



OPENSIFT: LES BASES POUR DÉBUTANT

ULRICH MONJI

PLAN



Présentation du formateur

Introduction au DevOps et PaaS

Composants d'OpenShift

Utilisateurs et Projets

Build et Déploiement

Gestion du réseau

Gestion du stockage

Scaling avec Openshift

Introduction au catalogue

Mini-projet

PLAN



Présentation du formateur

Introduction au DevOps et PaaS

Composants d'OpenShift

Utilisateurs et Projets

Build et Déploiement

Gestion du réseau

Gestion du stockage

Scaling avec Openshift

Introduction au catalogue

Mini-projet

PRÉSENTATION DU FORMATEUR

- Atos-Worldline - Ingénieur Système
 - Build et Run de plateforme Cloud
 - Virtualisation - Conteneurisation - Automatisation
 - Comptes clients: Carrefour, Auchan, ARJEL, SAMU
- Adneom - Consultant IT
- Groupe SII - Consultant IT (Cloud/Devops)
 - Consultant chez Orange France
 - Migration d'une application monolithique en microservice
- Formateur chez Eazytraining



PLAN



Présentation du formateur

Introduction au DevOps et PaaS

Composants d'OpenShift

Utilisateurs et Projets

Build et Déploiement

Gestion du réseau

Gestion du stockage

Scaling avec Openshift

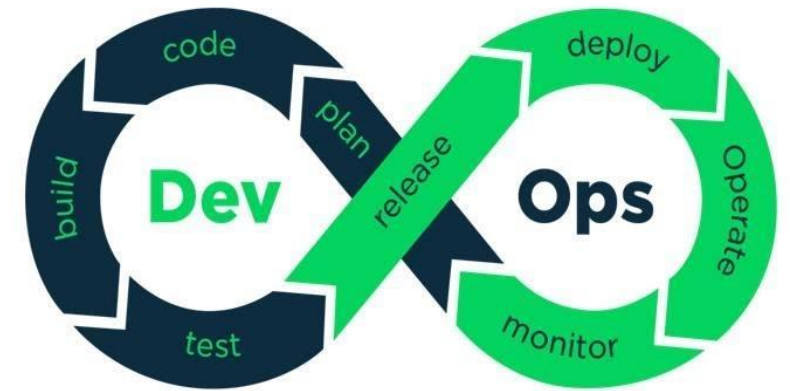
Introduction au catalogue

Mini-projet

INTRODUCTION AU DEVOPS ET PAAS (1/4): LE DEVOPS

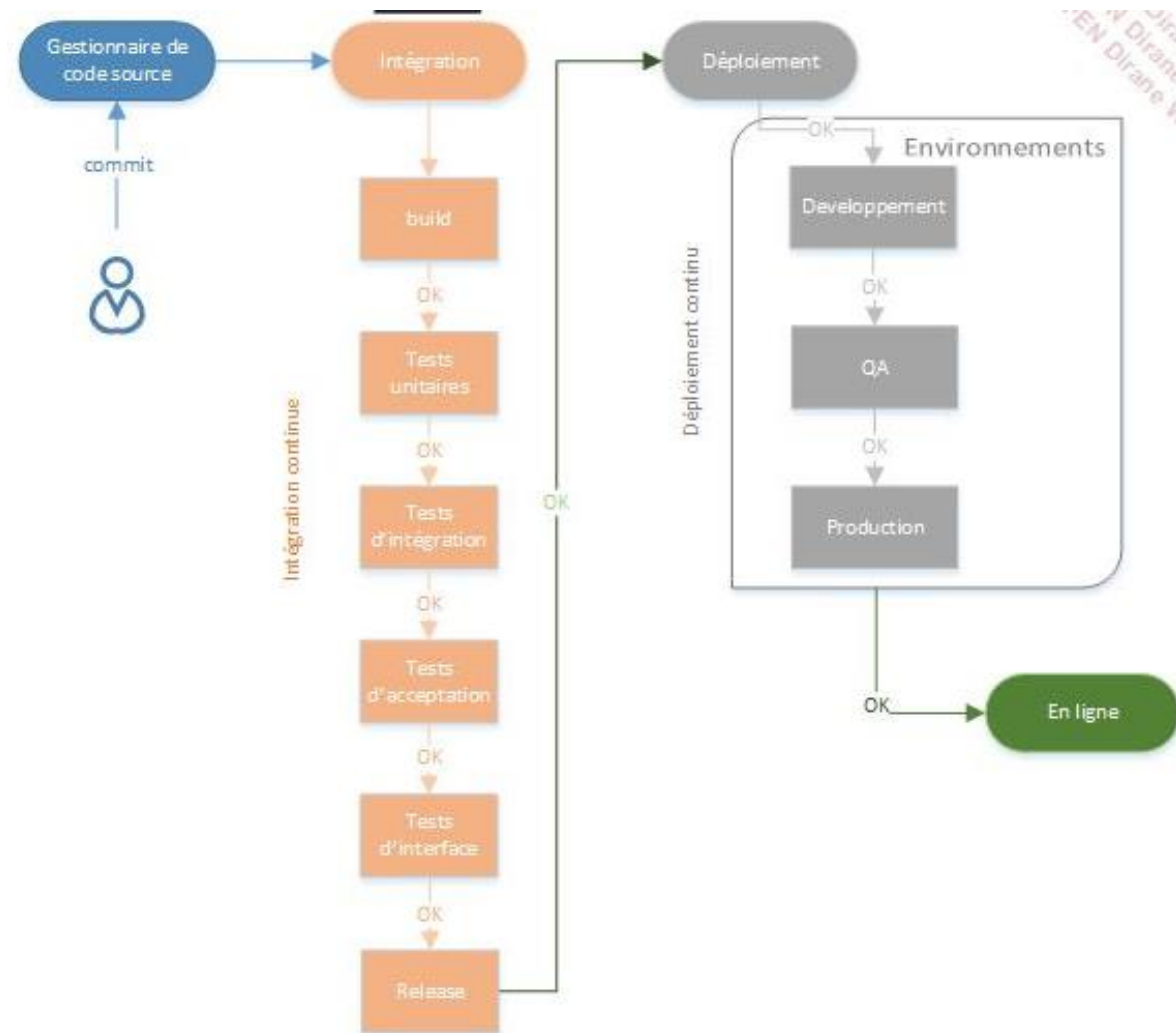
- Agile: méthode de développement
- DevOps: agilité dans le Dev et l'Ops = CI + CD

Agile vs. DevOps

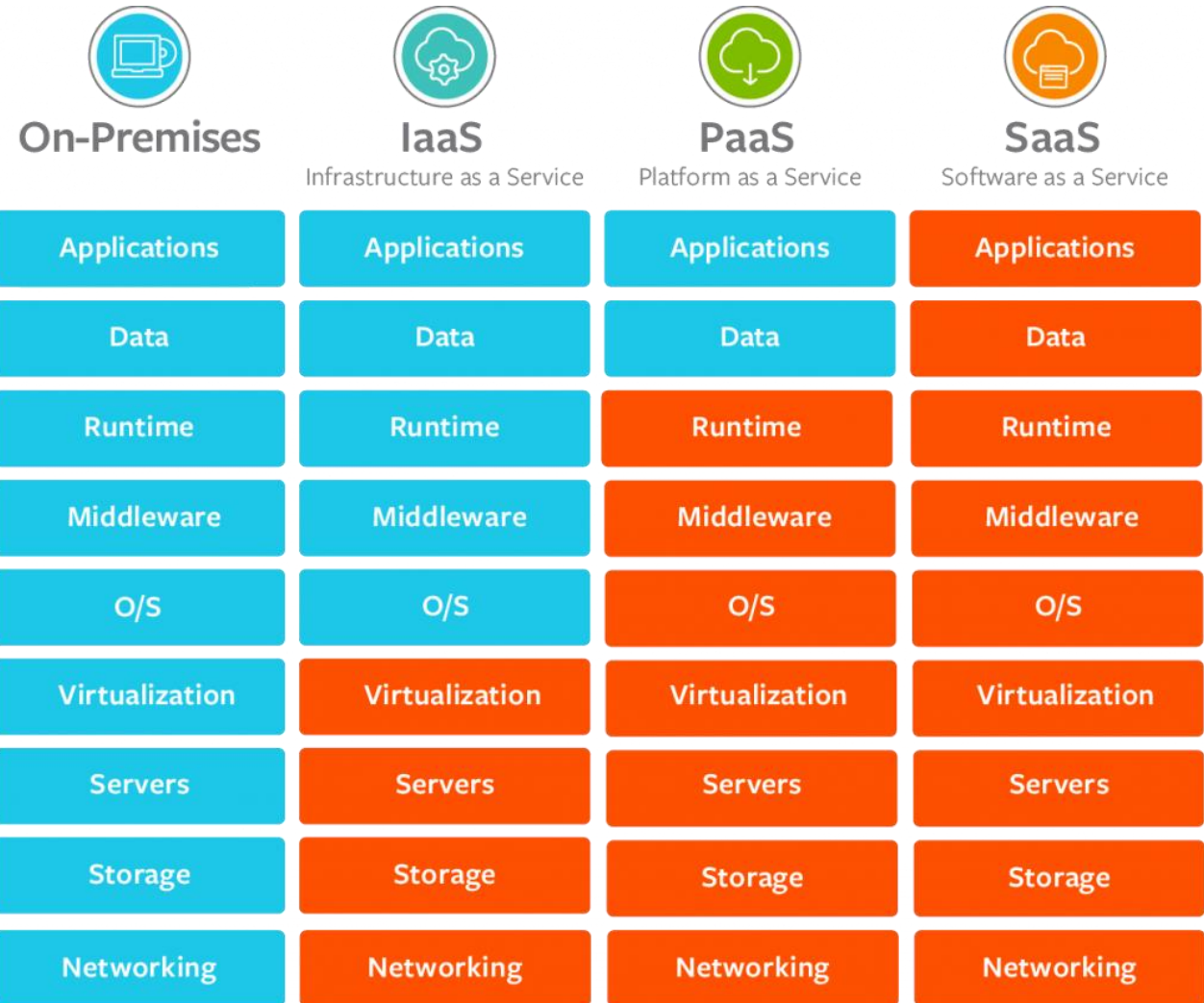


INTRODUCTION AU DEVOPS ET PAAS (2/4): CI/CD

- Intégration en continu
- Test en continu
- Déploiement en Continu
- Cloud



INTRODUCTION AU DEVOPS ET PAAS (3/4): PAAS



INTRODUCTION AU DEVOPS ET PAAS (4/4): PREREQUIS



PLAN



Présentation du formateur

Introduction au DevOps et PaaS

Composants d'OpenShift

Utilisateurs et Projets

Build et Déploiement

Gestion du réseau

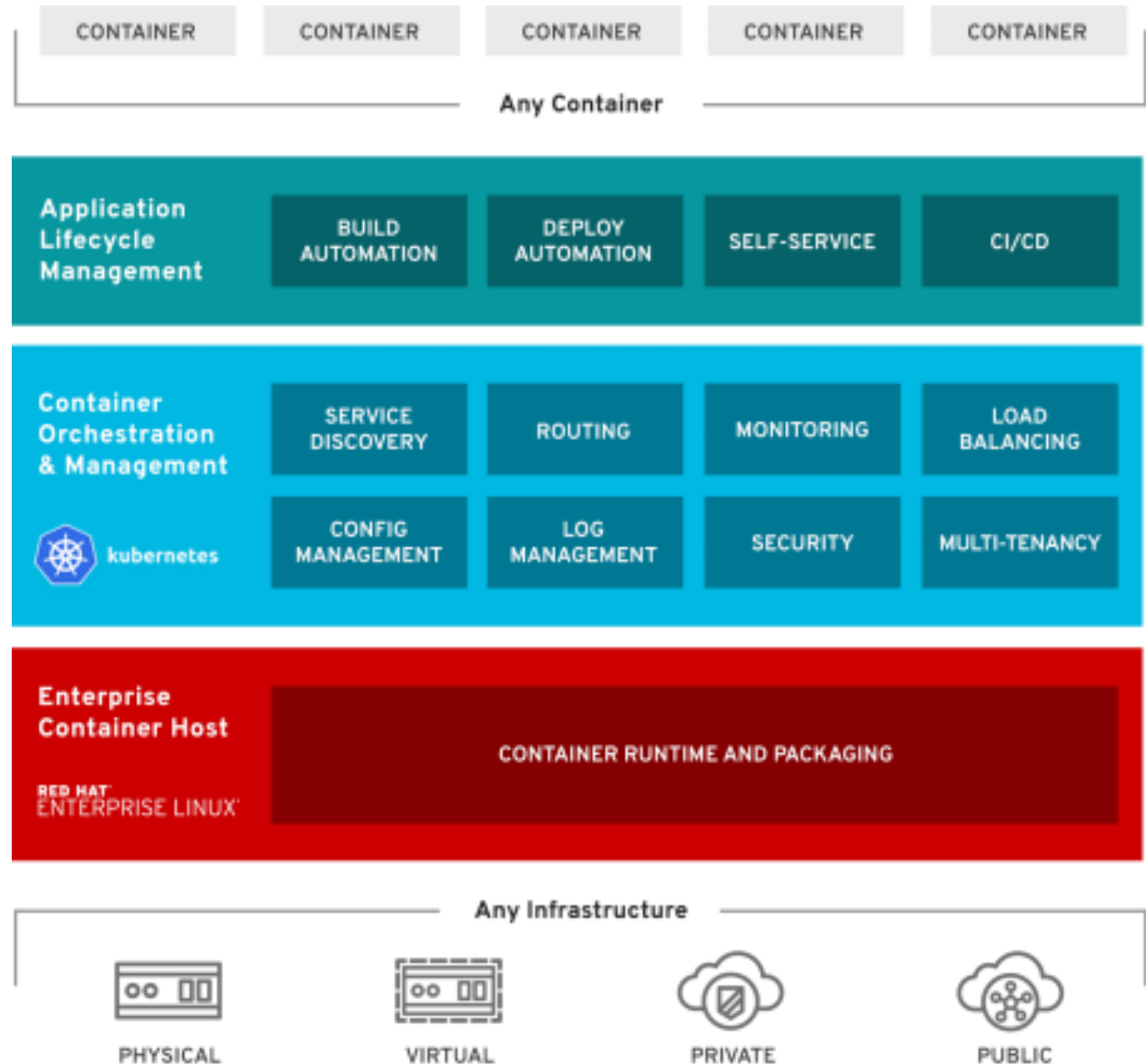
Gestion du stockage

Scaling avec Openshift

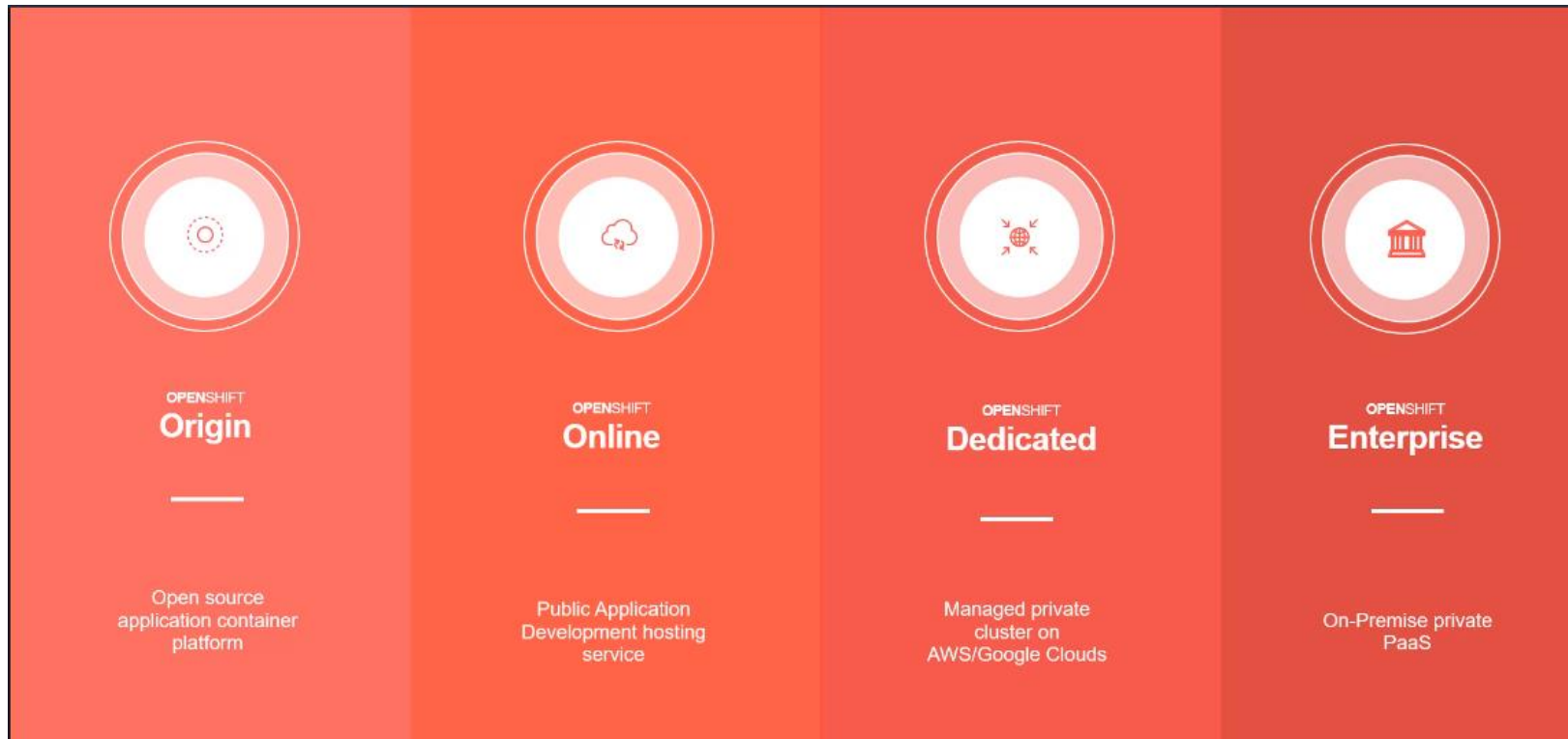
Introduction au catalogue

Mini-projet

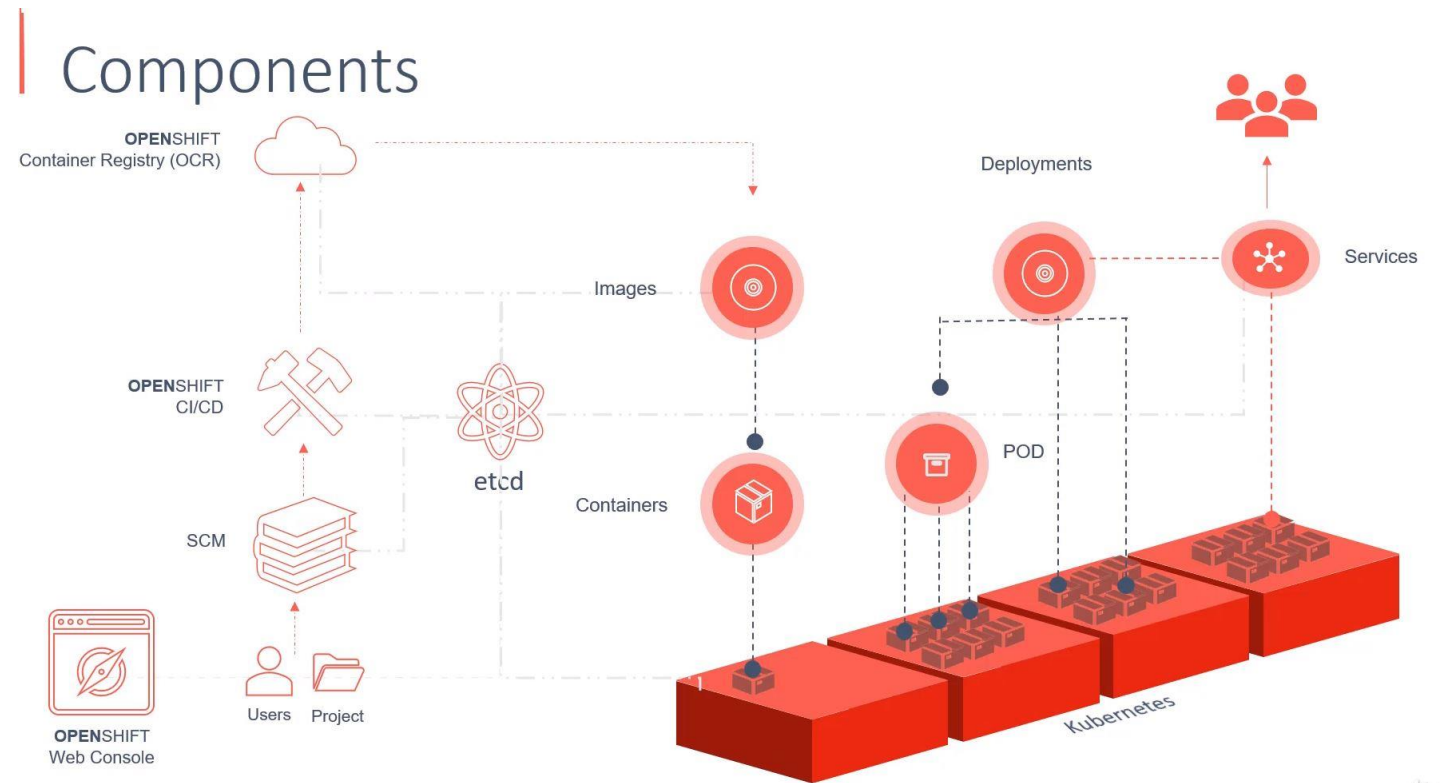
COMPOSANTS D'OPENSIFT (1/6): DEFINITION



COMPOSANTS D'OPENSHIFT (2/6): EDITIONS



COMPOSANTS D'OPENSHIFT (3/6): ARCHITECTURE



COMPOSANTS D'OPENSHIFT (4/6): INTÉRACTION



Web Console

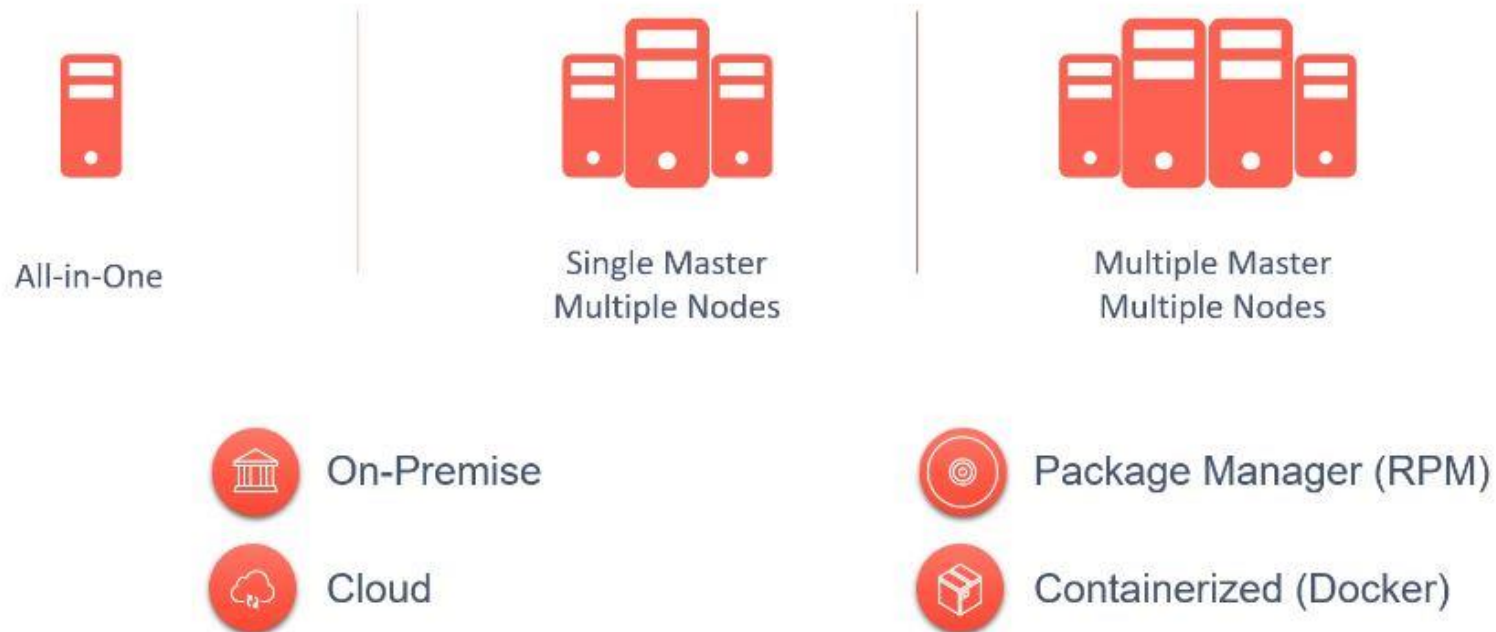


CLI

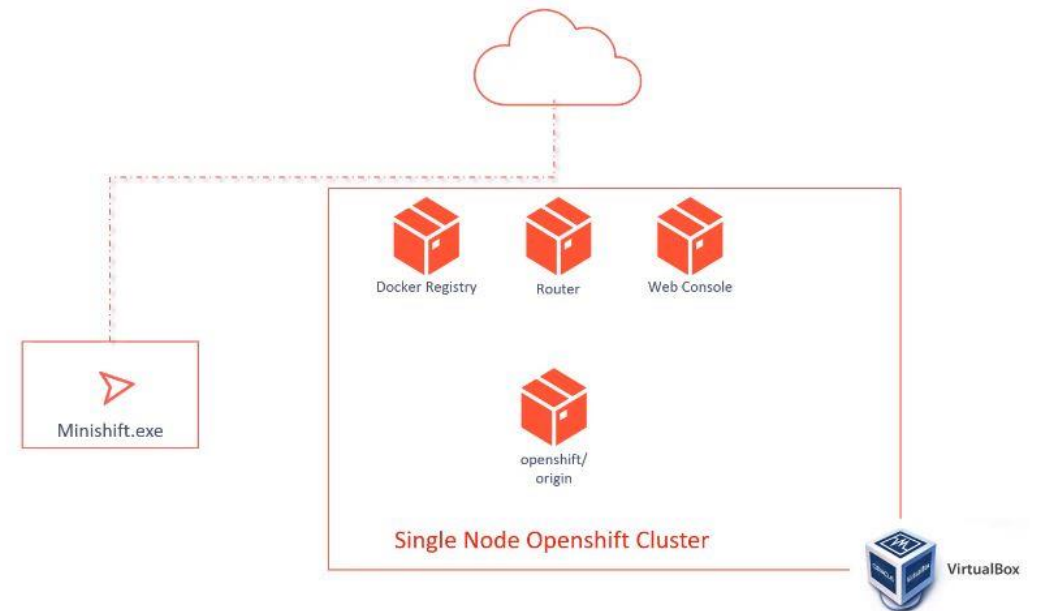


REST API

COMPOSANTS D'OPENSIFT (5/6): METHODE D'INSTALLATION



COMPOSANTS D'OPENS SHIFT (6/6): MINISHIFT



TP-0: PRÉREQUIS D'INSTALLATION

- Installation de virtualbox
- Installation de vagrant
- Créez un compte github
- Et forkez les deux projets utilisés pour la formation

<https://github.com/eazytrainingfr/simple-webapp.git>

<https://github.com/eazytrainingfr/simple-webapp-docker.git>

TP-1: INSTALLATION D'OPENSIFT

- Déploiement d'opensift avec vagrant
- Connexion via la console et la cli

PLAN



Présentation du formateur

Introduction au DevOps et PaaS

Composants d'OpenShift

Utilisateurs et Projets

Build et Déploiement

Gestion du réseau

Gestion du stockage

Scaling avec Openshift

Introduction au catalogue

Mini-projet

UTILISATEURS ET PROJETS (1/3): UTILISATEURS



Regular

`developer`



System

`system:admin`

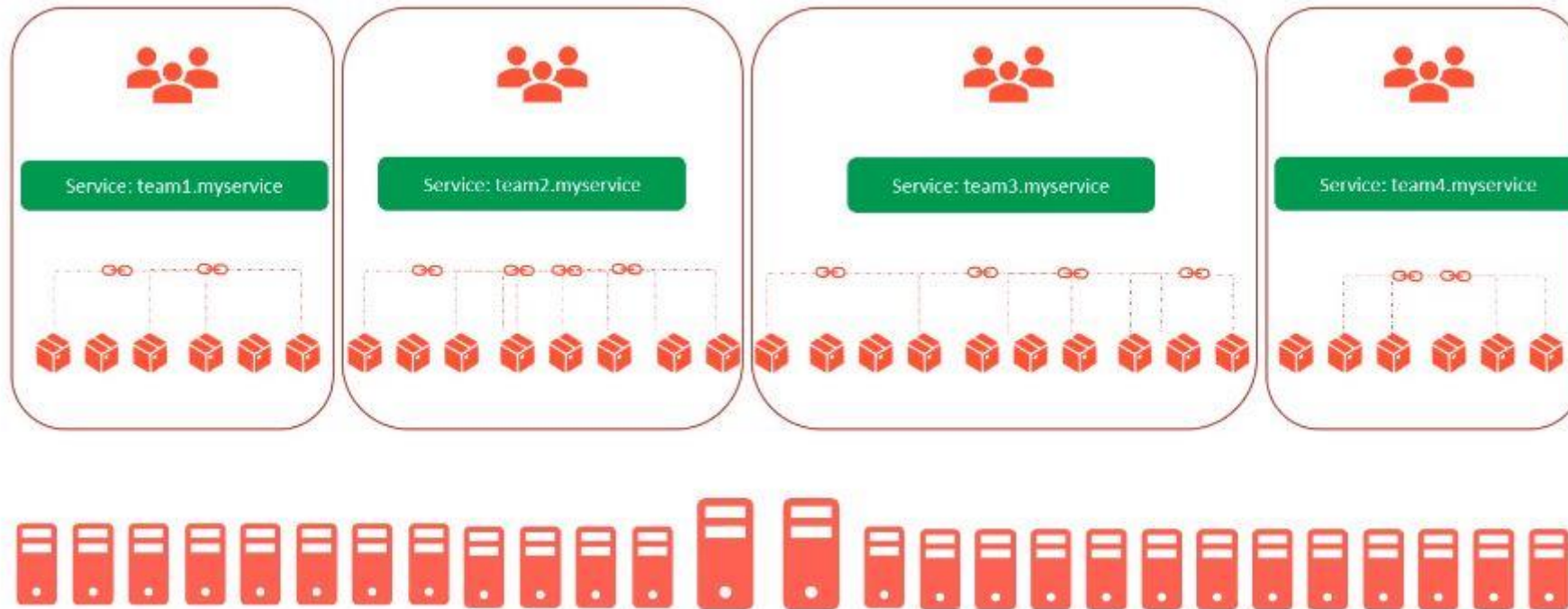
`system:master`



Service

`system:serviceaccount:proj1:db_user`

UTILISATEURS ET PROJETS (2/3): PROJETS





Il ne peut pas se
connecter en
console



Il a tous les droits
sur le cluster

UTILISATEURS ET PROJETS (3/3): ADMIN

TP-2: CRÉATION DE COMPTE ET PROJET

- Créez à l'aide de la console votre compte nominatif de type Regular user
- Créez un projet dans votre compte nommé eazytraining
- Connectez-vous via la cli à openshift et récupérez la liste des utilisateurs et des projets
- Récupérez ses mêmes infos via l'API (utilisation de curl avec token)

PLAN



Présentation du formateur

Introduction au DevOps et PaaS

Composants d'OpenShift

Utilisateurs et Projets

Build et Déploiement

Gestion du réseau

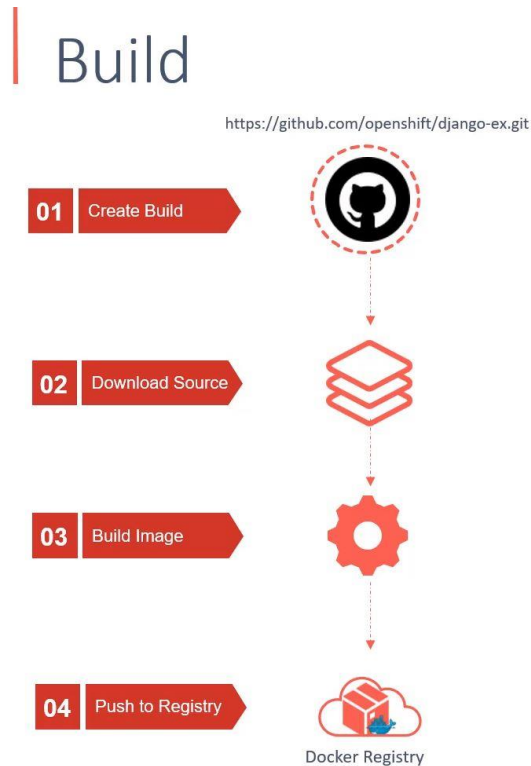
Gestion du stockage

Scaling avec Openshift

Introduction au catalogue

Mini-projet

BUILD ET DÉPLOIEMENT (1/10): BUILD



OPENSIFT ORIGIN

test-project

Overview Applications Builds

Builds [Learn More](#)

Filter by label

Name	Last Build	Status
django-ex	#2	✓ Complete

Details Environment Logs Events

Status: ✓ Complete Log from Apr 26, 2018 6:41:02 PM to Apr 26, 2018 6:43:37 PM

```
1 Cloning "https://github.com/openshift/django-ex.git" ...
2 Commit: 37f7fc41432b9c07265c5896a4fb226caa870427 (Merge pull request #115 from hhorak/python-3.6)
3 Author: Monza Horak <hhorak@redhat.com>
4 Date: Tue Apr 24 18:48:25 2018 +0200
5 ---> Installing application source ...
6 ---> Installing dependencies ...

21 Installing collected packages: pytz, django, sqlparse, django-debug-toolbar, gunicorn, psycopg2, whitenoise
22 Successfully installed django-1.11.12 django-debug-toolbar-1.8 gunicorn-19.4.5 psycopg2-2.7.3.1 pytz-2018.4 sqlparse-0.2.4 whitenoise-3.3.1
23 You are using pip version 9.0.1, however version 10.0.1 is available.
24 You should consider upgrading via the 'pip install --upgrade pip' command.
25 ---> Collecting Django static files ...

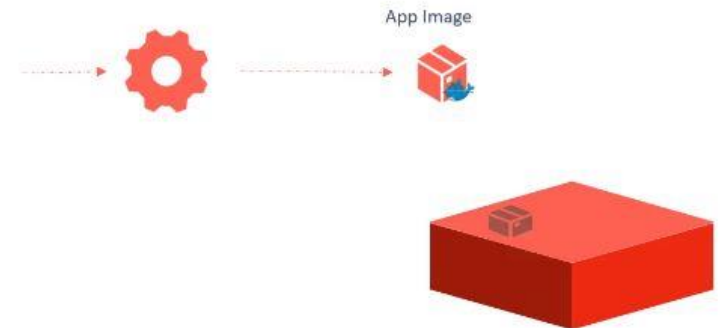
205 Pushing image 172.30.1.1:5000/test-project/django-ex:latest ...
206 Pushed 0/10 layers, 10% complete
207 Pushed 1/10 layers, 11% complete
208 Pushed 2/10 layers, 21% complete
215 Pushed 9/10 layers, 97% complete
216 Pushed 10/10 layers, 100% complete
217 Push successful
```

BUILD ET DÉPLOIEMENT (3/10): BUILD – S2I

Build Strategy

2 Source-To-Image (S2I)

```
app.py
1 import os
2 from flask import Flask
3 app = Flask(__name__)
4
5 @app.route("/")
6 def main():
7     return "Welcome!"
8
9 @app.route('/how are you')
10 def hello():
11     return 'I am good, how about you?'
12
13 if __name__ == "__main__":
14     app.run(host="0.0.0.0", port=8080)
```



BUILD ET DÉPLOIEMENT (3/10): BUILD - DOCKER

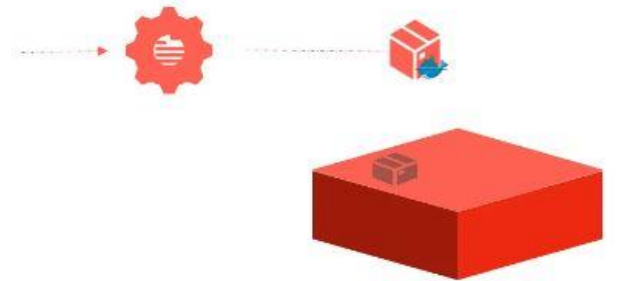
Build Strategy

1 Docker Build

app.py

Dockerfile

```
1 FROM ubuntu:16.04
2
3 RUN apt-get update && apt-get install -y python python-pip
4
5 RUN pip install flask
6
7 COPY app.py /opt/
8
9 ENTRYPOINT FLASK_APP=/opt/app.py flask run --host=0.0.0.0
```



BUILD ET DÉPLOIEMENT (4/10): BUILD - CUSTOM

| Build Strategy

3 Custom Build

Java



app.jar

Python



app.tar.gz

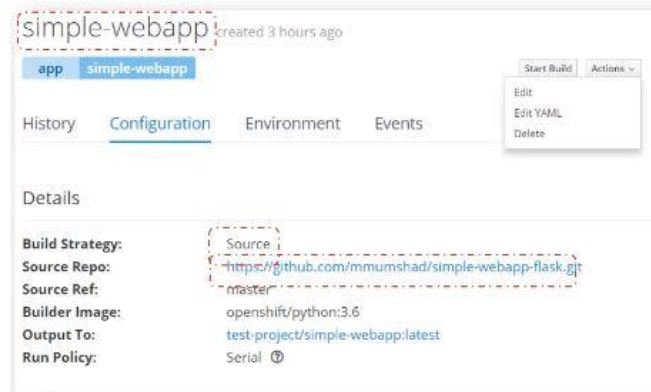
Ruby



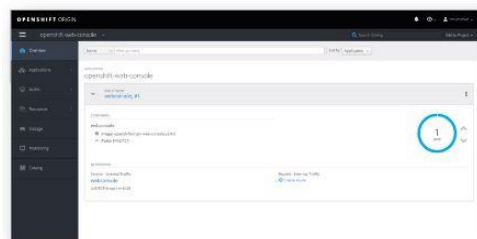
app.gem

BUILD ET DÉPLOIEMENT (5/10): BUILD CONFIG

View Build Config



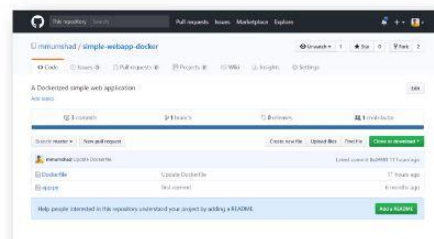
```
S2i-build-config.yaml
kind: "BuildConfig"
apiVersion: "v1"
metadata:
  name: "simple-webapp"
spec:
  runPolicy: "Serial"
  triggers:
  - type: "GitHub"
    github:
      secret: "b5e471d57f79f52e"
  - type: "Generic"
    generic:
      secret: "4be5b473f9985dcf"
  - type: "ImageChange"
  source:
    git:
      uri: "https://github.com/mmumshad/simple-webapp-flask.git"
  strategy:
    type: Source
    sourceStrategy:
      from:
        kind: "ImageStreamTag"
        name: "python:3.6"
  output:
    to:
      kind: "ImageStreamTag"
      name: "simple-webapp:latest"
```

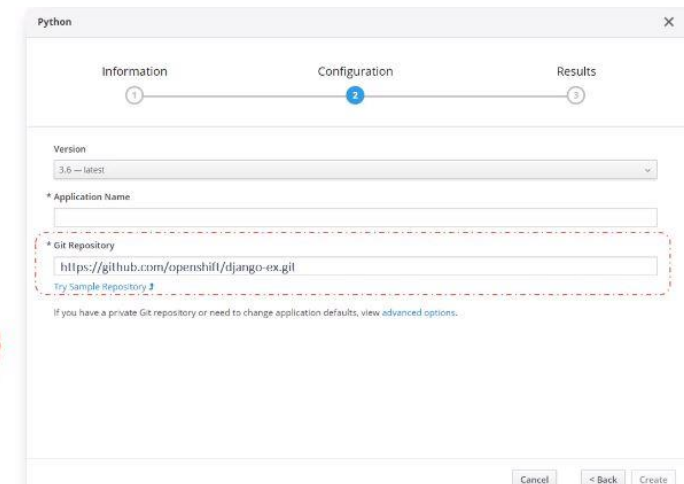
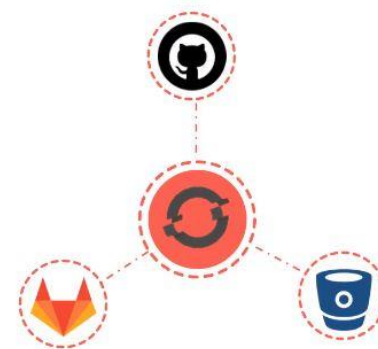
Openshift



Automated Build

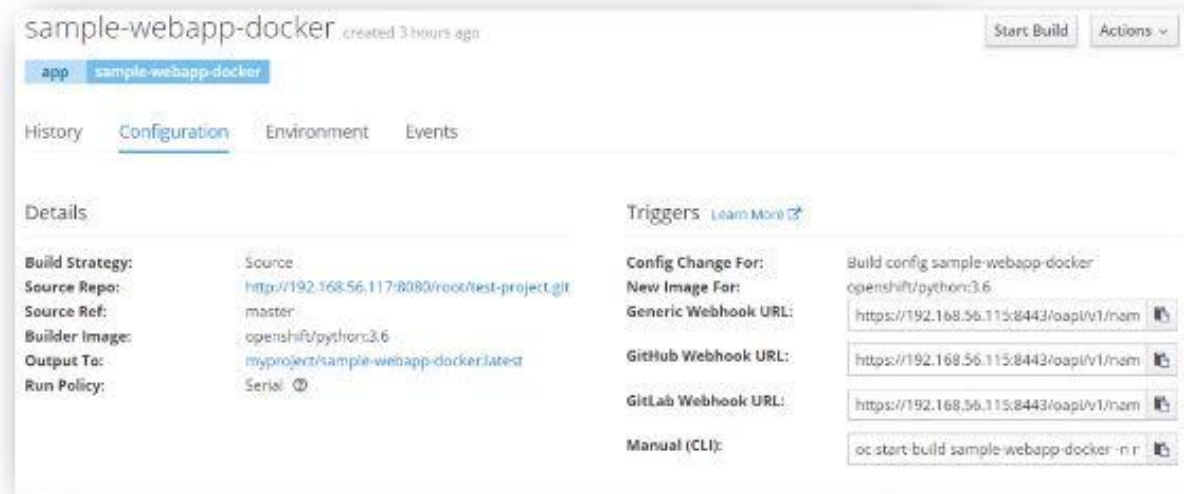


Code Repository

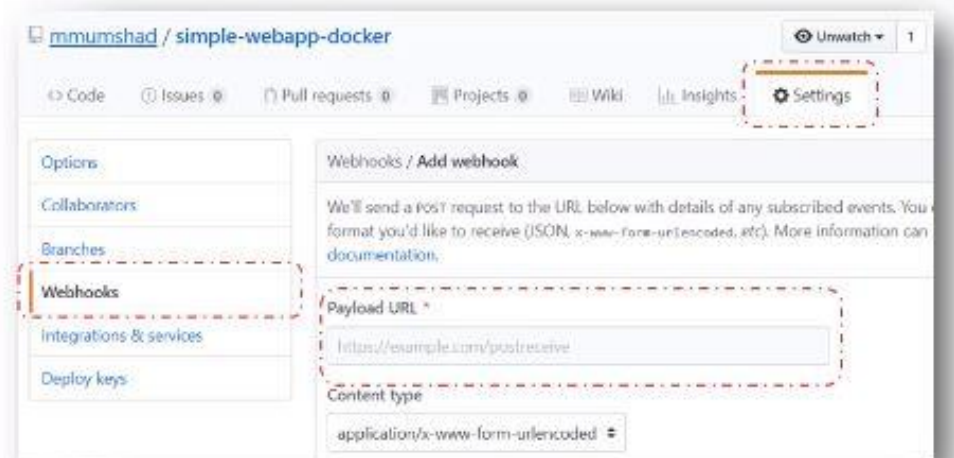


BUILD ET DÉPLOIEMENT (6/10): BUILD TRIGGER ET PULL

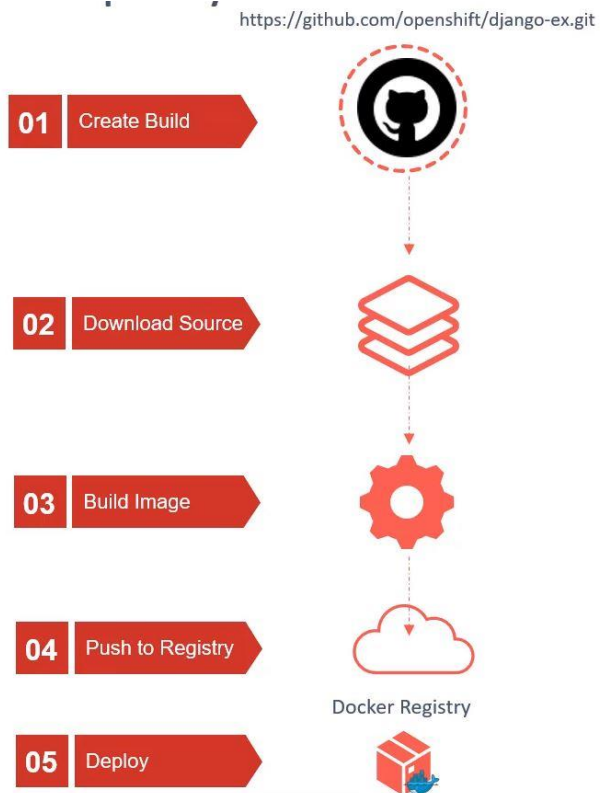
BUILD ET DÉPLOIEMENT (7/10): BUILD WEBHOOK



OpenShift



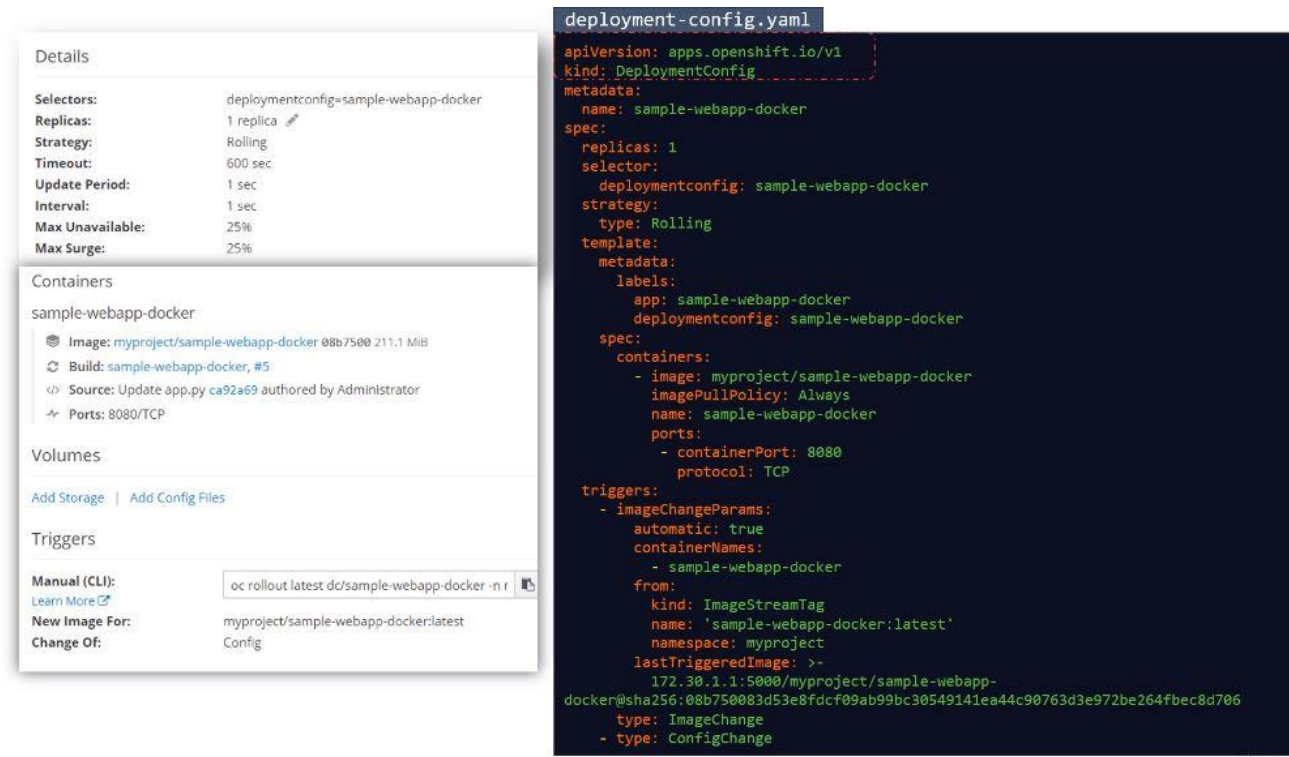
Code Repository



BUILD ET DÉPLOIEMENT (8/10): DÉPLOIEMENT

- Recreate
- Rolling Update
- Blue/Green
- A/B

BUILD ET DÉPLOIEMENT (9/10): DÉPLOIEMENT CONFIG



The image displays two side-by-side screenshots. The left screenshot shows the OpenShift console interface for a deployment configuration named 'sample-webapp-docker'. The right screenshot shows the corresponding 'deployment-config.yaml' file.

OpenShift Console Details:

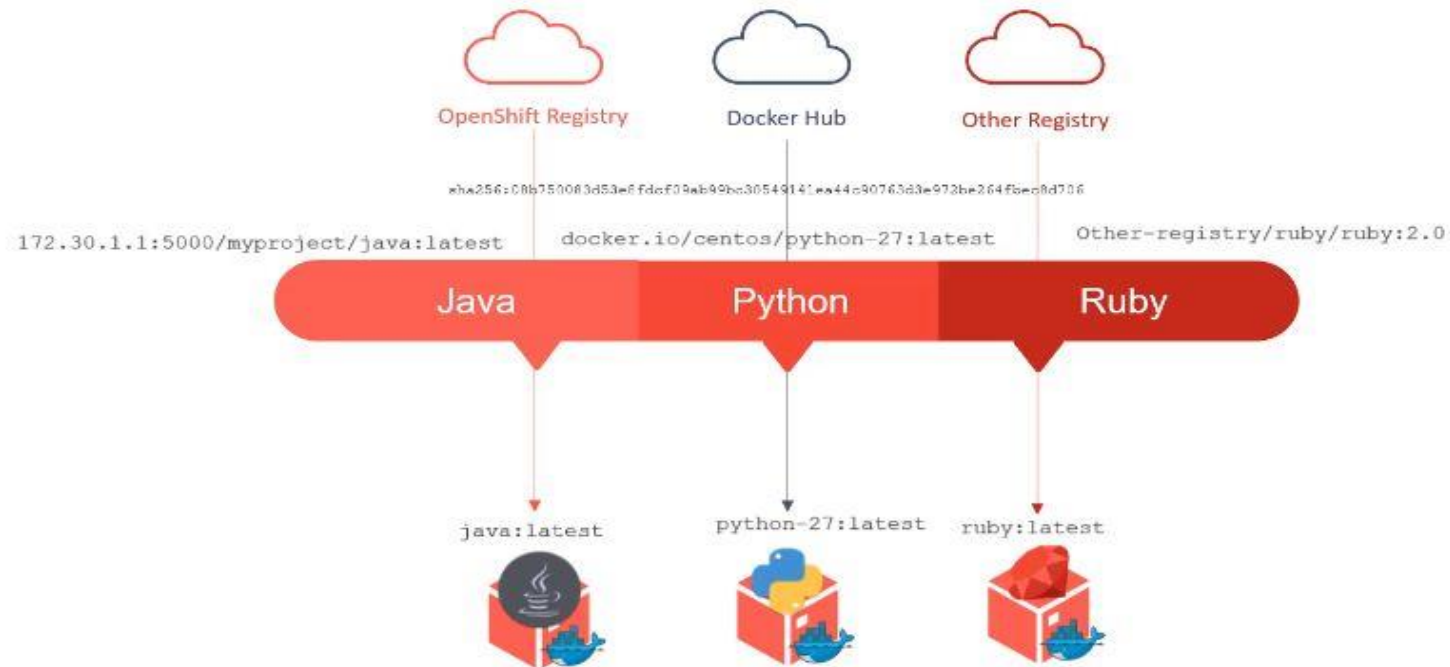
- Details:**
 - Selectors: deploymentconfig=sample-webapp-docker
 - Replicas: 1 replica
 - Strategy: Rolling
 - Timeout: 600 sec
 - Update Period: 1 sec
 - Interval: 1 sec
 - Max Unavailable: 25%
 - Max Surge: 25%
- Containers:**
 - sample-webapp-docker
 - Image: myproject/sample-webapp-docker 08b7500 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`
 - New Image For:** myproject/sample-webapp-docker:latest
 - Change Of:** Config

deployment-config.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'
            namespace: myproject
          lastTriggeredImage: >=
            172.30.1.1:5000/myproject/sample-webapp-
            docker@sha256:08b750083d53e8fdcf09ab99bc30549141ea44c90763d3e972be264fbec8d706
          type: ImageChange
        - type: ConfigChange
```

BUILD ET DÉPLOIEMENT (10/10): IMAGE STREAM

| Image Streams



TP-3: DÉPLOYEZ VOS PREMIÈRES APPLICATIONS

- Tous les déploiements doivent être fait dans votre compte et via le projet eazytraining
- Créez votre premier déploiement (nommé django) à partir de l'application python <https://github.com/sclorg/django-ex.git>
- Vérifiez que le déploiement (nommé django) est bien accessible via l'url fourni dans l'onglet deployment (ajuster votre DNS si besoin)
- Créez un second déploiement (nommé simple-webapp) à partir du code (build strategy = S2i) : <https://github.com/<votre compte>/simple-webapp.git>
- Vérifiez que le déploiement est bien accessible via l'url fourni dans l'onglet deployment (ajuster votre DNS si besoin)
- Mettez en place un webhook afin de builder et dedéployer automatiquement après toute modification sur votre application simple-webapp-docker via github. Modifiez dans l'URL du webhook le port 8443 par 6443 si vous utilisez l'environnement de labs EAZYTraining
- Testez et validez le fonctionnement du webhook

PLAN



Présentation du formateur

Introduction au DevOps et PaaS

Composants d'OpenShift

Utilisateurs et Projets

Build et Déploiement

Gestion du réseau

Gestion du stockage

Scaling avec Openshift

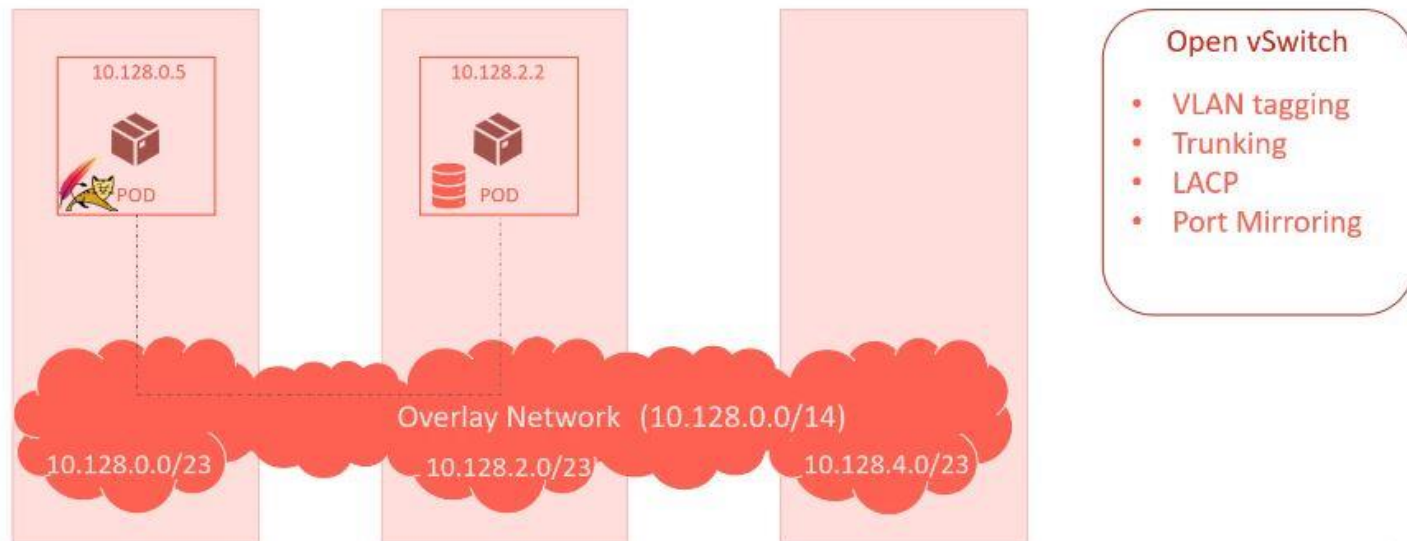
Introduction au catalogue

Mini-projet

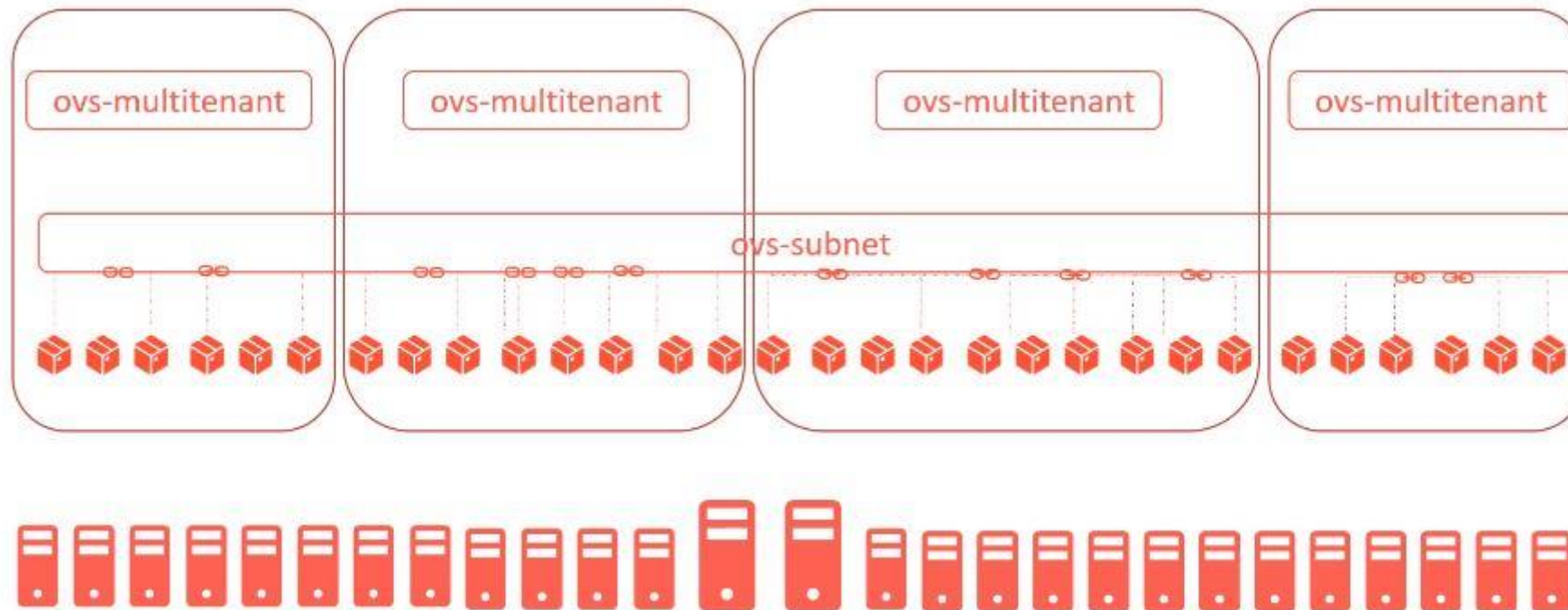
GESTION DU RESEAU (1/6): ARCHITECTURE

```
> oc get pods -o wide
```

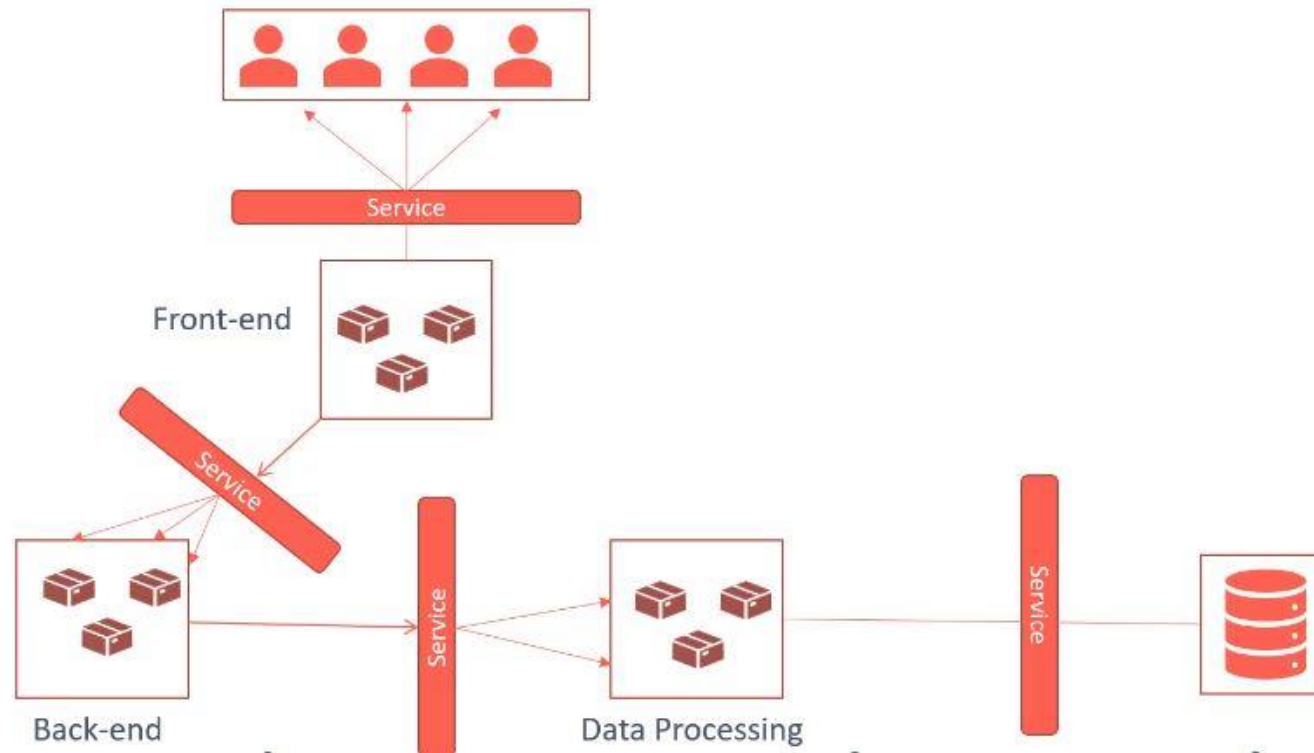
my-web-app	1/1	Running	0	2d	10.128.0.5	localhost
my-sql-db	1/1	Running	0	1d	10.128.2.2	localhost



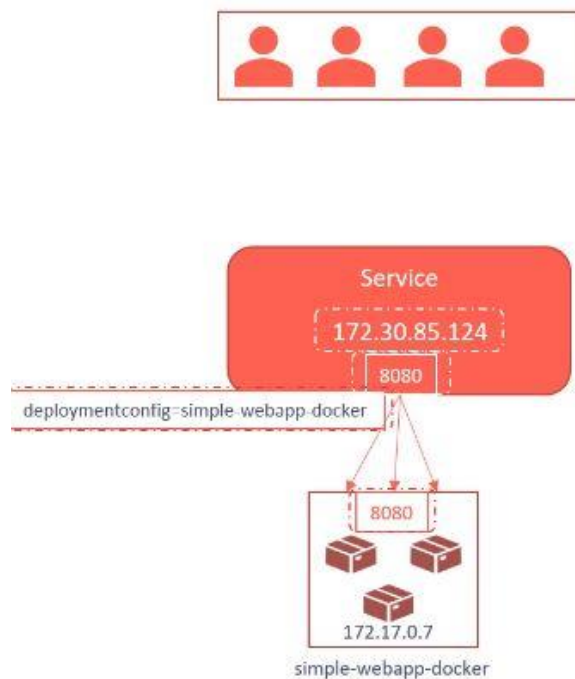
GESTION DU RESEAU (2/6): OVS



GESTION DU RESEAU (3/6): SERVICE INTRODUCTION



GESTION DU RESEAU (4/6): SERVICE CLUSTER IP



simple-webapp-docker created 2 days ago

app simple-webapp-docker

Details Events

Selectors: deploymentconfig=simple-webapp-docker

Type: ClusterIP

IP: 172.30.85.124

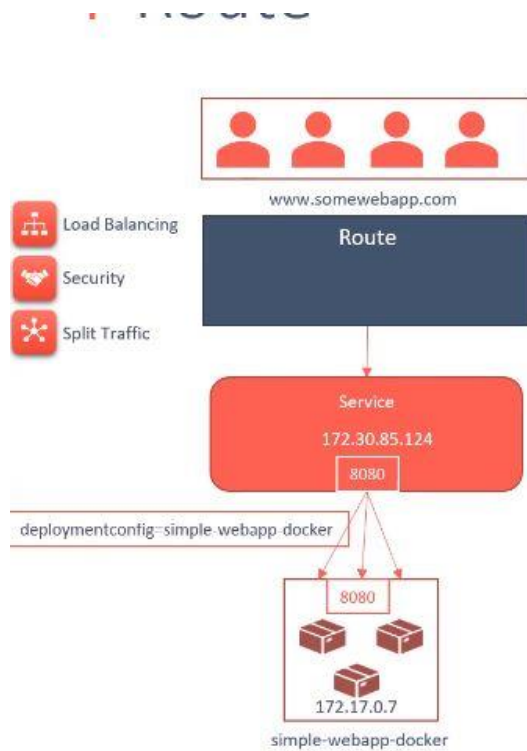
Hostname: simple-webapp-docker.test-project.svc ⓘ

Session affinity: None

Traffic

Route	Service Port	Target Port
simple-webapp-docker	8080/TCP (8080-tcp)	8080

GESTION DU RESEAU (5/6): ROUTE



Create Route

Routing is a way to make your application publicly visible.

* **Name**

A unique name for the route within the project.

Hostname

Public hostname for the route. If not specified, a hostname is generated.

Path

Path that the router watches to route traffic to the service.

* **Service**

Service to route to.

Target Port

Target port for traffic.

Alternate Services

☐ Split traffic across multiple services

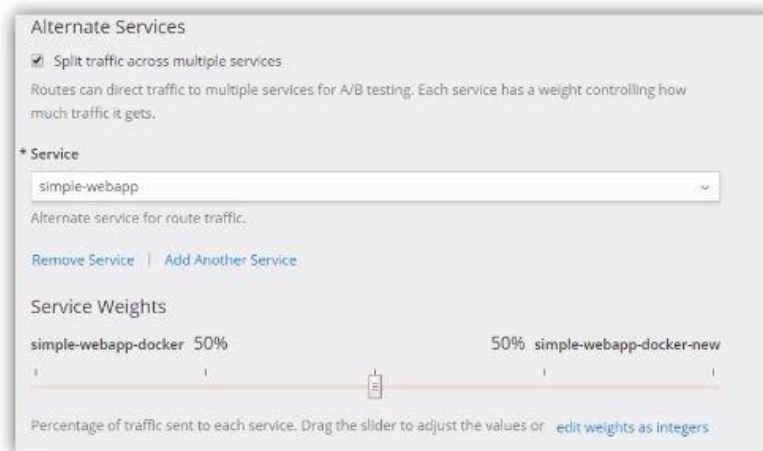
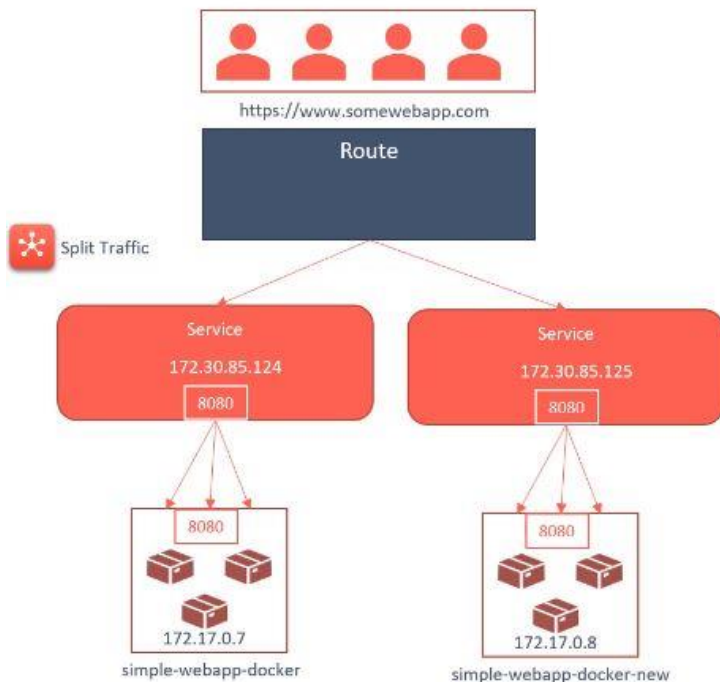
Routes can direct traffic to multiple services for A/B testing. Each service has a weight controlling how much traffic it gets.

Security

☐ Secure route

Routes can be secured using several TLS termination types for serving certificates.

GESTION DU RESEAU (6/6): ROUTE – SLIP TRAFFIC



TP-4: CREATION DE SERVICE ET ROUTE

- Créez un build à partir de celui de simple-webapp, ce dernier sera de type Docker Build contrairement à celui de simple-webapp, ainsi nous vous conseillons de partir du fichier yaml du build de simple-webpp afin de l'adapter à simple-webapp-docker, vous devrez re-importer le fichier après modification
- Le repo à utiliser pour ce build est le suivant: <https://github.com/<votre compte>/simple-webapp-docker.git>
- Créez un déploiement toujours à partir de celui de simple-webapp
- Créez un service de type cluster IP pour exposer votre application dans le cluster, vous pouvez vous inspirer de celui de simple-webapp ou à partir des exemples disponible sur internet
- Créez également une route afin d'atteindre l'application depuis l'extérieur comme c'est le cas pour simple-webpp
- Afin de faciliter la lisibilité des ressources que vous allez créer, utilisez comme nom simple-webpp-docker
- Vérifiez que tout fonctionne bien et vos votre application est disponible

PLAN



Présentation du formateur

Introduction au DevOps et PaaS

Composants d'OpenShift

Utilisateurs et Projets

Build et Déploiement

Gestion du réseau

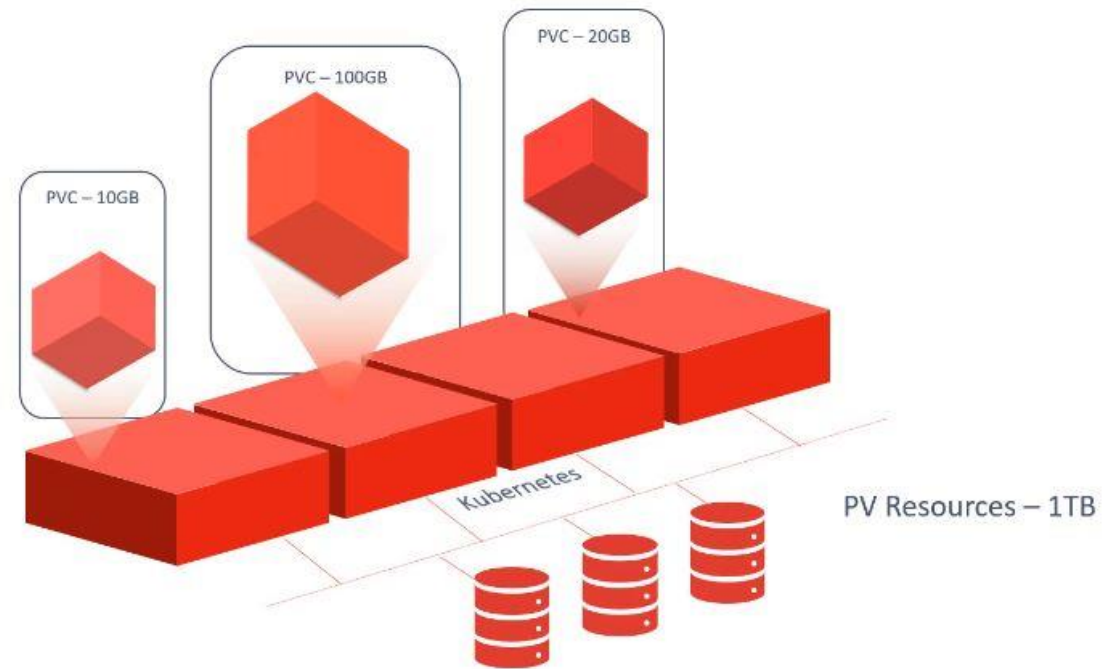
Gestion du stockage

Scaling avec Openshift

Introduction au catalogue

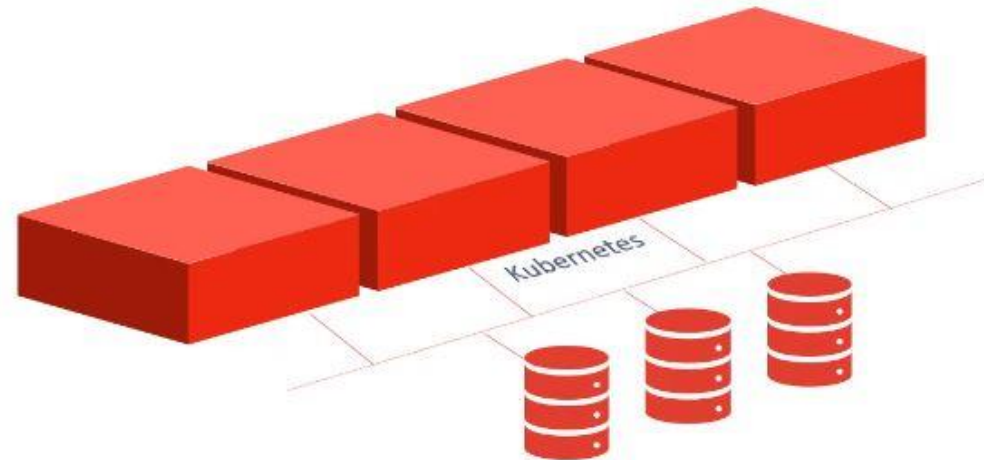
Mini-projet

GESTION DU STOCKAGE (1/2): VOLUME



GESTION DU STOCKAGE (2/2): PLUGINS

- Local
- iSCSI
- Fibre Channel
- NFS
- GlusterFS
- Ceph RDB
- OpenStack Cinder
- AWS Elastic Block Store
- GCE Persistent Disk
- Azure Disk
- Azure File
- VMWare vSphere



TP-5: RENDEZ VOS DONNEES PERSISTANTE

- Créez un storage nommé data-storage
- Attachez ce storage à votre deployment simple-webapp-docker dans /data
- Vérifiez à l'aide de la console terminal que le montage a bien été effectué

PLAN



Présentation du formateur

Introduction au DevOps et PaaS

Composants d'OpenShift

Utilisateurs et Projets

Build et Déploiement

Gestion du reseau

Gestion du stockage

Scaling avec Openshift

Introduction au catalogue

Mini-projet

SCALING AVEC OPENSIFT (1/1):

Sample Project

SERVICE: **mongodb** 27017/TCP → 27017

DEPLOYMENT: MONGODB, #1 2 minutes ago from config change

1 pod

CONTAINER: MONGODB
Image: rhsc/mongodb-26-rhel7
Ports: 27017/TCP

SERVICE: NODEJS-MONGODB-EXAMPLE 8080/TCP → 8080

nodejs-rhel-ose.vagrant.dev

✓ Build nodejs-mongodb-example #1 completed. [View Log](#) [Dismiss](#)

DEPLOYMENT: NODEJS-MONGODB-EXAMPLE, #1 a few seconds ago from image change

1 pod

CONTAINER: NODEJS-MONGODB-EXAMPLE
Image: ose-test-project/nodejs-mongodb-example (8fbd96e)
Build: #1 from <https://github.com/ContainerSolutions/nodejs-ex.git>
Ports: 8080/TCP



TP-6: SCALEZ VOTRE APPLICATION

- Passez le nombre de replica de l'application à 3
- Accédez à l'application via la route et actualisez plusieurs fois la page ! Que remarquez vous ? À quoi cela est dû ?

PLAN



Présentation du formateur

Introduction au DevOps et PaaS

Composants d'OpenShift

Utilisateurs et Projets

Build et Déploiement

Gestion du réseau

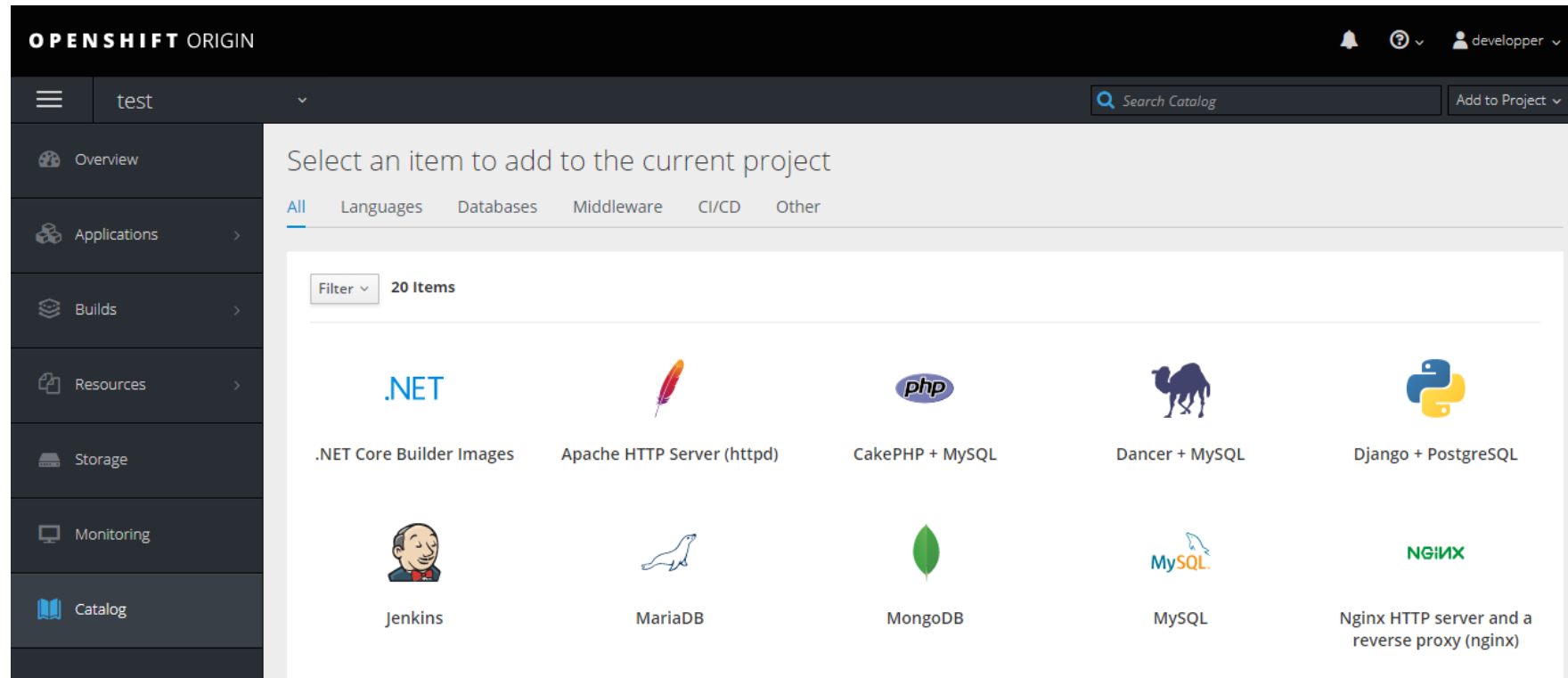
Gestion du stockage

Scaling avec Openshift

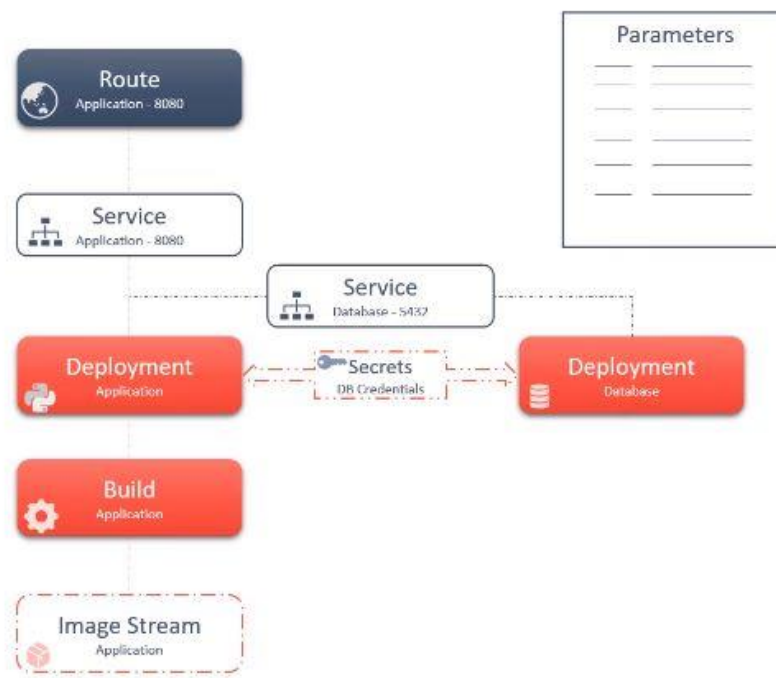
Introduction au catalogue

Mini-projet

INTRODUCTION AU CATALOGUE (1/2): VUE D'ENSEMBLE



INTRODUCTION AU CATALOGUE (2/2): TEMPLATE




Django + PostgreSQL

TP-7: RAJOUTEZ UN TEMPLATE AU CATALOGUE

- Rajoutez le template Redis *sans* stockage persistant :
<https://github.com/openshift/origin/tree/master/examples/db-templates>
- Déployez Redis à partir de ce template
- Consulter le fichier json afin de mieux le comprendre

PLAN



Présentation du formateur

Introduction au DevOps et PaaS

Composants d'OpenShift

Utilisateurs et Projets

Build et Déploiement

Gestion du réseau

Gestion du stockage

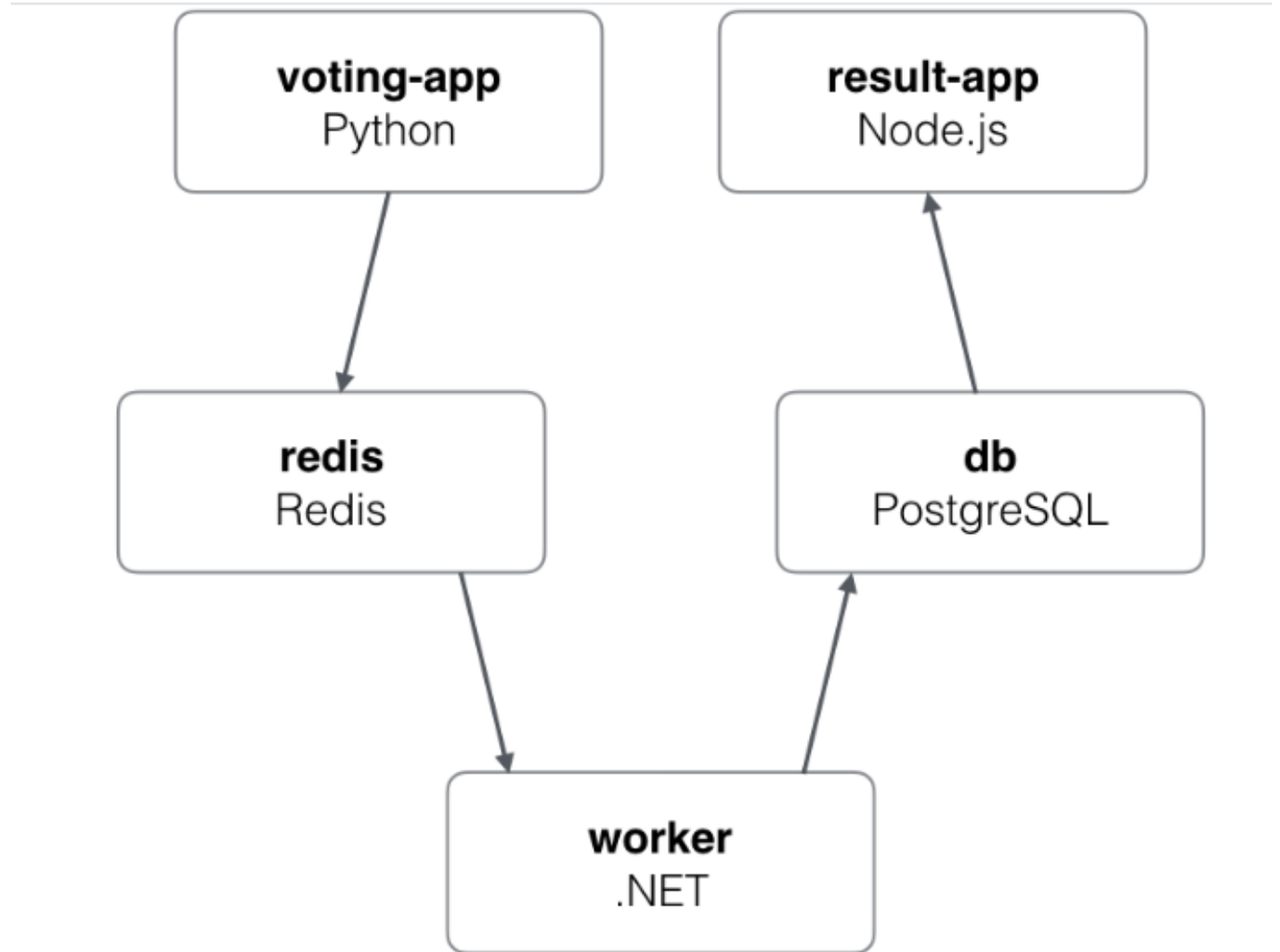
Scaling avec Openshift

Introduction au catalogue

Mini-projet

MINI-PROJET: VOTING APP

- Déployez à l'aide d'OpenShift l'application :
<https://github.com/mmumshad/example-voting-app.git>
- A la fin de votre travail, envoyez nous le lien de vos endpoint à easytrainingfr@gmail.com et nous vous dirons si votre solution respecte les bonnes pratiques



MERCI

