

Openshift

Kubernetes with a human face

Vadim Rutkovsky

vrutkovs@redhat.com

Openshift, where's Kubernetes?



Openshift - batteries included

- Internal OAuth server for authentication
- Internal container registry
- ImageStreams + BuildConfigs + DeploymentConfigs
- CI/CD via Jenkins Pipelines



DevOps 101

```
10 git commit  
20 git push  
30 ??? some ops magic ???  
40 GOTO 10
```

Look mom, no Dockerfile!

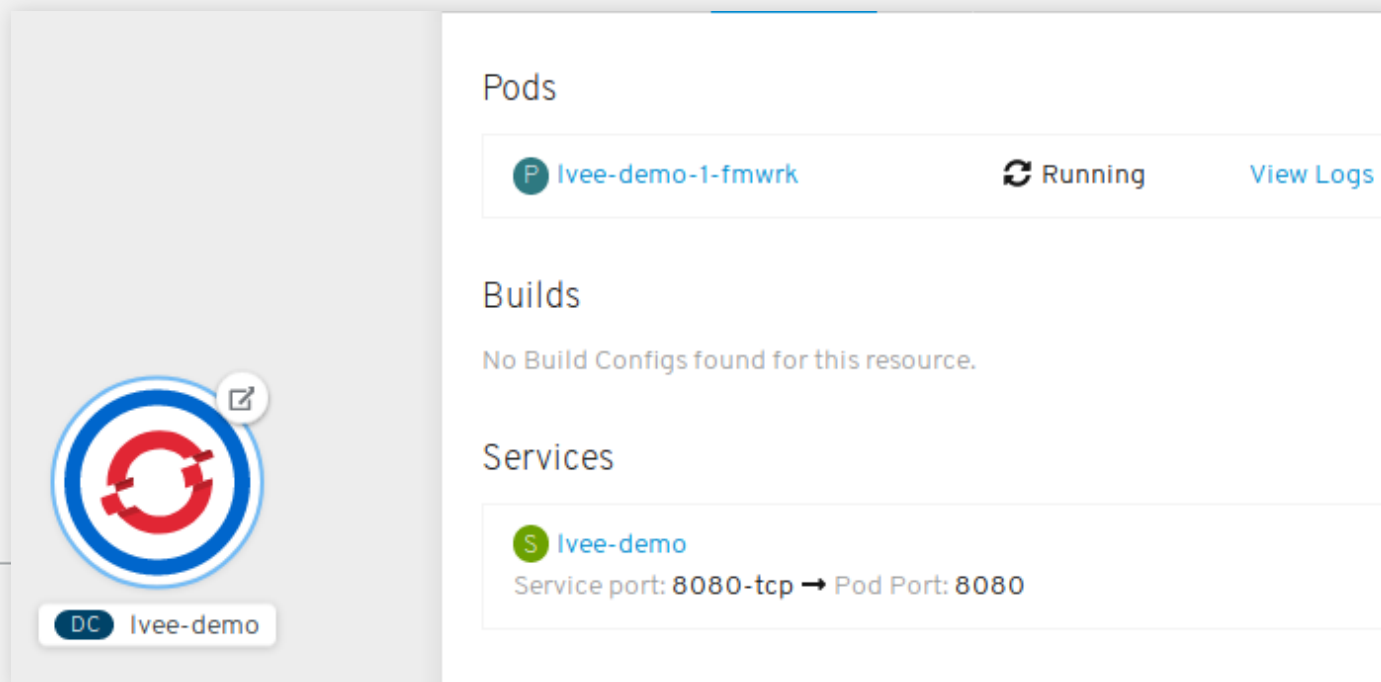
```
$ oc login https://cloud.vrutkovs.eu -t  
$ oc use demovrproject lvee  
$ oc new-app \  
  --name=lvee-demo \  
  https://github.com/vrutkovs/openshift-demo  
$ oc expose svc/demo --  
host=demo.cloud.vrutkovs.eu
```

Build log

```
Cloning "https://github.com/vrutkovs/openshift-demo" ...
  Commit:      b74070426bbba32ba085846804b8b6b909880eeb (Simplify
  Author:      Vadim Rutkovsky <vrutkovs@redhat.com>
  Date:        Fri May 4 23:37:38 2018 +0200
--> Installing application source ...
--> Installing dependencies ...
Collecting aiohttp==2.3.10 (from -r requirements.txt (line 1))
Downloading https://files.pythonhosted.org/packages/7e/af/b2c6b59
Collecting yarl>=1.0.0 (from aiohttp==2.3.10->-r requirements.txt
Downloading https://files.pythonhosted.org/packages/61/67/df71b36
...
Installing collected packages: idna, multidict, yarl, idna-ssl, c
Running setup.py install for idna-ssl: started
Running setup.py install for idna-ssl: finished with status 'done
Successfully installed aiohttp-2.3.10 async-timeout-3.0.0 chardet



Pushing image 172.30.16.196:5000/lvee/lvee-demo:latest ...
Pushed 0/6 layers, 3% complete
...
Pushed 6/6 layers, 100% complete
Push successful
```

Web Console



The screenshot shows a web console interface for a resource named "lvee-demo". On the left side, there is a circular icon with a red and blue target-like pattern and a small edit icon. Below the icon is a label "DC lvee-demo". The main content area is divided into three sections: "Pods", "Builds", and "Services".


Pods

 lvee-demo-1-fmwrk	 Running	View Logs
---	---	---------------------------

Builds

No Build Configs found for this resource.

Services

 lvee-demo	Service port: 8080-tcp → Pod Port: 8080
---	---

Dockerfile + route settings in YAML

```
$ oc new-project lvee-custom  
$ oc new-app --name=lvee-custom \  
    http://github.com/vrutkovs/openshift-demo#custom-  
$ oc create -f route.yaml
```


CI/CD with Jenkins Pipelines

```
$ oc new-project pipelines
$ oc new-app --name=jenkins-pipeline \
  http://github.com/vrutkovs/openshift-
demo#jenkins
```

```
stage("Build") {
  openshiftBuild
  buildConfig: "pipeline-app", showBuildLogs: "true"
}
stage("Deploy to dev") {
  openshiftDeploy deploymentConfig: "pipeline-app"
}

stage("Smoketest") {
  sh "curl -kLvs
    http://pipeline-app.pipelines.svc:8080/Minsk |
    grep 'Hello, Minsk'"
}

stage("Deploy to staging") {
  openshiftTag
  srcStream: "pipeline-app", srcTag: "latest",
  destinationStream: "pipeline-app", destinationTag: "staging"
  openshiftDeploy
  deploymentConfig: "pipeline-app",
  namespace: "staging-namespace"
}
```

Pipelines » jenkins-pipeline » #3

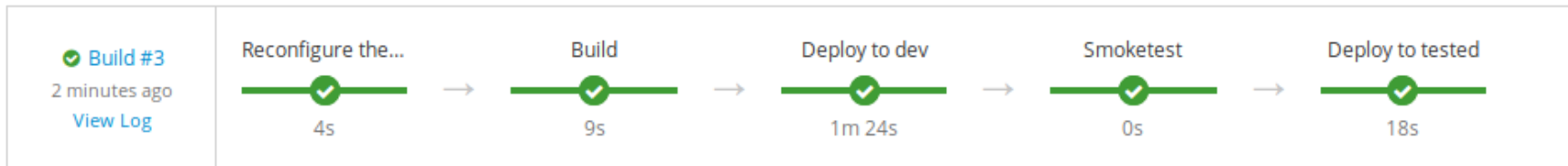
jenkins-pipeline-3 created 2 minutes ago

Rebuild

Actions ▾

app jenkins-pipeline buildconfig jenkins-pipeline openshift.io/build-config.name jenkins-pipeline [More labels...](#)

Details [Events](#)



Status

Status: ✓ Complete – [View Log](#)

Jenkins Pipeline view

The screenshot displays the Jenkins Pipeline view for a pipeline named 'jenkins-pipeline 3'. The interface includes a top navigation bar with tabs for 'Pipeline', 'Changes', 'Tests', and 'Artifacts'. Below the navigation bar, the current branch is shown as 'Branch: -' and the commit is 'Commit: -'. The pipeline execution details show a duration of '1m 59s' and a change by 'vrtkovs' from 'OpenShift Build pipelines/jenkins-pipeline-3' from a GitHub repository. A description box contains the text: 'OpenShift Build pipelines/jenkins-pipeline-3 from http://github.com/vrtkovs/openshift-demo'. The pipeline execution is visualized as a horizontal line with several stages: 'Start', 'Reconfigure the namespace', 'Build', 'Deploy to dev', 'Smoketest', 'Deploy to tested', and 'End'. Each stage is marked with a green checkmark, indicating successful completion.

✓ pipelines / pipelines/jenkins-pipeline 3 Pipeline Changes Tests Artifacts ↻ ⚙️ ↗ Logout ✕

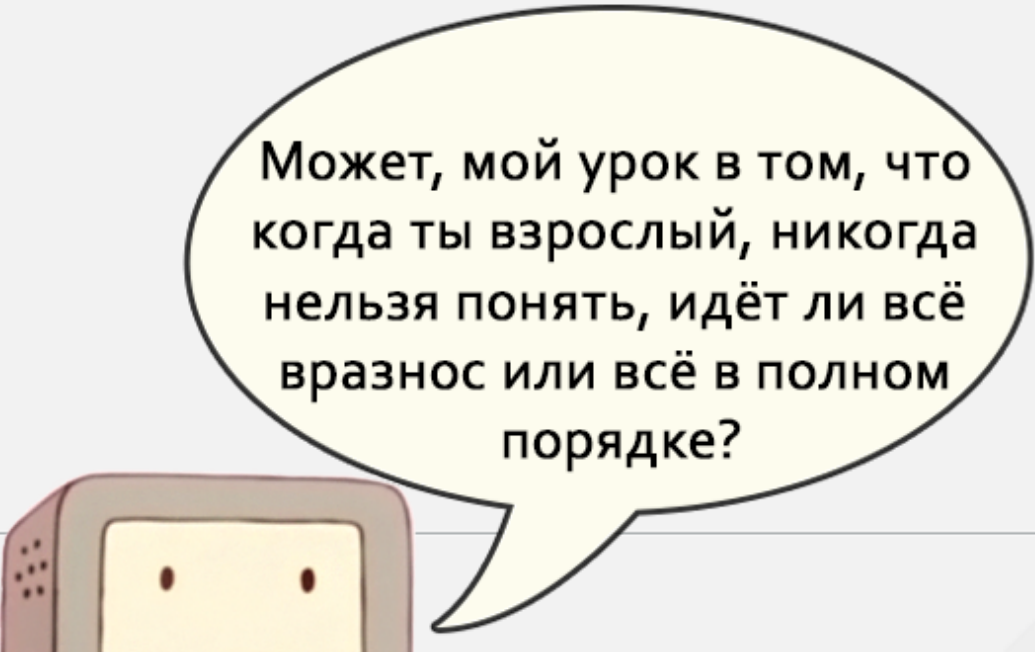
Branch: - 1m 59s Changes by vrtkovs

Commit: - a minute ago OpenShift Build pipelines/jenkins-pipeline-3 from http://github.com/vrtkovs/openshift-de...

Description OpenShift Build pipelines/jenkins-pipeline-3 from http://github.com/vrtkovs/openshift-demo

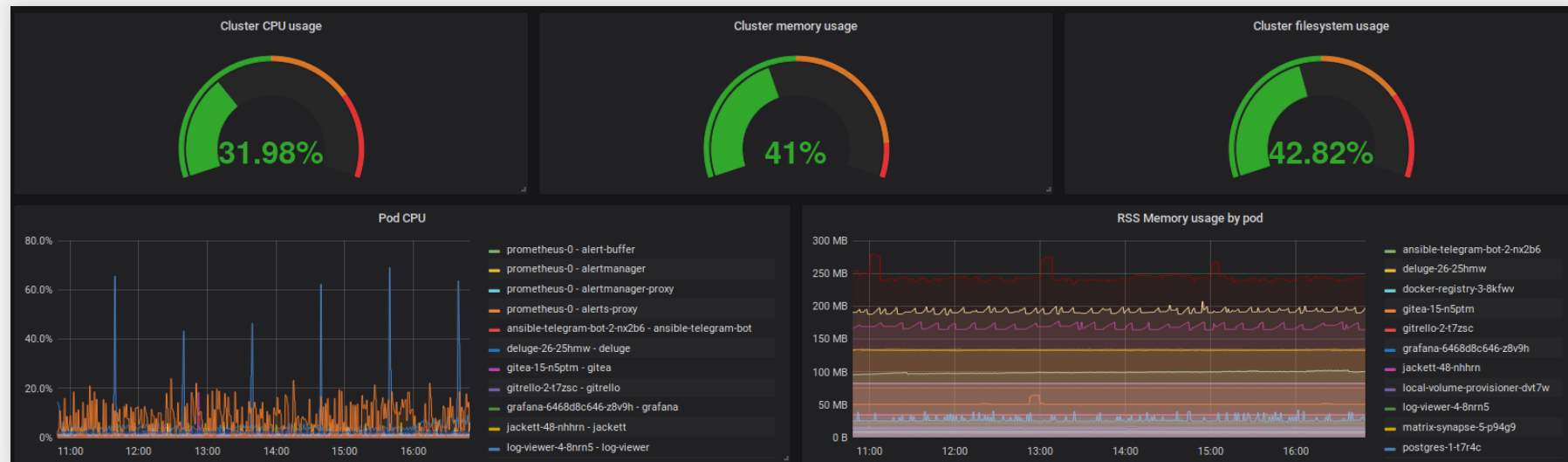
Start Reconfigure the namespace Build Deploy to dev Smoketest Deploy to tested End

Monitoring and Metrics

A cartoon character with a yellow face and two small black dots for eyes is shown from the chest up. It has a large, light-yellow speech bubble coming from its mouth. The speech bubble contains Russian text. The character is positioned at the bottom center of the slide.

Может, мой урок в том, что
когда ты взрослый, никогда
нельзя понять, идёт ли всё
вразнос или всё в полном
порядке?

Prometheus + Grafana



See you later, operator

Operators - k8s pattern, application connects to k8s API server, uses CRDs to create k8s objects and may actively watch them

Prometheus Operator maintains and configures Prometheus cluster

Vitesse Operator creates, scales and backs up MySQL containers in kubernetes

Where Do I Find Operators?

<https://operatorhub.io>

Operator Lifecycle Manager takes care of operator provisioning, update and configuration

OpenShift-specific operator - MachineConfig

MachineConfig - custom object, which contains a list of encoded files and systemd units

MachineConfigs are assembled into MachineConfigPools, assigned to a node group

MachineConfigOperator runs a daemon on the hosts and synchronises files and systemd unit state with the k8s object specification

OpenShift-specific operator - Machine API

Machine API Operator takes care of additional nodes provisioning

Additional entities - Machine and MachineSet - is introduced to keep the info about desired node configs and quantity

MachineSet can be scaled, creating a new Machine instance. A new Machine joins the cluster via TLS bootstrapping

Operator benefits

- * One image to rule them all
- * Cluster config = sum of operator configs (GitOps)
 - * Operator status -> cluster health status
- * Upgrading a cluster is essentially updating every operator

Operated Operating System

RHEL CoreOS is RHEL8, designed to run containers only. Community counterpart - Fedora CoreOS

RHEL CoreOS release cycle is bound to OpenShift, not RHEL

RHEL Core OS = ContainerLinux ideas + RHEL packages

RHEL CoreOS specifics

Ignition to declaratively configure the system

ostree to make use of read-only root and atomic transactions

MachineConfigDaemon to apply updates and custom configs to existing instances

Give it a try

<https://try.openshift.com>

Openshift Online

<https://manage.openshift.com/>

Code Ready Containers - local OpenShift4

<https://code-ready.github.io/crc/>

