



SCALING JENKINS WITH DOCKER SWARM, KUBERNETES OR MESOS?

[Carlos Sanchez](#) / [csanchez.org](#) / [@csanchez](#)



Watch online at carlossg.github.io/presentations

ABOUT ME

Engineer @ CloudBees, Private SaaS Edition Team

Contributor to

Jenkins Mesos plugin

Jenkins and Maven official Docker images

Author of Jenkins Kubernetes plugin

Long time OSS contributor at Apache, Eclipse, Puppet,...

**DOCKER DOCKER
DOCKER**





Kernel Sanders

@lstoll

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

BUT IT IS NOT TRIVIAL

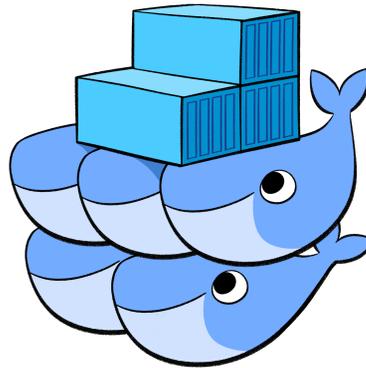


CLUSTER SCHEDULING

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



MESOS



kubernetes



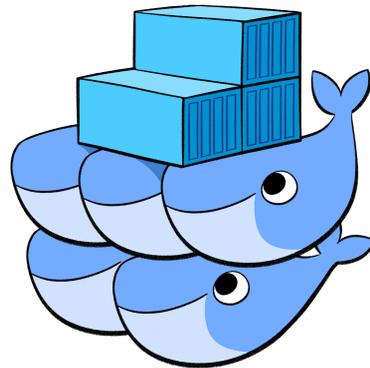
A distributed systems kernel



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

OFFICIAL REPOSITORY

jenkins ★

Last pushed: 11 days ago

Repo info Tags

Supported tags and respective `Dockerfile` links

- `latest`, `1.609.2` ([Dockerfile](#))

For more information about this image and its history, please see the [relevant manifest file \(library/jenkins\)](#) in the `docker-library/official-images` [GitHub repo](#).

Jenkins

The Jenkins Continuous Integration and Delivery server.

This is a fully functional Jenkins server, based on the Long Term Support release .



Jenkins

DOCKER PULL COMMAND

```
docker pull jenkins
```

DESCRIPTION

Official Jenkins Docker image

PUBLIC | AUTOMATED BUILD

jenkinsci/jnlp-slave

Last pushed: 6 days ago

Repo Info

Tags

Dockerfile

Build Details

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?



iCACHONDEO.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 \neq 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

Docker URL

Credentials  Add 

Connection Timeout

Read Timeout

Container Cap

Images

 **Docker Template**

Docker Image

[Container settings...](#)

Instance Capacity

Remote Filing System Root

Labels

Labels

ssh

Usage

Use this node as much as possible



Experimental Options...

Images

ID	<input type="text" value="evarga/jenkins-slave"/>
Labels	<input type="text"/>
Credentials	<div><input type="text" value="jenkins"/> ▼ Add</div>
Remote Filing System Root	<input type="text" value="/home/jenkins"/>
Remote FS Root Mapping	<input type="text"/>
Instance Cap	<input type="text"/>
DNS	<input type="text"/>
Port bindings	<input type="text"/>
Bind all declared ports	<input type="checkbox"/>
Hostname	<input type="text"/>
Idle termination time	<input type="text" value="5"/>
JavaPath	<input type="text"/>
JVM Options	<input type="text"/>
Docker Command	<input type="text"/>
LXC Conf Options	<input type="text"/>
Volumes	<input type="text"/>
Volumes From	<input type="text"/>
Run container privileged	<input type="checkbox"/>

Prefix Start Slave Command

Suffix Start Slave Command

Delete

JENKINS DOCKER PIPELINE

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

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

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

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

    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

Context Path

Side containers

Name

Container

Docker Image

Force pull



Supprimer

Name



Container

Docker Image

Force pull



Supprimer

Add a side container

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) ▾

 Add ▾

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 ?

Mesos Offer Selection Attributes ?

Additional Jenkins Slave JVM arguments ?

Additional Jenkins Slave Agent JNLP arguments ?

Mark this Slave Info as default for all Jobs ?

Use Docker Containerizer

Container Type

Docker

Docker Image ?

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

Add Port Mapping

Label String



Usage

Utilize this node as much as possible



Node Properties

Environment variables

List of key-value pairs

name

value



Delete

Add

Delete

Add Node Property ▼

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

1024



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: 'v0.10.0'

    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, con
    [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

THANKS

csanchez.org

