Large-Scale Software Development

Tools and Methods
Scenarios - actions

1. Single developer, multiple changes — Version control system
2. Many developers, multiple changes — Distributed version control system
3. Many groups of developers, multiple changes — Package management system
4. Building artefacts based on multiple files with dependencies — build scripts
5. Conducting multiple actions with inter-dependencies on multiple files ... — Flexible build system
6. Automatically sensing changes and conducting such actions based on changes — Continuous integration tools
Scenarios - configuration

• Configuration files
• Setup scripts
• Container environments
• Virtual Machines
Version control — Git
Scenarios - actions

1. Single developer, multiple changes — Version control system
2. Many developers, multiple changes — Distributed version control system
3. Many groups of developers, multiple changes — Package management system
4. Building artefacts based on multiple files with dependencies — build scripts
5. Conducting multiple actions with inter-dependencies on multiple files ... — Flexible build system
6. Automatically sensing changes and conducting such actions based on changes — Continuous integration tools
Version control - GIT
Sample Disorganized Project

• “Hey, Anders, could you send me a copy of those changes you made last Tuesday?”
• “Ola, this function doesn’t work anymore. Did you change something?”
• “Sorry, I can’t seem to find those old classes. I guess you’ll just have to re-implement them.”
• “OK, we’ve all been working hard for the last week. Now let’s integrate everyone’s work together.”
What is version control?

Basic functionality:
- keep track of changes made to files (allows roll-backs)
- merge the contributions of multiple developers

Benefits:
- facilitates backups
- increased productivity (vs manual version control)
- encourages experimentation
- helps to identify/fix conflicts
- makes source readily available – less duplicated effort
Additional benefits

Accountability
  who wrote the code?
  do we have the rights to it?

Support software engineering
  hooks for peer reviews

Software branches
  different versions of software need to be maintained, ensure bug fixes shared

Record Keeping
  Commit logs may tie to issue tracking system or be used to enforce guidelines
More Benefits

Support Distribution of Work
  Telecommuting, outsourcing, open-source projects
  Use in conjunction with “good communication habits” – via email etc.

Rapid Development (XP/Agile)
  Supports frequent refactoring
  Helps automate frequent system builds
The Big Picture

Branching

- Git sees commit this way...
- Branch annotates which commit we are working on
git branch experiment
```
$ git branch
  * default
  experiment
```
git checkout experiment
git commit
git checkout default
git commit

C0 → C1 → C4

C2 → C3

default

HEAD

experiment
git checkout experiment
git commit
Merging

• What do we do with this mess?
  – Merge them
Merging

• Steps to merge two branch
  – Checkout the branch you want to merge onto
  – Merge the branch you want to merge
git checkout default
git checkout default

git merge experiment
git commit
git checkout default
git merge experiment
Scenarios - actions

1. Single developer, multiple changes — Version control system
2. Many developers, multiple changes — Distributed version control system
3. Many groups of developers, multiple changes — Package management system
4. Building artefacts based on multiple files with dependencies — build scripts
5. Conducting multiple actions with inter-dependencies on multiple files ... — Flexible build system
6. Automatically sensing changes and conducting such actions based on changes — Continuous integration tools
Package management systems

```
dpkg
  (>= 5.1.1alpha+20120614)
  (>= 1.32)
  (>= 1.23)
  ————>
  libbz2-1.0
  liblzma5
  libselinux1
  tar
```
Dependency management issues

• Is a request to modify the current software component graph satisfiable?
  – Are additions compatible with other components?
  – Are deletions safe with respect to other dependencies?

• Given a component, determine versions of other components we can safely rely on
Dependency management as satisfiability

\[(a \lor b \lor c) \land (d \lor e \lor f) \ldots = \text{TRUE}\]

\[(a \lor b \lor c) \land (-c) \land (-b \lor -a) \ldots = \text{TRUE}\]

Rules

A requires B provided by B1, B2, B3
Rule: \((-A \lor B1 \lor B2 \lor B3)\)

A conflicts with B provided by B1, B2, B3
3 Rules: \((-A \lor -B1), (-A \lor -B2), (-A \lor -B3)\)
Dependency management issues

• Y depends on X >= 1.8. X makes binary incompatible changes from v. 1.9 to v. 2.0...
• Can components be installed from local sources as well as from remote?
• Should OS-specific dependency management or language-specific be used?
# Software package management systems

<table>
<thead>
<tr>
<th>Name</th>
<th>Environment</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>NuGet</td>
<td>.Net CLR</td>
<td>XML</td>
</tr>
<tr>
<td>Gradle</td>
<td>JVM</td>
<td>XML</td>
</tr>
<tr>
<td>dpkg/APT</td>
<td>Linux</td>
<td>Ar archive</td>
</tr>
<tr>
<td>Rubygems</td>
<td>Ruby</td>
<td>Ruby</td>
</tr>
<tr>
<td>MSI</td>
<td>Windows</td>
<td>In-file DB</td>
</tr>
<tr>
<td>BSD Ports</td>
<td>OS X/Linux/BSD</td>
<td>Makefile</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Maven

mvn -h

Life cycles
- Clean
- Default
- Site

validate
compile
test
package
verify
install
deploy
Olas-MacBook-Pro:java-petclinic olale$ mvn graphwalker:test
[INFO] Scanning for projects...
[ ... ]
[INFO] >>> graphwalker-maven-plugin:3.4.2:test (default-cli) > [graphwalker]test-compile @ java-petclinic >>>
[INFO] [INFO] --- graphwalker-maven-plugin:3.4.2:generate-sources (generate-sources) @ java-petclinic ---
[INFO] [INFO] --- graphwalker-maven-plugin:3.4.2:validate-models (default-cli) @ java-petclinic ---
[INFO] [INFO] --- graphwalker-maven-plugin:3.4.2:generate-sources (default-cli) @ java-petclinic ---
[INFO] [INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ java-petclinic ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 10 resources
[INFO] [INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ java-petclinic ---
[INFO] Nothing to compile - all classes are up to date
[INFO] [INFO] --- graphwalker-maven-plugin:3.4.2:validate-test-models (default-cli) @ java-petclinic ---
Maven — structure

mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
Maven — Configuration

<parent>
  <groupId>org.graphwalker.example</groupId>
  <artifactId>graphwalker-example</artifactId>
  <version>3.4.2</version>
</parent>

<artifactId>java-petclinic</artifactId>
Automation and languages for ... development

Multiple developers/languages/components

CI work automation

Test automation, packaging, ...

Java

Build tools

JVM

C

Compiler

OS

ASM
Jenkins

Workflow automation tool
Jenkins

Workflow automation tool - pipelines

```groovy
node { // <1>
  stage('Build') { // <2>
    sh 'make' // <3>
  }

  stage('Test') {
    sh 'make check'
    junit 'reports/**/*.xml' // <4>
  }

  stage('Deploy') {
    sh 'make publish'
  }
}
```

Groovy (JVM-based language)
Travis CI

A pull request is created

GitHub tells Travis CI the build is mergeable

Hooray! Your build passes!

Travis CI updates the PR that it passed

You merge in the PR goodness
Automation and languages for ... deployment

Chef/Puppet/Ansible/Salt/Terraform

Xen/Solaris Zones, Docker/Chroot

Programming language

Automation of setup

Isolation of components

Single file, structured non-programming language. Requirements installed, updated, started manually. Deployment through manual scripts
Chef/Puppet/Ansible/Salt/Terraform/…

- Automates setup of environments for development/deployment
- Introduces programming languages for your infrastructures
  - Declarative (Terraform/Puppet) - What is the desired end-state?
  - Imperative (Chef/Ansible) - How do you achieve it?
Comparison

Imperative Shell Code

```bash
if [ 0 -ne $(getent passwd elmo > /dev/null)?$ ]
then
  useradd elmo --gid sysadmin -n
fi

GID=`getent passwd elmo | awk -F: '{print $4}'`
GROUP=`getent group $GID | awk -F: '{print $1}'`

if [ "$GROUP" != "$GID" ] && [ "$GROUP" != "sysadmin" ]
then
  usermod --gid $GROUP $USER
fi

if [ `getent group sysadmin | awk -F: '{print $1}'` == "" ]
then
  groupadd sysadmin
fi
```

Declarative Puppet Code

```puppet
user { 'elmo':
  ensure => present,
  gid => 'sysadmin',
}

group { 'sysadmin':
  ensure => present,
}
```
Jenkins
Scenarios - actions

1. Single developer, multiple changes — Version control system
2. Many developers, multiple changes — Distributed version control system
3. Many groups of developers, multiple changes — Package management system
4. Building artefacts based on multiple files with dependencies — build scripts
5. Conducting multiple actions with inter-dependencies on multiple files ... — Flexible build system
6. Automatically sensing changes and conducting such actions based on changes — Continuous integration tools
Containers (Docker in particular)
Why containers
Market View: Evolution of IT

1995

Thick, client-server app on thick client

Well-defined stack:
- O/S
- Runtime
- Middleware

Monolithic
Physical Infrastructure

2015

Thin app on mobile, tablet

Assembled by developers using best available services

Running on any available set of physical resources (public/private/virtualized)
Challenges

2015

How to ensure services interact consistently, avoid dependency hell

How to migrate & scale quickly, ensure compatibility

How to avoid n X n different configs

Assembled by developers using best available services

Running on any available set of physical resources (public/private/virtualized)

Thin app on mobile, tablet

Assembled by developers using best available services

Running on any available set of physical resources (public/private/virtualized)

Thin app on mobile, tablet
The Challenge

Multiplicity of Stacks

- Static website
  - nginx 1.5 + modsecurity + openssl + bootstrap 2
- Background workers
  - Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs
- Web frontend
  - Ruby + Rails + sass + Unicorn
- User DB
  - postgresql + pgv8 + v8
- Queue
  - Redis + redis-sentinel
- Analytics DB
  - hadoop + hive + thrift + OpenJDK
- API endpoint
  - Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client

Multiplicity of hardware environments

- Development VM
- QA server
- Customer Data Center
- Public Cloud
- Production Cluster
- Production Servers
- Contributor's laptop

Do services and apps interact appropriately?

Can I migrate smoothly and quickly?

The Challenge

- Multiplicity of Stacks
- Multiplicity of hardware environments

Do services and apps interact appropriately?

Can I migrate smoothly and quickly?

The Challenge

- Multiplicity of Stacks
- Multiplicity of hardware environments

Do services and apps interact appropriately?

Can I migrate smoothly and quickly?

The Challenge

- Multiplicity of Stacks
- Multiplicity of hardware environments

Do services and apps interact appropriately?

Can I migrate smoothly and quickly?

The Challenge

- Multiplicity of Stacks
- Multiplicity of hardware environments

Do services and apps interact appropriately?

Can I migrate smoothly and quickly?

The Challenge

- Multiplicity of Stacks
- Multiplicity of hardware environments

Do services and apps interact appropriately?

Can I migrate smoothly and quickly?

The Challenge

- Multiplicity of Stacks
- Multiplicity of hardware environments

Do services and apps interact appropriately?

Can I migrate smoothly and quickly?
## Results in N X N compatibility nightmare

<table>
<thead>
<tr>
<th></th>
<th>Development VM</th>
<th>QA Server</th>
<th>Single Prod Server</th>
<th>Onsite Cluster</th>
<th>Public Cloud</th>
<th>Contributor’s laptop</th>
<th>Customer Servers</th>
</tr>
</thead>
</table>
Perhaps A useful analogy…
Do I worry about how goods interact (e.g., coffee beans next to spices)?
Can I transport quickly and smoothly (e.g., from boat to train to truck)?

Multiplicity of Goods

Multiplicity of methods for transporting/storing

Cargo Transport Pre-1960
Also an NxN Matrix

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
Solution: Intermodal Shipping Container

A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.

...in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another.

Do I worry about how goods interact (e.g. coffee beans next to spices)?

Can I transport quickly and smoothly (e.g. from boat to train to truck)?

Multiplicity of Goods

Multiplicity of methods for transporting/storing
This eliminated the NXN problem…
and spawned an Intermodal Shipping Container Ecosystem

- 90% of all cargo now shipped in a standard container
- Order of magnitude reduction in cost and time to load and unload ships
- Massive reduction in losses due to theft or damage
- Huge reduction in freight cost as percent of final goods (from >25% to <3%)
  -> massive globalizations
- 5000 ships deliver 200M containers per year
Docker is a shipping container system for code

An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container...

...that can be manipulated using standard operations and run consistently on virtually any hardware platform
Or...put more simply

Developer: Build Once, Run Anywhere (Finally)

Operator: Configure Once, Run Anything

Can I migrate smoothly and quickly?

Do services and apps interact appropriately?

Multiplicity of hardware environments

Development VM  QA server  Customer Data Center  Public Cloud  Production Cluster  Contributor's laptop

Static website  User DB  Web frontend  Queue  Analytics DB
Docker solves the NXN problem

<table>
<thead>
<tr>
<th>Static website</th>
<th>Web frontend</th>
<th>Background workers</th>
<th>User DB</th>
<th>Analytics DB</th>
<th>Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Docker containers" /></td>
<td><img src="image2" alt="Docker containers" /></td>
<td><img src="image3" alt="Docker containers" /></td>
<td><img src="image4" alt="Docker containers" /></td>
<td><img src="image5" alt="Docker containers" /></td>
<td><img src="image6" alt="Docker containers" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Development VM</th>
<th>QA Server</th>
<th>Single Prod Server</th>
<th>Onsite Cluster</th>
<th>Public Cloud</th>
<th>Contributor’s laptop</th>
<th>Customer Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image7" alt="Development VM" /></td>
<td><img src="image8" alt="QA Server" /></td>
<td><img src="image9" alt="Single Prod Server" /></td>
<td><img src="image10" alt="Onsite Cluster" /></td>
<td><img src="image11" alt="Public Cloud" /></td>
<td><img src="image12" alt="Contributor’s laptop" /></td>
<td><img src="image13" alt="Customer Servers" /></td>
</tr>
</tbody>
</table>
What is Docker?

_Docker is an open-source project that automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating system–level virtualization on Linux._

[Source: en.wikipedia.org]
Docker: Name

docker [naut.]: der Dockarbeiter, der Hafenarbeiter

- Provide a uniformed wrapper around a software package: «Build, Ship and Run Any App, Anywhere» [www.docker.com]
  - Similar to shipping containers: The container is always the same, regardless of the contents and thus fits on all trucks, cranes, ships, ...
Docker vs. Virtual Machine

Source: https://www.docker.com/whatisdocker/
Docker Technology

- libvirt: Platform Virtualization
- LXC (LinuX Containers): Multiple isolated Linux systems (containers) on a single host
- Layered File System

[Source: https://docs.docker.com/terms/layer/]

[Image: Diagram of Docker architecture showing layers and components such as Base Image, Debian, Container, Image, Kernel, and writable layers.]
Docker History

- 2013-03: Releases as Open Source
- 2013-09: Red Hat collaboration (Fedora, RHEL, OpenShift)
- 2014-03: 34th most starred GitHub project
- 2014-05: JAX Innovation Award (most innovative open technology)
Run Platforms

• Various Linux distributions (Ubuntu, Fedora, RHEL, Centos, openSUSE, ...)
• Cloud (Amazon EC2, Google Compute Engine, Rackspace)
• 2014-10: Microsoft announces plans to integrate Docker with next release of Windows Server
Hello World

Simple Command - Ad-Hoc Container

- `docker run ubuntu echo Hello World`
- `docker images [-a]`
- `docker ps -a`
Terminology - Image

• Persisted snapshot that can be run
  – *images*: List all local images
  – *run*: Create a container from an image and execute a command in it
  – *tag*: Tag an image
  – *pull*: Download image from repository
  – *rmi*: Delete a local image
    • This will also remove intermediate images if no longer used
Terminology - Container

• Runnable instance of an image
  – *ps*: List all running containers
  – *ps –a*: List all containers (incl. stopped)
  – *top*: Display processes of a container
  – *start*: Start a stopped container
  – *stop*: Stop a running container
  – *pause*: Pause all processes within a container
  – *rm*: Delete a container
  – *commit*: Create an image from a container
Image vs. Container

Base Image
\textit{ubuntu:latest}

New Image
\textit{iid1}

Container
\textit{cid1}

Container
\textit{cid3}

Container
\textit{cid4}

Container
\textit{cid2}

Commit

Run

Base Image

Run

Run

Cmd $\rightarrow$ new state
Dockerfile

- Create images automatically using a build script: «Dockerfile»
- Can be versioned in a version control system like Git or SVN, along with all dependencies
- Docker Hub can automatically build images based on dockerfiles on Github
Dockerfile Example

• Dockerfile:
  – FROM ubuntu
    ENV DOCK_MESSAGE Hello My World
    ADD dir /files
    CMD ["bash", "someScript"]
• docker build [DockerFileDir]
• docker inspect [imageId]
Mount Volumes

- `docker run -ti -v /hostLog:/log ubuntu`

- Run second container: Volume can be shared
  ```bash
  docker run -ti --volumes-from firstContainerName ubuntu
  ```
Publish Port

- `docker run -t -p 8080:80 ubuntu nc -l 80`
  - Map container port 80 to host port 8080
  - Check on host: `nc localhost 8080`

- **Link with other docker container**
  - `docker run -ti --link containerName:alias ubuntu`
  - See link info with `set`
Around Docker

• Docker Images: Docker Hub
• Vagrant: «Docker for VMs»
• Automated Setup
  – Puppet, Chef, Ansible, ...
Docker Hub

• Public repository of Docker images
  – [https://hub.docker.com/](https://hub.docker.com/)
  – docker search [term]

• Automated: Has been automatically built from Dockerfile
  – Source for build is available on GitHub