Cloud Native Development-Using IBM Cloud Pak for Applications

IBM Developer

IBM

OpenShift : Deploying to OpenShift

Developer Advocate
Teck talk : Mangesh Patankar

Agenda

- Quick Recap
- OpenShift – Deployments
- OpenShift deployment - As a set of layers
- Deployment
  - Deployment Config
  - History
  - Rollback
  - Deployment Strategies – Recreate, Rolling Upgrade
  - Advance Deployment Strategies - A/B, Blue Green
- Hands on

Recap

Session 1

What is Openshift ?

WebConsole

CLI

Resources Openshift

Cloud Pak for Applications (CP4A)

Session 2

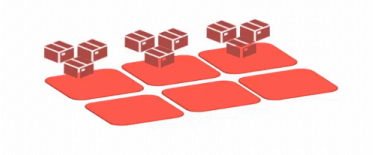
Cloud Native Application - Why, What and How?

Cloud Native Runtimes and Frameworks on IBM Cloud Platform - CP4A

Red Hat OpenShift

Tools

Kubernetes



cri-o

docker

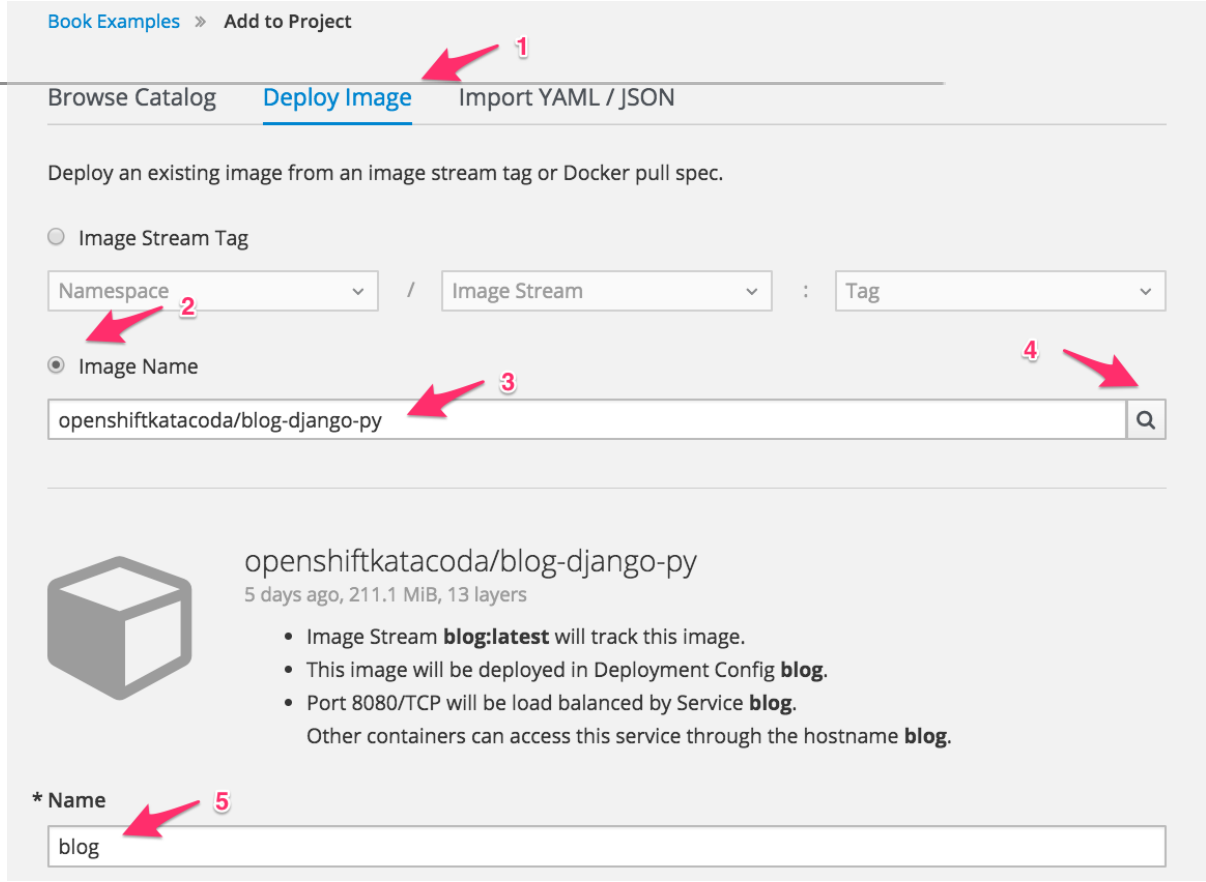
Docker

MICROPROFILE
OPTIMIZING ENTERPRISE JAVA

Deploy Applications to OpenShift

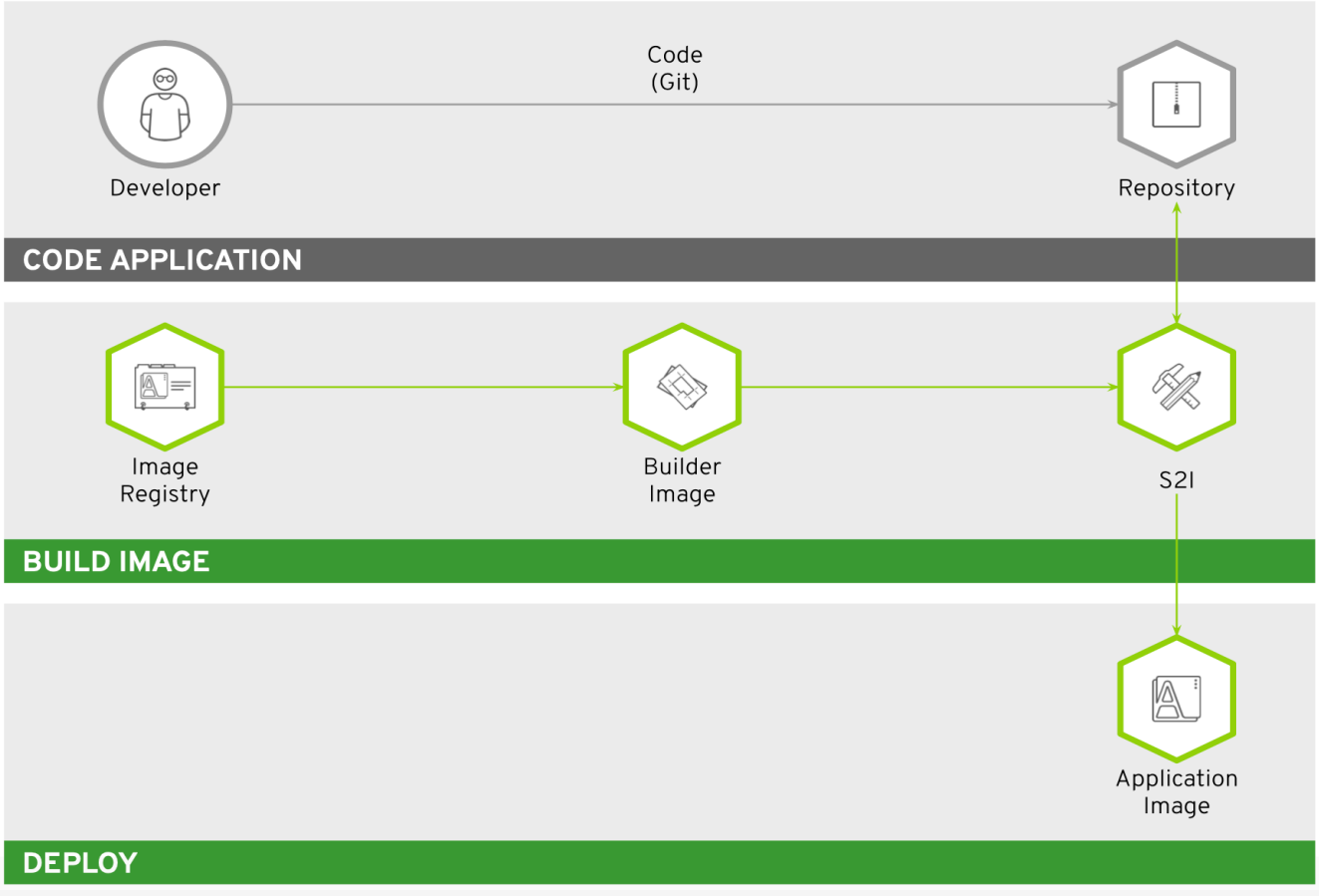
- Deploy an application from an existing container image.
- Build and deploy from source code contained in a Git repository using a Source-to-Image builder.
- Build and deploy from source code contained in a Git repository from a Dockerfile.

Deploy application image



Building and Deploying from Source

Source-to-Image - for building and deploying code



Building an Image from a Dockerfile

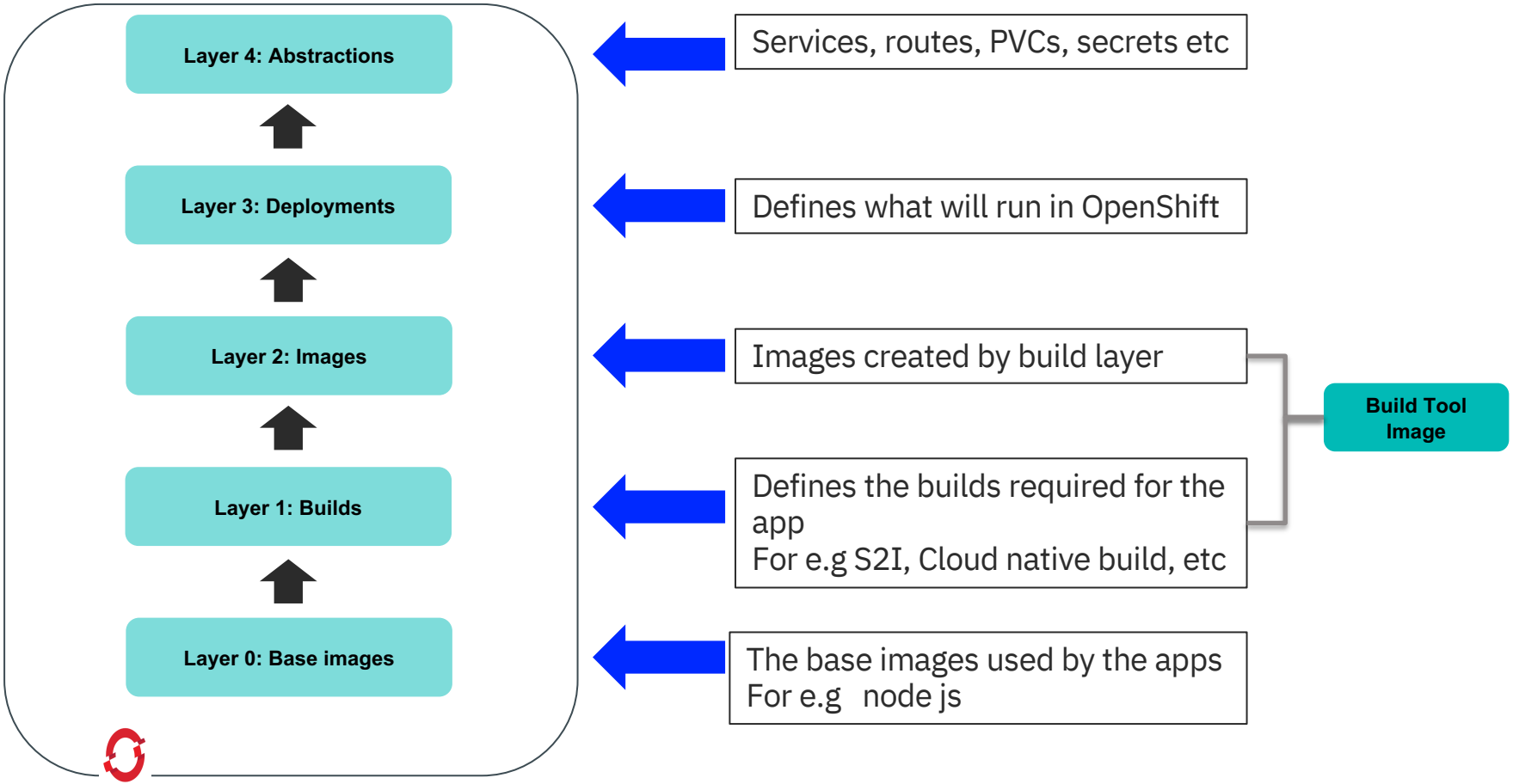
Docker Build Strategy

```
FROM openshift/python:3.5
USER root
RUN yum install -y wget
USER 1001
```

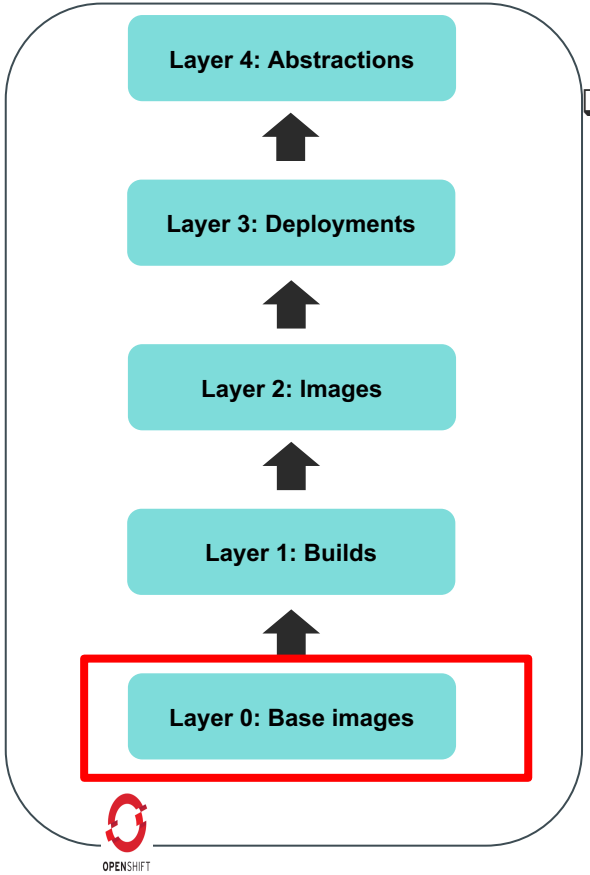
```
$ cat Dockerfile | oc new-build --name python-plus --dockerfile=-
--> Found image a080357 (2 days old) in image stream "openshift/python" under
    tag "3.5" for "openshift/python:3.5"

    ...

    * A Docker build using a predefined Dockerfile will be created
      * The resulting image will be pushed to image stream "python-plus:latest"
      * Use 'start-build' to trigger a new build

--> Creating resources with label build=python-plus ...
    imagestream "python-plus" created
    buildconfig "python-plus" created
--> Success
    Build configuration "python-plus" created and build triggered.
    Run 'oc logs -f bc/python-plus' to stream the build progress.
```

Think of OpenShift deployment as a set of layers

OpenShift Deployment – Developer view



Layer 0 – Base image examples



- Base operating systems, specific runtimes, databases, application servers and n
  - Available as **Image Streams**, an enhanced set of metadata about each image

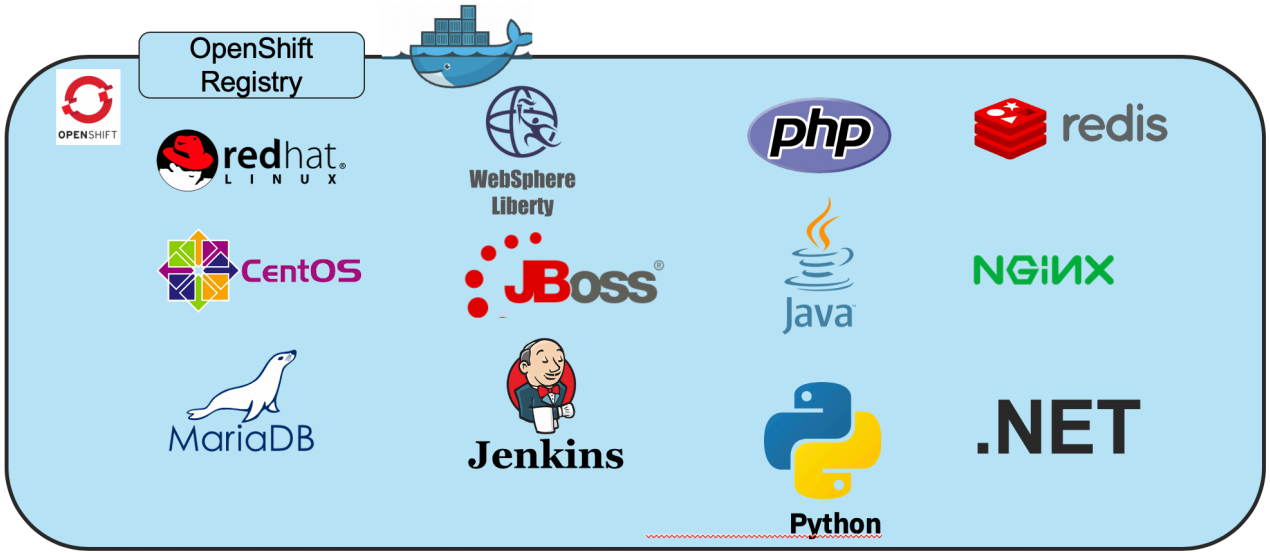
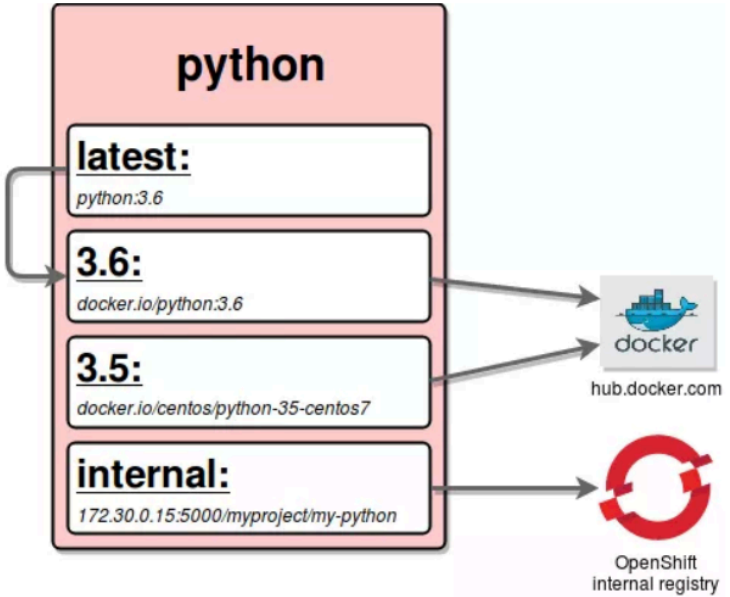


Image Streams

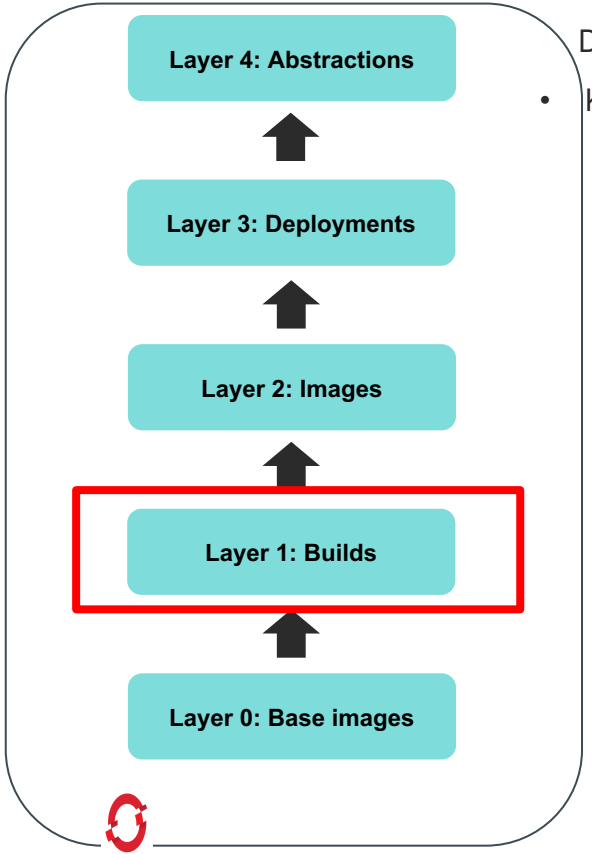
- An image stream represents one or more Docker images identified by tags.
  - Presents a single virtual view of related images
    - Can refer to images from any of the following:
      - Its own image repository in OpenShift's integrated Docker Registry
      - Other image streams
      - Docker image repositories from external registries
    - Image Streams are trigger an event when underlying image is changed (even if tag remains the same)

Image Stream example

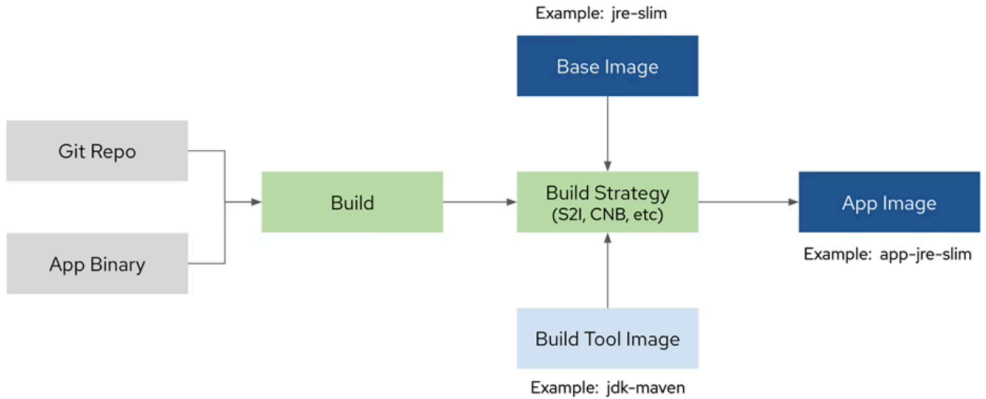


Graphic source:
https://blog.openshift.com/image-streams-faq/

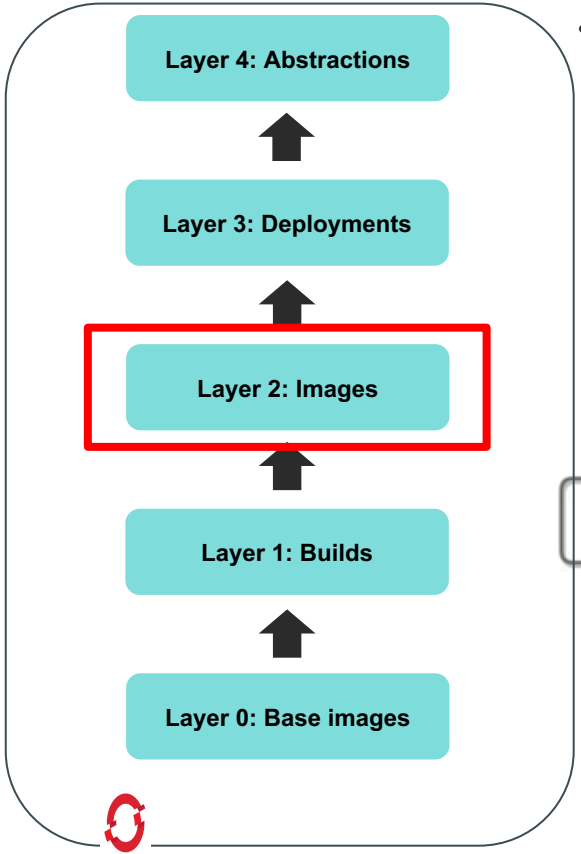
Level 1: Builds



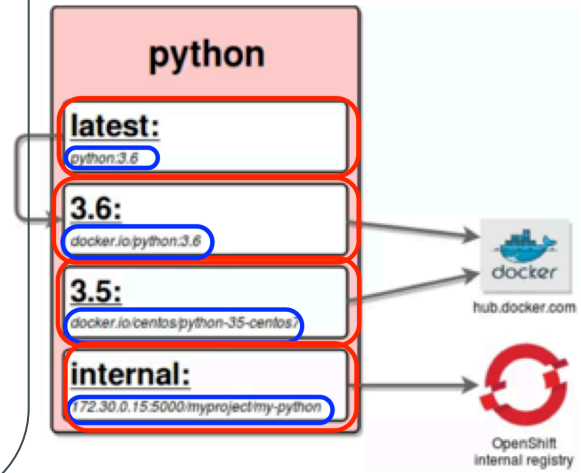
- Defined via a **BuildConfig** object
- A blueprint for a process to transform base images + source code, app binaries, or Dockerfiles to an app image:
- Key attributes:
  - **Input** (input to the build process) - e.g. file, directory, GH repo etc
  - **Strategy** (how to build the app image) -options
    - *S2I* – Use a specialized builder image to generate app image (more later)
    - *Docker* - Use a Docker file to generate app image
    - *Pipeline* - A Jenkins pipeline that generates the app image from source
    - *Custom* – Encapsulate your build process via a custom builder image
  - **Output** (result of the build process) – typically an ImageStream tag



Level 2: Images



- Defined via an **ImageStream** object
  - An abstraction for working with Docker images inside OpenShift
  - Key attributes:
    - **ImageStreamImage** (reference to actual image) - typically not used directly
    - **ImageStreamTag** (reference to a given ImageStream and tag)

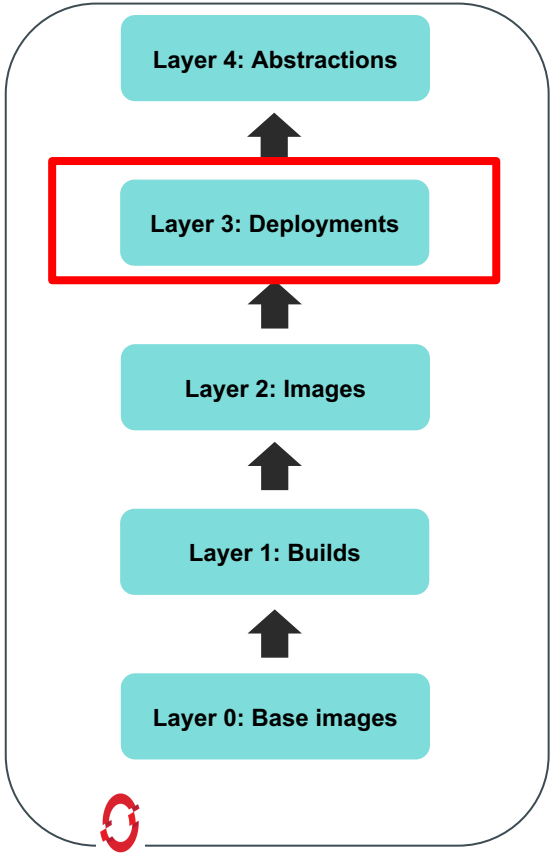


Key:

ImageStreamImage

ImageStreamTag

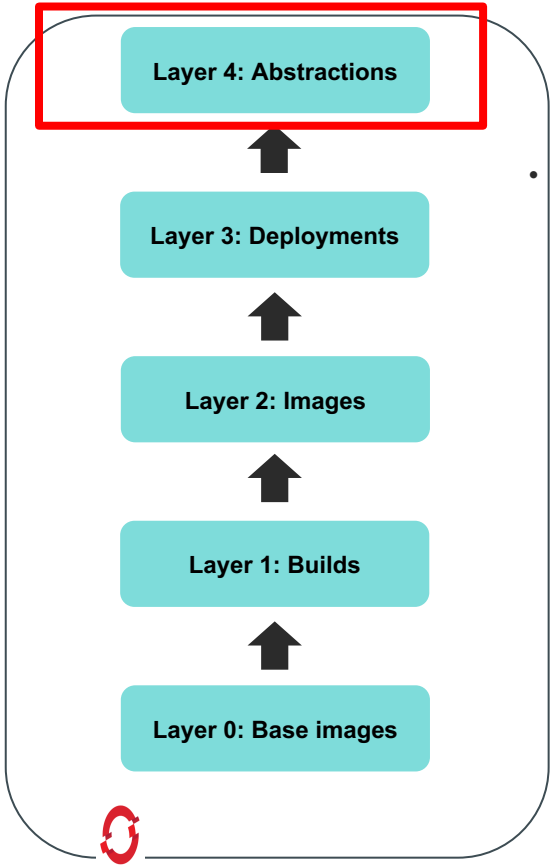
Level 3: Deployments



- Defined via a **DeploymentConfig** object
  - Encapsulates the K8s Deployment and adds deployment strategies and triggers
  - Key attributes:
    - **Strategy** (how to deploy if the underlying Deployment is already deployed) Options:
      - *Recreate*– Blow away old deployment first and then deploy new one
      - *Rolling (default)* – Zero downtime rollout via K8s *RollingUpdate*
      - *Advanced* - (Note: require routes)
        - *Blue-Green*
        - *A/B*
        - *etc*
    - **Triggers** (define conditions under which the deployment is automatically triggered)

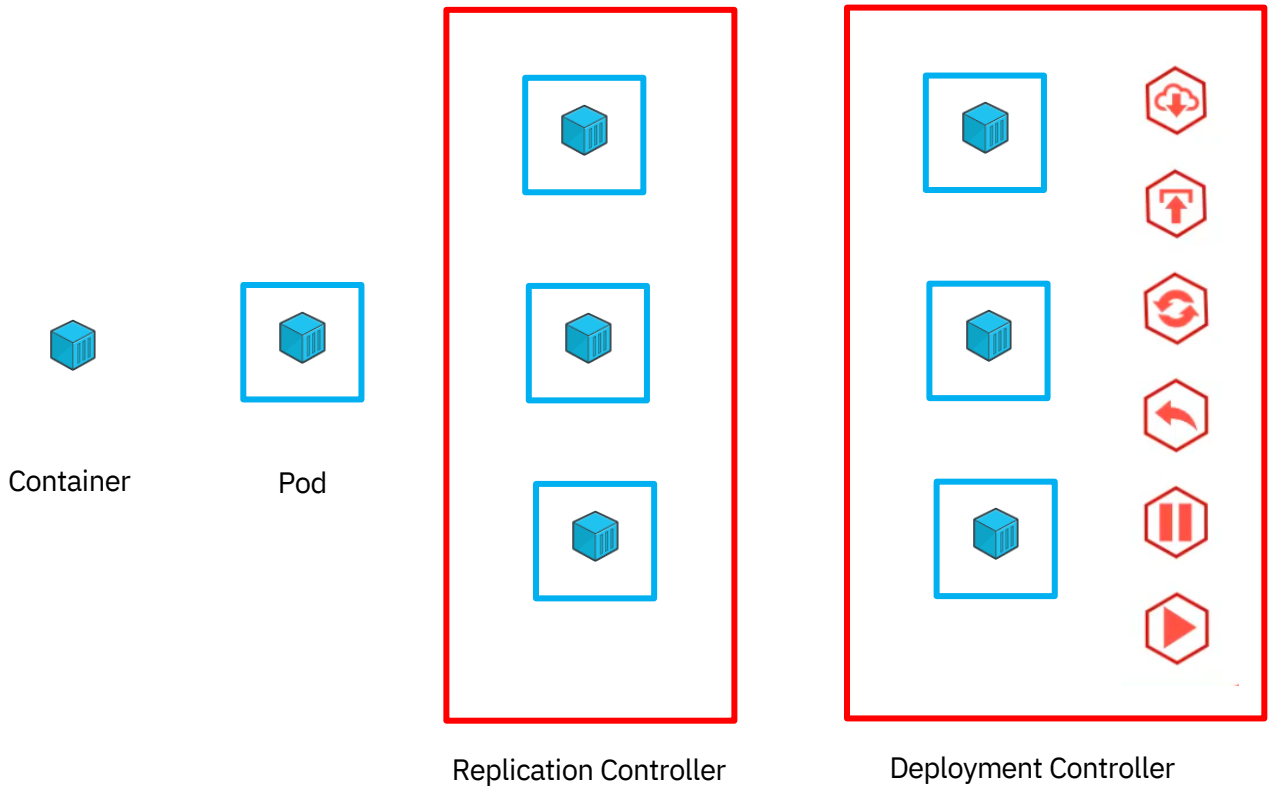
      e.g Input image updated, config changes

Level 4: Abstractions

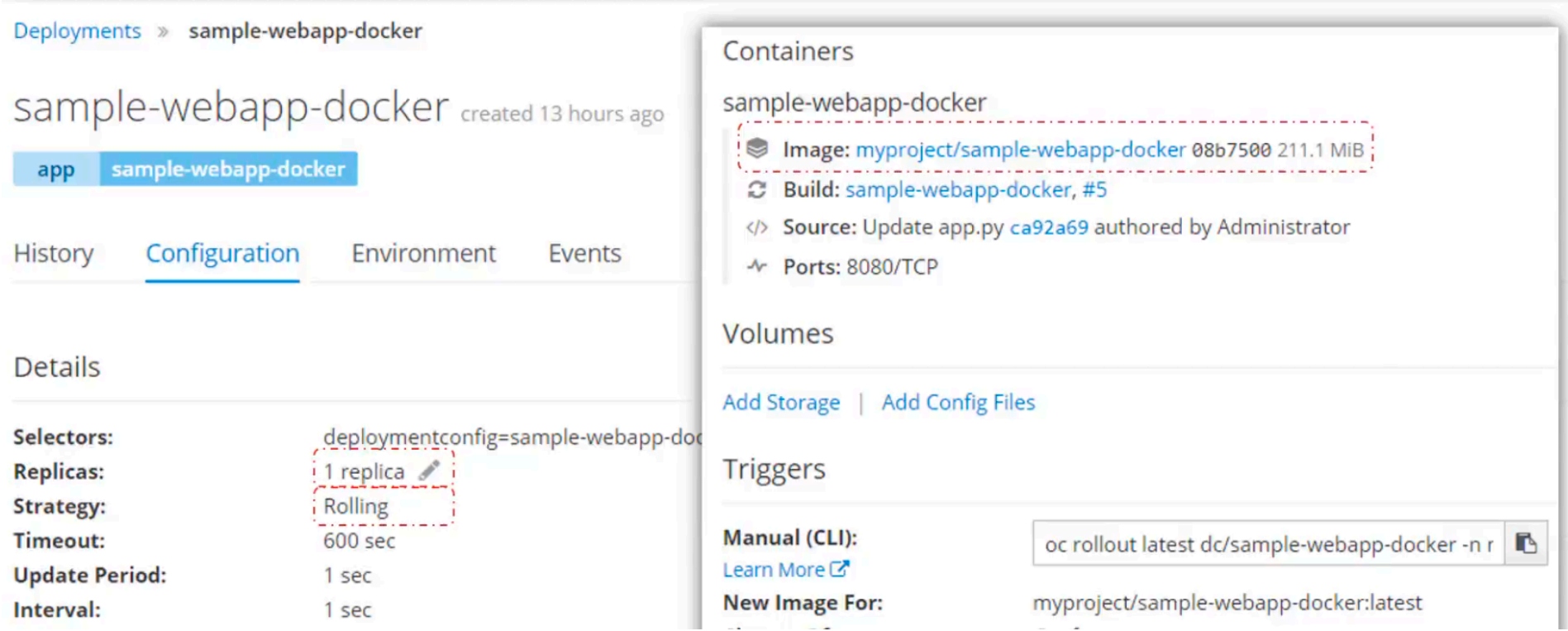


- Defines all of the additional resources needed for an app like networking, storage, security etc
  - Frequently used resources: (there are several others)
    - **Service** – Internal load balancer and router. Directs network traffic to replicated Pods
    - **Route** – Exposes a Service at a defined host name to allow access from external traffic
    - **Persistent Volume Claim** – Permanent storage used by apps to persist data after the app has stopped running
    - **Secret** – Mechanism for holding sensitive data. Decouples sensitive data from the the Pods that use them.

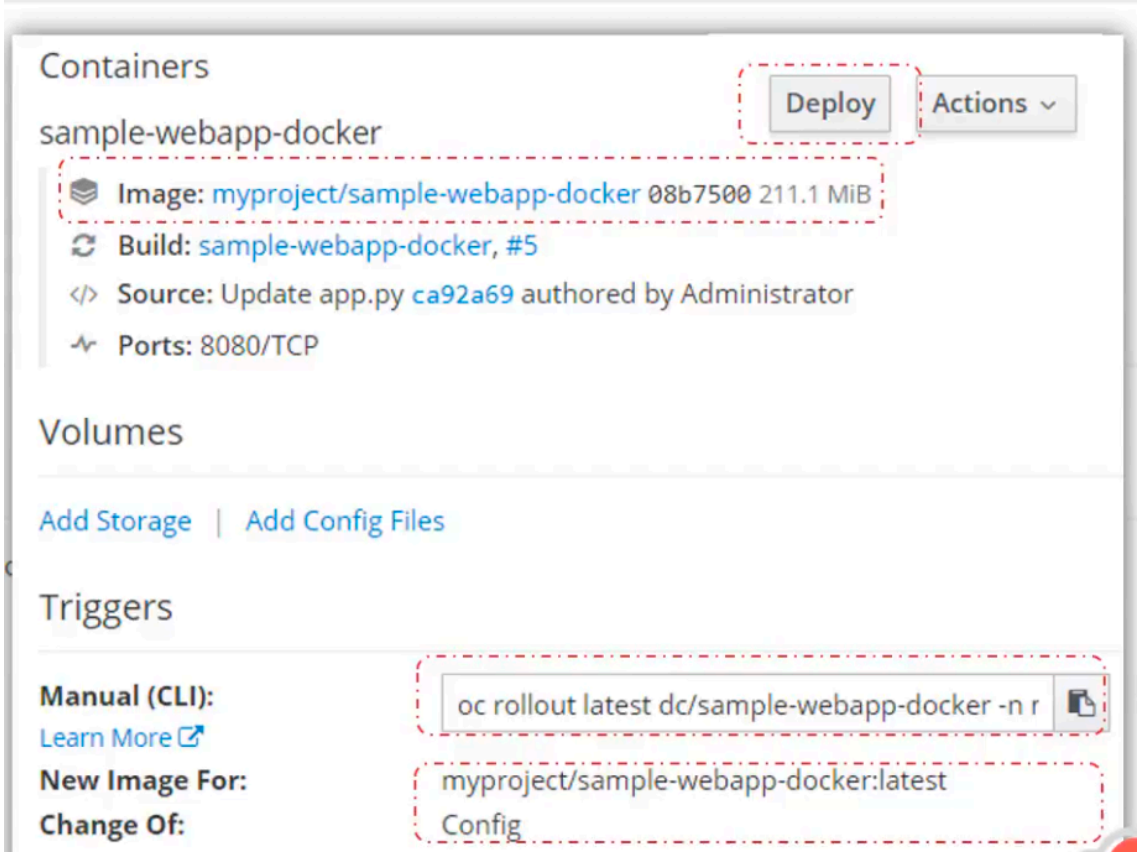
Deployment Controller



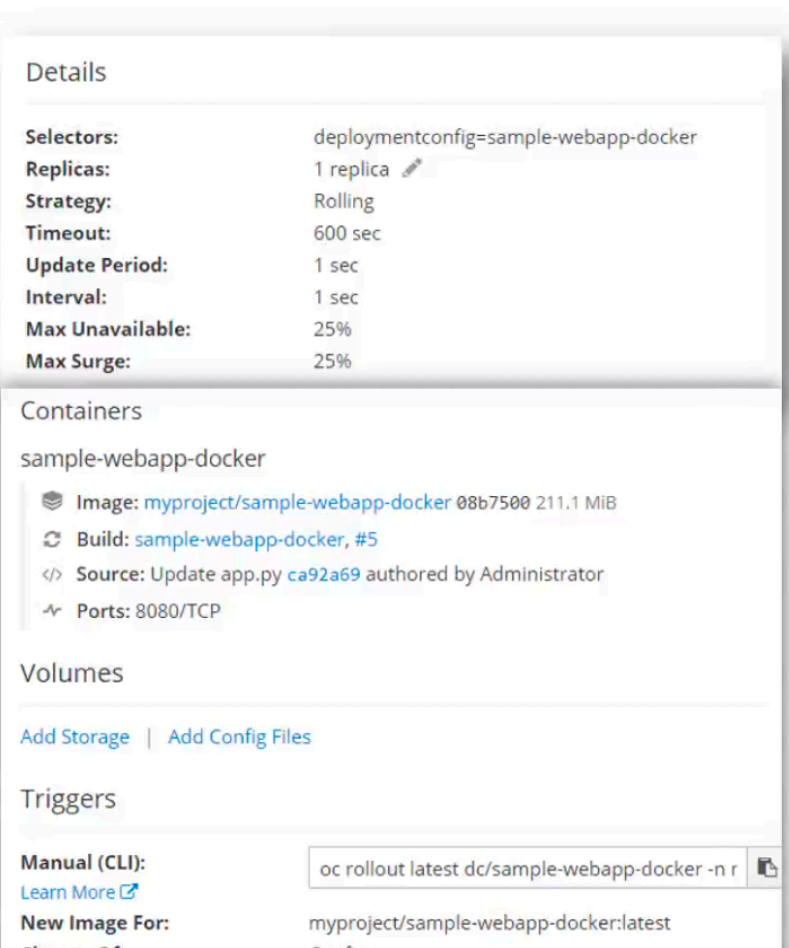
Deployment


Deployments » sample-webapp-docker
sample-webapp-docker created 13 hours ago
app sample-webapp-docker
History Configuration Environment Events
Details
Selectors: deploymentconfig=sample-webapp-doc
Replicas: 1 replica
Strategy: Rolling
Timeout: 600 sec
Update Period: 1 sec
Interval: 1 sec
Containers
sample-webapp-docker
Image: myproject/sample-webapp-docker 08b7500 211.1 MiB
Build: sample-webapp-docker, #5
Source: Update app.py ca92a69 authored by Administrator
Ports: 8080/TCP
Volumes
Add Storage | Add Config Files
Triggers
Manual (CLI):
oc rollout latest dc/sample-webapp-docker -n r
Learn More
New Image For:
myproject/sample-webapp-docker:latest

Deployment


Containers
Deploy
Actions
sample-webapp-docker
Image: myproject/sample-webapp-docker 08b7500 211.1 MiB
Build: sample-webapp-docker, #5
Source: Update app.py ca92a69 authored by Administrator
Ports: 8080/TCP
Volumes
Add Storage | Add Config Files
Triggers
Manual (CLI):
oc rollout latest dc/sample-webapp-docker -n r
Learn More
New Image For:
myproject/sample-webapp-docker:latest
Change Of:
Config

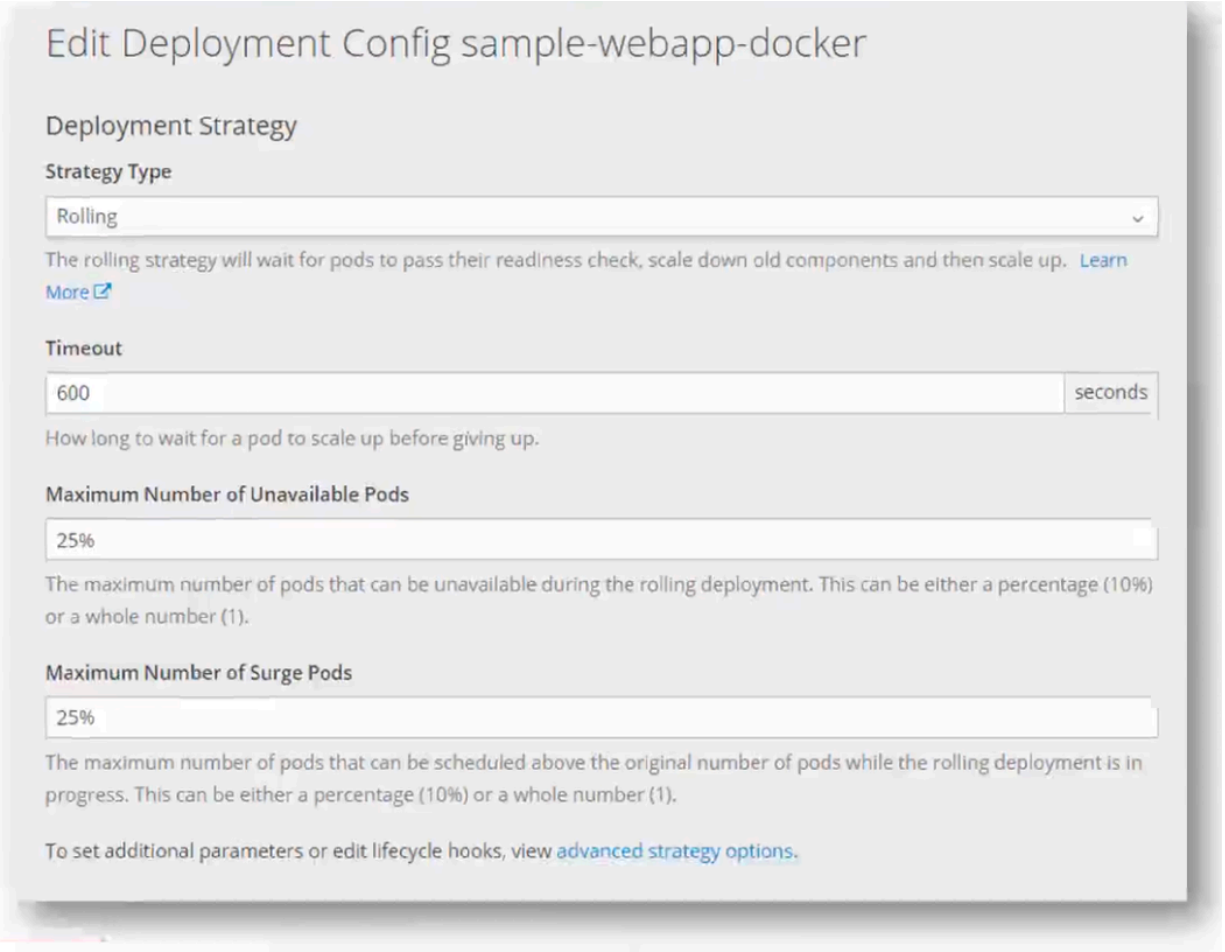
Deployment YAML



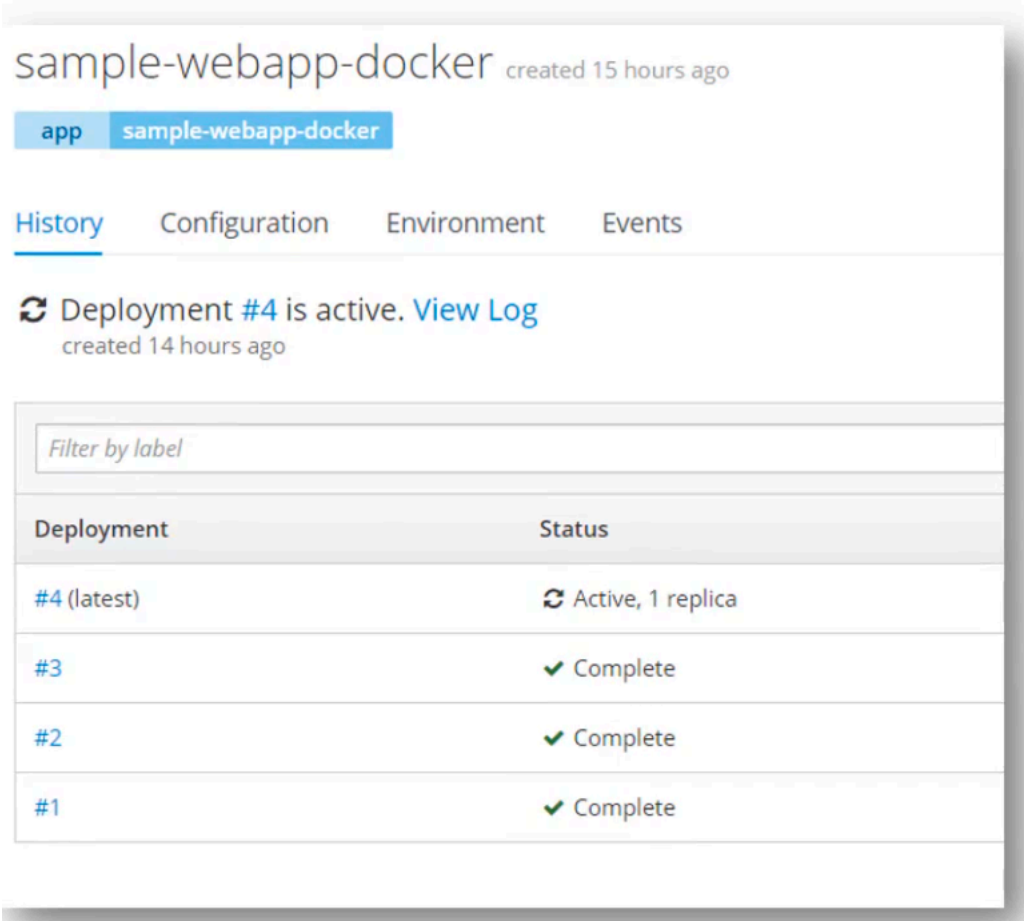
deployment-config.yaml

```yaml
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  name: sample-webapp-docker
spec:
  replicas: 1
  selector:
    deploymentconfig: sample-webapp-docker
  strategy:
    type: Rolling
  template:
    metadata:
      labels:
        app: sample-webapp-docker
        deploymentconfig: sample-webapp-docker
    spec:
      containers:
        - image: myproject/sample-webapp-docker
          imagePullPolicy: Always
          name: sample-webapp-docker
          ports:
            - containerPort: 8080
              protocol: TCP
  triggers:
    - imageChangeParams:
        automatic: true
        containerNames:
          - sample-webapp-docker
        from:
          kind: ImageStreamTag
          name: 'sample-webapp-docker:latest'
```

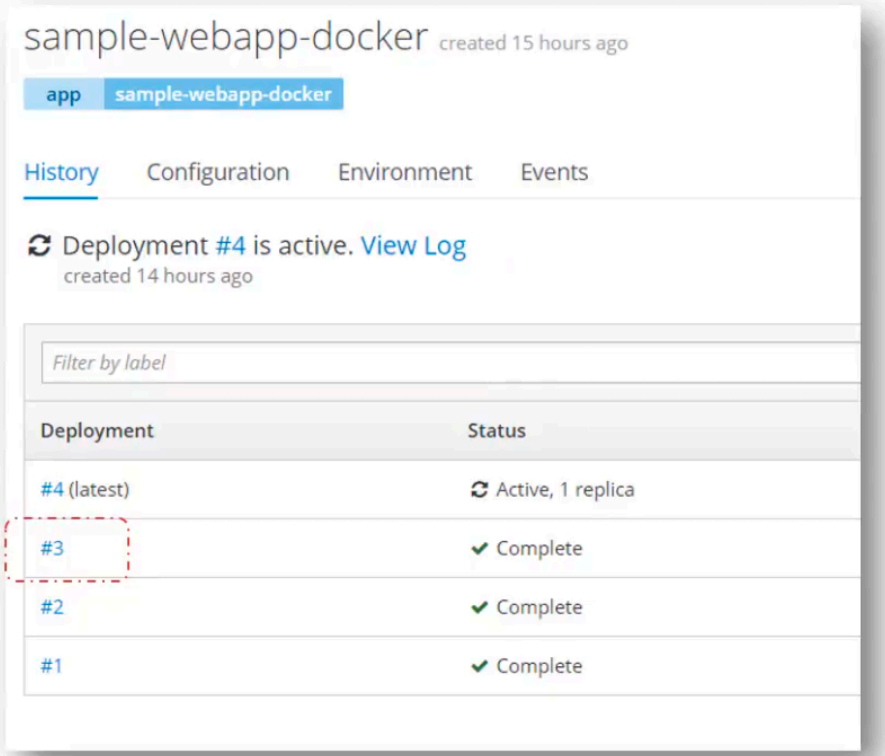
Edit Deployment Configuration

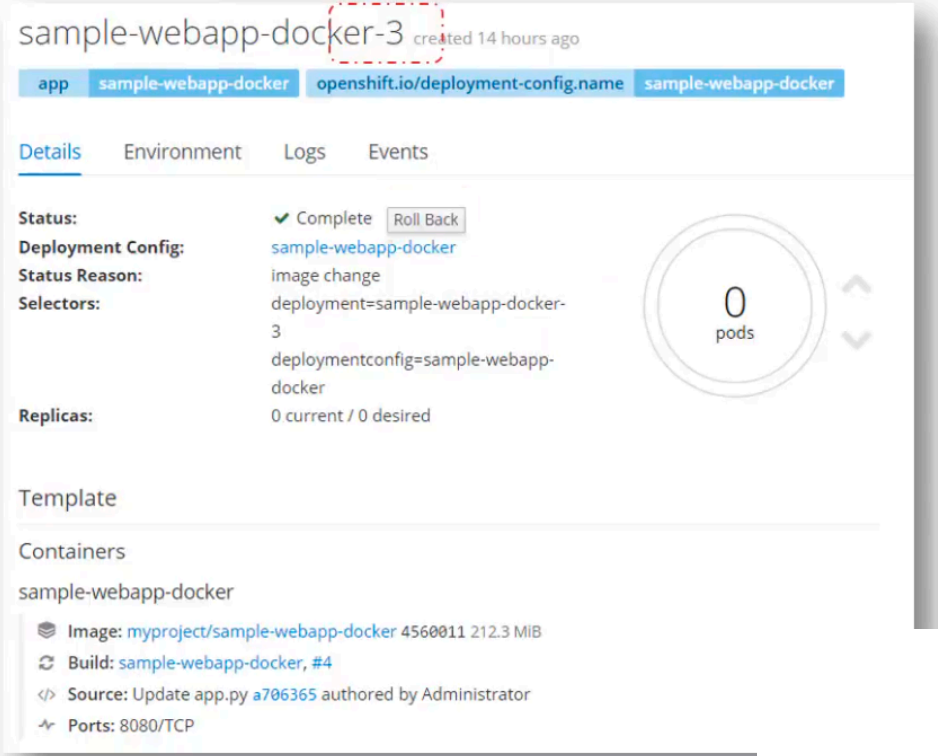


Deployment History



Rollback


sample-webapp-docker created 15 hours ago
app
sample-webapp-docker
History
Configuration
Environment
Events
Deployment #4 is active. View Log
created 14 hours ago
Filter by label
Deployment
Status
#4 (latest)
Active, 1 replica
#3
Complete
#2
Complete
#1
Complete


sample-webapp-docker-3 created 14 hours ago
app
sample-webapp-docker
openshift.io/deployment-config.name
sample-webapp-docker
Details
Environment
Logs
Events
Status:
Complete
Roll Back
Deployment Config:
sample-webapp-docker
Status Reason:
image change
Selectors:
deployment=sample-webapp-docker-3
deploymentconfig=sample-webapp-docker
Replicas:
0 current / 0 desired
0
pods
Template
Containers
sample-webapp-docker
Image: myproject/sample-webapp-docker 4560011 212.3 MiB
Build: sample-webapp-docker, #4
Source: Update app.py a706365 authored by Administrator
Ports: 8080/TCP

Deployment Strategy – Why?

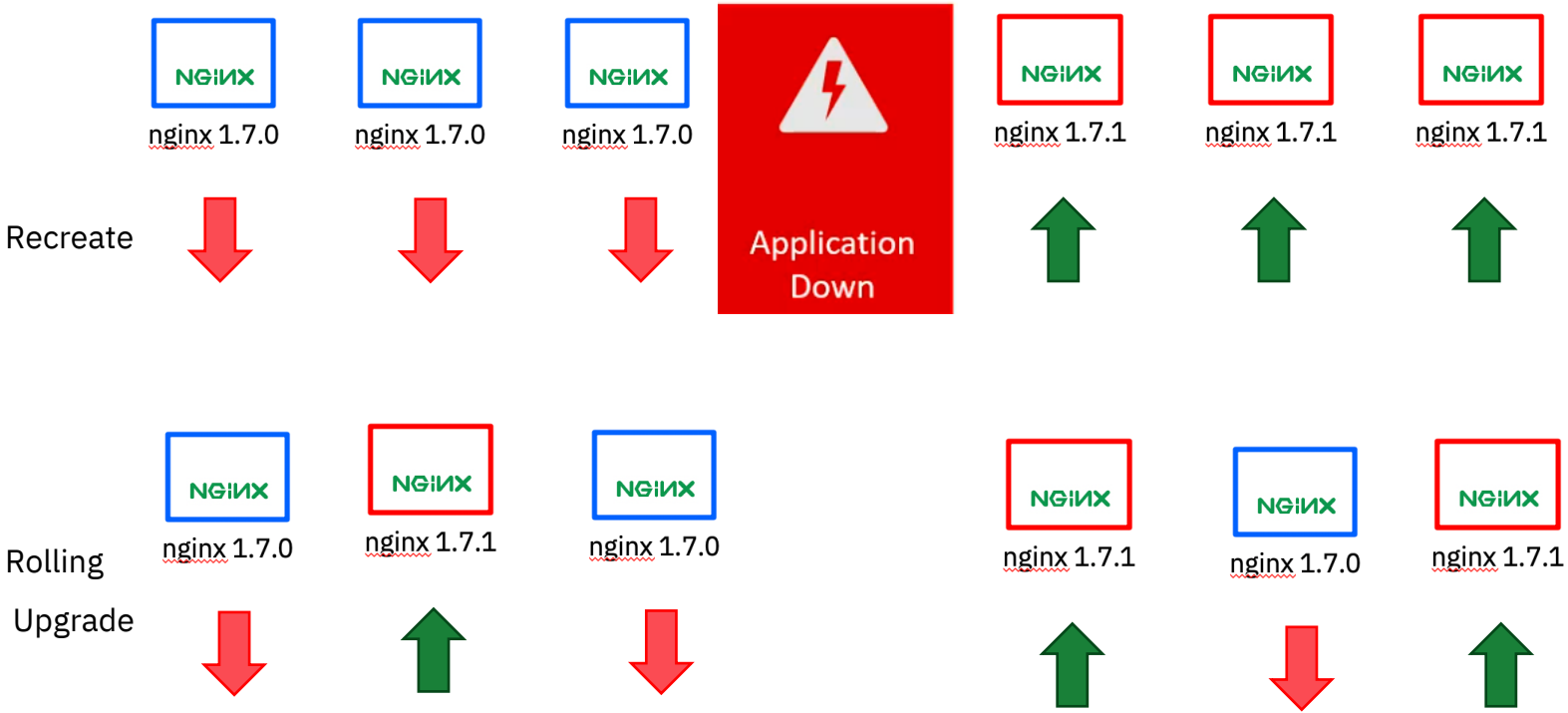
- A way to change or upgrade an application.
- The aim is to make the change without downtime
- In a way that the user barely notices the improvements.

Things to consider - Define Deployment Strategy

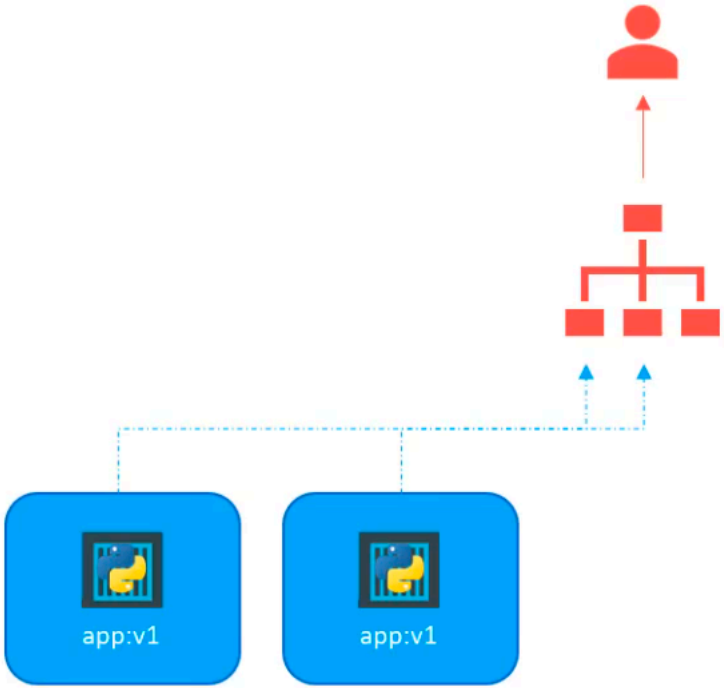
1. Long running connections need to be handled gracefully.
2. Database conversions can get tricky and will need to be done and rolled back along with the application.
3. If the application is a hybrid of microservices and traditional components downtime may be needed to complete the transition.
4. You need the infrastructure to do this.
5. If you have a non-isolated test environment, you can break both new and old versions.

Deployment Strategies

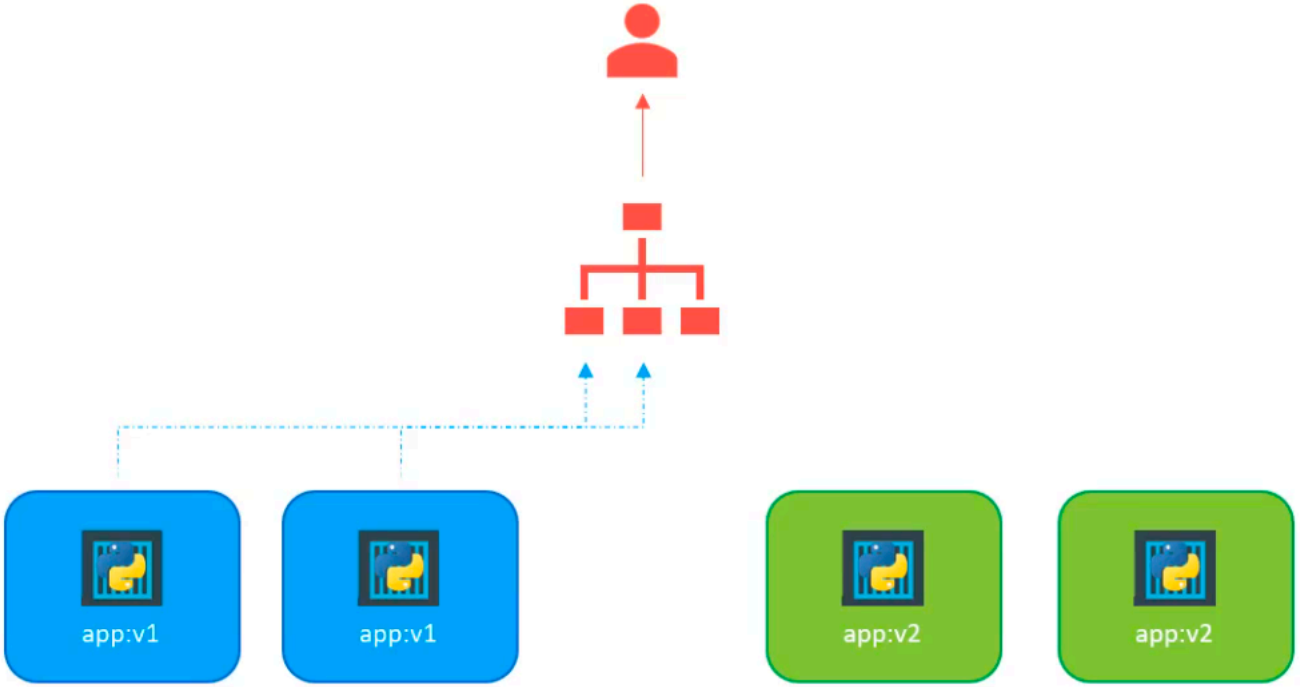
Take down application and not want another version



Advanced Strategies – Blue Green



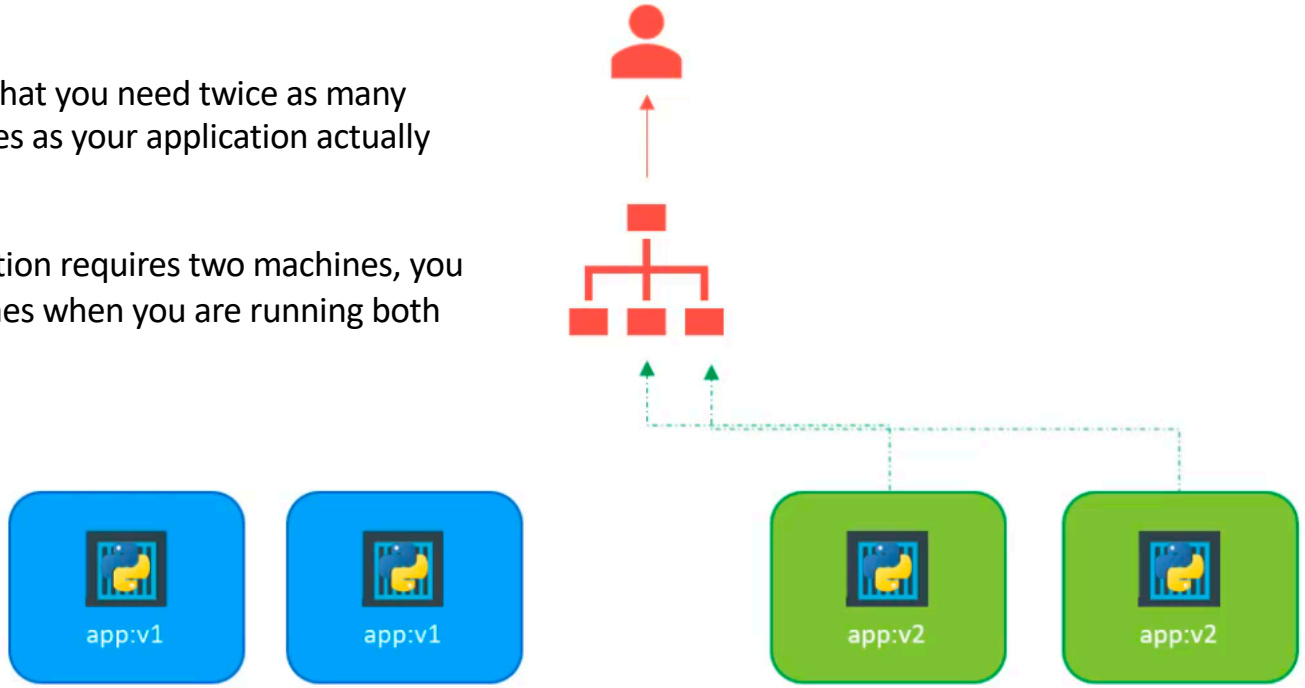
Advanced Strategies – Blue Green



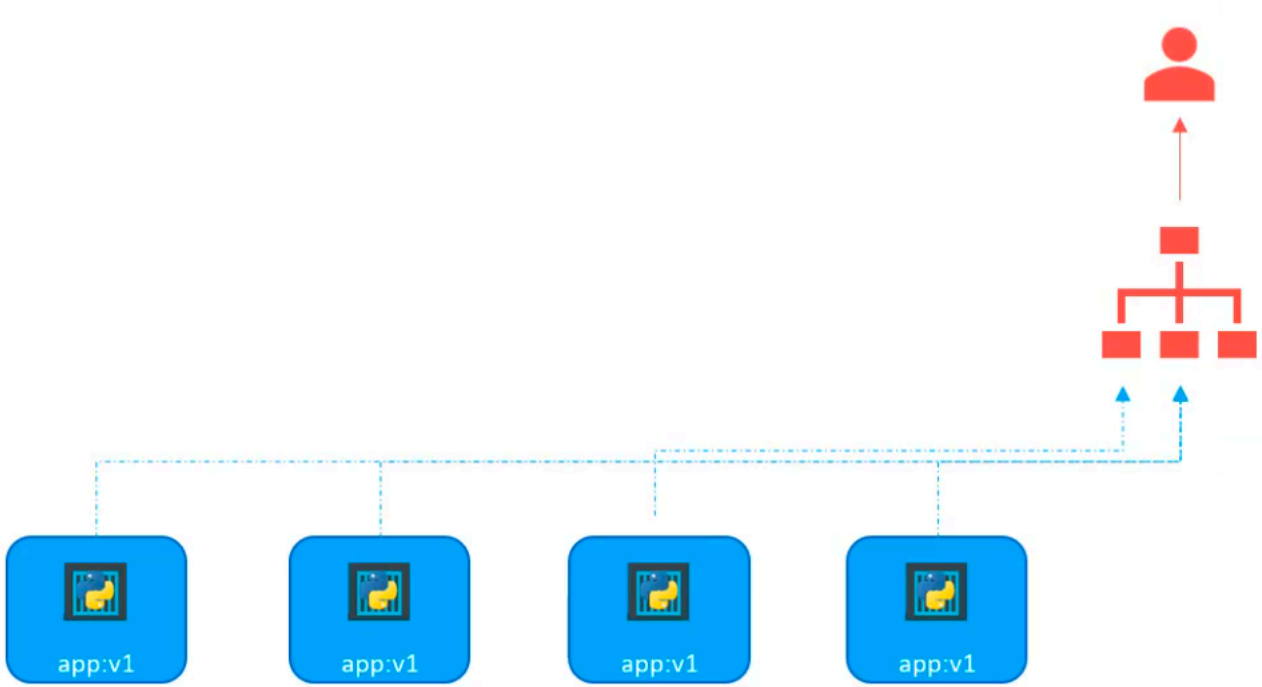
Advanced Strategies – Blue Green

The downside is that you need twice as many compute resources as your application actually needs.

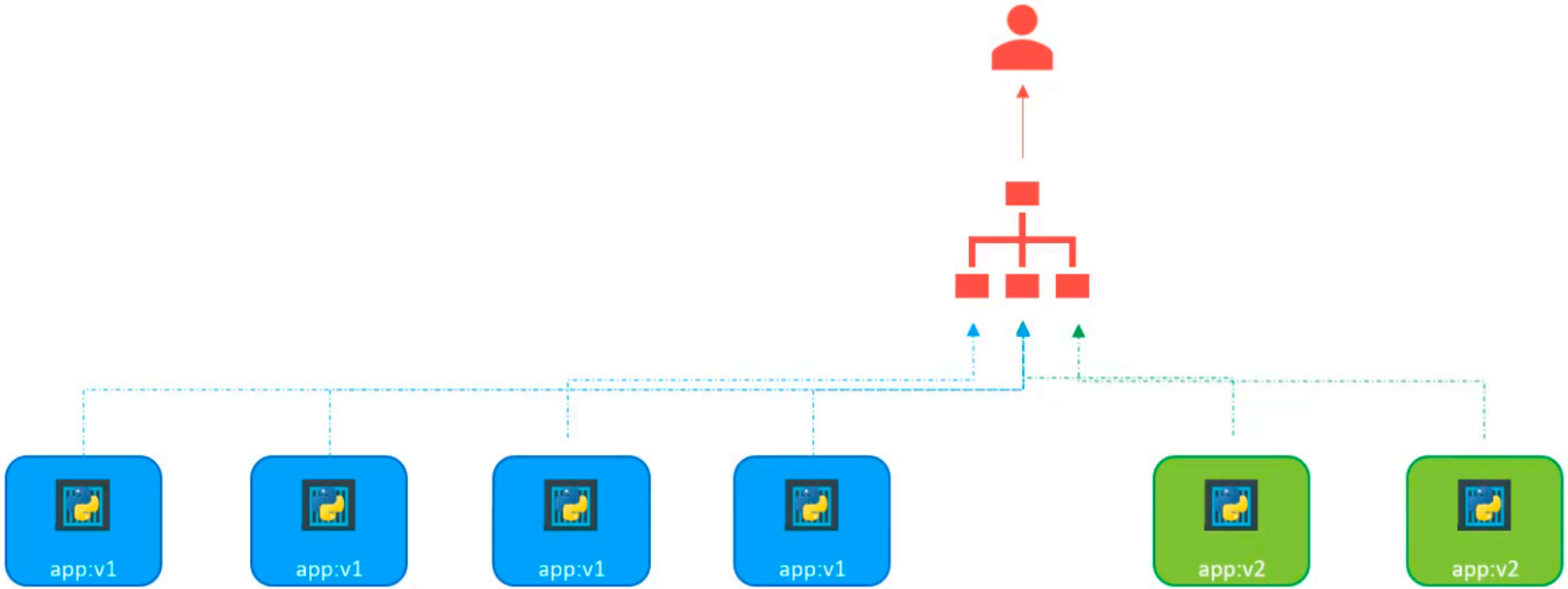
So if your application requires two machines, you need four machines when you are running both deployments.



Advanced Strategies – A/B

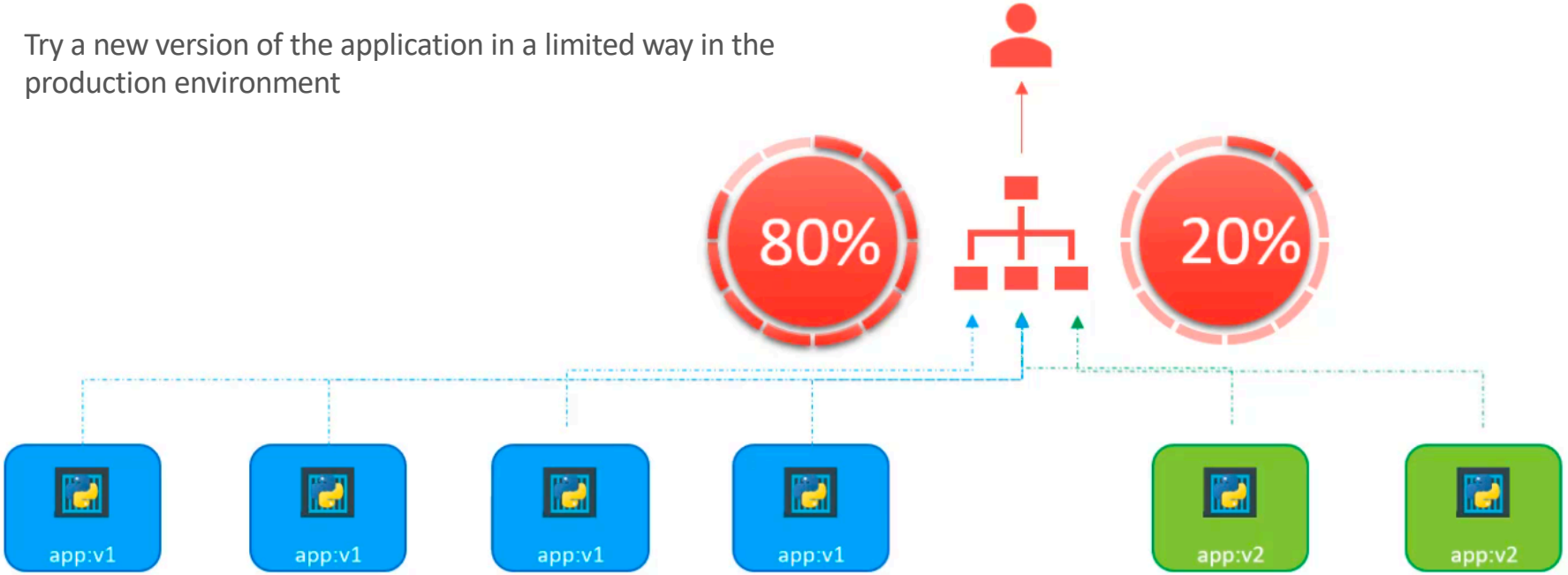


Advanced Strategies – A/B

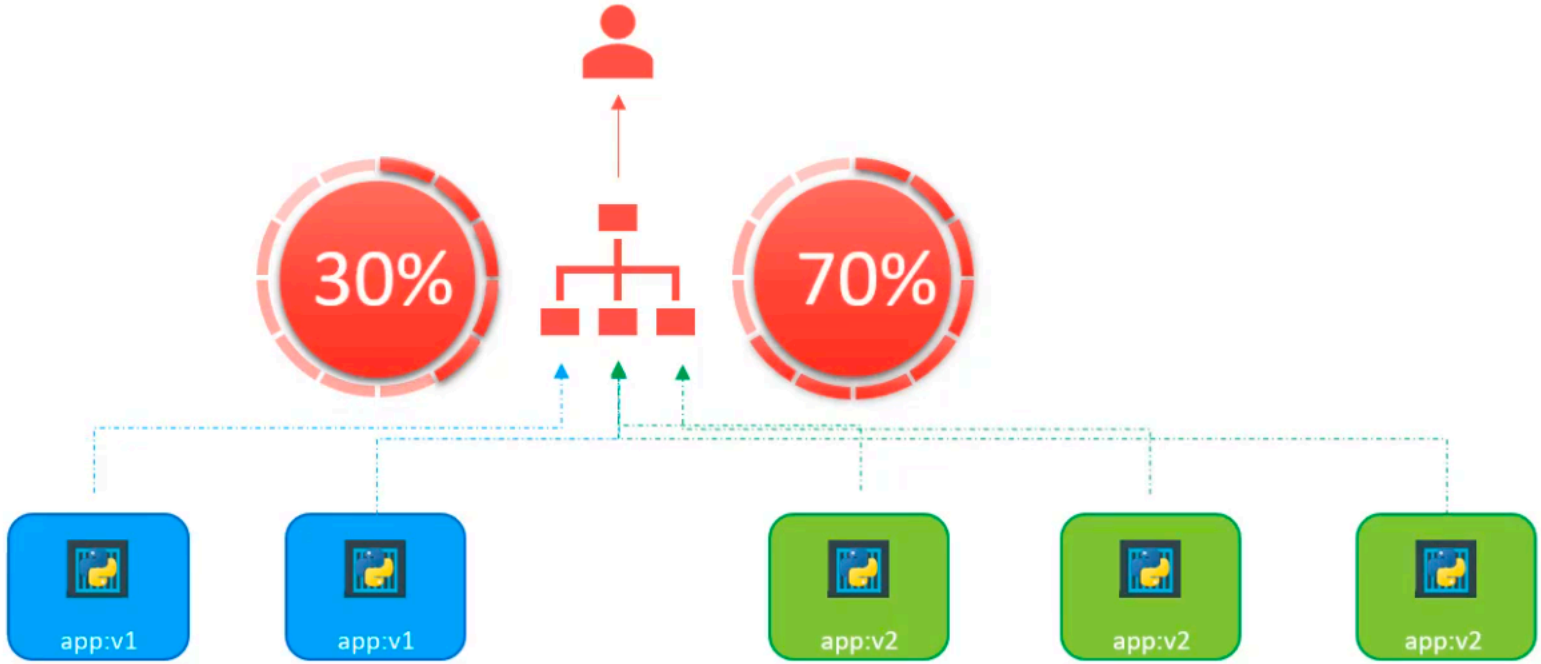


Advanced Strategies – A/B

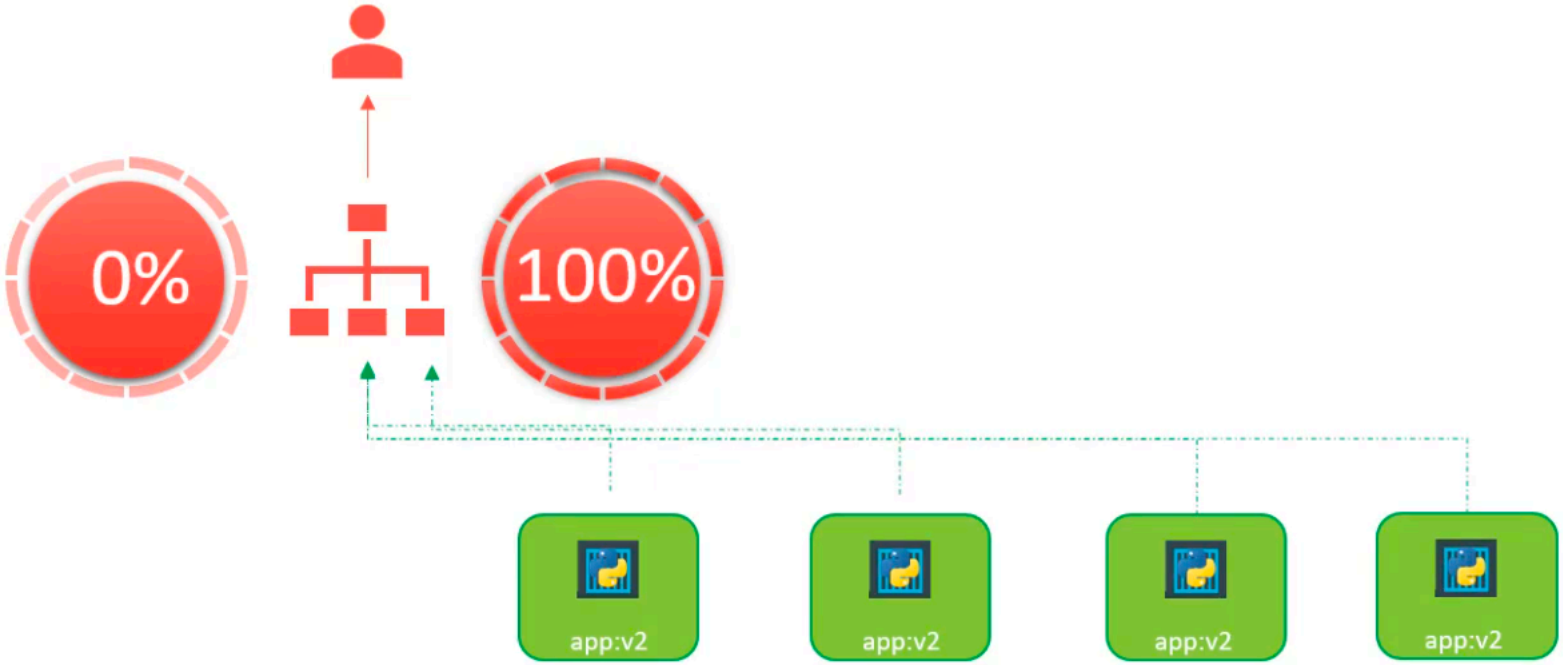
Try a new version of the application in a limited way in the production environment



Advanced Strategies – A/B



Advanced Strategies – A/B



Commands

```
> oc rollout latest dc/simple-webapp-docker
```

```
> oc rollout history dc/simple-webapp-docker
```

```
> oc rollout describe dc simple-webapp-docker
```

```
> oc rollout undo dc/simple-webapp-d
```

Demo Time!