DISCLAIMER

• OpenShift has many MANY features that will not be discussed in this short presentation

• This presentation will focus on OpenShift features that pertain to source-to-deployment flows

• The OpenShift documentation and tutorials are a great place to learn more
  - https://docs.openshift.org/latest/
  - https://www.youtube.com/user/rhopenshift
Kubernetes is an open source system for managing containerized applications across multiple hosts, providing basic mechanisms for deployment, maintenance, and scaling of applications.
KUBERNETES

- Consists of two fundamental components
  - Resources (data)
    - Store the identity, metadata, and desired state of the elements in the system (spec)
  - Controllers (code loops)
    - Take action to move elements in the system toward their desired state (status)
RESOURCES

- APIVersion
  - v1, apps/v1beta1, etc
- Kind
  - Pod, ReplicaSet, Deployment, Service, etc
- Name
  - Uniquely identified a resource of a certain kind
- Labels
  - Resource selection mechanism
RESOURCES

apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
...

PODS

- Atom of scheduling
- One or more containers
- Containers share the network namespace
apiVersion: v1
type: Pod
metadata:
  labels:
    app: nginx
spec:
  containers:
    - name: nginx
      image: nginx:1.7.9
      ports:
        - containerPort: 80
DEPLOYMENTS

- Influenced by DeploymentConfigs in OpenShift
- Contains pod template
- Controller creates a ReplicaSet for each generation of the deployment
- Generation is incremented when deployment resource is modified (e.g. new image version) and controller conducts transition to new generation ReplicaSet
DEPLOYMENTS

```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
```

KUBERNETES

Great Foundation!
THE GAP

- Kubernetes Deployments work in terms of container images
- Developers work in terms of source code
THE GAP

HOW TO: DRAW A HORSE
BY VAN OKTOP

1. Draw 2 circles
2. Draw the legs
3. Draw the face
4. Draw the hair
5. Add small details.
THE GAP

Create a Dockerfile FROM ??? that installs the runtime for your application, inject your source code, pull dependencies, compile, build the image, tag the image, write the procedure on a napkin titled "The Official Build Process", test the image (it built, right? ship it!), push the image to a registry, make sure the Kubernetes nodes are configured to pull from that registry, then deploy your image in a Kubernetes deployment.

Repeat as necessary.
THE GAP

Source Code → Small Details

Deployment

ReplicaSet

Pod → Pod → Pod
BUILDCONFIGS

- The central resource of the OpenShift build system
- Uses source code and "build strategy" to produce an application image
- Application images are pushed to registry, triggering a deployment for any DeploymentConfig that has an ImageChange trigger on the updated image
- Builds can be automatically triggered in a number of ways
  - Webhooks (github, gitlab, bitbucket, generic)
  - ImageChange events (BuildConfig chaining)
SOURCE-TO-IMAGE (S2I)

- Language/runtime specific image that is transformed into the application image
- Two main scripts inside the image
  - assemble
    - Copies source into desired location
    - Builds the source (compile, download deps, etc)
  - run
    - Entrypoint for the application image

https://github.com/openshift/source-to-image
BUILD STRATEGY

- Source code is converted to an application image using a number of build strategies
  - Docker strategy
    - Use Dockerfile in base of source code repo
  - Source strategy
    - Use an S2I image
  - JenkinsPipeline
    - Use a Jenkins job
  - Custom strategy
    - Use a fully custom builder image
SOURCE-TO-IMAGE (S2I)

- Supported languages/runtimes
  - Java (Maven), Node.js, Perl, PHP, Python, Ruby, Wildfly (formerly JBoss AS)
- Developers can fork and customize the supported S2Is or create completely custom ones
IMAGESTREAMS

- Provides a watchable resource that mirrors the image registry metadata relevant to the project
- Enables ImageChange triggers for BuildConfigs and DeploymentConfigs
- Abstracts tags from the registry
  - Project can define the tag to image mapping
- Groups images by application
DEPLOYMENTCONFIGS

- Predecessor to Kubernetes Deployment resource
- Only really difference these days is DeploymentConfigs have ImageChange triggers
DEMO
GET STARTED

- Download client tools
  - https://github.com/openshift/origin/releases
- oc cluster up
QUESTIONS

https://github.com/openshift/origin

https://docs.openshift.org/latest