Lightweight PaaS for Jenkins CI Environments with Docker

Josef Fuchshuber
QAware GmbH
http://www.qaware.de

June 25, 2014
About me

Job:
- Senior Software Architect (JEE)
- Manager of software development tools

Company: QAware GmbH, Munich
- Custom software company
- Customers:
  - Telecommunication industry
  - Automotive industry
Agenda

• Introduction to Docker

• Dynamic Build Slaves with Docker

• Lightweight Paas: Continuous Deployment with Jenkins and Docker
“Containerization is the new virtualization”
The Challenge

source: https://www.docker.io/the_whole_story
The Docker solution

source: https://www.docker.io/the_whole_story
Quick Facts: 🐋 docker

- Docker Inc. (formerly dotCloud Inc.)
- Introduced in March 2013
- Current version 0.11 is the release candidate for 1.0
- Huge Github community (12,000 stars and counting)
- Open source: Apache v2 licence
Why should we use Docker?

• It makes it easy to create lightweight, portable, and self-sufficient containers.
• Configure once, run anywhere
• Solving dependency hell
• Huge win for automation and deployments
• Containers are perfect for:
  – Continuous integration & test applications
  – Build services
  – Run services
  – Build your own Platform-as-a-Service (PaaS)
Docker is a client-server application

• Docker client and daemon can run on the same system.
  – You can connect a Docker client with a remote Docker daemon.
  – They communicate via sockets or through a RESTful API.
• Users interact with the client to command the daemon, e.g. to create, run, and stop containers.
• The daemon, receiving those commands, does the job, e.g. run a container, stop a container.
Some Docker vocabulary

- Container
- Image
- Layer
- Dockerfile
- Registry
- Repository
- Tag
Insights: Under the hood

Union file systems (UnionFS)
- Union file systems operate by creating layers.
- They combine layers into a single image.
- Supports: AUFS, btrfs, vfs and DeviceMapper

Namespaces & control groups
- Provide isolated workspace for containers.
- Controlling resource (CPU, memory, block I/O, network, etc.)

Container formats
- Wraps these all together:
  - Libcontainer
  - LXC
  - ...

...
VMs vs. Containers

Type 1 / Type 2 Virtualization

Hardware

Hypervisor 1

Hypervisor 2

Host OS

Guest OS

Bins/Libs

Apps

Containerization

Hardware

Docker

Host OS

Container

Container

Container
The Docker workflow

Source Code Repository

Dockerfile

Developer guy

Developer Host (Linux OS)

Docker Engine

Operation guy

Server Host (Linux OS)

Docker Engine

Container

Docker Image Registry

Container X

Container Y
Agenda

• Introduction to Docker

• Dynamic Build Slaves with Docker

• Lightweight Paas: Continuous Deployment with Jenkins and Docker
Example of a Jenkins infrastructure

- Jenkins (Master)
- SVN/Git
- Repositories (Maven, RPM, ...)
- SonarQube
- Jenkins Slave (Ubuntu Linux)
- Jenkins Slave (CentOS Linux)
- Jenkins Slave (Windows Server 2012)
- Jenkins Slave (Mac OS X Server)
What Docker can change

Jenkins (Master)

Docker Host

Jenkins Slave (Ubuntu Linux)

Jenkins Slave (CentOS Linux)

Jenkins Slave (Windows Server 2012)

Jenkins Slave (Mac OS X Server)

Docker Repository

SVN/Git

Repositories (Maven, RPM, ...)

SonarQube
The idea: The Docker build workflow

1. Developers look for base images in public/private Docker repositories
2. Optional: Customize it and push it back to the Docker repository
3. Jenkins administrator defines images as Jenkins slave
4. Define Jenkins job(s) for this Docker slave.
5. Run Jenkins job:
   1. Jenkins starts defined image as Docker container
   2. Jenkins runs the job inside the container
   3. Jenkins stops the container (tag it)
Quick Facts: Docker Plugin

• Links:
  – https://wiki.jenkins-ci.org/display/JENKINS/Docker+Plugin
  – https://github.com/jenkinsci/docker-plugin

• Features:
  – Dynamic provisioning of a Jenkins slave on a Docker host
  – Run a single build job
  – Tear-down the slave
  – Commit the container
Requirements for the container

• Connectable: SSH server

• Accessible: User (e.g. „jenkins“)

• Runnable: Java JDK

• Documentation: https://wiki.jenkins-ci.org/display/JENKINS/Docker+Plugin

• Ready-made jenkins slave: „evarga/jenkins-slave“
Why use dynamic slaves with Docker

- Fast startup
- Every job runs in its own clear container
- Job-parallelization is no problem
- Lazy resource binding and no long-running processes
- Devops: Separation of concerns
  - Developer: Worries about the container's inside
  - Ops: Worries about the container's outside
DEMO
Agenda

• Introduction to Docker

• Dynamic Build Slaves with Docker

• Lightweight Paas: Continuous Deployment with Jenkins and Docker
Definition of QAware‘s „TI architecture“

- JAR/WAR/EAR
- (Cron) jobs
- ...

- OS
- VM (Java, .NET, …)
- Server (db, web, …)
- Libraries
- ...

- Server
- Memory
- Network equipment
- ...

Application packages
System software
Hardware
Remote protocols
Why to use an own lightweight PaaS solution in your CI/CD process?

• You can‘t use a commercial PaaS.

• You have a complex TI architecture.

• Integrate early and often! We need an scalable automated build and deployment process.
Lightweight PaaS components

Packaged application

Jenkins

Lightweight Docker PaaS Solution

Docker base images

Docker daemon
What is Provisioning?

- Application deployment and server configuration.
- Prepare server and runtime environments; OS configuration.
- Prepare OS.
- Prepare virtualization layer / cloud infrastructure.
- Prepare hardware.

Application is running!

- Application Provisioning
- Server Provisioning
- Bootstrapping
- Bare Metal Provisioning
- Hardware

Application packages
System software
Hardware
Docker makes „Infrastructure as Code“ easy

- Load and start images
- Prepare images with Dockerfiles
- Base images
- Docker runtime
The Jenkins build-pipeline for Docker deployments

- Compile, test & package
- Create app packages
- Provision container
- Run integration-test
- Deploy & run test env.

- Build repo
- Artifact repo
- Docker repo
- Docker file(s)
Quick Facts: Docker build publish Plugin

- Links:
  - [https://wiki.jenkins-ci.org/display/JENKINS/Docker+build+publish+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Docker+build+publish+Plugin)
  - [https://github.com/jenkinsci/docker-build-publish-plugin](https://github.com/jenkinsci/docker-build-publish-plugin)

- Features:
  - Only a Dockerfile needed to build your project
  - Tag the image build
  - Publish to private or public Docker repository
Quick Facts: Docker build step plugin

• Links:
  – https://wiki.jenkins-ci.org/display/JENKINS/Docker+build+step+plugin

• Features:
  – Execute Docker commands into your job as a build step
  – Export build variables
    • DOCKER_CONTAINER_IDS – The IDs of created/started containers
    • DOCKER_IP_$HOSTNAME – The IP of running container with hostname $HOSTNAME
DEMO
Thank You To Our Sponsors

Platinum

![CloudBees Logo]

Gold

![zend Logo]  The PHP Company
![MidVision Logo]  Release the innovation
![codecenric Logo]

Silver

![XebiaLabs Logo]  the deployment automation company

Corporate

![Cloudera Logo]

Community

![Java User Group Berlin Brandenburg Logo]
![Lightweight Java User Group Munich Logo]