SDM FINAL PROJECT

4th semester

Melissa Ma

Adi Tanase

Damian Socha

Sebastian Socha

ERHVERVS AKADEMI SYDVEST

1 Table of Contents

2 What are we trying to accomplish

We had to deliver a software project with automated Continuous Integration and Continuous Delivery setup trough Linux distribution. Considering that our main goal was to implement the knowledge we achieved on this semester we decided to do for a simple java program with high unit test coverage.

The product is developed in Eclipse/NetBeans on a java platform with the starting structure of Maven. The name of the project is called HelloMaven2 and that may be a funny name but is reflecting our interest for maven tool and how any times we tried to get it right. Meaning that we had to develop a previous version which it didn't made it to production. But the problems we faced along this entire process we will elaborate them later on in this report.

Overall we are trying to develop code in a Continuous Delivery environment. That should lead to a solid product and having us the team collaborate to each other.

3 Options we decide to approach for the project

3.1 Maven

Initially we developed our HelloMaven1 just as a java project which is the first version of our app, fair enough, everything worked, but at the point when the CI occurred we realized that maven is a better approach in order to avoid extra set-up.

Maven is a tool for managing life-cycle and build the project. Starting from its creation through the compilation, unit testing, integration, creation of documentation and deployment finished program. Reflecting upon is definition and consistency we decided to summarize some of its advantages and disadvantages so can all of has a starting point in the same direction in this entire process.

3.1.1 Advantages:

Managing relationships and dependencies intermediate.

Simple configuration file pom.xml.

Easy selection of tasks from the command line.

Easy versioning and tagging code.

A large number of plug-ins (plugins) for both simple and complex tasks.

Integration with all popular IDE (Eclipse/NetBeans).

Ability to manage multiple modules of the project at a time.

Full support for unit testing, integration and stress-performance.

3.1.2 Disadvantages:

Incomplete or not consistent with the reality of documentation.

The lack of consistency in actions requiring overwriting and adding configuration.

Troublesome running when not connected to the Internet.

3.1.3 Problems:

When we agreed upon that Maven is the tool we want to work with we used the feature provided by the IDE Eclipse to convert our first version to a maven project.

That lead to a locally functionally maven project with the pom.xml desired but that it didn't help us that much on the Jenkins set-up. Here we will face errors because the plugin maven will look in the standard folder to build and build tests. So here we spent some time to work around this issue.

Standard folder target it by the maven - -- src/main/java --and ours --- src----. After trying to somehow make it to look into our desired folder with not success we moved upon developing a new app. Here we created the maven project straight from the start no conversion. All worked just fine afterwards.

3.2 Vagrant

We worked with vagrant to create and automate virtual environment cooperating with oracle virtual box. Sticking with the old fashion way the windows command prompt we did a-- vagrant init-- and here you go a predefine Vagrantfile.

Vegrantfile is used to setup configuration of the machine required for a project, it contains information on how to configure and provision our virtual machines. It allow us to use scripts and plugins.

Ok so we had this file now it is time to do our own setup. We used the documentation from the slides provided by the teacher and we create multi-environment machines. Actually two boxes, one for deploying our app build to apache server and another one for hosting Jenkins.

3.2.1 Architecture

- Vagrant file
- comp.sh
- web.sh
- general.sh

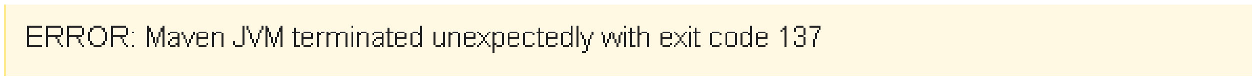
In order to have a readable vagrant file we shared the jobs among the shell script files (.sh) which are the equivalent of the windows bat files. Provisioning from them from the vagrant file, it will execute them in the desired box.

3.2.2 Why Vagrant?

We are aware that Docker is also an option for creating containers as well and it might be faster due the less overhead. But we find vagrant quit appealing and a lot of documentation is provided. Being more comfortable with the vagrant technologies we end up with the result we expect it from it.

3.2.3 Problems:

When a build occurred from the Jenkins, we had this time of error which provide some work to do in order to mitigate.



```
ERROR: Maven JVM terminated unexpectedly with exit code 137
```

After doing some documentation we got instructed to increase the value of memory at MAVEN_OPTS. We did that for some time with different values but not a really success has been provided.

What we did, was just to do a drastic change of our boxes. We migrate from the Ubuntu/trusty64 to /precise64 which it ran just smooth after we did vagrant up for the new targets.

3.3 Repository

3.3.1 What opportunities does Git offer to us?

With Git we can create repositories, to allow team members to edit multiple files simultaneously, stratify design, manage versions, create branching design, and combine the project with others. Generating detailed statistics project life including the percentage contribution of each of the authors, distribution of working hours, etc.

3.4 Jenkins

A great open-source CI tool which definitely plays a big role in the Continuous Delivery. This tool will engage us developers to detect and solve errors within our code with a great range of time.

After our Jenkins is installed and is running on the localhost:8081(apache running on :8080), a setup is need in order to run our desired projects.

First we did the manage plugins which is a must for our desired running project.

Plugins we installed:

- GitHub plugin
- Jenkins SSH plugin
- Cobertura plugin
- Credentials plugin

Because of the time limit range we didn’t manage to find a final solution for having an automatic setup for it. But in the user guide is explain the steps we proceed in order to get it up running manually.

3.5 Continuous Integration

3.6 Definition

A beneficial practice for us which consists of frequent, regular turning (integration) current changes in the code to the main repository. In practice, each member of the development team should at least once a day to put their work done in the repository. It is also an indispensable element to ensure the correctness of the code is compiled after the integration. We summarize some of the main advantages and disadvantages.

3.6.1 Advantages

Reducing the cost and amount of work needed to connect the work done by different people
Early detection of bugs
Early detection of failed unit tests
Automate deployment

3.6.2 Disadvantage

Time needed to setup CI

4 Continuous delivery

4.1 Definition

The way that we see Continuous delivery within our app is likely to demonstrate a series of practices to ensure that code can be quickly, reiterate and safely deployed to production. This is automated by delivering every change to a code product-like environment and ensuring business applications and services function as expected through high coverage automated testing.

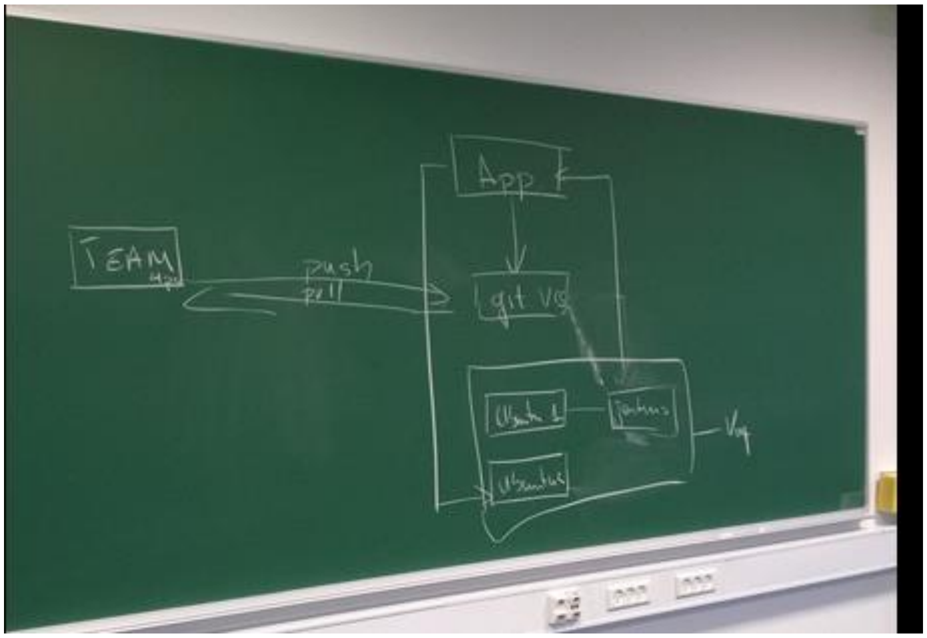
4.2 Benefits

As we did with the Maven on reflecting about the advantages we did the same approach here. First we did documentation to get an understanding of what is this concept and how it can expand our abilities to deliver better code.

The way we sit is described below with the bullet points.

- Speed: we get to deliver faster of the most important improvements and fixes.
- Quality: by having a defined schedule upon release we end up with fewer bugs and better final product
- Focus: us the team we got to focus more on the code and a more solid code instead of sharing our try each time to implement similar configuration for our systems
- Transparency: this point is very important because is leading to less stress among developers. They can focus on a task at the time. No stress: happy developers.

A simple synthetize of how we first imagine visually the continuous delivery will look like.



4.3 Problems:

When we tried to deploy our build to the apache server on the second box we faced this error where ssh connection it could not be establish. Because of the time limitation this part we didn't successfully managed to create.

The way we tried to communicate was with the help of the Jenkins ssh plugin. We tried to establish connection after we setup the appropriate host and the login credentials but as is mentioned earlier no success from it.

If we had more time, the way we will attack this is it will be by sharing a common key between this two environments and add this at the Jenkins setup.

5 Continuous test driven development

We are using Junit Annotations framework for test. The annotations made some of the programming test easier and more explicit. Because they do a lot of thing in the background.
We approach the TDD method in a skeptical way but later on we could the see the power of it. It improve

or understanding of what we were doing and what it lead to a better anticipation of the result.

But having the tests that will not be enough for Jenkins to see our tested code and issue the expected result. What we need it to do was to tell the pom.xml about the dependencies we had issue earlier by using Junit.

What we need it to do was just to add the necessary dependency and it helped us to succeed with the CI.

```
<dependencies>
      <dependency>
              <groupId>junit</groupId>
              <artifactId>junit</artifactId>
              <version>4.11</version>
      </dependency>
</dependencies>
```

6 Conclusion

Project “HelloMaven2” had to use Continuous Integration and Continuous Delivery and also to do build up automated testing. Which we almost got it all done, except that we got stuck a bit at the ssh security. But we are aware of how the continuous delivery process is defined and we have some ideas on how we will mitigate this. Some idea it was but came a bit too late to export the build to a Git repository.

It improved a lot our communication within the team and a better management product we want to develop.

Overall we are satisfied with the result because now we have solid understanding and a starting point which means less documentation jump into action.

7 Appendix

7.1 Installation guide

7.2 Prerequisite

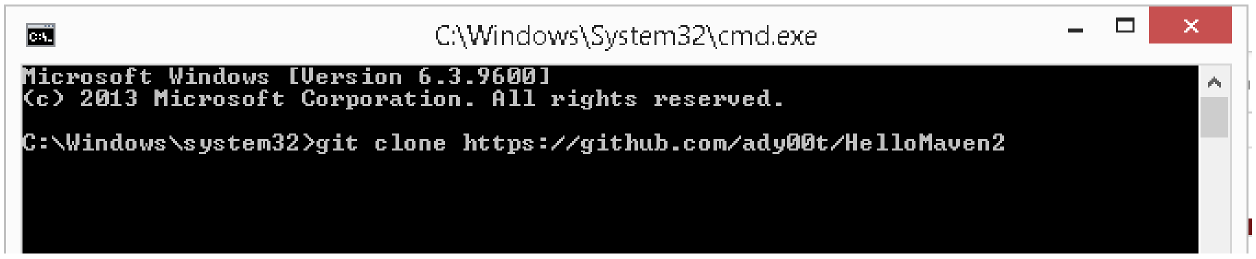
- Vagrant
- Maven
- GitHub

7.3 Steps

7.3.1 Step one

First of all it needs to find a desired target where you want to store the “HelloMaven2” project. After that open the windows cmd or git bash.

Issue the following command:

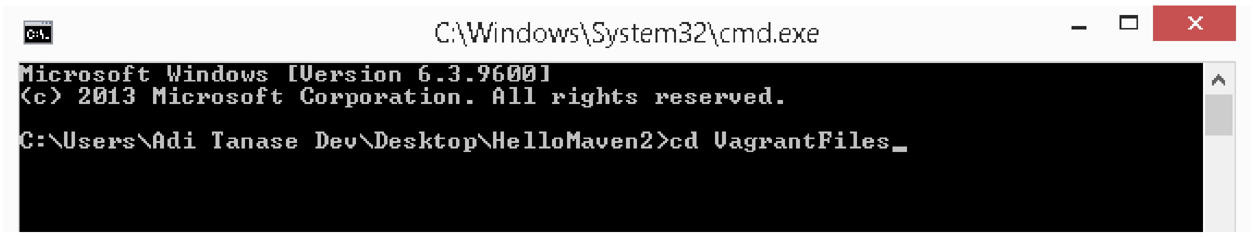


C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>git clone https://github.com/ady00t/HelloMaven2
```

7.3.2 Step two

Navigate inside the VagrantFiles directory of your project.

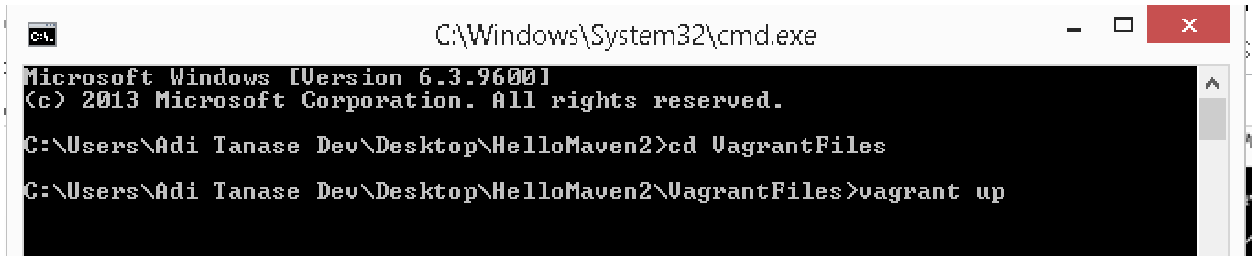


C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Adi Tanase Dev\Desktop\HelloMaven2>cd VagrantFiles_
```

7.3.3 Step three

Issue the following command.

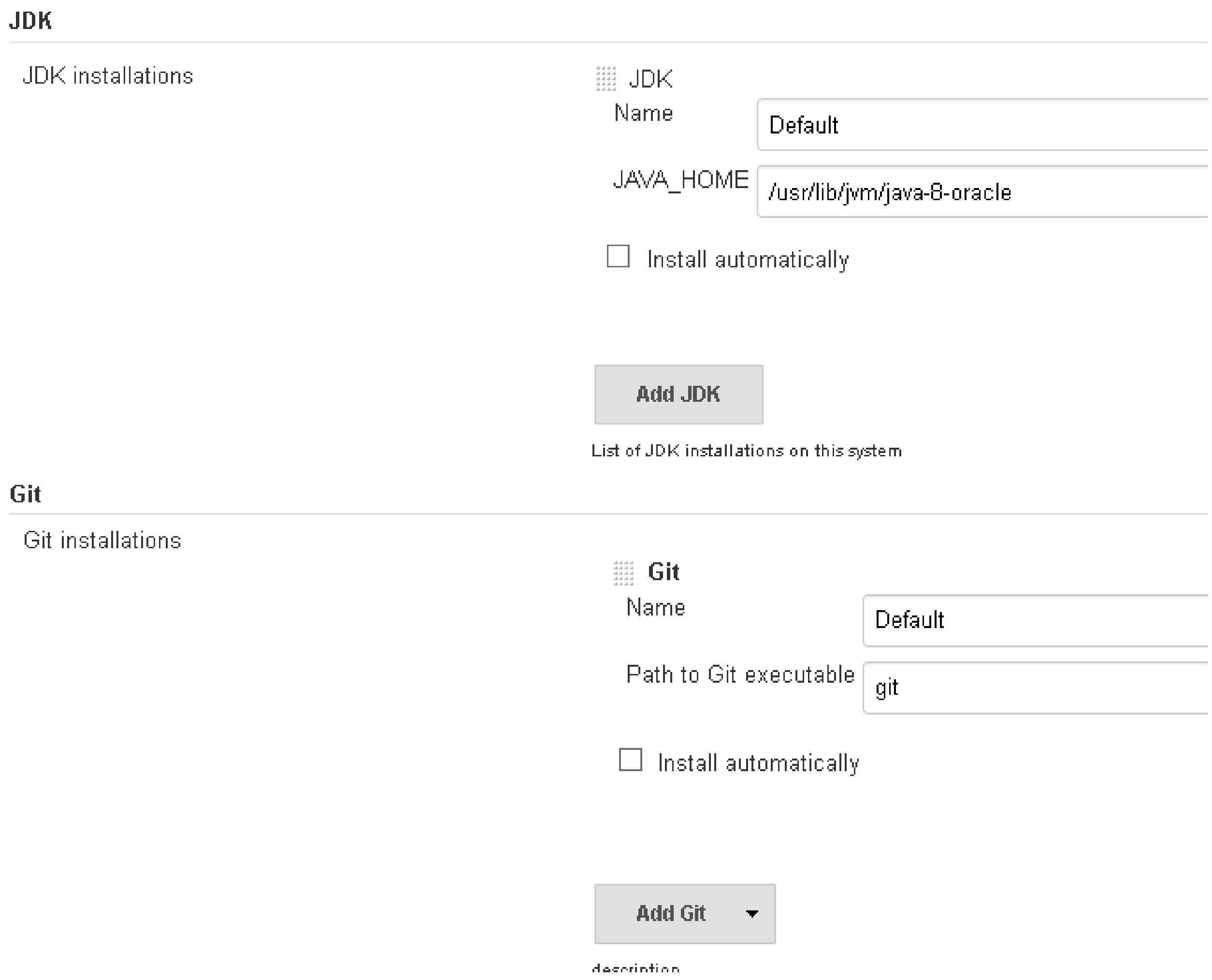


```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Adi Tanase Dev\Desktop\HelloMaven2>cd VagrantFiles

C:\Users\Adi Tanase Dev\Desktop\HelloMaven2\VagrantFiles>vagrant up
```

7.3.4 Step four

Run Jenkins at the address –localhost:8081. Get the necessary plugins described at the Jenkins chapter in this report.

After that go into Configure System and do the following setup.



JDK

JDK installations

JDK

Name: Default

JAVA_HOME: /usr/lib/jvm/java-8-oracle

☐ Install automatically

List of JDK installations on this system

Git

Git installations

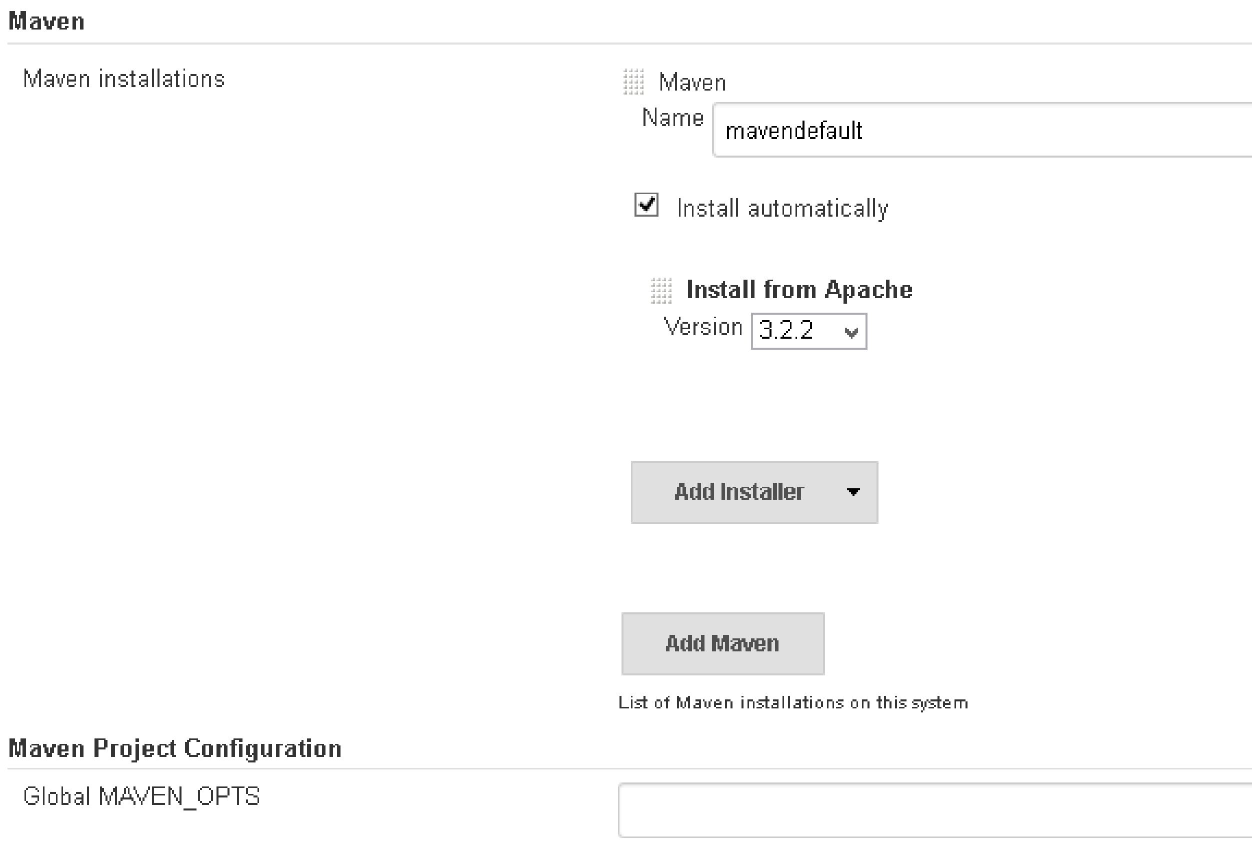
Git

Name: Default

Path to Git executable: git

☐ Install automatically

description


Maven

Maven installations

Maven

Name: mavendefault

☑ Install automatically

Install from Apache

Version: 3.2.2

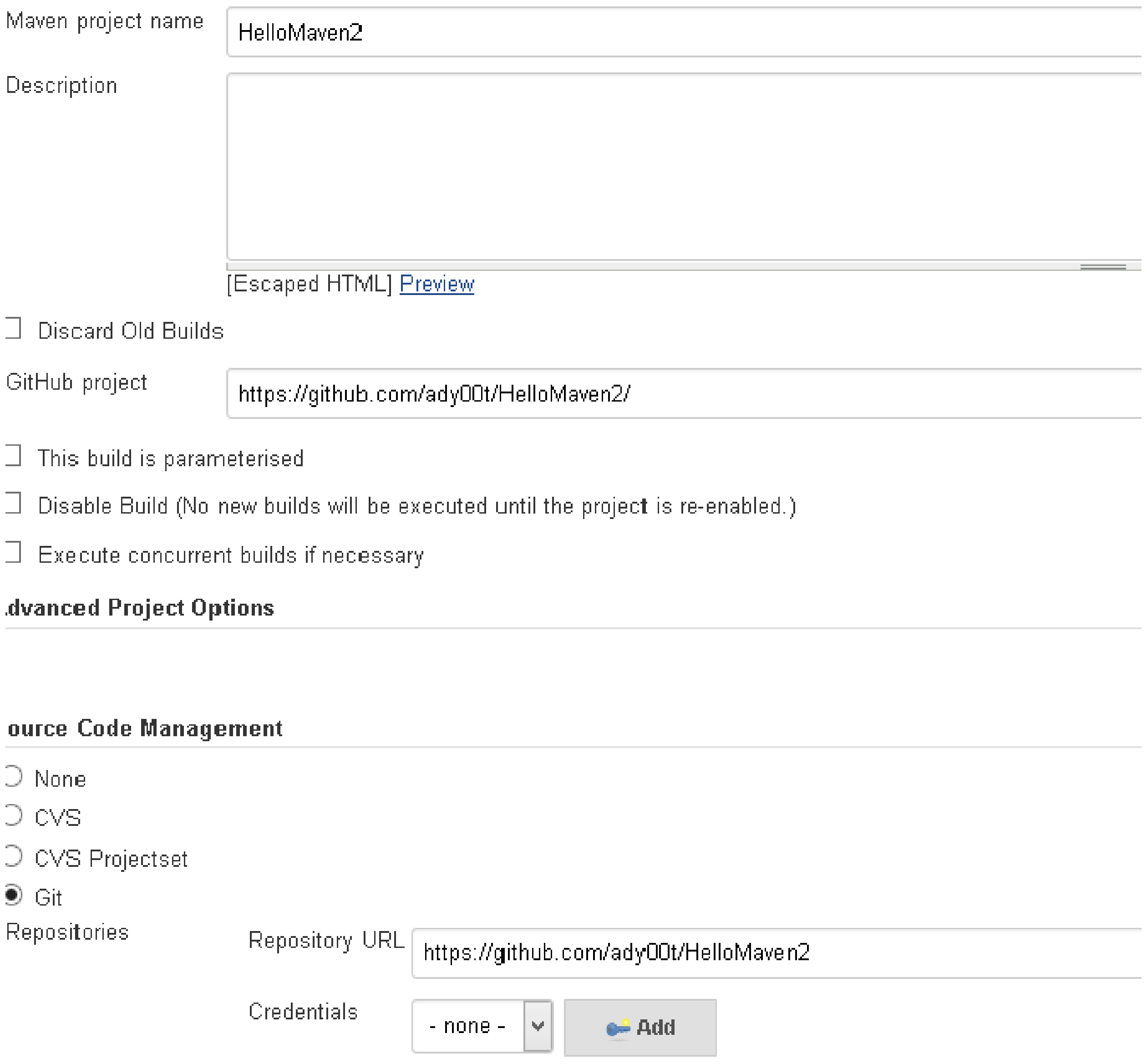
List of Maven installations on this system

Maven Project Configuration

Global MAVEN_OPTS

7.3.5 Step fifth

Add your project to you CI tool Jenkins. Do the following input after you choose a new job to create.



Maven project name: HelloMaven2

Description:

[Escaped HTML] Preview

☐ Discard Old Builds

GitHub project: https://github.com/ady00t/HelloMaven2/

☐ This build is parameterised

☐ Disable Build (No new builds will be executed until the project is re-enabled.)

☐ Execute concurrent builds if necessary

.dvanced Project Options

ource Code Management

○ None

○ CVS

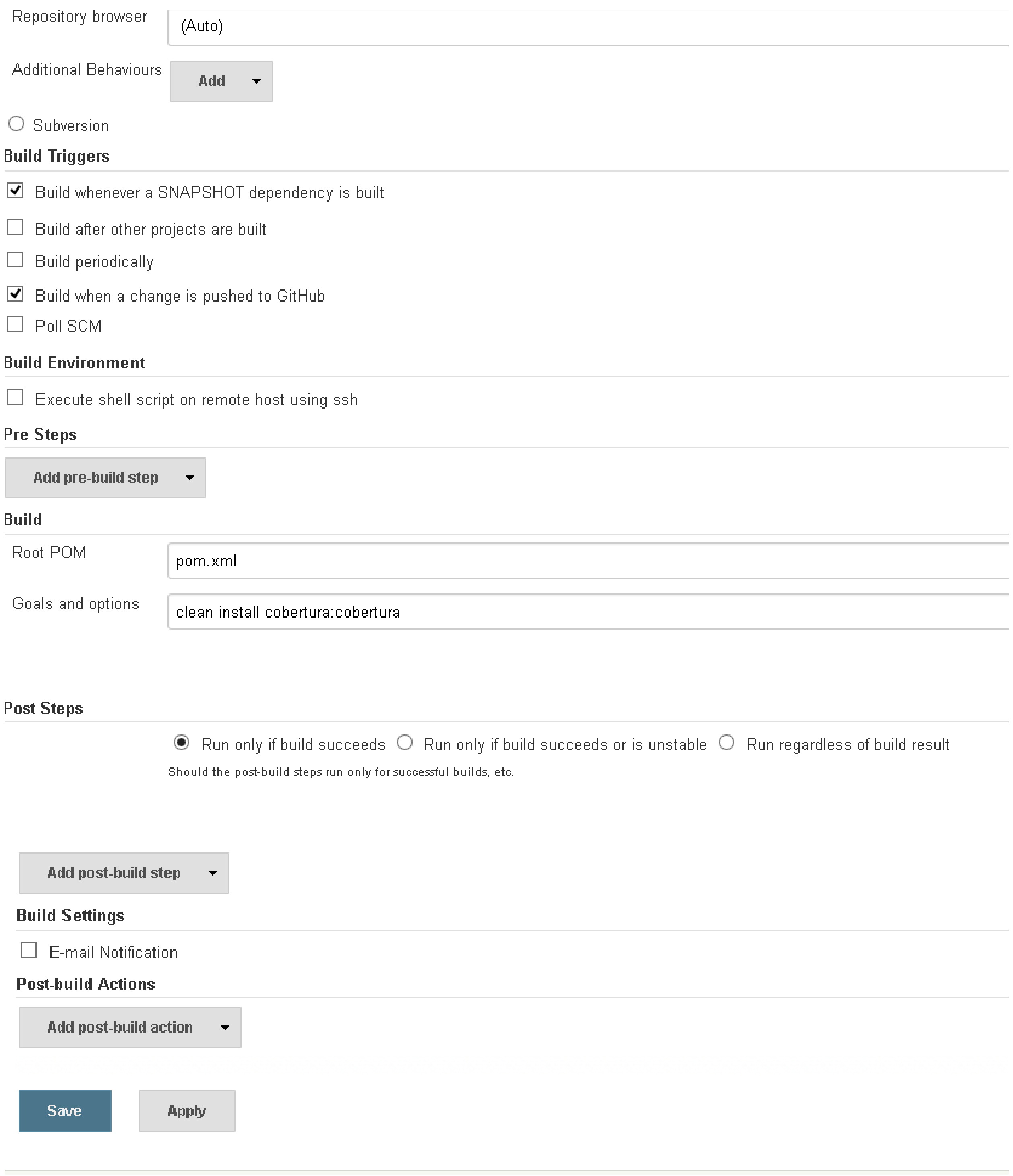
○ CVS Projectset

● Git

Repositories

Repository URL: https://github.com/ady00t/HelloMaven2

Credentials: - none - Add



Repository browser: (Auto)

Additional Behaviours:

○ Subversion

Build Triggers

- [✓] Build whenever a SNAPSHOT dependency is built
- [ ] Build after other projects are built
- [ ] Build periodically
- [✓] Build when a change is pushed to GitHub
- [ ] Poll SCM

Build Environment

- [ ] Execute shell script on remote host using ssh

Pre Steps

Build

Root POM: pom.xml

Goals and options: clean install cobertura:cobertura

Post Steps

● Run only if build succeeds ○ Run only if build succeeds or is unstable ○ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Build Settings

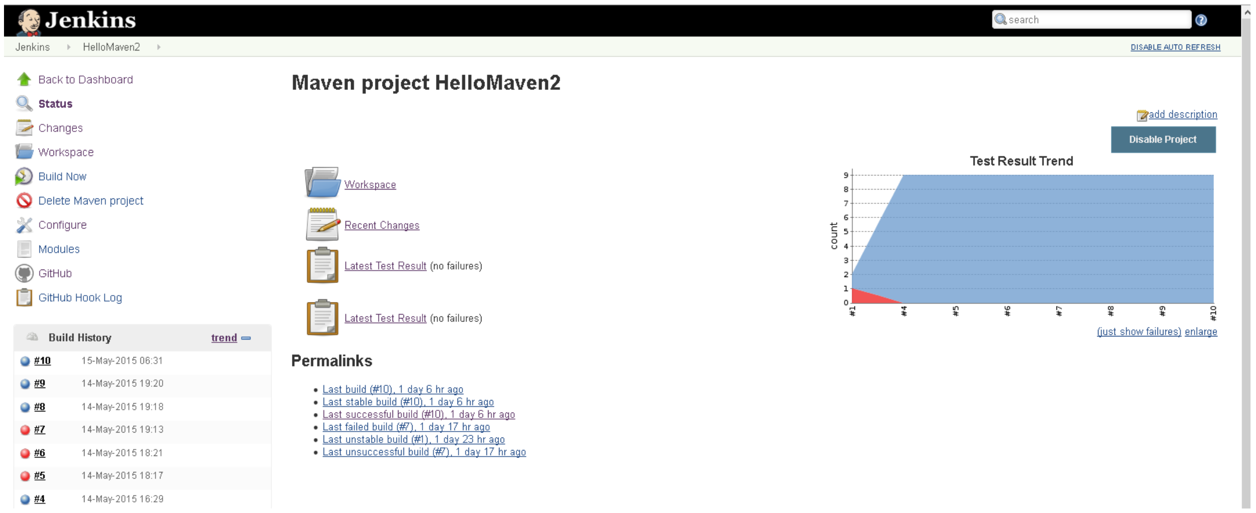
- [ ] E-mail Notification

Post-build Actions

And than Save.

7.3.6 Final Step-run the item

Here is sample of the initial product under CI . Observe the results:



Jenkins

search

Jenkins › HelloMaven2 ›

DISABLE AUTO REFRESH

Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Maven project
Configure
Modules
GitHub
GitHub Hook Log

Build History	trend
🔵 #10	15-May-2015 06:31
🔵 #9	14-May-2015 19:20
🔵 #8	14-May-2015 19:18
🔴 #7	14-May-2015 19:13
🔴 #6	14-May-2015 18:21
🔴 #5	14-May-2015 18:17
🔵 #4	14-May-2015 16:29

Maven project HelloMaven2

add description

Disable Project

Workspace

Recent Changes

Latest Test Result (no failures)

Latest Test Result (no failures)

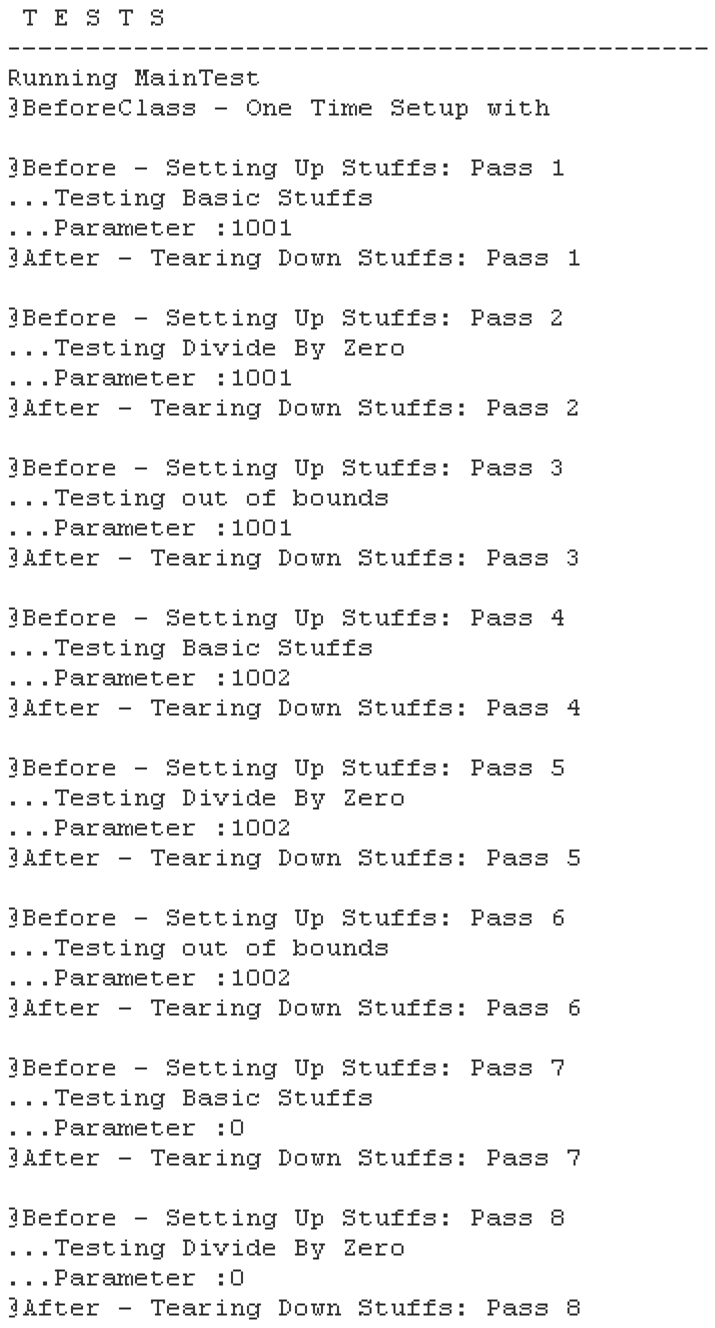
Test Result Trend

(just show failures) enlarge

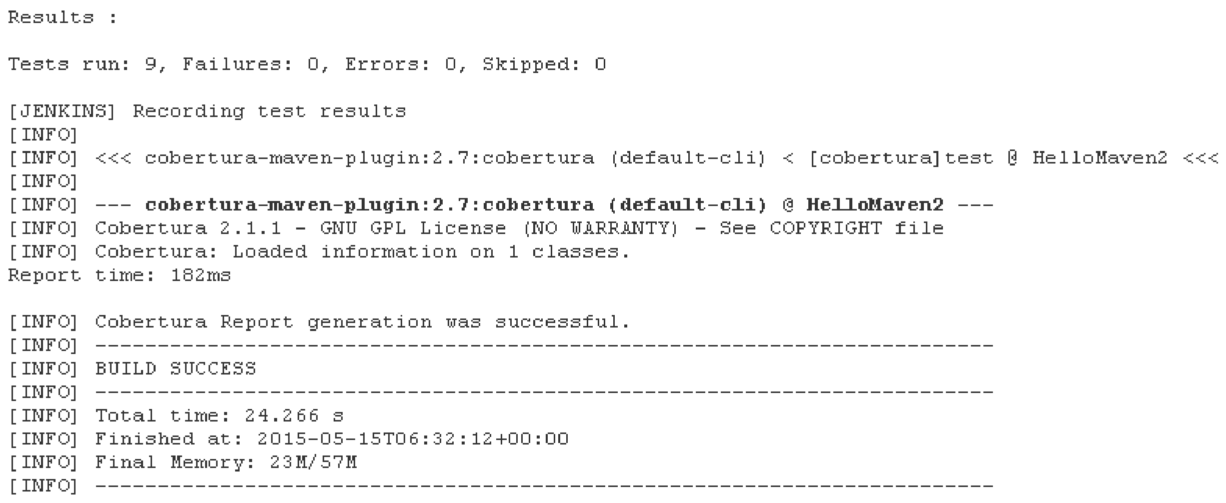
Permalinks

- Last build (#10), 1 day 6 hr ago
- Last stable build (#10), 1 day 6 hr ago
- Last successful build (#10), 1 day 6 hr ago
- Last failed build (#7), 1 day 17 hr ago
- Last unstable build (#1), 1 day 23 hr ago
- Last unsuccessful build (#7), 1 day 17 hr ago

Console output



```
 T E S T S
-------------------------------------------------------
Running MainTest
@BeforeClass - One Time Setup with

@Before - Setting Up Stuffs: Pass 1
...Testing Basic Stuffs
...Parameter :1001
@After - Tearing Down Stuffs: Pass 1

@Before - Setting Up Stuffs: Pass 2
...Testing Divide By Zero
...Parameter :1001
@After - Tearing Down Stuffs: Pass 2

@Before - Setting Up Stuffs: Pass 3
...Testing out of bounds
...Parameter :1001
@After - Tearing Down Stuffs: Pass 3

@Before - Setting Up Stuffs: Pass 4
...Testing Basic Stuffs
...Parameter :1002
@After - Tearing Down Stuffs: Pass 4

@Before - Setting Up Stuffs: Pass 5
...Testing Divide By Zero
...Parameter :1002
@After - Tearing Down Stuffs: Pass 5

@Before - Setting Up Stuffs: Pass 6
...Testing out of bounds
...Parameter :1002
@After - Tearing Down Stuffs: Pass 6

@Before - Setting Up Stuffs: Pass 7
...Testing Basic Stuffs
...Parameter :0
@After - Tearing Down Stuffs: Pass 7

@Before - Setting Up Stuffs: Pass 8
...Testing Divide By Zero
...Parameter :0
@After - Tearing Down Stuffs: Pass 8
```



```
Results :

Tests run: 9, Failures: 0, Errors: 0, Skipped: 0

[JENKINS] Recording test results
[INFO]
[INFO] <<< cobertura-maven-plugin:2.7:cobertura (default-cli) < [cobertura]test @ HelloMaven2 <<<
[INFO]
[INFO] --- cobertura-maven-plugin:2.7:cobertura (default-cli) @ HelloMaven2 ---
[INFO] Cobertura 2.1.1 - GNU GPL License (NO WARRANTY) - See COPYRIGHT file
[INFO] Cobertura: Loaded information on 1 classes.
Report time: 182ms

[INFO] Cobertura Report generation was successful.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 24.266 s
[INFO] Finished at: 2015-05-15T06:32:12+00:00
[INFO] Final Memory: 23M/57M
[INFO] ------------------------------------------------------------------------
```