IF1006 – DevOps Software delivery way: DevOps Pipeline

Fish

@fisholito

jfsc@cin.ufpe.br

What is DevOps?

A set of practices to help organizations to deliver software fast without loss quality – [Culture]

Thinking in process such as practices

- Continuous Delivery/Deployment (Continuous integration, deployment every time)

- Treat operations personnel as first class citizen.

- Promote and support change of roles and sharing of knowledge.

- Apply software engineering disciplines on infrastructure code development (eg. shell scripts)
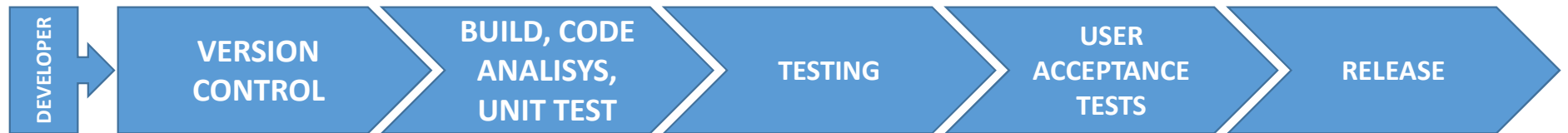
CDe vs CD
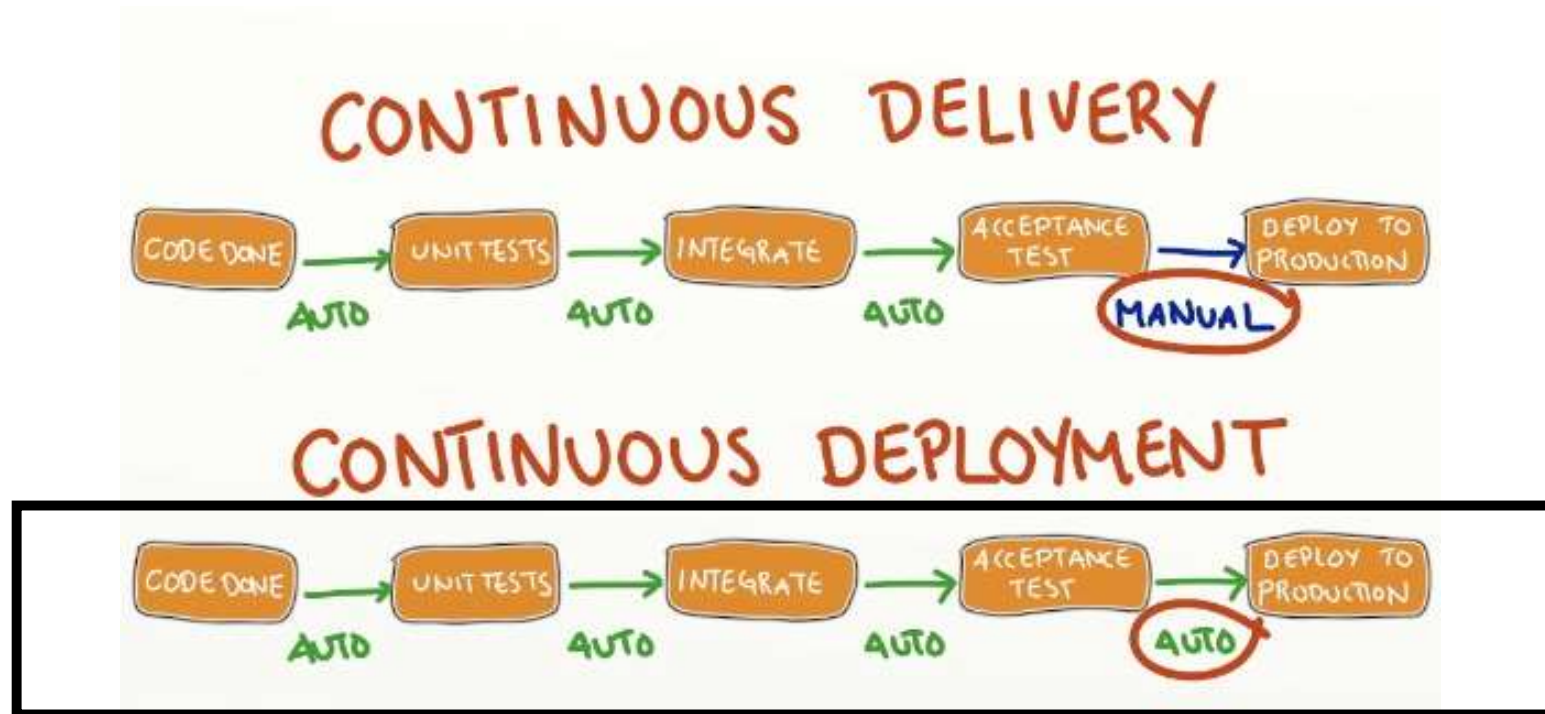
Imagine what you need to do for get a fully Continuous Deployment environment?
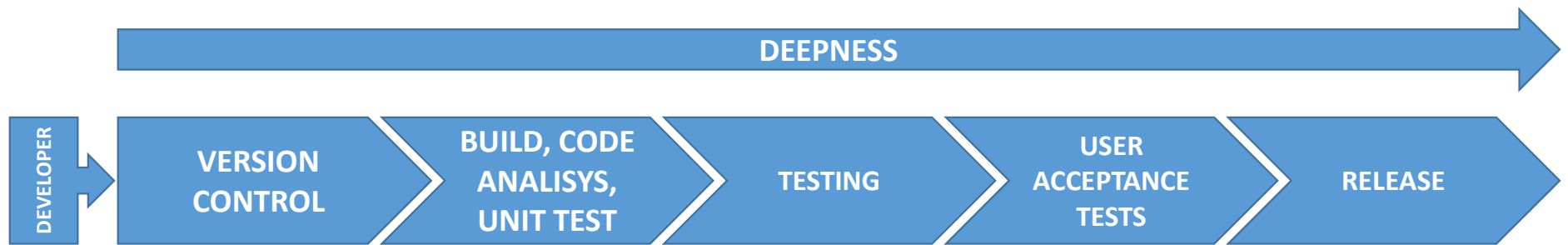
Pipeline, an approach that intercepts DevOps practices

DEVELOPER	VERSION CONTROL	BUILD, CODE ANALISYS, UNIT TEST	TESTING	USER ACCEPTANCE TESTS	RELEASE

A pipeline is an abstraction of the delivery process (from construction until user).
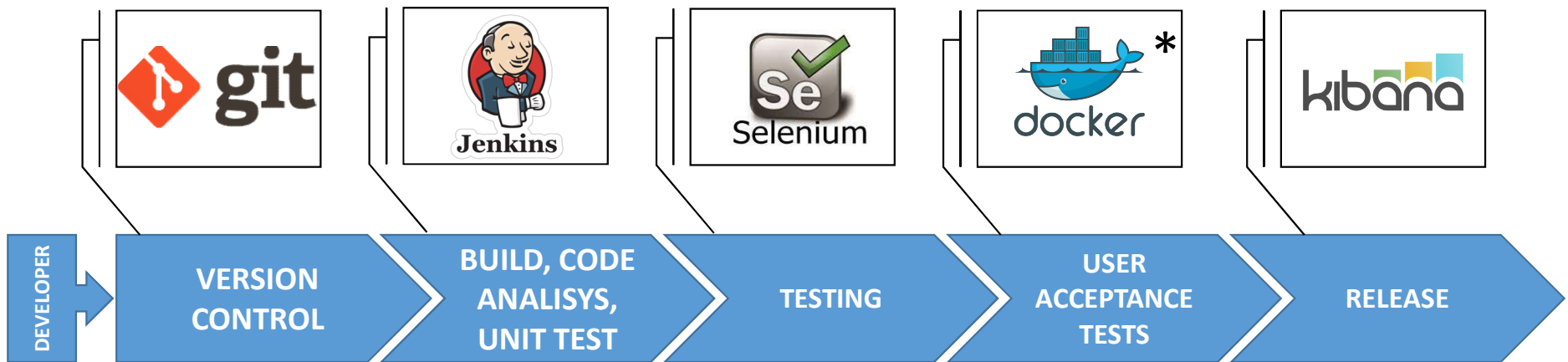**The tool set are its encarnation.**

Pipeline, how deep are you?

Have you control of whole pipeline?

Currently, this a tipical pipeline that I've seen?

We will talk about a "full" version

Pipeline, some tools

DEVELOPER → **VERSION CONTROL** (git) → **BUILD, CODE ANALISYS, UNIT TEST** (Jenkins) → **TESTING** (Selenium) → **USER ACCEPTANCE TESTS** (docker *) → **RELEASE** (kibana)

* Use containers at beginning of development. Here, I inserted ant UAT for convenience
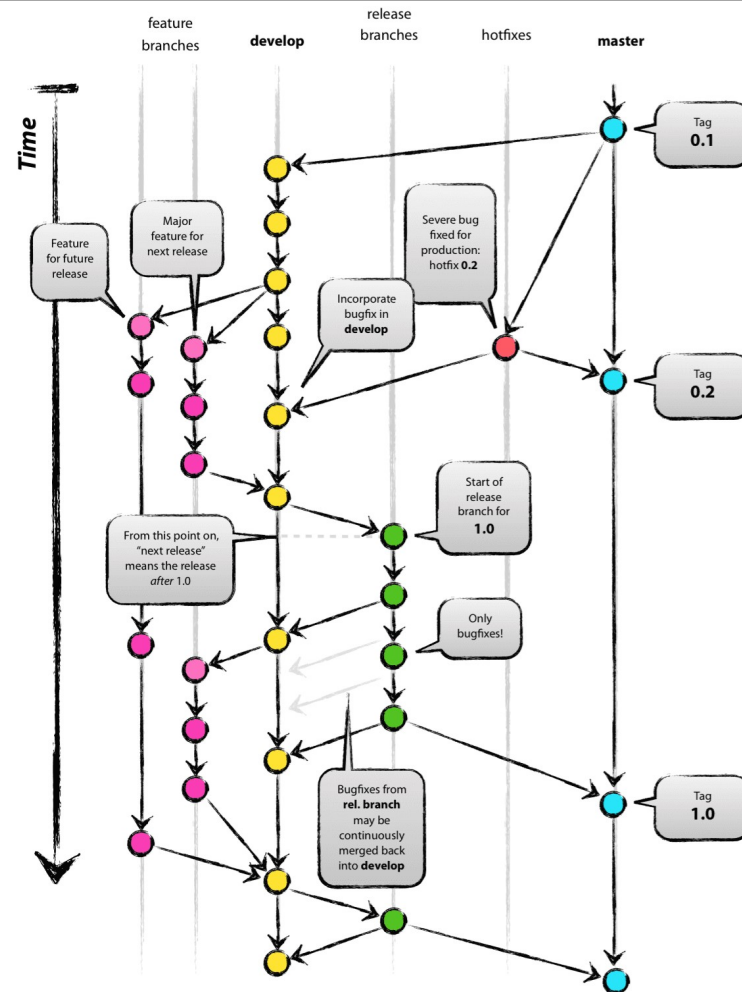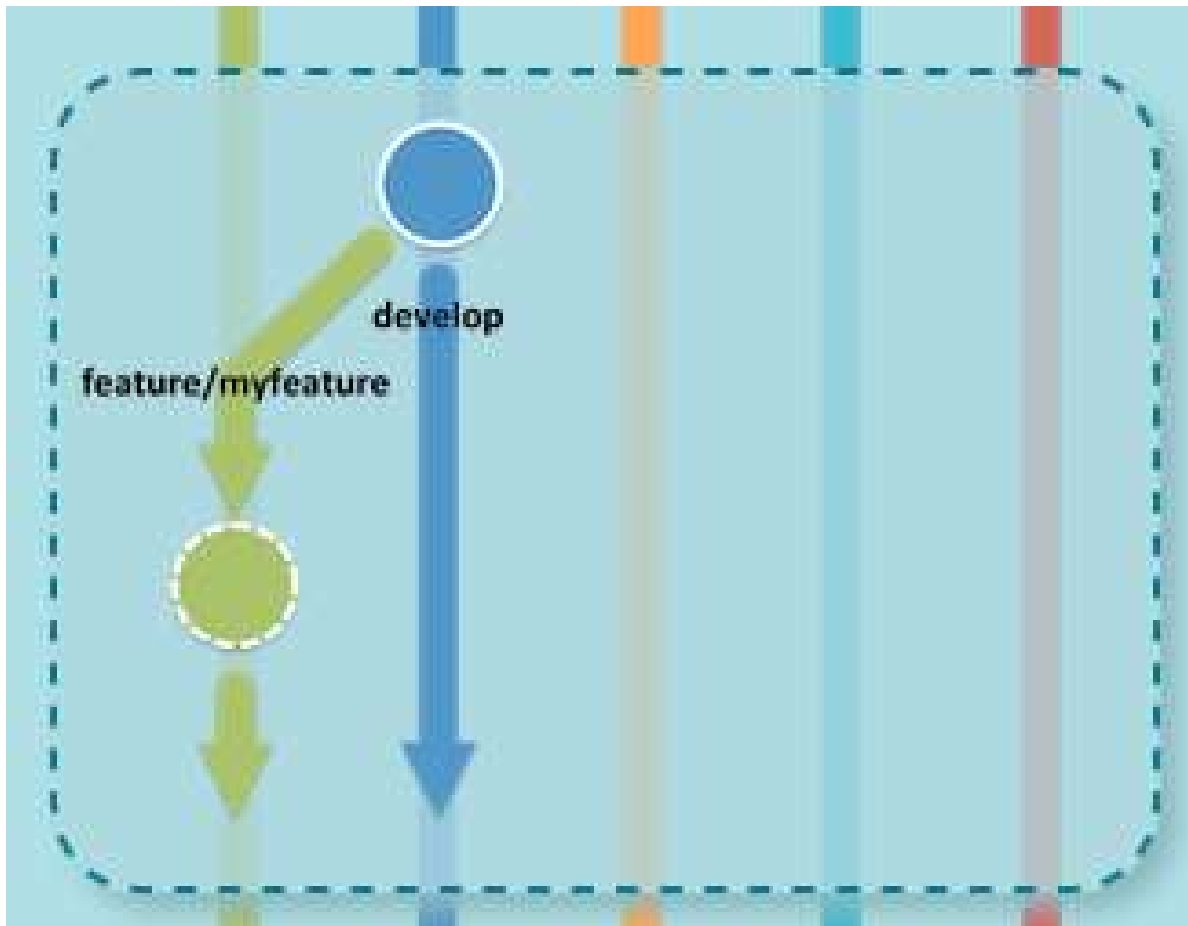
Pipeline, environments

Some important points to have CD

- I need hardware (Cloud) to provide places to run tests;
- I need know laws about data security policies
- How "deep" can I go trough environments. Ex. The customer don´t let us access his production environment.
- Will my customer apreciate the idea of deploy in production for every 11 secs? [amazona aws]
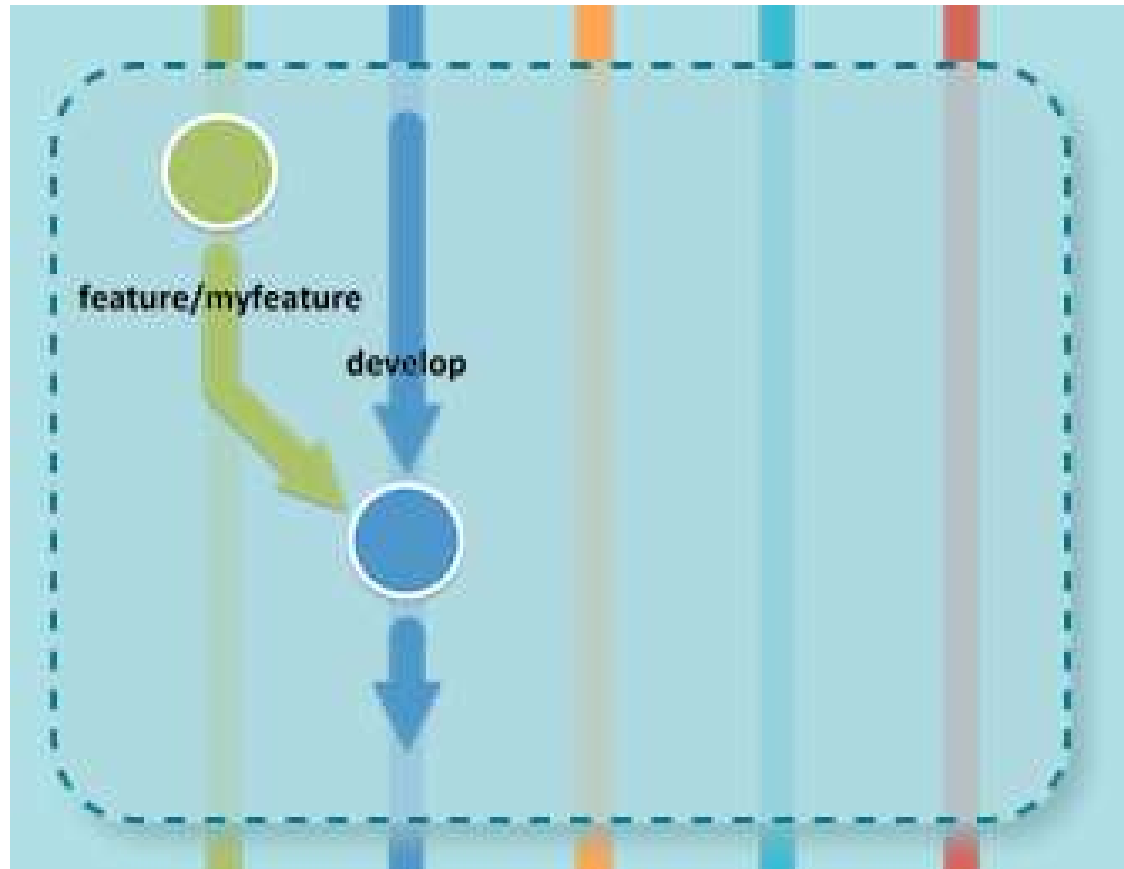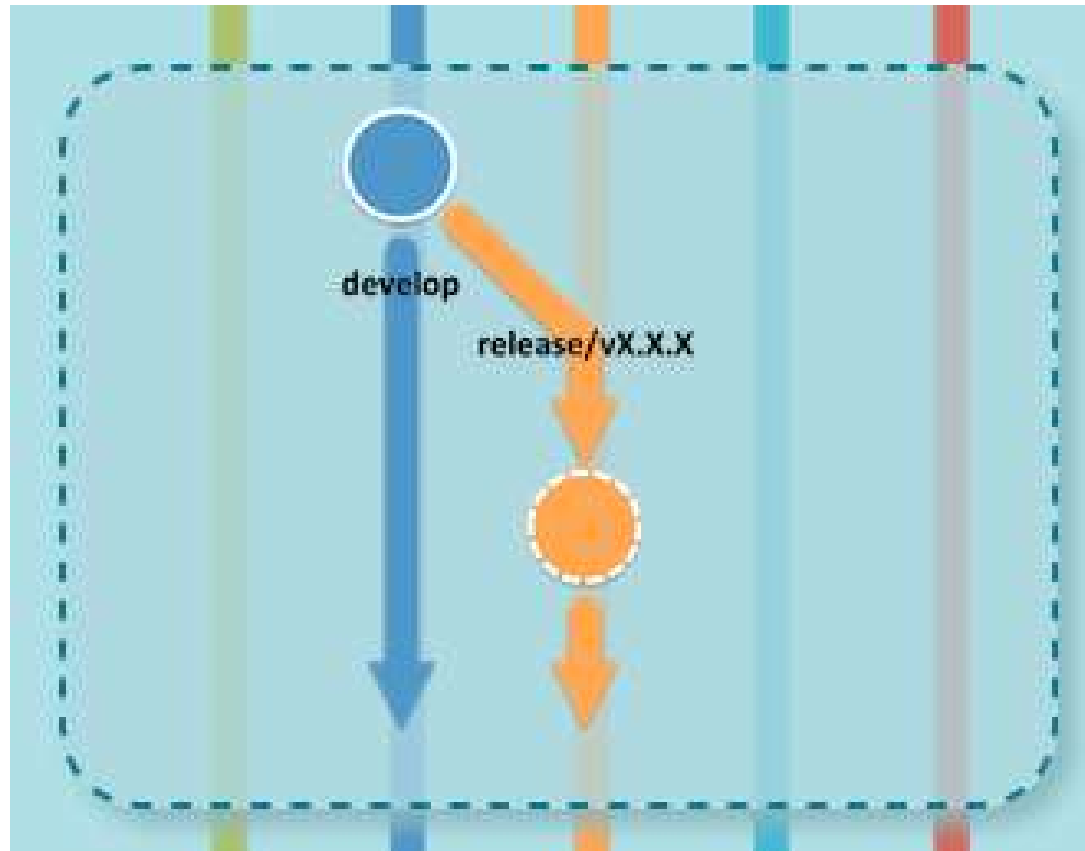
Branching

Branching

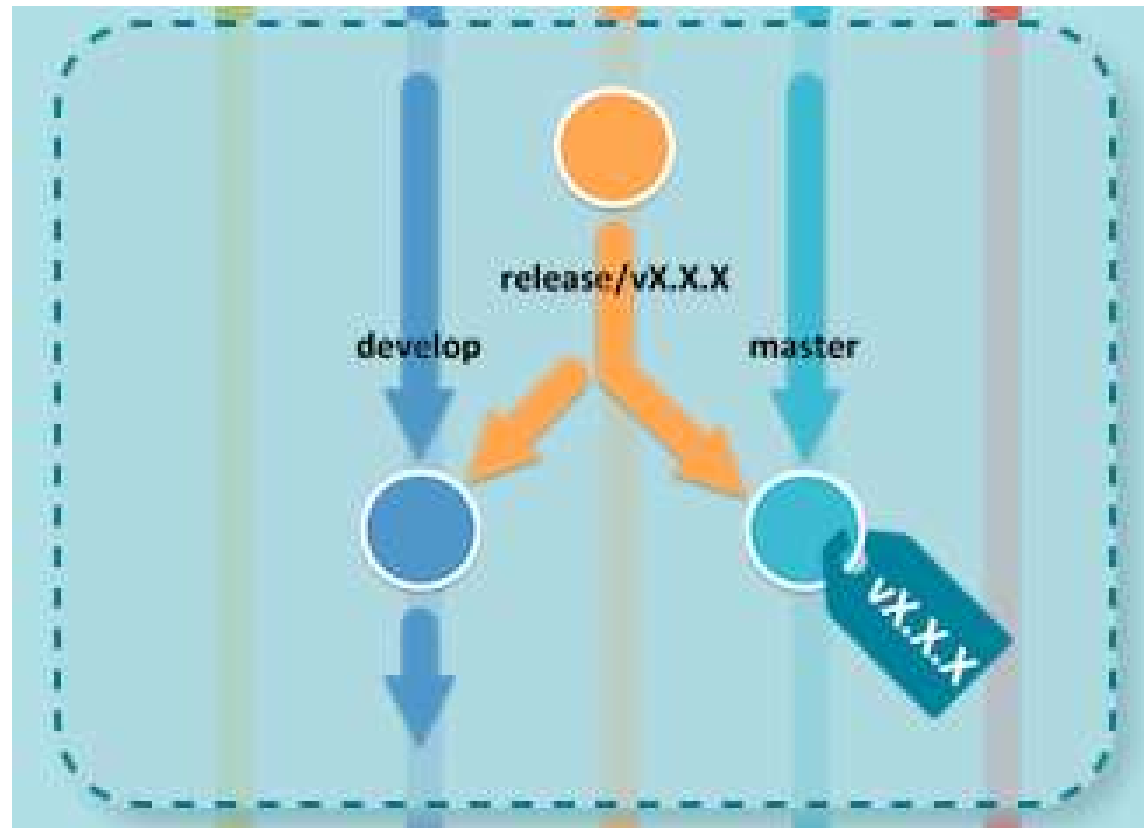From http://nvie.com/

Branching

feature/myfeature

develop

From http://nvie.com/

Branching

From http://nvie.com/

Branching

From http://nvie.com/
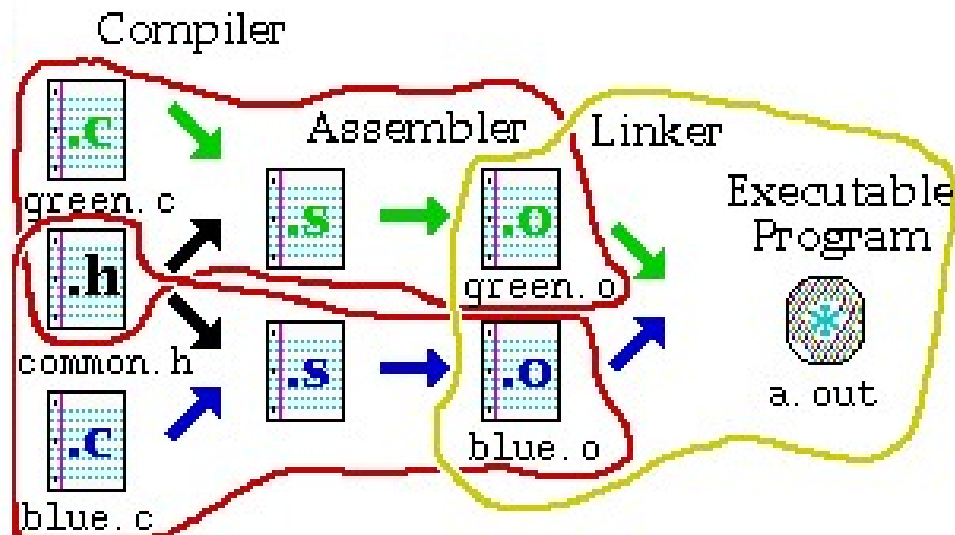
Build: building a deployable

1. Generating sources.
2. Compiling sources.
3. Compiling test sources.
4. Executing tests (unit tests, integration tests, etc).
5. Packaging (into jar, war, ejb-jar, ear, rpm).
6. Running health checks (static analyzers like Checkstyle, Findbugs, PMD, test coverage, Sonarqube).
7. encapsulating environments
8. Generating reports.

Build:Get source

git clone https://github.com/jfsc/spring-petclinic.git

Build: Compiling

Compiler

Assembler Linker

green.c → .s → .o (green.o)

common.h

blue.c → .s → .o (blue.o)

Executable Program

a.out

```
andes:~rcomp1/softdev> gcc -c green.c
andes:~rcomp1/softdev> ls -ls green.o
3 -rw-r--r--  1 13042 users 2312 Mar 13 13:40 green.o
andes:~rcomp1/softdev> file green.o
green.o: ELF 64-bit LSB relocatable, AMD x86-64, version 1 (SYSV), not stripped
andes:~rcomp1/softdev> gcc -c blue.c
andes:~rcomp1/softdev> gcc green.o blue.o
andes:~rcomp1/softdev> ls -ls a.out
8 -rwxr-xr-x  1 13042 users 7864 Mar 13 13:40 a.out
andes:~rcomp1/softdev> a.out
Result of Monte Carlo integration is 3.582862
andes:~rcomp1/softdev> gcc -o green green.o blue.o
andes:~rcomp1/softdev> file green
green: ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV), for GNU/Linux 2.4.
andes:~rcomp1/softdev> green
Result of Monte Carlo integration is 3.582862
andes:~rcomp1/softdev>
```

From https://goo.gl/MNLHZl

Build: Compile and run UnitTests

```
javac -cp .:"/Applications/IntelliJ IDEA 13 CE.app/Contents/lib/*" SetTest.java
```

```
java -cp .:"/Applications/IntelliJ IDEA 13 CE.app/Contents/lib/*" org.junit.runner.JUnitCore
SetTest
JUnit version 4.11
.
Time: 0.007
OK (1 test)
```

Build: Packaging

- (*.JAR;*.WAR.*.EAR, *.DLL)
- (*.RPM, *.EXE, *.O)
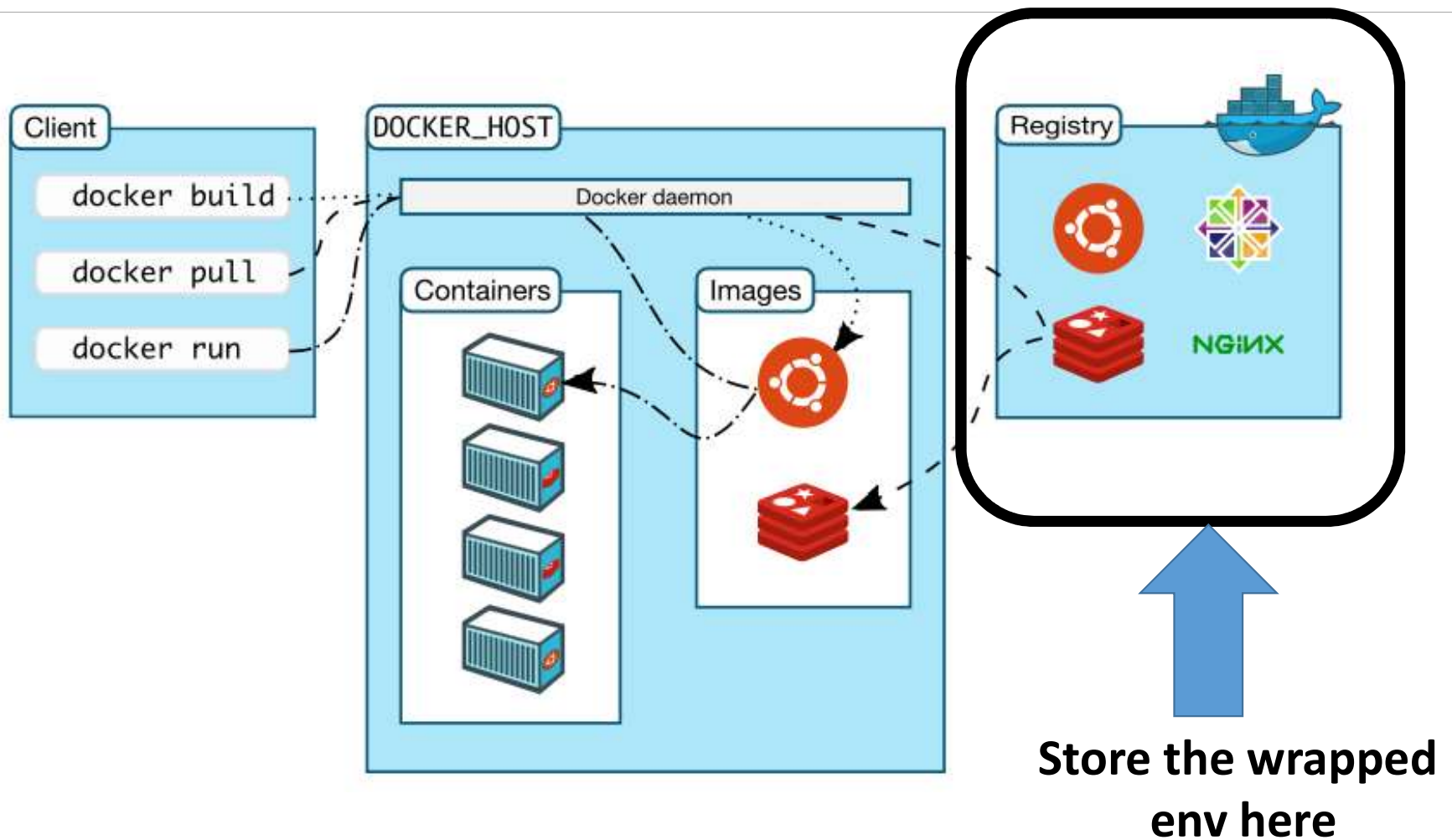
SonarQube (*)
Cobertura

Build: Store Envs

- Docker
- VMs

Build: Store Envs : Docker

Build: Store Envs : Docker

Store the wrapped env here
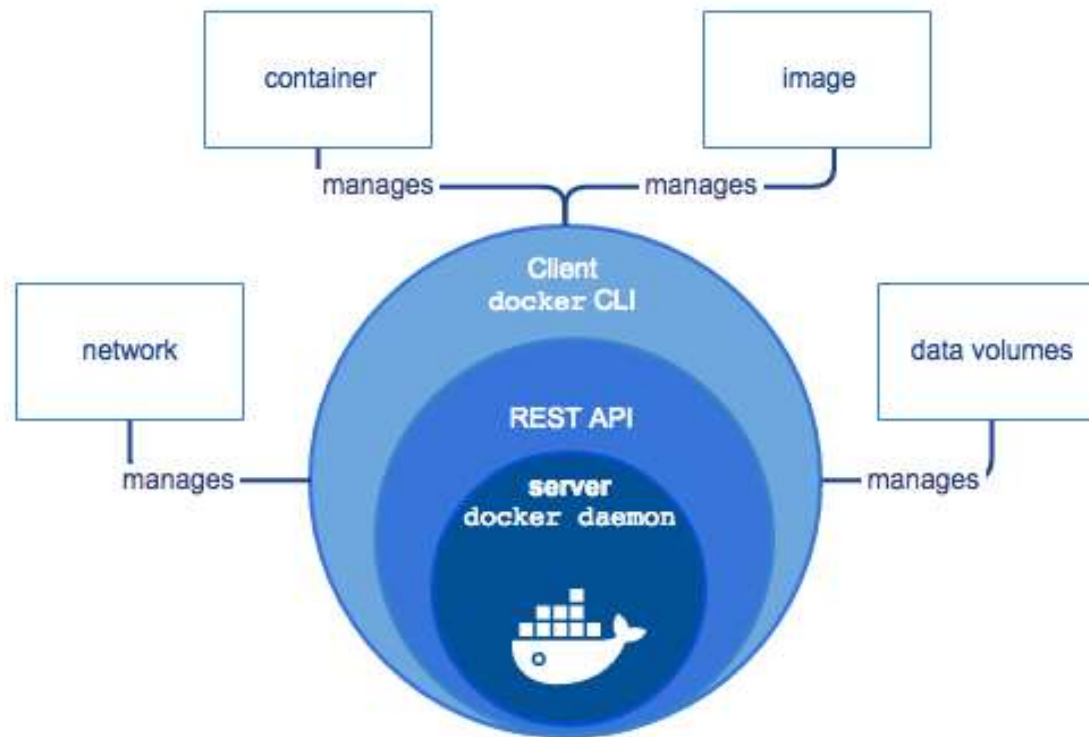
Testing

- Manual (Exploratory)
- Automatic

UAT:User Acceptance Testing

- Smoke Tests
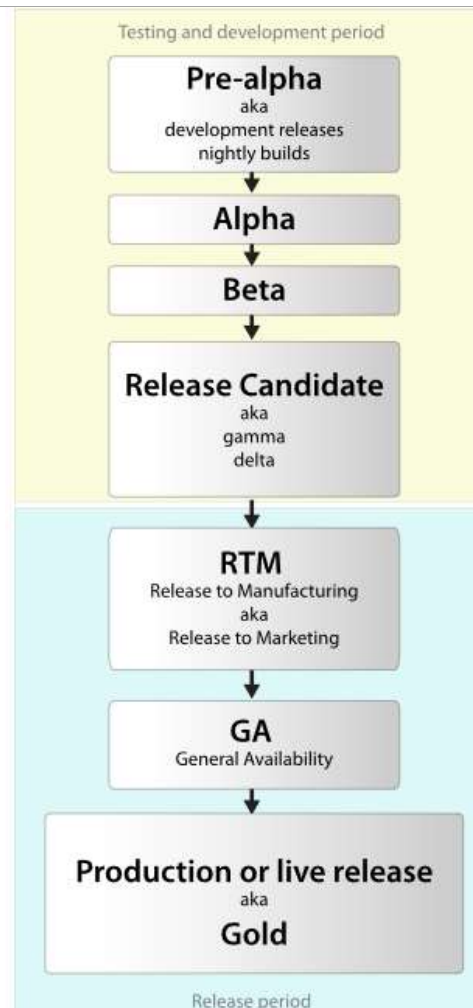- Customer Validation

PRODUCTION

- Release
- Measurement

Release Management

"Release management is the process of managing, planning, scheduling and controlling a software build through different stages and environments; including testing and deploying software releases" [wiki]

A Release lifecycle

[wiki]

1 . 3 . 5

BREAKING . FEATURE . FIX

Breaking change

New **Feature**

Fixing bugs

Change Management

- When problems happen (pls, identify before the user):

1. You need analyse the request of customer to identify if is a new functionality or a new one.
2. Move the task to the current Sprint or