Improving the CI workflows with Docker

Vojtěch Juránek
(https://github.com/vjuranek)

JBoss - a division by Red Hat

7. 2. 2015, Developer conference, Brno

Outline

- Ideas how to use containers for
  - minimizing false negatives during testing,
  - making integration tests more easy,
  - creating unified dev/QA/prod environment,
  - etc.
- Jenkins Docker plugins.
- Some other useful tools.

Containers, Docker and all the other things

- ▸ Docker, Docker, Docker . . . a buzzword for ∼ 1.5 year.
- So, is the Docker a silver bullet for CI/CD or testing?
- Surprisingly . . .
- . . . no, it isn't :-)
- . . . but still can help, at least little bit.

Containers, Docker and all the other things

- ▸ Docker, Docker, Docker . . . a buzzword for $\sim$ 1.5 year.
- So, is the Docker a silver bullet for CI/CD or testing?
- Surprisingly . . .
- . . . no, it isn't :-)
- . . . but still can help, at least little bit.

Containers, Docker and all the other things

- Docker, Docker, Docker . . . a buzzword for ∼ 1.5 year.
- So, is the Docker a silver bullet for CI/CD or testing?
- Surprisingly . . .
- . . . no, it isn't :-)
- . . . but still can help, at least little bit.

Containers, Docker and all the other things

- ▸ Docker, Docker, Docker . . . a buzzword for $\sim$ 1.5 year.
- So, is the Docker a silver bullet for CI/CD or testing?
- Surprisingly . . .
- . . . no, it isn't :-)
- . . . but still can help, at least little bit.

Containers, Docker and all the other things

- ▸ Docker, Docker, Docker . . . a buzzword for $\sim$ 1.5 year.
- So, is the Docker a silver bullet for CI/CD or testing?
- Surprisingly . . .
- . . . no, it isn't :-)
- . . . but still can help, at least little bit.

Common CI issues

- False negatives due to issues with build environment:
    - missing or misconfigured tools,
    - conflicting builds,
    - some other kind of environment issue.
- Unable to reproduce or debug a failure:
    - VM is already destroyed or returned back to the machine pool,
    - another build already has done some changes, killed some processes etc.

Common CI issues

- Different dev, QA and prod environments.
- Have you ever heard (or said:-) *"It works on my machine just fine!"*?
  - I did, many times, usually as *"Jenkins is broken, because it fails tests which are passing on my machine!"*
- In general, production environment is not easily available:
  - can be pretty hard to implement in test suite,
  - even setup itself can be quite time consuming.
- Implement proper test environment or mock required services could be so expensive, that some integration test are not implemented at all.
- Speed! CI feedback has to be as quick as possible.

Fig. from blog.codinghorror.com

Tip #1: Define and share your dev/prod environment

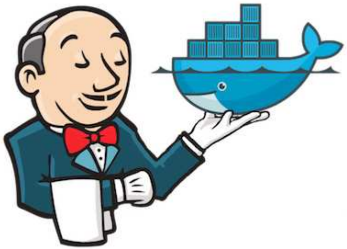
- Unified environment for all involved teams as a container.
- Easy to do changes, no need to wait for admins to do the changes on your test servers.
- You can experiment and eventually share the changes in environment very easily.
- Other folks needn't to have knowledge how to setup all parts of your app properly and spend any time with configuration.
- If you hit any issue, you can easily share with others for investigation.
- There are of course other ways how to achieve it (tools like Puppet or distributing VMs), but Docker makes it super easy and fast.
- DevOps!

Tip #2: Use containers for build isolation

- Run each build in fresh container.
- Very fast boot (c.f. to provisioning machine from Beaker of even to boot new machine in the cloud or VM).
- No interference with other builds or stale processes from previous build.
- Well defined environment for each build.
- Avoid ”dependency hell” - OS has package ABC in version X, but my app needs it in version Y, while another needs it in version Z.

Jenkins & Docker

Sounds good? So, how to do it easily?
Jenkins!



- Jenkins CI has (AFAIK) the best Docker support in CI field - but still far from perfect.
  - No blocker, but various minor, more or less annoying issues.
  - Some useful features still missing.
- However, other CI tools has limited or no Docker support at all.

Jenkins Docker plugin (https://wiki.jenkins-ci.org/display/JENKINS/Docker+Plugin)

- "Cloud" provider of Docker slaves.
- Jenkins will dynamically start new containers and connect them as ordinary Jenkins slaves.
- Containers are created based on jobs waiting in the queue (according to labels assigned to jobs and containers).
- Once slaves are idle for some time, they are removed from Jenkins as well as from Docker.

- Container has to have JVM and provide SSH service, you can use e.g.:
  - Fedora slave: `docker pull vjuranek/jenkins-ssh-slave`
    ▸ Docker Hub link
  - Ubuntu slave: `docker pull evarga/jenkins-slave`
    ▸ Docker Hub link

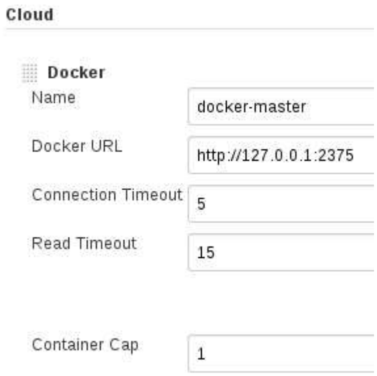
Jenkins Docker plugin - configuration

Docker configuration:

- Docker server has to run with TCP port enabled:
- Add -H tcp://127.0.0.1:2375 to Docker exec start options (/etc/systemd/system/docker.service on FC 20)

Jenkins global configuration:

- Setup Docker URL.
- Connection timeouts.
- And maxim # of containers which can run simultaneously.



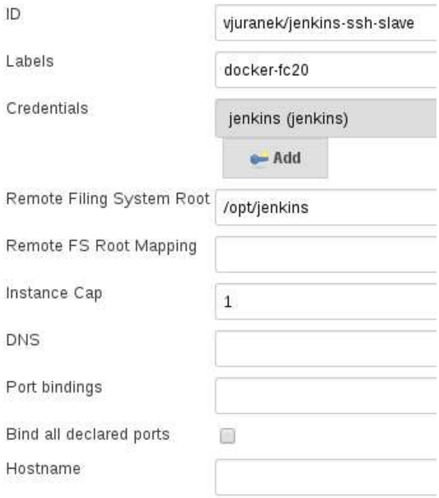
Cloud

Docker

Name	docker-master
Docker URL	http://127.0.0.1:2375
Connection Timeout	5
Read Timeout	15
Container Cap	1

Jenkins Docker plugin - configuration

- Assign labels to your images.
- Setup SSH credentials
- Configure the image (more in advanced settings).
- Tight you jobs to required label.



ID	vjuranek/jenkins-ssh-slave
Labels	docker-fc20
Credentials	jenkins (jenkins)
	Add
Remote Filing System Root	/opt/jenkins
Remote FS Root Mapping	
Instance Cap	1
DNS	
Port bindings	
Bind all declared ports	☐
Hostname	

Create your custom container

Continue further with defining your own Jenkins slave image!

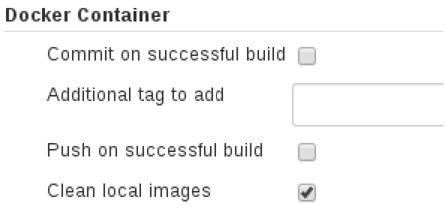
- Container needs to have JVM and Jenkins needs to be able to ssh to the container.
- Example Docker file based on Fedora:

```
FROM fedora:20
MAINTAINER Vojtech Juranek <vjuranek@redhat.com>

# Execute system update
RUN yum -y update && yum clean all

# Install JDK and ssh server
RUN yum -y install java-1.7.0-openjdk-devel openssh-server && yum clean all

# Generate ssh key
RUN ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key -N ''

# Create Jenkins user and Jenkins group
RUN groupadd -r jenkins -g 1001 && useradd -u 1001 -r -g jenkins -m -d /opt/jenkins -
RUN echo "jenkins:jenkins" | chpasswd

# Expose standard SSH port
EXPOSE 22

CMD ["/usr/sbin/sshd", "-D"]
```

Jenkins Docker plugin

Per project configuration:

- You can commit and push container when build succeeds.
- Start and stop containers during/after the build.



Docker Container

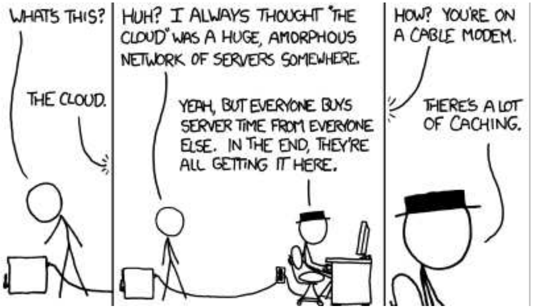
Commit on successful build	☐
Additional tag to add	
Push on successful build	☐
Clean local images	☑

Pros and cons:

+ Complex plug, which is able to handle a lot of things.
− Complex configuration.
− Sometimes not very easy to find out what could be wrong (especially when there is some bug in the plugin itself).

Tip #3: Embed you application into container

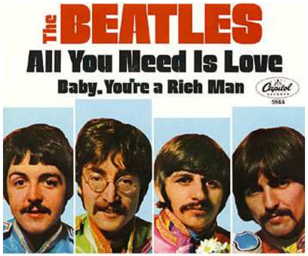
- You can go one more step further and distribute you application embedded in the container.
    - Not necessarily means that you have to deliver you app to customers as a container, you can use it just for testing.
- Describe you build process in Docker file.
- Re-spins should be quite fast as Docker does **a lot of caching**.
- Sharing is very easy with Docker registry.


WHAT'S THIS?
THE CLOUD.
HUH? I ALWAYS THOUGHT "THE CLOUD" WAS A HUGE, AMORPHOUS NETWORK OF SERVERS SOMEWHERE.
YEAH, BUT EVERYONE BUYS SERVER TIME FROM EVERYONE ELSE. IN THE END, THEY'RE ALL GETTING IT HERE.
HOW? YOU'RE ON A CABLE MODEM.
THERE'S A LOT OF CACHING.

Part of xkcd #908

Jenkins Docker build publish plugin

(https://wiki.jenkins-ci.org/display/JENKINS/Docker+build+publish+Plugin)



Source: Wikipedia

All you need is love ...
and a Docker file!

- Plugin builds a Docker file and push the image into repository and registry.
- Useful especially for Docker-based applications.
- Useful mainly for private Docker registry, for public you can easily use Docker Hub automated builds to do the work for your.
    - Could be e.g. the last step in dev build flow - project is built and Docker image is created.
    - Create image can be subsequently consumed by QA, stage somewhere etc.

Tip #4: Use Docker for better integration tests

- Integration tests are sometimes pretty far from what happen in the production, as
  - there could be lots of simplifications, e.g. client/server apps communicate over loopback during the test,
  - it's pretty hard to implement realistic scenarios,
  - even setup itself can be quite time consuming.
- Implement proper test environment or mocking of the services is so expensive, that some integration test are not implemented at all.

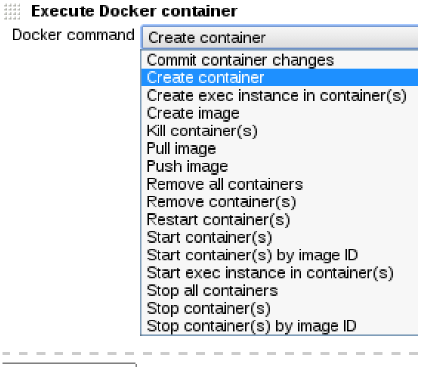
- With Docker, you can start real service during the integration test phase.
  - Several ways how to achieve it.
  - Start several services as Docker containers is pretty quick.
  - Better than mocking or even running service from the test (e.g. network traffic is over vLANs etc.).
- Testing is much more realistic and therefore valuable.

Jenkins Docker build step plugin

(https://wiki.jenkins-ci.org/display/JENKINS/Docker+build+step+plugin)

- Allows you to add various Docker operation as a build step into your Jenkins project.
- Similar to Docker plugin you have to configure Docker URL in global Jenkins configuration.

- Just select what you want to do in you project.
- Allows to wait for defined ports of started container to make sure container is fully running.
- Exports IP addresses as env. variables which can be consumed by other parts, etc.


Execute Docker container

Docker command: Create container

- Unfortunately currently not compatible with Docker plugin.

Some other useful Docker tools from the Java world

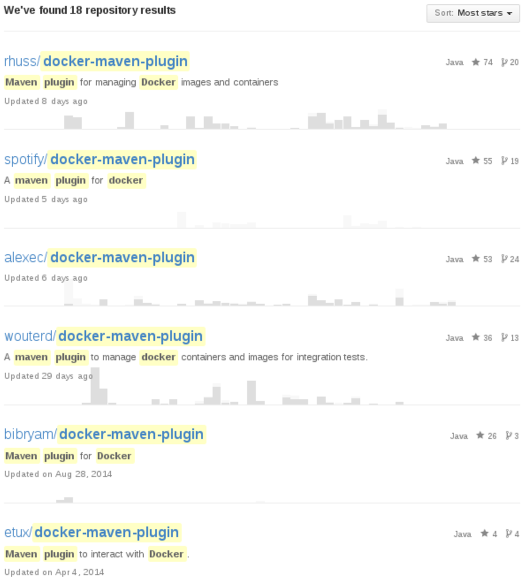
- Jenkins is language agnostic, can be used with any kind of project.
- For various languages and frameworks exists dedicated tools.
- Not always you want Jenkins to be a central part of your processes.
- Sometimes you needs to run e.g. integration tests on you local machine and start Docker directly from your test suite or build tool.
- Examples of such tools from the Java world:
  - Maven Docker plugin(s)
  - Arquillian Cube

Not a Java developer?

- Never mind, there are tools also for other languages.
- Or implement Docker extension for your favorite framework.
- There are Docker client libraries for many languages, see Docker Remote API Client Libraries list.

Maven plugins for Docker

- There are many *docker-maven* plugins.
- E.g. https://github.com/alexec/docker-maven-plugin



We've found 18 repository results

Sort: Most stars

rhuss/docker-maven-plugin — Java ★ 74 20
Maven plugin for managing Docker images and containers
Updated 8 days ago

spotify/docker-maven-plugin — Java ★ 55 19
A maven plugin for docker
Updated 5 days ago

alexec/docker-maven-plugin — Java ★ 53 24
Updated 6 days ago

wouterd/docker-maven-plugin — Java ★ 36 13
A maven plugin to manage docker containers and images for integration tests.
Updated 29 days ago

bibryam/docker-maven-plugin — Java ★ 26 3
Maven plugin for Docker
Updated on Aug 28, 2014

etux/docker-maven-plugin — Java ★ 4 4
Maven plugin to interact with Docker.
Updated on Apr 4, 2014

Docker Maven plugin

- Place Docker file into `src/main/docker/your-app`

```
 [...]
<plugin>
  <groupId>com.alexecollins.docker</groupId>
  <artifactId>docker-maven-plugin</artifactId>
  <configuration>
    <version>1.15</version>
    <host>http://127.0.0.1:2375</host>
  </configuration>
  <executions>
    <execution>
      <id>start-conatiner</id>
      <phase>pre-integration-test</phase>
      <goals>
        <goal>start</goal>
      </goals>
    </execution>
     [...]
  </executions>
</plugin>
 [...]
```

Integration with other tools

Maven Failsafe plugin:

```
 [...]
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-failsafe-plugin</artifactId>
  <version>2.18.1</version>
  <configuration>
    <systemPropertyVariables>
      <demo.ldap.ip>${docker.ispn-ldap.ipAddress}</demo.ldap.ip>
    </systemPropertyVariables>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>integration-test</goal>
        <goal>verify</goal>
      </goals>
    </execution>
  </executions>
</plugin>
 [...]
```

Arquillian Cube

- Arquillian is a test runner with dependency injection, container life cycle management and other features.
- Arquillian Cube is Arquillian integration with Docker.
- Currently `1.0.0.Alpha` - you can expect further development.

Container definition in dedicated file `arquillian.xml`:

```
<extension qualifier="docker">
  <property name="serverVersion">1.15</property>
  <property name="serverUri">http://localhost:2375</property>
  <property name="dockerContainers">
      ldap:
        image: vjuranek/docker-maven-demo_ispn-ldap
        exposedPorts: [389/tcp]
        await:
          strategy: polling
          sleepPollingTime: 2 s
  </property>
</extension>

<container qualifier="containerless" default="true">
  <configuration>
    <property name="containerlessDocker">ldap</property>
    <property name="embeddedPort">389</property>
  </configuration>
</container>
```

Demo

- Trivial app which queries LDAP.
- LDAP server launched in the container.
- Available on GitHub.

- Jenkins Docker build step plugin.
- Docker Maven plugin.

- Arquillian Cube, if some time remains.

Putting it all together

- Define you dev/prod environment in a Dockerfile and share the container(s) the other teams/team members.
- Use it for defining CI slave which exactly fits your needs.
- Build you application as a container.
- Use is for integration test which are executed in realistic production environment built on top of other containers.
- Automate everything in your CI system.

Thank you for your attention!

How do like it? Was it useful?
Let me know on http://devconf.cz/f/88

Questions?