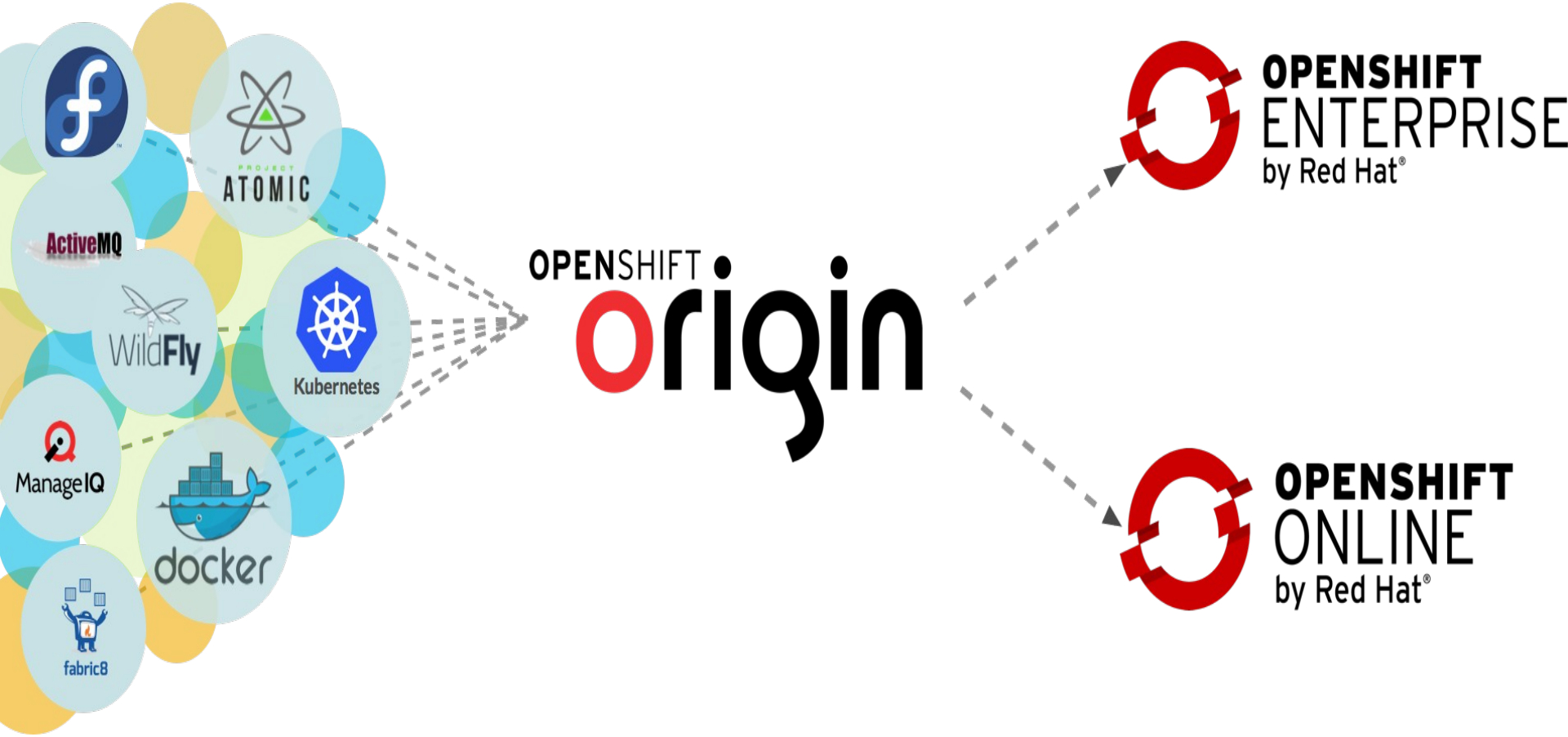

OPENSHIFT
by Red Hat
OpenShift Enterprise 3
Overview and Demo

Agenda

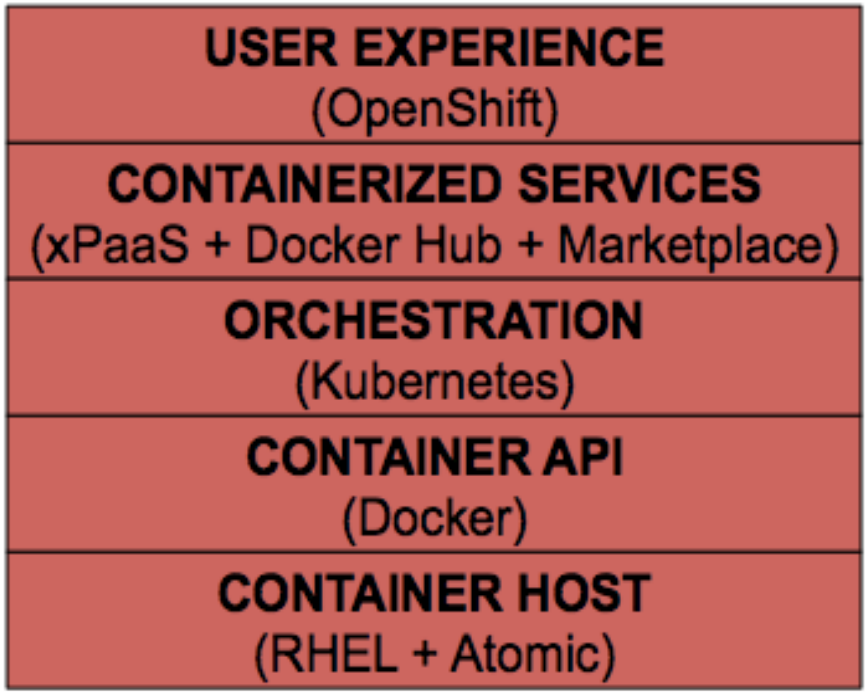
- OpenShift Enterprise 3
  - Overview
  - Concepts
- OpenShift Enterprise 3 “How it Works”
- OpenShift Enterprise 3 Demo
  - Docker Image Deployments
  - Scale
  - Source to Image
  - New Builds
- OpenShift Enterprise 3 RoadMap

Community Powered Innovation



OPENSHIFT STACK V3

- Standard containers API
- Container-optimized OS
- Web-scale orchestration
- Expanded choice of services
- Enhanced developer and operator experience
- Industry standard PaaS stack!



OPENSHIFT V3 CONCEPTS

Docker Image: Defines a filesystem for running an isolated Linux process (typically an application

Docker container: Running instance of a Docker image with its own isolated filesystem, network, and process spaces

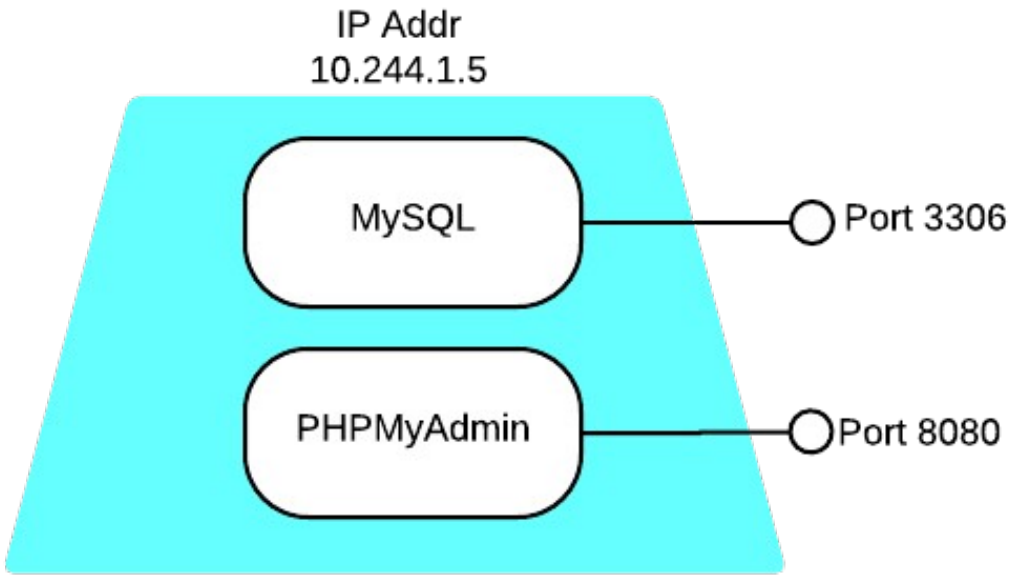
Pod: Kubernetes object that groups related Docker containers that need to share network, filesystem or memory together for placement on a node. Multiple instances of a Pod can run to provide scaling and redundancy.

Replication Controller: Kubernetes object that ensures N (as specified by the user) instances of a given Pod are running at all times.

Service: Kubernetes object that provides load balanced access to multiple pods. Services are accessible only inside the OpenShift environment (non-user facing).

OPENSHIFT V3 CONCEPTS

- A single pod with two containers each exposing a port on pod's IP address

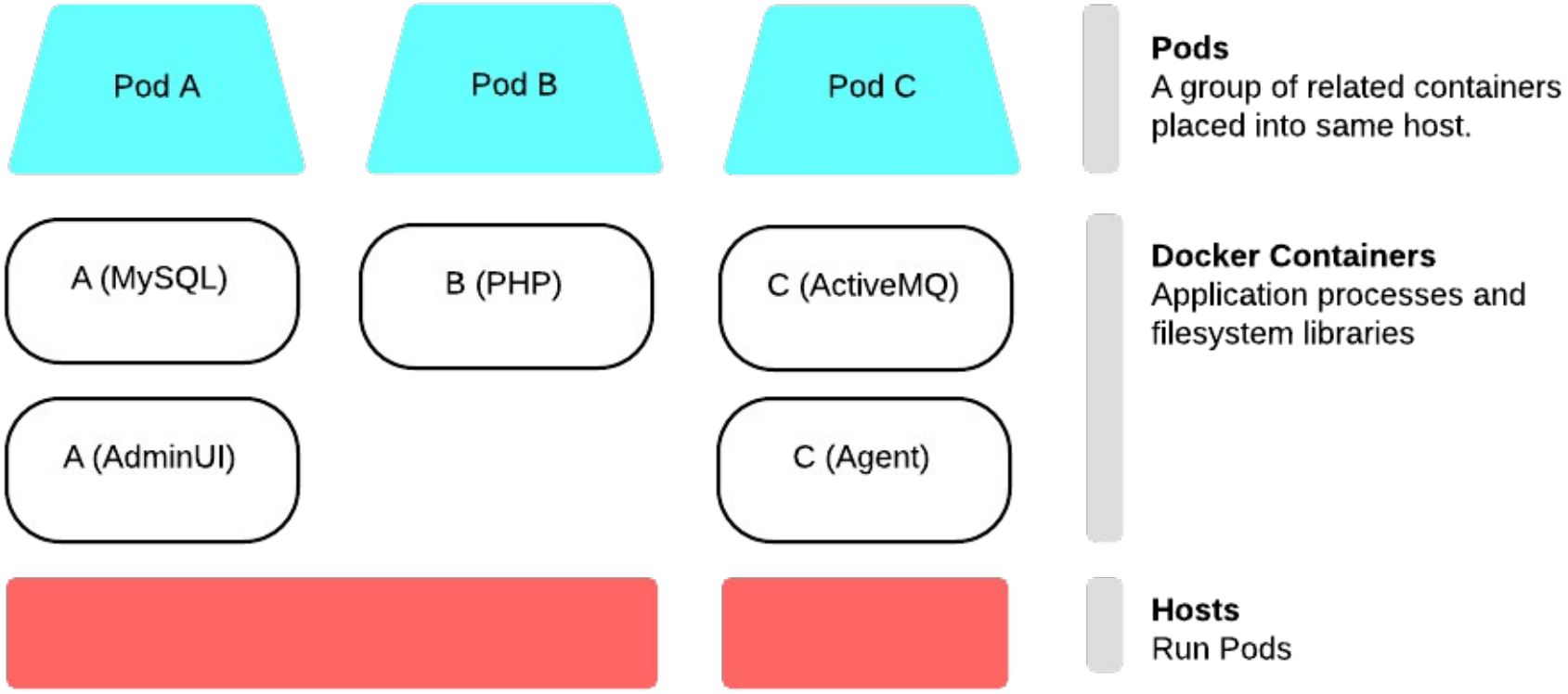


Pod Networking
Each Pod has a single IP address. All containers within the pod share
the same IP and port space.

Shared Ports
Each container must share pod ports. No conflicts allowed

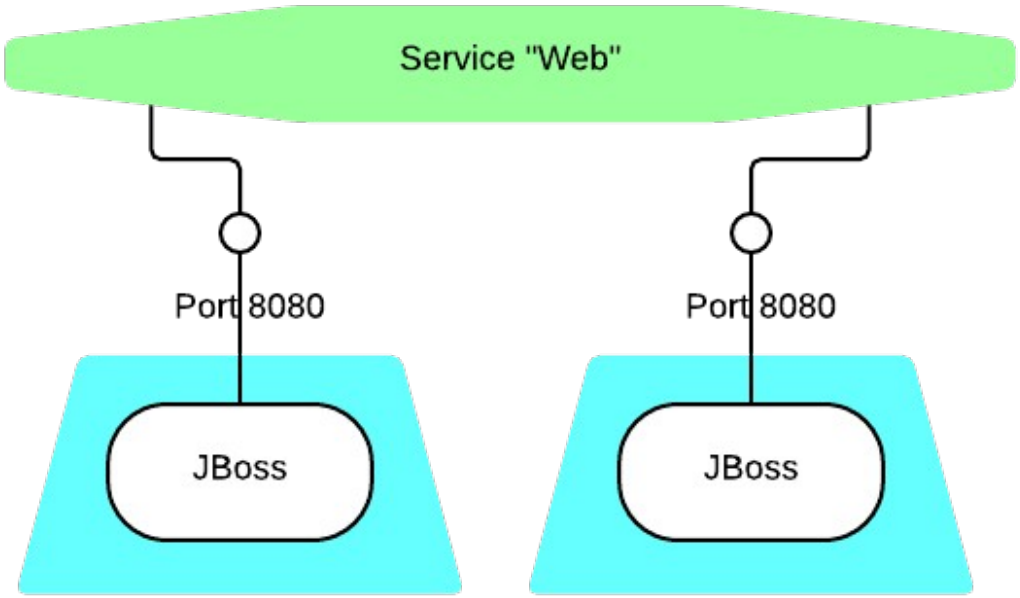
OPENSHIFT V3 CONCEPTS

Three different pods each running a set of related containers



OPENSHIFT V3 CONCEPTS

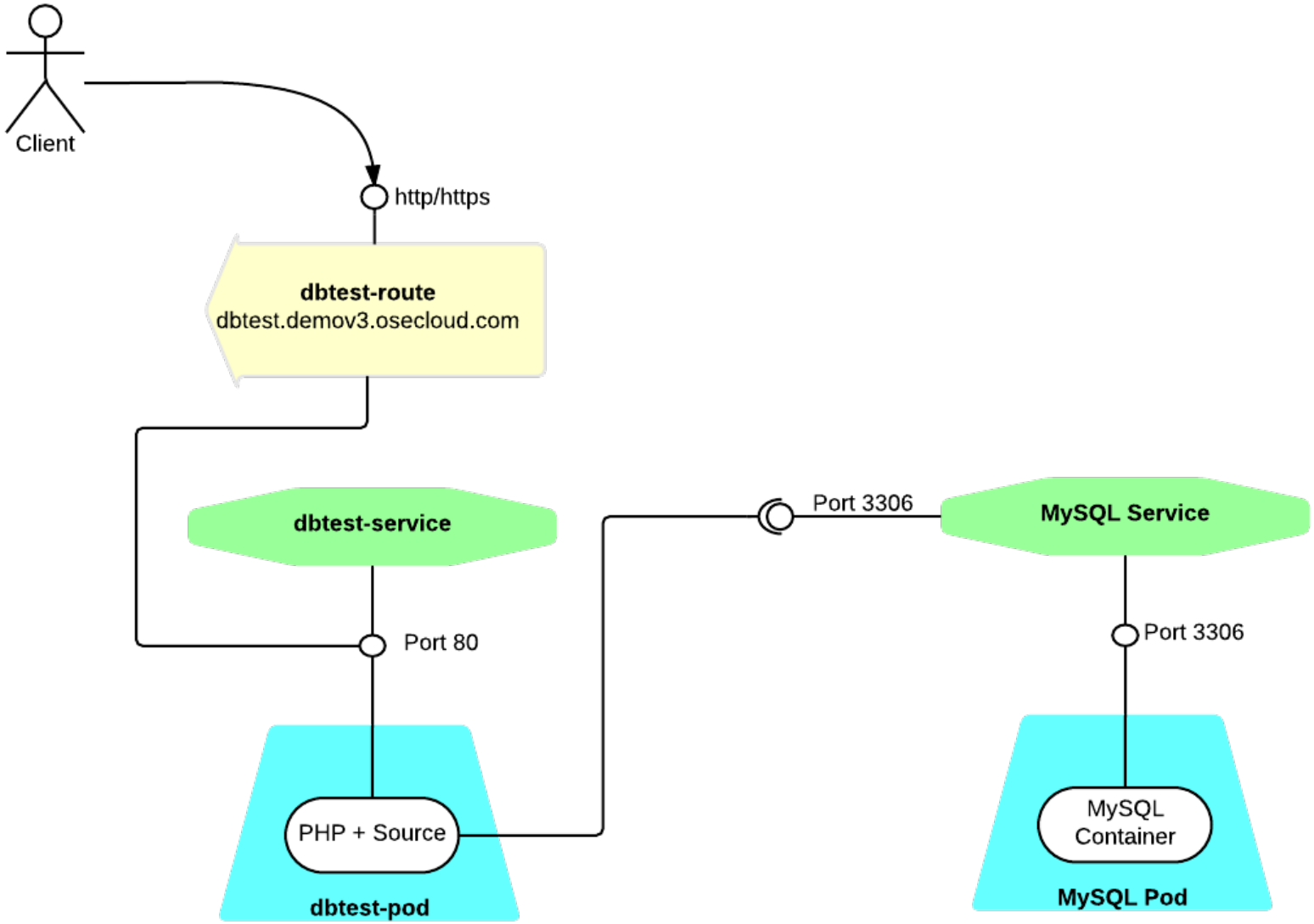
Multiple instances of a single pod are load balanced and accessed via a Service



Services balance Pods
A Service is an IP:Port that provides balanced access across a group of pods. Services are available inside the OpenShift environment, but are not public-facing.

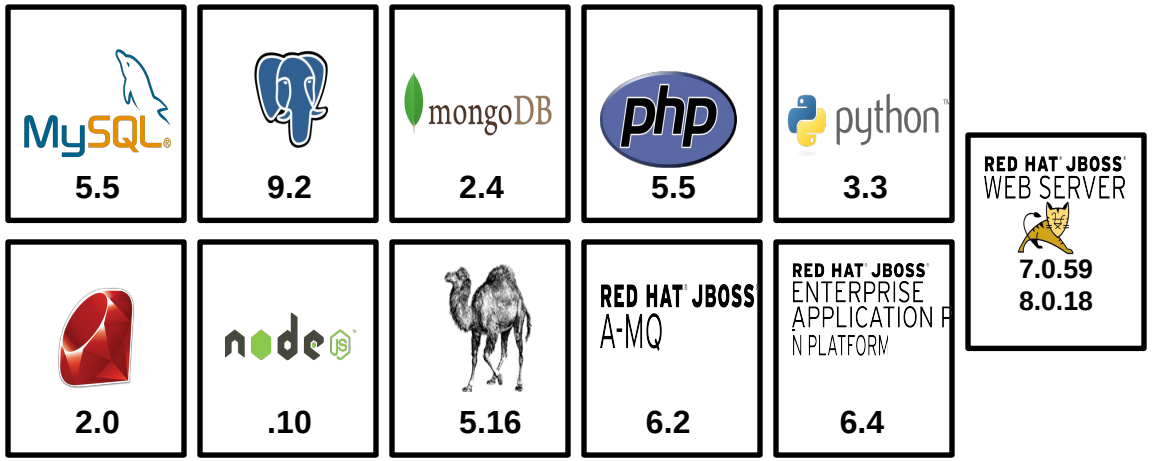
Replication Controllers ensure availability
An RC ensures that N copies of a pod exist at all times. Automated manipulation of RCs is how OpenShift achieves autoscaling.

EXAMPLE: PHP APPLICATION INTERACTING WITH MYSQL DATABASE

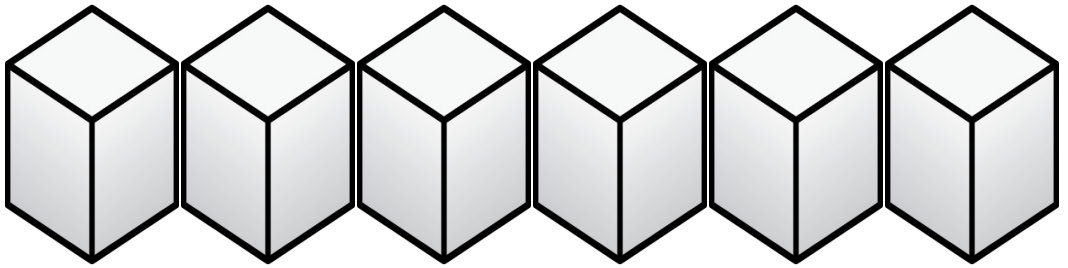


Massive Supported Ecosystem

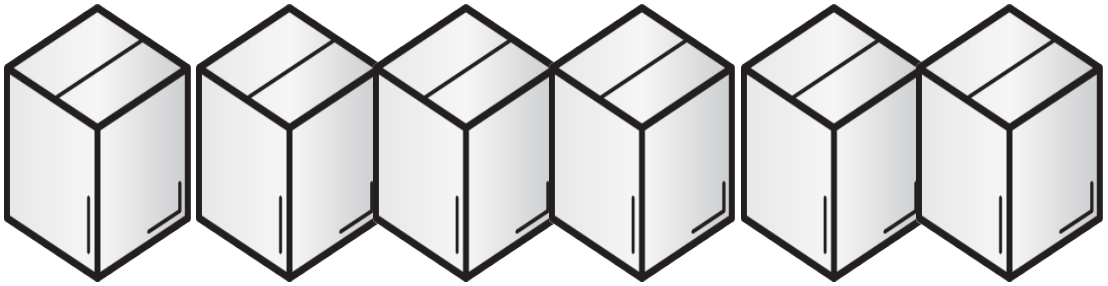
Software Collections & JBoss
CVE Fixes
Bug Patches
Support Life Cycle
Technical Support



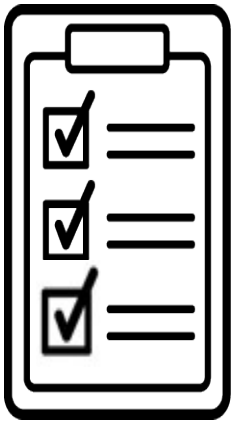
Red Hat Certified Containers
CVE Fixes
Bug Patches
Technical Support



Any Docker Registry
Supported Container API
Supported Execution



Application Design Freedom

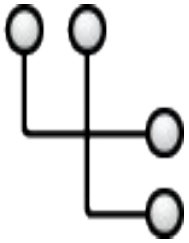


Persistent File System

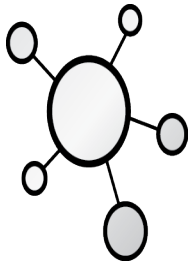
Remotely Accessible

Real Addressable IP

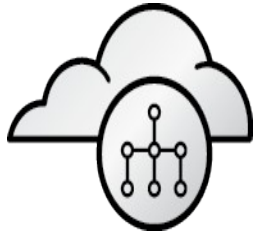
Classic 3 Tier Enterprise Apps



Cloud Native Microservices



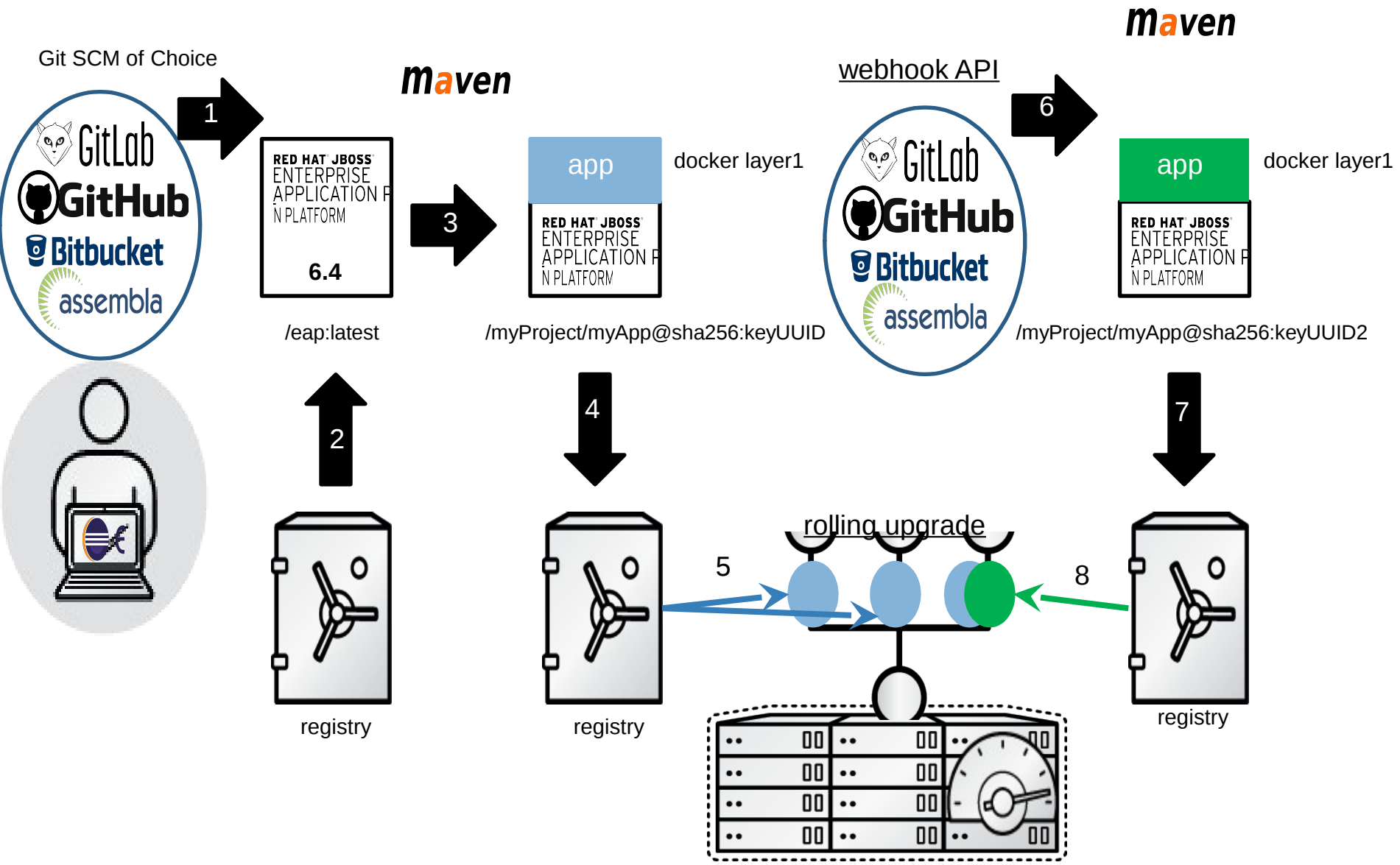
Off PaaS Service Mixtures



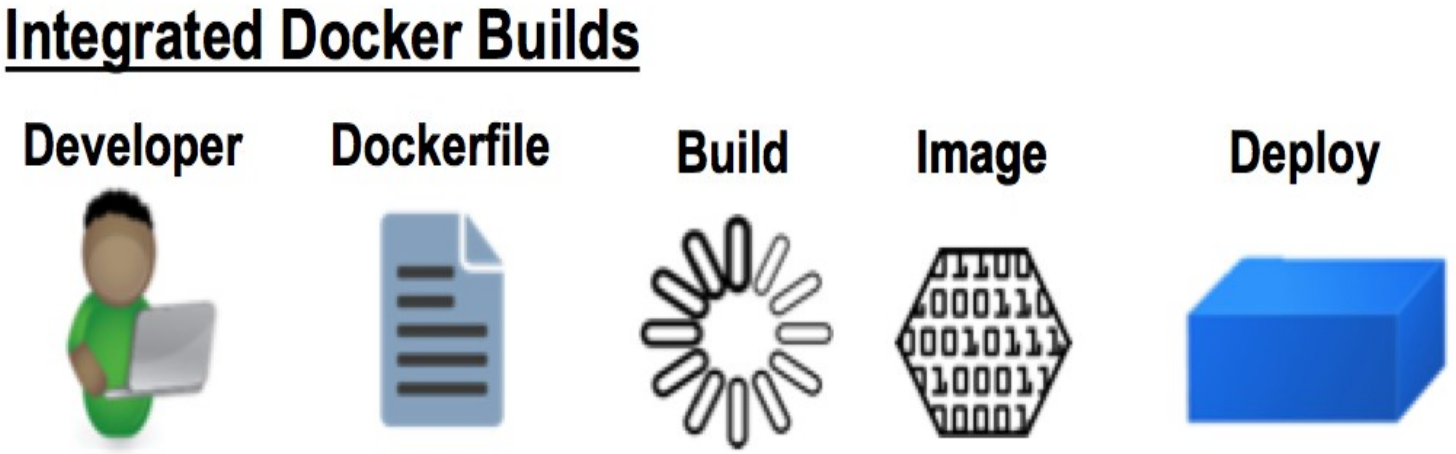
Opinionated PaaS

1.) Deploy Just a Database Service
2.) Cluster Services
3.) Share File System Between App Instances
4.) Start Deployment in the Frontend or Backend
5.) Increased Routing Control
6.) Dependable IP addressing

DevOps Experience



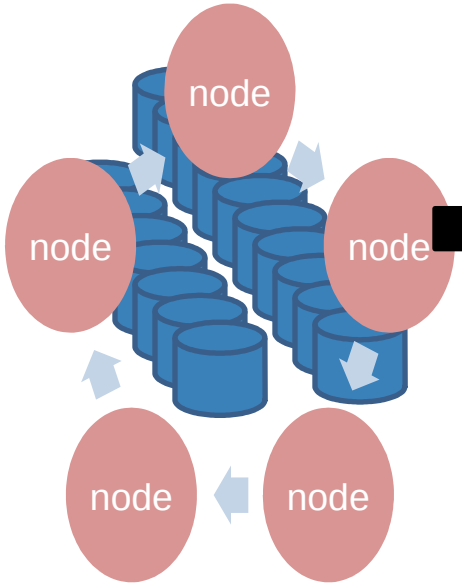
When Docker is Enough: new-app



Persistent Remote Storage

Define Storage Volumes

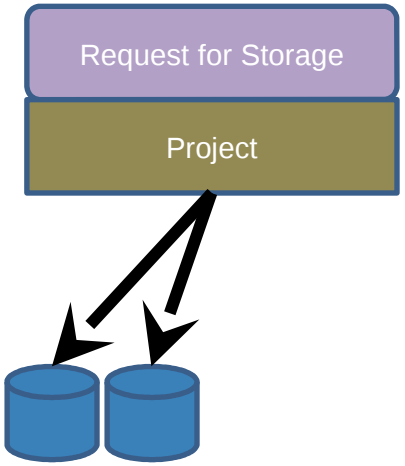
- Member of the Cluster
- Backed by an actual storage entity
- Such as NFS, GCE pDisk, Amazon EBS




```
spec:
 capacity:
   size: 10
 awsElasticBlockStore:
   volumeId: "abc123"
```

Manage Volume Claims

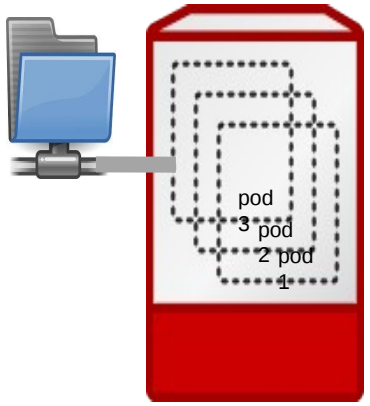
- Member of the Namespace
- Resource controlled by the Project Quota




```
spec:
 accessModes:
   - ReadWriteOnce
     ReadOnlyMany
```

Map Volumes to Pods

- Volume to a Pod
- Mounts the file system to the volume
- Volume can unmount without lost of data

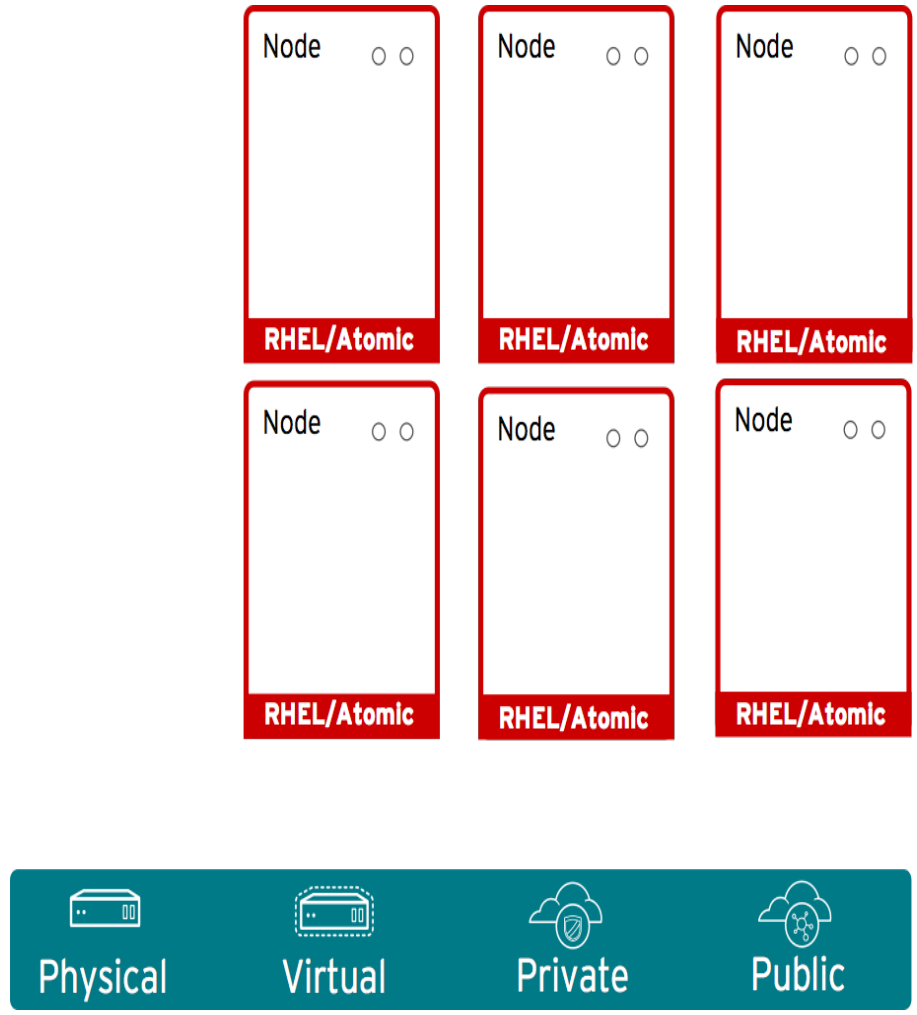



```
spec:
 containers:
   - image: dockerfile/nginx
     name: myfrontend
     volumeMounts:
     - mountPath: "/var/www/html"
       name: mypd
```

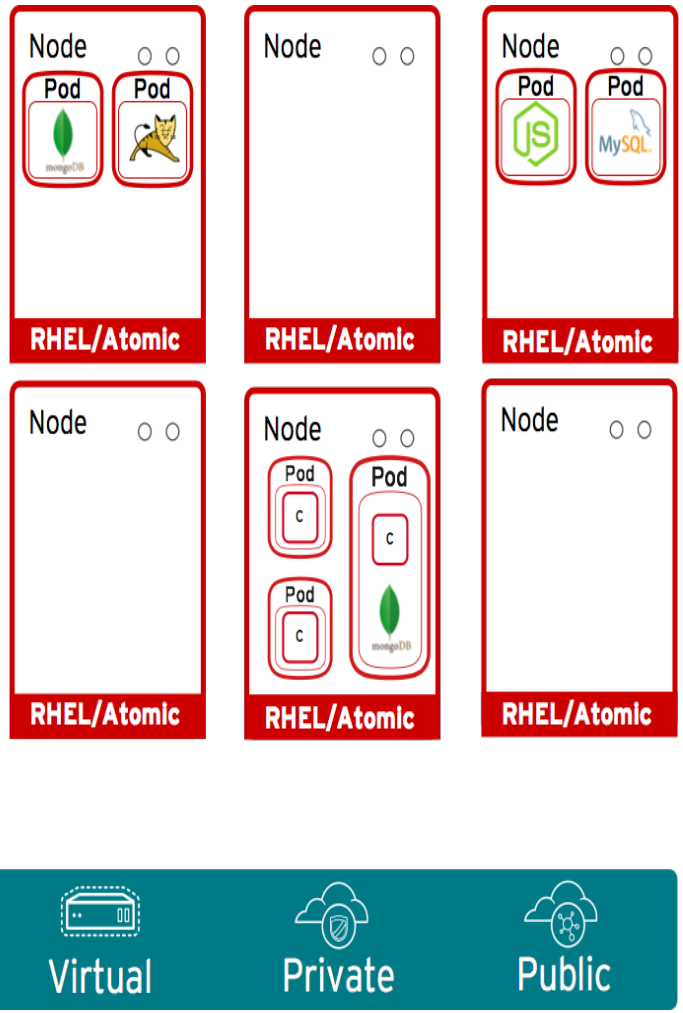

OpenShift Enterprise 3
“How it Works”

OpenShift Runs on Your Choice of Infrastructure

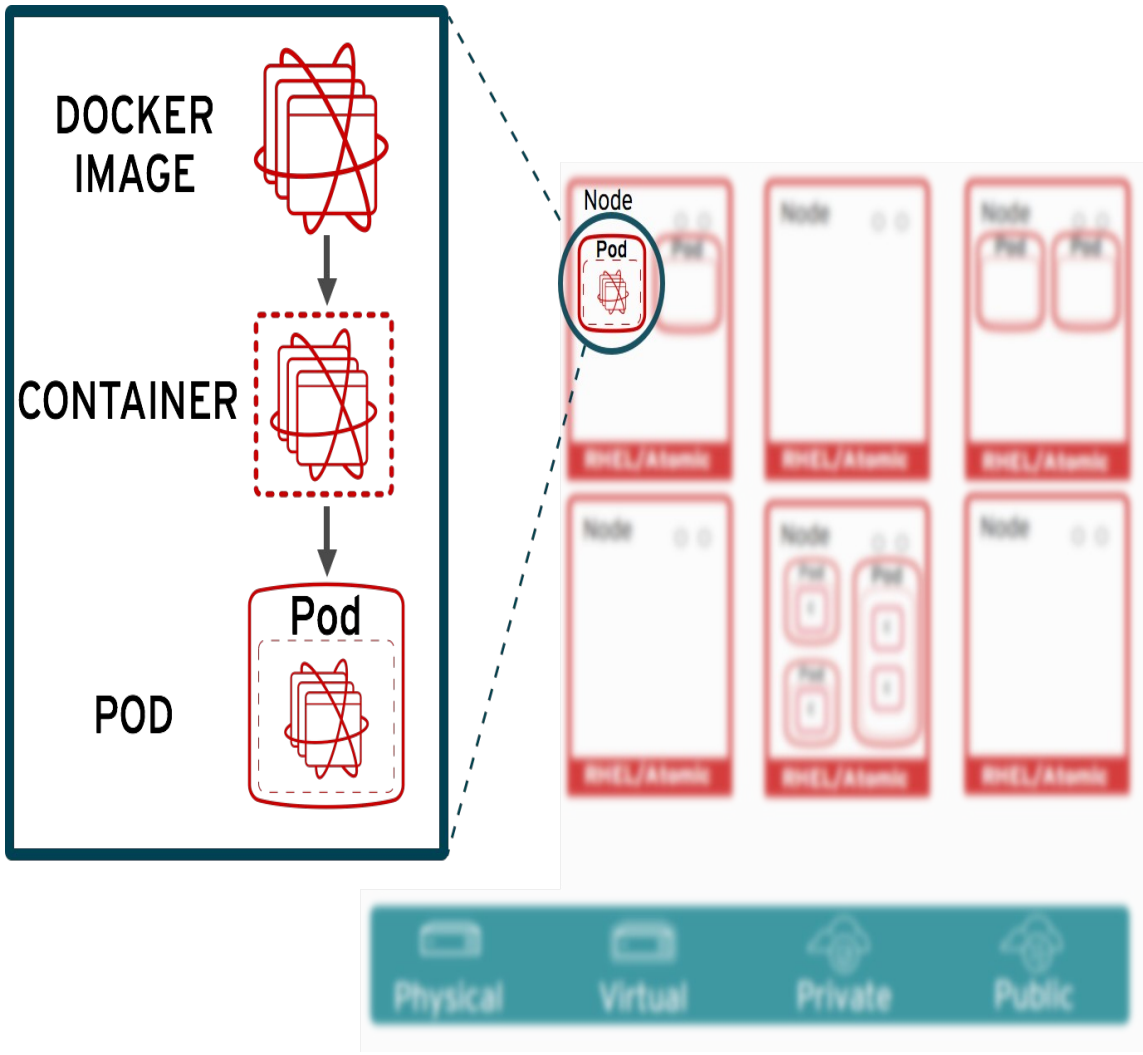
Nodes Are Instances of RHEL Where Apps Will Run



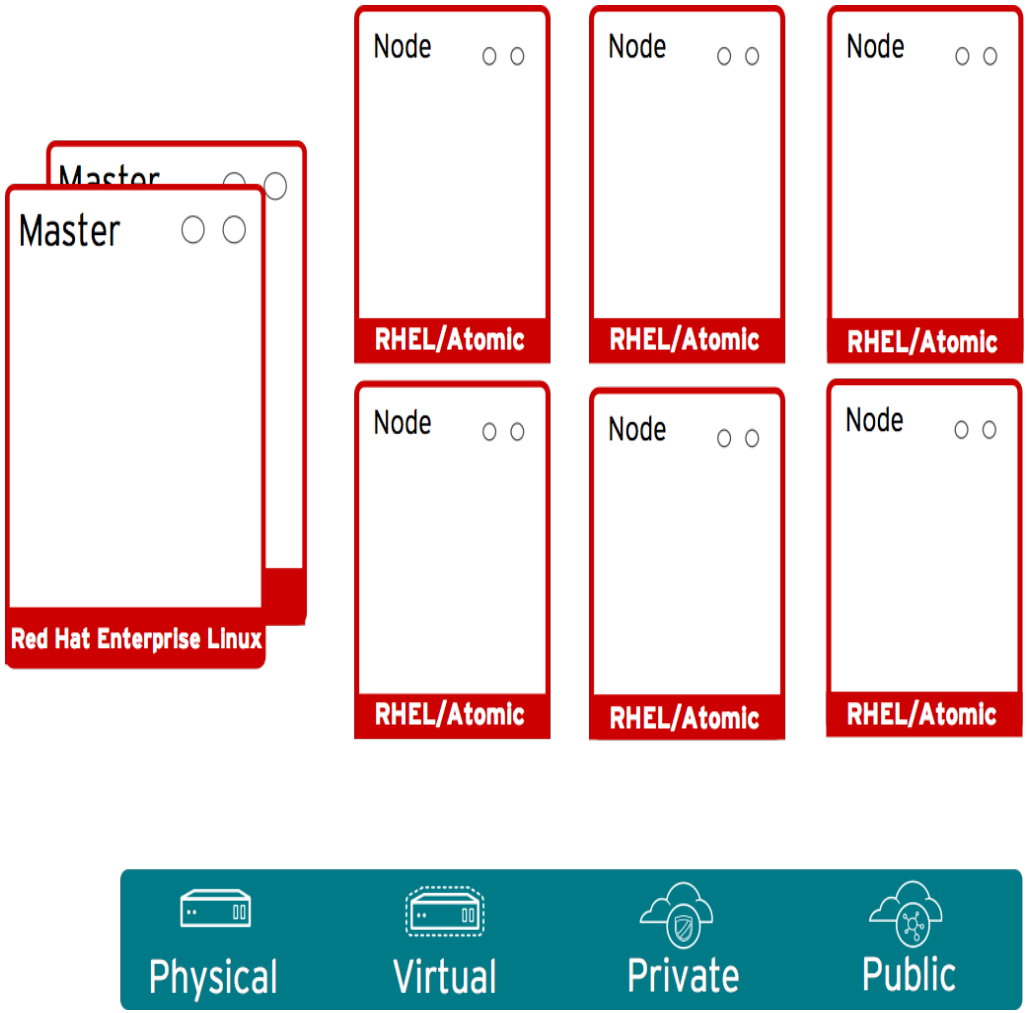
App Services Run In Docker Containers On Each Node



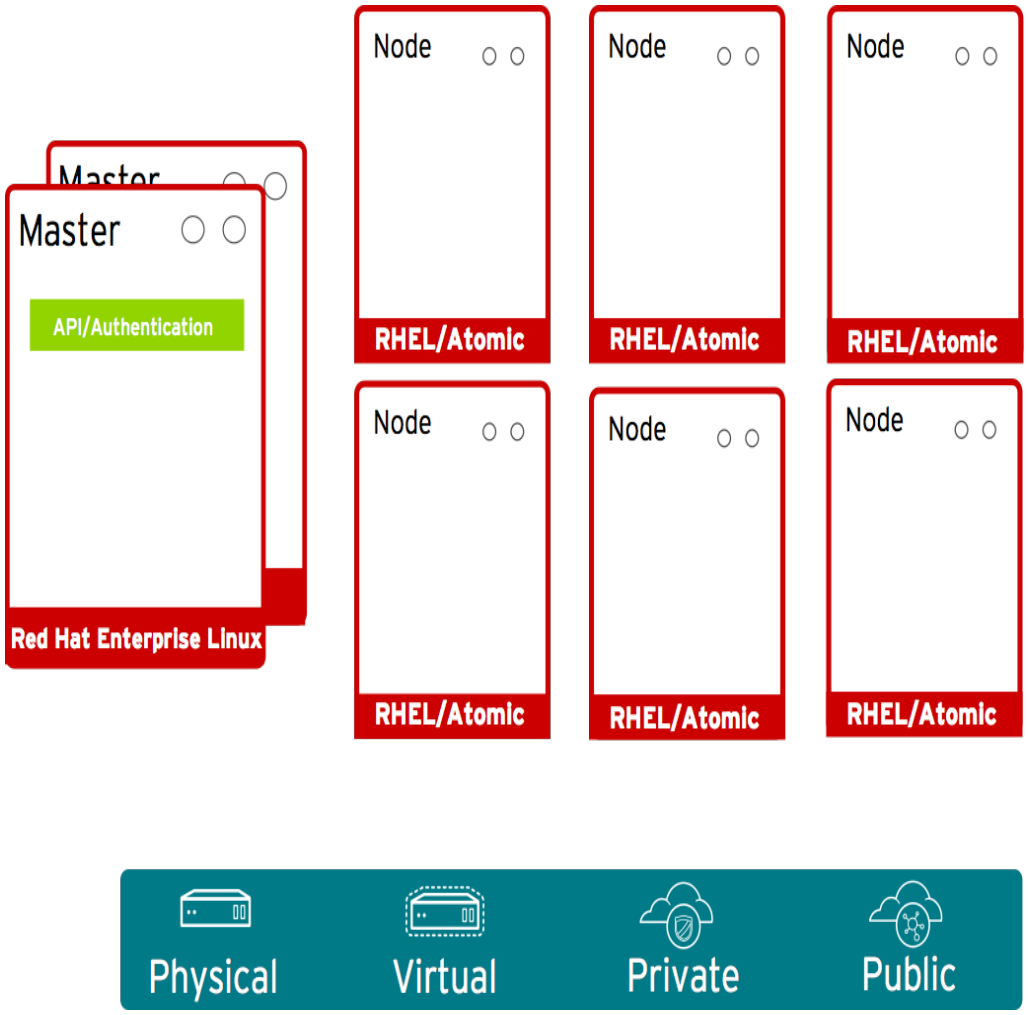
Pods Runs One or More Docker Containers As a Unit



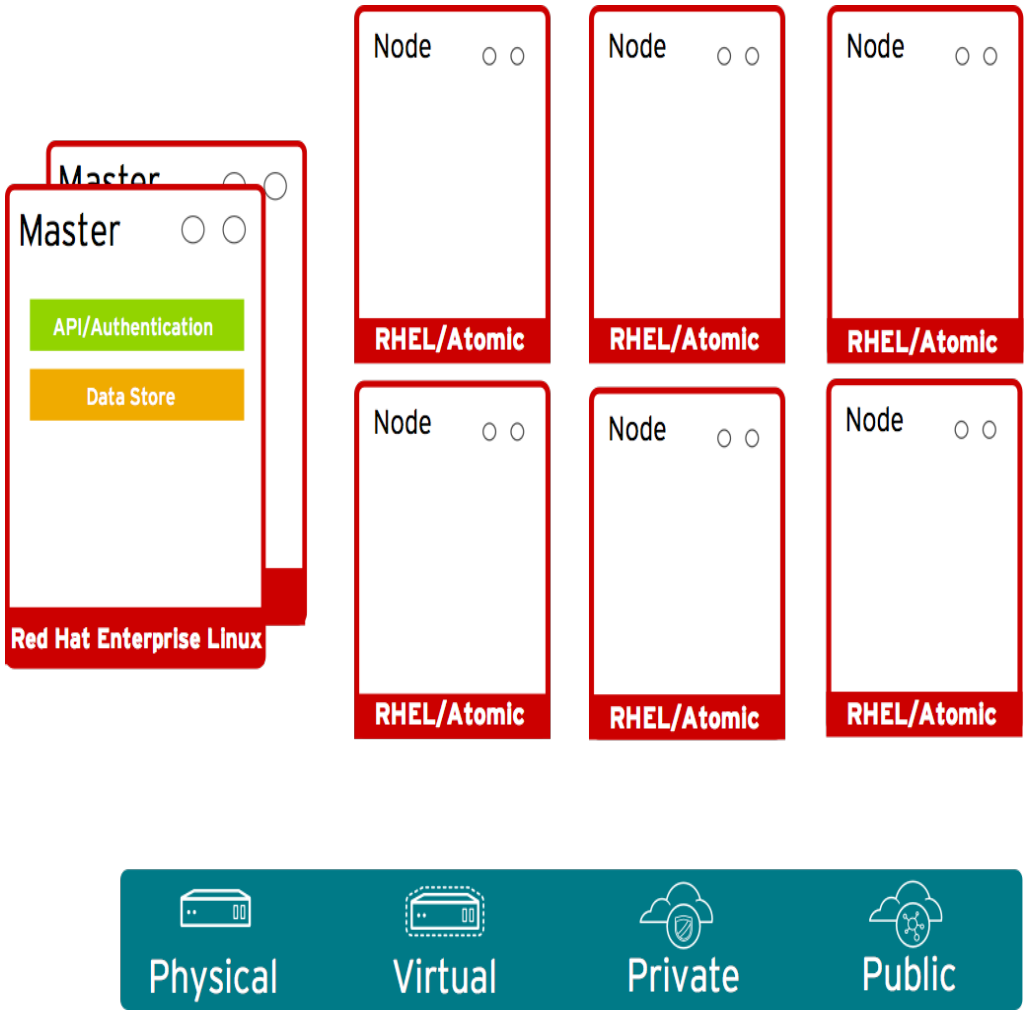
Masters Leverage Kubernetes to Orchestrate Nodes / Apps



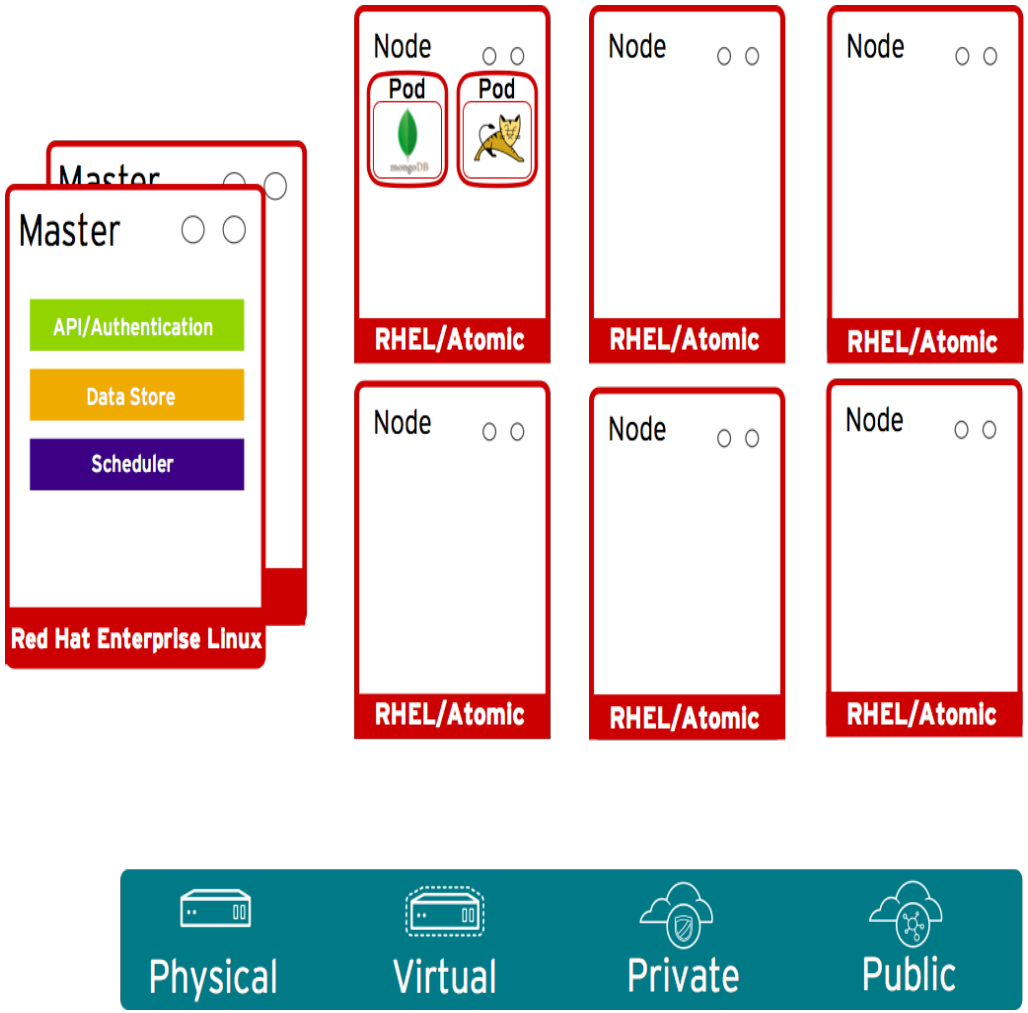
Master Provides Authenticated API for Users & Clients



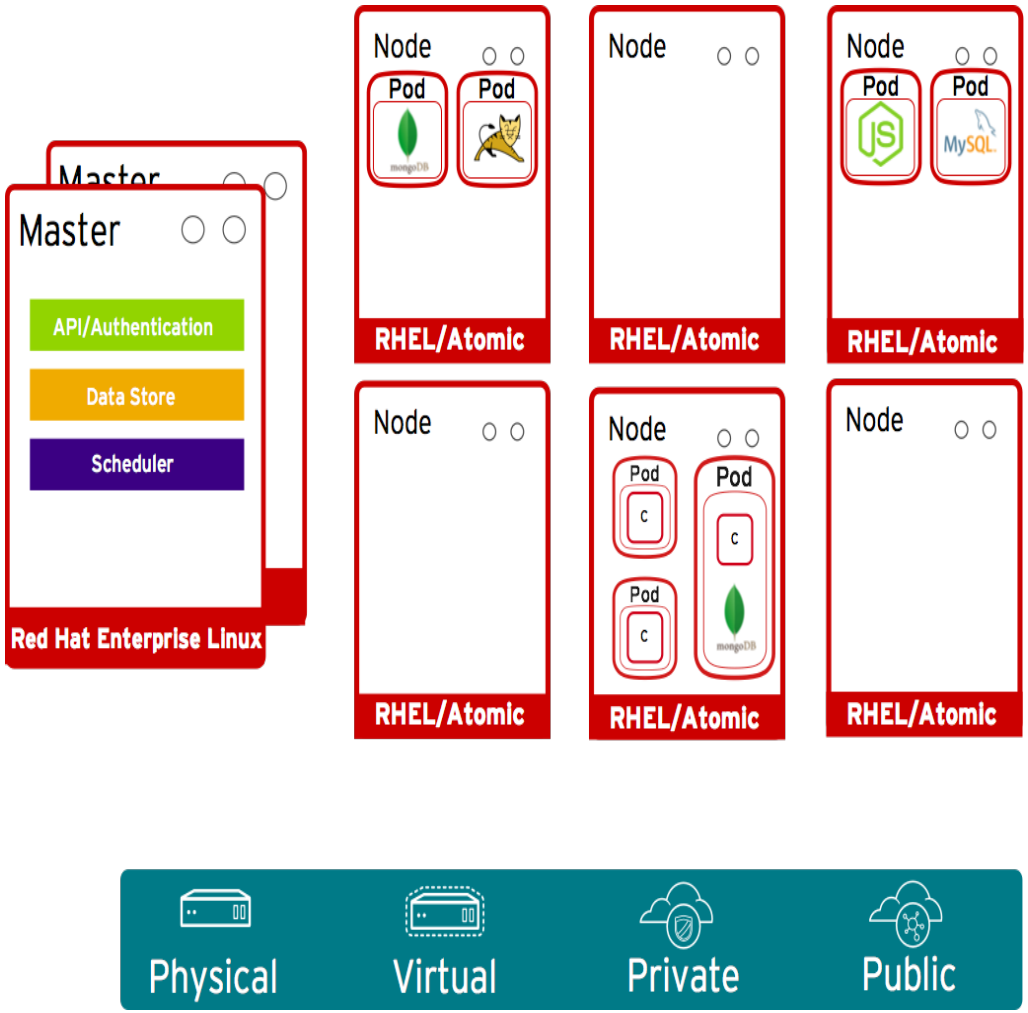
Master Uses etcd Key-Value Data Store for Persistence



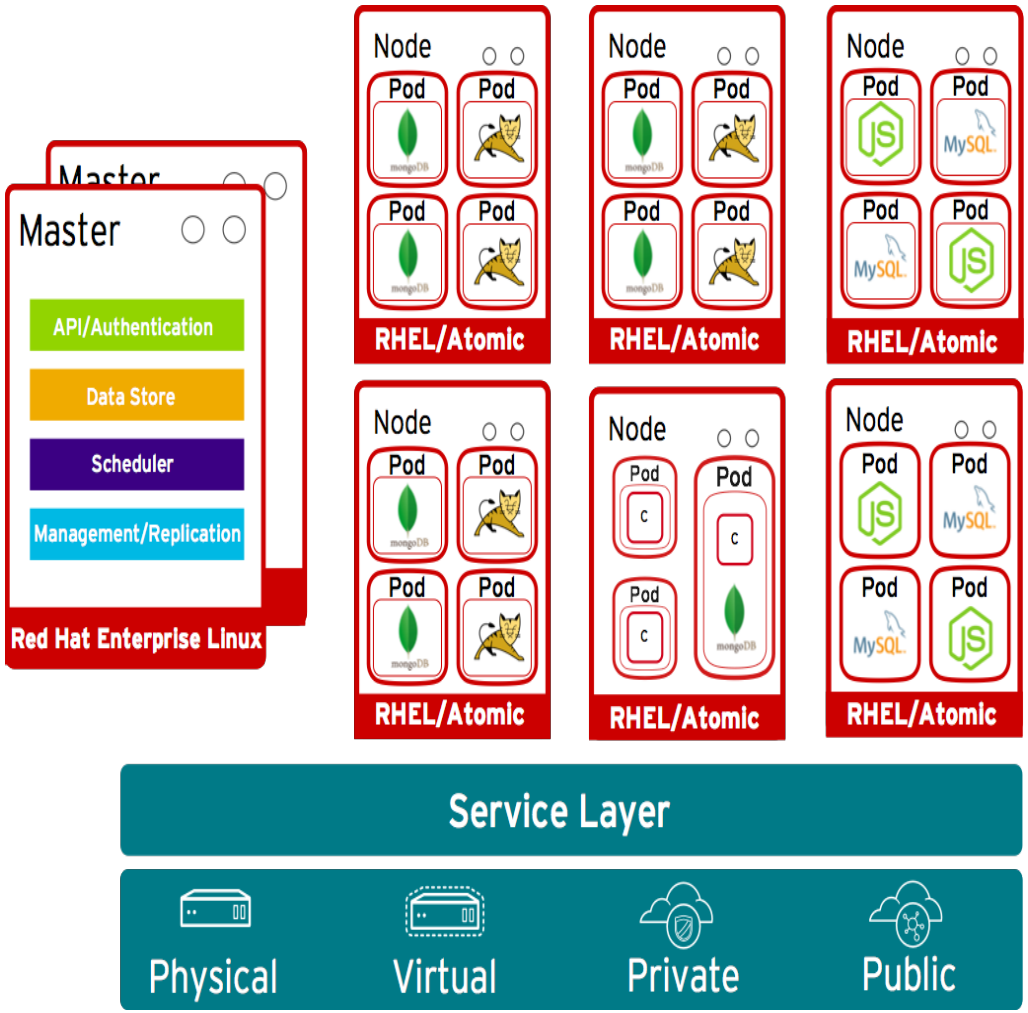
Master Provides Scheduler for Pod Placement on Nodes



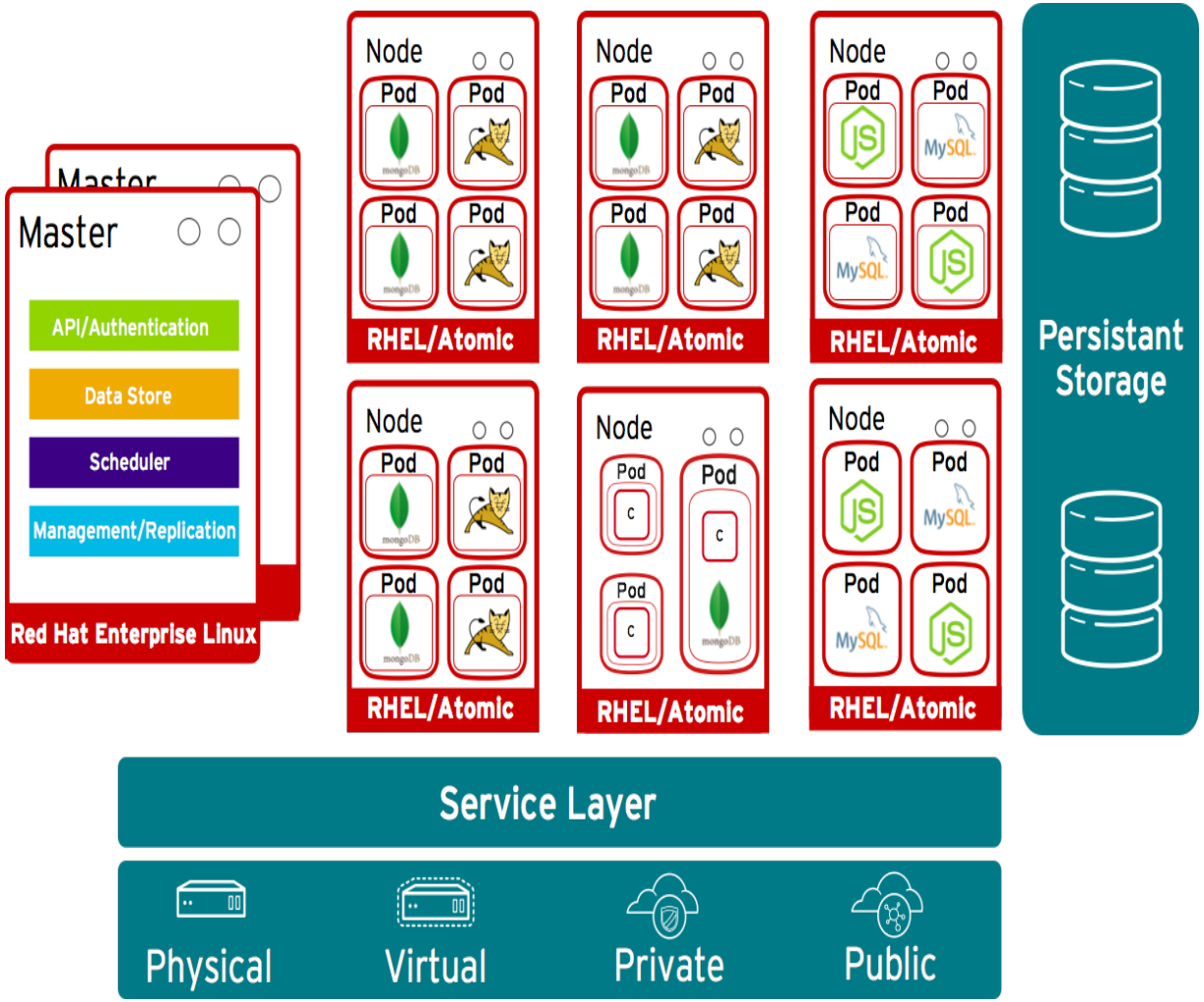
Pod Placement Is Determined Based on Defined Policy



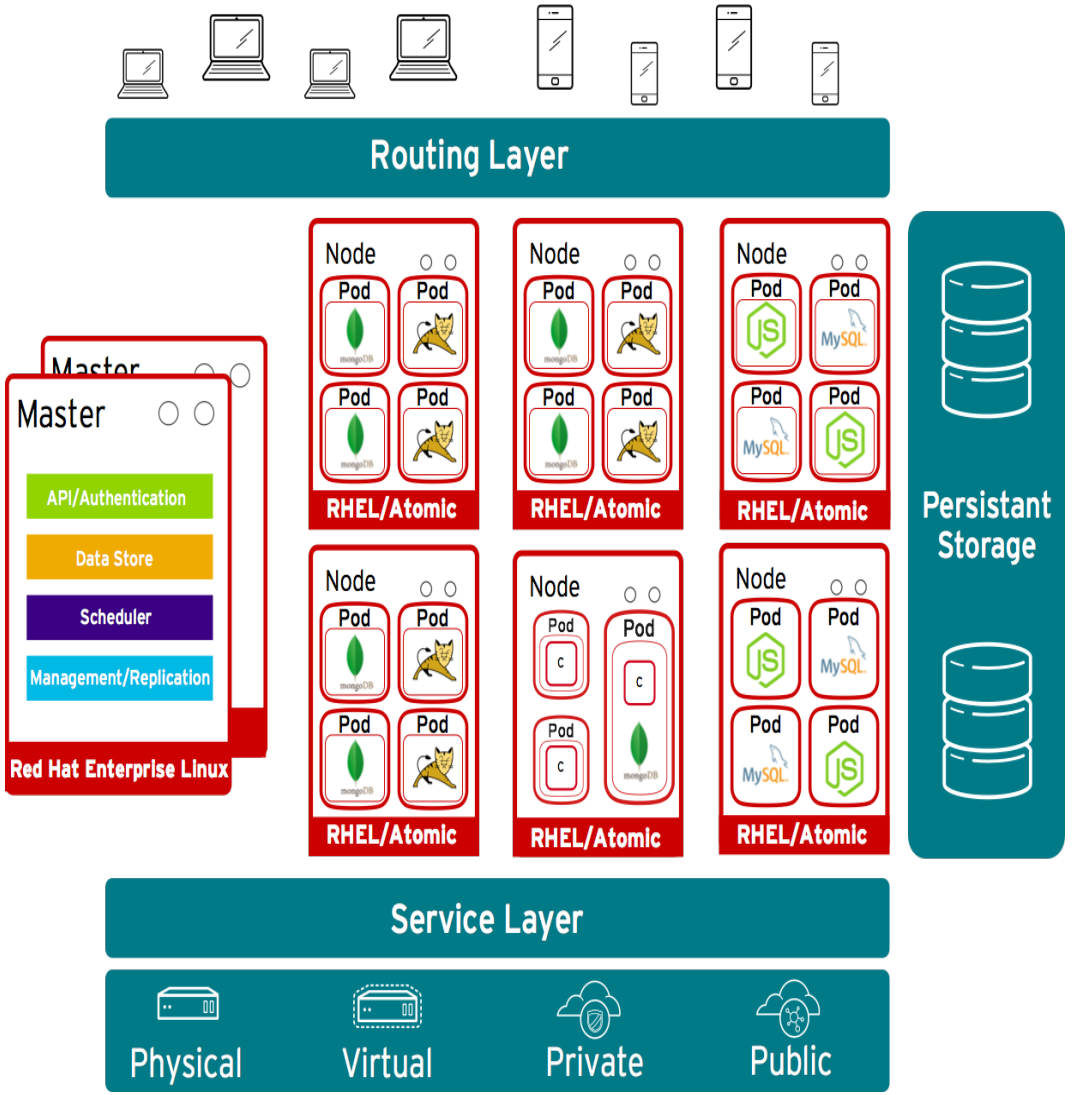
Services Allow Related Pods To Connect To Each Other



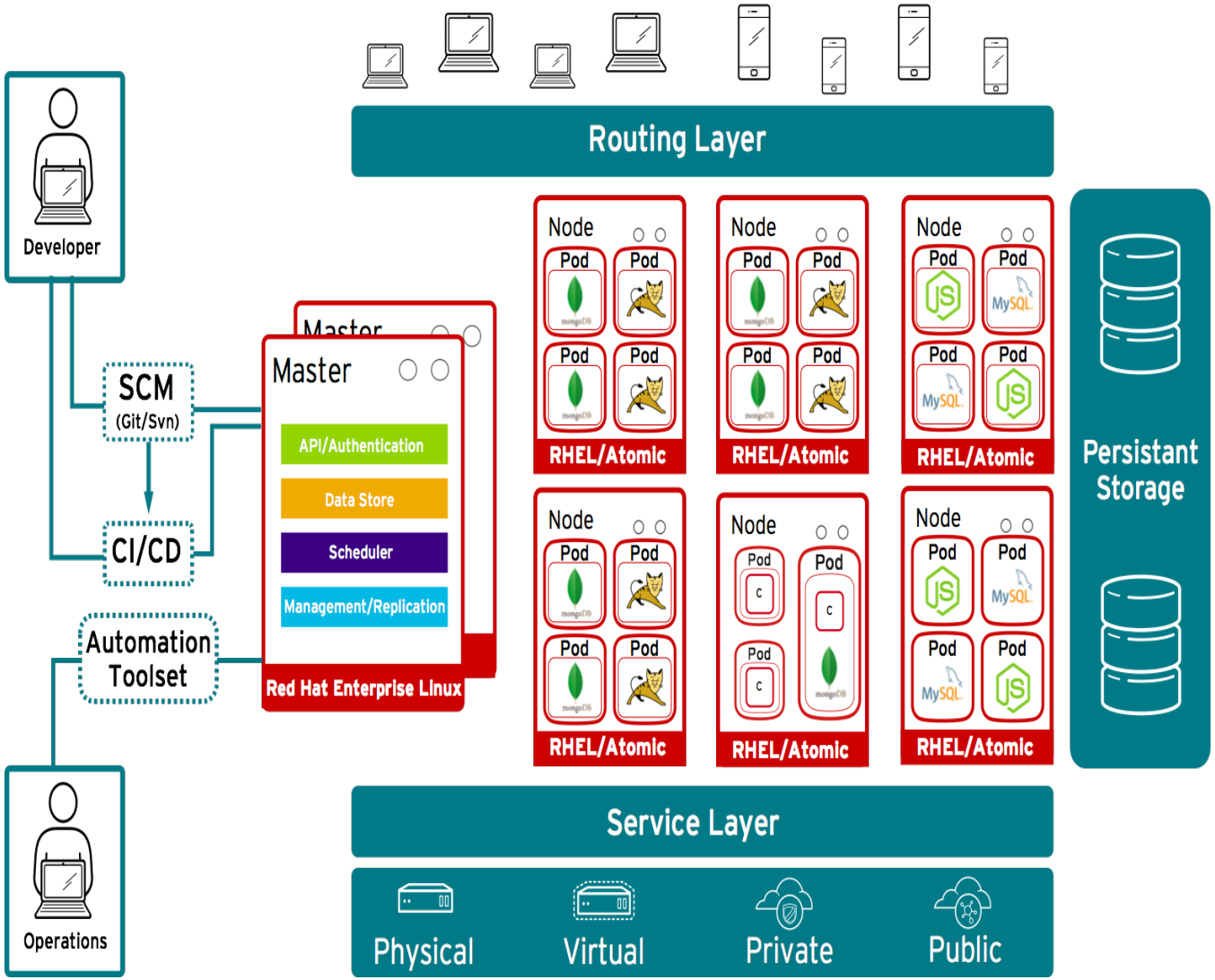
Pods Can Attach to Shared Storage for Stateful Services

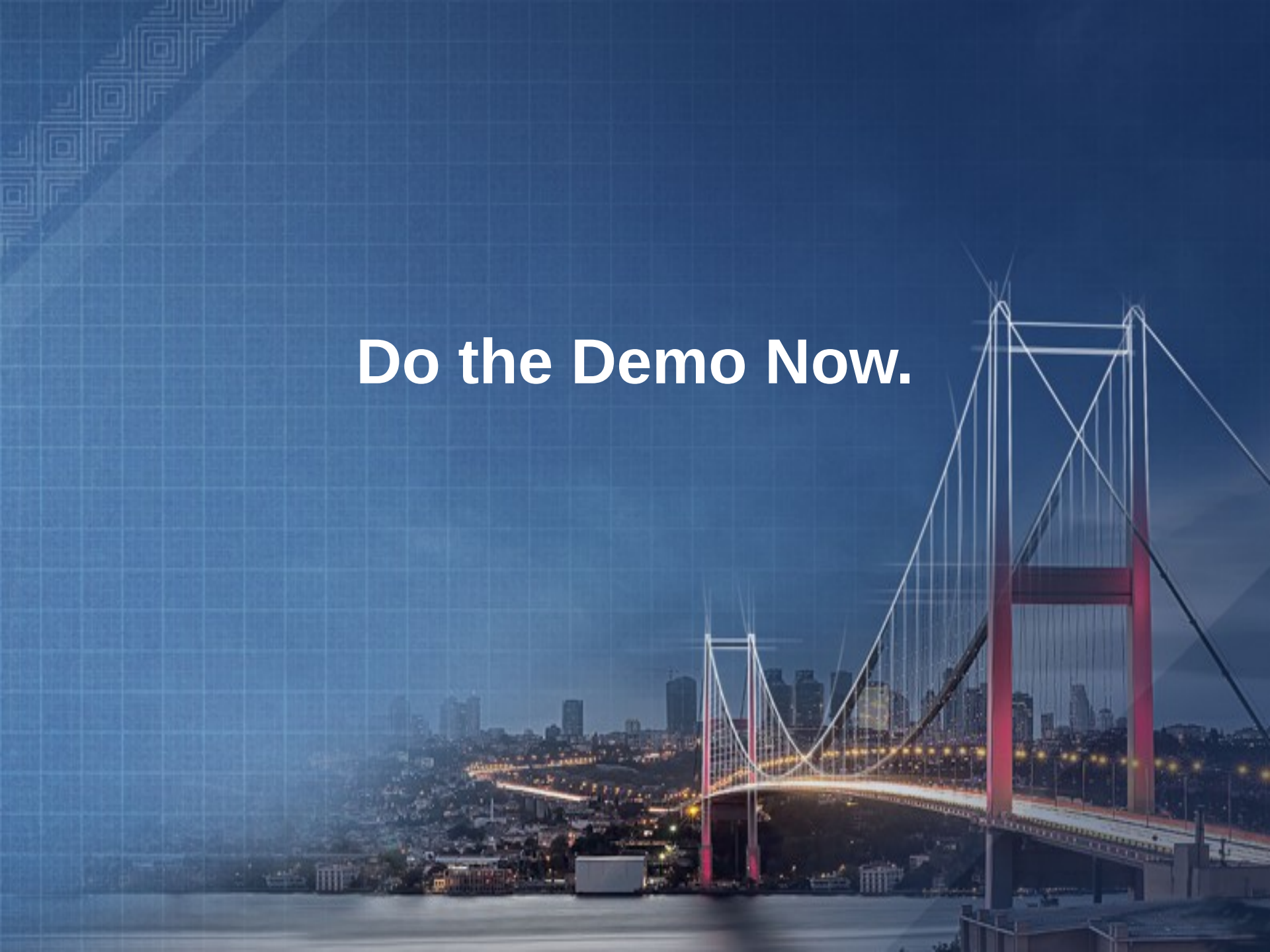


Routing Layer Routes External App Requests to Pods



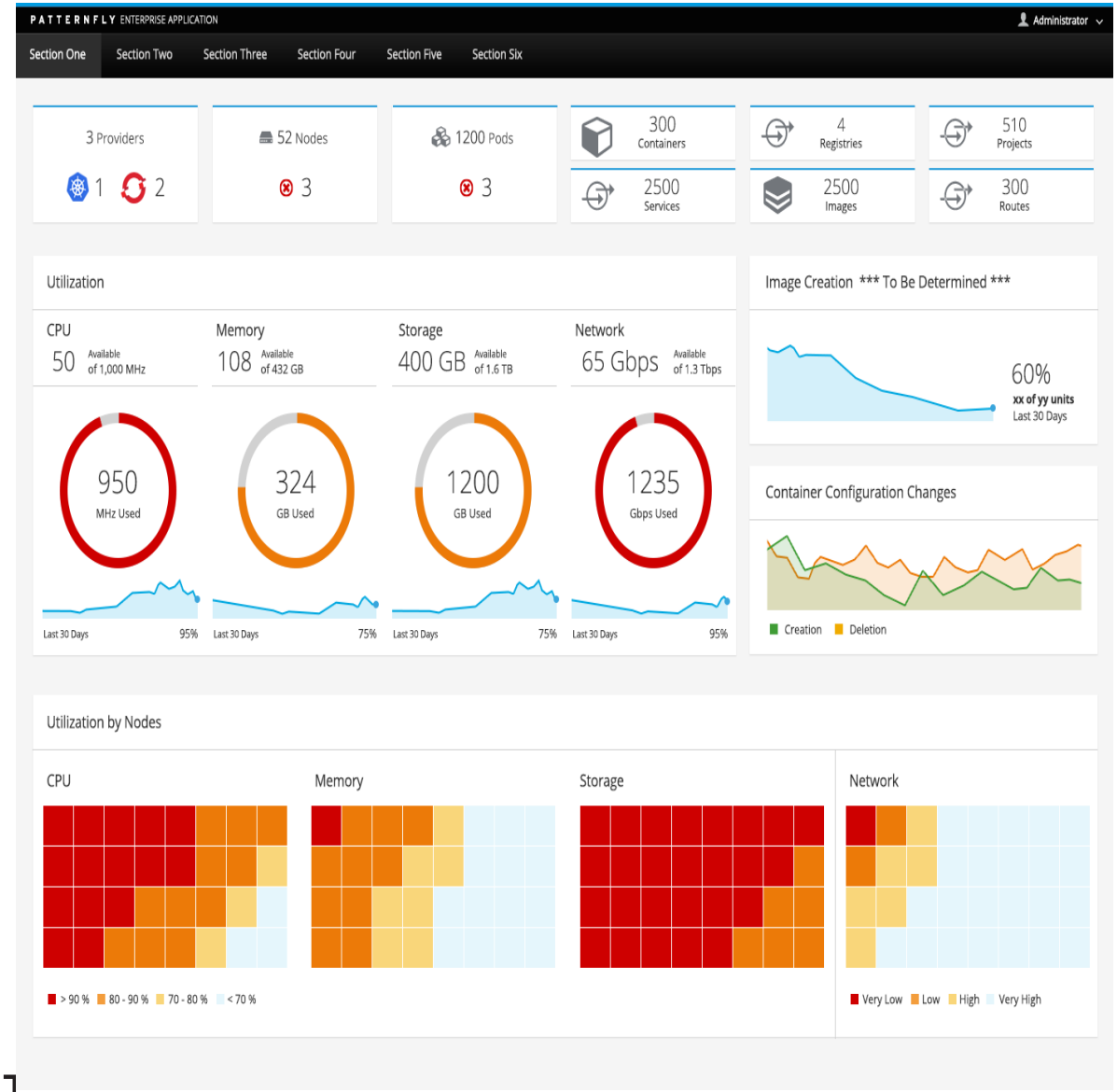
Developers Access OpenShift Via Web, CLI or IDE



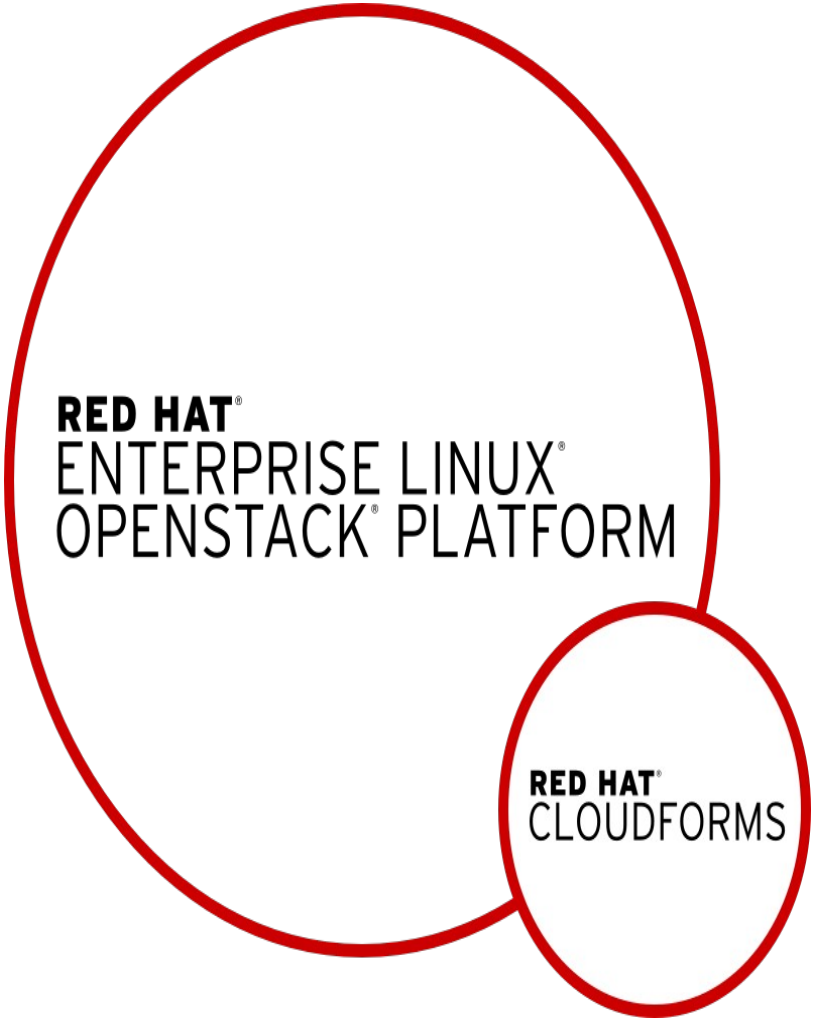

Do the Demo Now.

What’s Next & Roadmap

Administration & Container Management with CloudForms



Expanded OpenStack Integration

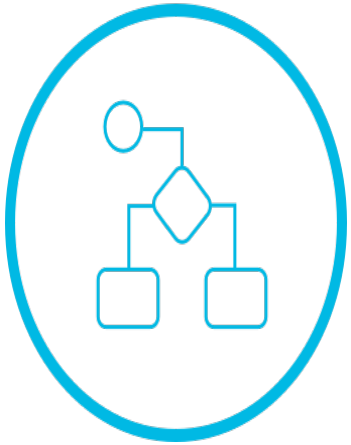


- Automating deployment of OpenShift clusters, add & remove Kubernetes Nodes
- Networking provider integration with Neutron
- Storage integration with OpenStack Cinder (Block) and Manila (File)
- Manage OpenStack and OpenShift with CloudForms

Integration (Fuse), Mobile (FeedHenry) & Decision/Rules (BRMS) Services

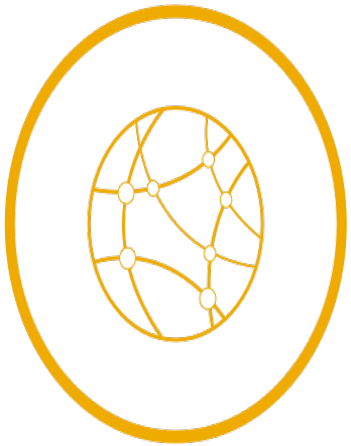
Application Container Services

- JBoss Enterprise Application Platform
- JBoss Web Server / Tomcat
- JBoss Developer Studio



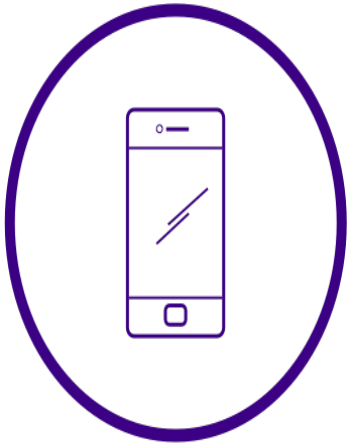
Integration Services

- Fuse *
- A-MQ
- Data Virtualization *



Business Process Services

- Business Process Management *
- Business Rules Management System *



Mobile Services

- Red Hat Mobile / FeedHenry *

* Coming Soon

OpenShift Online & OpenShift Dedicated

- OpenShift 3 Public Cloud services managed by Red Hat
  - OpenShift Dedicated - dedicated clusters (Nodes & Masters) for each customer
  - Deploy and run your applications on your own dedicated infrastructure Integrate with on-premise services
  - Available now in Tech Preview
- OpenShift Online will also be migrating to v3 (1HCY2016)

OpenShift 3 Tentative Release Schedule

3.0 - June 2015

- Docker container runtime & image packaging format
- Kubernetes orchestration & mgt.
- Source-to-Image & Docker builds
- JBoss EAP 6.4, JWS 3.0, A-MQ 6.2
- SCL images (Node, Python, PHP, Ruby...)
- Shared storage volumes for stateful apps
- Projects & team collaboration
- OAuth & enterprise auth integration (LDAP)
- Enhanced Web, CLI and IDE interfaces
- Manual scaling

3.0.x - Q3CY2015

- Auto-scaling basic
- F5 and External Routing Examples
- Reference architectures
- Bug fixes

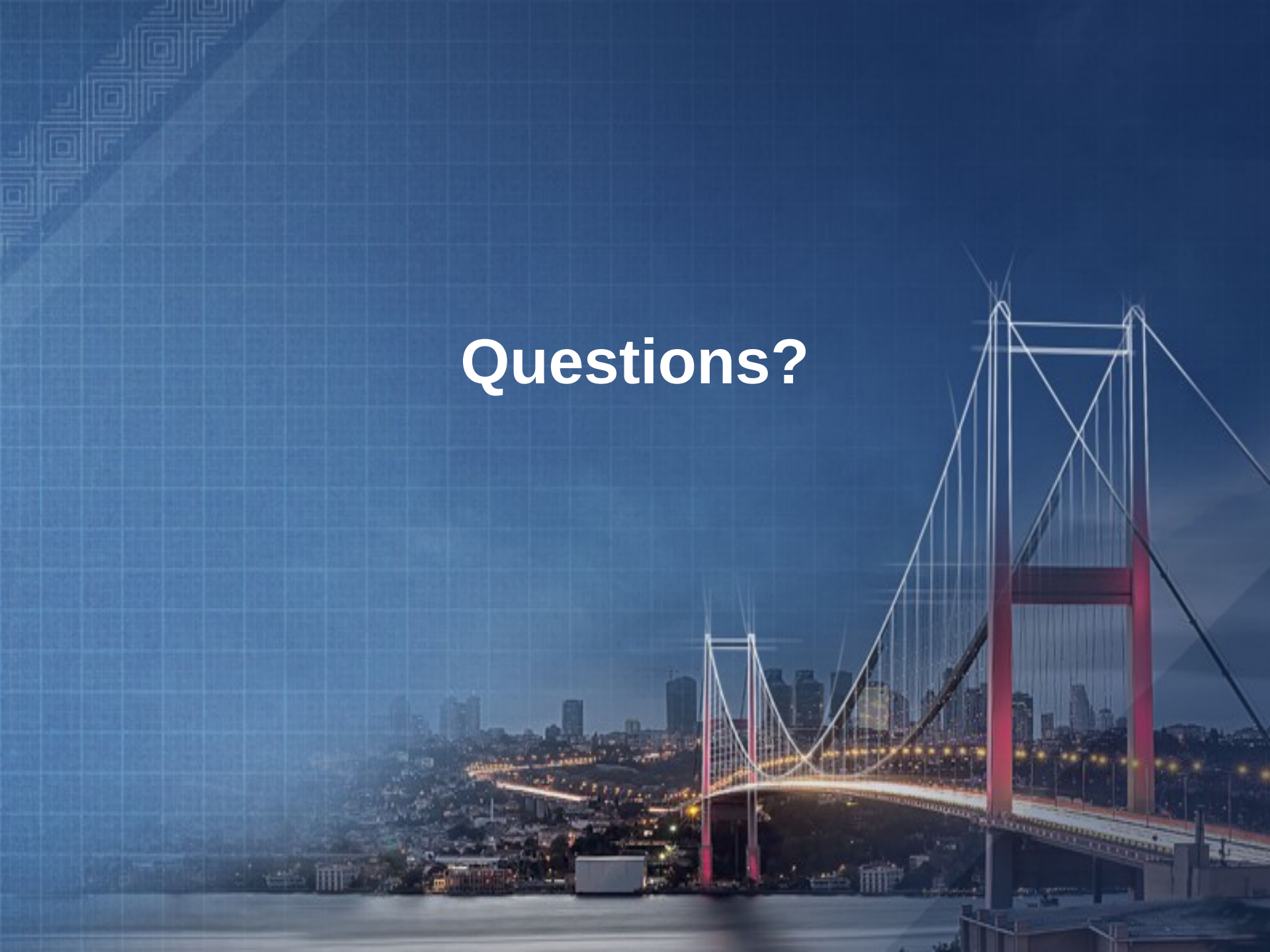
3.1 - Q4CY15

- Metric-driven autoscaling
- Integration Service / Fuse 6.x
- Mobile Service / FeedHenry
- Decision Service / BRMS (?)
- Cache Service / JDG (?)
- Eclipse IDE completion
- Web/CLI UX enhancements
- SCL 2 image updates
- External service bridge/registry (?)
- CloudForms Provider
- Additional storage plugins
- Networking enhancements
- Enhanced logging / ELK (?)
- CPU/Memory Overcommit
- Better Master HA
- Job Controller
- LDAP teams integration enhancements
- Better CI integration

3.2 - 1HCY16

- Idling (?)
- Expanded Fabric8 Integration
- Non-SNI / non-HTTP routing
- Git hosting
- OpenStack Network & Storage Integrations
- CloudForms Active Management
- More TBD

OPENSHIFT by Red Hat


Questions?