SCALING JENKINS WITH DOCKER

SWARM, KUBERNETES OR MESOS?

Carlos Sanchez / csanchez.org / @csanchez

ABOUT ME

Engineer @ CloudBees, scaling Jenkins

Author/contributor of Jenkins Kubernetes and Mesos plugins

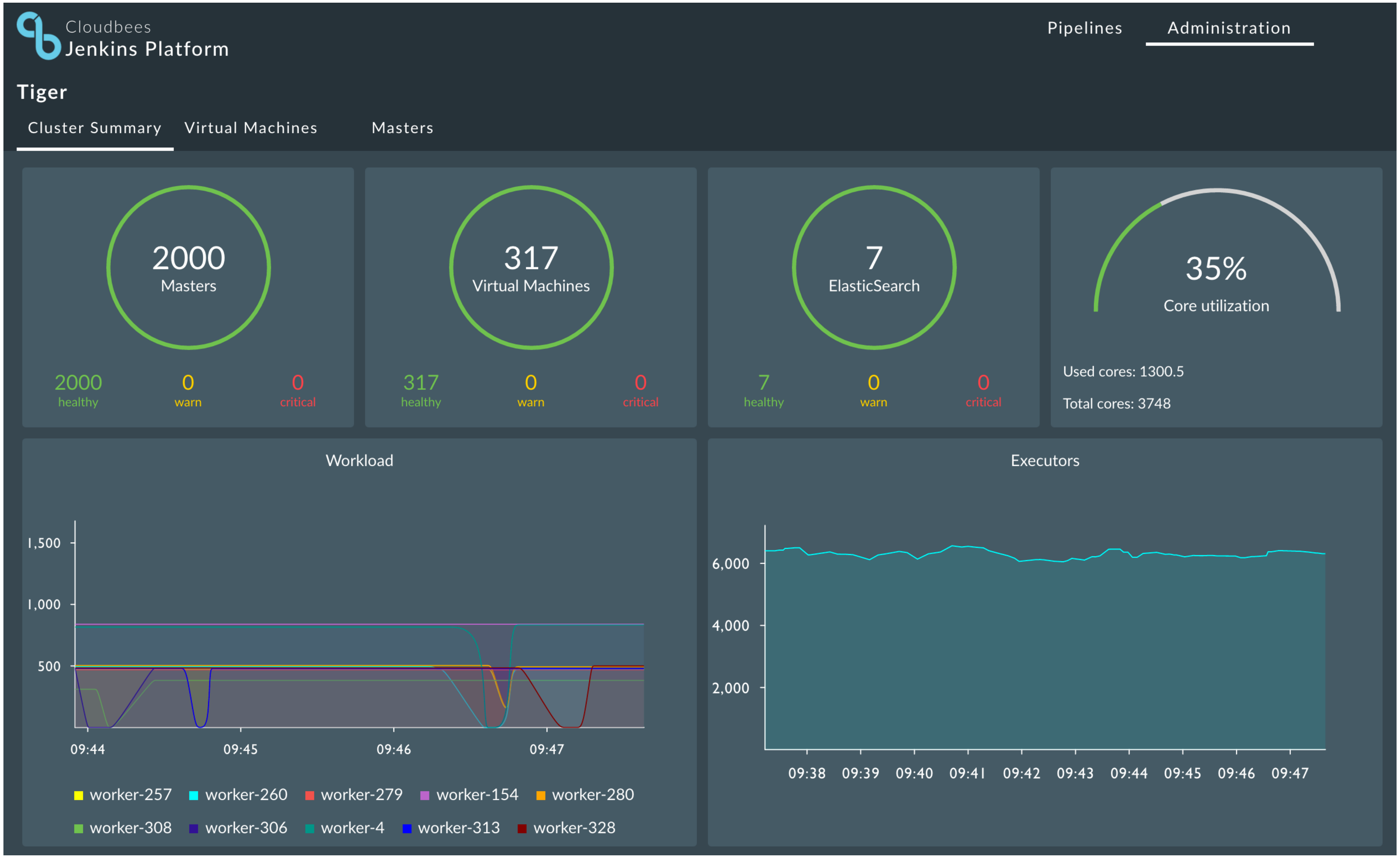
Long time OSS contributor at Apache (Apache Maven), Eclipse, Puppet,…

Google Developers Experts

OUR USE CASE

Scaling Jenkins

Your mileage may vary



Cloudbees
Jenkins Platform

Pipelines   Administration

Tiger

Cluster Summary   Virtual Machines   Masters

2000 Masters — 2000 healthy, 0 warn, 0 critical

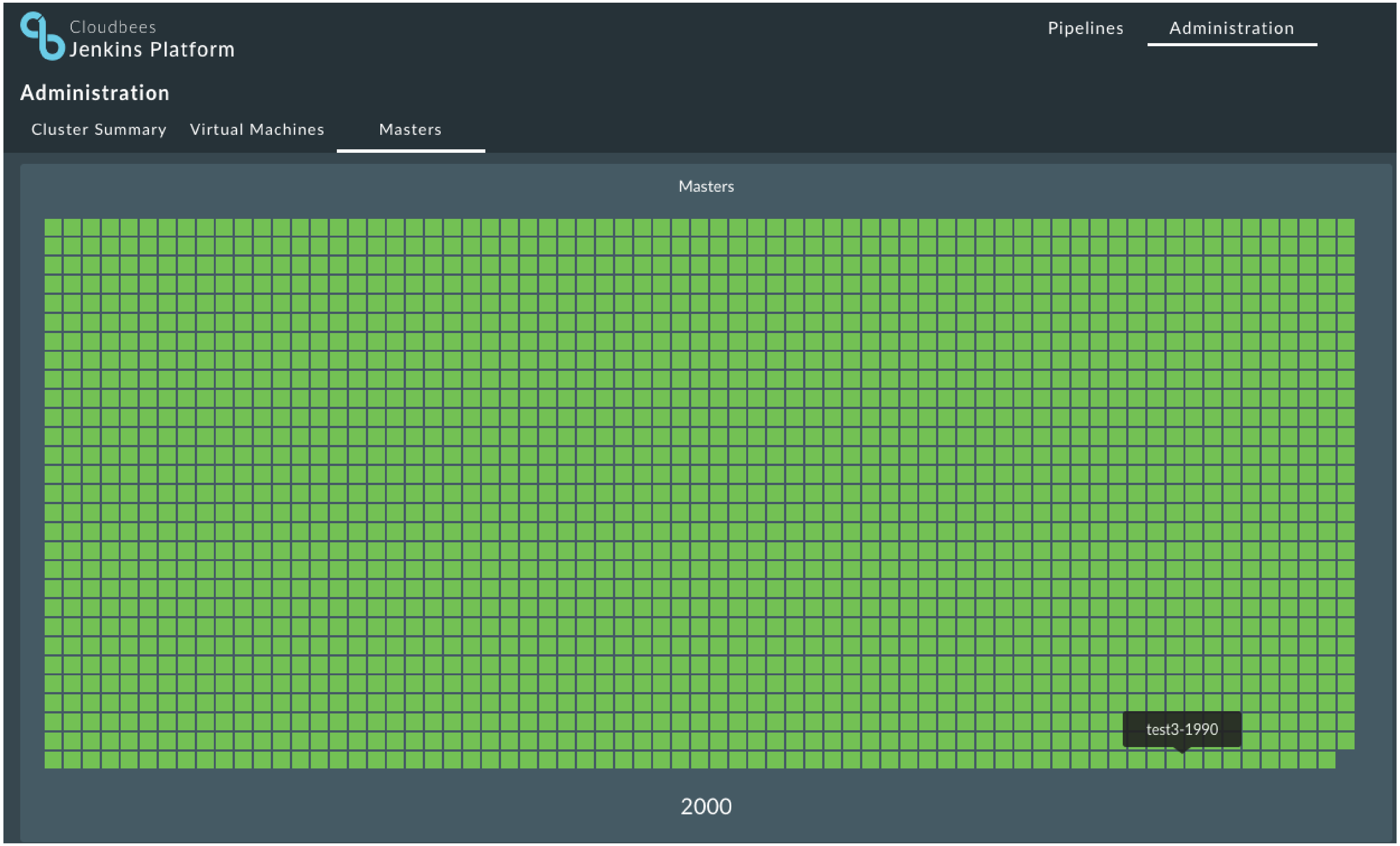
317 Virtual Machines — 317 healthy, 0 warn, 0 critical

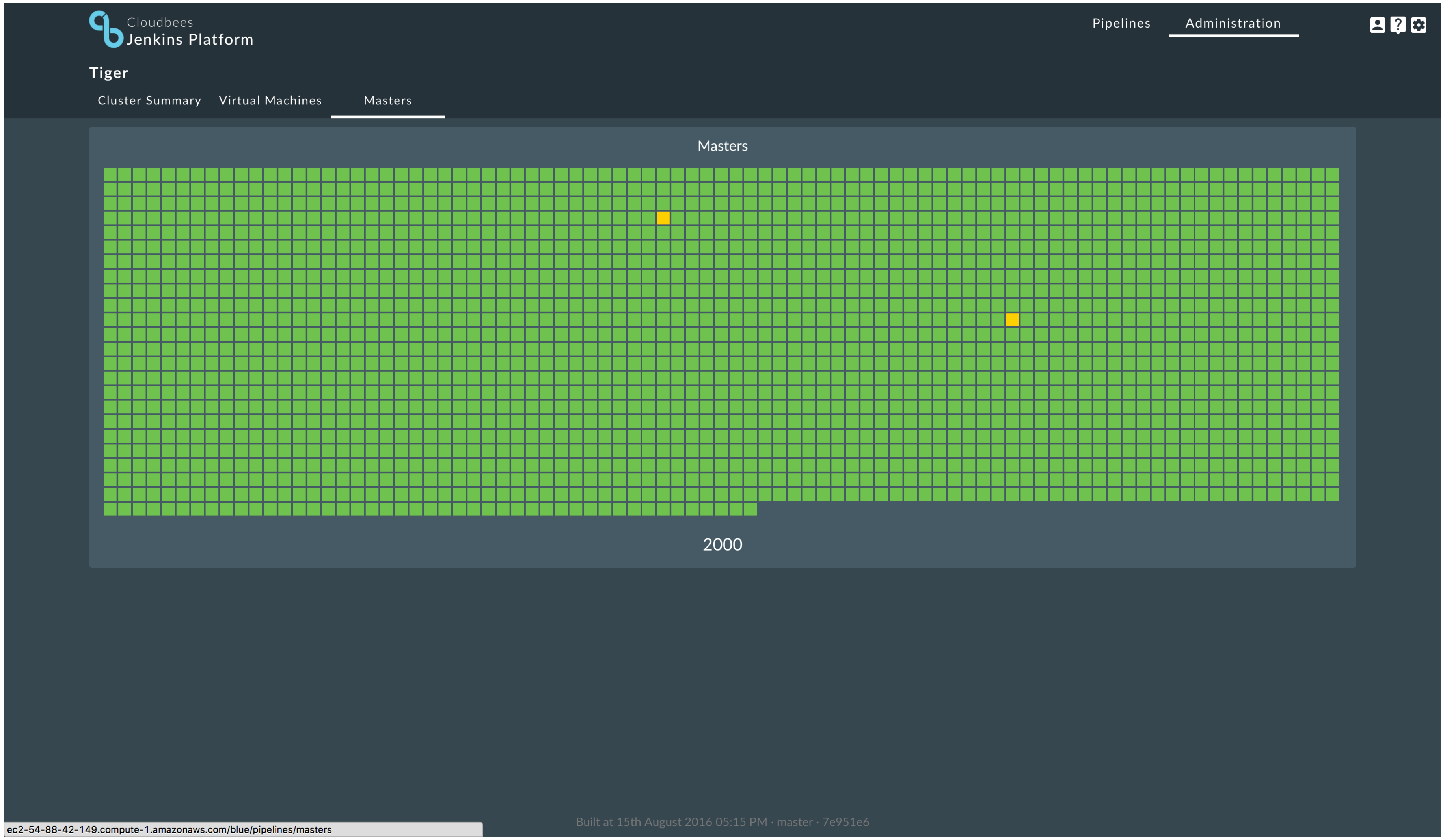
7 ElasticSearch — 7 healthy, 0 warn, 0 critical

35% Core utilization

Used cores: 1300.5

Total cores: 3748


Cloudbees
Jenkins Platform
Pipelines
Administration
Administration
Cluster Summary
Virtual Machines
Masters
Masters
test3-1990
2000


Cloudbees
Jenkins Platform
Pipelines
Administration
Tiger
Cluster Summary
Virtual Machines
Masters
Masters
2000
Built at 15th August 2016 05:15 PM · master · 7e951e6
ec2-54-88-42-149.compute-1.amazonaws.com/blue/pipelines/masters

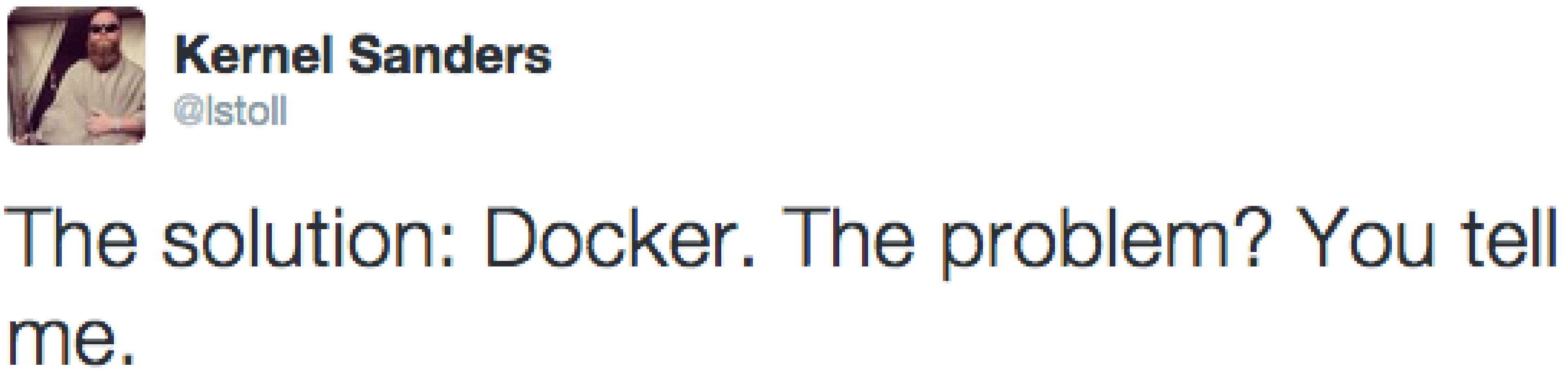
A 2000 JENKINS MASTERS CLUSTER

- 3 Mesos masters (m3.xlarge: 4 vCPU, 15GB, 2x40 SSD)
- 317 Mesos slaves (c3.2xlarge, m3.xlarge, m4.4xlarge)
- 7 Mesos slaves dedicated to ElasticSearch: (c3.8xlarge: 32 vCPU, 60GB)

12.5 TB - 3748 CPU

Running 2000 masters and ~8000 concurrent jobs

DOCKER DOCKER DOCKER



Kernel Sanders
@lstoll

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

BUT IT IS NOT TRIVIAL

CLUSTER SCHEDULING

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

MESOS

kubernetes

MESOS

A distributed systems kernel

hadoop Spark kafka

APACHE MESOS

- Started before 2011
- Runs tasks, any binary or Docker, `rkt`, `appc` images
- Frameworks run on top of Mesos
  - Mesosphere Marathon: long running services
  - Apache Aurora: long running services
  - Chronos: distributed cron-like system
- Used in Twitter, Airbnb, eBay, Apple, Verizon, Yelp,...

DOCKER SWARM

DOCKER SWARM

- By Docker Inc.
- Uses the same Docker API
- No need to modify existing tooling

DOCKER ENGINE SWARM MODE

- New Swarm mode in Docker 1.12
- No need to install extra software, each daemon can run as a Swarm member
- New `service` object to describe distributed containers
    - Existing tooling needs to be updated

kubernetes

KUBERNETES

- Based on Google Borg
- Run in local machine, virtual, cloud
- Google provides Google Container Engine (GKE)
- Other services run by stackpoint.io, CoreOS Tectonic, Azure,...
- Minikube for local testing

SCALING JENKINS

Two options:

- More build agents per master
- More masters

SCALING JENKINS: MORE BUILD 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 build 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

Covered by CloudBees Jenkins Operations Center and CloudBees Jenkins Platform Private SaaS Edition

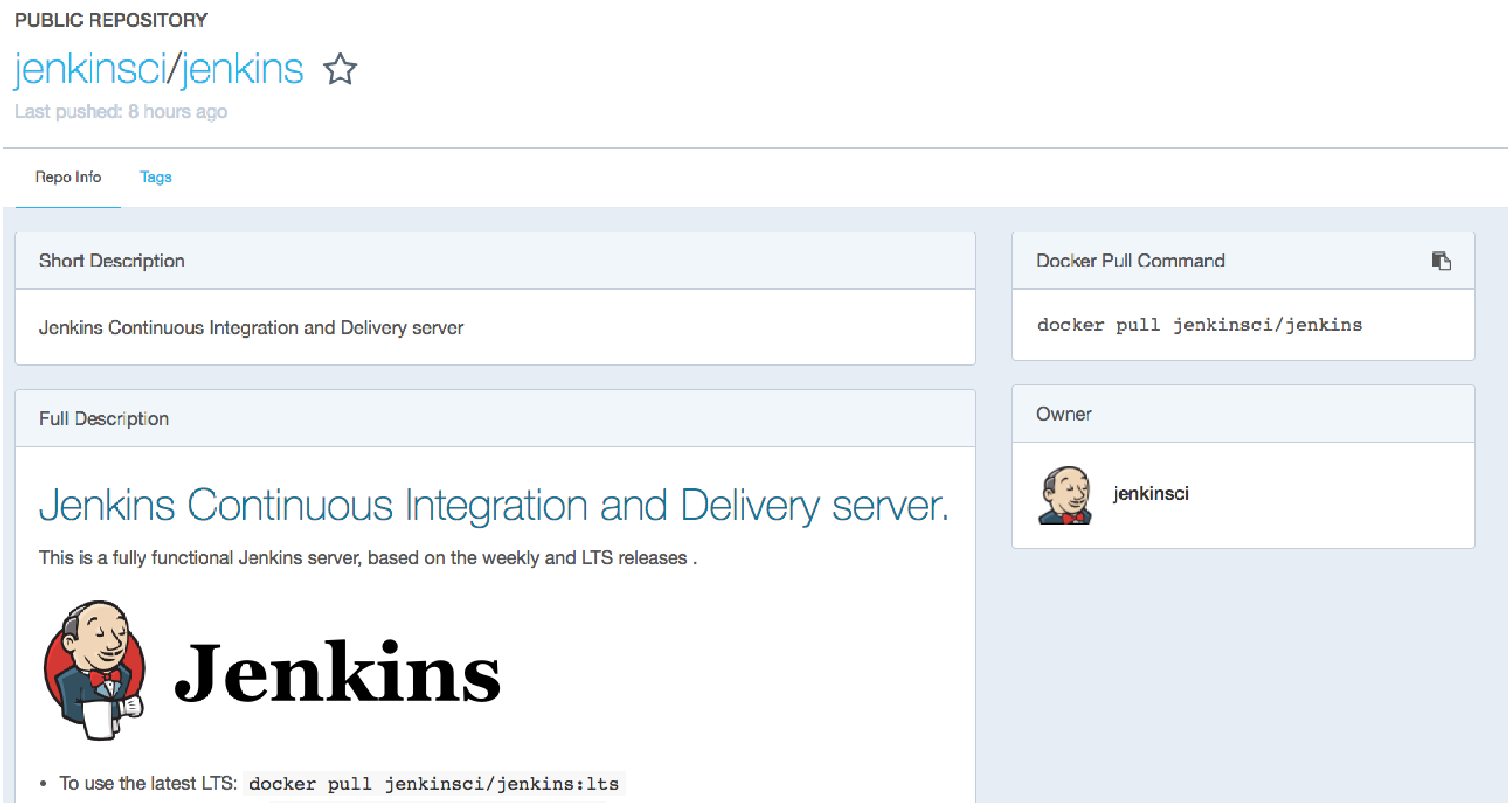
@DEVOPS_BORAT

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

RUNNING IN DOCKER



PUBLIC REPOSITORY

jenkinsci/jenkins ☆

Last pushed: 8 hours ago

Short Description

Jenkins Continuous Integration and Delivery server

Docker Pull Command

```
docker pull jenkinsci/jenkins
```

Owner

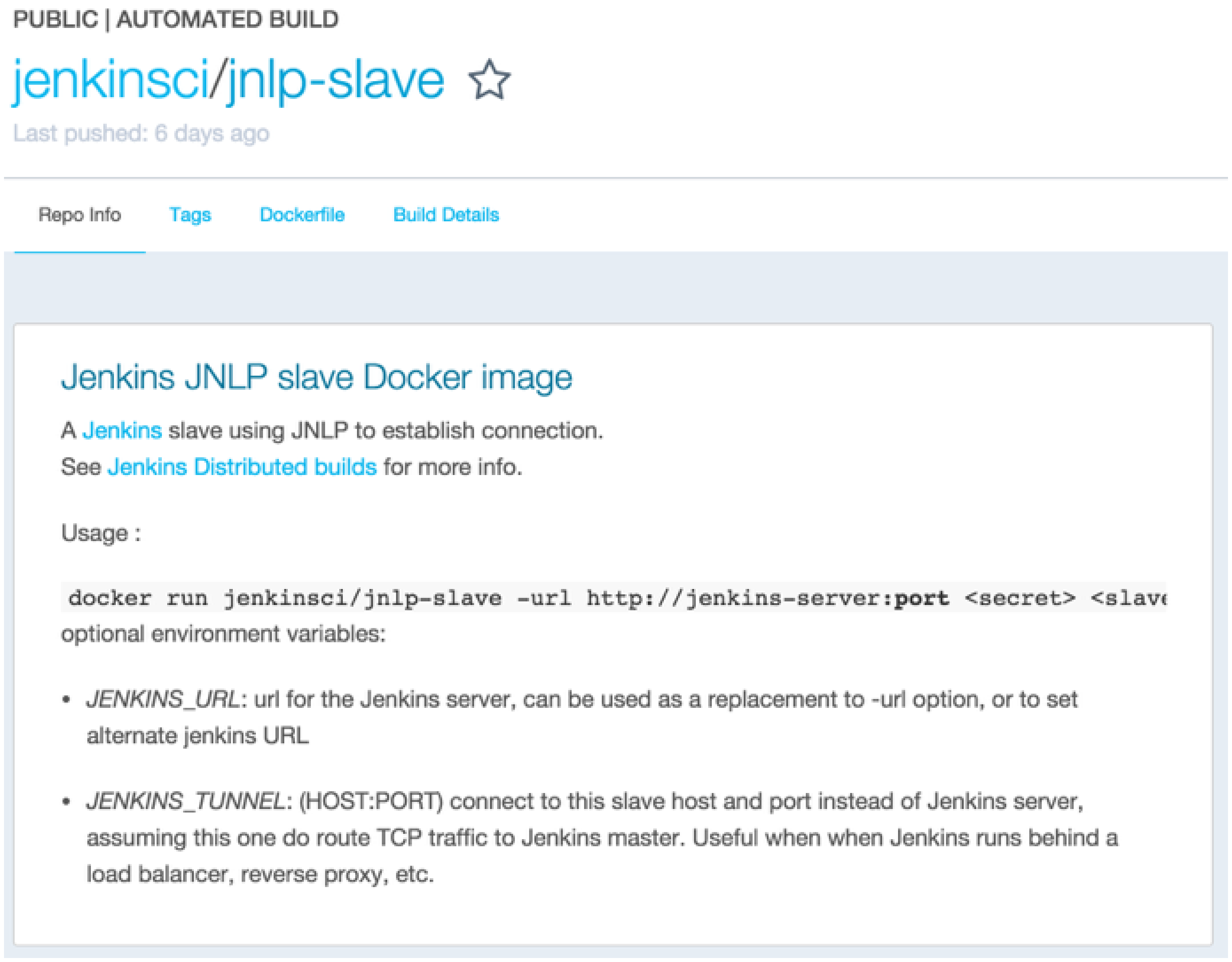
jenkinsci

Full Description

Jenkins Continuous Integration and Delivery server.

This is a fully functional Jenkins server, based on the weekly and LTS releases .

- To use the latest LTS: `docker pull jenkinsci/jenkins:lts`



PUBLIC | AUTOMATED BUILD

jenkinsci/jnlp-slave

Last pushed: 6 days ago

Jenkins JNLP slave Docker image

A Jenkins slave using JNLP to establish connection.
See Jenkins Distributed builds for more info.

Usage :

```
docker run jenkinsci/jnlp-slave -url http://jenkins-server:port <secret> <slave
```

optional environment variables:

- *JENKINS_URL*: url for the Jenkins server, can be used as a replacement to -url option, or to set alternate jenkins URL
- *JENKINS_TUNNEL*: (HOST:PORT) connect to this slave host and port instead of Jenkins server, assuming this one do route TCP traffic to Jenkins master. Useful when when Jenkins runs behind a load balancer, reverse proxy, etc.

CLUSTER SCHEDULING

Isolated build agents and jobs

Using Docker

Capabilities can be dropped

GROUPING CONTAINERS

Example:

- Jenkins agent
- Maven build
- Selenium testing in
  - Firefox
  - Chrome
  - Safari

5 containers

GROUPING CONTAINERS

Mesos	In progress MESOS-2449
Swarm	Supports grouping through Docker Compose Can force execution in the same host
Kubernetes	Supports the concept of Pods natively All running in the same host

MEMORY LIMITS

Scheduler needs to account for container memory requirements and host available memory

Prevent containers for using more memory than allowed

Mesos	required
Swarm	optional
Kubernetes	optional (plus namespaces)

Memory constrains translate to Docker --memory

WHAT DO YOU THINK HAPPENS WHEN?

Your container goes over memory quota?

¡CACHONDEO.COM

WHAT ABOUT THE JVM?

WHAT ABOUT THE CHILD PROCESSES?

CPU LIMITS

Scheduler needs to account for container CPU requirements and host available CPUs

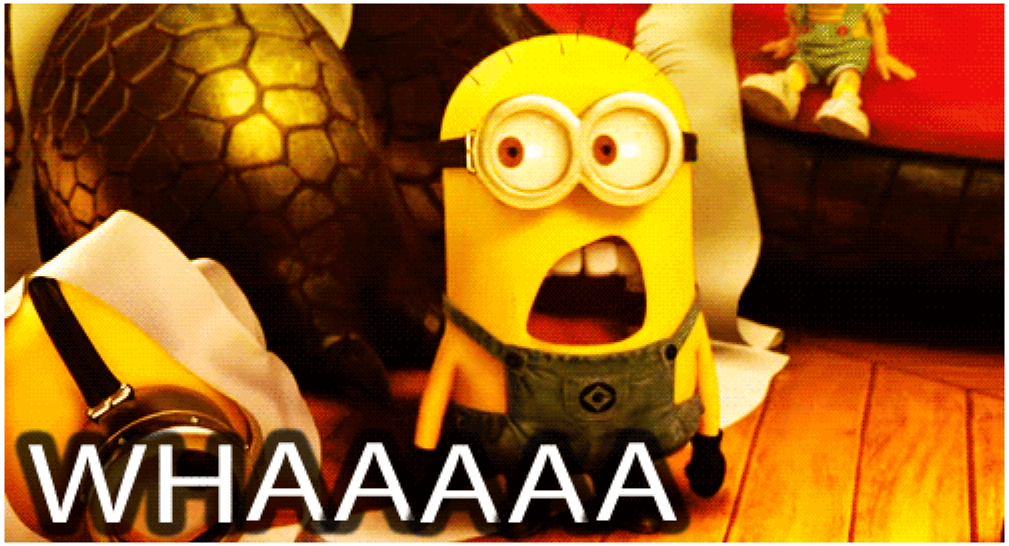
Mesos	required
Swarm	optional
Kubernetes	optional (plus namespaces)

CPU translates into Docker `--cpu-shares`

WHAT DO YOU THINK HAPPENS WHEN?

Your container tries to access more than one CPU

Your container goes over CPU limits



Totally different from memory

STORAGE

Handling distributed storage

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

Mesos	Docker volume support in 1.0+
Swarm	Docker volume plugins: RexRay, Convoy, Flocker,...
Kubernetes	Persistent volumes

PERMISSIONS

Containers should not run as root

Container user id != host user id

i.e. `jenkins` user in container is always 1000 but matches `ubuntu` user in host

CAVEATS

Only a limited number of EBS volumes can be mounted

Docs say `/dev/sd[f-p]`, but `/dev/sd[q-z]` seem to work too

NFS users must be centralized and match in cluster and NFS server

NETWORKING

Jenkins masters open several ports

- HTTP
- JNLP Build agent
- SSH server (Jenkins CLI type operations)

Jenkins agents connect to master:

- inbound (SSH)
- outbound (JNLP)

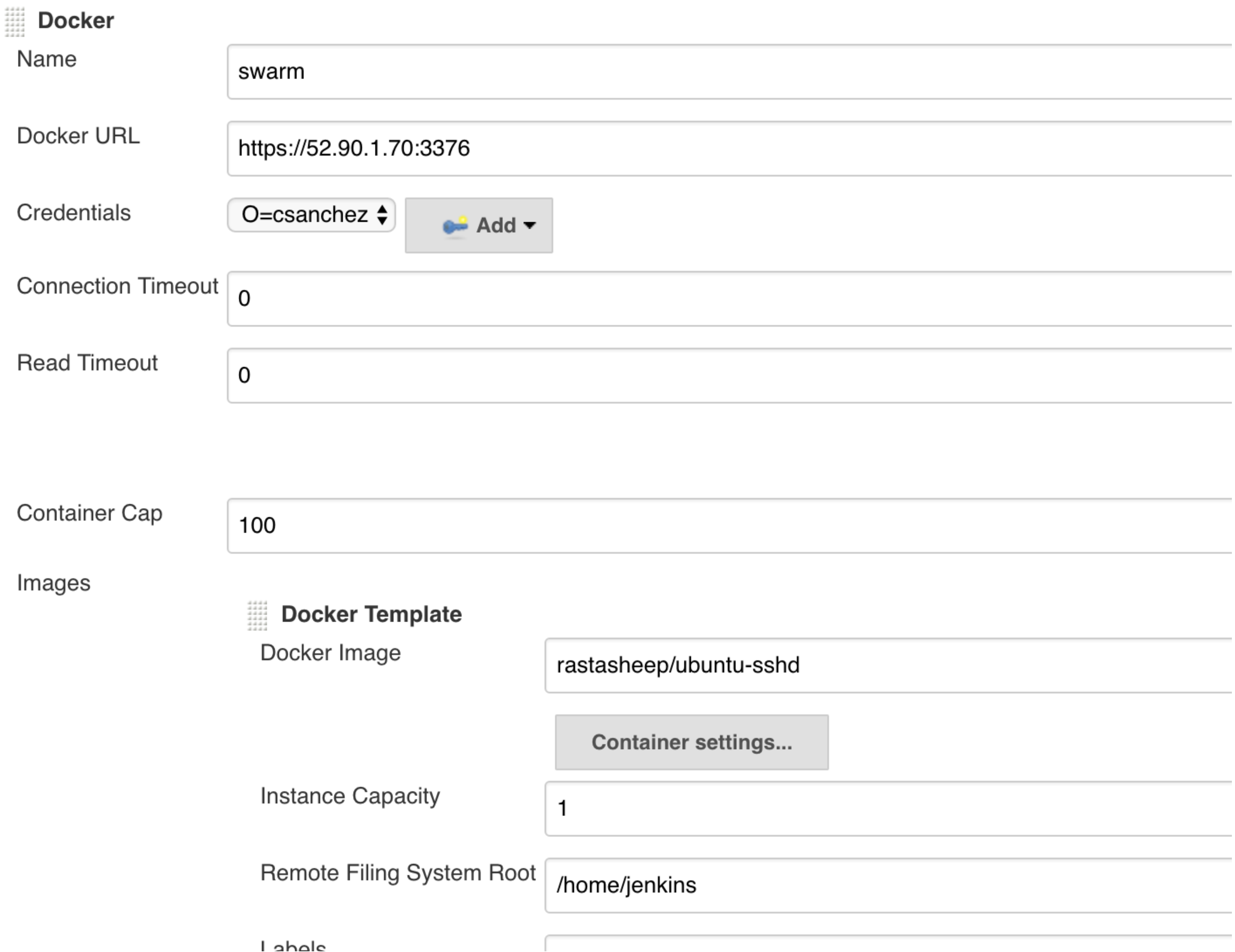
Allows getting one IP per container

Mesos	Network Isolator Modules: Calico, Weave
Swarm	Docker overlay, and others from plugins
Kubernetes	Multiple networking options: GCE, Weave, Calico,...

JENKINS PLUGINS

JENKINS DOCKER PLUGINS

- Dynamic Jenkins agents with Docker plugin or Yet Another Docker Plugin
  - No support yet for Docker 1.12 Swarm mode
- Agent image needs to include Java, downloads slave jar from Jenkins master
- Multiple plugins for different tasks
  - Docker build and publish
  - Docker build step plugin
  - CloudBees Docker Hub/Registry Notification
  - CloudBees Docker Traceability
- Great pipeline support


Docker

Name: swarm

Docker URL: https://52.90.1.70:3376

Credentials: O=csanchez

Connection Timeout: 0

Read Timeout: 0

Container Cap: 100

Images

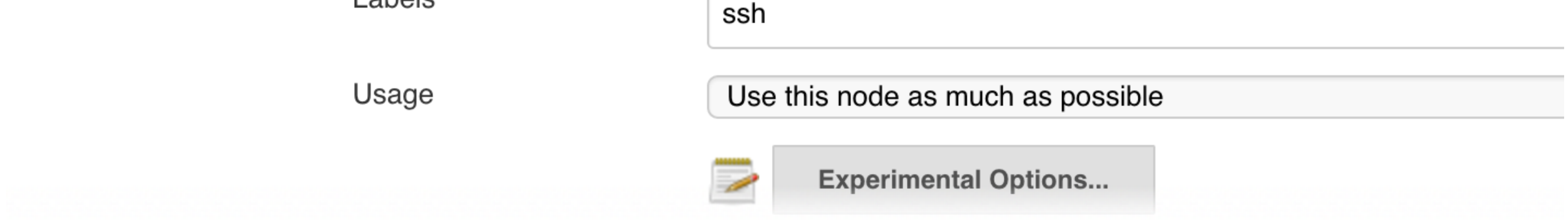
Docker Template

Docker Image: rastasheep/ubuntu-sshd

Instance Capacity: 1

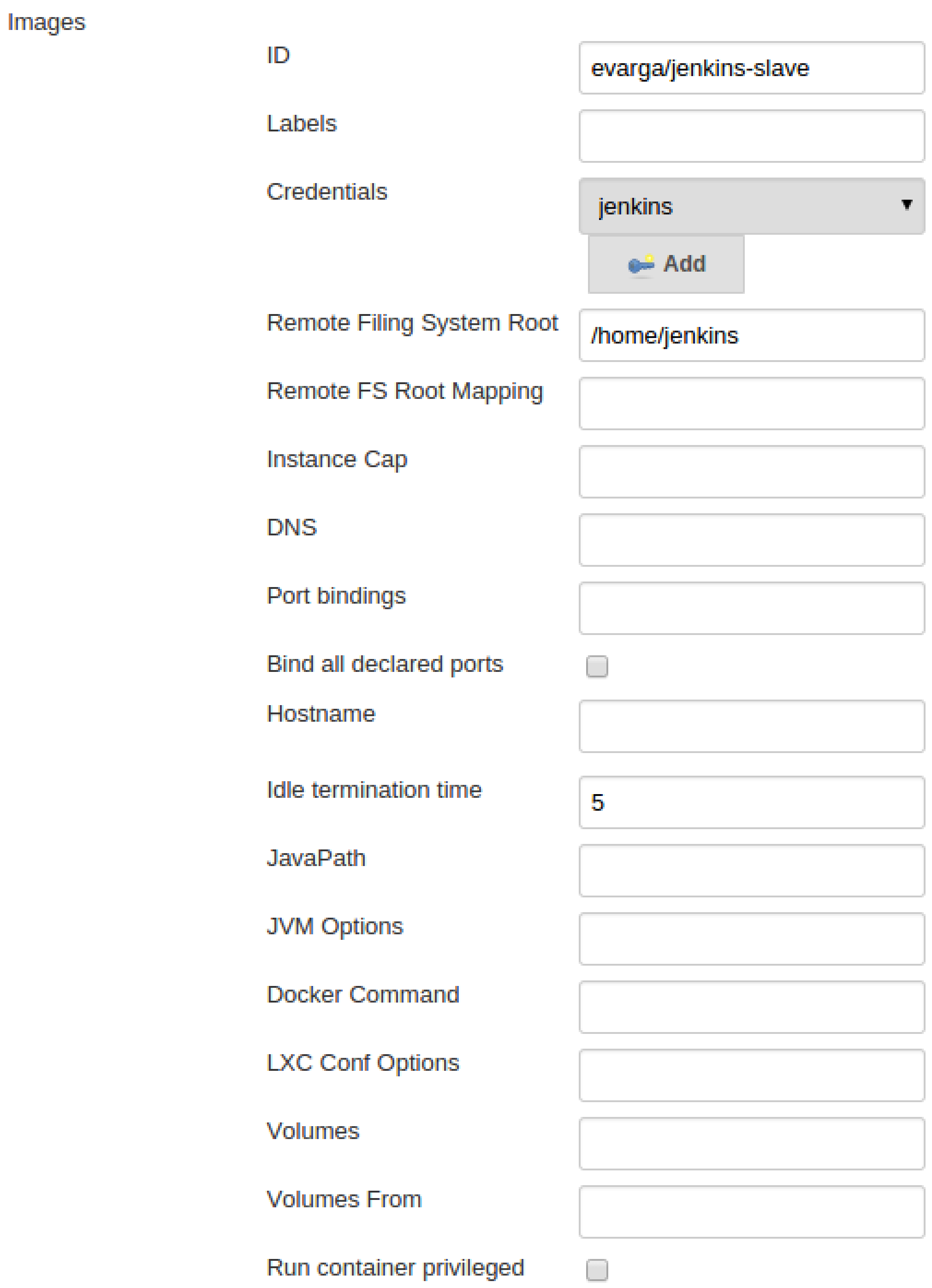
Remote Filing System Root: /home/jenkins

Labels

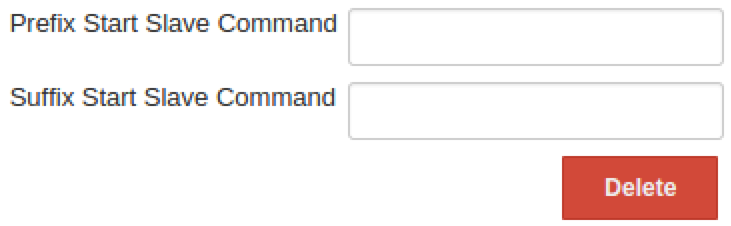


Labels ssh

Usage Use this node as much as possible



Images	
ID	evarga/jenkins-slave
Labels	
Credentials	jenkins
Remote Filing System Root	/home/jenkins
Remote FS Root Mapping	
Instance Cap	
DNS	
Port bindings	
Bind all declared ports	☐
Hostname	
Idle termination time	5
JavaPath	
JVM Options	
Docker Command	
LXC Conf Options	
Volumes	
Volumes From	
Run container privileged	☐


Prefix Start Slave Command

Suffix Start Slave Command

JENKINS DOCKER PIPELINE

```
def maven = docker.image('maven:3.3.9-jdk-8');

stage 'Mirror'
maven.pull()
docker.withRegistry('https://secure-registry/', 'docker-registry-logi

  stage 'Build'
  maven.inside {
    sh "mvn -B clean package"
  }

  stage 'Bake Docker image'
  def pcImg = docker.build("examplecorp/spring-petclinic:${env.BUILD_

  pcImg.push();
}
```

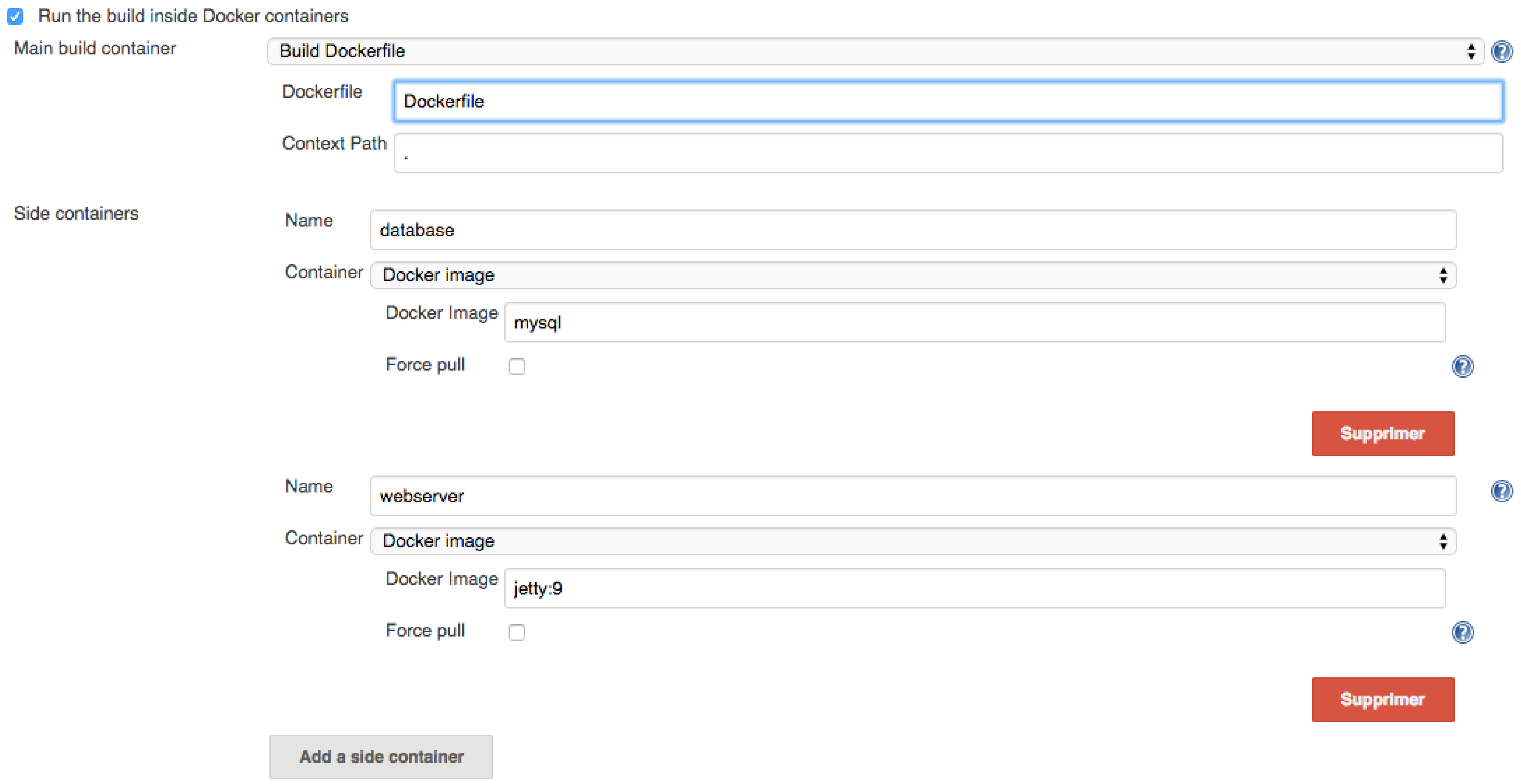
JENKINS DOCKER SLAVES PLUGIN

Use any Docker image, no need for Java

Definition in pipeline

Can have side containers

Just released!



☑ Run the build inside Docker containers

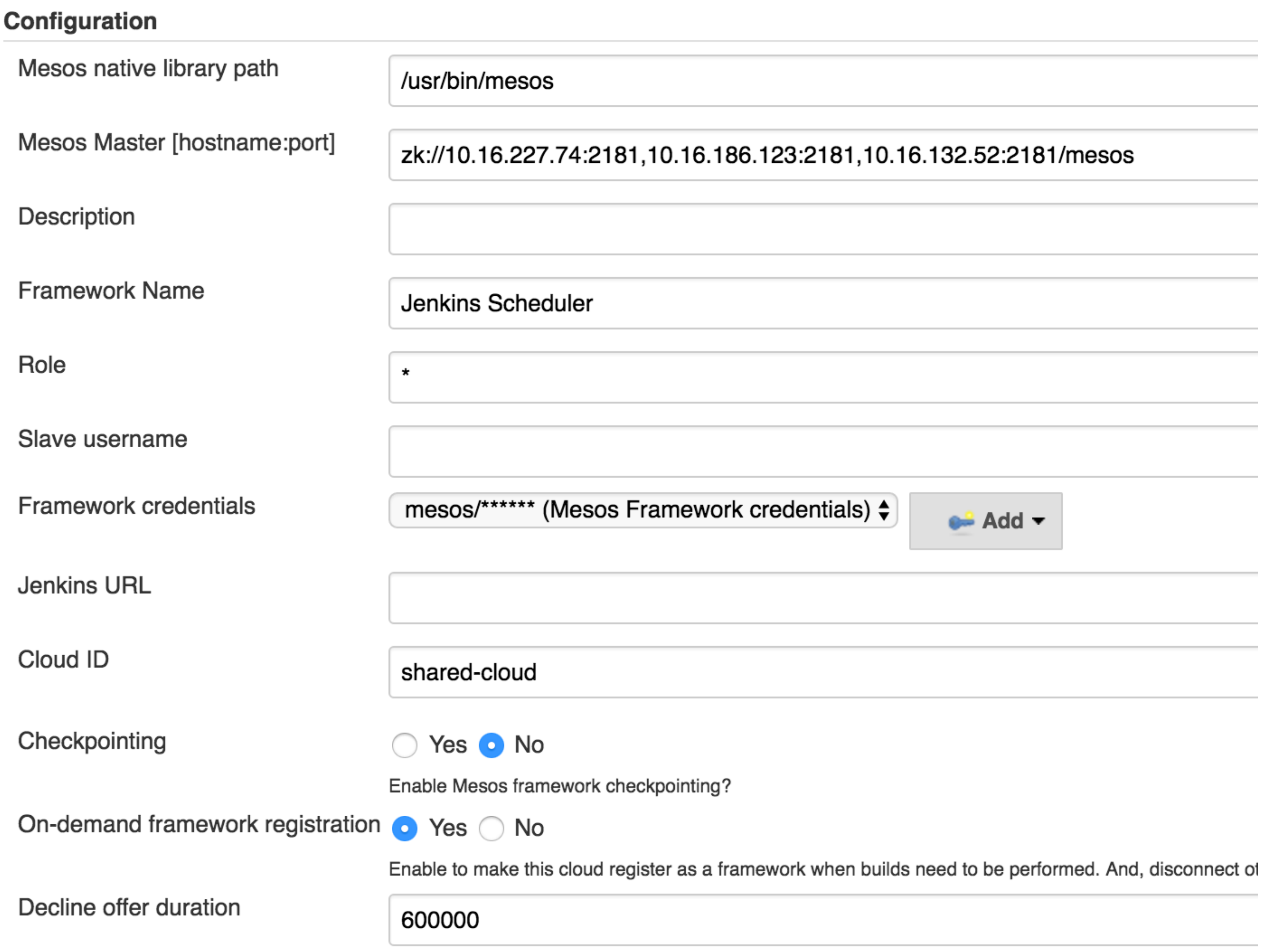
Main build container	Build Dockerfile
Dockerfile	Dockerfile
Context Path	.
Side containers	
Name	database
Container	Docker image
Docker Image	mysql
Force pull	☐
Name	webserver
Container	Docker image
Docker Image	jetty:9
Force pull	☐

Building Maven

```
dockerNode("maven:3.3.3-jdk-8") {
  sh "mvn -version"
}
```

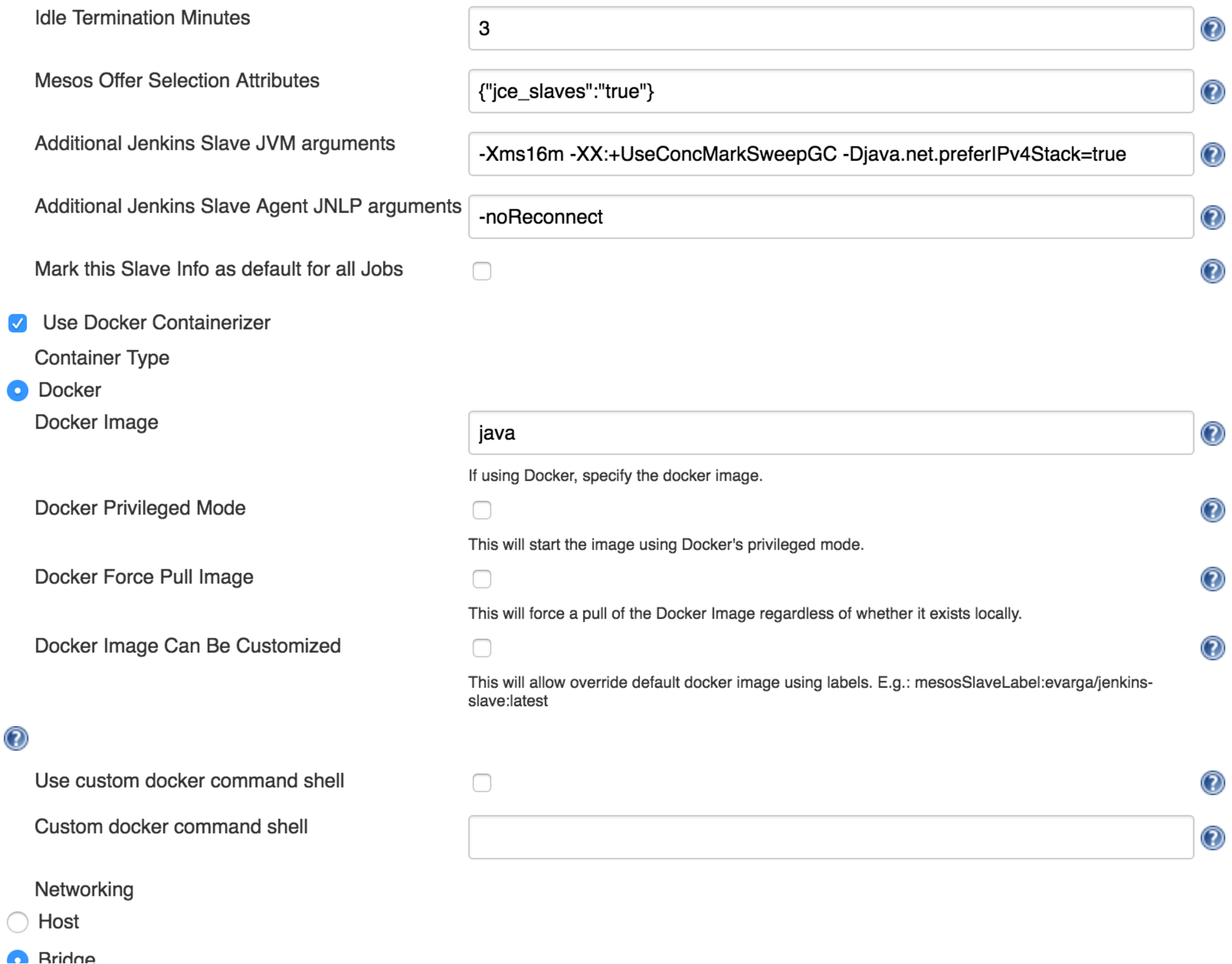
JENKINS MESOS PLUGIN

- Dynamic Jenkins agents, both Docker and isolated processes
- Agent image needs to include Java, grabs slave jar from Mesos sandbox
- Can run Docker commands on the host, outside of Mesos



Configuration

Mesos native library path	/usr/bin/mesos
Mesos Master [hostname:port]	zk://10.16.227.74:2181,10.16.186.123:2181,10.16.132.52:2181/mesos
Description	
Framework Name	Jenkins Scheduler
Role	*
Slave username	
Framework credentials	mesos/****** (Mesos Framework credentials)
Jenkins URL	
Cloud ID	shared-cloud
Checkpointing	Yes [✓ No]
	Enable Mesos framework checkpointing?
On-demand framework registration	[✓ Yes] No
	Enable to make this cloud register as a framework when builds need to be performed. And, disconnect of
Decline offer duration	600000



Idle Termination Minutes	3
Mesos Offer Selection Attributes	{"jce_slaves":"true"}
Additional Jenkins Slave JVM arguments	-Xms16m -XX:+UseConcMarkSweepGC -Djava.net.preferIPv4Stack=true
Additional Jenkins Slave Agent JNLP arguments	-noReconnect
Mark this Slave Info as default for all Jobs	☐

✓ Use Docker Containerizer

Container Type

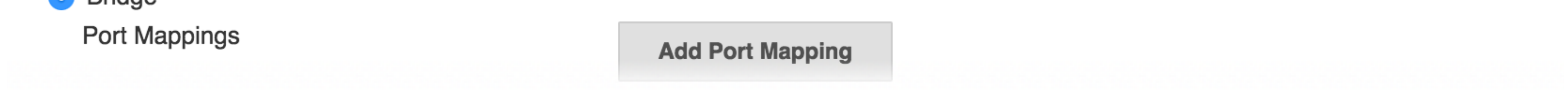
● Docker

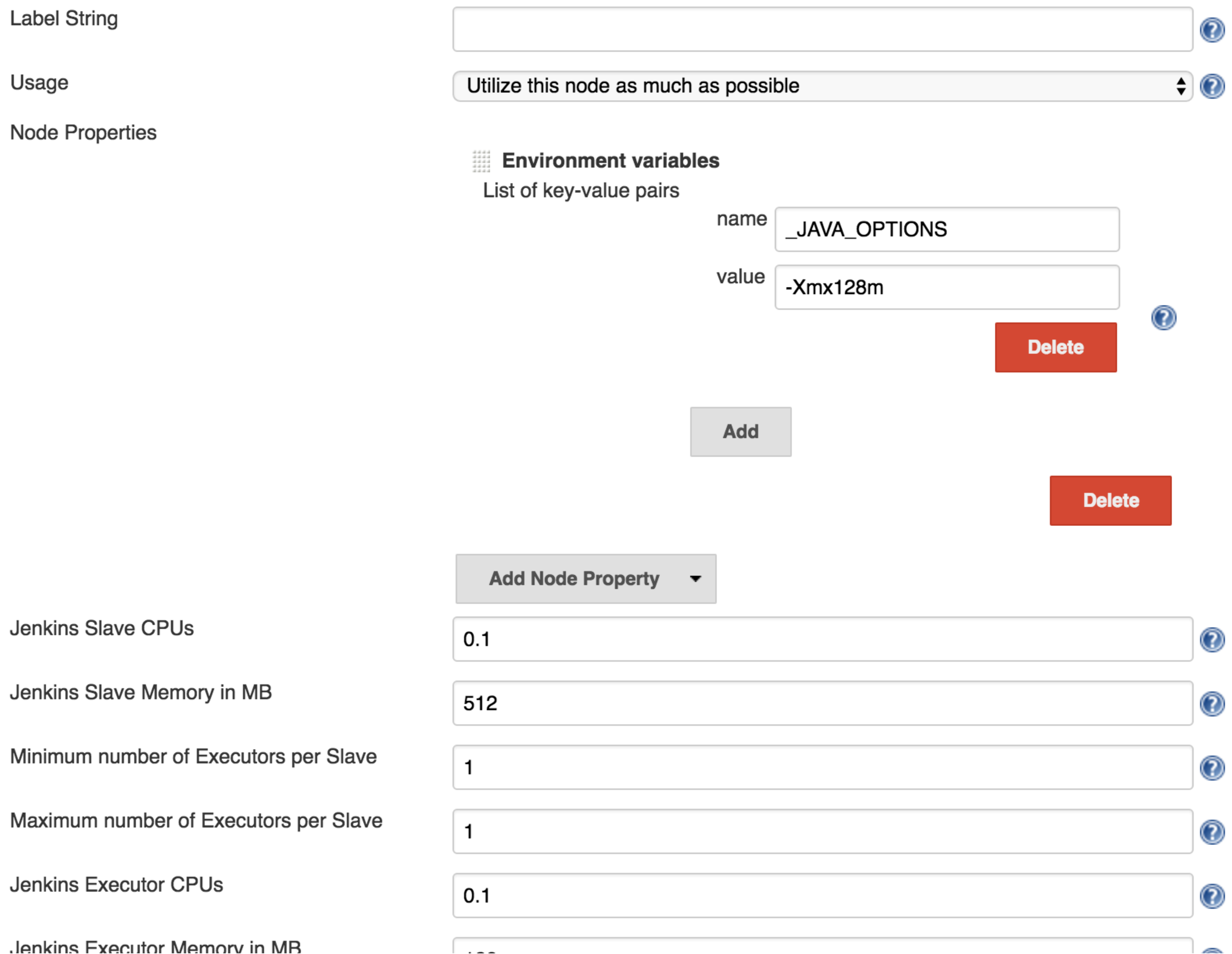
Docker Image	java If using Docker, specify the docker image.
Docker Privileged Mode	☐ This will start the image using Docker's privileged mode.
Docker Force Pull Image	☐ This will force a pull of the Docker Image regardless of whether it exists locally.
Docker Image Can Be Customized	☐ This will allow override default docker image using labels. E.g.: mesosSlaveLabel:evarga/jenkins-slave:latest
Use custom docker command shell	☐
Custom docker command shell	

Networking

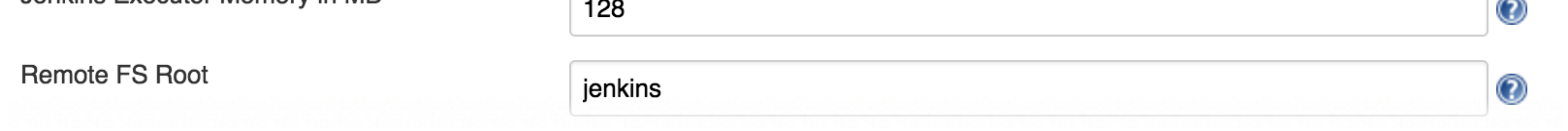
○ Host

● Bridge


Port Mappings
Add Port Mapping



Label String	
Usage	Utilize this node as much as possible
Node Properties	**Environment variables** List of key-value pairs
	name: _JAVA_OPTIONS
	value: -Xmx128m
Jenkins Slave CPUs	0.1
Jenkins Slave Memory in MB	512
Minimum number of Executors per Slave	1
Maximum number of Executors per Slave	1
Jenkins Executor CPUs	0.1
Jenkins Executor Memory in MB	



Jenkins Executor Memory in MB	128
Remote FS Root	jenkins

JENKINS MESOS PLUGIN

Can use Docker pipelines with some tricks

- Need Docker client installed
- Shared docker.sock from host
- Mount the workspace in the host, visible under same dir

MESOS PLUGIN AND PIPELINE

```
node('docker') {
    docker.image('golang:1.6').inside {

        stage 'Get sources'
        git url: 'https://github.com/hashicorp/terraform.git', tag: '

        stage 'Build'
        sh """#!/bin/bash -e
        mkdir -p /go/src/github.com/hashicorp
        ln -s `pwd` /go/src/github.com/hashicorp/terraform
        pushd /go/src/github.com/hashicorp/terraform
        make core-dev plugin-dev PLUGIN=provider-aws
        popd
        cp /go/bin/terraform-provider-aws .
        """


        stage 'Archive'
        archive "terraform-provider-aws"
    }
}
```

JENKINS KUBERNETES PLUGIN

- Dynamic Jenkins agents, running as Pods
- Multiple container support
  - One jnlp image, others custom
- Pipeline support for both agent Pod definition and execution will be in next version

JENKINS KUBERNETES PIPELINE

```
podTemplate(label: 'mypod', containers: [
        [name: 'jnlp', image: 'jenkinsci/jnlp-slave:alpine', args: '$
        [name: 'maven', image: 'maven:3-jdk-8', ttyEnabled: true, com
        [name: 'golang', image: 'golang:1.6', ttyEnabled: true, comma
    ]) {

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

        stage 'Get a Golang project'
        git url: 'https://github.com/hashicorp/terraform.git'
        container('golang') {
            stage 'Build a Go project'
            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-de
            """
        }
```

JENKINS PLUGINS RECAP

- Dynamic Jenkins agent creation
- Using JNLP slave jar
  - In complex environments need to use the `tunnel` option to connect internally
- Using the Cloud API
  - Not ideal for containerized workload
  - Agents take > 1 min to start provision and are kept around
  - Agents can provide more than one executor

JENKINS ONE SHOT EXECUTOR

Improved API to handle one off agents

Optimized for containerized agents

Plugins need to support it

GRACIÑAS

csanchez.org

csanchez

carlossg