Software Engineering and Architecture
Continuous Delivery in Practice

Olivier Liechti
HEIG-VD
olivier.liechti@heig-vd.ch

MSE | MASTER OF SCIENCE IN ENGINEERING

Phase 1: Get the basics (manual process)

Tasks: working with Git repositories

- **Create your fork of this Github repo:**
  - https://github.com/wasadigi/SEA-SampleWebApp
- **Clone your fork on your machine and configure the upstream repo.**
  - https://help.github.com/articles/fork-a-repo

MSE | MASTER OF SCIENCE IN ENGINEERING

Tasks: installing the first **build** & **run** tools

- **We will build our software with maven**
- **We will run our software in the Glassfish 4.0 application server**

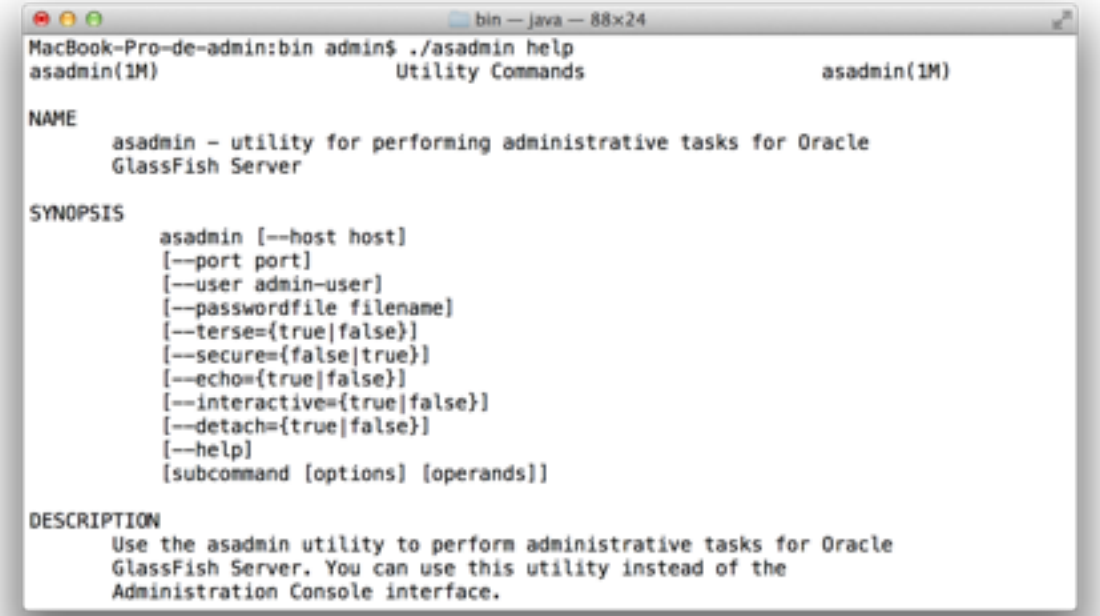
Tasks: building the application manually, with mvn

- We have a `pom.xml` file, which describes how our application should be built.
- If we move to the directory containing this file and issue the following command, we trigger the build process manually:
  - `mvn clean install`
- We can look in the target directory and find the result of our build process. The `.war` file is a software package that we can deploy in the application server.

MSE | MASTER OF SCIENCE IN ENGINEERING

Tasks: Glassfish, domains and asadmin

- Glassfish is an **application server**. It provides a runtime environment in which we can deploy our application.
- Glassfish offers the notion of **domain**. Think of a domain as an instance of the app server, which binds to specific TCP ports. Think of a domain as an independent container in which you can deploy different apps, or versions of apps.
- You can manage Glassfish via a **web console** (default port: 4848), but also with a very powerful command: `asadmin`.



```
MacBook-Pro-de-admin:bin admin$ ./asadmin help
asadmin(1M)                      Utility Commands                      asadmin(1M)

NAME
       asadmin - utility for performing administrative tasks for Oracle
       GlassFish Server

SYNOPSIS
           asadmin [--host host]
           [--port port]
           [--user admin-user]
           [--passwordfile filename]
           [--terse={true|false}]
           [--secure={false|true}]
           [--echo={true|false}]
           [--interactive={true|false}]
           [--detach={true|false}]
           [--help]
           [subcommand [options] [operands]]

DESCRIPTION
       Use the asadmin utility to perform administrative tasks for Oracle
       GlassFish Server. You can use this utility instead of the
       Administration Console interface.
```

MASTER OF SCIENCE
IN ENGINEERING

Tasks: playing with domains and apps, manually

- **Let’s create a special domain for our QA team**. We can use the following commands:
  - `asadmin list-domains`
  - `asadmin create-domain --user admin --nopassword --portbase 9000 domainQA`
  - `asadmin start-domain domainQA`
  - `asadmin stop-domain domainQA`
  - `asadmin delete-domain domainQA`
- **Once we have a Glassfish domain, we can deploy our `.war` file**
  - Via the admin web console
  - With `asadmin deploy --port 9048 --force myapp.war`

Checkpoint

- **At this point, we should be able:**
  - To pull changes from the Github repo
  - Manually invoke maven in a terminal window and get a .war file
  - Create a “fresh” domain (deleting the previous one if it exists)
  - Deploy the .war file with asadmin

Phase 2: Automate the build & deployment process

Tasks: install the “director”: jenkins

- **We need a tool to drive the continuous delivery pipeline. We will use jenkins:** `http://jenkins-ci.org/`
- Jenkins is distributed as a .war file, which we could deploy in Glassfish. However, we can also run it in its own embedded server with this command:
  - `java -jar jenkins.war`
- Once jenkins is running, we can open its **web console** (default port: 8080)

Tasks: installing optional plug-ins

- Jenkins has a **very large community** of users and a **lot of plug-ins** are available to do lots of interesting things.
- **In this lab, we will use the following plug-ins:**
  - Build Pipeline Plugin
  - Email Extension Plugin
  - Clone Workspace SCM Plug-in
- **Install them by going into:**
  - Jenkins > Manage Jenkins > Manage Plugins > Available

Tasks: create a job to manage the QA domain

- Each time we release a version of our app to the QA team, we want to make sure that we have a clean state.
- For this reason, we will create a brand new Glassfish domain, using the asadmin command.
- In this simple scenario, we don't need files from our Git repo. We will write the commands directly in Jenkins (not the best choice!).
- Let's create a **"free-style" Jenkins job**:
  - Jenkins > New Item > Build a free-style software project
- We have a single **build step**, which consists of executing the following shell script:

```
/Applications/NetBeans/glassfish-4.0/glassfish/bin/asadmin stop-domain domainQA || true
/Applications/NetBeans/glassfish-4.0/glassfish/bin/asadmin delete-domain domainQA || true
/Applications/NetBeans/glassfish-4.0/glassfish/bin/asadmin create-domain --user admin --nopassword --portbase 9000 domainQA
BUILD_ID=dontKillMe /Applications/NetBeans/glassfish-4.0/glassfish/bin/asadmin start-domain domainQA
```

Tasks: create a job to build and deploy the app

- An important step of our delivery pipeline is obviously to build our application. We will ask Jenkins to work with maven in order to do that.
- Let us create a second Jenkins job. This time, it will be a “maven2/3 project”.
- **Let’s try something quick and dirty**. We know where our pom.xml file is on our local machine, so we can enter its location in the Build > Root POM field.
- We can also add a **Post Step** and execute the following shell script:

```
/Applications/NetBeans/glassfish-4.0/glassfish/bin/asadmin deploy --port 9048 --force ./DemoApp/target/DemoApp-1.0-SNAPSHOT.war
```

point to the location of your .war file

- We can also **configure the e-mail notifications**. To do that, we also need to go into Jenkins > Manage Jenkins > E-mail Notification

Tasks: putting things together...

- **This is how our first pipeline should work:**
    - Step 1: get updates from the Git repo
    - Step 2: setup the QA domain
    - Step 3: build and the deploy the app
    - Step 4: notify the users and give them the URL of the app

Checkpoint

- **At this point, we should be able:**
  - Do a “one-click” deployment on your QA domain
  - See a visual representation of your pipeline in Jenkins
  - Investigate problems by looking at the Jenkins console
  - Receive a personalized e-mail with a link to the new version of the app

Phase 3: Move to the Cloud (build & run)

Demo: the Cloudbees PaaS platform

CloudBees