USING CONTAINERS FOR CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY

Carlos Sanchez

csanchez.org / @csanchez

cloudbees®

PARIS CONTAINER DAY

Watch online at carlossg.github.io/presentations

ABOUT ME

Engineer @ CloudBees, Scaling Jenkins

Author of Jenkins Kubernetes plugin

Contributor to Jenkins Mesos plugin & Jenkins and Maven official Docker images

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

Google Developers Experts

DOCKER DOCKER DOCKER

Kernel Sanders
@lstoll

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

USING CONTAINERS IS NOT TRIVIAL

Monolithic vs Microservices

Monolithic

Microservices

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 Enterprise

DOCKER AND JENKINS

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 REPOSITORY

jenkinsci/jenkins ☆

Last pushed: 8 hours ago

Repo Info Tags

Short Description

Jenkins Continuous Integration and Delivery server

Docker Pull Command

```
docker pull jenkinsci/jenkins
```

Full Description

Jenkins Continuous Integration and Delivery server.

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

Jenkins

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

Owner

jenkinsci

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.

JENKINS DOCKER PLUGINS

- Dynamic Jenkins agents with Docker plugin or Yet Another Docker Plugin
  - No support yet for Docker Swarm mode
- Isolated build agents and jobs
- Agent image needs to include Java, downloads slave jar from Jenkins master

JENKINS DOCKER PLUGINS

- Multiple plugins for different tasks
  - Docker build and publish
  - Docker build step plugin
  - CloudBees Docker Hub/Registry Notification
  - CloudBees Docker Traceability
- Great pipeline support

Images

ID	evarga/jenkins-slave
Labels	
Credentials	jenkins ▾
	Add
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

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-login') {

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

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

    pcImg.push();
  }
}
```

WHEN ONE MACHINE IS NO LONGER ENOUGH

- Running Docker across multiple hosts
- In public cloud, private cloud, VMs or bare metal
- HA and fault tolerant

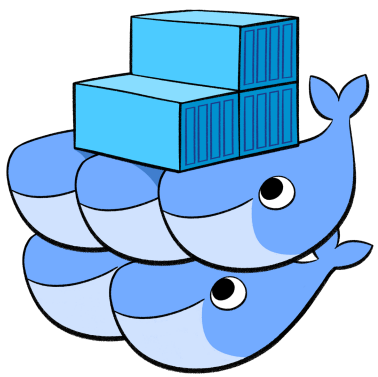
@DEVOPS_BORAT

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


MESOS




kubernetes

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

GROUPING CONTAINERS (PODS)

Example:

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

5 containers

STORAGE

Jenkins masters need persistent storage, agents (*maybe*)

Persistent volumes

- GCE disks
- GlusterFS
- NFS
- EBS
- etc

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

PERMISSIONS

```
containers: [...]
securityContext:
  fsGroup: 1000
volumes: [...]
```

Volumes which support ownership management are modified to be owned and writable by the GID specified in fsGroup

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)

Multiple networking options:

GCE, Flannel, Weave, Calico,...

One IP per Pod

Containers can find other containers in the same Pod using `localhost`

MEMORY LIMITS

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

Prevent containers for using more memory than allowed

Memory constraints translate to Docker --memory

https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/#how-pods-with-resource-limits-are-run

WHAT DO YOU THINK HAPPENS WHEN?

Your container goes over memory quota?


iCACHONDEO.COM

NEW JVM SUPPORT FOR CONTAINERS

JDK 8u131+ and JDK 9

```
$ docker run -m 1GB openjdk:8u131 java \
  -XX:+UnlockExperimentalVMOptions \
  -XX:+UseCGroupMemoryLimitForHeap \
  -XshowSettings:vm -version
VM settings:
    Max. Heap Size (Estimated): 228.00M
    Ergonomics Machine Class: server
    Using VM: OpenJDK 64-Bit Server VM
```

Running a JVM in a Container Without Getting Killed

https://blog.csanchez.org/2017/05/31/running-a-jvm-in-a-container-without-getting-killed

NEW JVM SUPPORT FOR CONTAINERS

```
$ docker run -m 1GB openjdk:8u131 java \
  -XX:+UnlockExperimentalVMOptions \
  -XX:+UseCGroupMemoryLimitForHeap \
  -XX:MaxRAMFraction=1 -XshowSettings:vm -version
VM settings:
    Max. Heap Size (Estimated): 910.50M
    Ergonomics Machine Class: server
    Using VM: OpenJDK 64-Bit Server VM
```

Running a JVM in a Container Without Getting Killed

https://blog.csanchez.org/2017/05/31/running-a-jvm-in-a-container-without-getting-killed

CPU LIMITS

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

CPU requests translates into Docker `--cpu-shares`

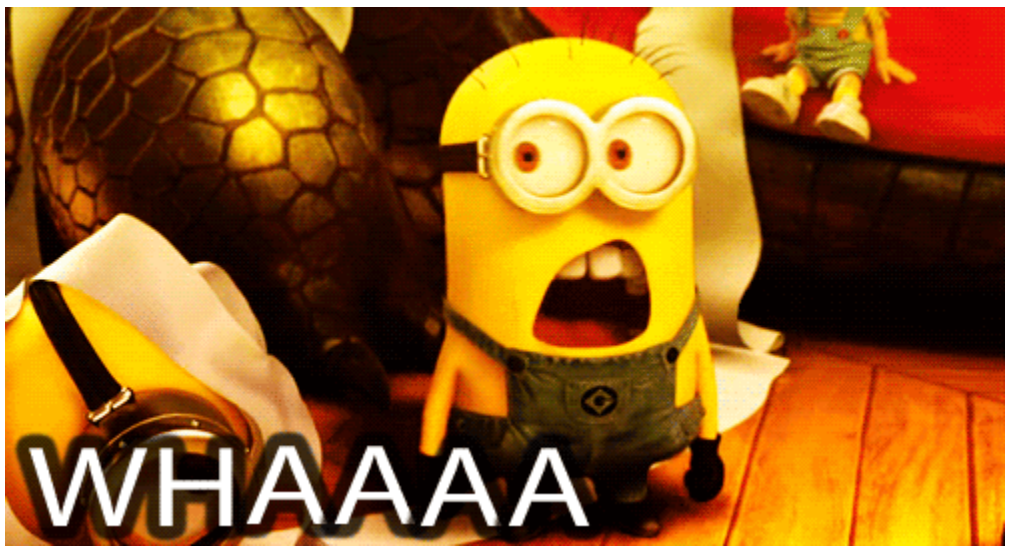
CPU limits translates into Docker `--cpu-quota`

https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/#how-pods-with-resource-limits-are-run

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

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
- Persistent workspace

JENKINS KUBERNETES PIPELINE

```
podTemplate(label: 'maven', containers: [
  containerTemplate(name: 'maven', image: 'maven:3.3.9-jdk-8-alpine
    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'
        }
      }
    }
  }
}
```

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-dev
        """
      }
    }
```

JENKINS PLUGINS CAVEATS

- 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 PLUGINS CAVEATS

- One Shot Executor
  - Improved API to handle one off agents
  - Optimized for containerized agents
  - Plugins need to support it

MERCI

csanchez.org

csanchez

carlossg

cloudbees®