

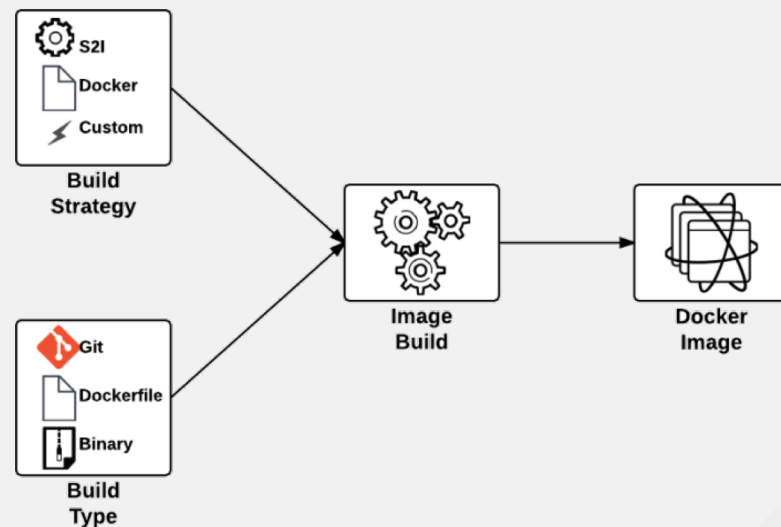


OPENSIFT CONTAINER PLATFORM

DEVSECOPS DEEP-DIVE

OPENSIFT BUILD STRATEGIES

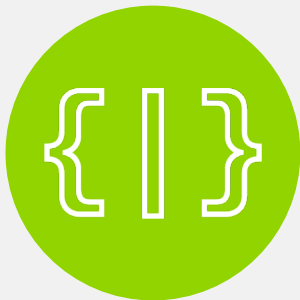
- Source: use source codes from git repository or Dockerfile as the build input
- Binary: Streaming content in binary format from a local file system to the builder
- Image: Additional files can be provides to the build process via images. Files will copy from source image to destination image.



OPENSIFT 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**

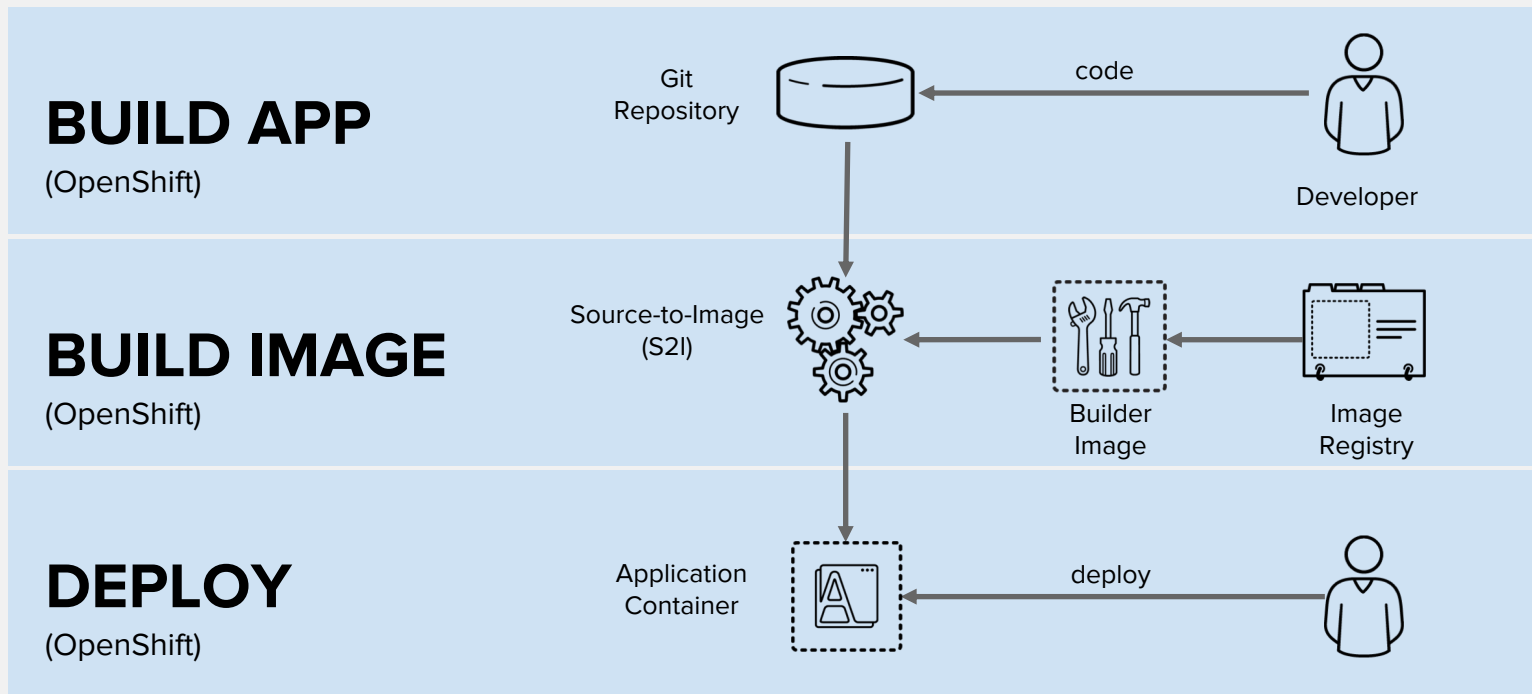


**DEPLOY YOUR
APP BINARY**

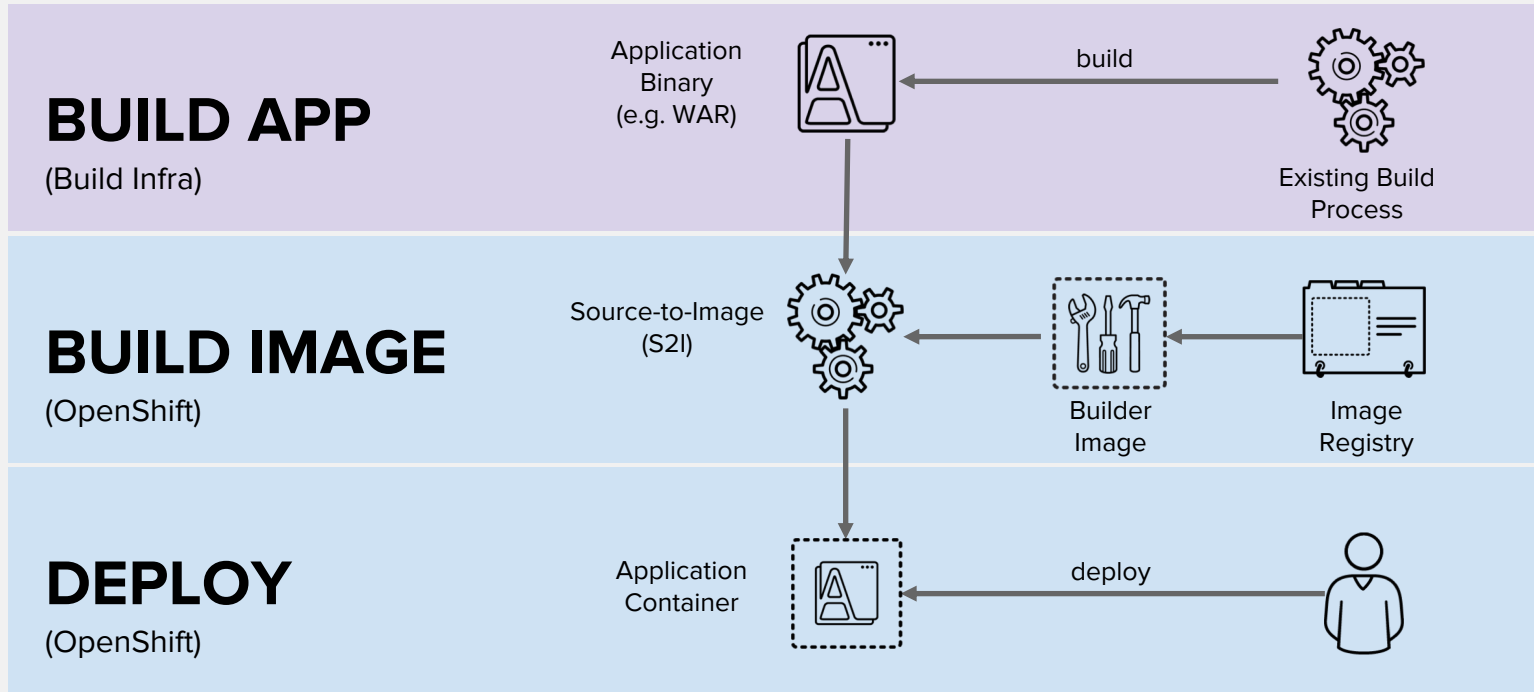


**DEPLOY YOUR
CONTAINER IMAGE**

SOURCE CODE DEPLOYMENT



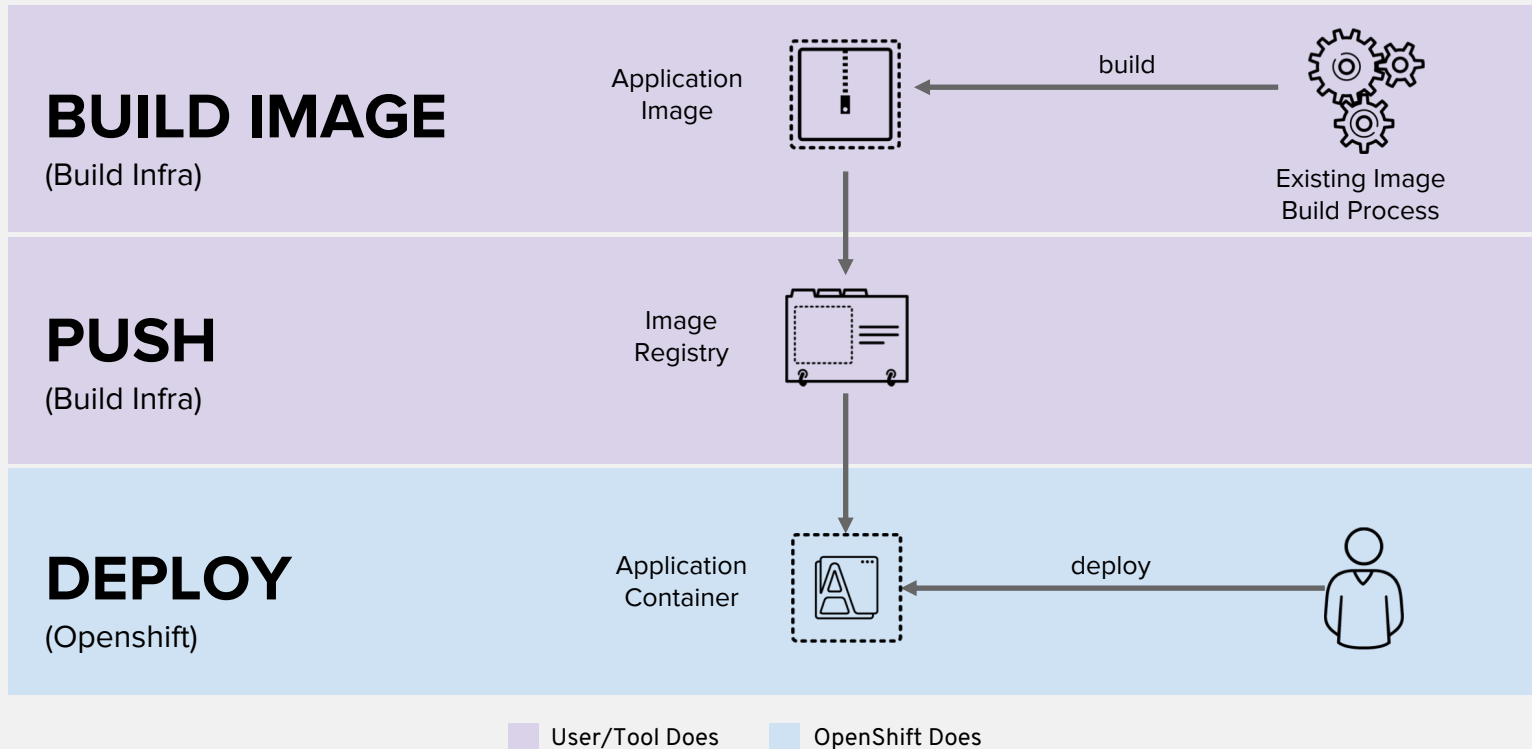
APP BINARY DEPLOYMENT



User/Tool Does

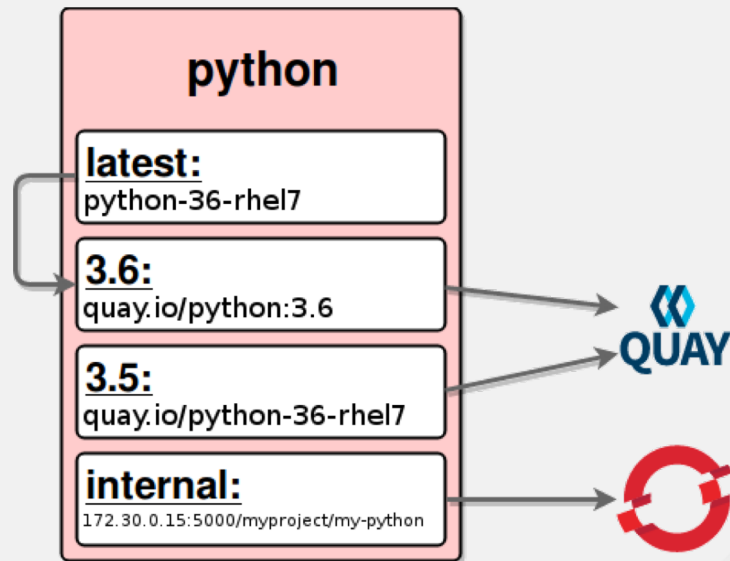
OpenShift Does

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=rhsc1/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=rhsc1/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

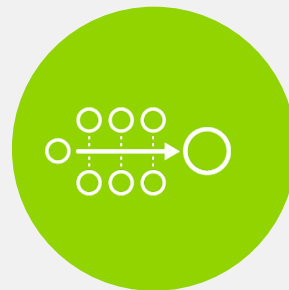
OPENSIFT LOVES CI/CD



**JENKINS-AS-A SERVICE
ON OPENSIFT**



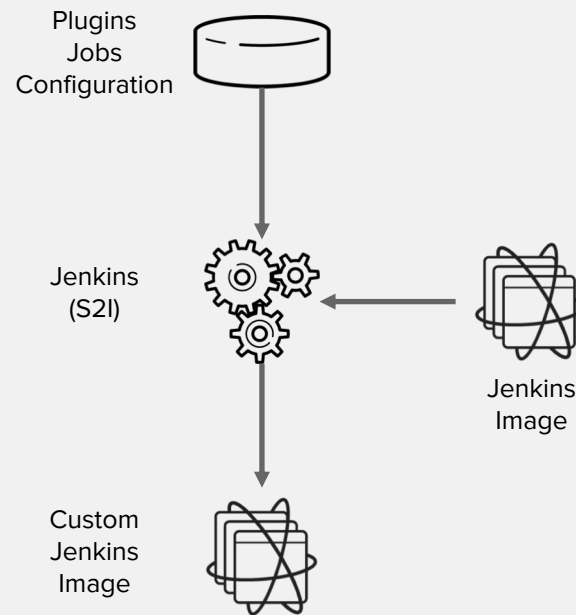
**HYBRID JENKINS INFRA
WITH OPENSIFT**



**EXISTING CI/CD
DEPLOY TO OPENSIFT**

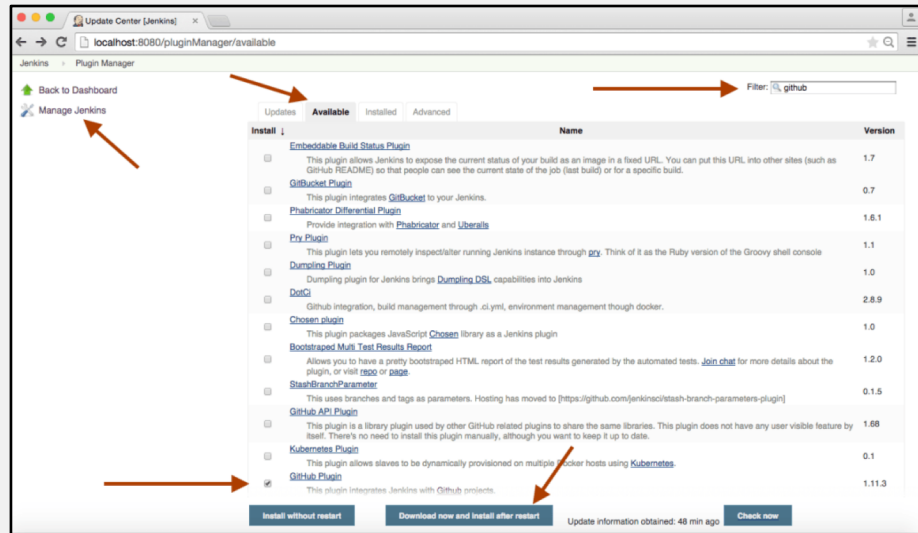
JENKINS-AS-A-SERVICE ON OPENSSHIFT

- 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



OPENSIFT JENKINS PLUGIN

Trigger OpenShift Build

URL of the OpenShift api endpoint

Unless you specify a value here, one of the default API endpoints will be used; see this field's help or <https://github.com/openshift/jenkins-plugin#common-aspects-across-the-rest-based-functions-build-steps-scm-post-build-actions> for details

The name of the BuildConfig to trigger

The name of the project the BuildConfig is stored in

Unless you specify a value here, the default namespace will be used; see this field's help or <https://github.com/openshift/jenkins-plugin#common-aspects-across-the-rest-based-functions-build-steps-scm-post-build-actions> for details

The authorization token for interacting with OpenShift

Unless you specify a value here, the default token will be used; see this field's help or <https://github.com/openshift/jenkins-plugin#common-aspects-across-the-rest-based-functions-build-steps-scm-post-build-actions> for details

Specify the commit hash the build should be run from

Allow for verbose logging during this build step plug-in Yes No

Specify the name of a build which should be re-run

Build wait time

Pipe the build logs from OpenShift to the Jenkins console Yes No

Verify whether any deployments triggered by this build's output fired Yes No

Verify OpenShift Deployment

URL of the OpenShift api endpoint

```
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')
      }
  }
```

OPENSIFT PIPELINES IN WEB CONSOLE

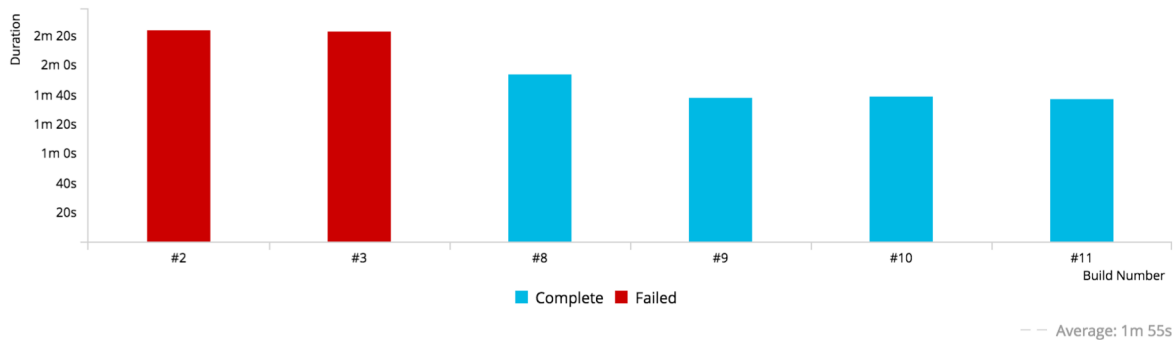
app-pipeline created 32 minutes ago

Start Build

Actions ▾

Summary Configuration

✓ Latest build #11 complete. [View Log](#)
started 16 minutes ago



Filter by label Add

✓ Build #11
19 minutes ago
[View Log](#)

build app



25s



build image



16s



deploy



45s

✓ Build #10
19 minutes ago
[View Log](#)

build app



26s



build image



16s



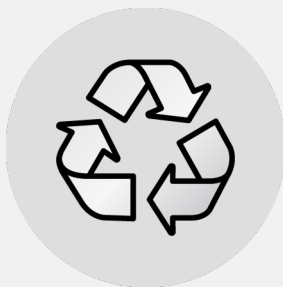
deploy



47s

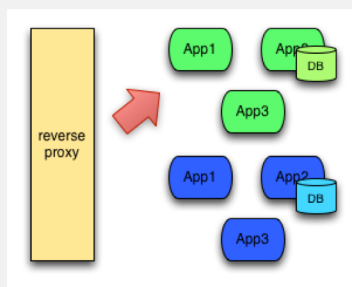
OPENSIFT DEPLOYMENT STRATEGIES

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.



THANK YOU



plus.google.com/+RedHat



facebook.com/redhatinc



linkedin.com/company/red-hat



twitter.com/RedHatNews



youtube.com/user/RedHatVideos