CONTINUOUS INTEGRATION CONTINUOUS DELIVERY CONTINUOUS DEPLOYMENT

PBA SOFTWAREUDVIKLING/
BSC SOFTWARE DEVELOPMENT

Christian Nielsen cnls@cphbusiness.dk

Tine Marbjerg tm@cphbusiness.dk

SPRING 2019

TODAY'S TOPICS

Overview

- Learning objectives
- Continuous principles, steps and challenges
- Vagrant / VirtualBox / DigitalOcean
  - Images
- Maven / Netbeans
  - Goals / Plugins / Profiles / Bash scripts / Suites
- Github / Ruby / Travis
  - SSH Keys / Passwords / Build file / Bash scripts
- Alternatives
  - Docker / Jenkins
- Examples
- Exercises / Assignment
- Guest lecturer: Joachim Rørbøl (Efio)

LEARNING OBJECTIVES

- Differentiate between automated testing, test automation, continuous integration, delivery and deployment and DevOps principles
- Run different tests separately using test suites and build goals, plugins, profiles and properties
- Test locally and remotely using a virtual machine image
- Deploy remotely using to a virtual machine image
- Set up an automation server capable of building, testing and deploying depending on test results

CONTINUOUS PRINCIPLES

AUTOMATED TESTING

Automated testing is the act of automatically conducting execution of various test cases, based on some test scripts and by using testing frameworks

TEST AUTOMATION

Test automation refers to automating the process of managing and tracking tests by using automation tools

CONTINUOUS INTEGRATION

Continuous integration is the practice of routinely merging all developer work and continually integrating code changes together into a shared mainline

CONTINUOUS DELIVERY

Continuous delivery involves having an automated release process and the ability of easily releasing code changes at any time

CONTINUOUS DEPLOYMENT

Continuous deployment consist of having every code change automatically tested and deployed when tests are passed

DEVOPS

Set of practices that automates the processes between software development teams (Dev) and information technology operations teams (Ops), in order to build, test, and release software faster and more reliably

CONTINUOUS PRINCIPLES

RELEASE PIPELINE

A build is generated based on code changes and automated tests are used to validate the build before releasing build

Software is continuously in development and always deployment ready, while testing is done as early, as often and as much as possible

CONTINUOUS STEPS

Locally

Developing / Building / Testing / Deploying

- Execute different types of tests (Unit / Integration)
- Build application
- Test using virtual machine

Remotely

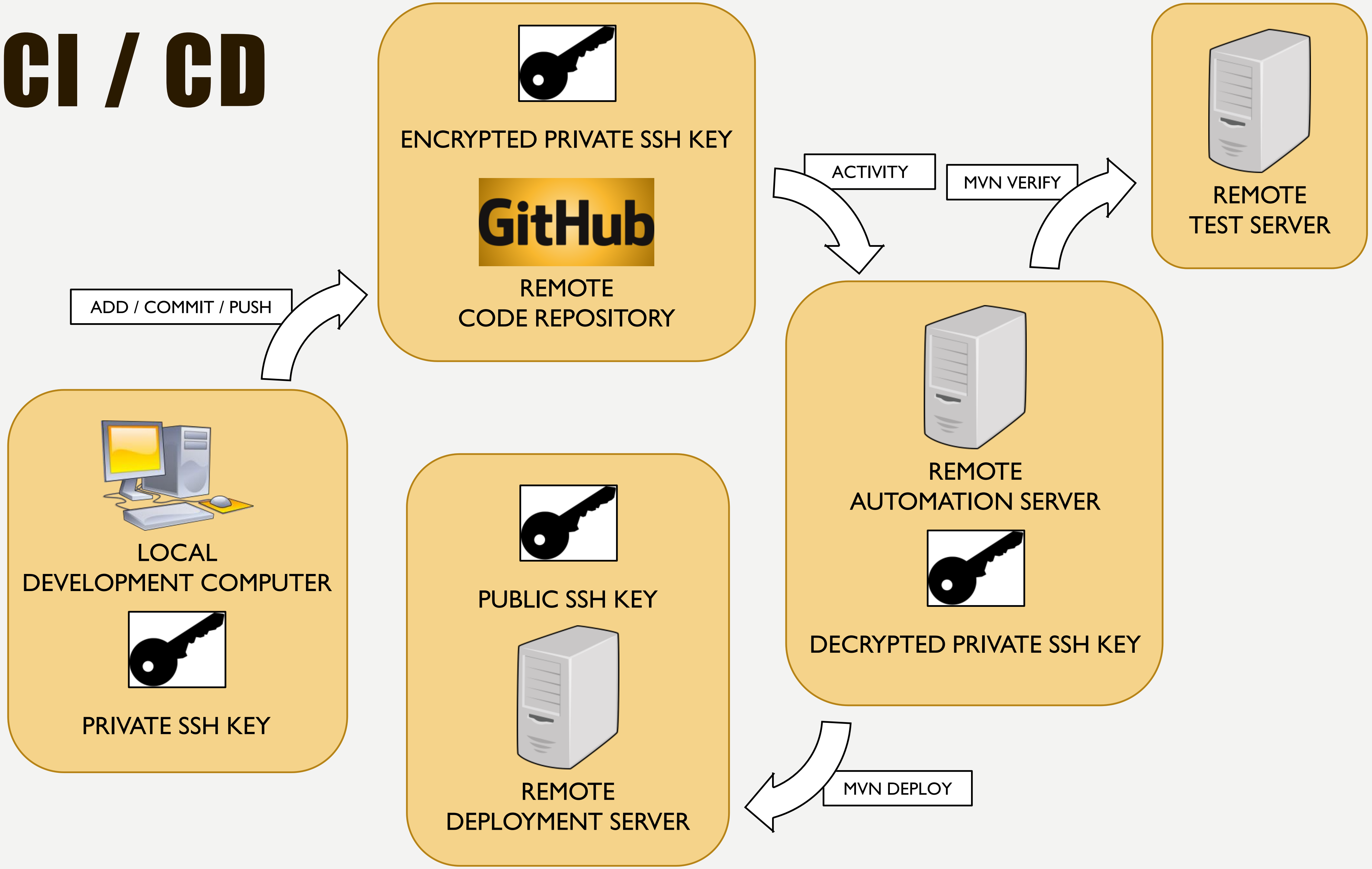
Adding / Commiting / Pushing / Triggering / Building / Testing / Deploying

- Add, commit and push code to github
- Trigger Travis
- Execute different types of tests with test suites (Unit / Integration)
- Build application
- Deploy application to test server
- Deploy application to host server depending on test results

CONTINUOUS CHALLENGES

- Running different test suites
- Merging branches depending on test results
- Updating droplet image
- Keeping server active while redeploying
- Generating reports and mails
- Checking status of build process
- Managing SSH keys and passwords between local and virtual machine, github repository, test and deployment servers

CI / CD


ENCRYPTED PRIVATE SSH KEY
GitHub
REMOTE
CODE REPOSITORY
ACTIVITY
MVN VERIFY
REMOTE
TEST SERVER
ADD / COMMIT / PUSH
REMOTE
AUTOMATION SERVER
DECRYPTED PRIVATE SSH KEY
LOCAL
DEVELOPMENT COMPUTER
PRIVATE SSH KEY
PUBLIC SSH KEY
REMOTE
DEPLOYMENT SERVER
MVN DEPLOY

VIRTUAL MACHINE

Emulation of a computer system / Sandboxed operating system

VirtualBox

Free open source hosted hypervisor supporting the creation, execution and management of virtual machines

Vagrant

Open source product for building and maintaining portable virtual software development

Image

Copy of the entire state of a computer system stored in a file

VIRTUAL MACHINE

Image requirements…

- Ubuntu 18.10
- OpenJDK 11.0.1
- Mysql 8.12
- Tomcat 9.0.16

Files…

Vagrantfile	Operating system
Install.sh	Installation script

Users…

UBUNTU	Username: root / Password: vagrant1234
	Username: vagrantuser / Password: vagrant1234
MYSQL	Username: root / Password: root
	Username: mysqluser / Password: mysql1234
TOMCAT	Username: tomcatuser / Password: tomcat1234

IP…

10.19.17.12

VAGRANT

Open source product for building and maintaining portable virtual software development

vagrant --version	Check installed vagrant version
vagrant up	Start up virtual machine
vagrant ssh	SSH into virtual machine
exit	Exit virtual machine
vagrant halt	Stop virtual machine
vagrant destroy	Destroy virtual machine
vagrant box remove	Delete virtual machine

MAVEN

Build tool for managing Java based projects, their dependencies, their plugins and their build process

Build life cycle is made up of phases that can be controlled with goals, profiles and properties

Phases…

clean / validate / compile / test / package / verify / install / deploy

Goals…

```
mvn test
mvn verify
```

Profiles…

```
mvn verify -P verifyLocal
```

Properties…

```
mvn test -D test=tests.unit.UT_Basic
mvn test -D test=tests.suites.Suite_UT
```

MAVEN

Plugins…

SureFire	Executes unit tests

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.22.1</version>
    <configuration>
        <includes>
            <include>%regex[^.*Unit.*]</include>
        </includes>
    </configuration>
</plugin>
```

MAVEN

Plugins…

FailSafe Executes unit and integration tests

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-failsafe-plugin</artifactId>
  <version>2.22.1</version>
  <configuration>
    <includes>
      <include>%regex[^.*Integration.*]</include>
    </includes>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>integration-test</goal>
        <goal>verify</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

MAVEN

Plugins…

ExecMaven Executes bash scripts

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <version>1.6.0</version>
  <executions>
     <execution>
       <id>execute-bash-script</id>
       <goals>
         <goal>exec</goal>
       </goals>
       <phase>pre-integration-test</phase>
       <configuration>
         <executable>cmd</executable>
         <arguments>
           <argument>/C</argument>
           <argument>${project.basedir}/scripts/maven.sh</argument>
         </arguments>
       </configuration>
     </execution>
  </executions>
</plugin>
```

MAVEN

Profiles…

Used to set up different build profiles that can be executed with ”mvn verify -P verifyLocal”

```
<profiles>
    <profile>
        <id>verifyLocal</id>
        <build>
            <plugins>
            </plugins>
        </build>
    </profile>
</profiles>
```

MAVEN

Properties…

Used to provide different build properties that can be executed with ”mvn test -D test=tests.unit.UT_Basic” or ”mvn test -D test=tests.suites.Suite_UT”

Both test classes and test suites can be executed with properties…

Naming convention

Default naming convention for JUnit unit tests classes are ”Test*”, “*Test” and “*TestCase” for unit tests

Default naming convention for JUnit integration tests classes are “IT*” and “*IT” for integration tests

Do not name tests classes and test suites with default prefixes or postfixes to disable execution of tests when building

Name test classes and test suites “Integration*”, “Unit*” or “I_T_E_S_T_*”, “U_T_E_S_T_*” or other conventions instead

DEPLOYMENT

Tomcat7 plugin…

Deployment to Tomcat server can be done by using Tomcat7 plugin and with ”mvn clean tomcat7:deploy” instead of manually using Tomcat manager

```
<plugin>
  <groupId>org.apache.tomcat.maven</groupId>
  <artifactId>tomcat7-maven-plugin</artifactId>
  <version>2.2</version>
  <configuration>
    <url>http://10.19.17.12:8080/manager/text</url>
    <server>TomcatServer</server>
    <path>/ProjectCICD</path>
    <username>root</username>
    <password>admin</password>
    <update>true</update>
  </configuration>
</plugin>
```

DEPLOYMENT

Bash script…

A more flexible approach would be to use ExecMaven plugin and execute a bash script on virtual machine

```
#!/bin/bash
echo "Maven.sh script executed..."
curl -v -u tomcatuser:tomcat1234 -T ./target/TestProjectCICD-1.0.war
'http://10.19.17.12:8080/manager/text/deploy?path=/TestProjectCICD&update=true'
```

DIGITAL OCEAN

Add SSH public key

Droplet <-> Custom image <-> Virtual machine

Set up test server and deployment server based on virtual machine image to use for CI / CD

SSH

ssh root@SERVER

ssh -i ~/.ssh/SSHprivatekeyfile root@SERVER

TRAVIS

Online server for building, testing and deploying projects from GitHub repositories, triggered by activity and based on a .travis.yml file in the GitHub repository

TRAVIS COMMAND LINE CLIENT

Used to encrypt content / files and interacting with Travis online

RUBY

Ruby is a dynamic, interpreted, reflective, object-oriented, general-purpose programming language

Required to install the Travis command line client

TRAVIS

.travis.yml

A build job life cycle consists of several different phases

1.	Install apt addons	OPTIONAL
2.	Install cache components	OPTIONAL
3.	before_install	
4.	install	
5.	before_script	
6.	script	
7.	before_cache	OPTIONAL
8.	after_success / after_failure	
9.	before_deploy	OPTIONAL
10.	deploy	OPTIONAL
11.	after_deploy	OPTIONAL
12.	after_script	

TRAVIS

Linux Commands can be used during job build life cycle

Echo…	**(Printing messages)**
- echo "Before install..."	
Environment variables…	**(Using environment variables)**
- echo $SOMEVAR	
Bash script…	**(Executing bash script)**
- bash scripts/install.sh	

TRAVIS

Linux Commands can be used during job build life cycle

OpenSSL… (Decrypting keys)

```
- openssl aes-256-cbc -K $encrypted_1fc90f464345_key -iv $encrypted_1fc90f464345_iv -in traviskey_openssh.enc -out ./traviskey_openssh -d
```

SCP… (Copying files)

```
- scp -o StrictHostKeyChecking=no -i ./traviskey_openssh ./target/TestProjectCICD-1.0.war root@$DOIP:~/testfolder/TestProjectCICD.war
```

SSH… (Making SSH connection / Executing bash script)

```
- ssh -o StrictHostKeyChecking=no -i ./traviskey_openssh root@$DOIP "bash -s" < ./scripts/digitalocean.sh "$TOMCAT_USER $TOMCAT_PASSWORD hello"
```

ALTERNATIVES

Docker…

Open source computer program that performs operating-system-level virtualization by using containers

https://www.docker.com/

Jenkins…

Open source automation server written in Java

https://jenkins.io/

EXERCISES

- **VIRTUAL MACHINE IMAGE**
  - Use image locally for test server
  - Use image remotely for test and deployment server
- **MAVEN**
  - Goals / Profiles / TestSuites
- **TRAVIS**
  - Build file / Bash scripts

GUEST LECTURER

- **JOACHIM RØRBØL**
  - Efio.dk
  - CI / CD / DevOps / Release pipeline

RESOURCES

https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

https://www.taniarascia.com/what-are-vagrant-and-virtualbox-and-how-do-i-use-them/

https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

http://wiki.netbeans.org/MavenBestPractices

https://travis-ci.com/

https://travis-ci.org/getting_started

https://docs.travis-ci.com/

https://docs.travis-ci.com/user/encrypting-files/

https://github.com/travis-ci/travis.rb

https://github.com/travis-ci/travis.rb#command-line-client

https://www.ruby-lang.org/en/

https://rubyinstaller.org/

https://www.docker.com/

https://jenkins.io/