AUTOMATING CLOUD-NATIVE DEVELOPMENT USING ANSIBLE AND OPENSHIFT

James Falkner
Technical Evangelist, Red Hat
Application Platform

Vince Power
Solution Architect, Red Hat
Ansible

LAB GUIDE

WHAT YOU WILL LEARN

- Ansible Basics (1 hour)
  - Ansible Concepts
  - How to develop and run Playbooks for automating infrastructure deployment based on OpenShift
- Cloud Native Application Development (2 hours)
  - Bootstrapping Development Environment
  - Developing Microservices with Spring Boot
  - Automating Production Releases
  - Distributed Tracing and Fault Tolerance with Jaeger & Istio

RED HAT® ANSIBLE® Automation

RED HAT® OPENSHIFT® Container Platform

RED HAT® OPENSHIFT® Application Runtimes

Eclipse Che · spring boot · Jenkins · Gogs · Sonatype Nexus · Istio · JAEGER

AUTOMATION WITH ANSIBLE

THE ANSIBLE WAY

CROSS PLATFORM

Agentless support for all major OS variants, physical, virtual, cloud and network devices.

HUMAN READABLE

Perfectly describe and document every aspect of your application environment.

PERFECT DESCRIPTION OF APPLICATION

Every change can be made by Playbooks, ensuring everyone is on the same page.

VERSION CONTROLLED

Playbooks are plain-text. Treat them like code in your existing version control.

DYNAMIC INVENTORIES

Capture all the servers 100% of the time, regardless of infrastructure, location, etc.

ORCHESTRATION PLAYS WELL WITH OTHERS

Every change can be made by Playbooks, ensuring everyone is on the same page.

WHAT CAN I DO WITH ANSIBLE?

Automate the deployment and management of your entire IT footprint.

Do this...

Orchestration	Configuration Management	Application Deployment	Provisioning	Continuous Delivery	Security and Compliance

On these...

Firewalls	Load Balancers	Applications	Containers	Clouds
Servers	Infrastructure	Storage	Network Devices	**And more...**

ANSIBLE AUTOMATES TECHNOLOGIES YOU USE

Time to automate is measured in minutes

CLOUD

AWS
Azure
Digital Ocean
Google
OpenStack
Rackspace
+more

OPERATING SYSTEMS

RHEL and Linux
UNIX
Windows
+more

VIRT & CONTAINER

Docker
VMware
RHV
OpenStack
OpenShift
+more

STORAGE

NetApp
Red Hat Storage
Infinidat
+more

WINDOWS

ACLs
Files
Packages
IIS
Regedits
Shares
Services
Configs
Users
Domains
+more

NETWORK

Arista
A10
Cumulus
Bigswitch
Cisco
Cumulus
Dell
F5
Juniper
Palo Alto
OpenSwitch
+more

DEVOPS

Jira
GitHub
Vagrant
Jenkins
Bamboo
Atlassian
Subversion
Slack
Hipchat
+more

MONITORING

Dynatrace
Airbrake
BigPanda
Datadog
LogicMonitor
Nagios
New Relic
PagerDuty
Sensu
StackDriver
Zabbix
+more

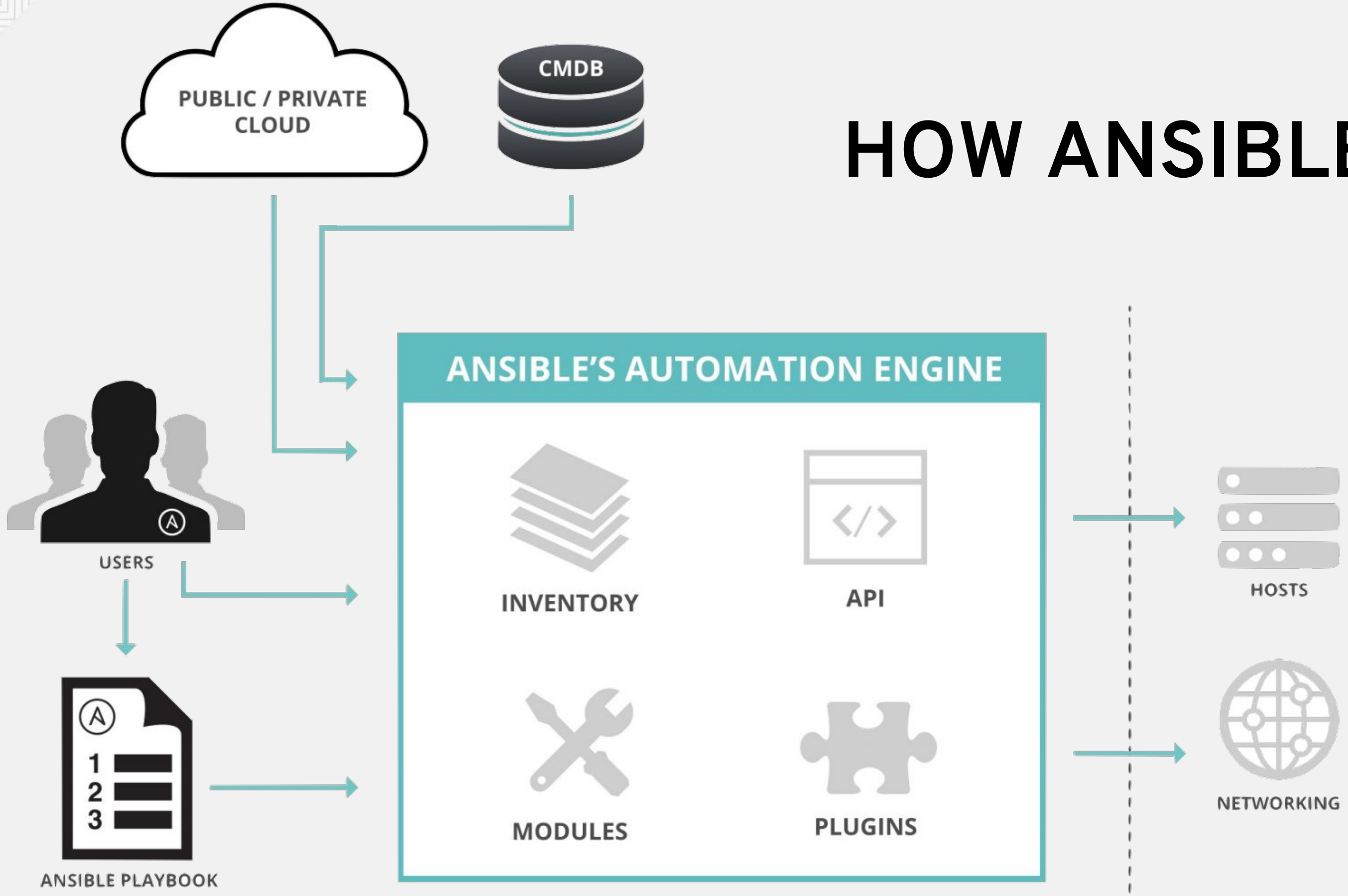
redhat.

QUICK REVIEW

Ansible

- is an automation platform mainly for administration
- is Agentless
- is written in Python
- manages UNIX, Linux, Network Devices, and Windows
- runs on any Linux distribution
- uses SSH (or WinRM) for connecting to hosts
- uses YAML for playbooks and roles
- Includes hundreds of modules out of the box

HOW ANSIBLE WORKS



INVENTORY

Ansible's list of hosts it works on is called its *inventory*.

Default inventory file is at `/etc/ansible/hosts`

To not use the default inventory pass the **`-i`** flag on the command line. This includes how to dynamically pull inventory using cobbler scripts from cloud providers.

This would ping all hosts in the `us-east-1d` region on AWS:

```
$ ansible -i ec2.py -u ubuntu us-east-1d -m ping
```

TASKS

Tasks are the application of a module to perform a specific unit of work.

- **file:** A directory should exist
- **yum:** A package should be installed
- **service:** A service should be running
- **template:** Render a configuration file from a template
- **get_url:** Fetch an archive file from a URL
- **git:** Clone a source code repository

AD-HOC COMMANDS

Ansible can run on off commands against any host or group in its inventory.

This would ensure nginx is installed and the latest version on webservers in your inventory:

```
$ ansible webservers -m yum -a "name=nginx state=latest"
```

PLAYBOOKS

A *playbook* is the primary way someone will use Ansible.

A playbook is a file contain one or more plays and is written in YAML. Most playbook will only contain a single play.

A play contains one or more tasks.

A task is a single execution of a module to do one thing to the hosts that are targeted.

PLAYBOOK EXAMPLE

```yaml
---
- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

  tasks:
  - name: httpd package is present
    yum:
      name: httpd
      state: latest

  - name: latest index.html file is present
    copy:
      src: files/index.html
      dest: /var/www/html/

  - name: httpd is started
    service:
      name: httpd
       state: started
```

PLAYBOOK EXAMPLE: NETWORK AUTOMATION

```
---
- name: configure ios interface
  hosts: ios01
  tasks:
    - name: collect device running-config
      ios_command:
        commands: show running-config interface GigabitEthernet0/2
        provider: “{{ cli }}”
      register: config

    - name: administratively enable interface
      ios_config:
        lines: no shutdown
        parents: interface GigabitEthernet0/2
        provider: “{{ cli }}”
      when: ‘”shutdown” in config.stdout[0]‘

    - name: verify operational status
      ios_command:
        commands:
            - show interfaces GigabitEthernet0/2
            - show cdp neighbors GigabitEthernet0/2 detail
        waitfor:
            - result[0] contains ‘line protocol is up’
            - result[1] contains ‘iosxr03’
            - result[1] contains ’10.0.0.42’
        provider: “{{ cli }}”
```

PLAYBOOK EXAMPLE: WINDOWS

```
- hosts: new_servers
  tasks:
  - name: ensure common OS updates are current
    win_updates:
    register: update_result

  - name: ensure domain membership
    win_domain_membership:
      dns_domain_name: contoso.corp
      domain_admin_user: '{{ domain_admin_username }}'
      domain_admin_password: '{{ domain_admin_password }}'
      state: domain
    register: domain_result

  - name: reboot and wait for host if updates or domain change require it
    win_reboot:
    when: update_result.reboot_required or domain_result.reboot_required

  - name: ensure local admin account exists
    win_user:
      name: localadmin
      password: '{{ local_admin_password }}'
      groups: Administrators

  - name: ensure common tools are installed
    win_chocolatey:
      name: '{{ item }}'
    with_items: ['sysinternals', 'googlechrome']
```

INCLUDES AND ROLES

To reuse a set of tasks you can:

- Import a playbook containing generic tasks
- Create a role that is much more flexible and reusable

Roles are better as they have a set structure and can include modules. They are the preferred way to distribute to a wider audience, like a vendor bundling management features.

ANSIBLE GALAXY

GALAXY

15,000 ROLES AT YOUR DISPOSAL

Reusable Roles and Container Apps that allow you to do more, faster

Built into the Ansible CLI and Tower

galaxy.ansible.com

GETTING STARTED

Have you used Ansible already?

Try Tower for free: **ansible.com/tower-trial**

Would you like to learn Ansible?

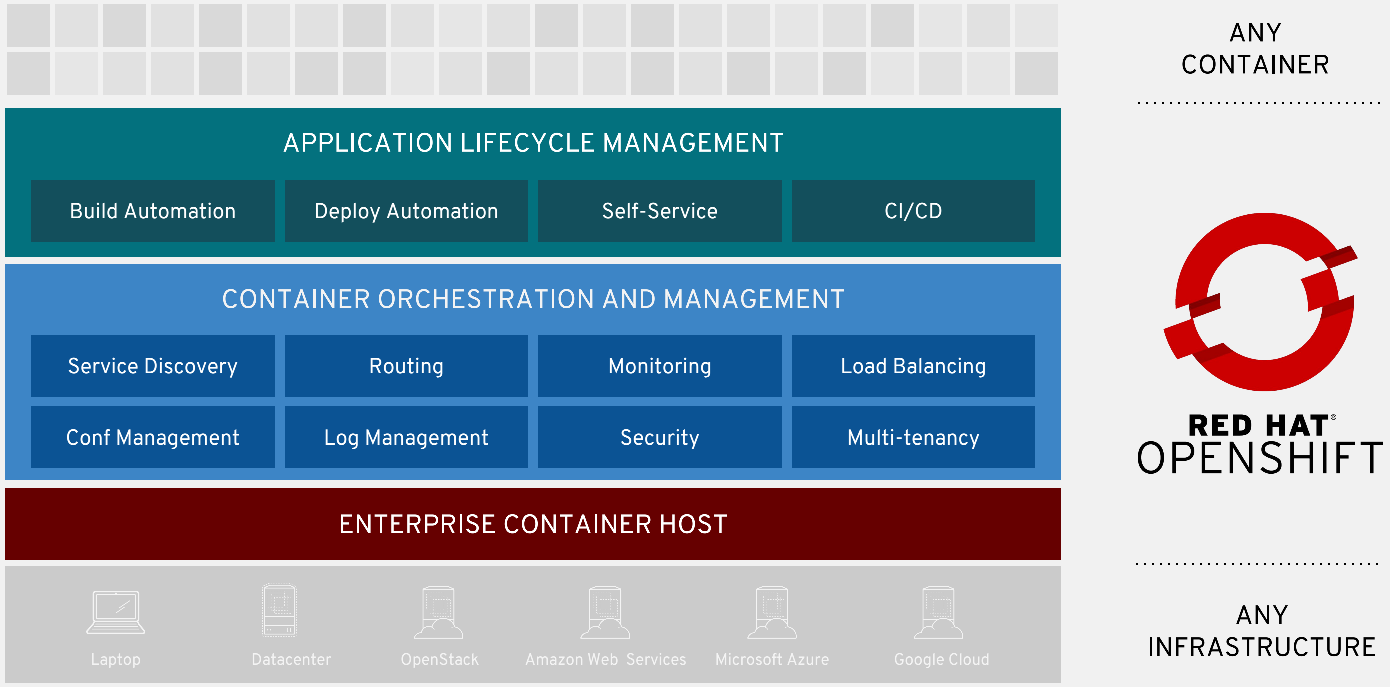
It’s easy to get started: **ansible.com/get-started**

Want to learn more?

Videos, webinars, case studies, whitepapers: **ansible.com/resources**

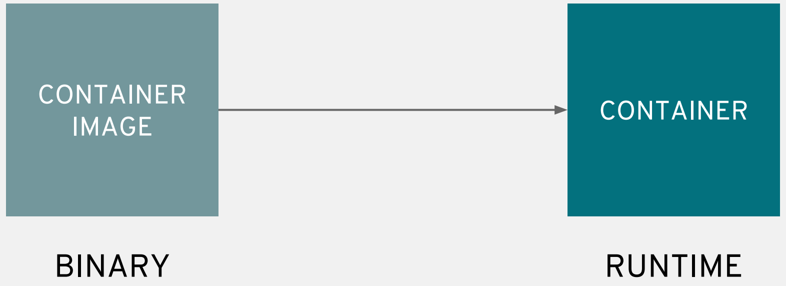
CLOUD NATIVE APP DEV WITH OPENSHIFT

CLOUD-NATIVE CAPABILITIES WITH RED HAT OPENSHIFT

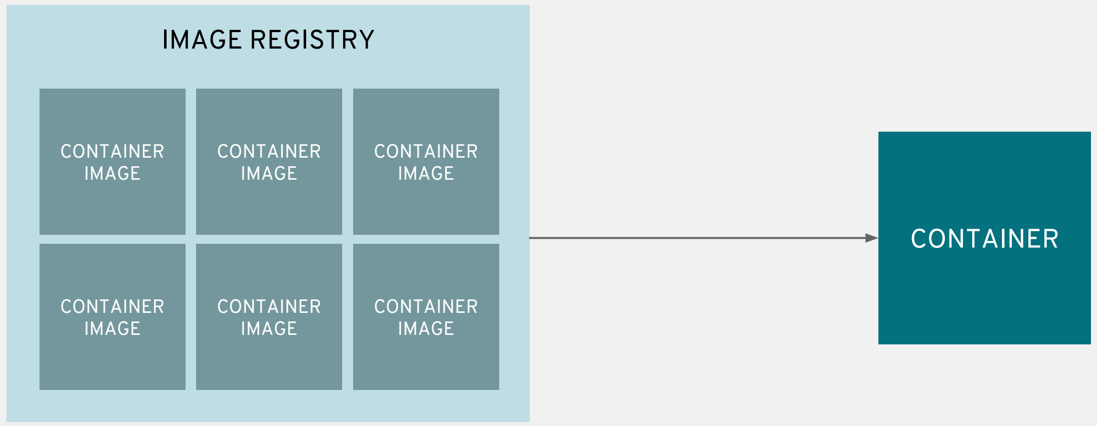


A container is the smallest compute unit

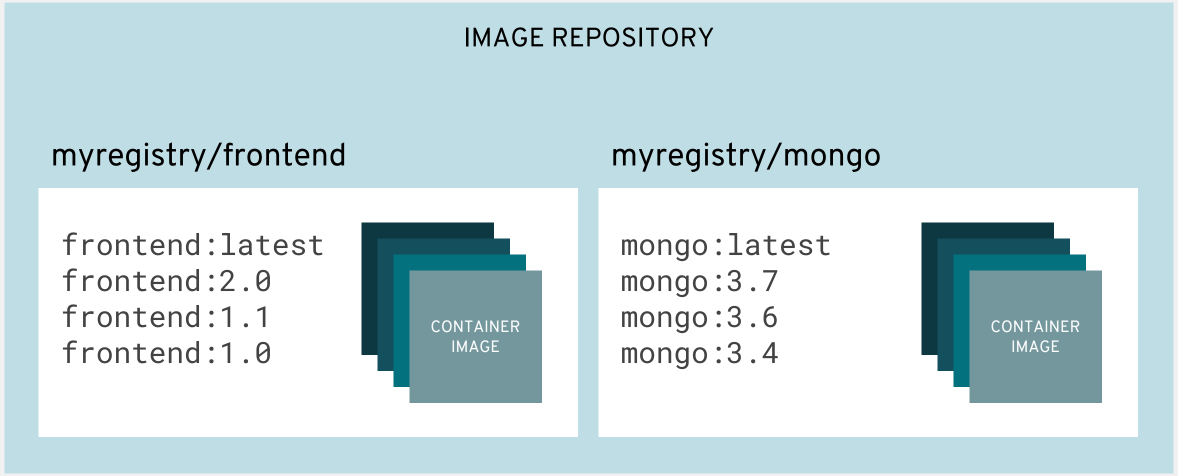
containers are created from container images during a build



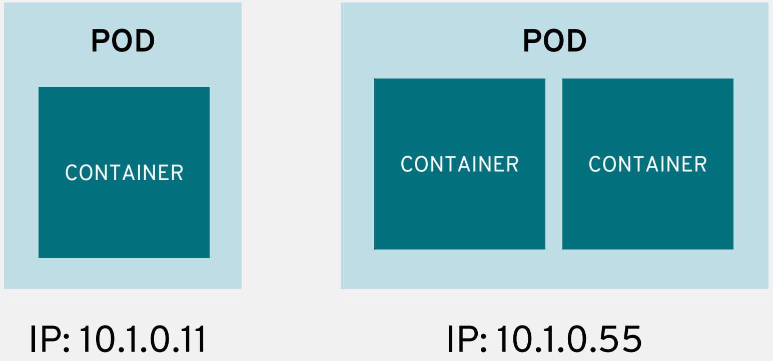
container images are stored in an image registry



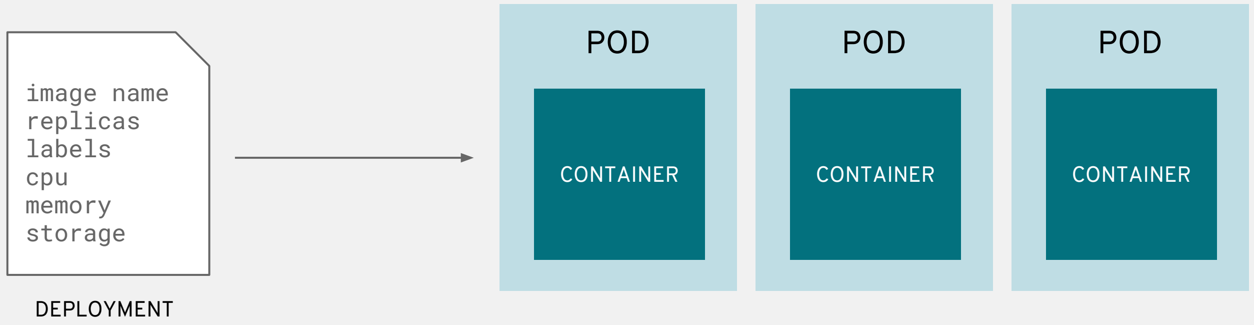
an image repository contains all versions of an image in the image registry



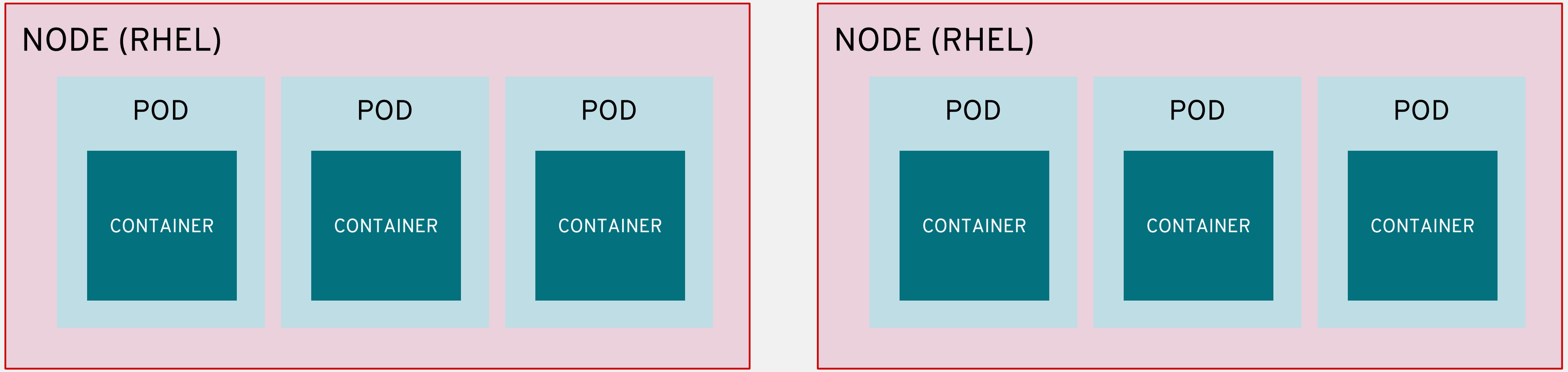
containers are wrapped in pods which are units of deployment and management, and share a common network address



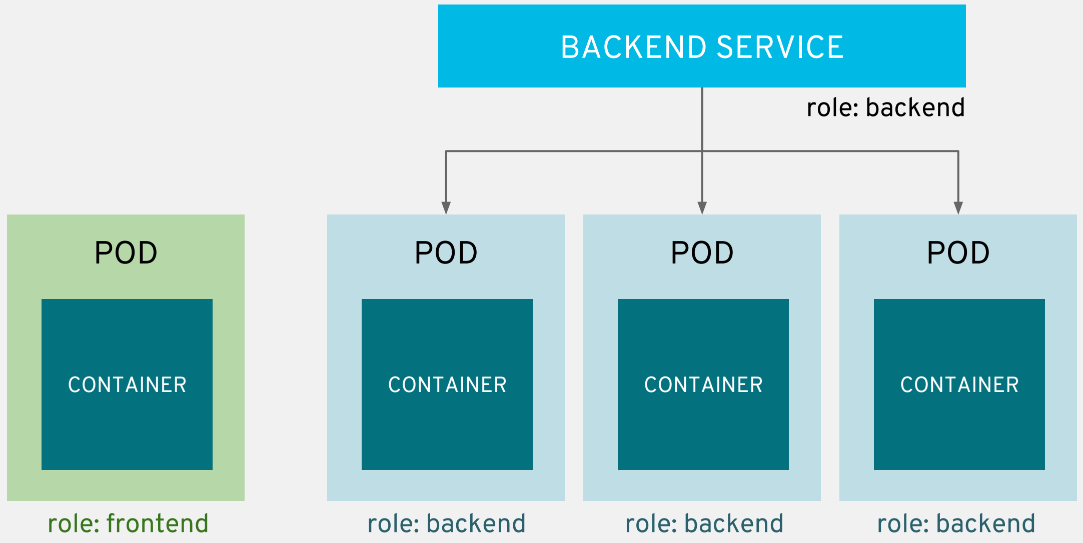
pods configuration is defined in a `deployment`



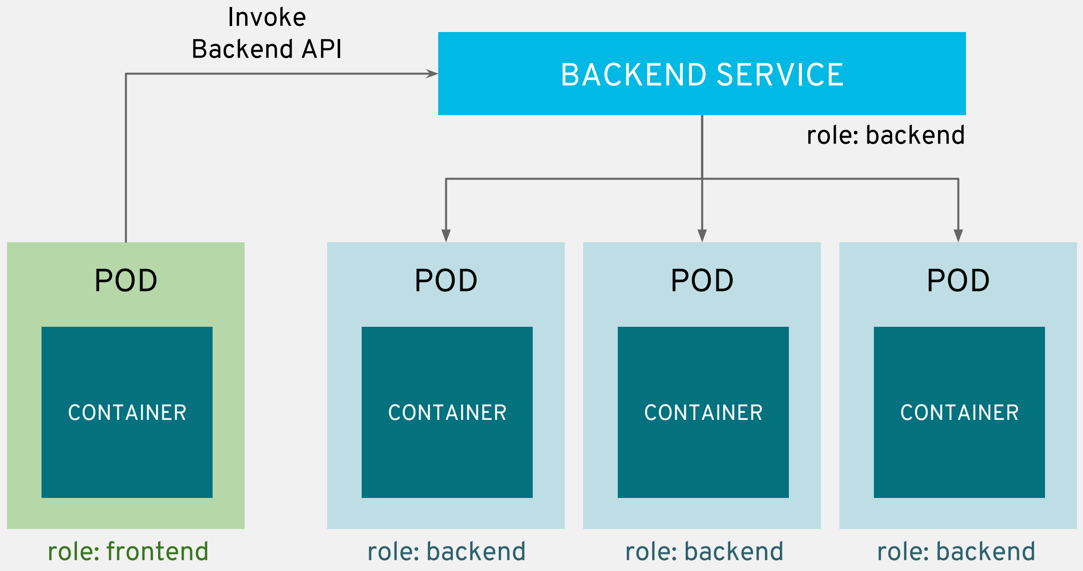
pods are deployed to and run on nodes



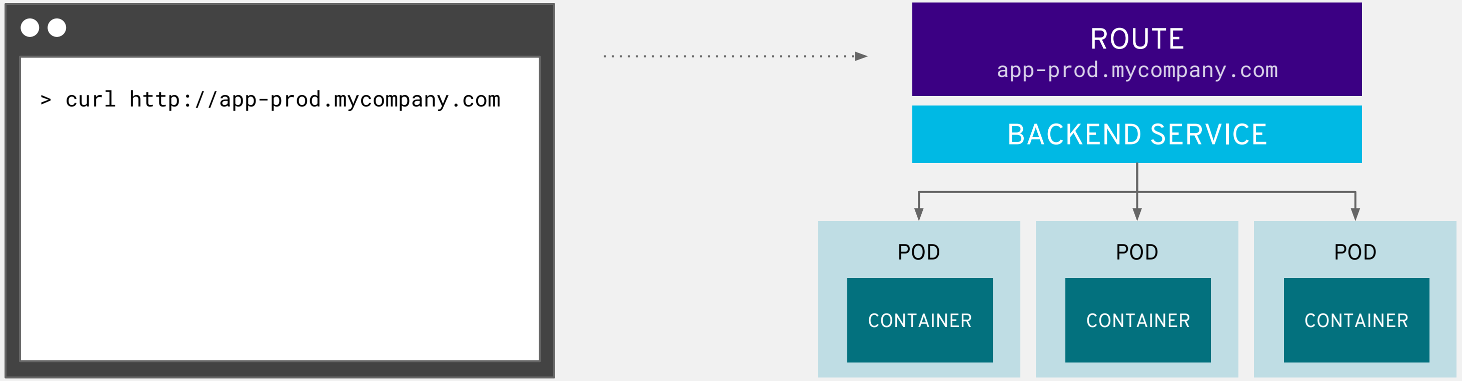
services provide internal load-balancing and service discovery across pods



apps can talk to each other via services

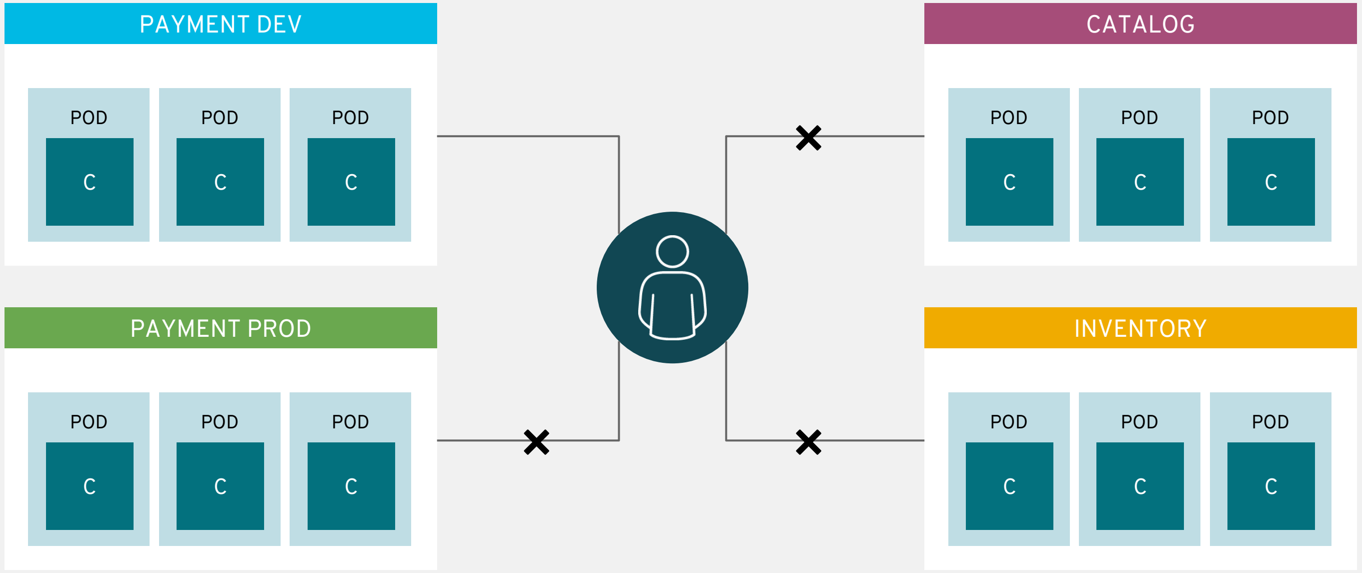


routes add services to the external load-balancer and provide readable urls for the app

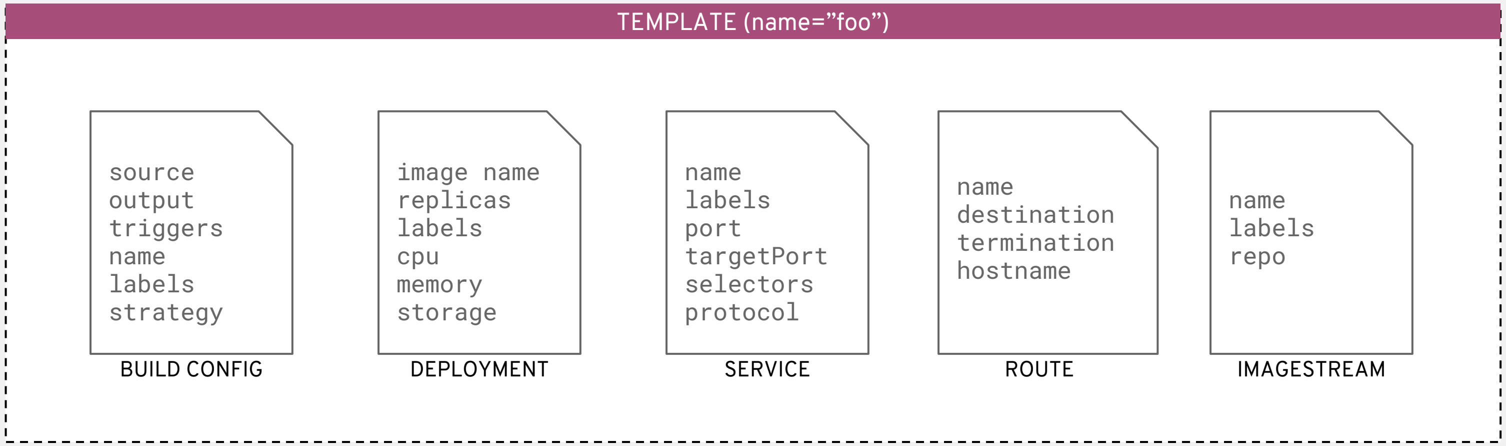


```
> curl http://app-prod.mycompany.com
```

projects isolate apps across environments, teams, groups and departments



templates define a blueprint for an application that can be instantiated within a project

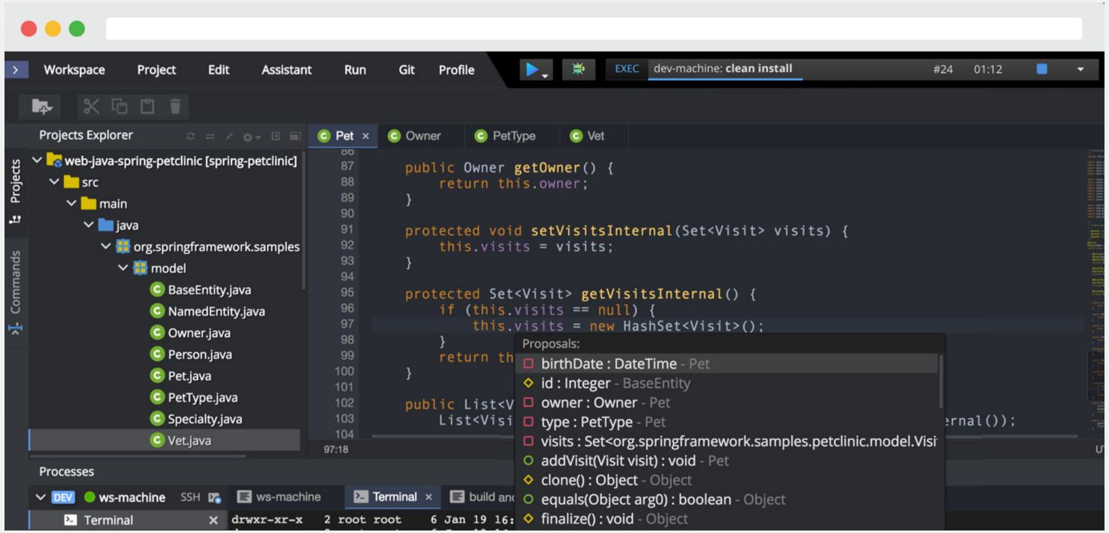


```
$ oc new-app foo
```

ECLIPSE CHE

Eclipse Che

An OpenShift-native developer workspace server and web IDE that accelerates projects on-premises or in the cloud



```
    public Owner getOwner() {
        return this.owner;
    }

    protected void setVisitsInternal(Set<Visit> visits) {
        this.visits = visits;
    }

    protected Set<Visit> getVisitsInternal() {
        if (this.visits == null) {
            this.visits = new HashSet<Visit>();
        }
        return th
    }

    public List<V
        List<Visi                                                    ernal());
```

```
Proposals:
birthDate : DateTime - Pet
id : Integer - BaseEntity
owner : Owner - Pet
type : PetType - Pet
visits : Set<org.springframework.samples.petclinic.model.Visi
addVisit(Visit visit) : void - Pet
clone() : Object - Object
equals(Object arg0) : boolean - Object
finalize() : void - Object
```

```
drwxr-xr-x    2 root root     6 Jan 19 16:
```

redhat.

Microservices with Spring Boot

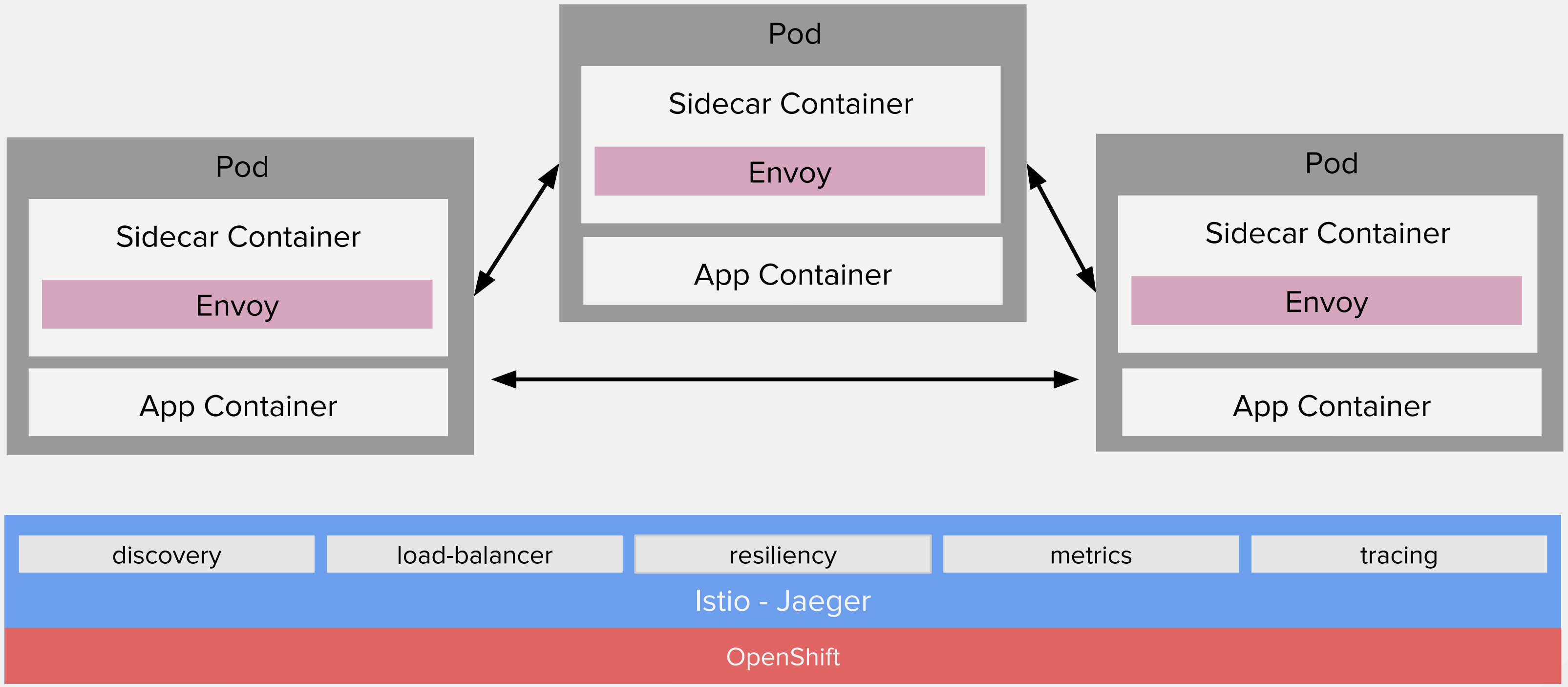
- Explore Spring Boot Maven project
- Create a domain model
- Create a RESTful service
- Run Spring Boot locally
- Deploy Spring Boot on OpenShift

OPENSHIFT PIPELINES

- CI/CD workflow via **Jenkins**
- Pipelines are started, monitored, and managed similar to other builds
- Auto-provisioning of Jenkins server
- On-demand Jenkins slaves
- Embedded Jenkinsfile or in Git repo

```
pipeline {
  agent {
    label 'maven'
  }
  stages {
    stage('build app') {
      steps {
        git url: 'https://git/app.git'
        sh "mvn package"
      }
    }
    stage('build image') {
      steps {
        script {
          openshift.withCluster() {
            openshift.startBuild("...")
          }
        }
      }
    }
  }
}
```

SERVICE MESH WITH ISTIO



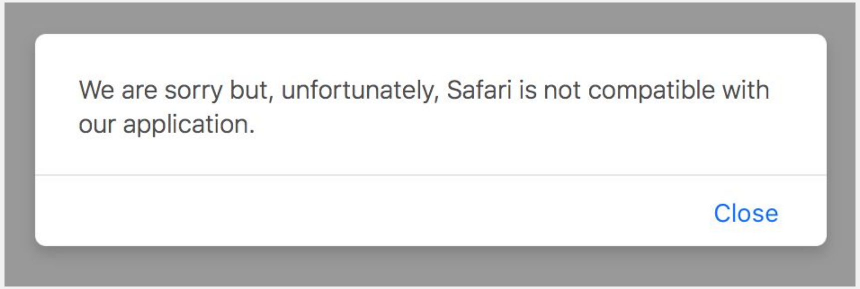
LET’S GET STARTED

WHAT DO YOU NEED?

A modern/recent web browser:

- Chrome
- Firefox
- Safari*
- IE/Edge*
- Opera*

If you get this...



We are sorry but, unfortunately, Safari is not compatible with our application.

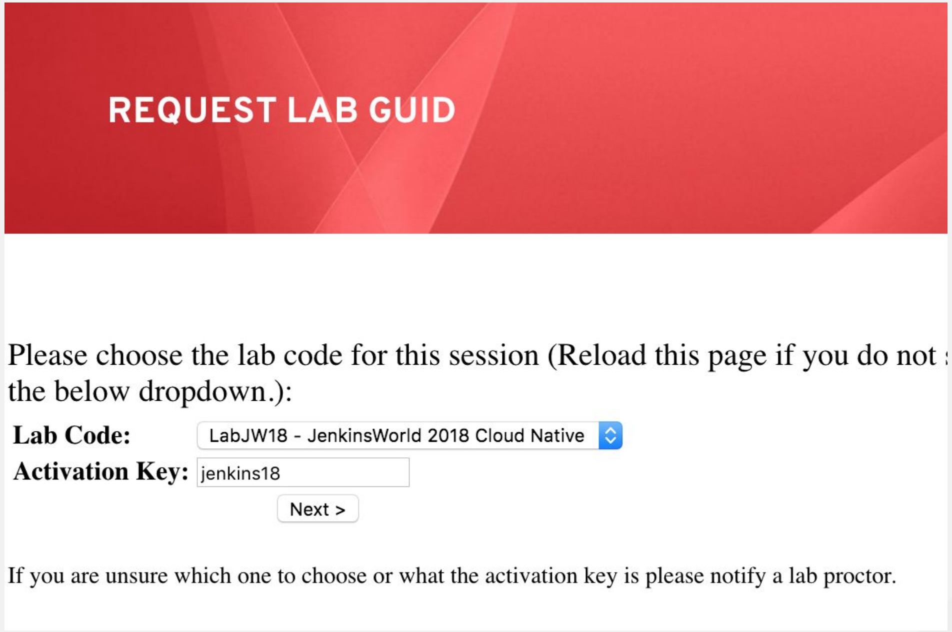
... Use Firefox or Chrome to grab a GUID, then any browser to continue

GET YOUR PERSONAL GUID

1. Grab a GUID at:
http://bit.ly/jw18-lab

2. Lab Code (drop-down):
JenkinsWorld 2018 Cloud Native

3. Activation Key
jenkins18

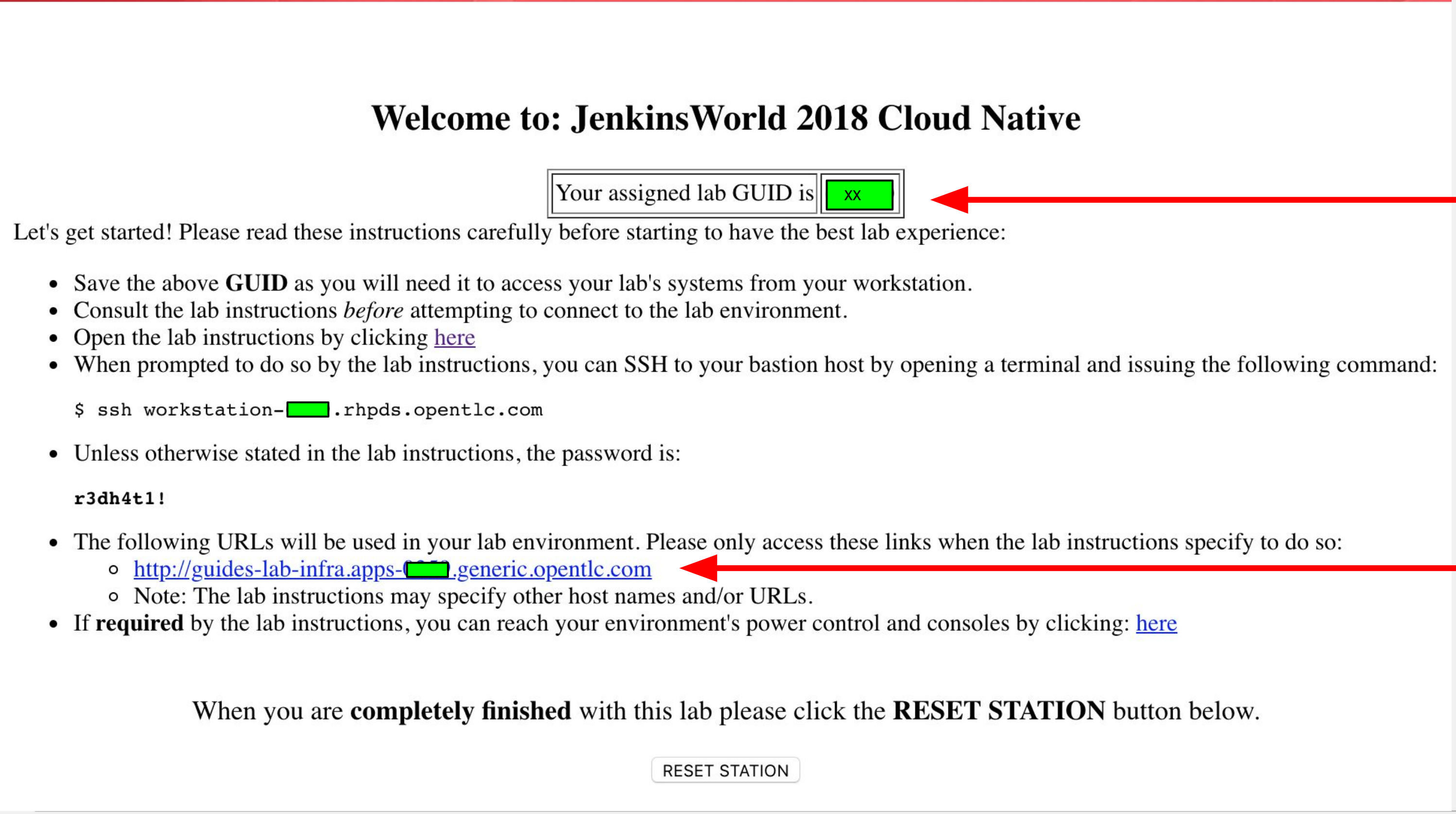


Please choose the lab code for this session (Reload this page if you do not
the below dropdown.):

Lab Code:	LabJW18 - JenkinsWorld 2018 Cloud Native
Activation Key:	jenkins18

If you are unsure which one to choose or what the activation key is please notify a lab proctor.

LAB INFORMATION



Welcome to: JenkinsWorld 2018 Cloud Native

Your assigned lab GUID is xx

Let's get started! Please read these instructions carefully before starting to have the best lab experience:

- Save the above **GUID** as you will need it to access your lab's systems from your workstation.
- Consult the lab instructions *before* attempting to connect to the lab environment.
- Open the lab instructions by clicking here
- When prompted to do so by the lab instructions, you can SSH to your bastion host by opening a terminal and issuing the following command:

```
$ ssh workstation-[redacted].rhpds.opentlc.com
```

- Unless otherwise stated in the lab instructions, the password is:

r3dh4t1!

- The following URLs will be used in your lab environment. Please only access these links when the lab instructions specify to do so:
  - http://guides-lab-infra.apps-[redacted].generic.opentlc.com
  - Note: The lab instructions may specify other host names and/or URLs.
- If **required** by the lab instructions, you can reach your environment's power control and consoles by clicking: here

When you are **completely finished** with this lab please click the **RESET STATION** button below.

RESET STATION

YOUR GUID

(DIFFERENT FOR EACH PERSON)

LAB GUIDE

CLICK AND BOOKMARK TO GET STARTED

WRAP-UP

WHAT YOU LEARNED

- Ansible Basics
  - Ansible Concepts
  - How to develop and run Playbooks for automating infrastructure deployment based on OpenShift (1 hour)
- Cloud Native Application Development
  - Bootstrapping Development Environment
  - Developing Microservices with Spring Boot
  - Automating Production Releases
  - Distributed Tracing and Fault Tolerance with Jaeger & Istio

RED HAT® ANSIBLE® Automation

RED HAT® OPENSHIFT® Container Platform

RED HAT® OPENSHIFT® Application Runtimes

Eclipse Che

spring boot

Jenkins

Gogs

Sonatype Nexus

Istio

JAEGER

redhat.

LAB SOURCE CODE

Lab instructions (Markdown) & Ansible Playbooks

Address: https://github.com/jamesfalkner/jw18-lab

Render the labs:

```
$ docker run -it -p 8080:8080 -v $(pwd):/app-data \
       -e CONTENT_URL_PREFIX="file:///app-data" \
       -e WORKSHOPS_URLS="file:///app-data/_rhsummit18.yml" \
       osevg/workshopper:latest
```

Source code to exercises:

Address:
https://github.com/openshift-labs/rhsummit18-cloudnative-labs