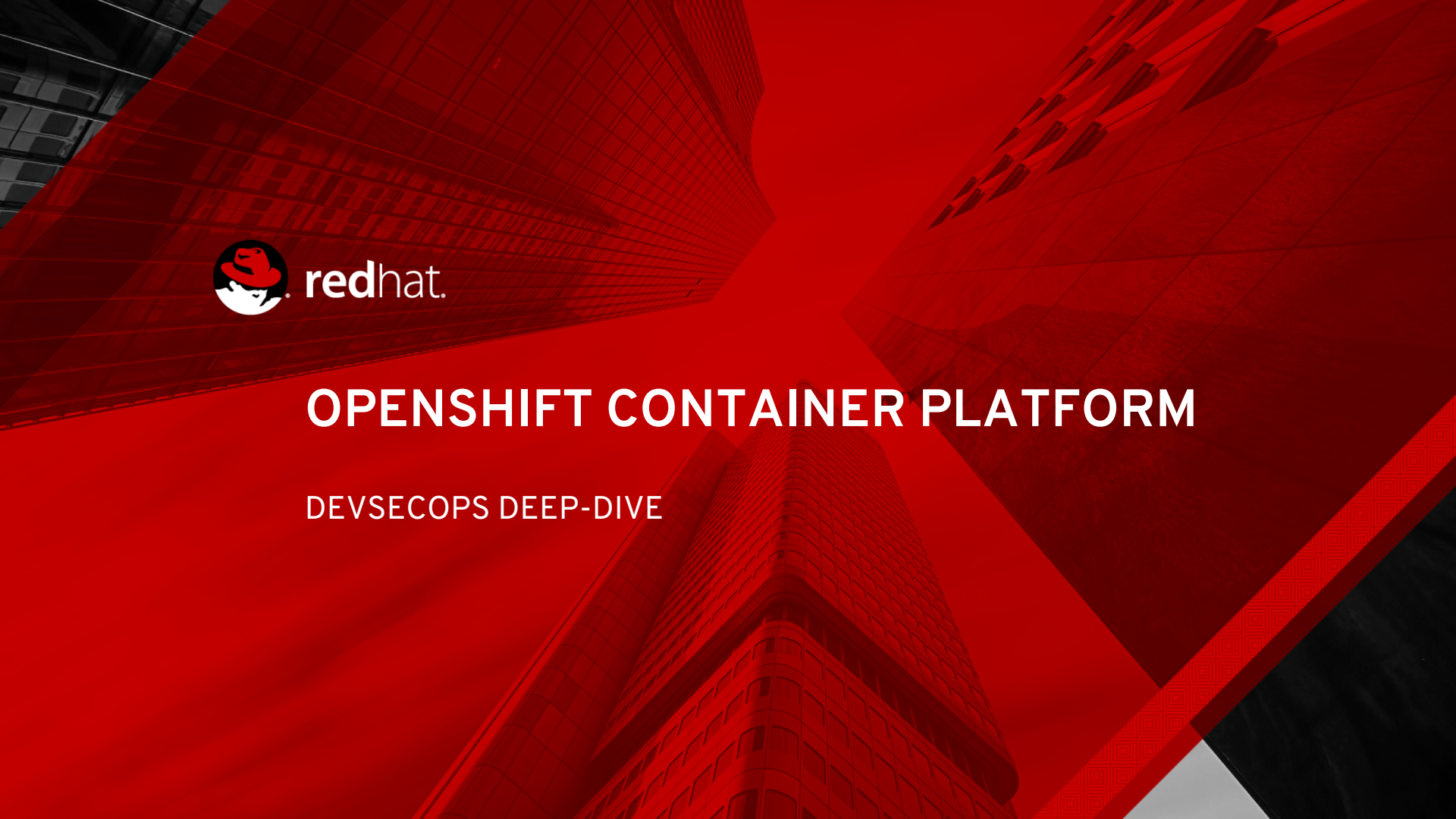
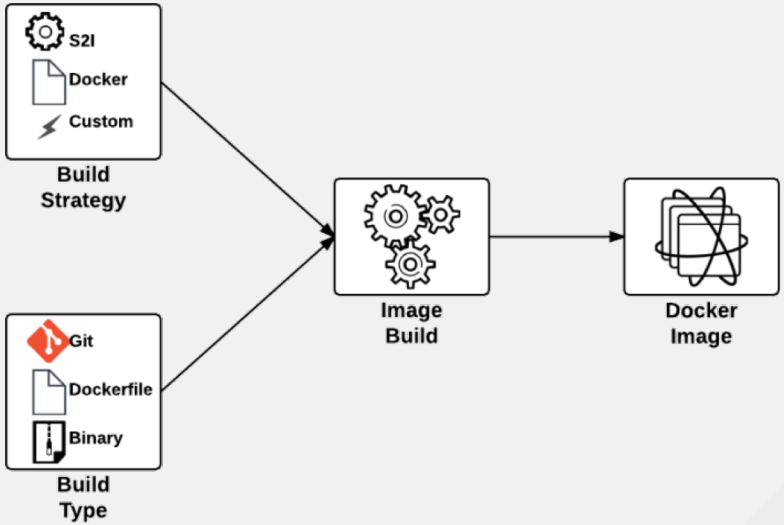


OPENSHIFT CONTAINER PLATFORM

DEVSECOPS DEEP-DIVE

OPENSHIFT BUILD STRATEGIES

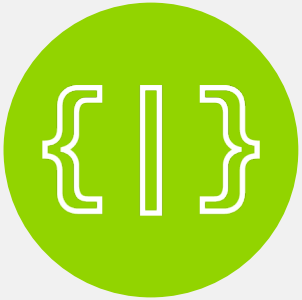
- Source: use source codes from git repository or Dockerfile as the build input
- Binary: Streaming content in binary format from a local file system to the builder
- Image: Additional files can be provides to the build process via images. Files will copy from source image to destination image.



OPENSHIFT BUILD STEPS INCLUDE

- Trigger a build in OpenShift
- Verify a build succeeded
- Trigger a deployment
- Scale a deployment up/down
- Verify a deployment succeeded
- Verify a service is accessible
- Tag an image
- Create Resource via YAML/JSON
- Delete any resource
- Cancel a build
- Cancel a deployment

BUILD AND DEPLOY CONTAINER IMAGES

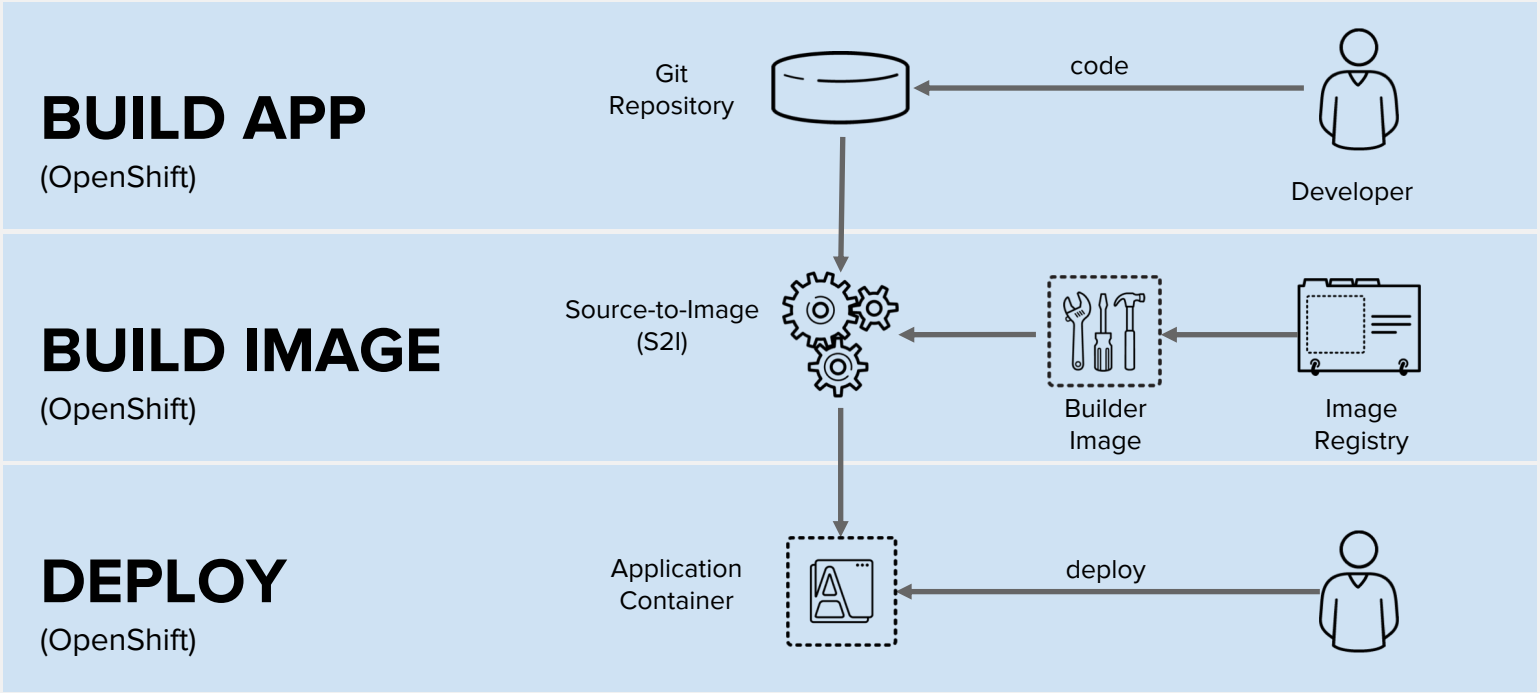


DEPLOY YOUR SOURCE CODE

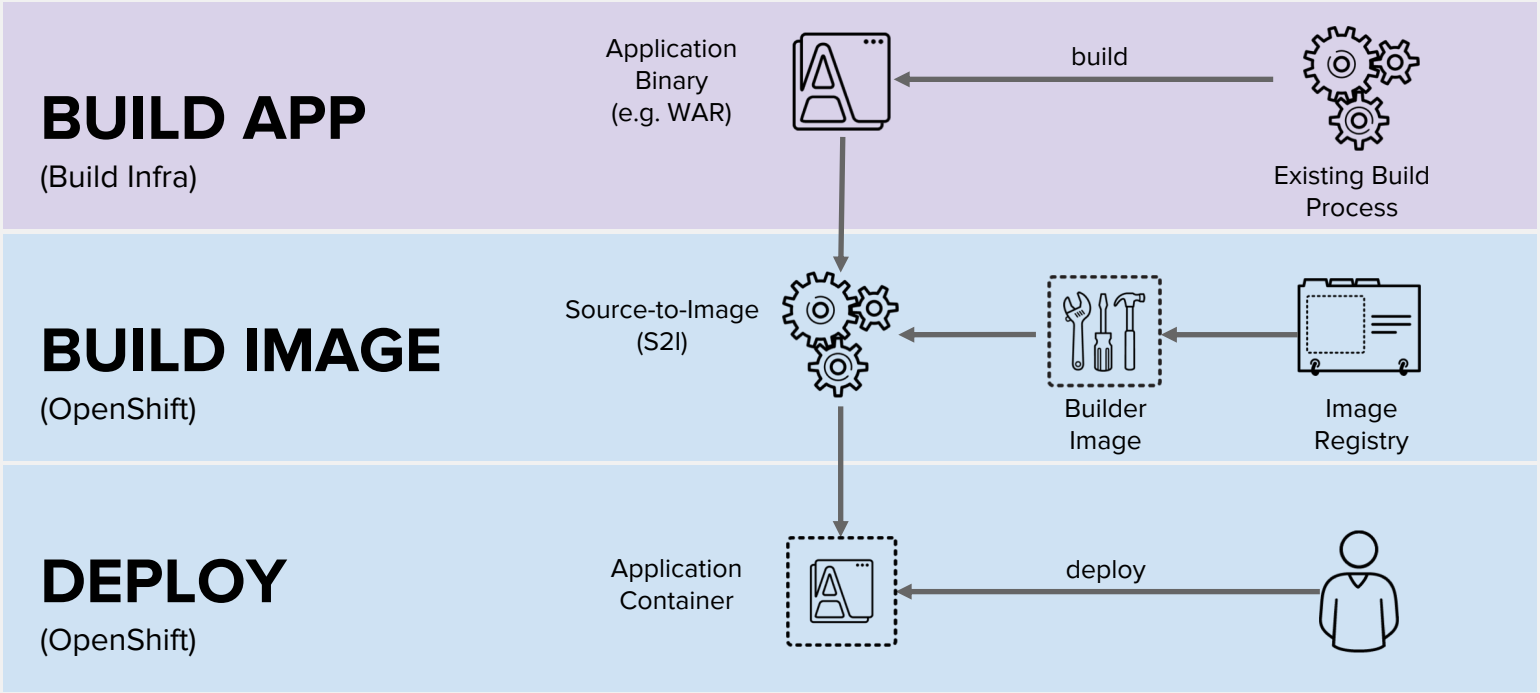
DEPLOY YOUR APP BINARY

DEPLOY YOUR CONTAINER IMAGE

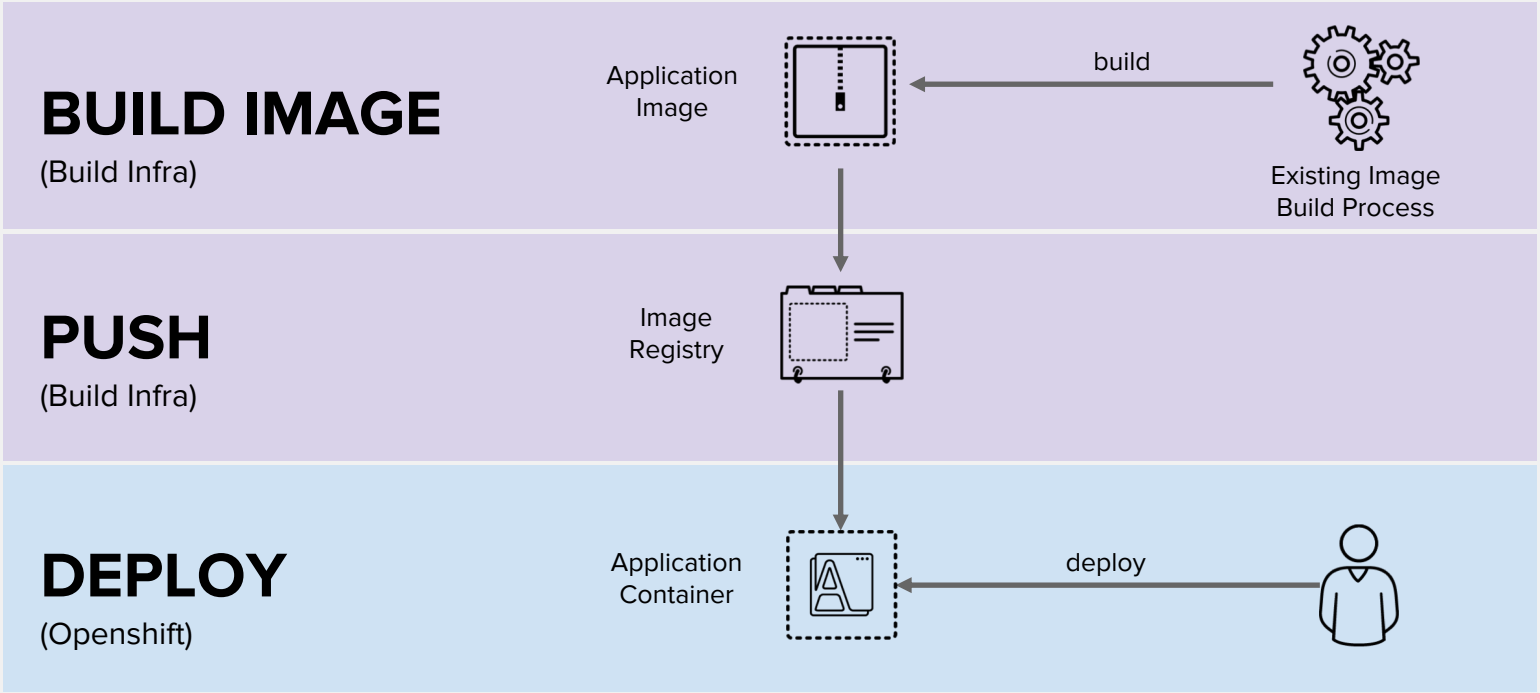
SOURCE CODE DEPLOYMENT



APP BINARY DEPLOYMENT

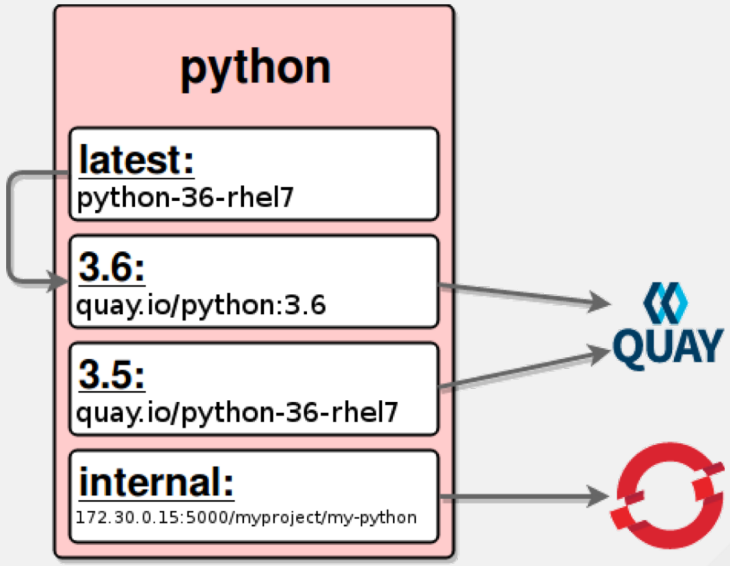


CONTAINER IMAGE DEPLOYMENT



WHAT ARE IMAGE STREAMS?

- Contains all of the metadata information about any given image that is specified in the Image Stream specification
- Does not contain the actual image data
- Ultimately points either to an external registry, e.g. registry.access.redhat.com, quay.io, OpenShift's internal, etc.



HOW ARE IMAGE STREAMS CREATED?

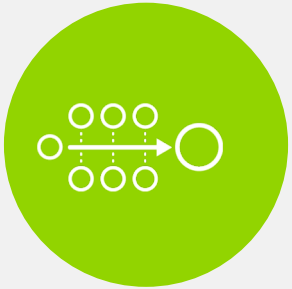
```
# Create a Image Stream named 'python' with a single tag pointing to 3.5
oc import-image python:3.5 --from=rhscl/python-35-rhel7 --confirm
```

- `python:3.5` - python is the new Image Stream that will be created as a result of this invocation. Additionally we are explicitly pointing that the imported image will be kept under the 3.5 Image Stream Tag of that Image Stream. If no tag part is specified the command will use latest.
- `--from=rhscl/python-35-rhel7` - states what external image the Image Stream Tag will point to.
- `--confirm` - informs the system that the python Image Stream should be created

OPENSHIFT LOVES CI/CD

JENKINS-AS-A SERVICE ON OPENSHIFT

HYBRID JENKINS INFRA WITH OPENSHIFT

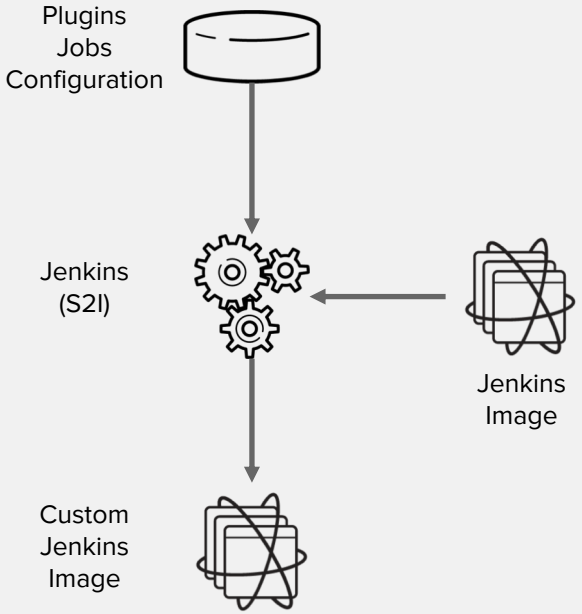


EXISTING CI/CD DEPLOY TO OPENSHIFT

JENKINS-AS-A-SERVICE ON OPENSHIFT

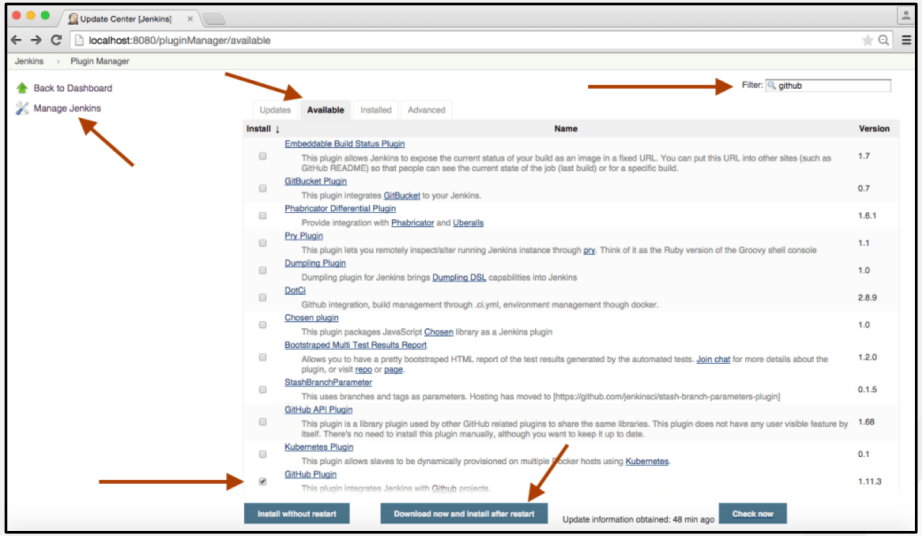
- Certified Jenkins images with pre-configured plugins
  - Provided out-of-the-box
  - Follows Jenkins 1.x and 2.x LTS versions

- Jenkins S2I Builder for customizing the image
  - Install plugins, configure Jenkins, configure build jobs

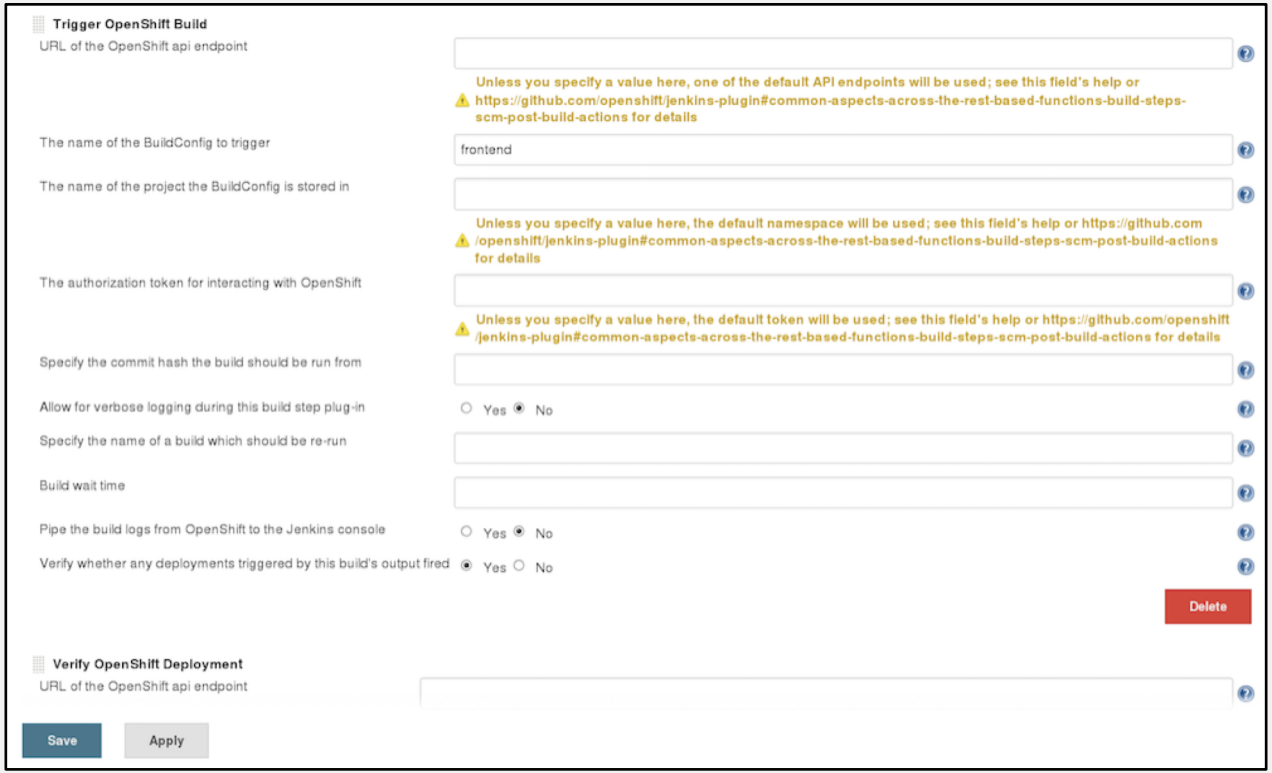


JENKINS PLUGIN

- The most fundamental part of a Pipeline
- Tell Jenkins *what* to do, and serve as the basic building block for both Declarative and Scripted Pipeline syntax

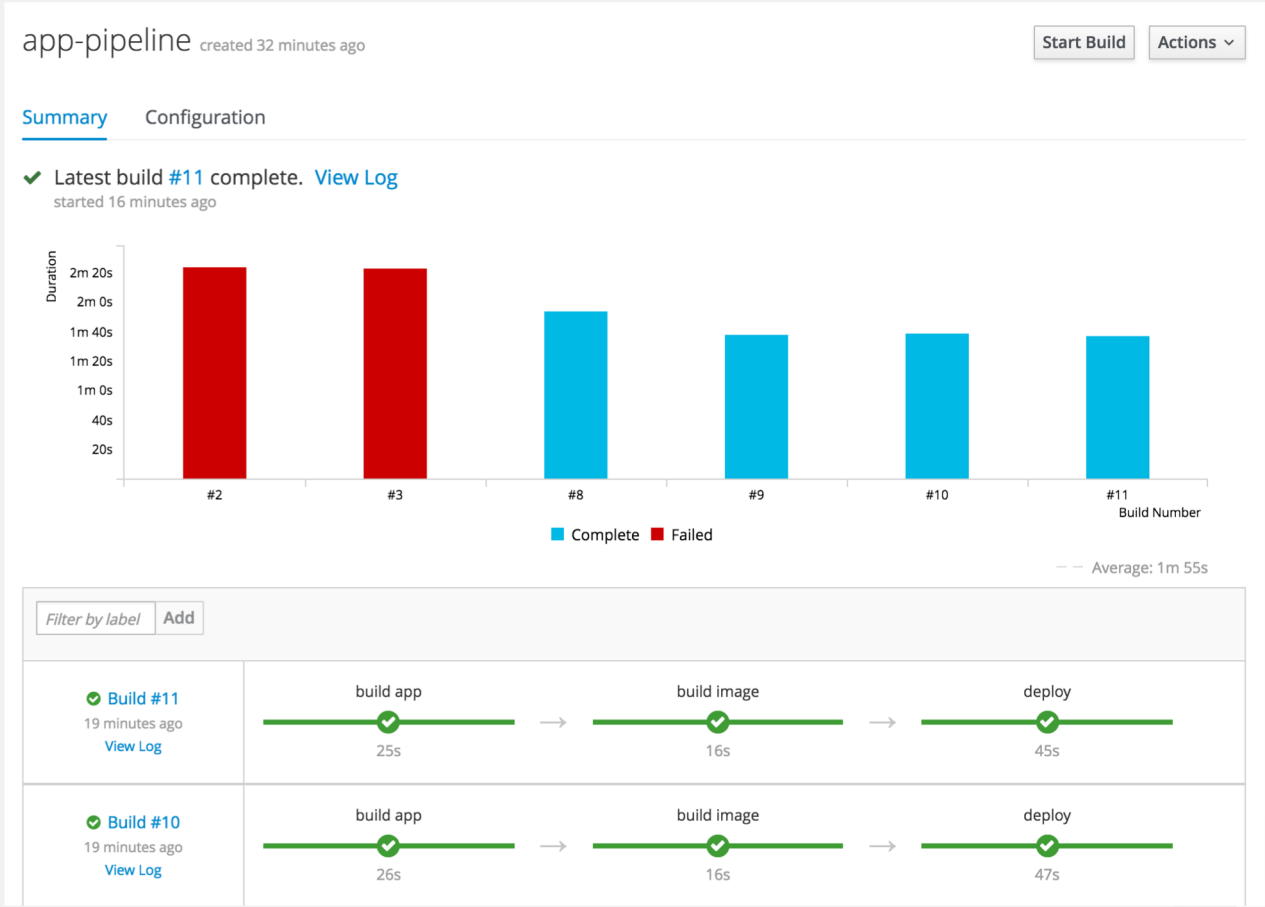


OPENSHIFT JENKINS PLUGIN



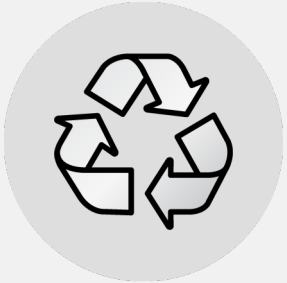
```
kind: BuildConfig
apiVersion: v1
metadata:
  name: sample-pipeline
  labels:
    Name: sample-pipeline
spec:
  triggers:
  - type: GitHub
    github:
      secret: secret101
  - type: Generic
    generic:
      secret: secret101
strategy:
  type: JenkinsPipeline
  jenkinsPipelineStrategy:
    jenkinsfile: |-
      node('maven') {
      stage 'build'
            openshiftBuild(buildConfig: 'ruby-sample-build', showBuildLogs:'true')
            stage 'deploy'
            openshiftDeploy(deploymentConfig: 'frontend')
      }
```

OPENSHIFT PIPELINES IN WEB CONSOLE



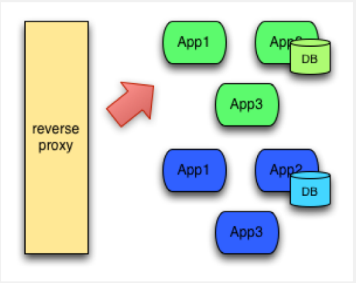
OPENSHIFT DEPLOYMENT STRATEGIES

Painless deployments with zero/reduced downtime through automation



ROLLING DEPLOYMENTS

A rolling deployment slowly replaces instances of the previous version of an application with instances of the new version of the application.



BLUE/GREEN DEPLOYMENTS

A blue/green deployment is a software deployment strategy that relies on two identical production configurations that alternate between active and inactive.

A/B DEPLOYMENTS

A/B testing (sometimes called split testing) is comparing two versions of a web page to see which one performs better.

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos