redhat.

MONOLITHS TO MICROSERVICES: APP TRANSFORMATION

Hands-on Technical Workshop

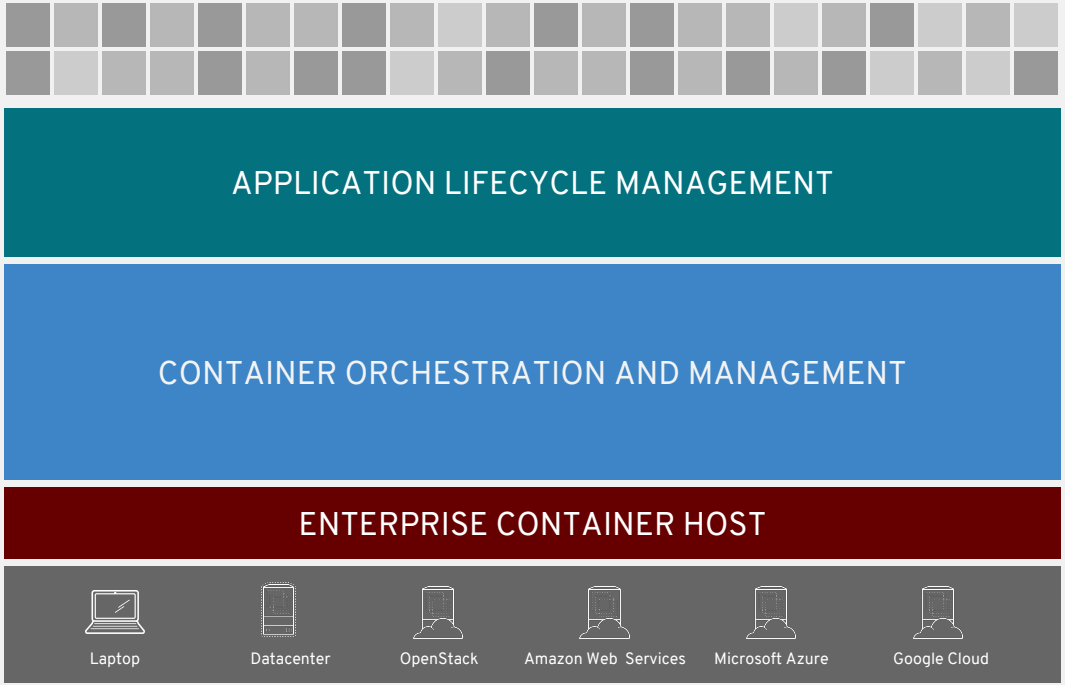
Thomas Qvarnström
Sr. Technical Marketing Manager
Middleware BU

James Falkner
Sr. Technical Marketing Manager
Middleware BU

A DEVELOPER INTRODUCTION TO OPENSHIFT

A secure and enterprise-grade container application platform based on Kubernetes for traditional and cloud-native applications

CLOUD-NATIVE CAPABILITIES WITH RED HAT OPENSHIFT

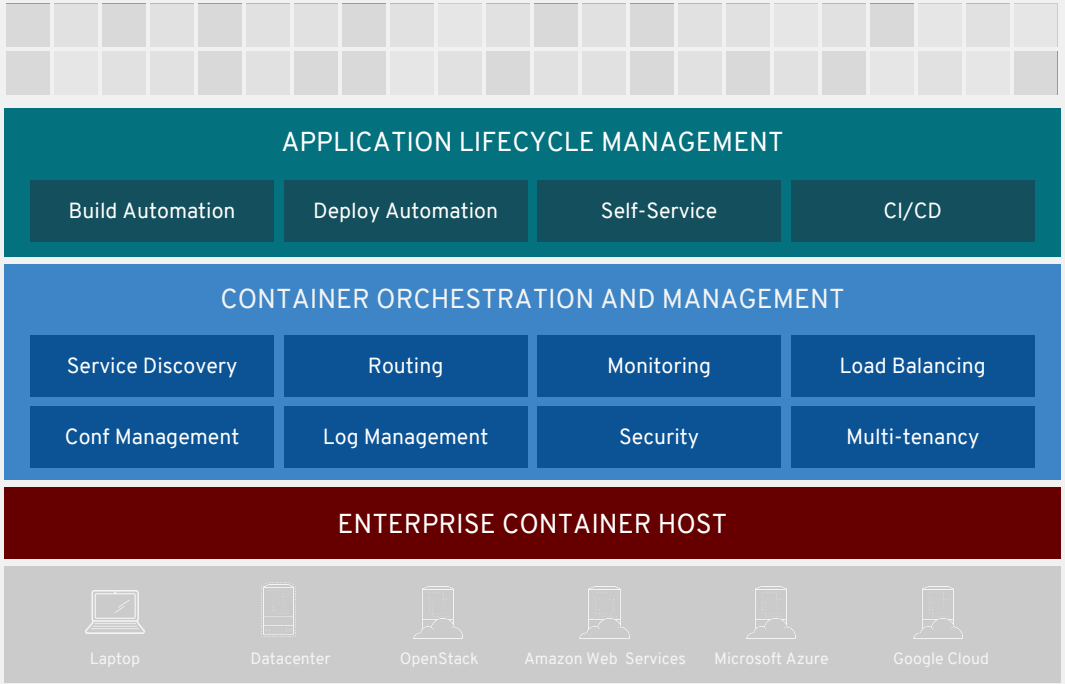


ANY CONTAINER

RED HAT® OPENSHIFT

ANY INFRASTRUCTURE

CLOUD-NATIVE CAPABILITIES WITH RED HAT OPENSHIFT



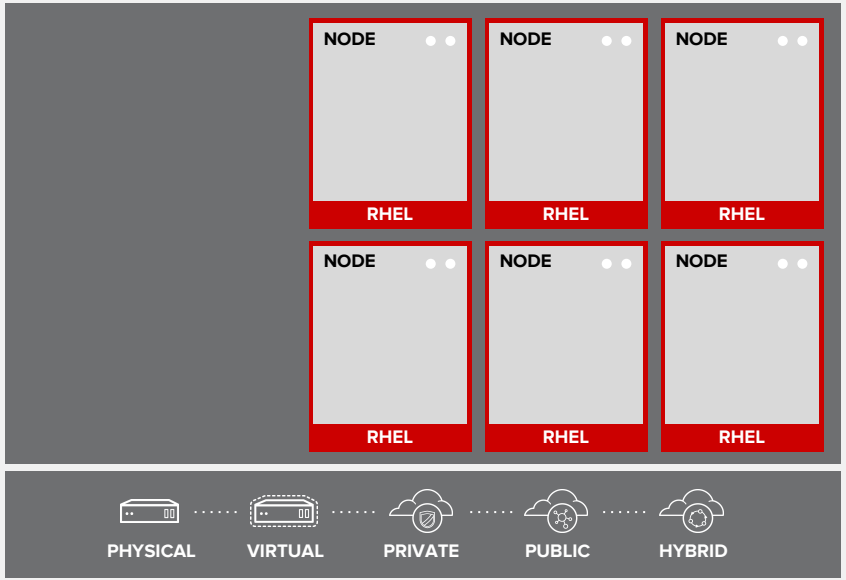
ANY CONTAINER

RED HAT® OPENSHIFT

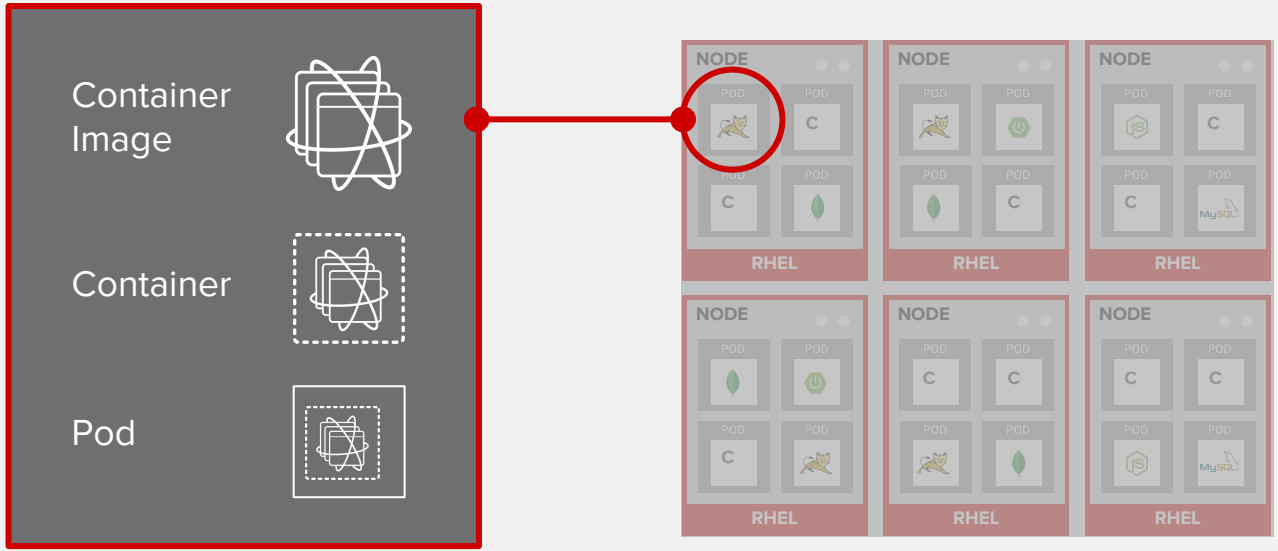
ANY INFRASTRUCTURE

YOUR CHOICE OF INFRASTRUCTURE

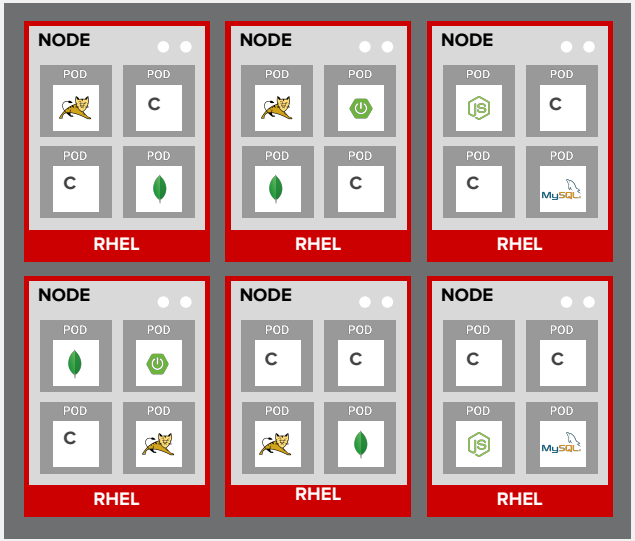
NODES RHEL INSTANCES WHERE APPS RUN



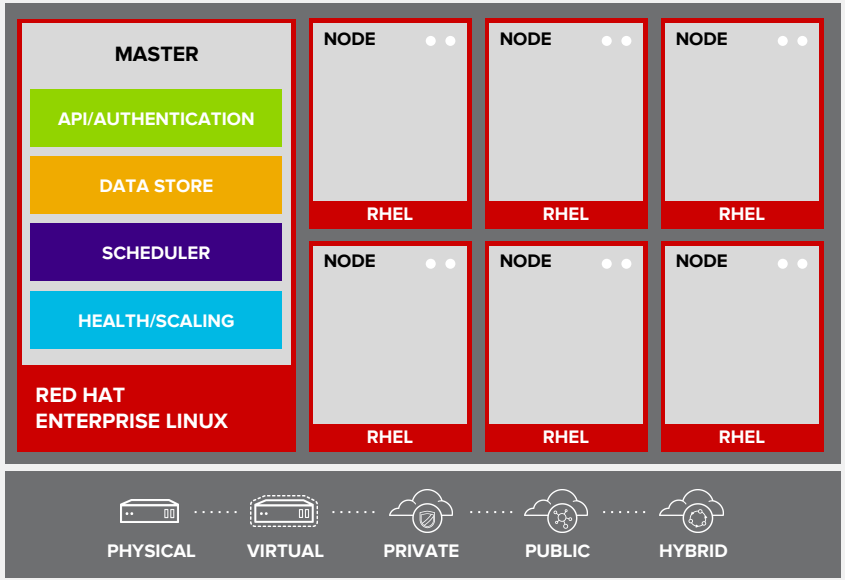
APPS RUN IN CONTAINERS



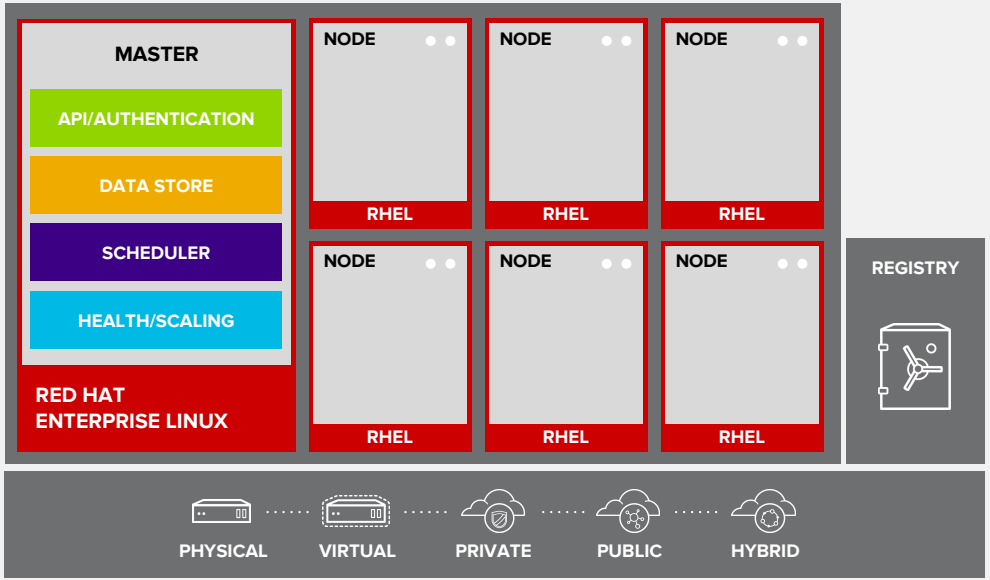
PODS ARE THE UNIT OF ORCHESTRATION



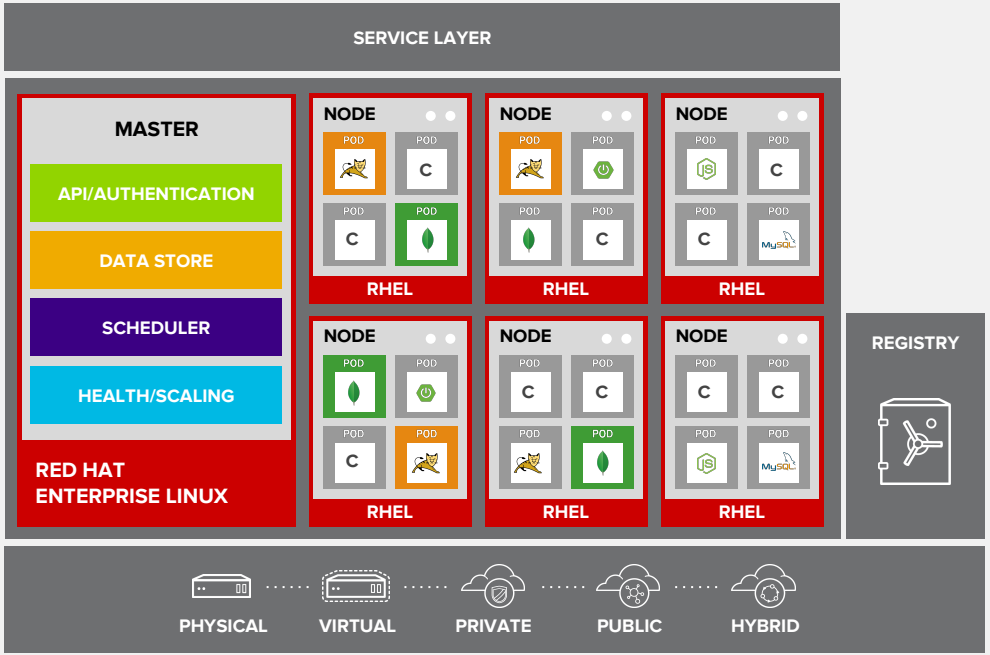
MASTERS ARE THE CONTROL PLANE



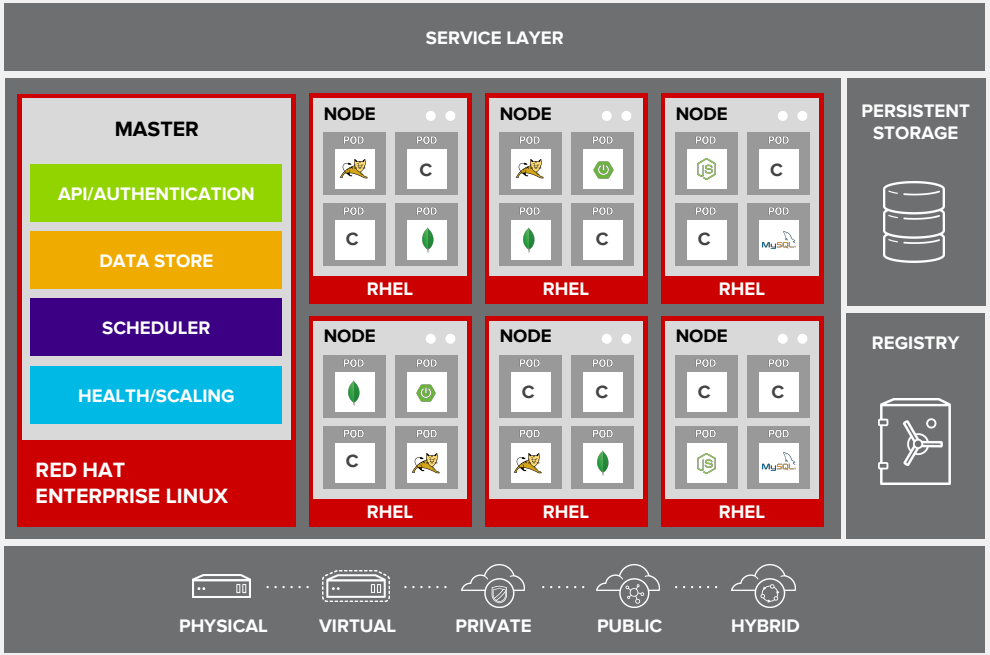
INTEGRATED CONTAINER REGISTRY



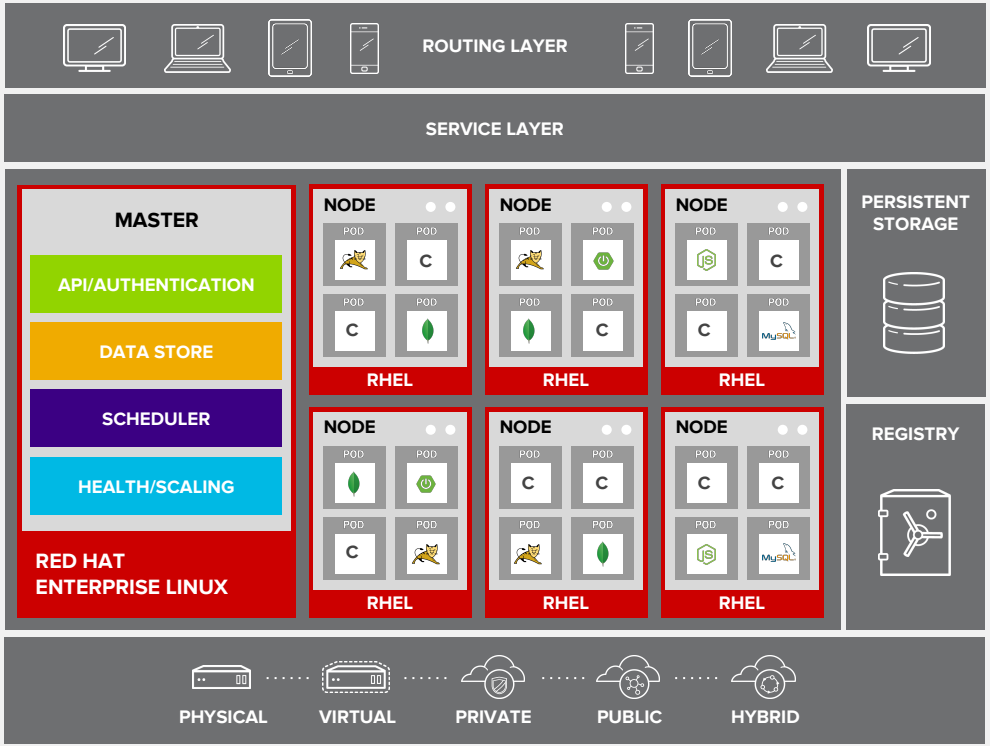
SERVICE DISCOVERY



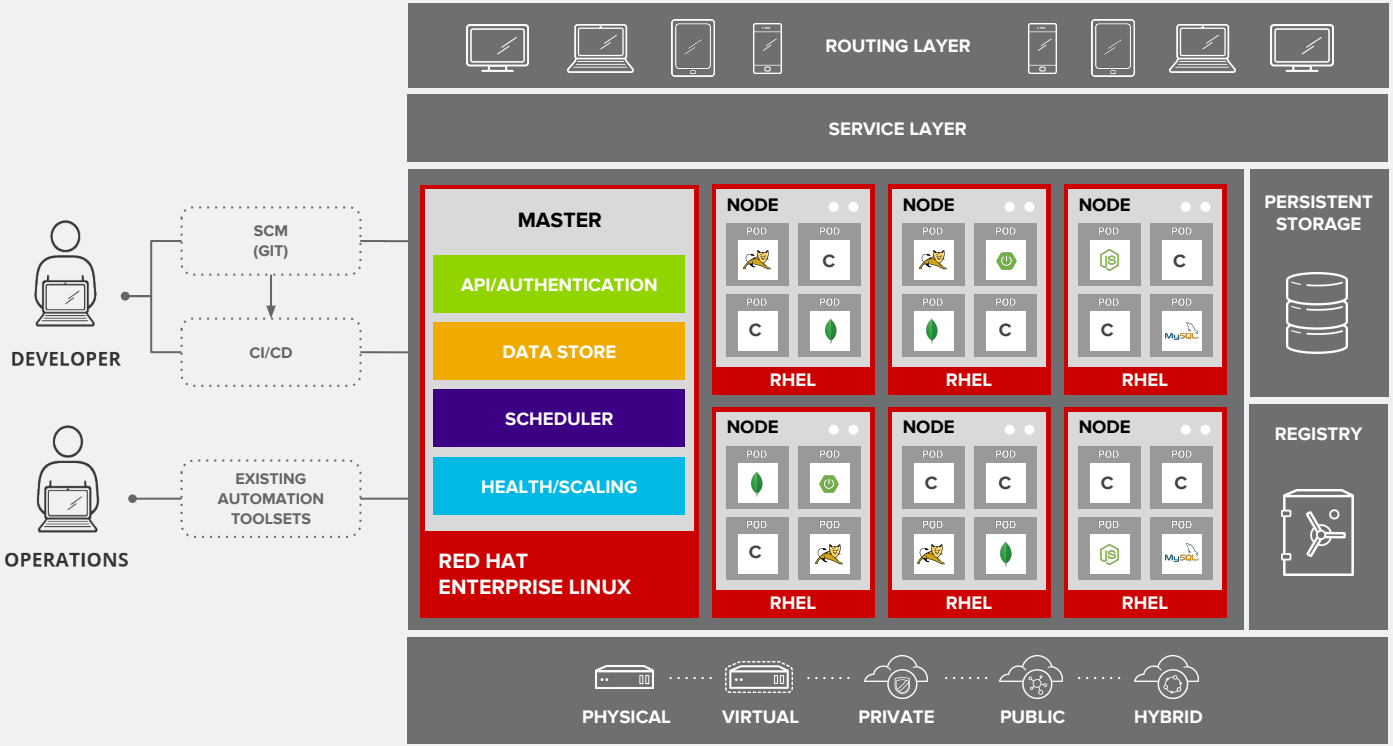
PERSISTENT DATA IN CONTAINERS



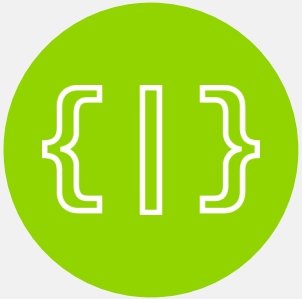
ROUTING AND LOAD-BALANCING



ACCESS VIA WEB, CLI, IDE AND API



BUILD AND DEPLOY APPLICATIONS ON OPENSHIFT



DEPLOY YOUR SOURCE CODE

DEPLOY YOUR APP BINARY

DEPLOY YOUR CONTAINER IMAGE

OPENSHIFT CONCEPTS

- **Project**
  A vehicle to scope resources in a cluster to avoid name collision, delegate management to trusted users and control resource consumption. Users must be given access to projects by administrators, or if allowed to create projects, automatically have access to their own projects.

OPENSHIFT CONCEPTS FOR DEVELOPERS

- **Container**
  Lightweight mechanisms for isolating running processes
- **Image**
  A binary that includes all of the requirements for running a single container, as well as metadata describing its needs and capabilities.
- **Image Stream**
  An image stream comprises one or more images identified by tags. It presents a single virtual view of related images from OpenShift registry, other image streams or external registries e.g. Docker Hub

OPENSHIFT CONCEPTS FOR DEVELOPERS

- **Pod**
  A small group (often only one) of tightly coupled Containers sharing network, storage, etc
- **Service**
  A set of pods that work together and expose services on TCP ports, and are automatically load-balanced
- **Route**
  A route is a way to externally expose a service by giving it an externally-reachable proper DNS hostname like www.example.com

OPENSHIFT CONCEPTS

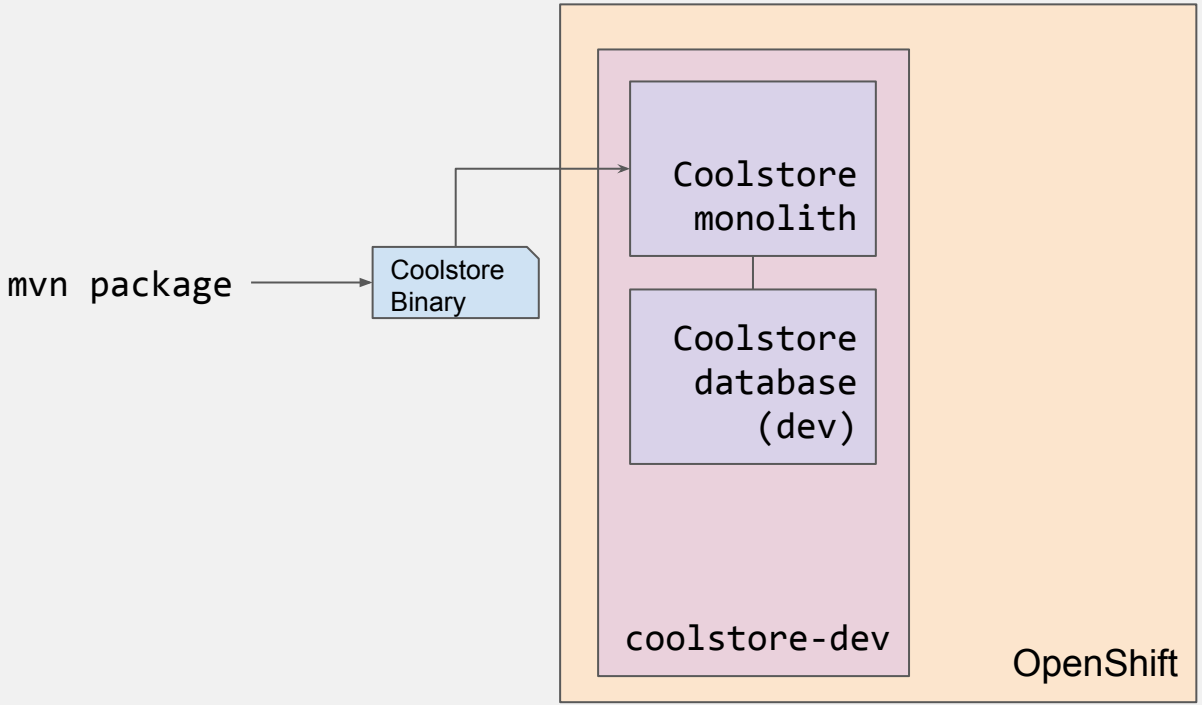
- **Replication Controller**
  Makes sure a specified number of homogeneous Pods are up and running. If there are too many pods, it will kill some. If there are too few, it will start more
- **Build**
  Process of transforming a set of parameters and source code into a runnable image
- **Build Config**
  Definition of the entire build process
- **Template**
  Describes a set of objects that can be parameterized and processed to produce a list of objects for creation by OpenShift

OPENSHIFT CONCEPTS

- **Deployment Configuration**
  Deployment creates a new replication controller and lets it start up new Pods. Deployment configuration defines the elements of the replication controller, triggers for creating a new deployment automatically, the strategy for transitioning between deployments and life cycle hooks to be invoked after a container is started and before it is stopped

LAB: DEVELOPER INTRODUCTION TO OPENSHIFT

CURRENT STATE



GOAL FOR LAB

In this lab you will learn:

- Important OpenShift concepts for developers
- How OpenShift makes developers and architects happier
- How to do efficient round-trip development:
  - Separate **dev** from **prod** environments
  - Quick deployments using **rsync** / port-forwarding
  - Debugging running applications with **jdb**
  - Promoting apps using **CI/CD Pipelines**

LAB: DEVELOPER INTRO TO OPENSHIFT

SCENARIO 3 A DEVELOPER INTRODUCTION TO OPENSHIFT

WRAP-UP AND DISCUSSION

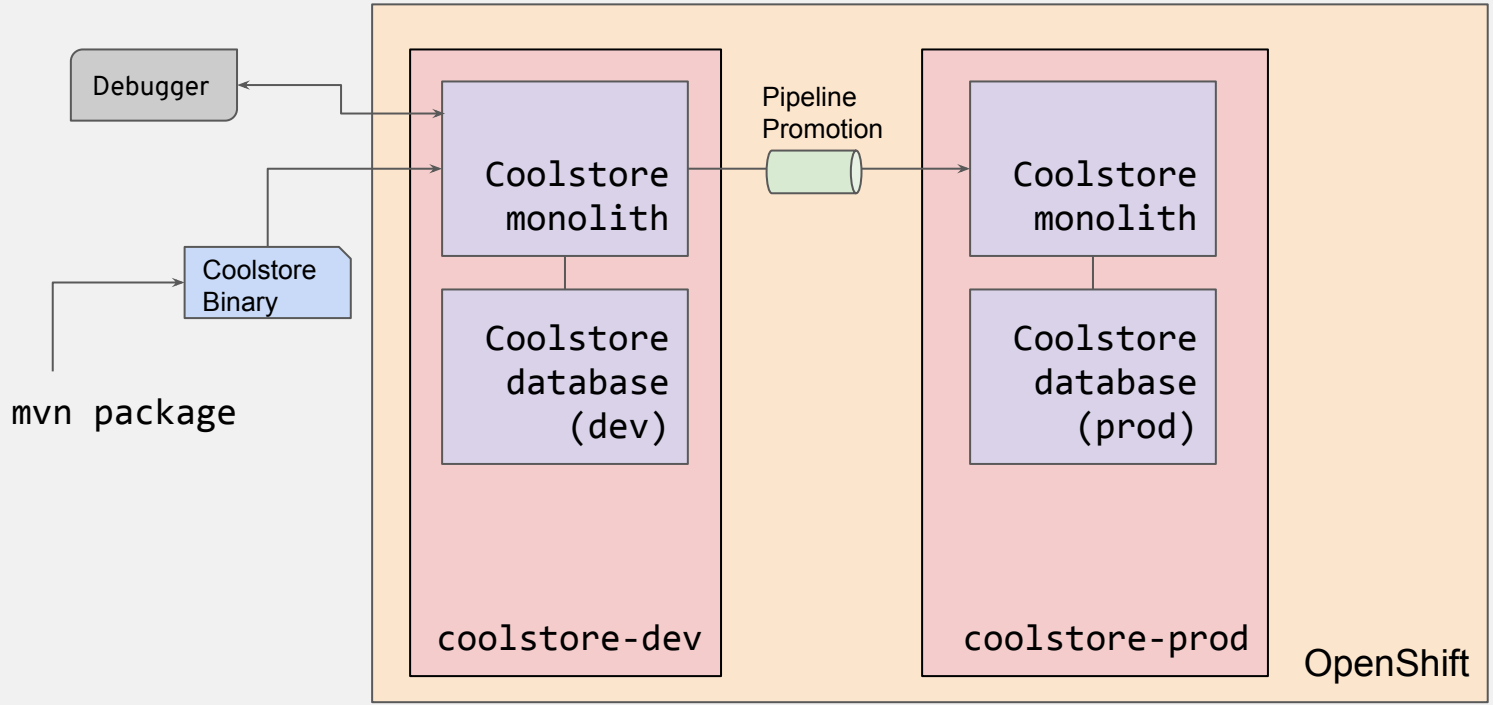
RESULT OF LAB

In this lab you learned how to:

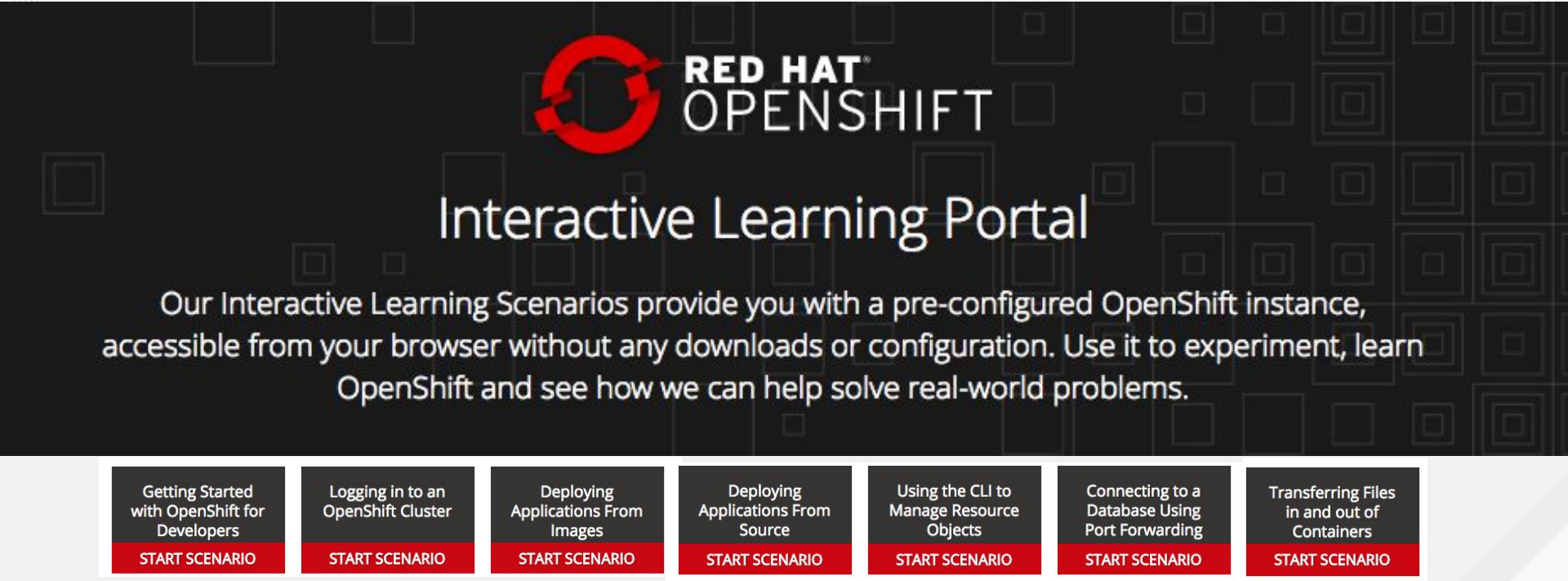
- Do quick deployments with **`oc rsync`**
- Debug running Java applications using **`jdb`**
- Create a production environment separate from dev
- Promote tested/verified builds between environments using OpenShift pipeline builds

You should now have two projects (dev and prod) running the same CoolStore app! In the next lab we will begin the process of breaking the monolith up into microservices.

DESIRED RESULT OF SCENARIO 3



LEARN MORE: learn.openshift.com



redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos