redhat.

OPENSHIFT CONTAINER PLATFORM TECHNICAL OVERVIEW

Giovanni Fontana
gfontana@redhat.com
Cloud Consultant
06/2018

OpenShift Workshop

Agenda

Day 1

OpenShift Introduction
9h - 12h

Network Concepts
13h - 15h

Commands & Troubleshooting
15h - 17h

Day 2

Security
9h - 12h

Persistent Storage
13h - 15h

Managing App Development
15h - 17h

Day 3

Metrics & Logging
9h - 11h

Quotas & Limits
11h - 15h

OpenShift Workshop

Agenda

Day 1

OpenShift Introduction
9h - 11h

Network Concepts
11h - 12h

Commands
14h - 16h

Controlling Access
16h - 71h

Day 2

Controlling Access
9h - 10h

Persistent Storage
10h - 12h

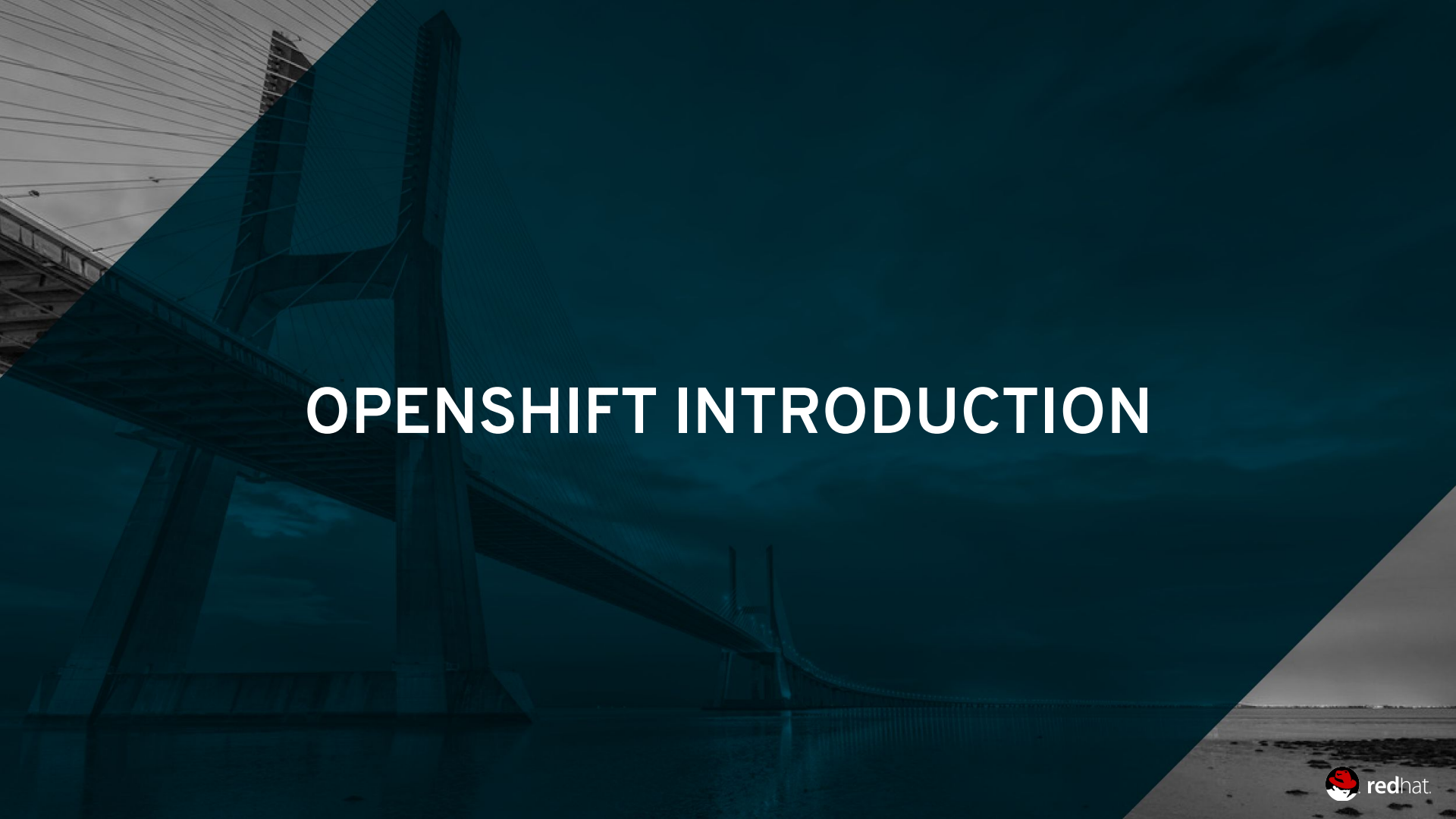
Managing App Development
14h - 16h

Metrics & Logging
Day 2 - 16h - 17h

Day 3

Metrics & Logging
9h - 11h

Quotas & Limits
11h - 15h


OPENSHIFT INTRODUCTION
redhat.

OPENSHIFT CONTAINER PLATFORM

Self-Service

Multi-language

Automation

Collaboration

Multi-tenant

Standards-based

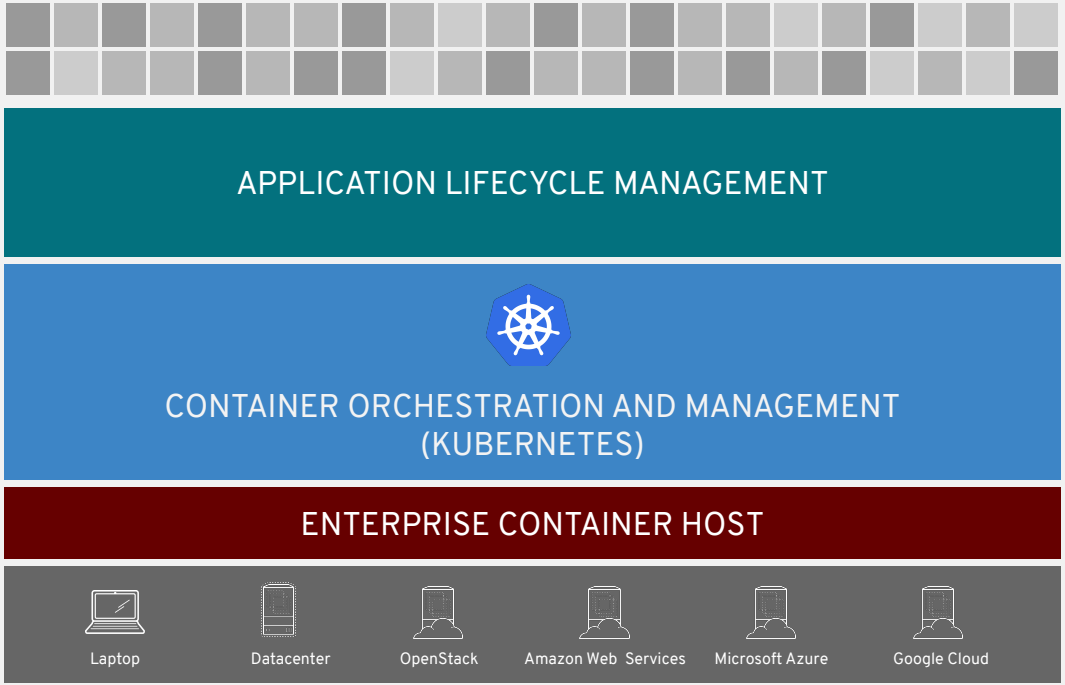
Web-scale

Open Source

Enterprise Grade

Secure

OPENSHIFT CONTAINER PLATFORM



ANY
CONTAINER

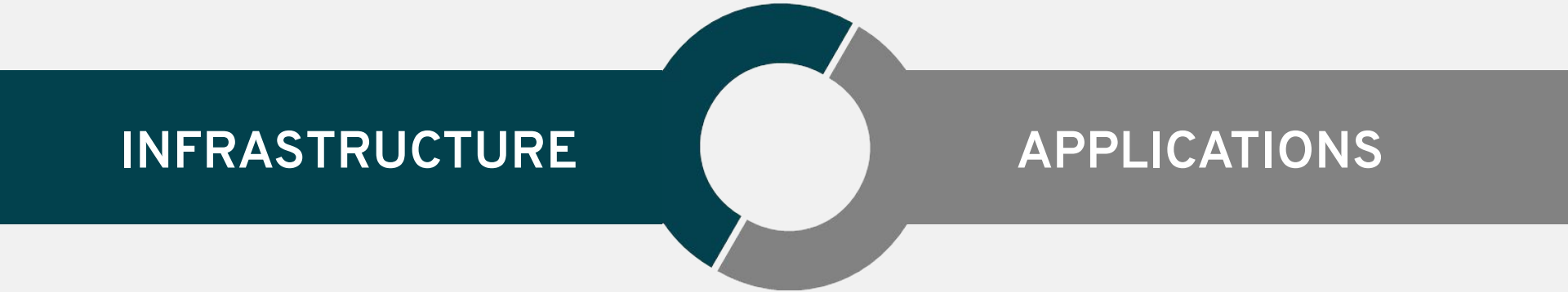
RED HAT®
OPENSHIFT

ANY
INFRASTRUCTURE

redhat.

WHAT ARE CONTAINERS?

It Depends Who You Ask

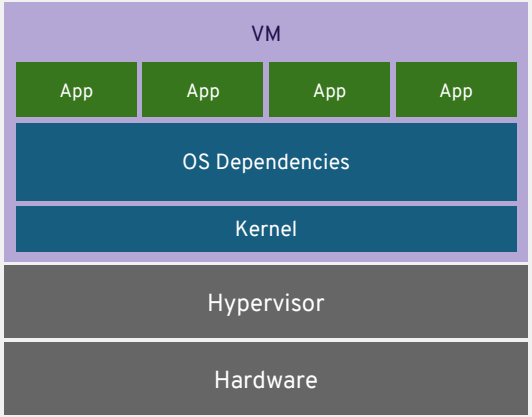


- Application processes on a shared kernel
- Simpler, lighter, and denser than VMs
- Portable across different environments

- Package apps with all dependencies
- Deploy to any environment in seconds
- Easily accessed and shared

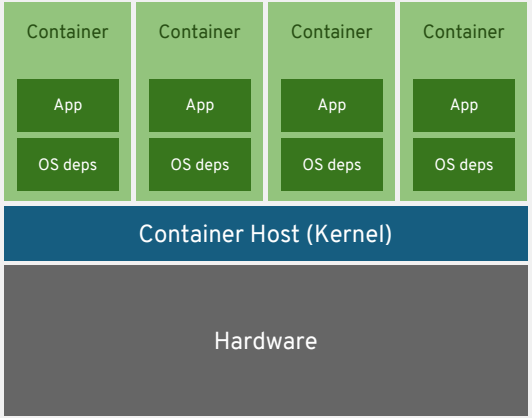
VIRTUAL MACHINES AND CONTAINERS

VIRTUAL MACHINES



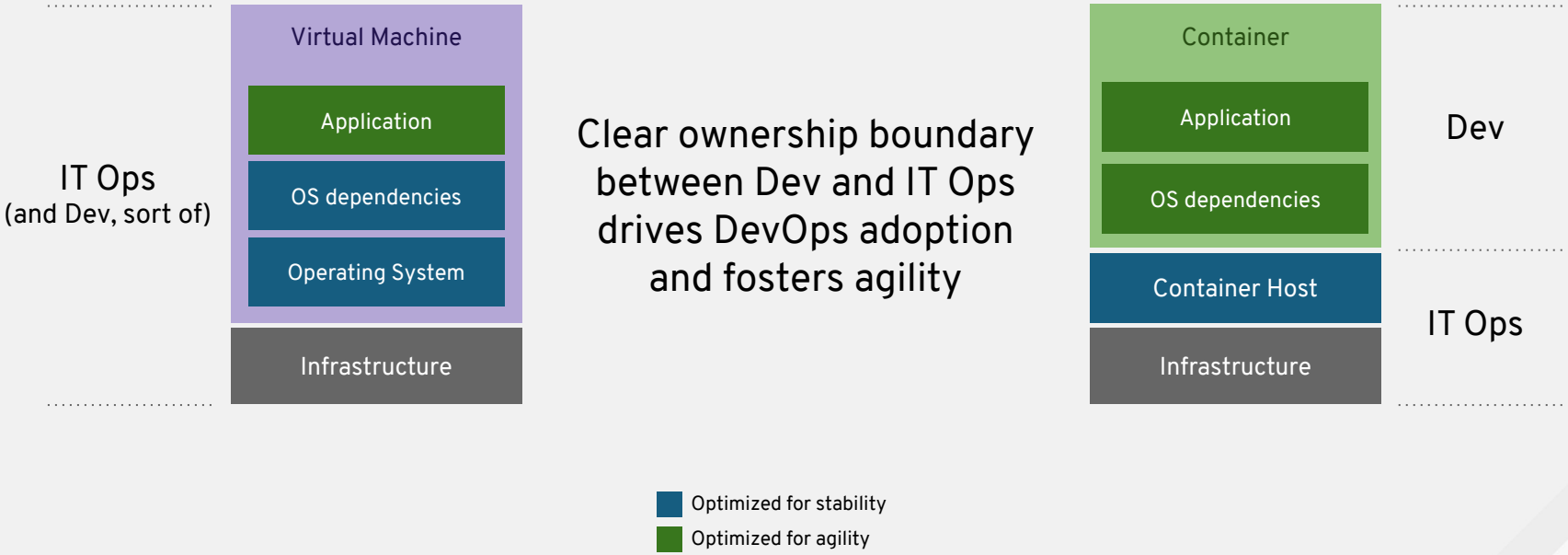
VM virtualizes the hardware

CONTAINERS



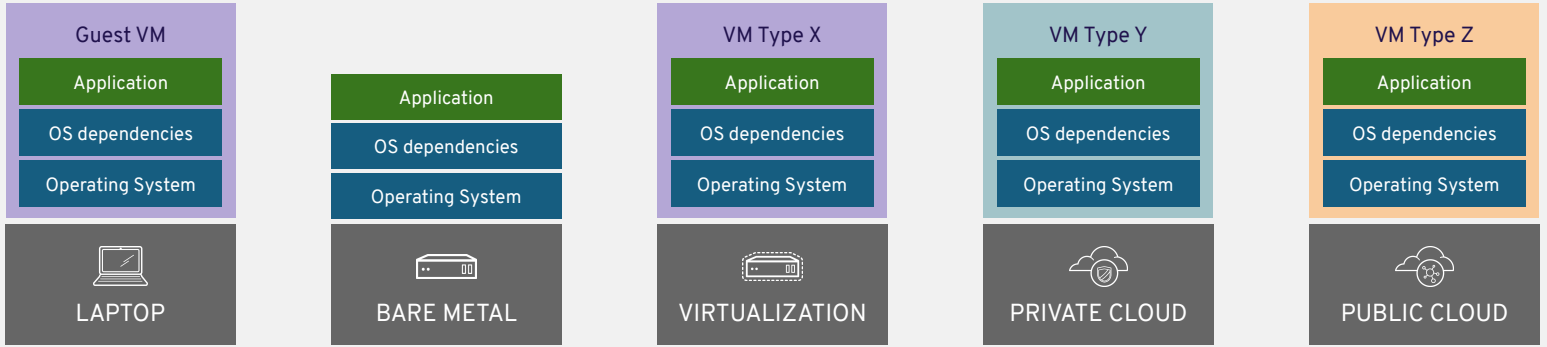
Container virtualizes the process

VIRTUAL MACHINES AND CONTAINERS



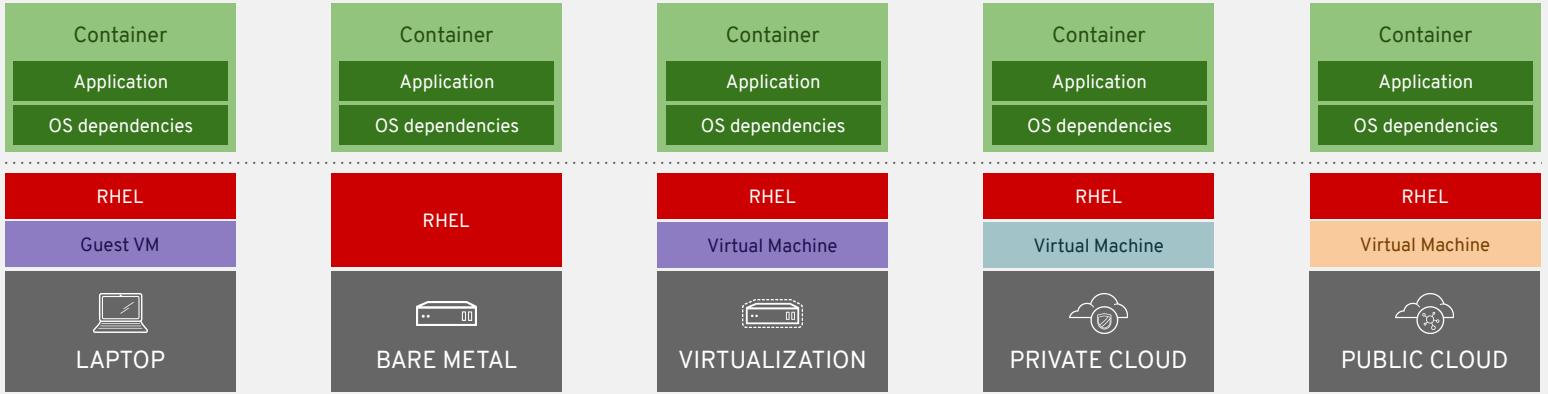
APPLICATION PORTABILITY WITH VM

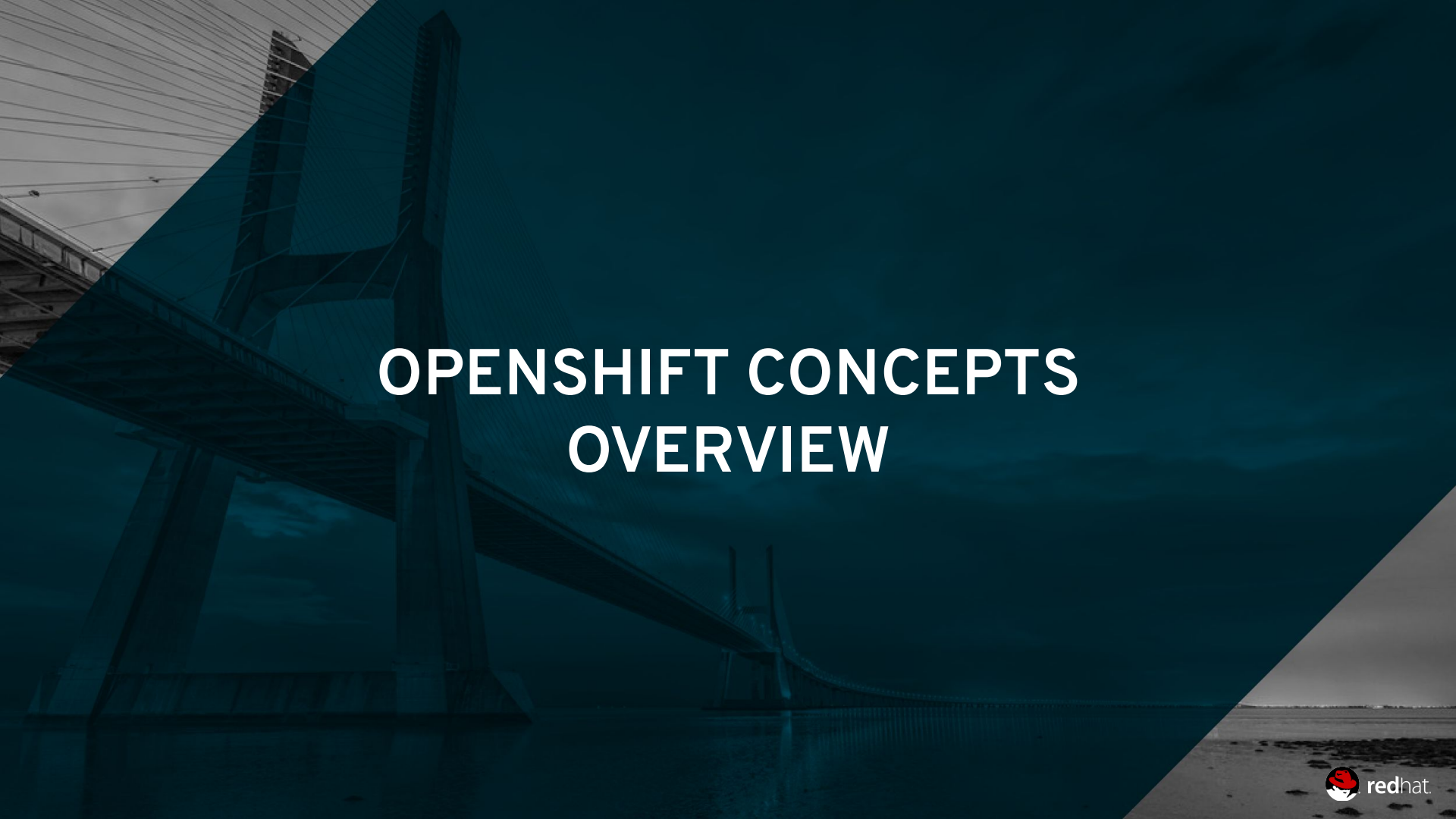
Virtual machines are NOT portable across hypervisor and do NOT provide portable packaging for applications



APPLICATION PORTABILITY WITH CONTAINERS

RHEL Containers + RHEL Host = Guaranteed Portability
Across Any Infrastructure

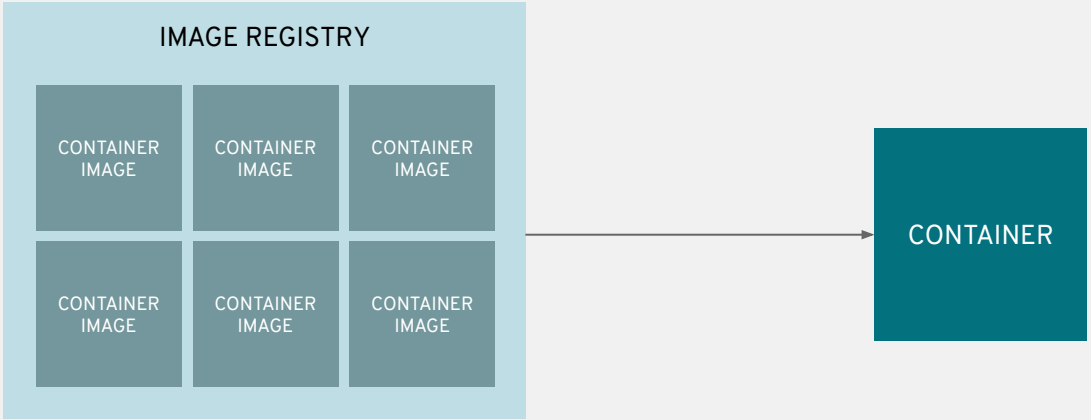



OPENSHIFT CONCEPTS
OVERVIEW
redhat.

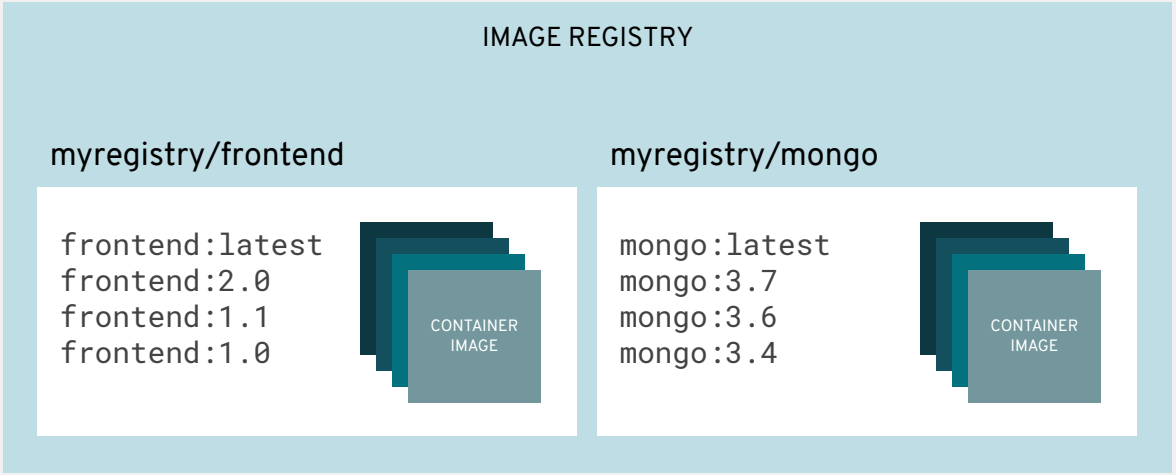
A container is the smallest compute unit

containers are created from container images

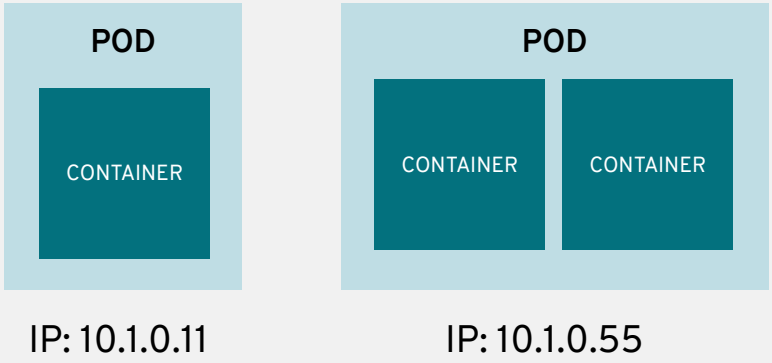
container images are stored in an image registry



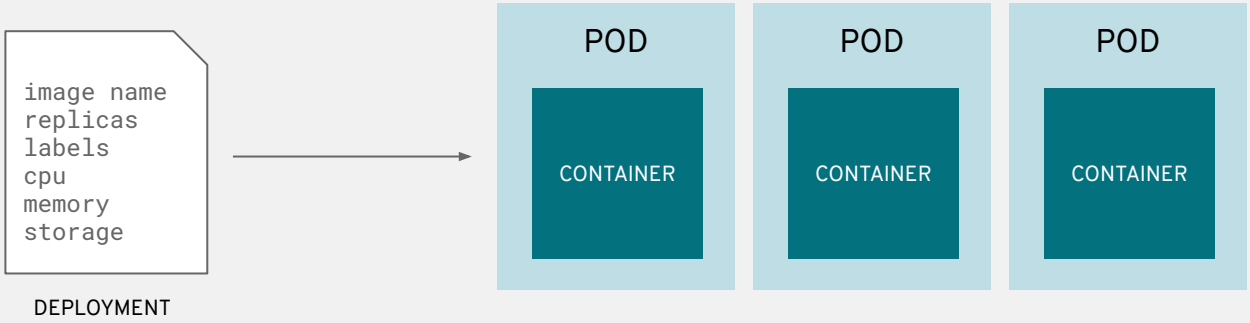
an image repository contains all versions of an image in the image registry



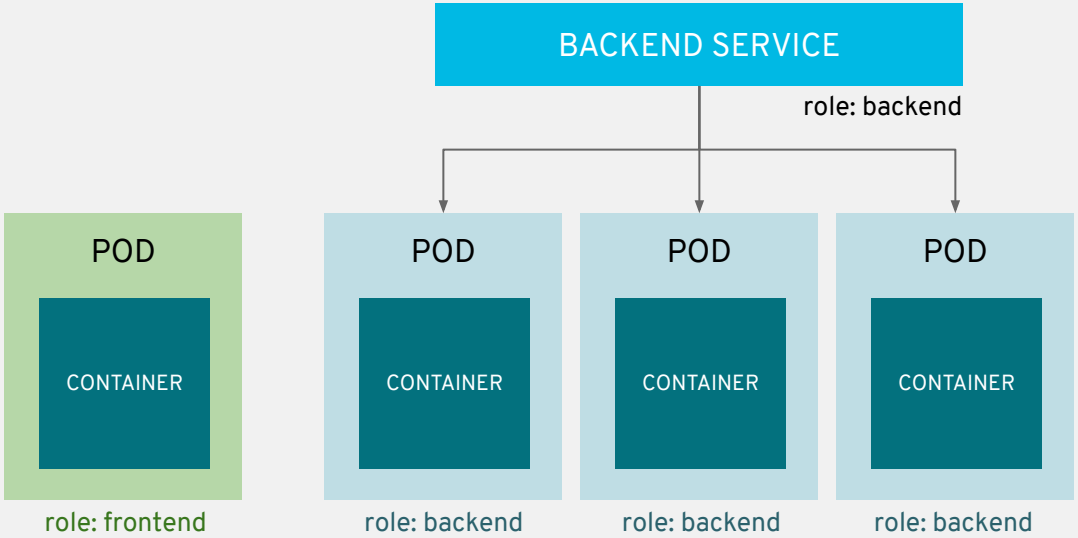
containers are wrapped in pods which are units of deployment and management



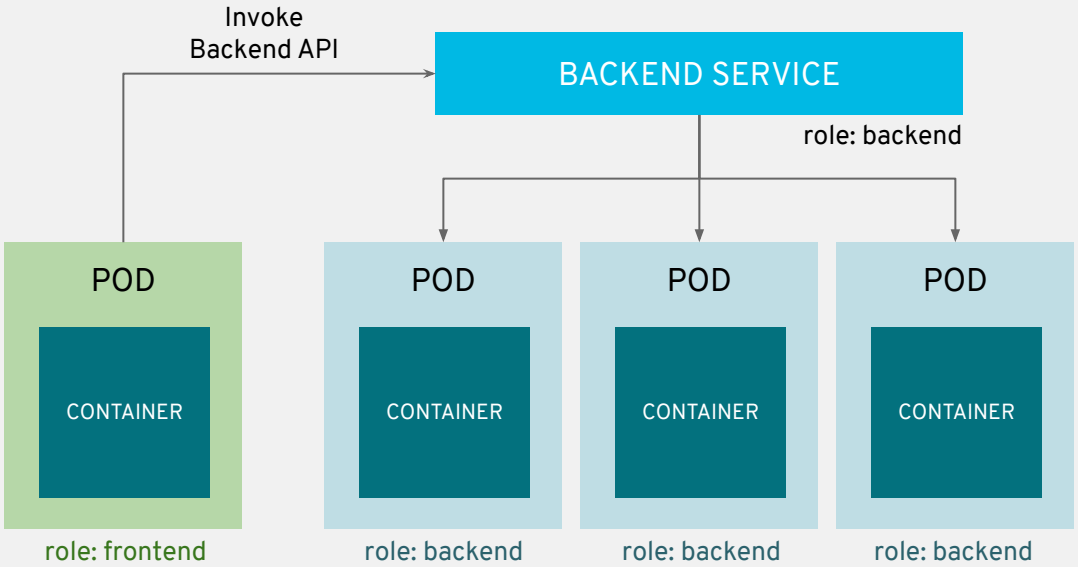
pods configuration is defined in a `deployment`



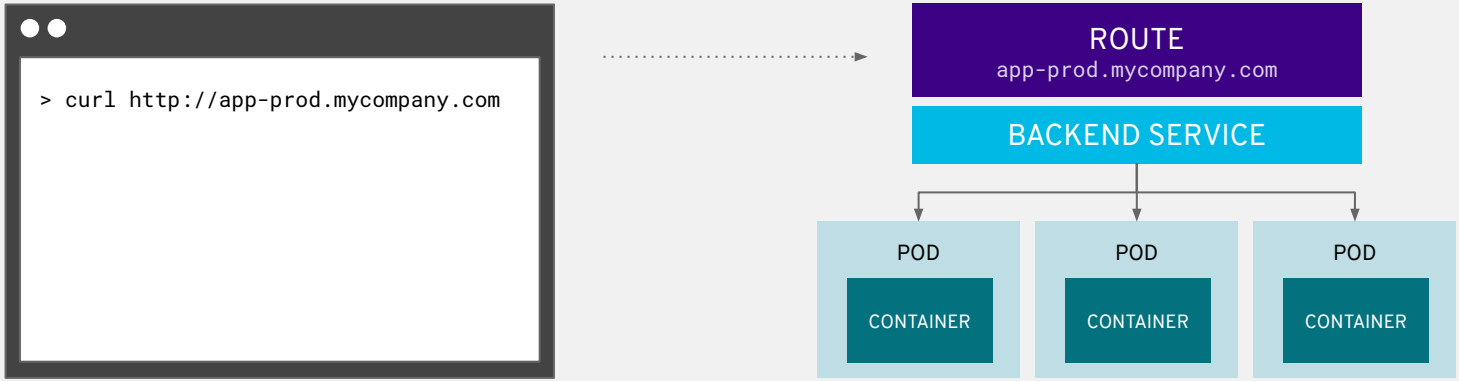
services provide internal load-balancing and service discovery across pods



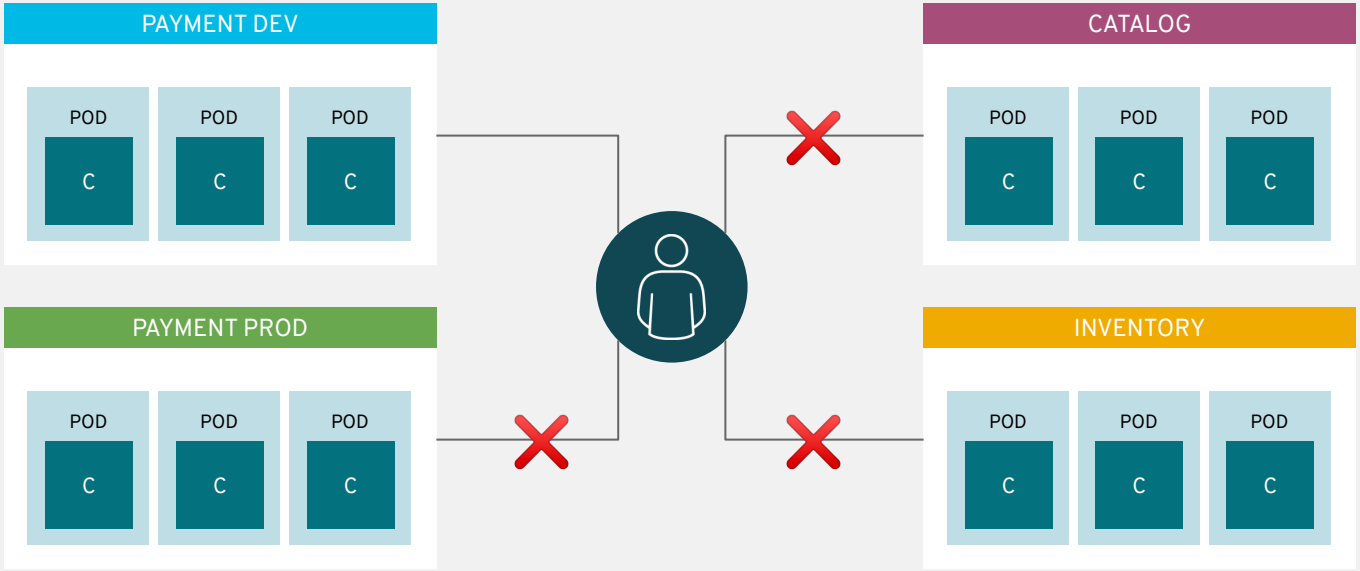
apps can talk to each other via services



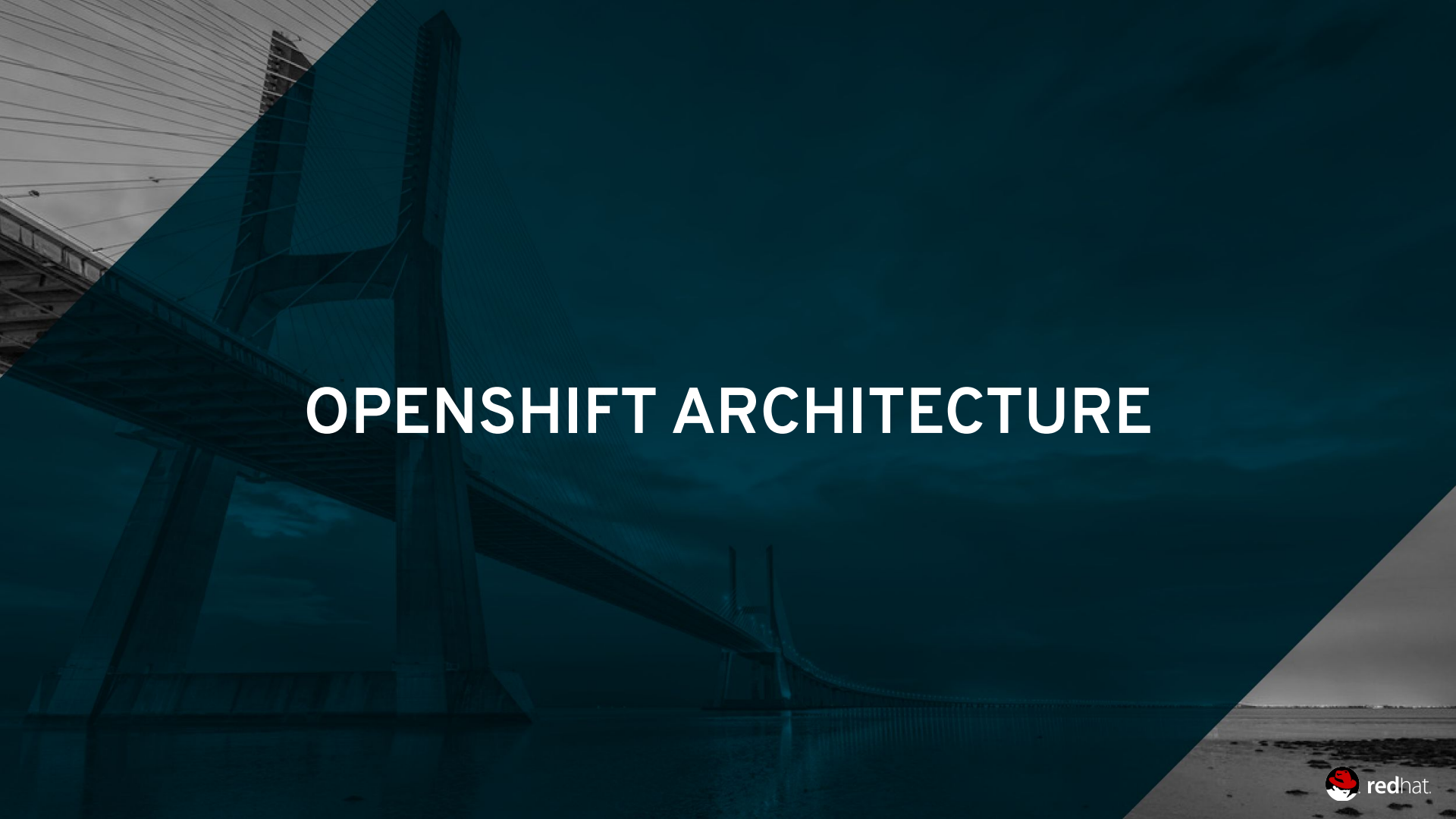
routes add services to the external load-balancer and provide readable urls for the app



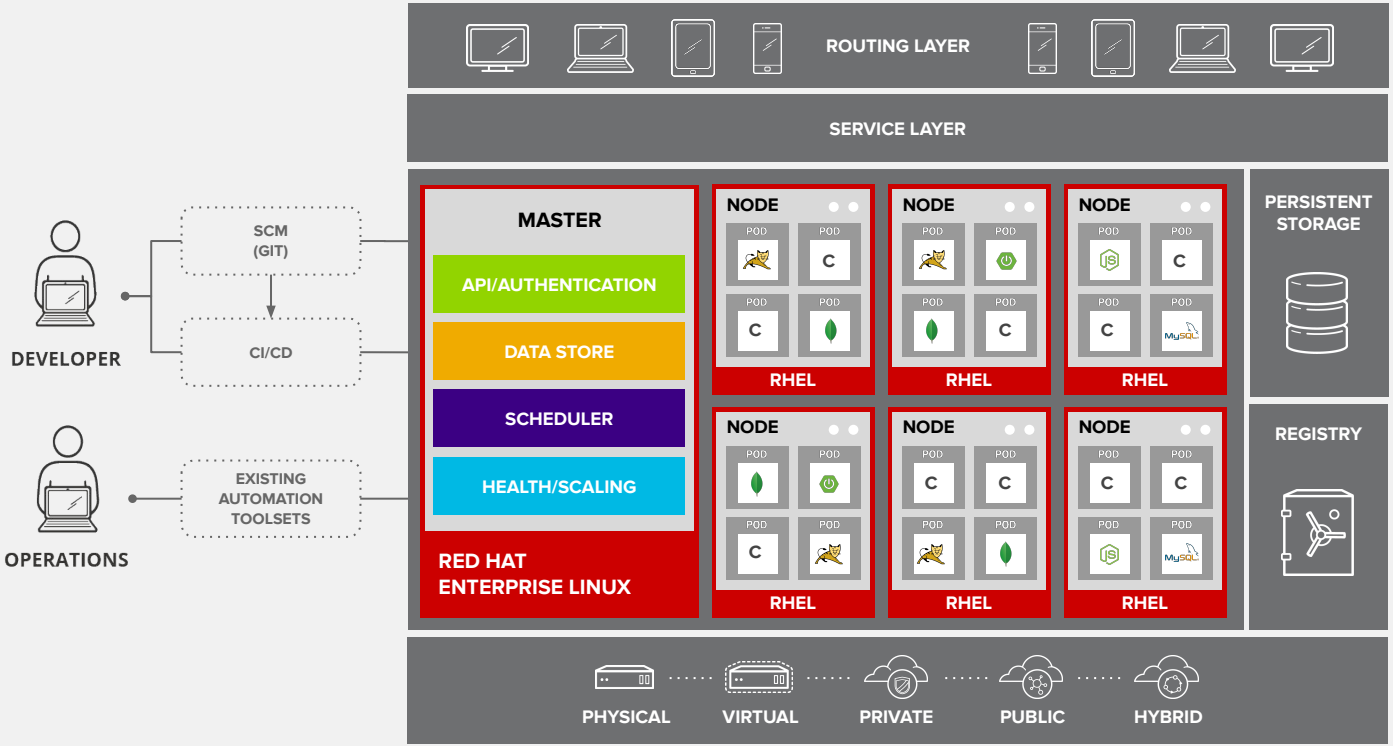
projects isolate apps across environments, teams, groups and departments



QUIZ

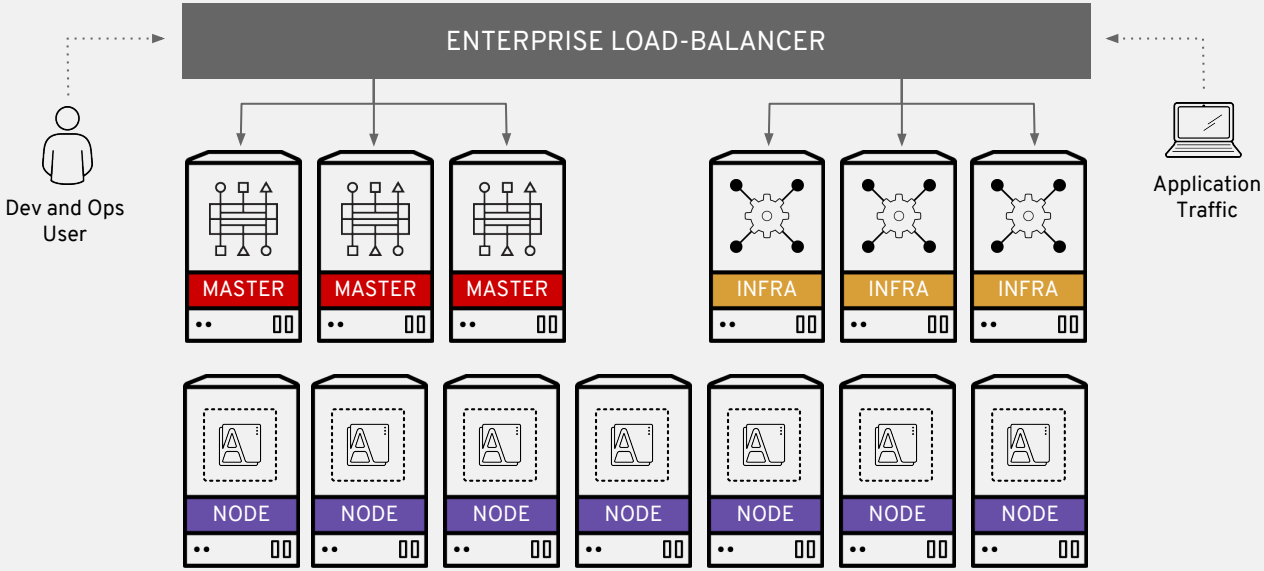

OPENSHIFT ARCHITECTURE
redhat.

ACCESS VIA WEB, CLI, IDE AND API



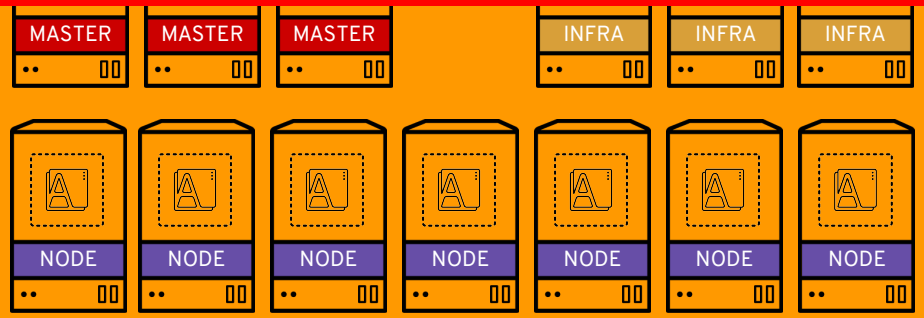
redhat.

FULL HIGH-AVAILABILITY ARCHITECTURE



FULL HIGH-AVAILABILITY ARCHITECTURE

AQUI SLIDE SOBRE O AMBIENTE BANESTES!



redhat.

OpenShift Workshop

Agenda

Day 1

OpenShift Introduction
9h - 12h

Network Concepts
13h - 15h

Commands & Troubleshooting
15h - 17h

Day 2

Security
9h - 12h

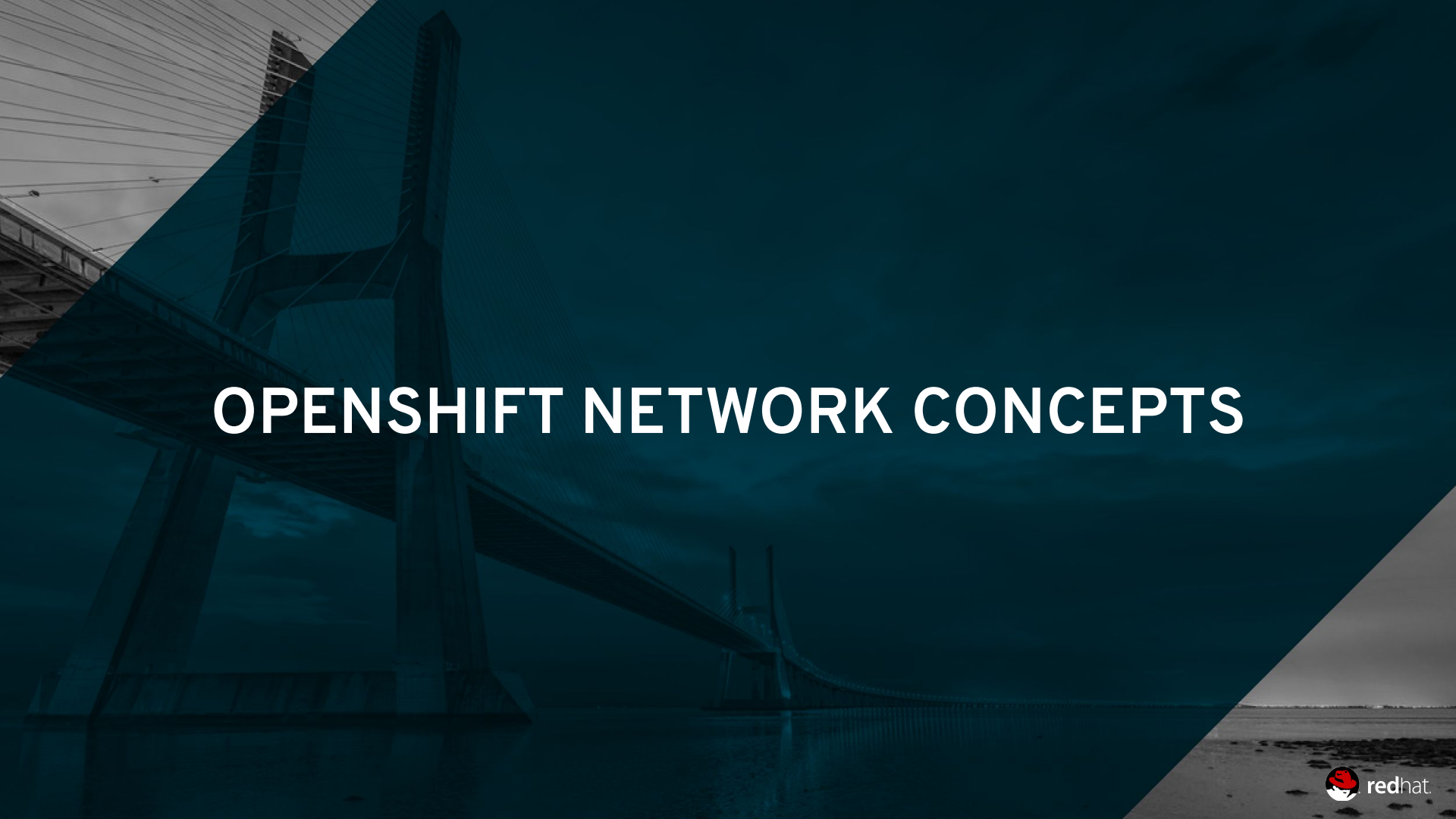
Persistent Storage
13h - 15h

Managing App Development
15h - 17h

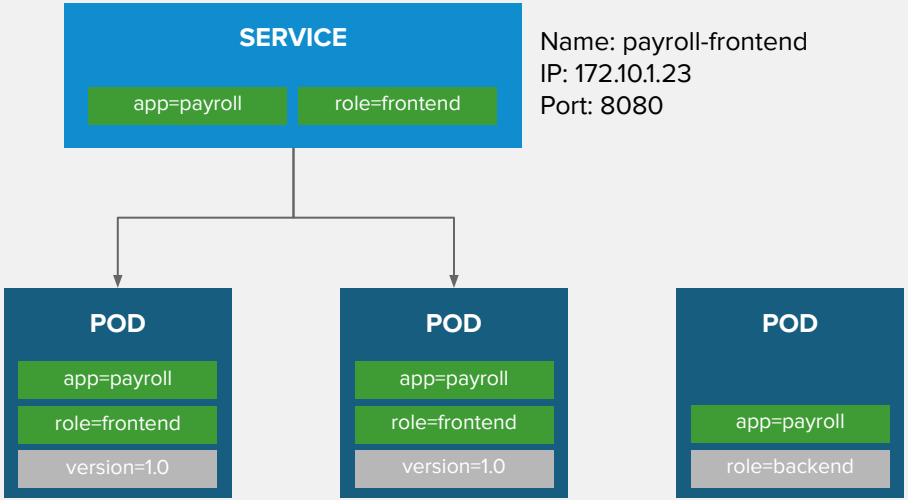
Day 3

Metrics & Logging
9h - 11h

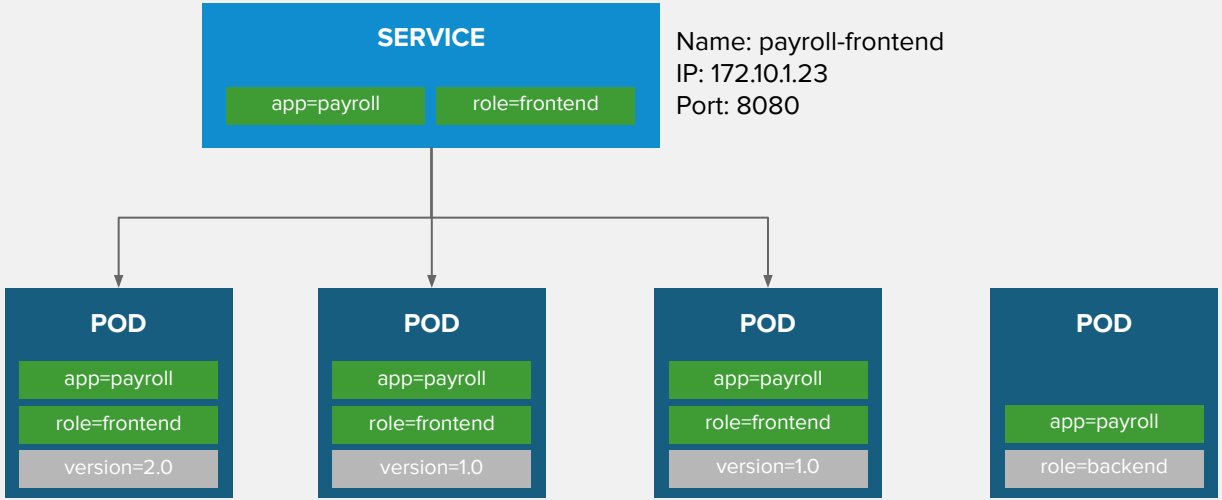
Quotas & Limits
11h - 15h


OPENSHIFT NETWORK CONCEPTS

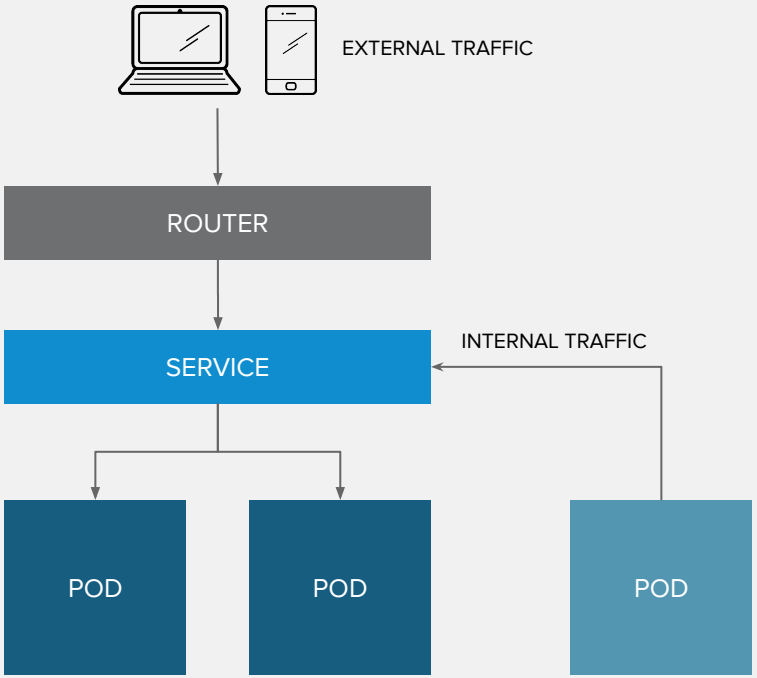
BUILT-IN SERVICE DISCOVERY
INTERNAL LOAD-BALANCING



BUILT-IN SERVICE DISCOVERY
INTERNAL LOAD-BALANCING

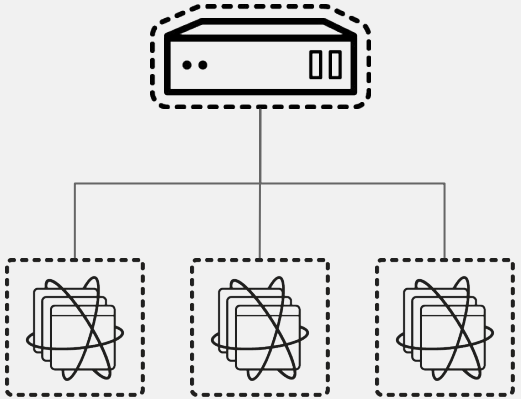


ROUTE EXPOSES SERVICES EXTERNALLY



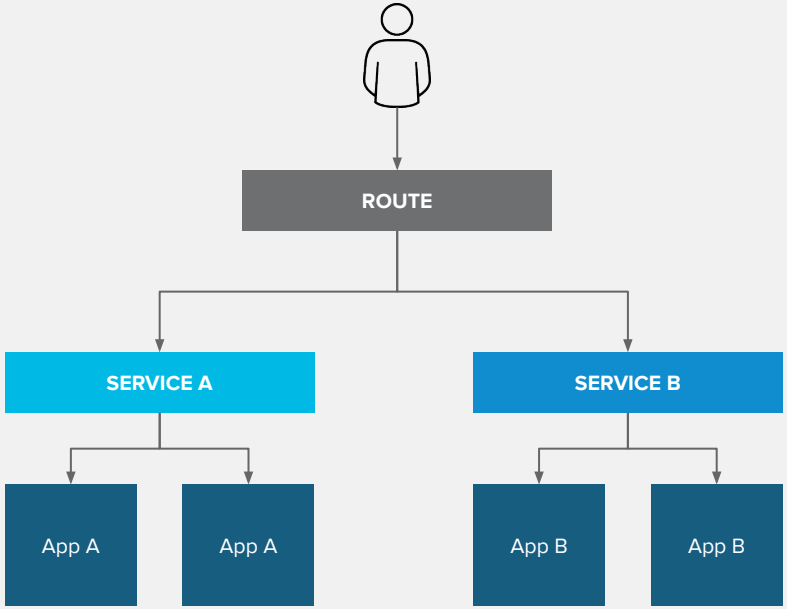
ROUTING AND EXTERNAL LOAD-BALANCING

- Pluggable routing architecture
  - HAProxy Router
  - F5 Router
- Multiple-routers with traffic sharding
- Router supported protocols
  - HTTP/HTTPS
  - WebSockets
  - TLS with SNI
- Non-standard ports via cloud load-balancers, external IP, and NodePort

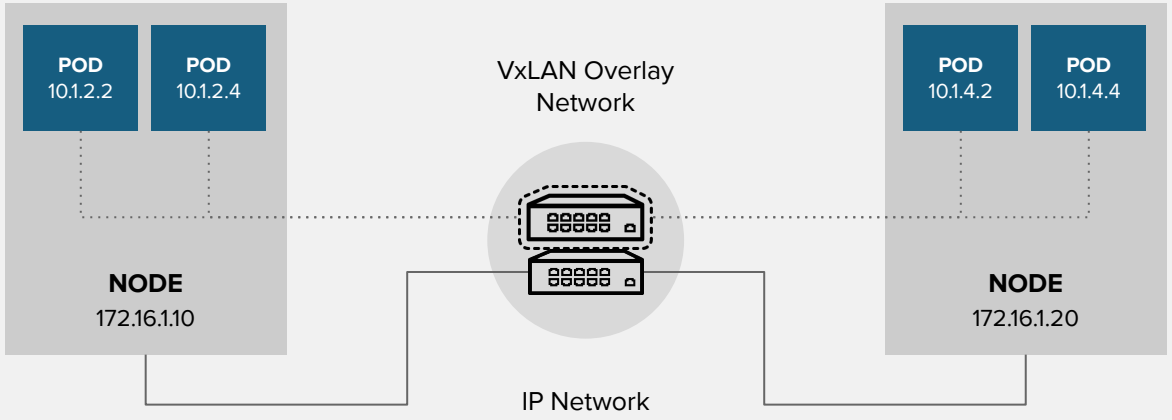


ROUTE SPLIT TRAFFIC

Split Traffic Between Multiple Services For A/B Testing, Blue/Green and Canary Deployments



OPENSHIFT NETWORKING



OPENSHIFT SDN

FLAT NETWORK (Default)

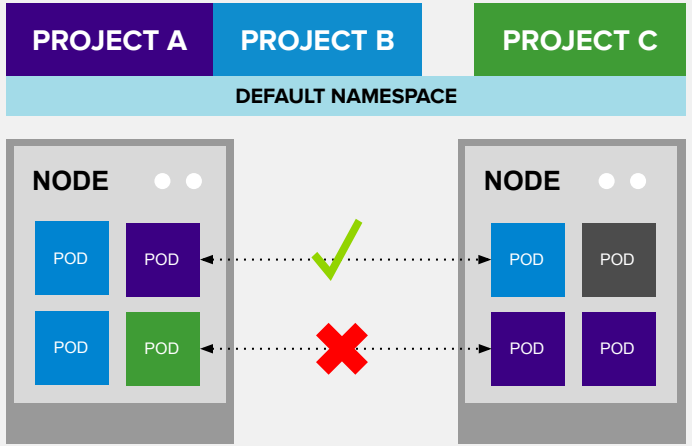
- All pods can communicate with each other across projects

MULTI-TENANT NETWORK

- Project-level network isolation
- Multicast support
- Egress network policies

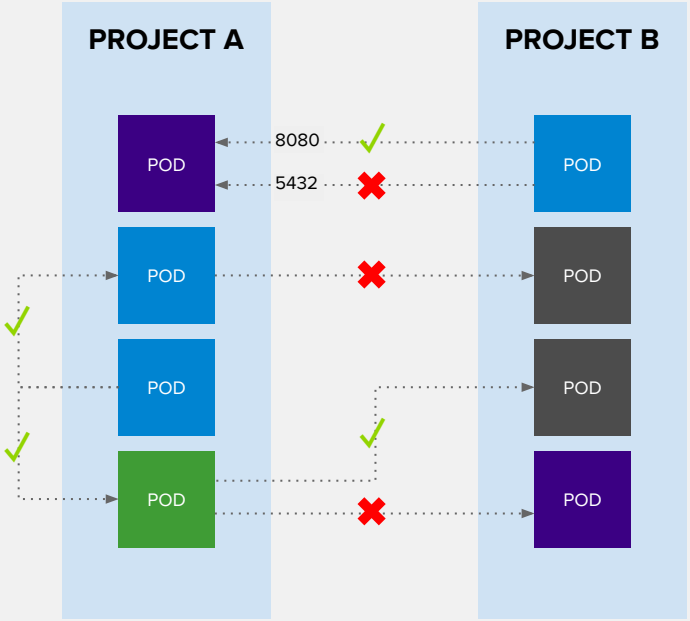
NETWORK POLICY

- Granular policy-based isolation



Multi-Tenant Network

OPENSHIFT SDN - NETWORK POLICY



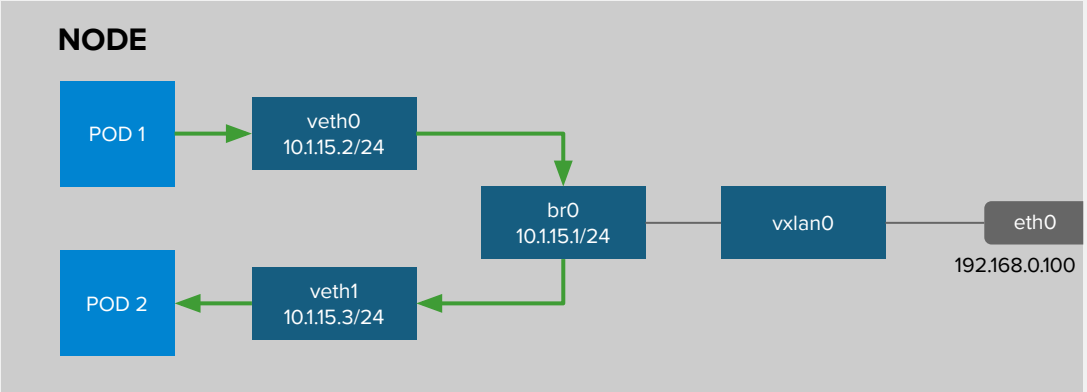
Example Policies

- Allow all traffic inside the project
- Allow traffic from green to gray
- Allow traffic to purple on 8080

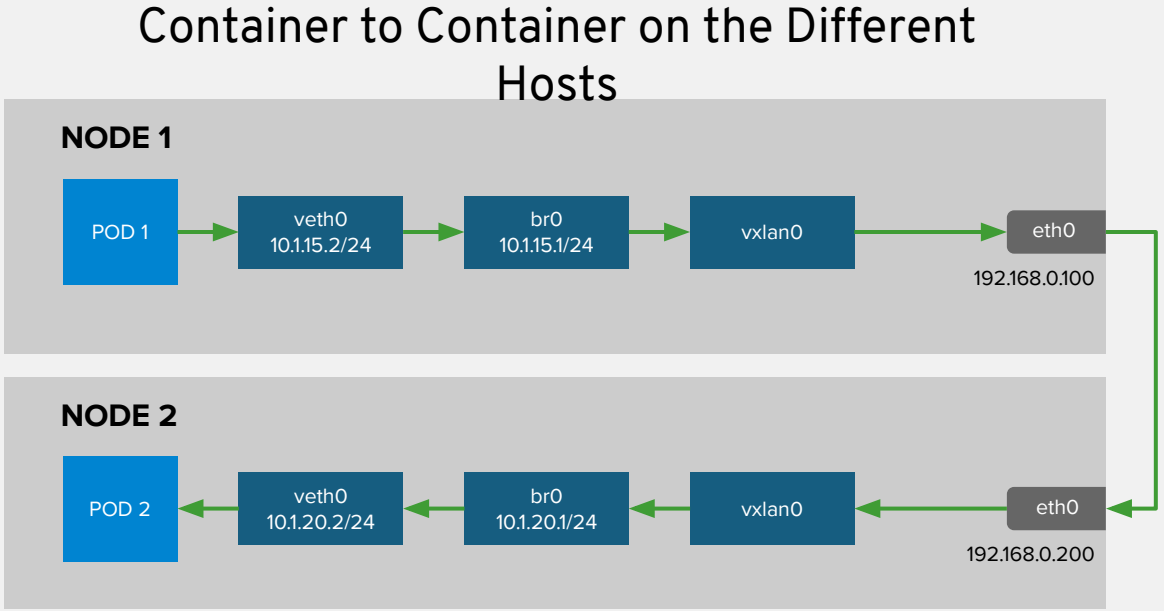
```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
  name: allow-to-purple-on-8080
spec:
  podSelector:
    matchLabels:
      color: purple
  ingress:
  - ports:
    - protocol: tcp
      port: 8080
```

OPENSHIFT SDN - OVS PACKET FLOW

Container to Container on the Same Host

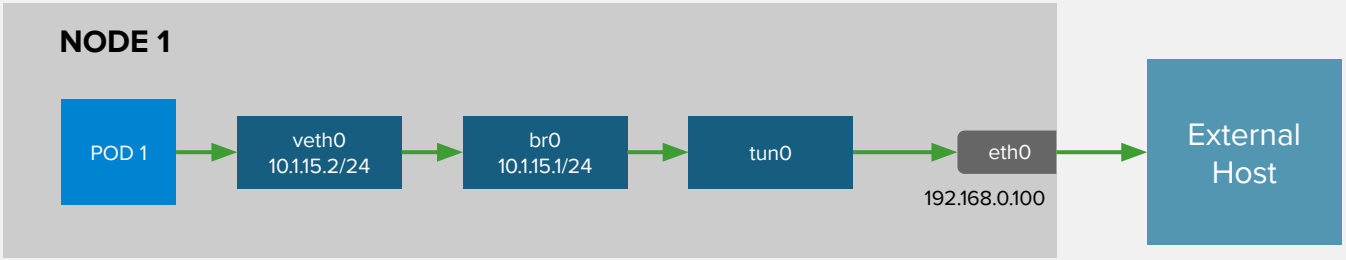


OPENSHIFT SDN - OVS PACKET FLOW



OPENSHIFT SDN - OVS PACKET FLOW

Container Connects to External Host



QUIZ

LAB

OpenShift Workshop

Agenda

Day 1

OpenShift Introduction
9h - 12h

Network Concepts
13h - 15h

Commands & Troubleshooting
15h - 17h

Day 2

Security
9h - 12h

Persistent Storage
13h - 15h

Managing App Development
15h - 17h

Day 3

Metrics & Logging
9h - 11h

Quotas & Limits
11h - 15h


OPENSHIFT COMMANDS
redhat.

oc Command-line Tool

INSTALAÇÃO

- On Red Hat Enterprise Linux (RHEL) systems with valid subscriptions, the tool is available as an RPM file and installable using the **yum install** command.

```
[user@host ~]$ sudo yum install -y atomic-openshift-clients
```

- For other Linux distributions and other operating systems, such as Windows and macOS, native clients are available for download from the Red Hat Customer Portal. This also requires an active OpenShift subscription. These downloads are statically compiled to reduce incompatibility issues.

Useful Commands to Manage OpenShift

After the **oc** CLI tool has been installed, you can use the **oc help** command to display help information. There are **oc** subcommands for tasks such as:

- Logging in to and out of an OpenShift cluster.
- Creating, changing, and deleting projects.
- Creating applications inside a project.
- Creating a deployment configuration or a build configuration from a container image, and all associated resources.
- Creating, deleting, inspecting, editing, and exporting individual resources, such as pods, services, and routes inside a project.
- Scaling applications.
- Starting new deployments and builds.
- Checking logs from application pods, deployments, and build operations.

Useful Commands to Manage OpenShift

- You can use the **oc login** command to log in interactively, which prompts you for a server name, a user name, and a password, or you can include the required information on the command line.

```
[student@workstation ~]$ oc login https://master.lab.example.com:8443 \
-u developer -p openshift
```

- To check your current credentials, run the **oc whoami** command:

```
[student@workstation ~]$ oc whoami
```

- To create a new project, use the **oc new-project** command:

```
[student@workstation ~]$ oc new-project working
```

- Use run the **oc status** command to verify the status of the project:

```
[student@workstation ~]$ oc status
```

Useful Commands to Manage OpenShift

- To delete a project, use the **oc delete project** command:

```
[student@workstation ~]$ oc delete project working
```

- To log out of the OpenShift cluster, use the **oc logout** command:

```
[student@workstation ~]$ oc logout
Logged "developer" out on "https://master.lab.example.com:8443"
```

- It is possible to log in as the OpenShift cluster administrator from any master node without a password by using the **system:admin** argument for the **-u** option.

```
[root@master ~]# oc login -u system:admin
```

oc get

- If the administrator wants a summary of all of the most important components of the cluster, the **oc get all** command can be executed. This command iterates through the major resource types and prints out a summary of their information. For example:

```
NAME                    DOCKER REPO                    TAGS    UPDATED
is/registry-console     172.30.211.204:5000            3.3     2 days ago

NAME                  REVISION   DESIRED   CURRENT   TRIGGERED BY
dc/docker-registry    4          1         1         config

NAME                   DESIRED       CURRENT   READY     AGE
rc/docker-registry     -1     0      0         0         2d
(...)
```

Miscellaneous

- **oc export**
  - This command can be used to export a definition of a resource. Typical use cases include creating a backup, or to aid in modifying a definition. By default, the **export**command prints out the object representation in YAML format, but this can be changed by providing a **-o** option.
- **oc create**
  - This command can be used to create resources from a resource definition. Typically, this is paired with the **oc export** command for editing definitions.
- **oc delete RESOURCE_TYPE name**
  - This command can be used to remove a resource from the OpenShift cluster. Note that a fundamental understanding of the OpenShift architecture is needed here, because deleting managed resources such as pods results in newer instances of those resources being automatically re-created.
- **oc exec**
  - This command can be used to execute commands inside a container. You can use this command to run interactive as well as non-interactive batch commands as part of a script.

Resource Types

- **Container**
  - A definition of how to run one or more processes inside a portable Linux environment. Containers are started from an image and are usually isolated from other containers on the same machine.
- **Image**
  - A layered Linux file system that contains application code, dependencies, and any supporting operating system libraries. An image is identified by a name that can be local to the current cluster, or point to a remote Docker registry (a storage server for images).
- **Pod**
  - A set of one or more containers that are deployed onto a node and share a unique IP address and volumes (persistent storage). Pods also define the security and runtime policy for each container.

Resource Types

- **Label**
  - Labels are key-value pairs that can be assigned to any resource in the system for grouping and selection. Many resources use labels to identify sets of other resources.
- **Volume**
  - Containers are not persistent by default; their contents are cleared when they are restarted. Volumes are mounted file systems available to pods and their containers, and which may be backed by a number of host-local or network-attached storage endpoints. The simplest volume type is **`EmptyDir`**, which is a temporary directory on a single machine. Administrators can also allow you to request a *Persistent Volume* that is automatically attached to your pods.
- **Node**
  - Nodes are host systems set up in the cluster to run containers. Nodes are usually managed by administrators and not by end users.

Resource Types

- **Service**
  - A service is a logical name representing a set of pods. The service is assigned an IP address and a DNS name, and can be exposed externally to the cluster via a port or a route. An environment variable with the name ***SERVICE*_HOST** is automatically injected into other pods.
- **Route**
  - A route is a DNS entry that is created to point to a service so that it can be accessed from outside the cluster. Administrators can configure one or more routers to handle those routes, typically through a HAProxy load balancer.
- **Replication Controller**
  - A replication controller maintains a specific number of pods based on a template that matches a set of labels. If pods are deleted (because the node they run on is taken out of service), the controller creates a new copy of that pod. A replication controller is most commonly used to represent a single deployment of part of an application based on a built image.

Resource Types

- **Deployment Configuration**
  - A deployment configuration defines the template for a pod and manages deploying new images or configuration changes whenever the attributes are changed. A single deployment configuration is usually analogous to a single microservice. Deployment configurations can support many different deployment patterns, including full restart, customizable rolling updates, as well as pre and post lifecycle hooks. Each deployment is represented as a replication controller.
- **Build Configuration**
  - A build configuration contains a description of how to build source code and a base image into a new image. Builds can be source-based, using builder images for common languages such as Java, PHP, Ruby, or Python, or Docker-based, which create builds from a Dockerfile. Each build configuration has webhooks and can be triggered automatically by changes to their base images.

Resource Types

- **Build**
  - Builds create new images from source code, other images, Dockerfiles, or binary input. A build is run inside of a container and has the same restrictions that normal pods have. A build usually results in an image being pushed to a Docker registry, but you can also choose to run a post-build test that does not push an image.
- **Image Streams and Image Stream Tags**
  - An image stream groups sets of related images using tag names. It is analogous to a branch in a source code repository. Each image stream can have one or more tags (the default tag is called "latest") and those tags might point to external Docker registries, to other tags in the same stream, or be controlled to directly point to known images. In addition, images can be pushed to an image stream tag directly via the integrated Docker registry.

Resource Types

- **Secret**
  - The secret resource can hold text or binary secrets for delivery into your pods. By default, every container is given a single secret which contains a token for accessing the API (with limited privileges) at `/var/run/secrets/kubernetes.io/serviceaccount`. You can create new secrets and mount them in your own pods, as well as reference secrets from builds (for connecting to remote servers), or use them to import remote images into an image stream.
- **Project**
  - All of the above resources (except nodes) exist inside of a project. Projects have a list of members and their roles, such as `view`, `edit`, or `admin`, as well as a set of security controls on the running pods, and limits on how many resources the project can use. Resource names are unique within a project. Developers may request that projects be created, but administrators control the resources allocated to projects.

LAB


OPENSHIFT TROUBLESHOOTING
redhat.

OpenShift Troubleshooting Commands

- Standard **sosreport** utility that gathers information about the environment along with docker and OpenShift-related information:
  - ```
    [root@master ~]# sosreport -k docker.all=on -k docker.logs=on
    sosreport (version 3.3)

    This command will collect diagnostic and configuration information from
    this Red Hat Enterprise Linux system and installed applications.
    ```
- **oc get events**
  - *Events* allow OpenShift to record information about life-cycle events in a cluster. They allow developers and administrators to view information about OpenShift components in a unified way. The **oc get events** command provides information about events in an OpenShift namespace. Examples of events that are captured and reported are listed below:
    - Pod creation and deletion
    - Pod placement scheduling
    - Master and node status
    ```

OpenShift Troubleshooting Commands

- **oc get events**
  - Web console in the Monitoring → Events page

6:23:46 PM	Pod scaling-1-jb7zj	Created Created container with docker id 2d0b43a6b207; Security:[seccomp=unconfined]
6:23:42 PM	Pod scaling-1-dq36m	Failed ⚠ Failed to pull image "172.30.53.104:5000/scaling /scaling@sha256:88a9beeab467735484c5405e1f241b65cb03c2de993ba9e1bbbbd662 123985c0": Get http://172.30.53.104:5000/v2/: dial tcp 172.30.53.104:5000: getsockopt: connection refused
6:23:42 PM	Pod scaling-1-dq36m	Failed sync ⚠ Error syncing pod, skipping: failed to "StartContainer" for "scaling" with ErrImagePull: "Get http://172.30.53.104:5000/v2/: dial tcp 172.30.53.104:5000: getsockopt: connection refused"
6:23:42 PM	Pod scaling-1-dq36m	Back off Back-off pulling image "172.30.53.104:5000/scaling /scaling@sha256:88a9beeab467735484c5405e1f241b65cb03c2de993ba9e1bbbbd662 123985c0"
6:23:42 PM	Pod scaling-1-dq36m	Failed sync ⚠ Error syncing pod, skipping: failed to "StartContainer" for "scaling" with ImagePullBackOff: "Back-off pulling image \"172.30.53.104:5000/scaling /scaling@sha256:88a9beeab467735484c5405e1f241b65cb03c2de993ba9e1bbbbd662 123985c0\""

OpenShift Troubleshooting Commands

- **oc logs**
  - The **oc logs** command retrieves the log output for a specific build, deployment, or pod. This command works for builds, build configurations, deployment configurations, and pods.
- **oc rsh**
  - The **oc rsh** command opens a remote shell session to a container. This is useful for logging in and investigating issues in a running container.
- **oc rsync**
  - The **oc rsync** command copies the contents to or from a directory in a running pod. If a pod has multiple containers, you can specify the container ID using the **-c** option. Otherwise, it defaults to the first container in the pod. This is useful for transferring log files and configuration files from the container.
- **oc port-forward**
  - You can use the **oc port-forward** command to forward one or more local ports to a pod. This allows you to listen on a given or random port locally, and have data forwarded to and from given ports in the pod.

Troubleshooting Common Issues

- **Resource Limits and Quota Issues**
  - If you try to create more pods than is allowed in a project with quota restrictions on pod count, you will see the following output when you run the **oc get events** command:

```
14m
Warning  FailedCreate  {hello-1-deploy} Error creating: pods "hello-1" is forbidden:
exceeded quota: project-quota, requested: cpu=250m, used: cpu=750m, limited: cpu=900m
```

- ***ErrImagePull* and *ImgPullBackOff* Errors**
  - These errors are caused by an incorrect deployment configuration, wrong or missing images being referenced during deployment, or improper docker configuration. Use the **oc get events** and **oc describe** commands to check for details. Fix deployment configuration errors by editing the deployment config using the **oc edit dc/<deploymentconfig>** command.

Troubleshooting Common Issues

- **Incorrect Docker Configuration**
  - Incorrect docker configuration on masters and nodes can cause many errors during deployment. Specifically, check the **`ADD_REGISTRY`**, **`INSECURE_REGISTRY`**, and **`BLOCK_REGISTRY`** settings and ensure that they are valid. Use the **systemctl status**, **oc logs**, **oc get events**, and **oc describe** commands to troubleshoot.
  - You can change the docker service log levels by adding the **`--log-level`** parameter for the **`OPTIONS`** variable in the docker configuration file located at **`/etc/sysconfig/docker`**. For example, to set the log level to debug:

```
OPTIONS='--insecure-registry=172.30.0.0/16 --selinux-enabled --log-level=debug'
```

Troubleshooting Common Issues

- **Master and Node Service Failures**
  - Use the **systemctl status** command for troubleshooting issues with the **`atomic-openshift-master`**, **`atomic-openshift-node`**, **`etcd`**, and **`docker`** services. Use the **journalctl -u <unit-name>** command to view the system log for issues related to the previously listed services.
  - You can increase the verbosity of logging from the **`atomic-openshift-node`** and the **`atomic-openshift-master`** services by editing the **`--loglevel`** variable in the respective configuration files, and then restarting the associated service (**`/etc/sysconfig/atomic-openshift-master`**).
- **Failures in Scheduling Pods**
  - A sample pod scheduling failure due to insufficient CPU is shown below, as output from the **oc describe** command:

```
{default-scheduler }  Warning  FailedScheduling pod (FIXEDhello-phb4j) failed to fit in any node
fit failure on node (hello-wx0s): Insufficient cpu
fit failure on node (hello-tgfm): Insufficient cpu
```

LAB

OpenShift Workshop

Agenda

Day 1

OpenShift Introduction
9h - 12h

Network Concepts
13h - 15h

Commands & Troubleshooting
15h - 17h

Day 2

Security
9h - 12h

Persistent Storage
13h - 15h

Managing App Development
15h - 17h

Day 3

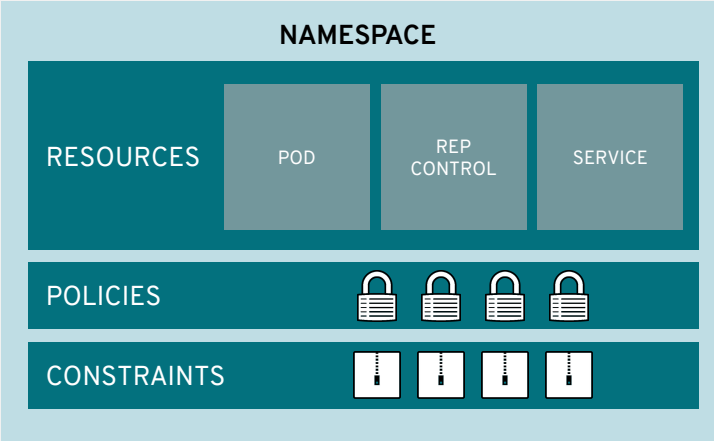
Metrics & Logging
9h - 11h

Quotas & Limits
11h - 15h


SECURITY
redhat.

K8S NAMESPACES

- A Kubernetes namespace provides a mechanism for grouping a set of related resources together. In OpenShift Container Platform, a project is a Kubernetes namespace with additional annotations.
- The following components apply to projects:
  - Objects: Pods, services, replication controllers, and more.
  - Policies: Rules that determine which actions users can or cannot perform on objects.
  - Constraints: Quotas for each kind of object that can be limited.



CLUSTER ADMINISTRATION

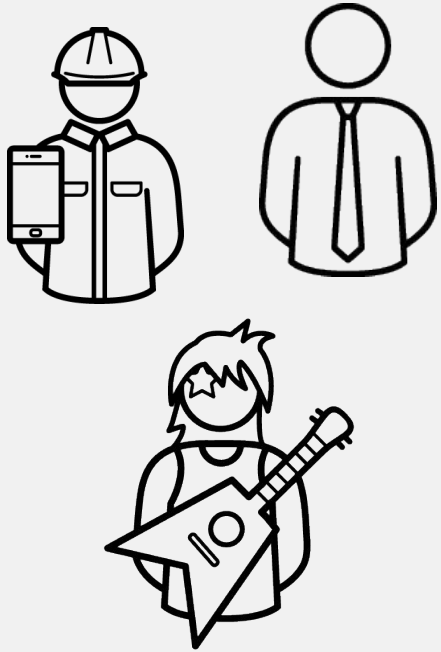
- Cluster administrators can **create projects and delegate administrative rights** for the project to any user.
- Administrators can **give users access** to certain projects, **allow** them to **create their own**, and **give** them **administrative rights** within **individual projects**.

```
[root@master~]$ oc adm policy
remove-cluster-role-from-group self-provisioner \
    system:authenticated system:authenticated:oauth

[root@master~]$ oc adm policy add-cluster-role-to-group \
    self-provisioner system:authenticated \
    system:authenticated:oauth
```

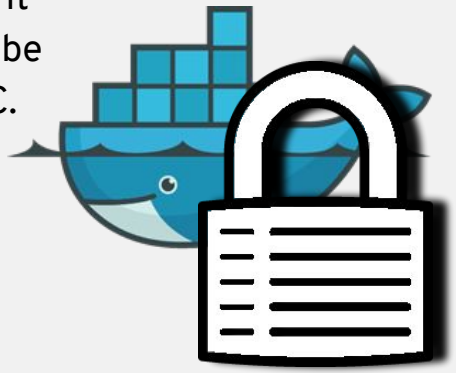
USER TYPES

- **Regular users:** Most interactive OCP users are represented. Regular users are represented with the User object.
- **System users:** Mainly for securely interact with the API. System users include a cluster administrator (with access to everything), a per-node user, users for use by routers and registries, and various others. Examples: system:admin, system:openshift-registry.
- **Service accounts:** Special system users associated with projects; created automatically when the project is first created, and project administrators can create more for the purpose of defining access to the contents of each project. Represented with the ServiceAccount object. Examples of service account: serviceaccount:default:deployer and system:serviceaccount:foo:builder.



Security Context Constraints (SCCs)

- OpenShift provides **security context constraints (SCCs)** which control the actions a pod can perform and what resources it can access. By default, the execution of any container will be granted only the capabilities defined by the restricted SCC.

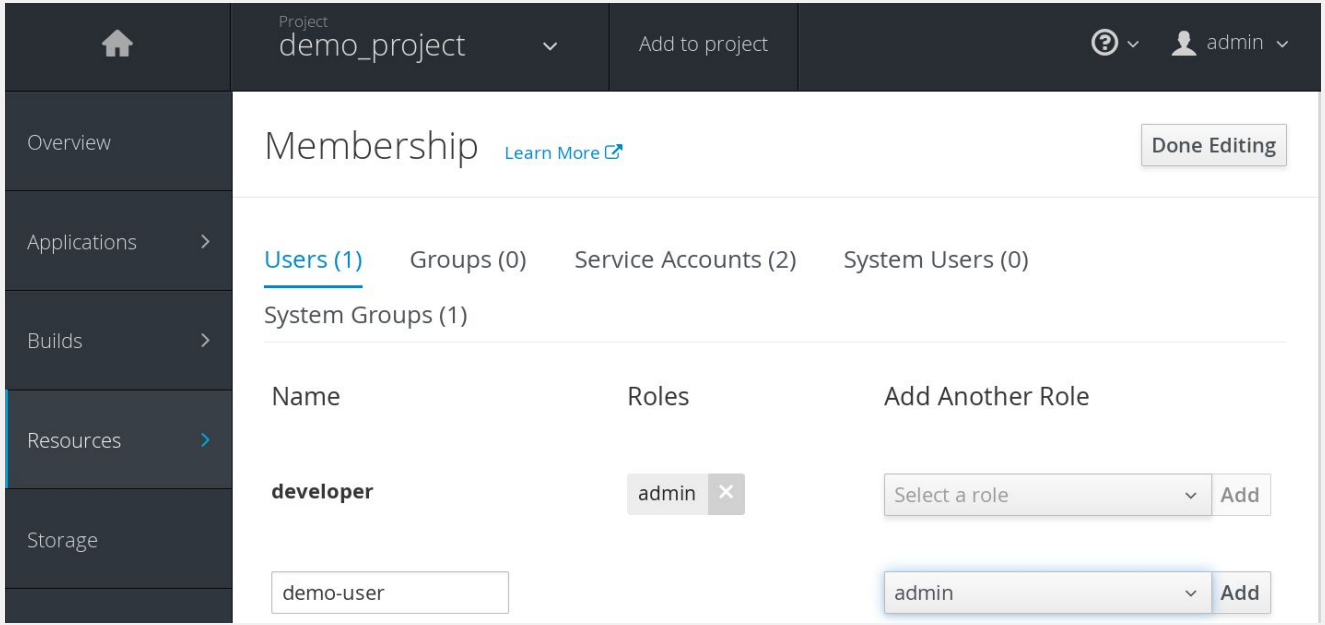


```
[user@demo ~]$ oc adm policy add-scc-to-user scc_name user_name
[user@demo ~]$ oc adm policy add-scc-to-group scc_name group_name

[user@demo ~]$ oc adm policy remove-scc-from-user scc_name user_name
[user@demo ~]$ oc adm policy remove-scc-from-group scc_name group_name
```

Managing User Membership (Webconsole)

- Webconsole → Resources → Membership



Managing User Membership (CLI)

```
[root@master ~]$ oc create user demo-user

[root@master ~]$ htpasswd /etc/origin/openshift-passwd demo-user

[root@master ~]$ oc policy add-role-to-user edit demo-user

[root@master ~]$ oc policy remove-role-from-user edit demo-user

[root@master ~]$ oc adm policy add-cluster-role-to-user cluster-admin admin


[root@bastion ~]# ansible masters -m shell -a "htpasswd -b /etc/origin/master/htpasswd developer
openshift"
```

Authentication Types

- Basic Authentication (Remote)
- Request Header Authentication
- Keystone Authentication
- LDAP Authentication
- GitHub Authentication


SECRETS & CONFIGMAPS
redhat.

SECRETS

- Hold sensitive information such as passwords, OpenShift Container Platform client configuration files, Docker configuration files, and private source repository credentials.
- Secret data can be referenced independently from its definition.
- Secret data volumes are backed by temporary file storage.
- Secret data can be shared within a namespace.

```
[user@demo ~]$ oc create secret generic secret_name \
    --from-literal=key1=secret1 \
    --from-literal=key2=secret2
```

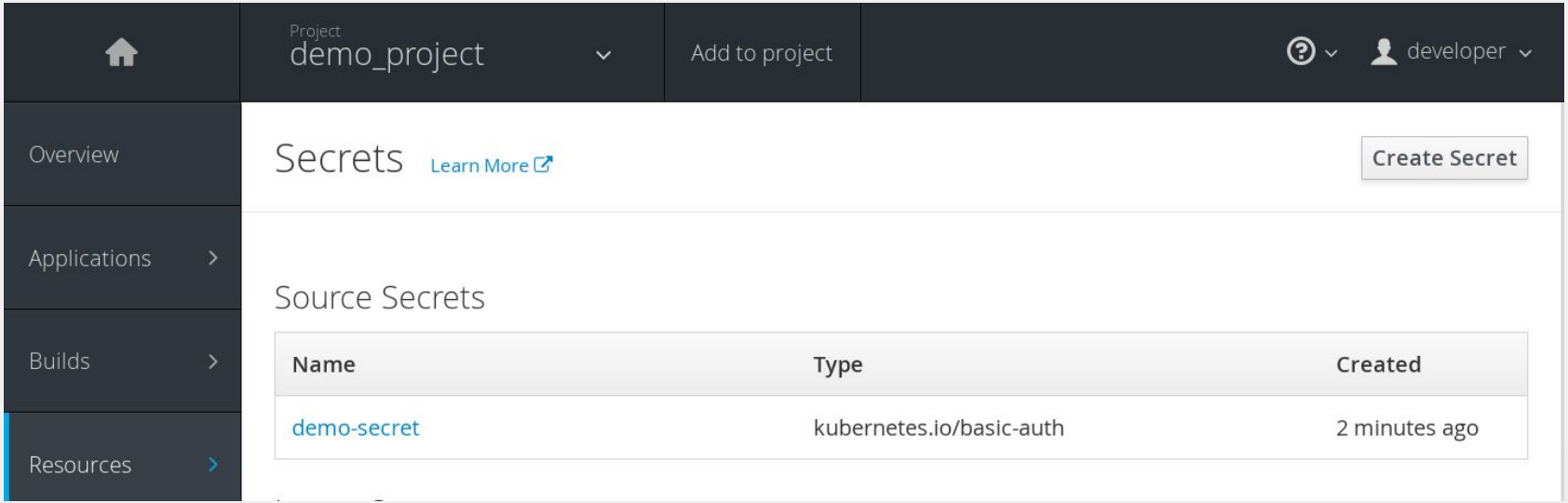
SECRETS

- Secret usage example:

```
env:
  - name: MYSQL_ROOT_PASSWORD
    valueFrom:
      secretKeyRef:
        key: username
        name: demo-secret
```

SECRETS (Webconsole)

- Webconsole → Resources → Secrets



CONFIGMAPS

- The ConfigMap object holds key-value pairs of configuration data that can be consumed in pods.

```
[user@demo ~]$ oc create configmap special-config \
    --from-literal=serverAddress=172.20.30.40

[user@demo ~]$ oc get configmaps special-config -o yaml
apiVersion: v1
data:
  key1: serverAddress=172.20.30.40
kind: ConfigMap
metadata:
  creationTimestamp: 2017-07-10T17:13:31Z
  name: special-config
(...)
```

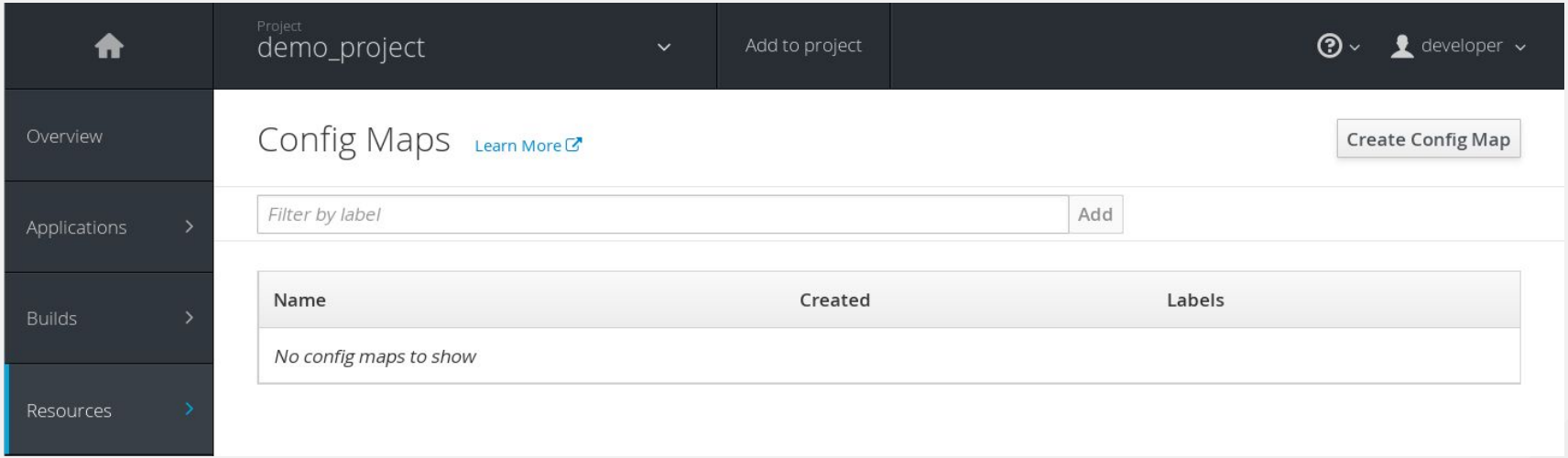
CONFIGMAPS

- The ConfigMap object holds key-value pairs of configuration data that can be consumed in pods.

```
env:
  - name: APISERVER
    valueFrom:
      configMapKeyRef:
        name: special-config
        key: serverAddress
```

CONFIGMAPS (Webconsole)

- Webconsole → Resources → Config Maps.



LAB

OpenShift Workshop

Agenda

Day 1

OpenShift Introduction
9h - 12h

Network Concepts
13h - 15h

Commands & Troubleshooting
15h - 17h

Day 2

Security
9h - 12h

Persistent Storage
13h - 15h

Managing App Development
15h - 17h

Day 3

Metrics & Logging
9h - 11h

Quotas & Limits
11h - 15h

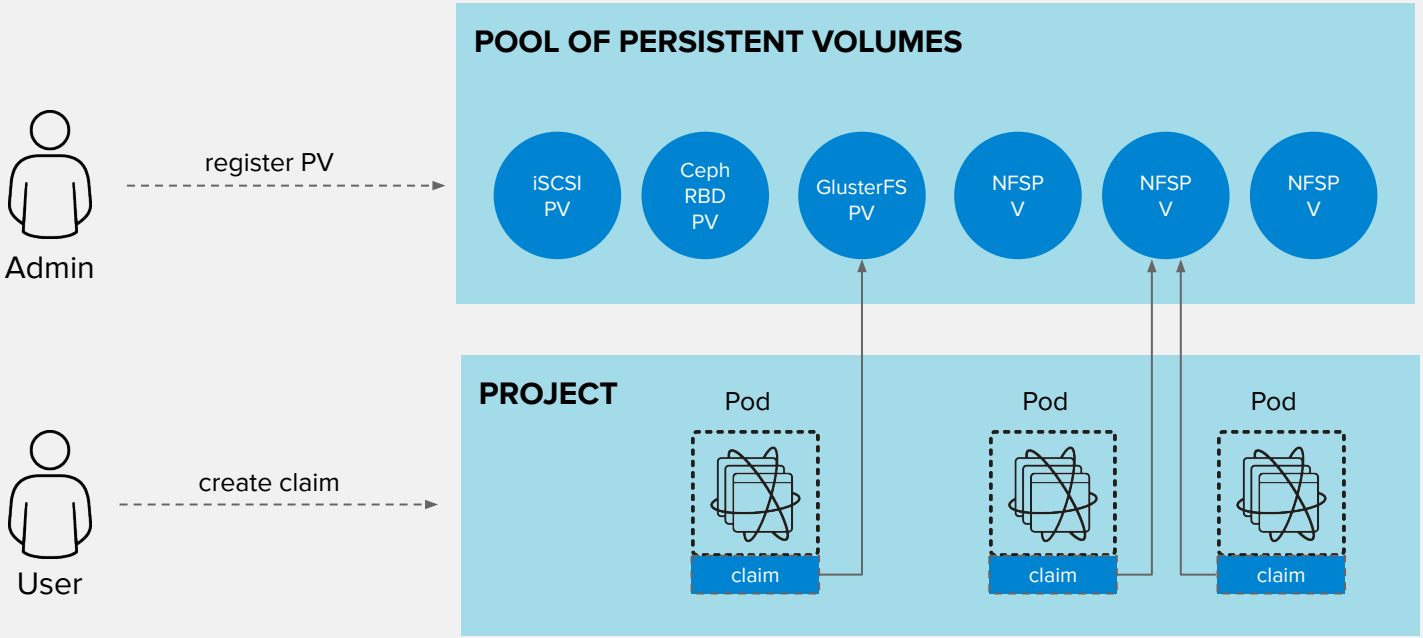

PERSISTENT STORAGE
redhat.

PERSISTENT STORAGE

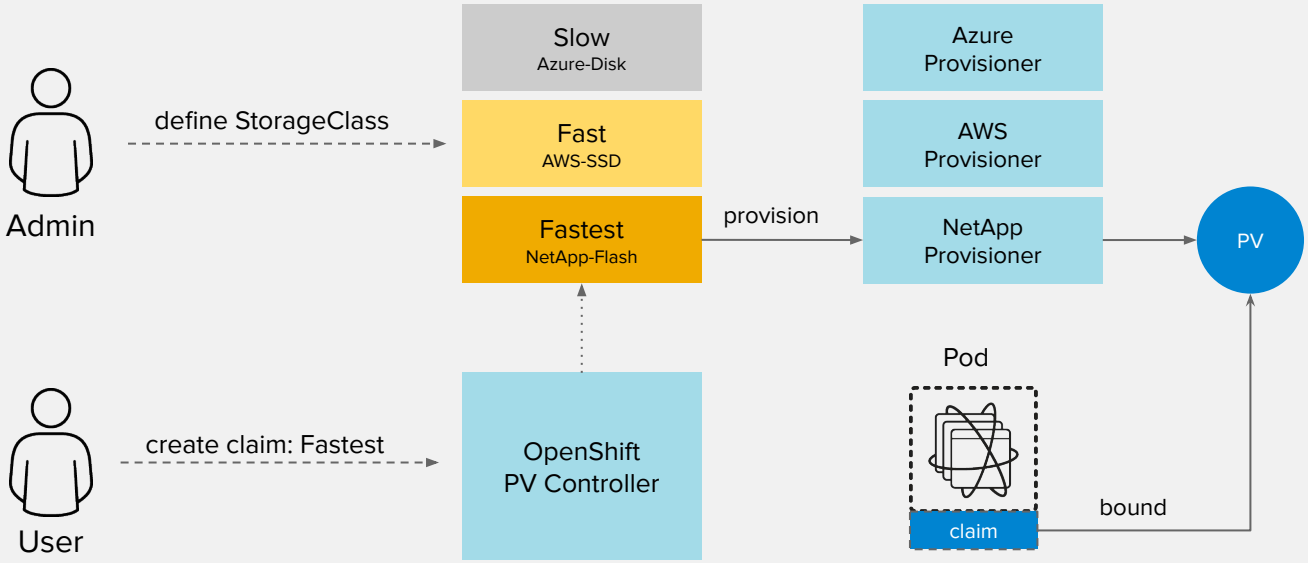
- Persistent Volume (PV) is tied to a piece of network storage
- Provisioned by an administrator (static or dynamically)
- Allows admins to describe storage and users to request storage
- Assigned to pods based on the requested size, access mode, labels and type

NFS	OpenStack Cinder	iSCSI	Azure Disk	AWS EBS	FlexVolume
GlusterFS	Ceph RBD	Fiber Channel	Azure File	GCE Persistent Disk	VMWare vSphere VMDK

PERSISTENT STORAGE



DYNAMIC VOLUME PROVISIONING

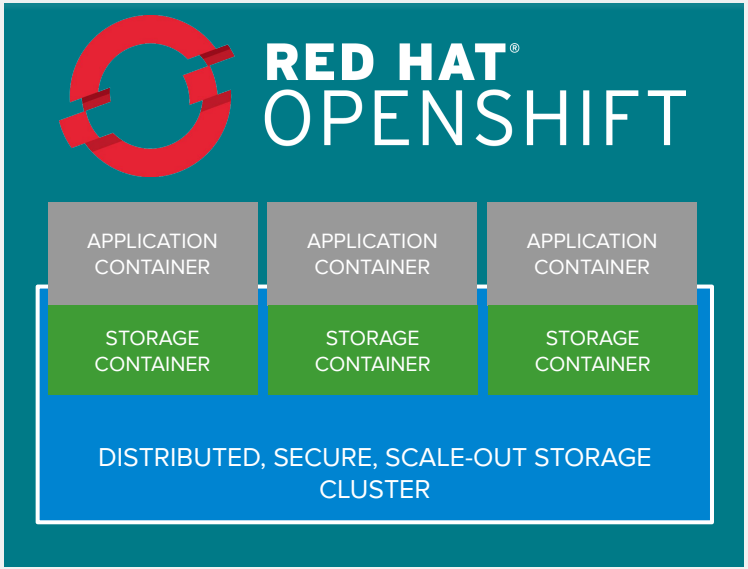


PERSISTENT VOLUME ACCESS MODES

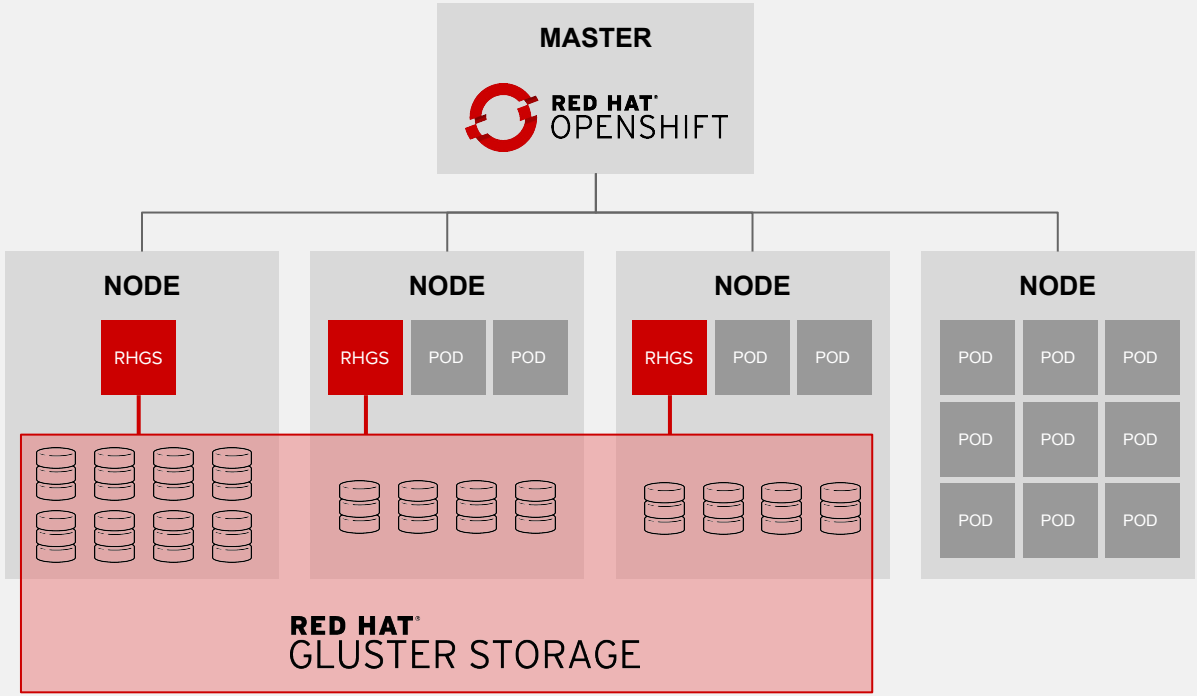
Access Mode	CLI Abbreviation	Description
ReadWriteOnce	RWO	The volume can be mounted as read/write by a single node.
ReadOnlyMany	ROX	The volume can be mounted read-only by many nodes.
ReadWriteMany	RWX	The volume can be mounted as read/write by many nodes.

OPENSHIFT CONTAINER STORAGE

- Containerized Red Hat Gluster Storage
- Native integration with OpenShift
- Unified Orchestration using Kubernetes for applications and storage
- Greater control & ease of use for developers
- Lower TCO through convergence
- Single vendor Support



OPENSHIFT CONTAINER STORAGE



ALLOCATING NFS

- At NFS server proceed with steps below:

```
$ mkdir /exports/<NEW_DIR>
$ chmod 777 /exports/<NEW_DIR>
$ vi /etc/exports.d/openshift-ansible.exports
   # Add line below in the end of file
    "/exports/<NEW_DIR>" *(rw,root_squash)
$ exportfs -ra
```

ALLOCATING PERSISTENT VOLUME

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: treinamento
spec:
  capacity:
    storage: 5Gi
  accessModes:
  - ReadWriteOnce
  nfs:
    path: /exports/<NEW_DIR>
    server: <nfs-server>
```

ALLOCATING PERSISTENT VOLUME

```
[student@workstation ~]$ oc set volume deploymentconfig/mysql \
    --add --overwrite --name=mysql-data -t pvc \
    --claim-name=mysql \
    --claim-size=1Gi \
    --claim-mode='ReadWriteMany'
persistentvolumeclaims/mysqldb-pvclaim
deployment config "mysqldb" updated


[student@workstation ~]$ oc get pvc
NAME              STATUS    VOLUME            CAPACITY   ACCESSMODES   AGE
mysqldb-pvclaim   Bound     mysqldb-volume    3Gi        RWX           15m
```

LAB

OpenShift Workshop

Agenda

Day 1

OpenShift Introduction
9h - 12h

Network Concepts
13h - 15h

Commands & Troubleshooting
15h - 17h

Day 2

Security
9h - 12h

Persistent Storage
13h - 15h

Managing App Development
15h - 17h

Day 3

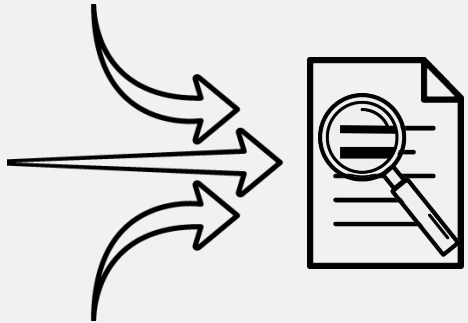
Metrics & Logging
9h - 11h

Quotas & Limits
11h - 15h

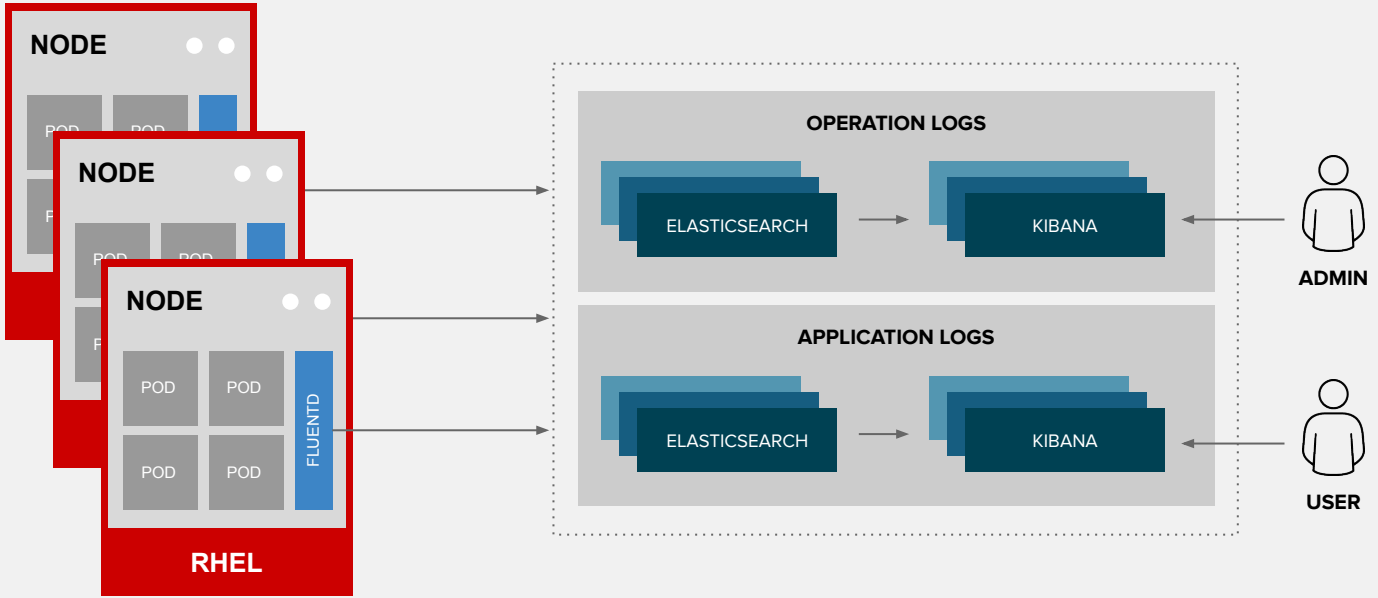

METRICS & LOGGING
redhat.

CENTRAL LOG MANAGEMENT WITH EFK

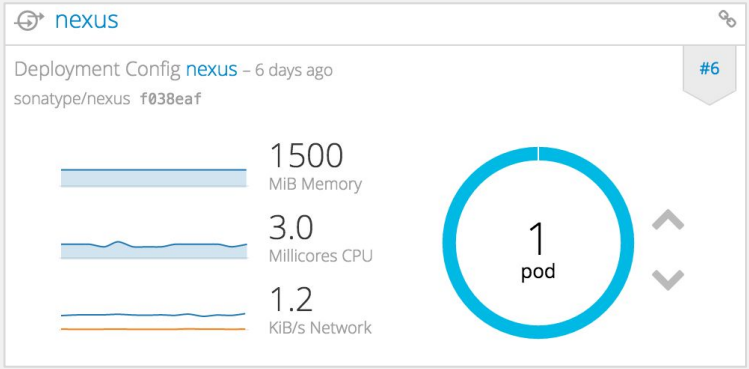
- EFK stack to aggregate logs for hosts and applications
  - **Elasticsearch:** a search and analytics engine to store logs
  - **Fluentd:** gathers logs and sends to Elasticsearch.
  - **Kibana:** A web UI for Elasticsearch.
- Access control
  - Cluster administrators can view all logs
  - Users can only view logs for their projects
- Ability to send logs elsewhere
  - External elasticsearch, Splunk, etc

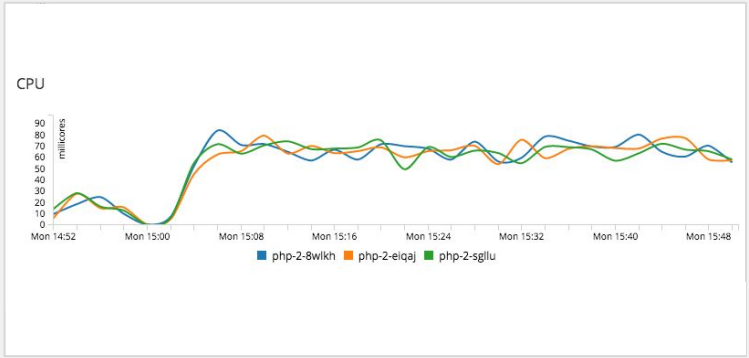


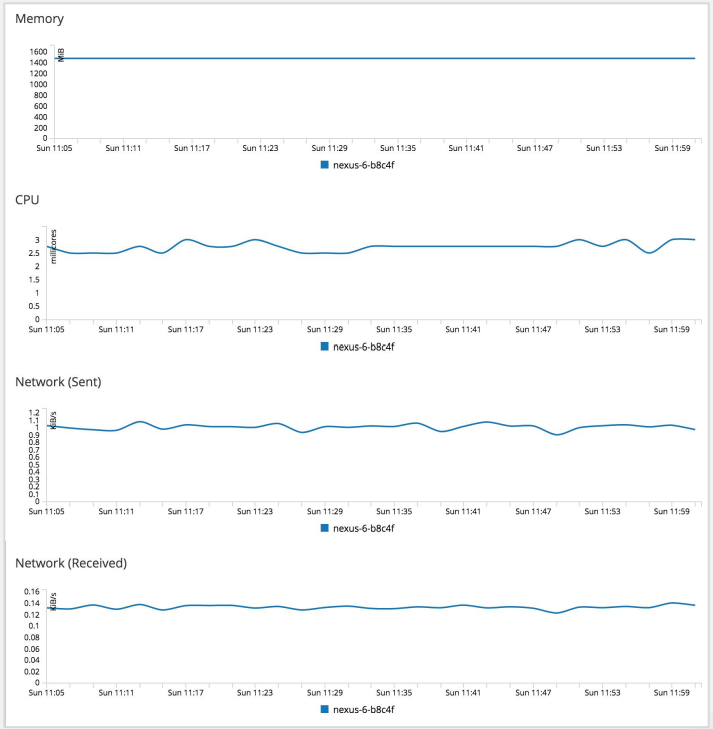
CENTRAL LOG MANAGEMENT WITH EFK



CONTAINER METRICS

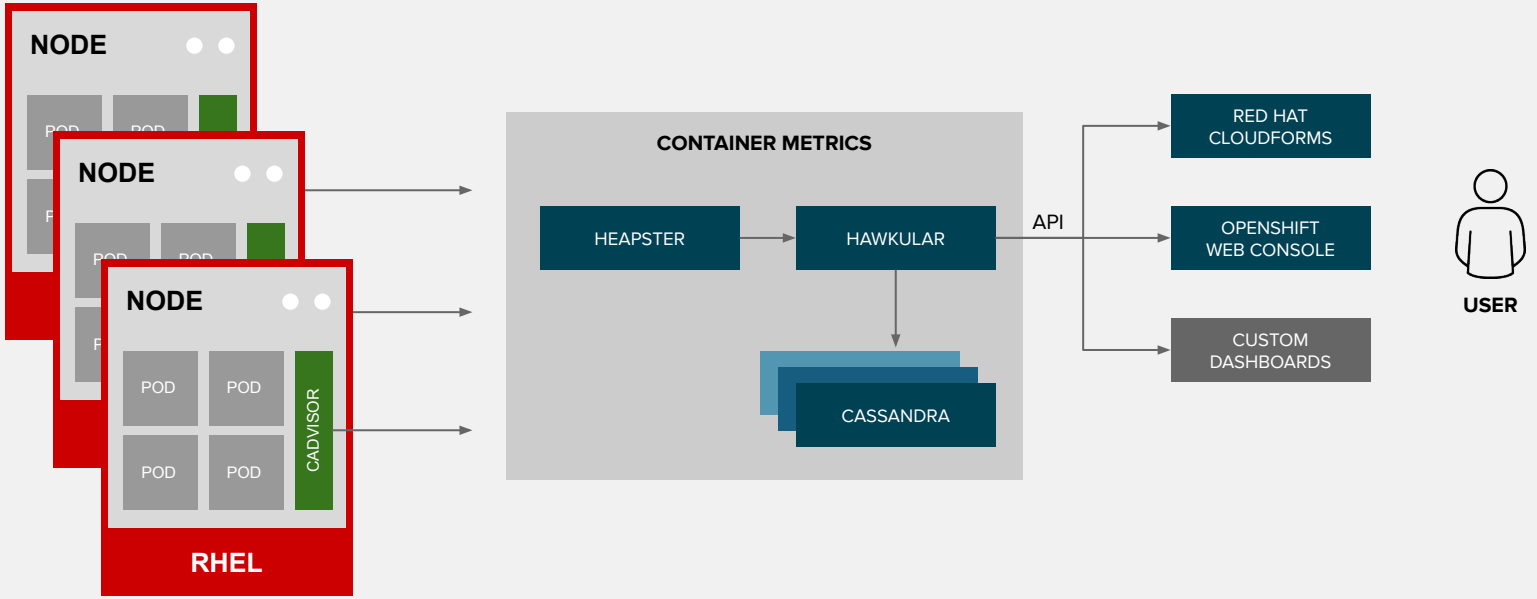






redhat.

CONTAINER METRICS



LAB

OpenShift Workshop

Agenda

Day 1

OpenShift Introduction
9h - 12h

Network Concepts
13h - 15h

Commands & Troubleshooting
15h - 17h

Day 2

Security
9h - 12h

Persistent Storage
13h - 15h

Managing App Development
15h - 17h

Day 3

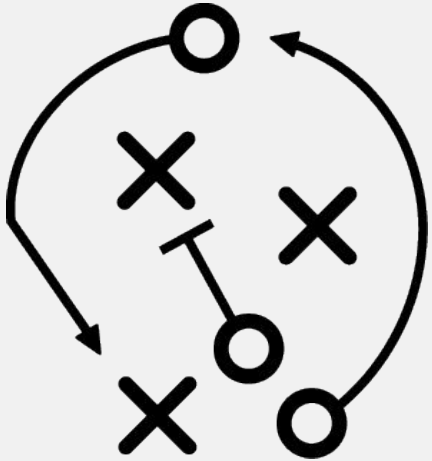
Metrics & Logging
9h - 11h

Quotas & Limits
11h - 15h


QUOTAS & LIMITS
redhat.

LIMITS AND RESTRICTIONS

- **Quotas**
  - Limit aggregate resource consumption per project.
- **Limit ranges**
  - Compute resource constraints in a project at the pod, container, image, image stream, and persistent volume claim level.
  - Amount of resources that a pod, container, image, image stream, or persistent volume claim can consume.
- **Compute resources**
  - Specify how much CPU and memory (RAM) each container needs in order to better schedule pods in the cluster and ensure satisfactory performance.



APPLYING QUOTAS

- **Object counts**
  - The number of Kubernetes resources, such as pods, services, and routes.

Object Count Name	Description
`pods`	Total number of pods
`replicationcontrollers`	Total number of replication controllers
`services`	Total number of services
`secrets`	Total number of secrets
`persistentvolumeclaims`	Total number of persistent volume claims

APPLYING QUOTAS

- **Compute resources**
  - The number of physical or virtual hardware resources, such as CPU, memory, and storage capacity.

Compute Resource Name	Description
`cpu`	Total CPU use across all containers
`memory`	Total memory use across all containers
`storage`	Total disk use across all containers

APPLYING QUOTAS

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: dev-quota
spec:
  hard:
    services: "10"
    cpu: "1300m"
    memory: "1.5Gi"
```

APPLYING LIMIT RANGES

Type	Resource Name	Description
Container	cpu	Minimum and maximum CPU allowed per container
Container	memory	Minimum and maximum memory allowed per container
Pod	cpu	Minimum and maximum CPU allowed across all containers in a pod
Pod	memory	Minimum and maximum memory allowed across all containers in a pod
Image	storage	Maximum size of an image that can be pushed to the internal registry
PVC	storage	Minimum and maximum capacity of the volume that can be requested by one claim

APPLYING LIMIT RANGES

```
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "dev-limits"
spec:
  limits:
    - type: "Pod"
      max:
        cpu: "2"
        memory: "1Gi"
      min:
        cpu: "200m"
        memory: "6Mi"
    - type: "Container"
      default:
        cpu: "1"
        memory: "512Mi"
```

APPLYING COMPUTE RESOURCES

```
apiVersion: v1
kind: Pod
spec:
  containers:
  - image: nginx
    name: nginx
    resources:
      requests:
        cpu: 100m
        memory: 200Mi
      limits:
        cpu: 200m
        memory: 400Mi
```

LAB

Links

- Official Docs:
  - https://docs.openshift.com/container-platform/3.9/welcome/index.html
- Openshift Blog:
  - https://blog.openshift.com
- Container Registry Red Hat:
  - https://access.redhat.com/containers/
- Kibana (logging):
  - https://www.elastic.co/guide/en/kibana/4.1/discover.html
- Authentication:
  - https://docs.openshift.com/container-platform/3.9/architecture/additional_concepts/authentication.html

Links

- Pipeline sample:
  - https://github.com/debianmaster/openshift-examples/tree/master/pipeline-example
- Authorization:
  - https://docs.openshift.com/container-platform/3.9/architecture/additional_concepts/authorization.html
- Hawkular Metrics & Grafana
  - https://www.hawkular.org/blog/2016/10/24/hawkular-metrics-openshift-and-grafana.html
- Quotas & Limits:
  - https://docs.openshift.com/enterprise/3.2/dev_guide/compute_resources.html

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos