Technische Universität Berlin

System Quality Engineering Project

MODEL-DRIVEN DEVOPS CONFIGURATION

July 30, 2018

Submitted by:

Rene Wiegmann Rollet

Sebastian Ortmanns

Preethi Ramachandran

Srividhya Sankar

Ahmad Hassan

Supervised by:

Christian Hein

Julia Magdalena Martini

Index

1.Introduction

Due to the exponential advancements in the IT industry especially in software development in recent times, software has become prevalent in everyday human activities and there is a constant pressure on developers to develop and release software features at a rapid pace. In addition to rapid development, better quality product, shorten release time to customers, customer satisfaction and stable and secure releases has become necessary requirements for an organization to stay ahead of their competitors and gain a stable position in the market. The Agile method is the software development method which focuses on small, incremental changes and software developers have to frequently deliver these software features.

While Software developer has to develop features at a rapid pace, people at operations have the responsibility of deploying these new features into the production and they also have to make sure that there will be no system failures due to these deployments and they have the responsibility to protect the stability of infrastructure.

There could arise a problem when, people at IT operations don't have enough inner working knowledge of the deployment and they don't have any idea what is happening on the other side. At the same time, the developer might haven't paid much attention to the performance of the developed feature while designing it. As a result, there is software performance issues which leads to excessive waste of time and also increases the cost of development and deployment.

These challenges can be solved with the help of DevOps which is the combination of Dev from Development and Ops from Operation. DevOps refer to wide range of tools and practices which increase flexibility in the interaction between Development and Operations. There are many definitions of DevOps, but most widely used is one stated from Chef company, "*Cultural and professional movement, focused on how we build and operate high-velocity organizations, born from the experiences from its practitioners* – DevOps definition based on Chef".

This report provides detailed description of the tools and software along with the information on the configuration and implementation of our DevOps project. It also contains the workflow of the project.

1.1 Benefits of DevOps

There are clear benefits of DevOps Configuration. This section will give the outline of some of the important ones.

- **Rapid Delivery:** It allows faster delivery of software features into production environment.
- **Improved Quality:** Through automated software tests, bugs can be detected at early stage of software development. The developer can fix the code and can make sure that the code is bug free.
- **Saving Time and Money:** The major advantage in the CI/CD cycle is that it involves automation of the manual processes for instance, the day to day bug fixes and features can be seamlessly integrated by dockerizing the complete flow of application set up to service starting and testing, this way it will save the time and money for business.
- **Improved Collaboration:**DevOps Configuration has brought cultural and organizational benefits in a company. Collaboration between Development and Operations teams is strongly encouraged throughout the software development Lifecycle, from conception to production.
- **Enhanced Feedback:** Frequent deploy of features means faster time to market and enhanced feedback loop. Now, companies have better and early idea, how customer react to the new features.
- **Security:** Another advantage of DevOps is improved security. DevOps delivery Teams uses shared repositories which enable quick reaction to security vulnerabilities.
- **Competitive Advantage:** **"**With much faster time to release the software feature to the market and continuous involvement of user's feedback, business can maintain a competitive advantage" *Payal Chakravarty.*

2. Background Knowledge:

Developing and Releasing a software product is complicated nowadays. To reduce development time and prevent failure at the end of development, organizations have developed strategies and tools to manage and automate the development, test and release of the software. These strategies and tools are discussed below.

2.1 Continuous Integration

Continuous Integration (CI) is the practice of automating the build and testing code whenever a developer make changes to Git or other Version Control System (VCS). After each small task completion, changes are integrated with the shared version control repository. Whenever a code is committed to the central repository, it triggers the automated build and developer's code changes are validated by creating a build and running automated test against that built.

CI helps to minimize the cost of integration. In the past, developers used to work at distinct locations and features are built separately in isolation and at the end of development cycle, developers have to merge their changes with the other team's code base and it can take several days which can create many merge conflict and it is also difficult to fix bugs at the later time. During CI process, developers can easily identify conflicts and bug issues between new and the updated code.

CI also puts emphasis on automation testing and it relies on test suites to run tests. When a code is merged into central repository, first build of this code is triggered and after that tests are run against the new build. If build or test fails, team is alerted to take care of the failed bug. This way, bugs are identified very early in the development cycle.

2.2 Continuous Delivery

Continuous Delivery (CD) is the process of automating the whole software release process. Basically, it is a step further to continuous integration. It starts from where the CI stops, so the reliable CI setup is prerequisite to implementing CD. The idea for CD is to deliver the new features and updates to customers more frequent in a sustainable way. By automating the delivery process, developers can now easily deploy their code to production at any time with one or few clicks. Now the developers become more productive because they are confident to release the software at any time.

For automating the test and deployment process, CD relay on deployment pipeline; deployment pipeline is an automated system that runs test suites against a build as a series of sequential stages. At every stage, the build either passes or fails the tests. If the build fails the test, it sends failed message to the team and if build passes the test case, it automatically moves to the new stage. At the final stage of the pipeline, build changes can be deployed to production environment with a single click at any time.

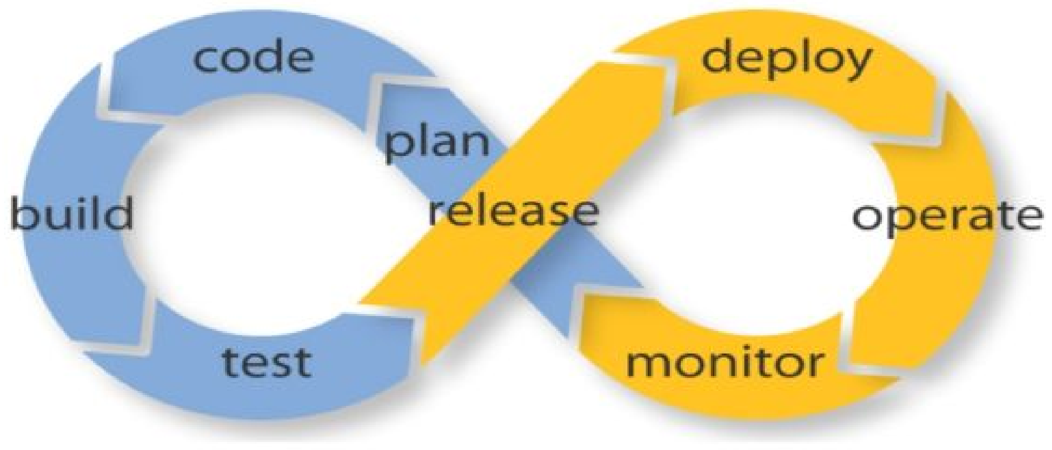


Fig 1: Devops Cycle

3.Concept and Design

3.1 Tools

DevOps tools help partners/stakeholders to implement, monitor, deploy, automate and analyze every process in DevOps. The usage of tools is classified into the following categories.

- Continuous Integration
- Continuous Management
- Deployment
- Build
- Monitoring
- Testing
- Code Analyzer
- Repository

Following table gives the general overview of each tools.

No	Tools	Description	Details
1	Jenkins	Continuous integration	● Open Source ● Automation ● Supports Version Control tools
2	GitHub	Version Control & Repository	● Web-based Git repository Manager ● Wiki ● Issue Tracking ● Open Source
3	Jira	Issue Tracker	● Bug Tracking ● Issue Tracking ● Project Management Functions
4	Ansible	Deployment-Configuration Management	● Open- Source ● Automates Software Provisioning, configuration Management

			● Application Deployment
5	Maven	Build	● Build Automation tools used primarily for Java Projects
6	GIT	Version Control & Repository	● Tracking changes in computer file
7	Vagrant	Virtualization	● Open Source ● Building & Managing VMs
8	Junit	Testing	● Test cases used for testing functional changes
9	Selenium	Testing	● Automation testing tool, tests User Interface
10	Jacoco	Code Analyzer	● Coverage analysis report available in Jenkins
11	Apache Tomcat Server	Web server	● Test deploy web server ● Production deploy web server

Vagrant

To build and manage virtual machine environments in a single workflow. Vagrant, an open source tool simplifies the workflow. It also reduces the development environment setup time. It provides easy to configurable, reproducible, and portable work environments. It supports major virtual solutions like VirtualBox, VMWare,Hyper-V, and also popular configuration tools like Puppet, Ansible and Chef .

- **VirtualBox**

VirtualBox is a cross platform virtualization software package. It is simple but yet very powerful. VirtualBox gives an ability to install multiple operating systems in a virtual environment on a system without any disturbance to host OS. Virtualbox prevents threats like viruses or malwares from guest to host OS because of the isolated environment. VirtualBox can be used in DevOps setup, Virtual Machine is ideal to deploy software to a consistent, known platform. VirtualBox can supports Python or Vagrant to help automate the process and deploy your software on a consistent platform.

Ansible

Ansible is a simple IT automation tool which automates software provisioning, configuration management and application deployment, cloud provisioning, and orchestration. Ansible uses YAML in the form of Playbooks that allow to describe automation jobs.

It works by connecting nodes and pushing out Ansible modules to them. Ansible then executes these modules by SSH (by default) and removes them when finished.

Jenkins

Jenkins is a popular open-source, extensible continuous integration and delivery server which is used to automate building, testing and delivery related task of a software and it uses so called jobs to define how to build, test and deploy projects. Jenkins flexibility can be improved by additional community developed plugins.

Jenkins can execute Freestyle, Apache ant, Apache Maven scripts and shell scripts for windows and Linux environments. Junit test report can be generated by builds while other test frameworks are also supported through plugins. Jenkins supports version control tools such as Subversion, Git, Mercurial, CVS and Perforce. Jenkins build can be triggered through committing, polling the repository, manually triggers, via API or other successful builds.

Jenkins uses pipeline and coverage report.

Apache Maven

Maven is a simple build automation tool which is primarily used for Java Projects. Maven is also defined as project management tool based on Project object model (POM) and aims to provide complete build life cycle framework of an application. Maven also makes easy for developers to easily build projects, easily add jars and other dependencies, it provides project information i.e. log document, dependency list, unit test reports. Maven is very helpful while updating central repository of JARs/WARs and other dependencies and these JARs/WARs can be shared across any distributed environments. It can easily integrate with SMC like Git or Subversion.

Version Control - Git

Version control system (VCS) or sometimes known as source code management (SCM) is a tool which helps development team to track and manage changes in their filesystem over time. VCS is a collaborative tool to share and integrate file system changes with other VCS users. VCS tracks every change in the code and in case of any mistake, developers can compare earlier versions of the code and can fix it. A repository is a VCS term which describes when VCS is tracking a filesystem. One of the benefit of VCS is in Continuous delivery.

- **Github:**

  Git is the most popular version control tools which is used today. It is a free and open source distributed VCS, a category known as DVCS. Developers can see timeline of their changes and progression of any project in one place. In Git, developers can make change to production code using branches. A repository in Git contains the entire collection of files and folders associated with the project.

Apache Tomcat

Apache Tomcat is an open-source web server and servlet container that was created to run servlet and Java Server Pages (JSP) web applications. A servlet is a server-side program, managed my container, which accepts clients request and return a customized or dynamic response for each request. Tomcat also provides additional functionalities to for complete web applications solutions such as Tomcat manager application, specialized realm implementations and Tomcat valves.

Junit Test

A unit test is a piece of code that executes a specific functionality in the code. Test usually run periodically, often if there is any change in source code. It is not suitable for testing complex interface interaction.

Selenium Test

Selenium is a free, open-source and a popular automation testing tool for testing web applications across different browsers and applications. Testing done using selenium is called selenium testing. Selenium is not just a single tool but has four components i.e. Selenium IDE, Selenium WebDriver, Selenium and Selenium Grid. Many of the web browsers supports Selenium by default. Tests can be written in any of the popular programming languages i.e. C#, Java, Perl, Php, Groovy, Python and Ruby. There is also a record/playback tool in Selenium for authoring tests without need to learn any scripting language.

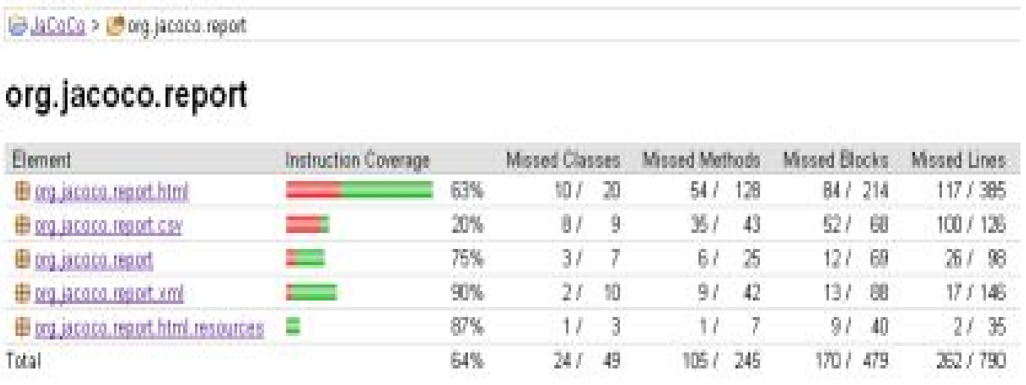
Jira

Jira Software is a popular incident management tool that integrates Bug Tracking, Agile Project management, Issue Tracking and workflow capabilities into a single application. It can be used in Help Desk, Support and Customer Service to create ticket and track the status of issued ticket. Jira supports more than a dozen reports to track progress of task deadlines and team members contribution and these reports are easy to configure and better visualization of metrics for stakeholders. It also supports with 1000's of add-in to connect with different software like Git, Jenkins etc.

Jacoco

To measure and report Java Code Coverage, we use a reporting tool called Jacoco.

The degree to which the source code runs can be described by Jacoco.



JaCoCo > org.jacoco.report

org.jacoco.report

Element	Instruction Coverage		Missed Classes	Missed Methods	Missed Blocks	Missed Lines
org.jacoco.report.html		63%	10 / 20	54 / 128	84 / 214	117 / 385
org.jacoco.report.csv		20%	8 / 9	35 / 43	52 / 68	100 / 126
org.jacoco.report		75%	3 / 7	6 / 25	12 / 69	26 / 98
org.jacoco.report.xml		90%	2 / 10	9 / 42	13 / 88	17 / 146
org.jacoco.report.html.resources		87%	1 / 3	1 / 7	9 / 40	2 / 35
Total		64%	24 / 49	105 / 245	170 / 479	262 / 790

Fig 2: Jacoco Jenkins report

4. Implementation:

The below diagram depicts the continuous integration and delivery workflow.

The workflow process creates a virtual machine with a preconfigured DevOps environment. An image of the virtual machine should be possible to be host on any system that is able to run VirtualBox. As soon as the vagrant is setup, it will install the virtual machine. Once the virtual machine is up and running, it will provision the ansible playbook which installs the necessary softwares, starts the jenkins and loads the jenkins jobs. A jira ticket will be created for every changes that has to be made in the code and it will be assigned to the developer. As soon as the developer pushes the code to github, the Jira ticket will be updated through which we can track the changes. At the same time, jenkins starts to build the project and performs Junit tests. If the test fails, automatically a Jira ticket will be created for the failed test case. If the test passes, build will be successful and it will deploy the project in the test server. In the test server, UI testing is performed and a coverage report is generated. If the test is passed, project will finally be deployed in the Tomcat server. The coverage report and Jira report are merged and a final report is generated.

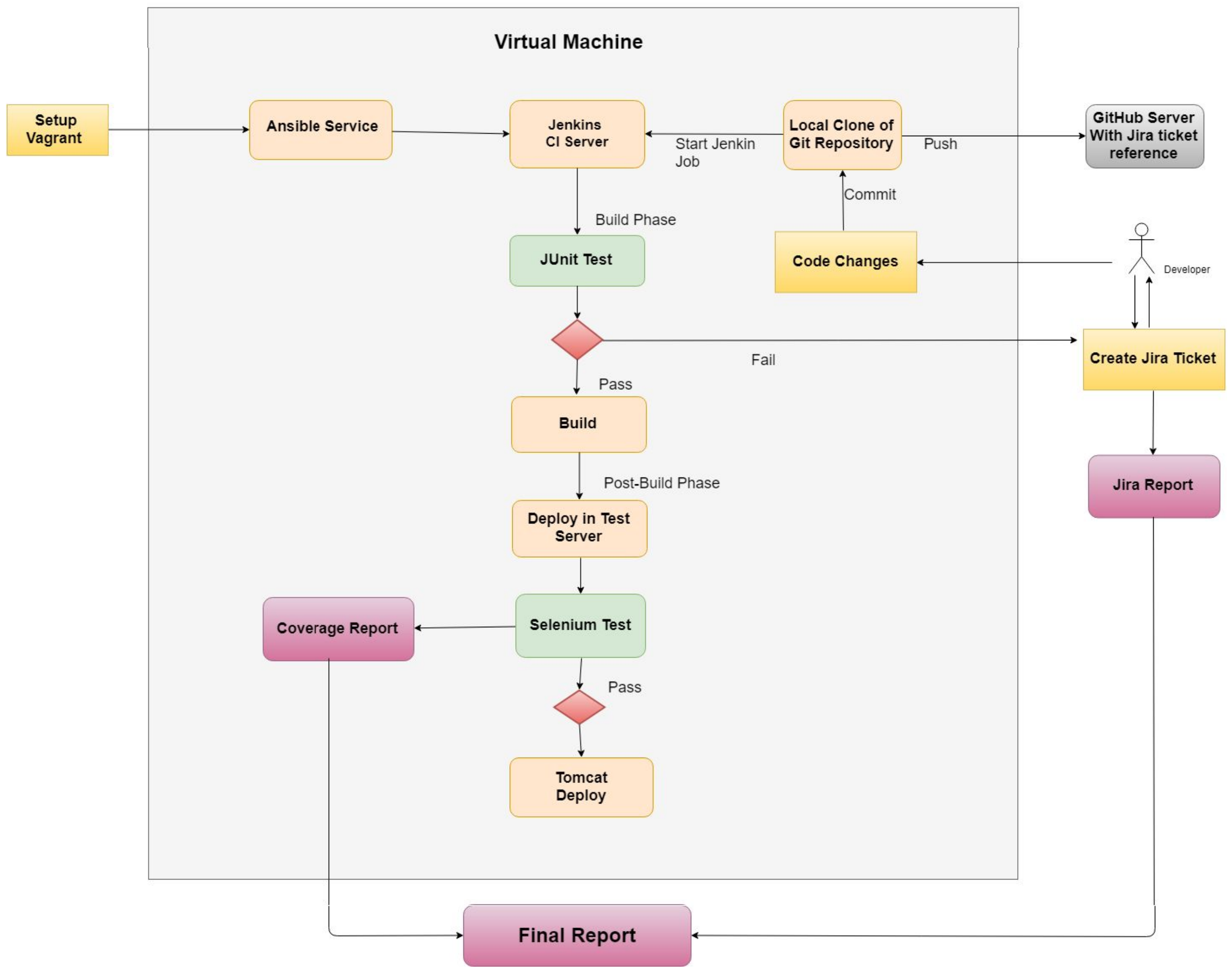


Fig 3: Workflow Diagram

Pre-requisites

Operating systems that are able to be used as a host machine are any of Red Hat, Debian, CentOS, OS X and BSDs. Windows can't be used, because Ansible as the central provisioning tool does not support it. The development and testing of the project was done with Ubuntu 16.04 LTS and Ubuntu 18.04 LTS.

Install VirtualBox to host the virtual machine

Install Vagrant to create and configure the virtual machine

Install Ansible to install software and manage the configuration of the virtual machine

Internet connection to allow Ansible to download necessary software during the provision of the virtual machine and to enable a connection to Jira during the creation of the report.

Create the virtual machine

The project can be cloned via git after all dependent software is installed on an operating system that is supported by Ansible. With a terminal go into any directory of the project and start the creation of the virtual machine with “vagrant up” command. There upon Vagrant will create a virtual machine within virtual box and will trigger Ansible to configure the virtual machine. This process will take about 10 to 15 minutes, depending on the host machine and the internet connection, since a lot of software needs to be downloaded and installed. Sometimes the provisioning fails because necessary software could not be downloaded on time. In this scenario vagrant can be provisioned with “vagrant provision” command, which will reinstall the necessary softwares. The creation of the virtual machine and the provisioning of the virtual machine are two distinct processes. After the virtual machine is running in VirtualBox any changes can be performed in the Ansible scripts with the vagrant provision command into effect. Only Ansible commands that were altered are triggered after a rerun of vagrant provision, which drastically reduce execution time. Some changes however will not be realized, and it is necessary to delete the whole virtual machine with vagrant destroy and rebuild it with vagrant up.

Verify the initialization of the virtual machine by enter it with the command “vagrant ssh”. The Jenkins server should now be running on port 7070 and the Tomcat server on port 8090. The server is accessible from a browser with the URLs http://localhost:7070 and http://localhost:8090 respectively. The credentials can be found in ./main.yml.

Vagrant Configuration

The file ./Vagrantfile contains the configuration of vagrant. Among other things it defines what virtual machine is used, the port mapping, the host name of the virtual machine and the provisioning tool. The ports for the Jenkins and Tomcat server are passed here as extra variables to Ansible.

Another noticeable fact is, that vagrant will create a folder /vagrant on the virtual machine that contains the project directory. This folder is shared by both the host machine and the virtual machine. That means, if you add the file ./example.txt on the host, this file is also accessible from the virtual machine /vagrant/example.txt.

Important commands

vagrant up: Command vagrant to create a virtual machine in VirtualBox and to run Ansible for the provision.

vagrant provision: Vagrant just runs Ansible for the provision. This command is only possible to execute if the virtual machine is already running in VirtualBox.

vagrant destroy: Destroy the whole virtual machine. Changes that are not saved with mechanisms mentioned later in this document will be irreversible deleted.

vagrant ssh: Access the virtual machine.

vagrant status: Get the status of the virtual machine vagrant is running.

Ansible

As the provisioner Ansible is responsible for all software on the virtual machine and almost all configuration of this software. After the setup of the virtual machine in VirtualBox, Vagrant will execute ./playbook.yml. This playbook will load the most important configuration variables from ./main.yml, such as the Jenkins and Tomcat credentials, the git username and git email address and the home path of Jenkins. After that multiple Jenkins roles and other tasks are executed.

- **Roles**

  Most of the software is installed via Ansible roles. The roles can be obtained from Github or from Galaxy, a web page that is dedicated to them. You can find all roles that are used by the project in the directory ./roles. The ./main.yml only contains a small fraction of variables to configure Ansible. Most roles have a directory defaults and a file main.yml which has various variables that can be used to configure and adjust the installed software (e.g. for Jenkins ./roles/geerlingguy.jenkins/defaults/main.yml).

Following software is installed via Ansible roles:

- Git
- Java
- Maven
- Jenkins
- Pip
- Jira
- Tomcat
- LibreOffice

Configure Jenkins

Jenkins is the central Continuous Integration and Continuous Deployment tool. Webhooks can be used to trigger jobs and terminal commands can be used from within the jobs to execute any kind of script on the host machine. There exist various plugins that will extend the functionality of Jenkins. Plugins can be found by using their IDs in the Jenkins user interface under http://localhost:7070/pluginManager/available. If a plugin needs to be added permanently,add its ID to the variable jenkins_plugins in ./main.yml.

Jenkins save all of its configurations in XML files in its home directory /var/lib/jenkins on the virtual machine. Configuration files needs to be saved manually with a script to keep modified configuration details even after the virtual machine is destroyed. Be aware, that only some of the configurations will be saved at the moment! This includes deleting, creating or altering the Jenkins jobs and the authentication keys managed by Jenkins. Some authentication keys of plugins like JiraTestResultReporter are stored in separate XML files. The configuration files will be saved in ./files/jenkins_config_files/. Please take into consideration that executing the script will lead to the deletion of any old configuration files.

- **How to install Java JDK and Maven via Ansible?**

  Installed Maven via Ansible using Ansible roles. For building Maven projects, the Java and Maven Binaries should be on the correct path and correct version in the virtual machine. In this project Java roles from "https://github.com/idealista/java-role" has been used. Here, the role has to be downloaded as ZIP archive and then should be unpacked into the roles folder of the project.

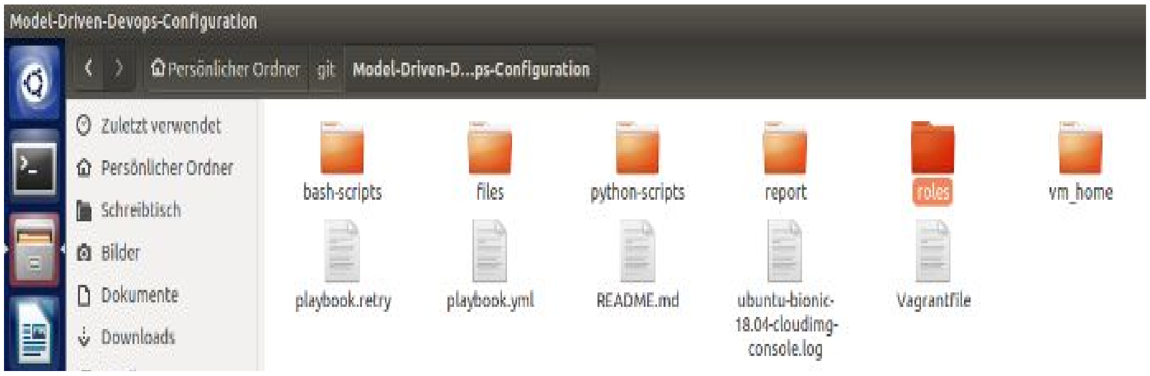
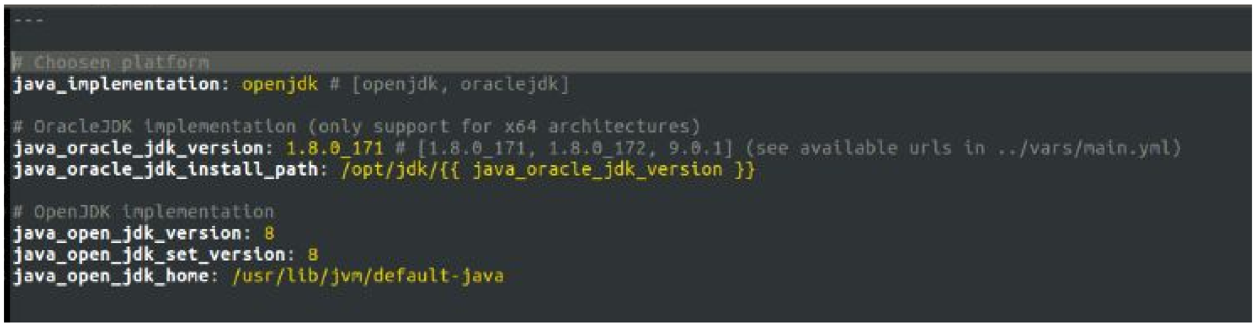


Fig 4.1: roles folder

To configure the path of the Java binaries one can use the main.yml inside die folder defaults in the role directory.

The next step is to inherit the role in the Ansible playbook.yml. In order to inherit role, add the name of the role directory (in our case java-role-master) under the point roles in the yml file(Important Note: Ansible installs the roles in the same order as it is written in the playbook file)



```
---

# Choosen platform
java_implementation: openjdk # [openjdk, oraclejdk]

# OracleJDK implementation (only support for x64 architectures)
java_oracle_jdk_version: 1.8.0_171 # [1.8.0_171, 1.8.0_172, 9.0.1] (see available urls in ../vars/main.yml)
java_oracle_jdk_install_path: /opt/jdk/{{ java_oracle_jdk_version }}

# OpenJDK implementation
java_open_jdk_version: 8
java_open_jdk_set_version: 8
java_open_jdk_home: /usr/lib/jvm/default-java
```

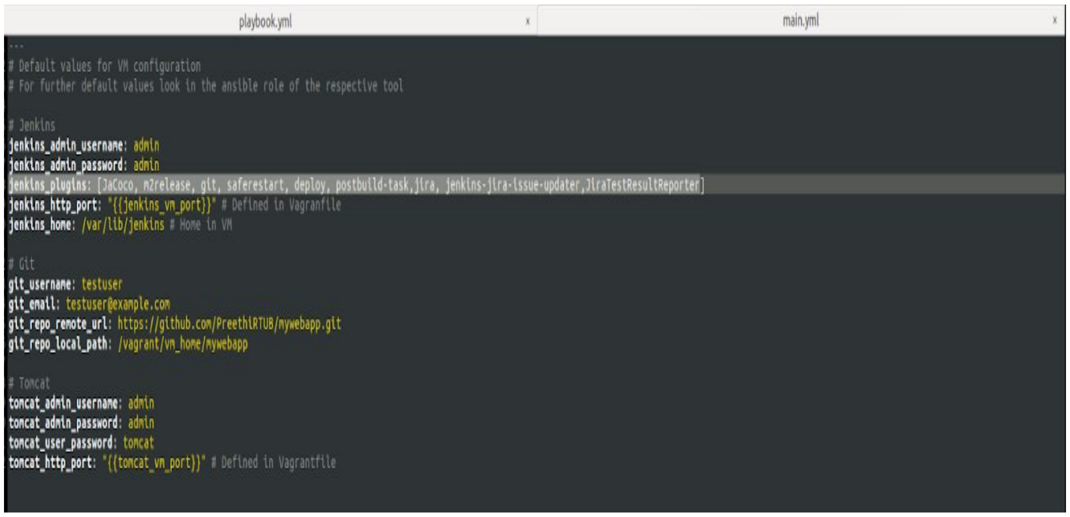
Fig 4.2: Java and Maven installations

So if there are any dependencies (like Maven depends on Java), write dependencies in the correct order. After this step, perform same steps/procedure with the Maven roles using the link https://github.com/tecris/ansible-maven. At the end it should look like above Fig 4.2:

- **How to use Jenkins with Java and Maven?**

  Now configure Jenkins to use the Java and Maven by configuring Jenkins jobs.

  Our Project Perspective:The main challenge in configuring Jenkins for our DevOps environment is to save the configurations and load into to the virtual machine while it is build from scratch. Since Jenkins is deployed locally on the guest machine our approach was to first build the virtual machine with a running instance of Jenkins in it. Then we configured the Jenkins server manually via the web interface and saved the configuration xml files which where stored in the Jenkins home folder on the VM. Before starting the VM, the plugins for Java and Maven should be loaded into Jenkins. This we achieved by editing the main.yml in the root directory of the git project.



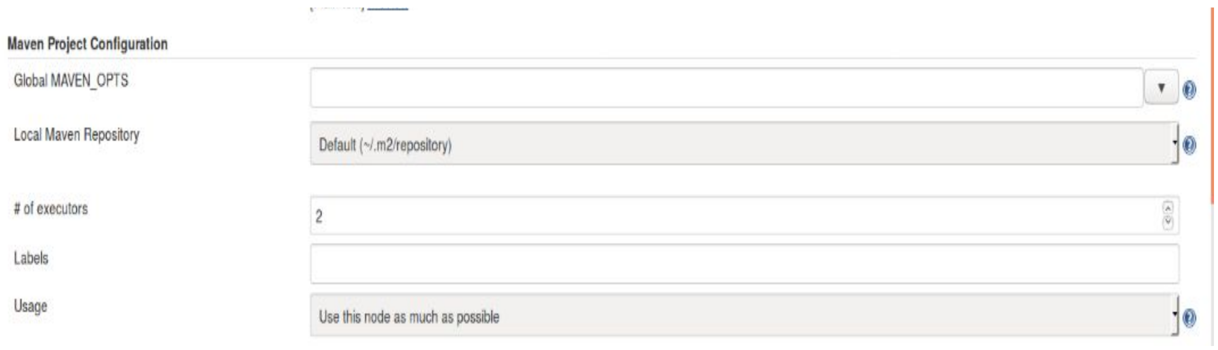
```
---
# Default values for VM configuration
# For further default values look in the ansible role of the respective tool

# Jenkins
jenkins_admin_username: admin
jenkins_admin_password: admin
jenkins_plugins: [JaCoco, m2release, git, saferestart, deploy, postbuild-task,jira, jenkins-jira-issue-updater,JiraTestResultReporter]
jenkins_http_port: "{{jenkins_vm_port}}" # Defined in Vagranfile
jenkins_home: /var/lib/jenkins # Home in VM

# Git
git_username: testuser
git_email: testuser@example.com
git_repo_remote_url: https://github.com/PreethiRTUB/mywebapp.git
git_repo_local_path: /vagrant/vm_home/mywebapp

# Tomcat
tomcat_admin_username: admin
tomcat_admin_password: admin
tomcat_user_password: tomcat
tomcat_http_port: "{{tomcat_vm_port}}" # Defined in Vagrantfile
```

Fig 4.3: Jenkins_plugins in main.yml

If we open the main.yml file there is "jenkins_plugins" which has all the plugin IDs. The plugin IDs can be found in the Jenkins plugin documentation.Note that the right plugins for the previously installed Ansible roles should be added correctly. After adding the plugin IDs, the corresponding web masks will appear in Jenkins web interface under manage jenkins→ global tool configuration.

Under Maven insert the following:

Further under configure jenkins → configure system do this:



Maven Project Configuration

Global MAVEN_OPTS:

Local Maven Repository: Default (~/.m2/repository)

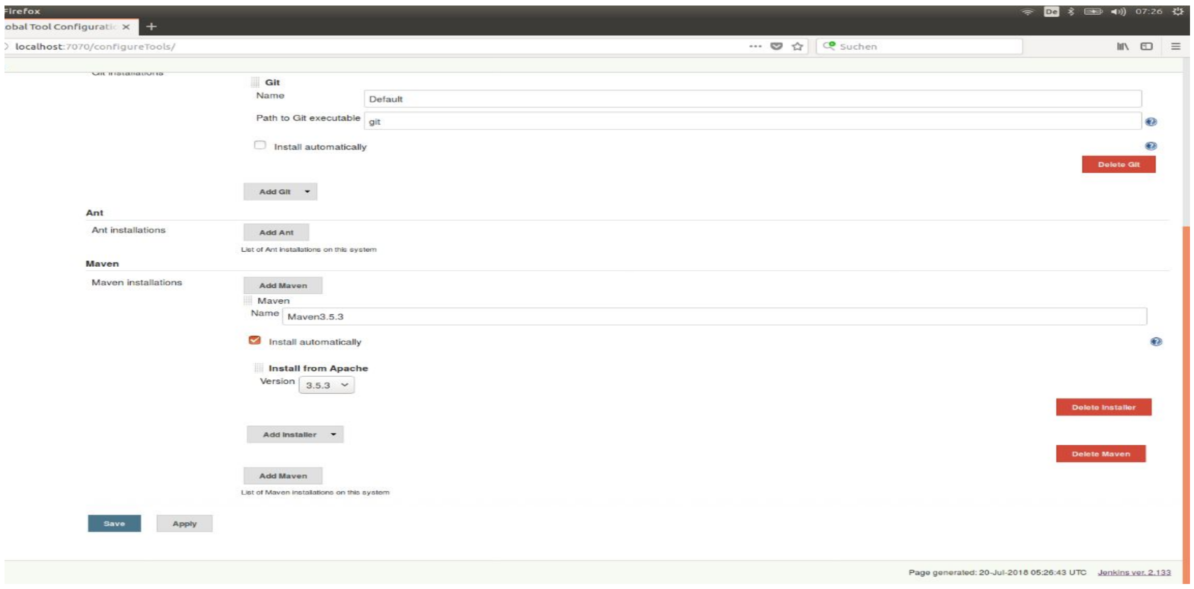
# of executors: 2

Labels:

Usage: Use this node as much as possible

Fig 4.4: Maven Configuration in Jenkins

For Java no further changes are needed if it lies on the right path. Default path for Jenkins is “/usr/lib/jvm/java-8-openjdk”. Now we need to save everything to be reused later.



localhost:7070/configureTools/

Git
Name: Default
Path to Git executable: git
☐ Install automatically

Ant
Ant installations
List of Ant installations on this system

Maven
Maven installations
Maven
Name: Maven3.5.3
☑ Install automatically
Install from Apache
Version: 3.5.3
List of Maven installations on this system

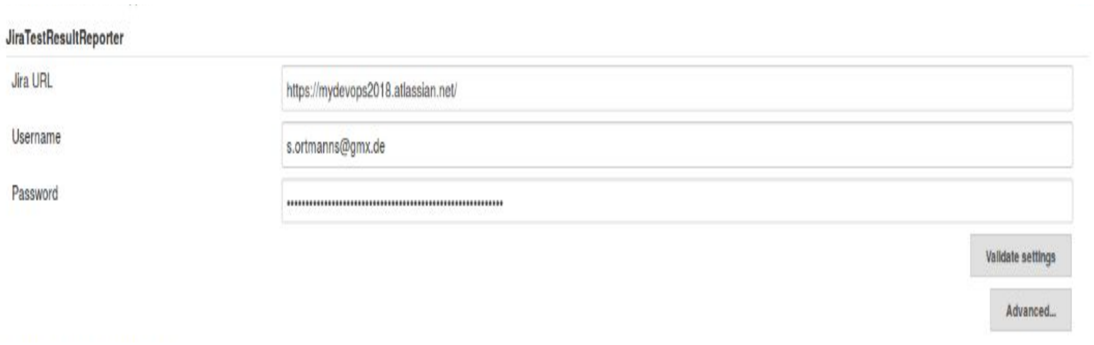
Page generated: 20-Jul-2018 05:26:43 UTC Jenkins ver. 2.133

Fig 4.5: Maven Config details

Therefore we used the python script saveConfigurations.py which can be executed. There is also the option to copy the configuration xml files from the Jenkins home directory and store it on the local host machine. Then copy the files via the Ansible playbook back in the VM, when rebuilding it. In our case most of that is done by another python script named loadConfiguration.py which is also triggered by Ansible.

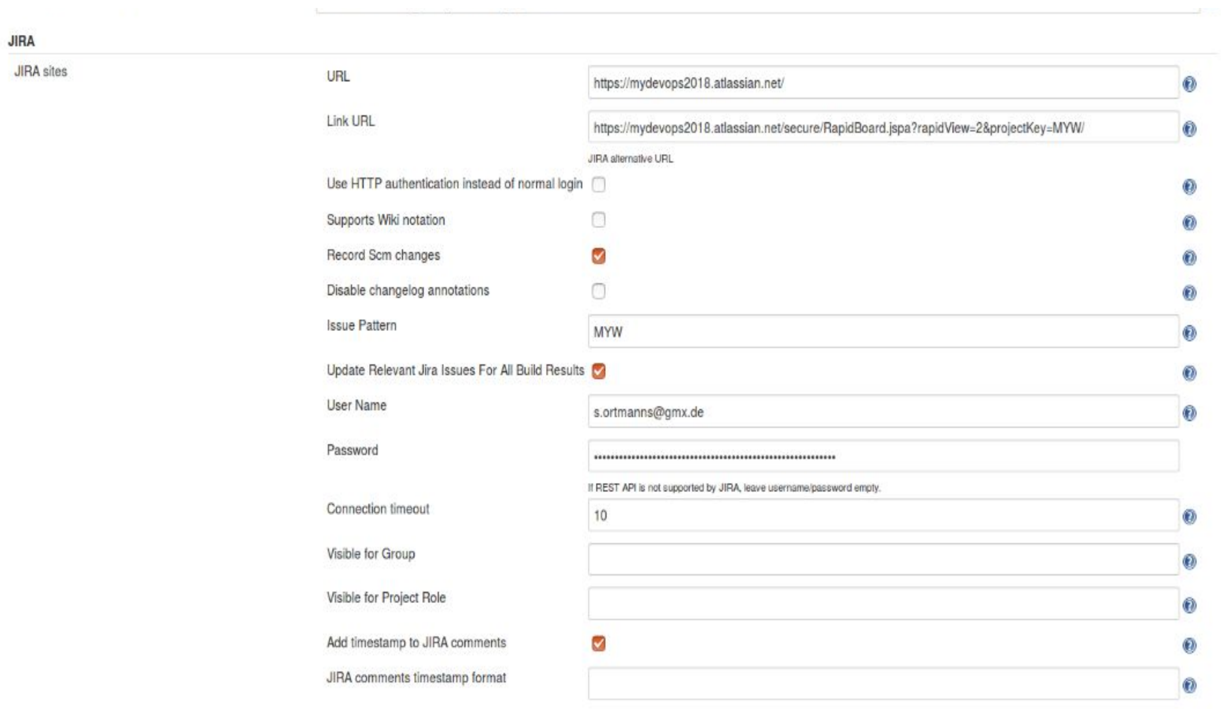
- **How to use Jira with Jenkins?**

First step install the Jira Ansible role from https://github.com/alvistack/ansible-role-jira path.
second step, load the plugins: jira, jenkins-jira-issue-updater and JiraTestResultReporter into Jenkins.[Note:Add only 3 plugin ID's in the jenkins_plugin line in the main.yml. Because Jira is a paid tool, In our project we used free online trial account, but instead of online account, Jira can be used by installing it on the local machine]. To make Jira work correctly connect it to the online Github account, like explained in below Jira Set up section.
Once the above setup is completed execute "vagrant up" command and navigate to the web interface of Jenkins again. Go to manage jenkins → configure system and fill the forms as shown below fig 4.6, 4.7 and finally save the configuration in the end.


JiraTestResultReporter

Jira URL	https://mydevops2018.atlassian.net/
Username	s.ortmanns@gmx.de
Password	••

Fig 4.6: Jira configuration(1)



JIRA

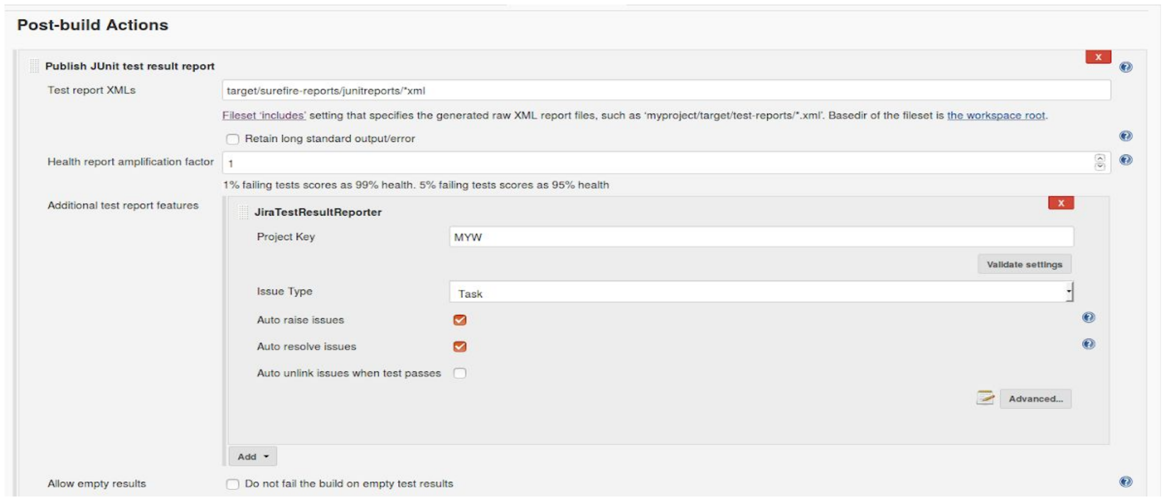
JIRA sites

URL	https://mydevops2018.atlassian.net/
Link URL	https://mydevops2018.atlassian.net/secure/RapidBoard.jspa?rapidView=2&projectKey=MYW/
	JIRA alternative URL
Use HTTP authentication instead of normal login	☐
Supports Wiki notation	☐
Record Scm changes	✓
Disable changelog annotations	☐
Issue Pattern	MYW
Update Relevant Jira Issues For All Build Results	✓
User Name	s.ortmanns@gmx.de
Password	••
	If REST API is not supported by JIRA, leave username/password empty.
Connection timeout	10
Visible for Group	
Visible for Project Role	
Add timestamp to JIRA comments	✓
JIRA comments timestamp format	

Fig 4.7: Jira configuration(2)

- **How to automate ticket creation?**

  To automate the ticket generation, first need a job in Jenkins. Go to the jobs details and click configure. Then choose the post-build actions register card, go all the way down and choose add post-build action. Then activate publish Junit test report and proceed as follows:



Post-build Actions

Publish JUnit test result report

Test report XMLs: target/surefire-reports/junitreports/*xml

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is the workspace root.

☐ Retain long standard output/error

Health report amplification factor: 1

1% failing tests scores as 99% health. 5% failing tests scores as 95% health

Additional test report features

JiraTestResultReporter

Project Key: MYW

Issue Type: Task

Auto raise issues: ✓

Auto resolve issues: ✓

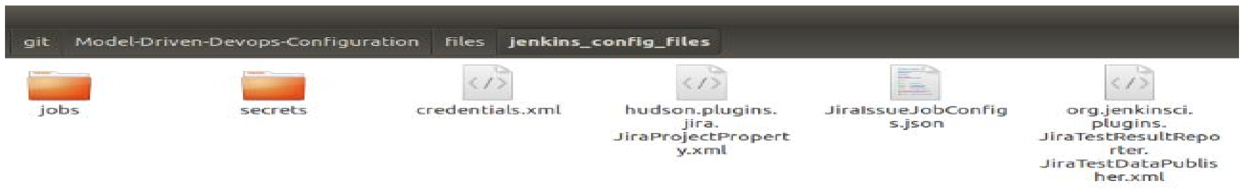
Auto unlink issues when test passes: ☐

Allow empty results: ☐ Do not fail the build on empty test results

Fig 4.8: Automate Ticket configuration

Now it is important to save the configuration the correct way. Therefore first of all use save Configuration python script again. Then go into the Jenkins home directory in the VM and copy the following two files into your local git repository into the folder files/jenkins_config_files: hudson.plugins.jira.JiraProjectProperty.xml, org.jenkinsci.plugins.JiraTestResultReporter.JiraTestDataPublisher.xml.

Next use the copy command in the Ansible playbook.yml to provision them into the VM when rebuild. Further we need a third file called JiraIssueJobConfigs.json, which you can find in {jenkins_home}/jobs/Build and proceed the same way. In the end, the new part of your playbook should look as below fig 4.9.



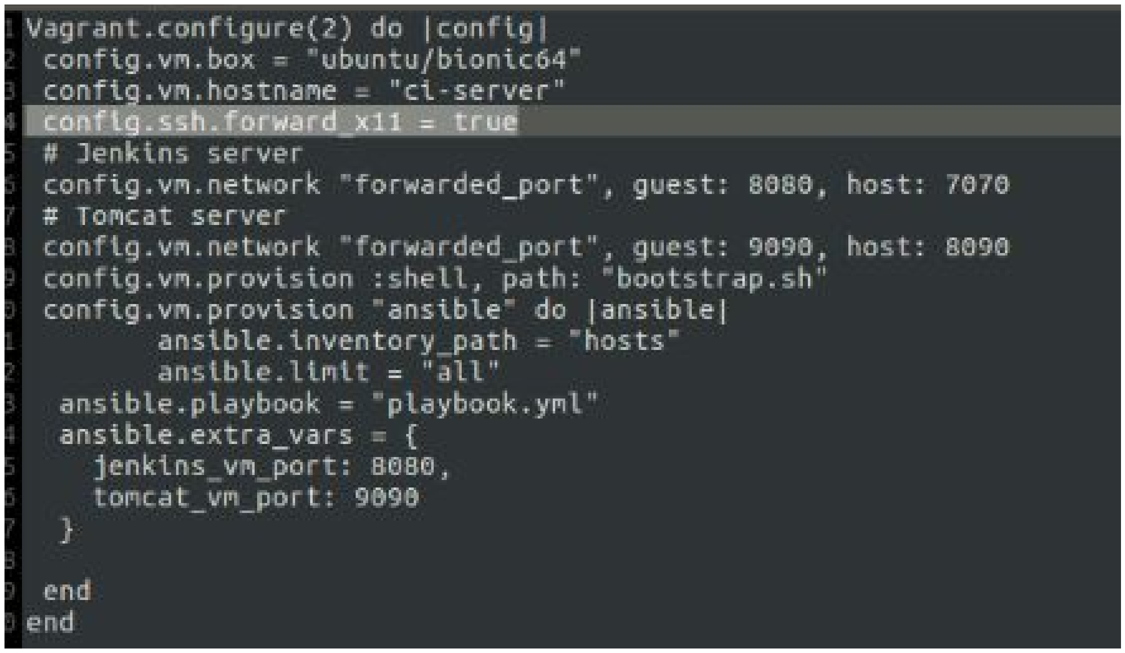
git | Model-Driven-Devops-Configuration | files | jenkins_config_files

jobs
secrets
credentials.xml
hudson.plugins.jira.JiraProjectProperty.xml
JiraIssueJobConfigs.json
org.jenkinsci.plugins.JiraTestResultReporter.JiraTestDataPublisher.xml

Fig 4.9: Jenkins_config_files folder

- **How to enable x11 forwarding in the VM ?**

  To enable x11 forwarding just add the line: “config.ssh.forward_x11 = true” to the Vagrantfile.



```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/bionic64"
  config.vm.hostname = "ci-server"
  config.ssh.forward_x11 = true
  # Jenkins server
  config.vm.network "forwarded_port", guest: 8080, host: 7070
  # Tomcat server
  config.vm.network "forwarded_port", guest: 9090, host: 8090
  config.vm.provision :shell, path: "bootstrap.sh"
  config.vm.provision "ansible" do |ansible|
         ansible.inventory_path = "hosts"
         ansible.limit = "all"
   ansible.playbook = "playbook.yml"
   ansible.extra_vars = {
     jenkins_vm_port: 8080,
     tomcat_vm_port: 9090
   }

  end
end
```

Fig 5: enable x11 forwarding in VM

- **How to configure and save configurations:**
  1. Configure Jenkins by using its user interface at http://localhost:7070/
  2. Enter the virtual machine by executing vagrant ssh from within the project directory
  3. Change to the directory on the virtual machine to /vagrant/python-scripts
  4. Execute python saveConfiguration.py admin admin, where 'admin' and 'admin' are the Jenkins username and password.

Git repository

- **Load repository**

  Ansible executes a Git clone command during the provision phase. This command will try to clone the repository mentioned in the variable git_repo_remote_url and installs it to the path on the virtual machine mentioned in the variable git_repo_local_path; Both variables are defined in ./main.yml. At the moment only repositories can be cloned that do not need verification, but with small changes a verification via username and password or SSH key should be realizable.

- **Change repository**

  The java project that is at the moment loaded into the virtual machine (https://github.com/GitUserName/mywebapp.git) uses Maven for executing the tests and build the project. The Jenkins jobs Build and Deployment are specialized to exactly run the project (especially the execution of the test, since this is realized with Maven profiles).

  Using a non Maven Java project will break test- and build-phase in the preconfigured Jenkins jobs and using a non Java project will also break coverage code analysis done by Jacoco. As per the project requirements Jenkins jobs can be configured.

Git hook

Under the current configuration Ansible will copy the Git hook file ./files/post-commit into the hooks directory of the loaded Git project. This file is executed after each commit of the project and will notify Jenkins about the changes.

Jira Set Up

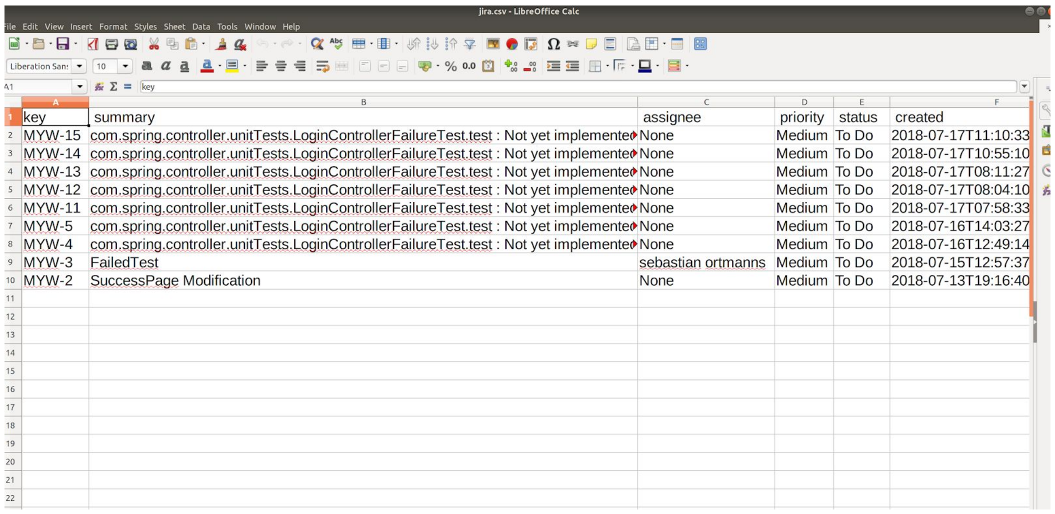
For using Jira in the project the following steps has to be done

- Create Jira admin account
- In github, generate OAuth for the account and get the Client ID and Client Password.
- In Jira, generate a DVCS account and link JIRA with github by giving the Client ID and Client Password.
- This automatically pulls all the repositories from the github to JIRA.

- Select the repository that is required for this automation and it gets integrated with JIRA
- Make sure the SMART COMMIT tab is checked so that when you make changes in github it automatically reflects in JIRA.
- Now create a ticket in JIRA by giving the summary or purpose of the ticket and assign the ticket to a developer.
- As soon as the developer makes changes in the project and commits the code with the ticket ID in the commit message, the JIRA ticket with that ticket ID is updated automatically.
- Through this we can track the status as well as the part of the code that has been modified in the commit.

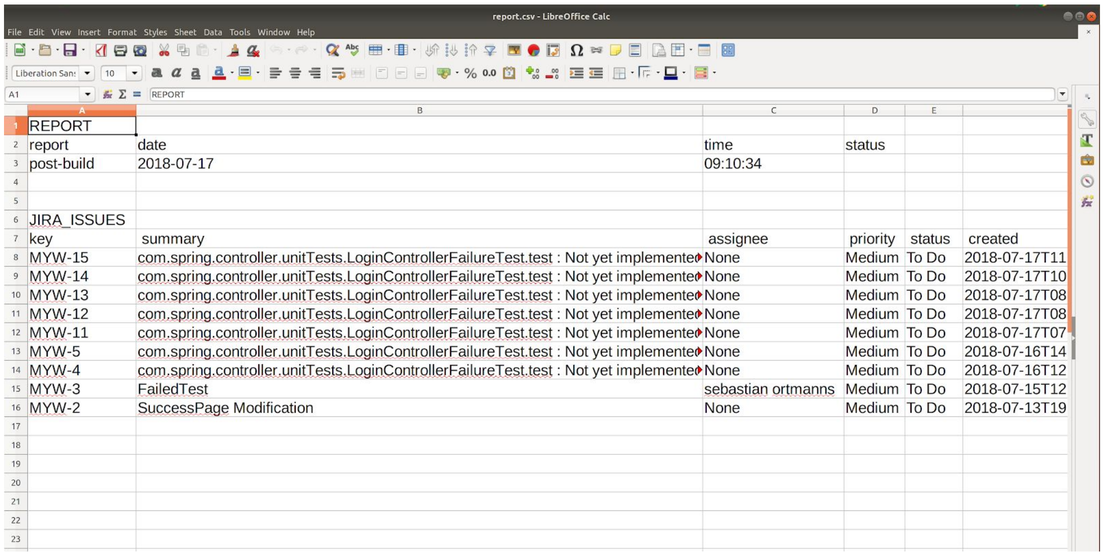
Report

The DevOps report is created by multiple python scripts. The script /vagrant/python-scripts/postBuildScript.py will be executed each time the Jenkins Build job is triggered and will itself execute other scripts that will create the report. The final report on the virtual machine can be found in the /home/vagrant/report. The CI/CD process in this project generate Jira.csv, Jacoco.csv and final integrated report.csv with Jira, Jacoco and build details. Below images shows the sample Jira and Final integrated reports.



key	summary	assignee	priority	status	created
MYW-15	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-17T11:10:33
MYW-14	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-17T10:55:10
MYW-13	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-17T08:11:27
MYW-12	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-17T08:04:10
MYW-11	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-17T07:58:33
MYW-5	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-16T14:03:27
MYW-4	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-16T12:49:14
MYW-3	FailedTest	sebastian ortmanns	Medium	To Do	2018-07-15T12:57:37
MYW-2	SuccessPage Modification	None	Medium	To Do	2018-07-13T19:16:40

Fig 6.1: Jira report with number of open issues


report.csv - LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sans
10
A1
REPORT

	A	B	C	D	E	F
1	REPORT					
2	report	date	time	status		
3	post-build	2018-07-17	09:10:34			
4						
5						
6	JIRA_ISSUES					
7	key	summary	assignee	priority	status	created
8	MYW-15	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-17T11
9	MYW-14	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-17T10
10	MYW-13	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-17T08
11	MYW-12	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-17T08
12	MYW-11	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-17T07
13	MYW-5	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-16T14
14	MYW-4	com.spring.controller.unitTests.LoginControllerFailureTest.test : Not yet implemente	None	Medium	To Do	2018-07-16T12
15	MYW-3	FailedTest	sebastian ortmanns	Medium	To Do	2018-07-15T12
16	MYW-2	SuccessPage Modification	None	Medium	To Do	2018-07-13T19

Fig 6.2: Merged report with Jira, Jacoco details

5.Model Driven Devops Configuration Meta Model

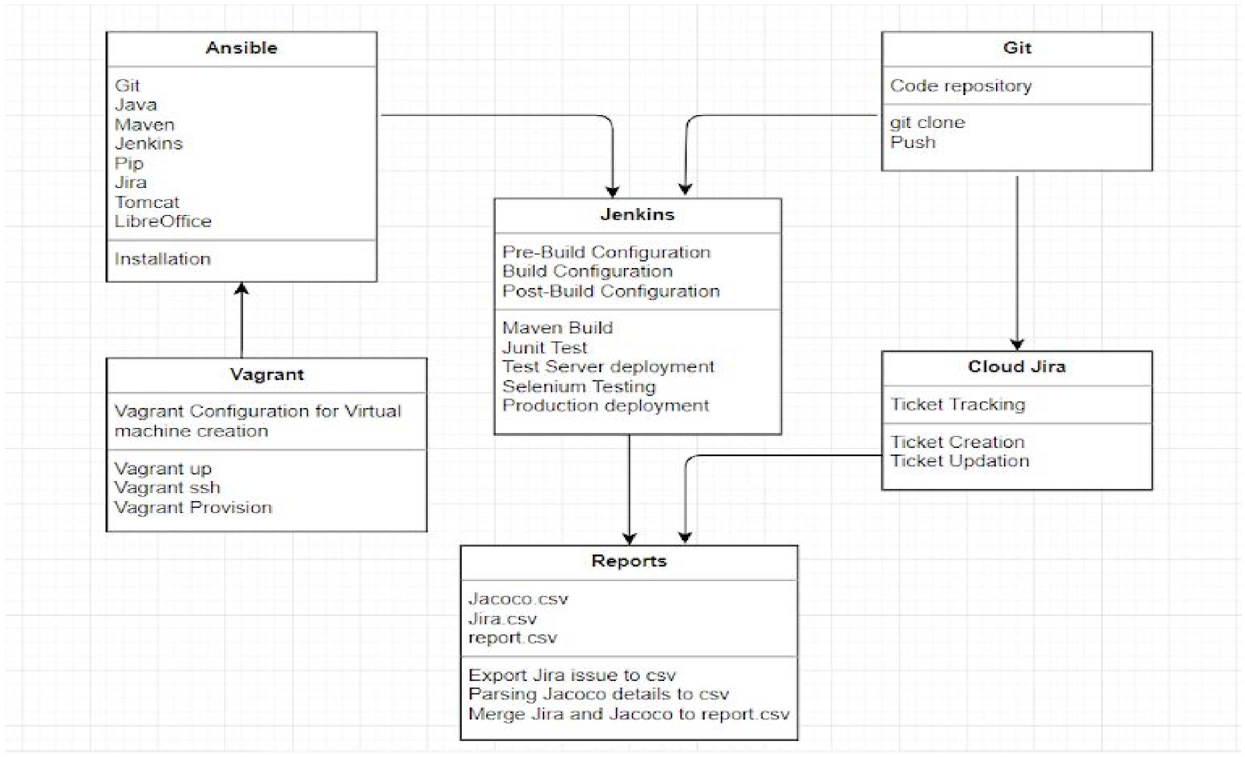


Fig 7: Meta Model

6.Conclusion

This project involves a combination of various tools and softwares to build a DevOps environment that enables the organizations to deliver services at a faster rate and at the same time maintaining the customer satisfaction. It aims at bringing the developers and operations teams together and also automating the workflows and delivery process. This environment helps in continuously measuring the application performance by monitoring, measuring and improving the code and operations every day. This solves agile drawbacks to ensure a smooth development Lifecycle. This project also records the traceability of the changes done at any point of time which helps developers to understand the status of the project. It ensures version control through continuous integration and continuous delivery**.**

References

[1] http://jultika.oulu.fi/files/isbn9789526217116.pdf

[2] http://www.doria.fi/bitstream/handle/10024/148944/DevOps.pdf?sequence=1

[3] https://blog.newrelic.com/2014/05/16/devops-name/

[4] https://dzone.com/articles/what-is-devops-the-beginners-guide-from-logzio

[5] https://books.google.de/books?id=N5RRDwAAQBAJ&printsec=frontcover#v=onepage&q&f=false

[6] https://dzone.com/articles/why-organizations-fail-to-adopt-cicd

[7] https://devops.com/ten-ways-devops-will-benefit-department/

[8]https://www.lntinfotech.com/wp-content/uploads/2018/01/LTI-Continuous-Integration-Using-Jenkins-WP.pdf

[9] https://www.ca.com/us/why-ca/continuous-delivery.html

[10] https://puppet.com/blog/continuous-integration-success-depends-on-automation

[11] https://www.digitalocean.com/community/tutorials/an-introduction-to-continuous-integration-delivery-and-deployment#what-is-continuous-deployment-and-why-is-it-helpful

[12] https://code-maze.com/what-is-continuous-integration/

[13] https://www.digitalocean.com/community/tutorials/ci-cd-tools-comparison-jenkins-gitlab-ci-buildbot-drone-and-concourse

[14] https://dzone.com/articles/how-to-setup-continuous-delivery-environment

[15] https://jenkins.io/doc/

[16] https://en.wikipedia.org/wiki/Jenkins_(software)

[17] http://www.vogella.com/tutorials/Jenkins/article.html

[18]

http://fileadmin.cs.lth.se/cs/personal/lars_bendix/research/ascm/in-depth/hembrinkstenberg-2013.pdf

[19] https://pdfs.semanticscholar.org/presentation/75dc/a05e3571606bd13a8610de58c353ba32de41.pdf

[20] https://docs.microsoft.com/en-us/azure/devops/what-is-continuous-integration

[21]

https://semaphoreci.com/blog/2017/07/27/what-is-the-difference-between-continuous-integration-continu ous-deployment-and-continuous-delivery.html

[22]

https://www.infoworld.com/article/3271126/application-development/what-is-cicd-continuous-integration -and-continuous-delivery-explained.html

[23] https://www.studytonight.com/maven/introduction-to-maven

[24]

https://www.owasp.org/images/a/a8/Presentation-2016-NOUNCE-Vagrant_-_Up_and_Running-v2.4.pdf

[25] https://dl.acm.org/citation.cfm?id=2613694

[26] https://capgemini.github.io/infrastructure/an-introduction-to-virtualbox/

[27]

https://bitbucket.org/product/version-control-software?_ga=2.4451387.96234399.1532662870-21203298 04.1528729643

[28] https://guides.github.com/introduction/git-handbook/

[29] https://doc.lagout.org/programmation/tech_web/Apache/Apache%20Tomcat%207.pdf

[30] https://en.wikipedia.org/wiki/Apache_Tomcat

[31] http://www.vogella.com/tutorials/JUnit/article.html

[32] https://www.credosystemz.com/training-in-chennai/best-selenium-training-in-chennai/

[33] https://www.guru99.com/introduction-to-selenium.html

[34] https://www.quora.com/Why-are-Selenium-testing-tools-used-nowadays-in-most-of-the-industry

[35] https://www.tutorialspoint.com/jira/jira_overview.htm

[36] https://confluence.atlassian.com/jirasoftwarecloud/jira-software-overview-779293724.html

[37] http://www.devopsdigest.com/devops-advantages-1

[38] https://en.wikipedia.org/wiki/Java_Code_Coverage_Tools

[39] https://www.geeksforgeeks.org/introduction-apache-maven-build-automation-tool-java-projects/

Images References

[1] https://dzone.com/articles/what-is-cicd

[2] https://www.eclemma.org/jacoco/