

# USING CONTAINERS FOR CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY

Carlos Sanchez

[csanchez.org](http://csanchez.org) / [@csanchez](https://twitter.com/csanchez)



Watch online at [carlossg.github.io/presentations](http://carlossg.github.io/presentations)

# ABOUT ME

Engineer, Scaling Jenkins



Author of Jenkins Kubernetes plugin

Contributor to Jenkins, Maven official Docker images

OSS contributor at Apache Maven, Eclipse, Puppet,...



# OUR USE CASE



Scaling Jenkins

Your mileage may vary



**Kernel Sanders**

@lstoll

The solution: Docker. The problem? You tell me.

Isolated Jenkins masters

Isolated agents and jobs

Memory and CPU limits

# BUT IT IS NOT TRIVIAL



*How would you design your  
infrastructure if you couldn't login?  
Ever.*

*Kelsey Hightower*

# CLUSTER SCHEDULING

- Running in public cloud, private cloud, VMs or bare metal
- HA and fault tolerant
- With Docker support of course



# EMBRACE FAILURE!



# SCALING JENKINS

Two options:

- More agents per master
- More masters

# SCALING JENKINS: MORE AGENTS

- Pros
  - Multiple plugins to add more agents, even dynamically
- Cons
  - The master is still a SPOF
  - Handling multiple configurations, plugin versions,...
  - There is a limit on how many agents can be attached

# SCALING JENKINS: MORE MASTERS

- Pros
  - Different sub-organizations can self service and operate independently
- Cons
  - Single Sign-On
  - Centralized configuration and operation

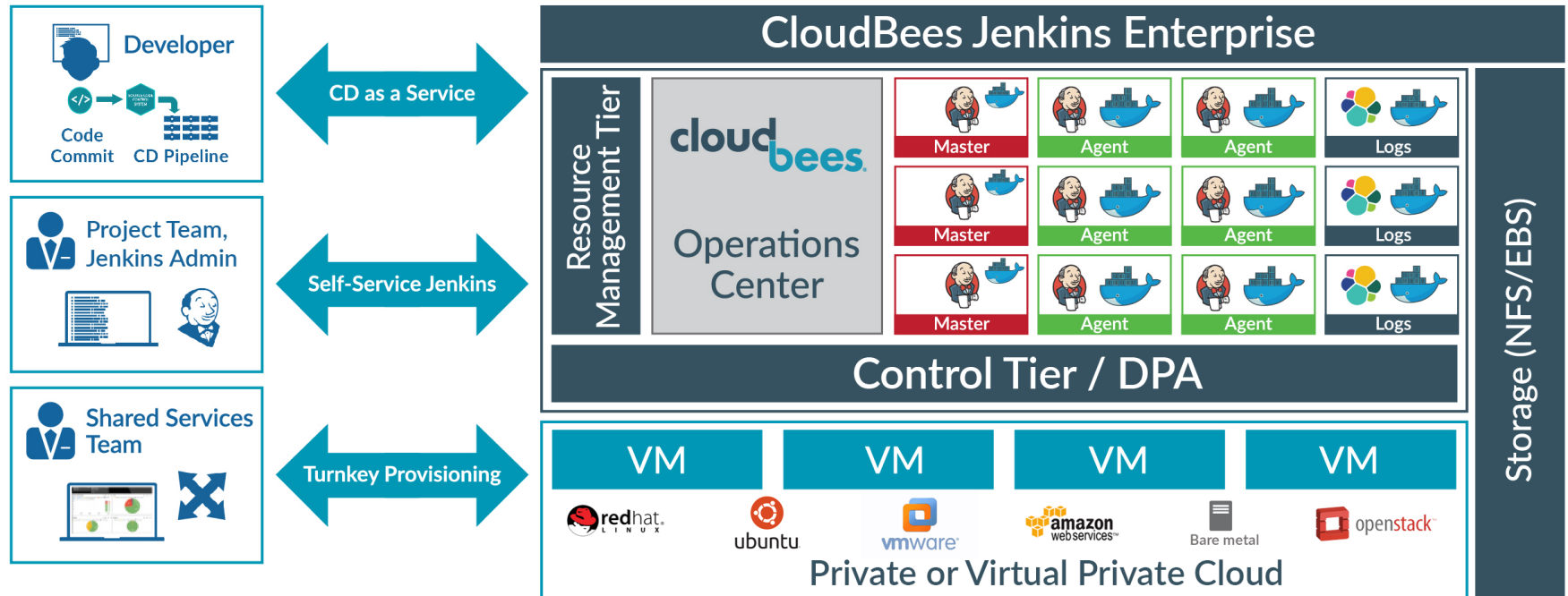
# CLOUDBEES JENKINS ENTERPRISE

The best of both worlds

CloudBees Jenkins Operations Center with multiple  
masters

Dynamic agent creation in each master

# CLOUDBEES JENKINS ENTERPRISE



# FIRST IMPLEMENTATION



- Apache Mesos
- Mesosphere Marathon
- Terraform
- Packer



&





We can run both Jenkins **masters** and **agents** in  
Kubernetes

# STORAGE

Handling distributed storage

Masters can move when they are restarted

Jenkins masters need persistent storage, agents  
(*typically*) don't

Using `PersistentVolumeClaim` so you can  
provide any implementation

# CAVEATS

- Performance

*"Worst-case scenario for just about any commonly used network-based storage"*

- Lack of multi-AZ for block storage, ie. EBS AWS

# NETWORKING

Jenkins masters open several ports

- HTTP
- JNLP agent

# NETWORKING: HTTP

We use ingress rules and `nginx` ingress controller

- Operations Center
- Jenkins masters

Path based routing `cje.example.com/master1`

# NETWORKING: JNLP

- Agents started dynamically in cluster can connect to masters internally
- Agents manually started outside cluster connect directly
  - Using NodePort

# AGENTS WITH INFINITE\* SCALE!

## Jenkins Kubernetes Plugin

- Dynamic Jenkins agents, running as Pods
- Multi-container support
  - One Jenkins agent image, others custom
- Jenkins Pipeline support for both agent Pod definition and execution
- Persistent workspace
- Auto configured

# ON DEMAND JENKINS AGENTS

```
podTemplate(label: 'mypod') {  
  node('mypod') {  
    sh 'Hello world!'  
  }  
}
```



# GROUPING CONTAINERS (PODS)

```
podTemplate(label: 'maven', containers: [  
  containerTemplate(name: 'maven', image: 'maven:3.3.9-jdk-8-at  
    ttyEnabled: true, command: 'cat') ]) {  
  
  node('maven') {  
    stage('Get a Maven project') {  
      git 'https://github.com/jenkinsci/kubernetes-plugin.git'  
      container('maven') {  
        stage('Build a Maven project') {  
          sh 'mvn -B clean package'  
        }  
      }  
    }  
  }  
}
```

# USING DECLARATIVE PIPELINE

```
pipeline {
  agent {
    kubernetes {
      label 'mypod'
      containerTemplate {
        name 'maven'
        image 'maven:3.3.9-jdk-8-alpine'
        ttyEnabled true
        command 'cat'
      }
    }
  }
  stages {
    stage('Run maven') {
      steps {
        container('maven') {
          sh 'mvn -version'
        }
      }
    }
  }
}
```

# MULTI-LANGUAGE PIPELINE

```
podTemplate(label: 'maven-golang', containers: [
  containerTemplate(name: 'maven', image: 'maven:3.3.9-jdk-8-alpine',
    ttyEnabled: true, command: 'cat'),
  containerTemplate(name: 'golang', image: 'golang:1.8.0',
    ttyEnabled: true, command: 'cat')]) {

  node('maven-golang') {
    stage('Build a Maven project') {
      git 'https://github.com/jenkinsci/kubernetes-plugin.git'
      container('maven') {
        sh 'mvn -B clean package' }}

    stage('Build a Golang project') {
      git url: 'https://github.com/hashicorp/terraform.git'
      container('golang') {
        sh """
        mkdir -p /go/src/github.com/hashicorp
        ln -s `pwd` /go/src/github.com/hashicorp/terraform
        cd /go/src/github.com/hashicorp/terraform && make core
        """
      }
    }
  }
}
```

# PODS: SELENIUM

Example:

- Jenkins agent
- Maven build
- Selenium Hub with
  - Firefox
  - Chrome

5 containers

```
podTemplate(label: 'maven-selenium', containers: [  
  
  containerTemplate(name: 'maven-firefox',  
    image: 'maven:3.3.9-jdk-8-alpine',  
    ttyEnabled: true, command: 'cat'),  
  
  containerTemplate(name: 'maven-chrome',  
    image: 'maven:3.3.9-jdk-8-alpine',  
    ttyEnabled: true, command: 'cat'),  
  
  containerTemplate(name: 'selenium-hub',  
    image: 'selenium/hub:3.4.0'),  
])
```

```
// because containers run in the same network space, we need
// make sure there are no port conflicts
// we also need to adapt the selenium images because they we
// designed to work with the --link option
```

```
containerTemplate(name: 'selenium-chrome',
  image: 'selenium/node-chrome:3.4.0', envVars: [
    containerEnvVar(key: 'HUB_PORT_4444_TCP_ADDR', value: 'loc
    containerEnvVar(key: 'HUB_PORT_4444_TCP_PORT', value: '444
    containerEnvVar(key: 'DISPLAY', value: ':99.0'),
    containerEnvVar(key: 'SE_OPTS', value: '-port 5556'),
  ]),
```

```
containerTemplate(name: 'selenium-firefox',  
  image: 'selenium/node-firefox:3.4.0', envVars: [  
    containerEnvVar(key: 'HUB_PORT_4444_TCP_ADDR', value: 'loc  
    containerEnvVar(key: 'HUB_PORT_4444_TCP_PORT', value: '444  
    containerEnvVar(key: 'DISPLAY', value: ':98.0'),  
    containerEnvVar(key: 'SE_OPTS', value: '-port 5557'),  
  ])
```

```
node( 'maven-selenium' ) {  
  stage( 'Checkout' ) {  
    git 'https://github.com/carlossg/selenium-example.git'  
    parallel (
```



```
firefox: {  
  container('maven-firefox') {  
    stage('Test firefox') {  
      sh """  
        mvn -B clean test -Dselenium.browser=firefox \  
          -Dsurefire.rerunFailingTestsCount=5 -Dsleep=0  
        """  
    }  
  }  
},
```

```
chrome: {  
  container('maven-chrome') {  
    stage('Test chrome') {  
      sh """  
        mvn -B clean test -Dselenium.browser=chrome \  
          -Dsurefire.rerunFailingTestsCount=5 -Dsleep=0  
        """  
    }  
  }  
}
```

# STORAGE

## Persistent volumes

- GCE disks
- GlusterFS
- NFS
- EBS
- etc

# USING PERSISTENT VOLUMES

```
apiVersion: "v1"
kind: "PersistentVolumeClaim"
metadata:
  name: "maven-repo"
  namespace: "kubernetes-plugin"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

```
podTemplate(label: 'maven', containers: [  
  containerTemplate(name: 'maven', image: 'maven:3.3.9-jdk-8-a  
    ttyEnabled: true, command: 'cat')  
], volumes: [  
  persistentVolumeClaim(mountPath: '/root/.m2/repository',  
    claimName: 'maven-repo', readOnly: false)  
]) {  
  
  node('maven') {  
    stage('Build a Maven project') {  
      git 'https://github.com/jenkinsci/kubernetes-plugin.git'  
      container('maven') {  
        sh 'mvn -B clean package'  
      }  
    }  
  }  
}
```

# RESOURCE REQUESTS AND LIMITS

```
podTemplate(label: 'mypod', containers: [  
  containerTemplate(  
    name: 'maven', image: 'maven', ttyEnabled: true,  
    resourceRequestCpu: '50m',  
    resourceLimitCpu: '100m',  
    resourceRequestMemory: '100Mi',  
    resourceLimitMemory: '200Mi'))]) {  
  ...  
}
```

# DEPLOYING TO KUBERNETES



**@DEVOPS\_BORAT**

DevOps Borat

To make error is human. To propagate error to all server in automatic way is **#devops**.



*If you haven't automatically destroyed  
something by mistake, you are not  
automating enough*

```
podTemplate(label: 'deployer', serviceAccount: 'deployer',  
  containers: [  
    containerTemplate(name: 'kubectl',  
      image: 'lachlanevenson/k8s-kubectl:v1.7.8',  
      command: 'cat',  
      ttyEnabled: true)  
  ]){  
  node('deployer') {  
    container('kubectl') {  
      sh "kubectl apply -f my-kubernetes.yaml"  
    }  
  }  
}
```

# kubernetes-pipeline-plugin

```
podTemplate(label: 'deploy', serviceAccount: 'deployer') {  
  
    stage('deployment') {  
        node('deploy') {  
            checkout scm  
            kubernetesApply(environment: 'hello-world',  
                file: readFile('kubernetes-hello-world-service.yaml'))  
            kubernetesApply(environment: 'hello-world',  
                file: readFile('kubernetes-hello-world-v1.yaml')) }}  
  
    stage('upgrade') {  
        timeout(time:1, unit:'DAYS') {  
            input id: 'approve', message:'Approve upgrade?'  
        }  
        node('deploy') {  
            checkout scm  
            kubernetesApply(environment: 'hello-world',  
                file: readFile('kubernetes-hello-world-v2.yaml'))  
        }}  
    }
```

Or Azure [kubernetes-cd-plugin](#)

```
kubernetesDeploy(  
  credentialType: 'KubeConfig',  
  kubeConfig: [path: '$HOME/.kube/config'],  
  
  configs: '*.yaml',  
  enableConfigSubstitution: false,  
)
```

# THE FUTURE

- Periodic snapshotting, ie. EBS volumes in AWS
- Affinity
- Dedicated workers for agents, infra autoscaling
- Multi-region
- Namespaces per team: quotas, isolation

# THANKS

[csanchez.org](http://csanchez.org)



cloudbees®