p-ISSN : 2443-2210
e-ISSN : 2443-2229
Jurnal Teknik Informatika dan Sistem Informasi
Volume 7 Nomor 1 April 2021

Continuous Integration and Continuous Delivery Platform Development of Software Engineering and Software Project Management in Higher Education

http://dx.doi.org/10.28932/jutisi.v7i1.3254
Riwayat Artikel
Received: 5 Januari 2021 | *Final Revision*: 15 Maret 2021 | *Accepted*: 24 Maret 2021

Sendy Ferdian[#1], Tjatur Kandaga[#2], Andreas Widjaja[✉#3], Hapnes Toba[#4],
Ronaldo Joshua[#5], Julio Narabel[#6]

Faculty of Information Technology, Universitas Kristen Maranatha,
Bandung, Indonesia

[1]sendy.fs@it.maranatha.edu
[2]tjatur.kandaga@it.maranatha.edu
[3]andreas.widjaja@it.maranatha.edu
[4]hapnestoba@it.maranatha.edu
[5]ronaldojoshua97@gmail.com
[6]juliolee90@gmail.com

Abstract — **We present a report of the development phase of a platform that aims to enhance the efficiency of software project management in higher education. The platform accommodates a strategy known as Continuous Integration and Continuous Delivery (CI/CD). The phase consists of several stages, followed by testing of the system and its deployment. For starters, the CI/CD platform will be deployed for software projects of students in the Faculty of Information Technology, Universitas Kristen Maranatha. The goal of this paper is to show a design of an effective platform for continuous integration and continuous delivery pipeline to accommodate source code compilation, code analysis, code execution, until its deployment, all in a fully automated fashion.**

Keywords— **continuous integration; continuous delivery; software project; project management**

I. Introduction

There is a main trend in software engineering [1] and project management where the development of application consists of phases from writing codes, building and integrating the application, alpha and beta testing, debugging, troubleshooting, managing configuration, setting up runtime environment, and finally deploying the applications [2]. Those phases are repetitive tasks, time-consuming and complex processes. The applications are likely to be very diverse and composed of various tools, commercially or open-source technologies, therefore managing application delivery lifecycle is difficult because of those complexities. Dealing with those daunting and complex tasks, one solution is using the DevOps [3] [4] approach.

The relatively new emerging DevOps is a culture where development, testing, and operations teams join together collaborating to deliver outcome results in a continuous and effective way [2] [5]. This way DevOps is a better approach for organizations to enter the marketplace in the middle of fierce competition and enabling them to build a better quality of applications; hence allowing to increase benefits and bringing customer expectation and satisfaction. It makes the organizations see opportunities and reducing the time necessary for customer feedback of a significant fix or new features in development. The DevOps ultimate goal is to reduce the development time from the initial concept until the end result. DevOps culture can significantly result in faster time to deployment and a much efficient outcome, and developers can perform continuous integration of their code and maintain the integrity of the code at all stages of development.

Here we are interested in the Continuous Integration [6] and Continuous Delivery [7] [8] combined practice approach, abbreviated as CI/CD, which is one of the

p-ISSN : 2443-2210
e-ISSN : 2443-2229

DevOps approaches, which we will apply to higher education software engineering and student software project management system. In higher education, particularly in information technology, information systems, and computer science education programs, where every student works on a software development project, management of those student projects is necessary to ensure their efficiency and effectiveness. For this reason, it is better to apply CI/CD approach to achieve a better software project management system.

II. Methodology of CI/CD Research

In this section and the following subsection, we describe CI/CD methodology using strategy approach in brief as shown in Figure 1.

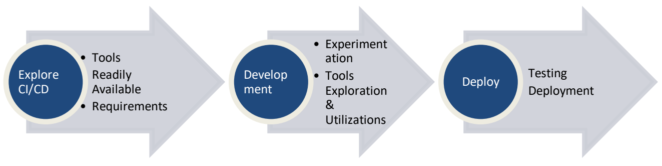


Figure 1. Methodology of CI/CD (strategy)

Here our methodology in Figure 1 works as the strategy described of the following: Firstly, CI/CD is explored by doing a survey of tools available, where among those tools there are readily available to be used and matching their requirements. Secondly, in the development stage, experimentations using selected tools are performed based on tools exploration and utilization. Lastly, testing of the system is performed leading to the deployment phase.

A. Basics Concept

According to Fowler [9] Continuous Integration is defined as a practice of a software development with frequent, at least daily, integration of works done by members of a team, which leads to daily multiple integrations. Every team is possible to perform testing of customer requirements with development. Continuous Delivery is defined as a discipline of a software development where a deployment pipeline is built in a way that at any time it can release the resulting software [10]. It assures end users whatever is being produced will meet their needs, hence it shows the customer, whenever they want, what developers are delivering. CI/CD strategy is also known as Continuous Deployment, see Zubin [11], Heller [12], Atlassian [13] and Manturewicz [14].

Related works and analyses of Continuous Integration and Continuous Delivery practices have been done extensively by some studies, for example, Lehtonen et al. [15], Humble et al. [7], Ståhl et al. [6] and Chen [8]. Those studies conclude that such practices lead faster, effective, and more efficient in the software development chain, and yet maintaining software engineering basic principles, as in [1]. An automation practice of CI/CD has been performed by Arachchi and Perera [16] which includes automation for agile software project management, while Krusche and Alperowitz [17] implement the continuous delivery practice in multi-customer project courses.

B. CI/CD Tools

Continuous Integration (CI) system [6] [9] is a regulator that involves many tools and combine them to achieve the goal of automating the software development process. There are some tools, for example:

1) *Source code control / version control* [18]: is a tool to centrally control source code integration from all programmers involved. It also functions as a backup of the source code, regulators when mismatches occur between codes sent by different programmers, and setting the source code version automatically. Examples of version control tools: CVS, SubVersion, Git, Mercurial, etc.
2) *Server or cloud version control service provider* [19]: is a service on the internet that provides a server or cloud environment for source code management. Examples are: Github, Gitlab, Bitbucket etc.
3) *Source code builder* [20]: produces the destination code associated with the program language and framework used. For example: Java compiler, C compiler, C++ compiler, C# compiler, Python interpreter, PHP interpreter, Make, Ant, Gradle, Maven, etc.
4) *Software testing tools* [21]: which can be further divided into testing an atomic function (unit testing), testing the incorporation of various software components (integration test), and testing the software interface (user interface / user acceptance test). Examples are: Junit, NUnit, TestNG, Selenium, Mockito, etc.
5) *Software deployment / installation tools*: are used to build final installation packages to be delivered to end users.

C. Readily Available Tools

In order to choose the most suitable tools for our projects, we analyse and evaluate some of CI/CD tools readily available on the market, as follows:

1) *TeamCity* [22]: is a Java-based CI/CD Server and managed by JetBrains. It was released on October 2, 2006. TeamCity can be accessed as a commercial server and licensed under an exclusive license. Free license gets access up to 100 build configurations and 3 Build Agents. It is used by developers and professionals in the field for DevOps [23] [24]. The advantage of TeamCity is that the server has a stable version because it is managed by JetBrains, which is

not Open Source. The number of TeamCity plugins is less than those of the open-source CI/CD server.

2) *Jenkins* [25]: is a leading open-source CI/CD server which provides hundreds of plugins to support development, deploying, and automating any IT projects. It supports versioning tools that are widely used by developers, professionals [26] [27] [28] and scientists [29]. Because Jenkins is open source, anyone can patch the main code from the server so that it is more powerful, giving it an advantage point. On the other hand, since it is an open source, it is accessible to many people, therefore the structure of Jenkins grows more complex and not so easy to configure.
3) *CircleCI* [30]: is a cloud-based CI/CD tool. CircleCI is also possible to be installed on a private server. It supports programming languages: Ruby, Python, Node, Java, PHP, RoR, DJ and JavaScript. Since it is cloud based, its user does not require physical hardware. However, it may cause problems when its cloud has problems.
4) *Gitlab CI* [31]: is a service part of GitLab that builds and tests software every time a developer pushes code into an application. *GitLab CD*: is a software service that places changes to every code in production that results in a daily production deployment. The advantages of *Gitlab CI-CD* are collaboration techniques that allow project team members to integrate their work every day. However, Gitlab CI-CD disadvantage is that it can cause problems if there are problems with the Gitlab cloud or changes to the Gitlab Server.
5) *Azure DevOps Server* (*formerly Team Foundation Server*) [32]: is a Microsoft product that provides source code management, reporting, requirements management, project management (Agile and WaterFall models), automated builds, lab management, testing and release management capabilities. Azure DevOps Server covers the entire application life cycle and DevOps capabilities. Azure DevOps Server can be used as a back-end to various integrated development environments (IDEs) but is designed for Microsoft Visual Studio and Eclipse on all platforms. The advantage of Azure DevOps Server is that it is very suitable for the project team that uses Microsoft technology. The disadvantage of Azure DevOps Server is that it requires a minimum license to purchase in Visual Studio to get a server license and is needed to purchase Azure DevOps so that the DevOps feature can run.
6) *Travis CI* [33]: is a CI/CD service that is equally managed by JetBrains, just like TeamCity. Travis CI itself utilizes the full potential of all TeamCity features. Strengths and Weaknesses of Travis CI are the same as TeamCity, it's just more widely used for open-source projects.
7) *GoCD* [34]: is an open-source tool used in software development to help teams and organizations automate the delivery of source code. GoCD supports automation throughout the build-test-release process from check-in code to deployment which helps to continue to produce stable software in relatively short cycles and ensure that the software will be released reliably at any time. GoCD is released under Apache License 2. The advantage of GoCD is that it supports several version controls tools, namely: Git, Mercurial, Subversion, Perforce and Team Foundation Server. The disadvantage of GoCD is that it requires a fairly complex configuration to be able to support other plugins.
8) *Bamboo* [35]: is a CI/CD tool developed by Atlassian. Although initially available as a cloud-based service, the cloud version was discontinued at the end of January 2017. Bamboo's strength is that it has a Git branching workflow and integrated placement projects and has built-in integration with other Atlassian software such as Jira, Confluence, Bitbucket, and HipChat. The weakness of Bamboo is that not all services are free and not open-source software, so the time needed to fix the problem can considerably be longer.
9) *Codeship* [36]: is a Continuous Delivery platform to hosts which help releasing software quickly, automatically and in frequency of several times a day. Codeship can shorten the development cycle so as to reduce the risk of bugs and increase innovation so as to help software companies to develop better and faster products by maintaining the testing and release process. Codeship automates the spread of the software and all necessary tasks involved with it. The advantage of Codeship is that it is managed by a company so the version can be stable. The weakness of Codeship is that every service provided has provisions that can always change according to what is offered.
10) *BuddyBuild* [37]: is a web-based and self-hosted CI/CD tool for Git developers that can be used to build, test and use websites and applications with code from GitHub, Bitbucket, and GitLab. BuddyBuild employs a Docker container with language and frameworks already installed to build, monitor actions and notices with DevOps. The advantage of BuddyBuild is that it has been integrated with GitHub, Bitbucket, and GitLab but has weakness: it is not easy to integrate with other plugins.
11) *AWS CodePipeline* [38]: a service from Amazon, is a Continuous Delivery service that helps developers automate the release flow for fast and reliable application and infrastructure updates. CodePipeline automates the creation, testing, and implementation phases of the release process every time there is a

code change, based on the release model that developers specify so that it allows the developers to send features and updates quickly and reliably. Developers can easily integrate AWS CodePipeline with third party services such as GitHub or with their own custom plugin. With AWS CodePipeline, developers only pay for what they use. There are no upfront fees or long-term agreements. So, the advantage of AWS CodePipeline is the ease of CI/CD services because it is managed by Amazon but has a disadvantage: the CI/CD services is not free.

D. CI/CD requirements

The requirement of CI/CD pipeline are described as follows:

a. Long term maintainability must be ensured;
b. Idempotence principles must be held, that is, should produce the same result if ran once or multiple times;
c. Builds, tests and deployment should be done in automated fashion; and
d. The pipeline must be orchestrated.

III. Development

The development of the platform consists of experimentation, exploration and utilization phases is described in the following:

A. Experimentation Stage

After analysing and comparing the features of the various tools available, we decided to choose Jenkins [39] and TeamCity [40].

We choose Jenkins because the license is free, the source code is available freely on the internet (open source), and can be integrated with many other devices. At present Jenkins is also de facto the most widely used CI/CD tool in the world. The reason we choose TeamCity because the installation process and settings are easier than other CI/CD tools, making it suitable for use as a first step in learning the CI/CD strategy. TeamCity licenses are available in both paid service and free models. The free model licenses are suitable for small teams that don't need too many build agents. A build agent is an environment where builds or jobs from a CI pipeline are executed.

B. Tools Exploration and Utilization

We explore CI/CD tools, Jenkins and TeamCity to experiments their performances and ease of usability. In the exploration, it is found that both tools are the most suitable for higher education student projects in informatics engineering study program.

We explore Jenkins and the results of our exploration: Jenkins is easy to install on various operating systems, Linux with various distributions and Windows 10; Jenkins is also available for Docker; Jenkins has weekly releases and provides long-term support; Jenkins is simple and has easy user interface; Jenkins has many third-party plugins and it suggests plugins necessary at the time of installation; Jenkins has a good job configuration page and easy to navigate; Jenkins is easy to configure and it has customizable user interface; and it has notification support of its build status. Those results of the exploration lead to our decision to select Jenkins among other tools.

Exploration of TeamCity has results: TeamCity has many features; It has great support for Java, .NET, Android, and iOS; It has many very useful plugins [41] written by JetBrains and its community; TeamCity supports almost every major version control system (VCS) such as Git and Azure DevOps Server; TeamCity has a good dashboard; It provides historical data to track builds; TeamCity provides integration with IntelliJ IDEA and Visual Studio; and it has an easy to navigate user interface. Those results are our main reason to select TeamCity among other tools.

C. System Architecture

The architecture of the platform Jenkins and TeamCity are shown in Figure 2 and Figure 3, respectively.

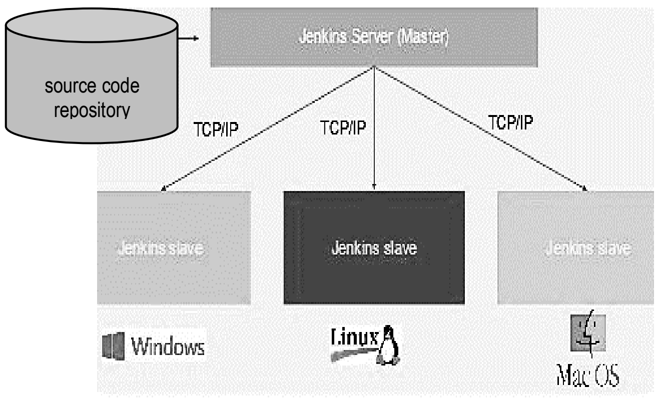


Figure 2. Jenkins architecture [39]

The architecture of Jenkins applies master-slave scheme where the server processes codes from the repository, which can be locally or remotely placed. The Jenkins slaves connected to the master and may have different operating systems.

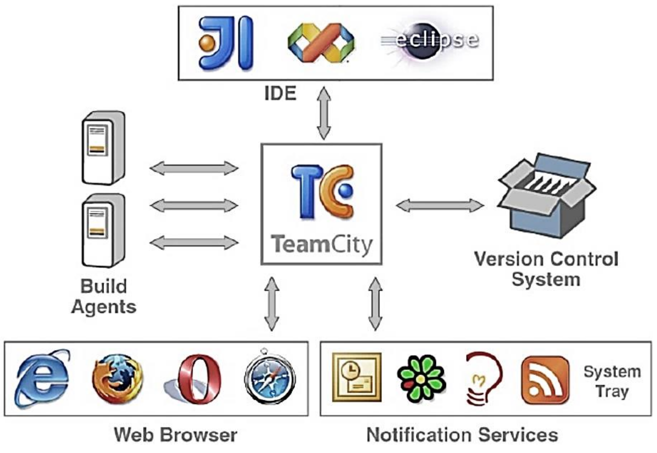


Figure 3. TeanCity architecture [40]

For TeamCity, the architecture [40] allows centralised scheme, therefore the users may use IDEs of their choices, in which the source code will be built by build agents, together with version control system, hence gives the user convenience thus accelerating the development.

IV. Testing and Deployment

The testing and deployment of the platform is including setup and testing phases will be described as follows:

A. System Setup

We install the system in a quite powerful hardware, utilizing a relatively fast and state-of-the-art processor, namely AMD Ryzen 9 3900X running at 3.8 GHz (4.6 GHz Max Boost), which is built using 7-nm lithography technology with 12 computing cores and capable to run 24 simultaneous threads, running in a fully 64-bit instructions set. Large amount of 64 GB high-clocked rate random access memory (RAM) is installed to empower the system to be able to handle multiple requests simultaneously. A very fast M.2 PCIe NVMe, 1 TB large capacity, solid state drive (SSD) is installed to eliminate I/O bottleneck caused by very high data streams. With this high specification hardware, we hope that the system will perform smoothly, even under high load pressure.

B. Deployment

Deployment of projects using Jenkins can be described as in pipeline process shown in Figure 4. The process includes CI/CD stages: committing code submitted, build, test, stage and deploy. With Jenkins there is also source code control system and including results reporting for later references if needed.

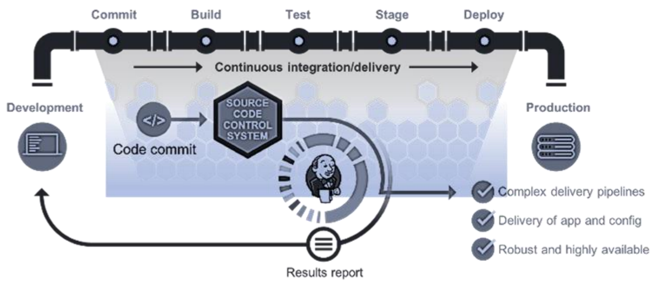

Commit
Build
Test
Stage
Deploy
Continuous integration/delivery
Development
Production
SOURCE CODE CONTROL SYSTEM
Code commit
Complex delivery pipelines
Delivery of app and config
Robust and highly available
Results report

Figure 4. Deployment pipeline using Jenkins [39] [42].

Pipeline of deployment using TeamCity is described as in Figure 5. The process includes delivery, version control, built, testing of unit, automated acceptance test, user acceptance tests and release. At every stage there is a feedback. The testing stages assure the quality of the resulting product.

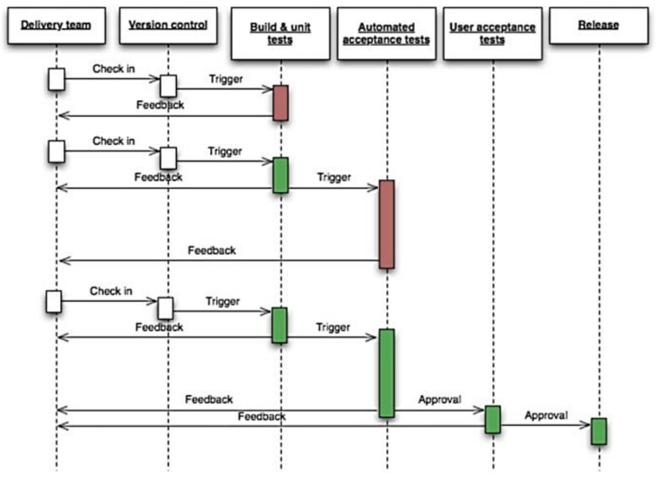


Figure 5. Deployment pipeline using TeamCity [23] [40]

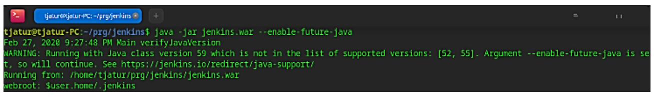
C. Testing with Various Cases Experimentation

We utilize Jenkins and TeamCity to experiments their performances and ease of usability for various cases: including many source codes in various programming languages, with repository stored locally or remotely.

Here we present an example of CI/CD experimentation using Jenkins: After installed on the system, Jenkins can be started by invoking it on the command line terminal, as shown in Figure 6, and when successful, Jenkins is fully up and running (see Figure 7). Initial screen when Jenkins is running is a dashboard view (see Figure 8), here the installation is successful and Jenkins is ready. When a project is submitted, it is appeared as in Figure 9. In this example we connect the GitHub repository with Jenkins, where the repository is shown in Figure 10. Pipeline instruction for Jenkins is placed in the repository with name "Jenkinsfile" with content is shown in Figure 11. Our experiment of pipelining the built using Jenkins is successful, with a project status is notified by Jenkins in the dashboard shown in Figure 12. Jenkins pipeline branch main

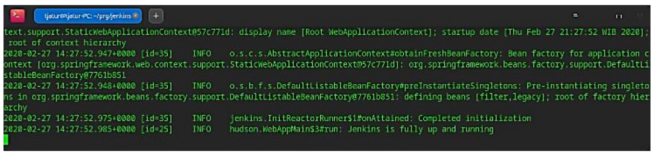
p-ISSN : 2443-2210
e-ISSN : 2443-2229

status with run number and status can be seen in Figure 13. Log Checkout from GitHub version control was done as in Figure 14. Figure 15 shows a Linux command to list files in a directory, the command is executed at shell script, followed by compilation of Java code (Figure 16). The compiled code is then successfully executed (Figure 17). The console output in Jenkins is shown in Figure 18. Actual console output of the whole process is shown in Figure 19 and Figure 20.



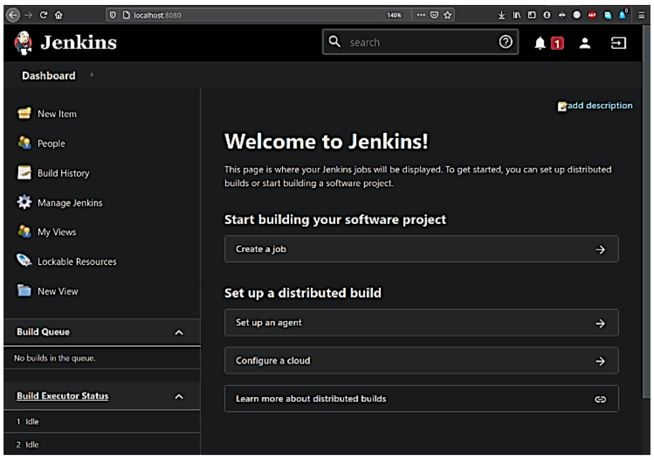
```
tjatur@tjatur-PC:~/prg/jenkins$ java -jar jenkins.war --enable-future-java
Feb 27, 2020 9:27:48 PM Main verifyJavaVersion
WARNING: Running with Java class version 59 which is not in the list of supported versions: [52, 55]. Argument --enable-future-java is se
t, so will continue. See https://jenkins.io/redirect/java-support/
Running from: /home/tjatur/prg/jenkins/jenkins.war
webroot: $user.home/.jenkins
```

Figure 6. Invoke Jenkins on the command line terminal



```
text.support.StaticWebApplicationContext@57c771d: display name [Root WebApplicationContext]; startup date [Thu Feb 27 21:27:52 WIB 2020];
 root of context hierarchy
2020-02-27 14:27:52.947+0000 [id=35]    INFO    o.s.c.s.AbstractApplicationContext#obtainFreshBeanFactory: Bean factory for application c
ontext [org.springframework.web.context.support.StaticWebApplicationContext@57c771d]: org.springframework.beans.factory.support.DefaultLi
stableBeanFactory@7761b851
2020-02-27 14:27:52.948+0000 [id=35]    INFO    o.s.b.f.s.DefaultListableBeanFactory#preInstantiateSingletons: Pre-instantiating singleto
ns in org.springframework.beans.factory.support.DefaultListableBeanFactory@7761b851: defining beans [filter,legacy]; root of factory hier
archy
2020-02-27 14:27:52.975+0000 [id=35]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
2020-02-27 14:27:52.985+0000 [id=25]    INFO    hudson.WebAppMain$3#run: Jenkins is fully up and running
```

Figure 7. Jenkins is fully up and running



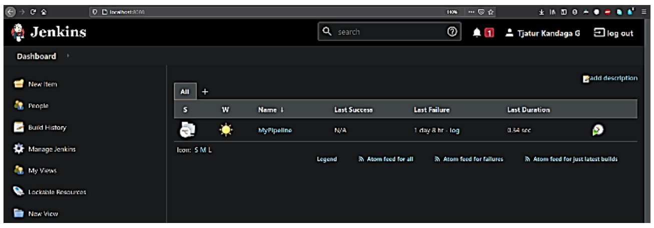
Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

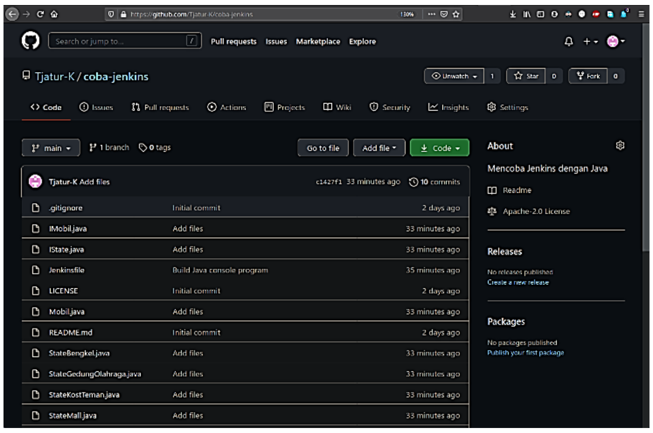
Set up a distributed build

Figure 8. Jenkins initial dashboard view



S	W	Name ↓	Last Success	Last Failure	Last Duration
		MyPipeline	N/A	1 day 8 hr - log	0.34 sec

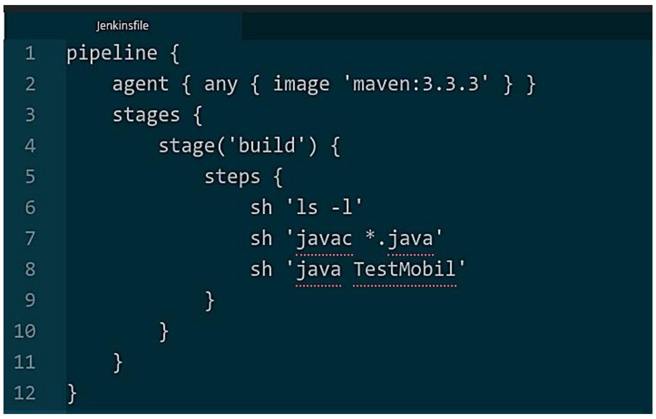
Figure 9. A project "MyPipeline" is added to Jenkins



Tjatur-K Add files c1427f1 33 minutes ago 10 commits

.gitignore	Initial commit	2 days ago
IMobil.java	Add files	33 minutes ago
IState.java	Add files	33 minutes ago
Jenkinsfile	Build Java console program	35 minutes ago
LICENSE	Initial commit	2 days ago
Mobil.java	Add files	33 minutes ago
README.md	Initial commit	2 days ago
StateBengkel.java	Add files	33 minutes ago
StateGedungOlahraga.java	Add files	33 minutes ago
StateKostTeman.java	Add files	33 minutes ago
StateMall.java	Add files	33 minutes ago

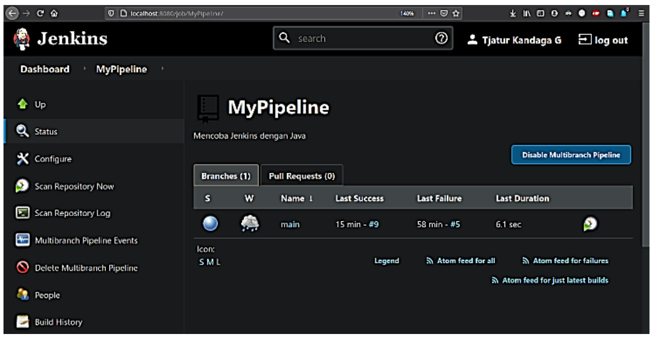
Figure 10. A repository in GitHub



Jenkinsfile

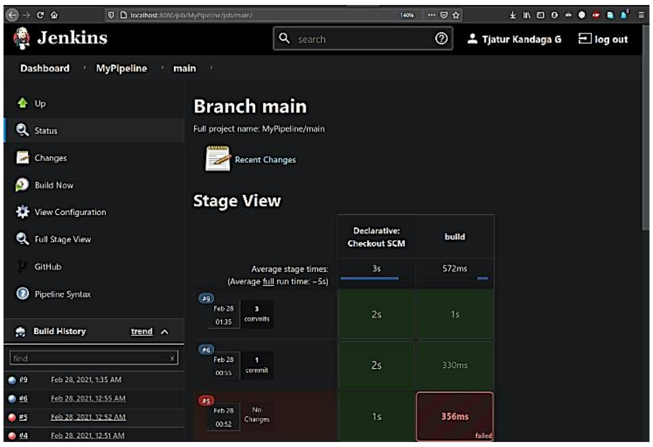
```
pipeline {
    agent { any { image 'maven:3.3.3' } }
    stages {
        stage('build') {
            steps {
                sh 'ls -l'
                sh 'javac *.java'
                sh 'java TestMobil'
            }
        }
    }
}
```

Figure 11. Content of "Jenkinsfile" in the repository.



S	W	Name ↓	Last Success	Last Failure	Last Duration
		main	15 min - #9	58 min - #5	6.1 sec

Figure 12. Pipeline status



Branch main

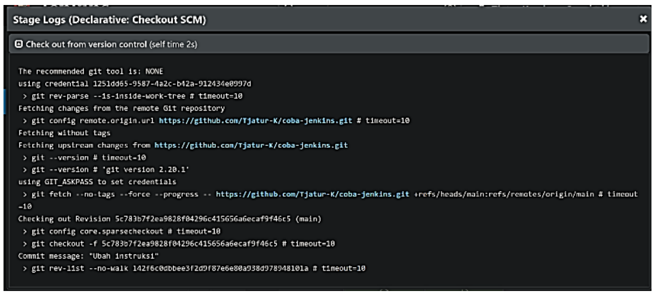
Full project name: MyPipeline/main

Stage View

	Declarative: Checkout SCM	build
Average stage times: (Average full run time: ~5s)	3s	572ms
#9 Feb 28 01:35 3 commits	2s	1s
#6 Feb 28 00:55 1 commit	2s	330ms
#5 Feb 28 00:52 No Changes	1s	356ms failed

```
#9  Feb 28, 2021, 1:35 AM
#6  Feb 28, 2021, 12:55 AM
#5  Feb 28, 2021, 12:52 AM
#4  Feb 28, 2021, 12:51 AM
```

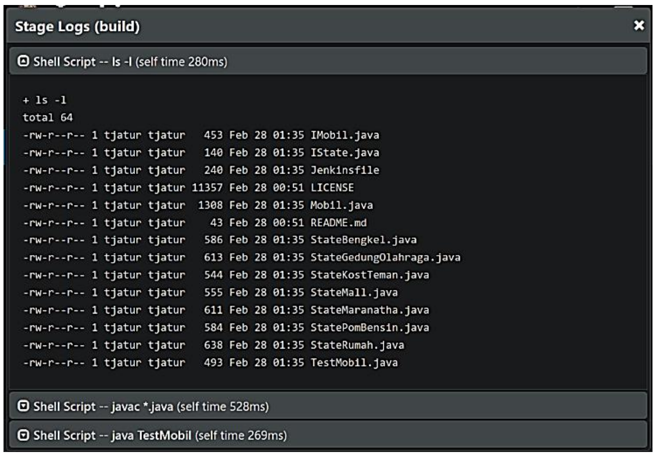
Figure 13. Jenkins pipeline branch main status


Stage Logs (Declarative: Checkout SCM)

Check out from version control (self time 2s)

```
The recommended git tool is: NONE
using credential 1251dd65-9587-4a2c-b42a-912434e0997d
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/Tjatur-K/coba-jenkins.git # timeout=10
Fetching without tags
Fetching upstream changes from https://github.com/Tjatur-K/coba-jenkins.git
 > git --version # timeout=10
 > git --version # 'git version 2.20.1'
using GIT_ASKPASS to set credentials
 > git fetch --no-tags --force --progress -- https://github.com/Tjatur-K/coba-jenkins.git +refs/heads/main:refs/remotes/origin/main # timeout
=10
Checking out Revision 5c783b7f2ea9828f04296c415656a6ecaf9f46c5 (main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 5c783b7f2ea9828f04296c415656a6ecaf9f46c5 # timeout=10
Commit message: "Ubah instruksi"
 > git rev-list --no-walk 142f6c0dbbee3f2d9f87e6e80a938d978948101a # timeout=10
```

Figure 14. Log checkout from Github version control


Stage Logs (build)

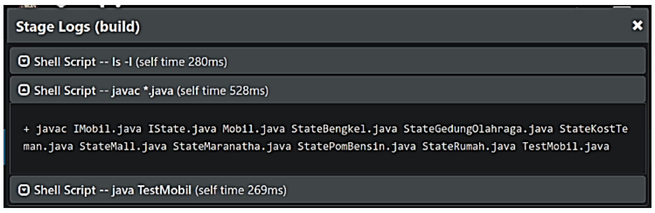
Shell Script -- ls -l (self time 280ms)

```
+ ls -l
total 64
-rw-r--r-- 1 tjatur tjatur   453 Feb 28 01:35 IMobil.java
-rw-r--r-- 1 tjatur tjatur   140 Feb 28 01:35 IState.java
-rw-r--r-- 1 tjatur tjatur   240 Feb 28 01:35 Jenkinsfile
-rw-r--r-- 1 tjatur tjatur 11357 Feb 28 00:51 LICENSE
-rw-r--r-- 1 tjatur tjatur  1308 Feb 28 01:35 Mobil.java
-rw-r--r-- 1 tjatur tjatur    43 Feb 28 00:51 README.md
-rw-r--r-- 1 tjatur tjatur   586 Feb 28 01:35 StateBengkel.java
-rw-r--r-- 1 tjatur tjatur   613 Feb 28 01:35 StateGedungOlahraga.java
-rw-r--r-- 1 tjatur tjatur   544 Feb 28 01:35 StateKostTeman.java
-rw-r--r-- 1 tjatur tjatur   555 Feb 28 01:35 StateMall.java
-rw-r--r-- 1 tjatur tjatur   611 Feb 28 01:35 StateMaranatha.java
-rw-r--r-- 1 tjatur tjatur   584 Feb 28 01:35 StatePomBensin.java
-rw-r--r-- 1 tjatur tjatur   638 Feb 28 01:35 StateRumah.java
-rw-r--r-- 1 tjatur tjatur   493 Feb 28 01:35 TestMobil.java
```

Shell Script -- javac *.java (self time 528ms)

Shell Script -- java TestMobil (self time 269ms)

Figure 15. Build output of instruction 1, which is listing directory contents


Stage Logs (build)

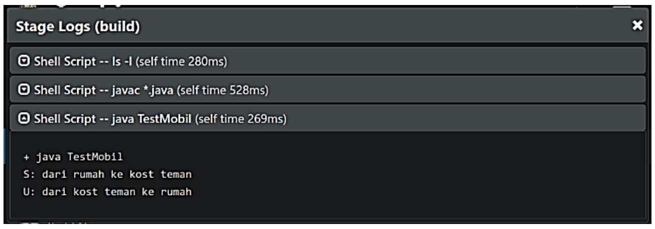
Shell Script -- ls -l (self time 280ms)

Shell Script -- javac *.java (self time 528ms)

```
+ javac IMobil.java IState.java Mobil.java StateBengkel.java StateGedungOlahraga.java StateKostTe
man.java StateMall.java StateMaranatha.java StatePomBensin.java StateRumah.java TestMobil.java
```

Shell Script -- java TestMobil (self time 269ms)

Figure 16. Build output of instruction 2, which is compiling Java code


Stage Logs (build)

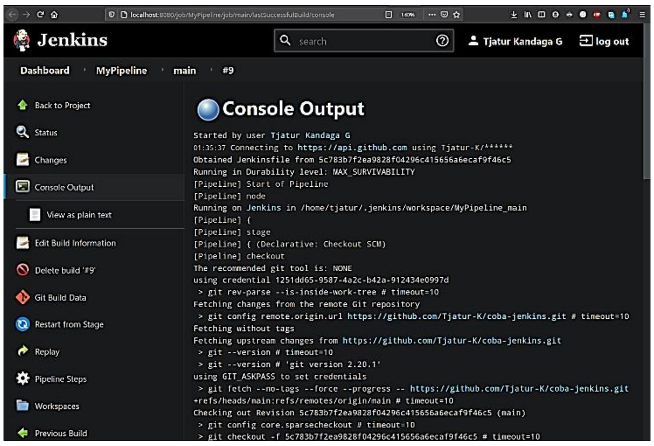
Shell Script -- ls -l (self time 280ms)

Shell Script -- javac *.java (self time 528ms)

Shell Script -- java TestMobil (self time 269ms)

```
+ java TestMobil
S: dari rumah ke kost teman
U: dari kost teman ke rumah
```

Figure 17. Build output for instruction 3, which is the execution of Java console program



Console Output

```
Started by user Tjatur Kandaga G
01:35:37 Connecting to https://api.github.com using Tjatur-K/******
Obtained Jenkinsfile from 5c783b7f2ea9828f04296c415656a6ecaf9f46c5
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /home/tjatur/.jenkins/workspace/MyPipeline_main
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
The recommended git tool is: NONE
using credential 1251dd65-9587-4a2c-b42a-912434e0997d
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/Tjatur-K/coba-jenkins.git # timeout=10
Fetching without tags
Fetching upstream changes from https://github.com/Tjatur-K/coba-jenkins.git
 > git --version # timeout=10
 > git --version # 'git version 2.20.1'
using GIT_ASKPASS to set credentials
 > git fetch --no-tags --force --progress -- https://github.com/Tjatur-K/coba-jenkins.git
+refs/heads/main:refs/remotes/origin/main # timeout=10
Checking out Revision 5c783b7f2ea9828f04296c415656a6ecaf9f46c5 (main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 5c783b7f2ea9828f04296c415656a6ecaf9f46c5 # timeout=10
```

Figure 18. Jenkins console output

```
Started by user Tjatur Kandaga G
01:35:37 Connecting to https://api.github.com using
Tjatur-K/******
Obtained Jenkinsfile from
5c783b7f2ea9828f04296c415656a6ecaf9f46c5
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in
/home/tjatur/.jenkins/workspace/MyPipeline_main
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
The recommended git tool is: NONE
using credential 1251dd65-9587-4a2c-b42a-912434e0997d
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url
https://github.com/Tjatur-K/coba-jenkins.git #
timeout=10
```

Figure 19. Actual console output (part 1)

p-ISSN : 2443-2210
e-ISSN : 2443-2229

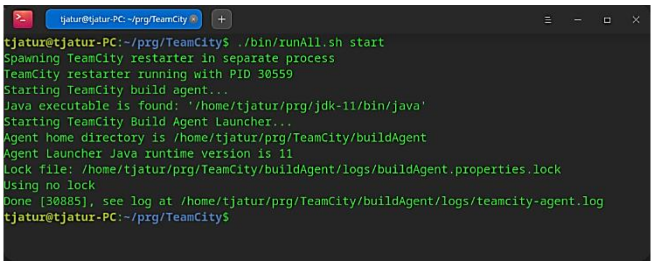
```
Fetching without tags
Fetching upstream changes from
https://github.com/Tjatur-K/coba-jenkins.git
 > git --version # timeout=10
 > git --version # 'git version 2.20.1'
using GIT_ASKPASS to set credentials
 > git fetch --no-tags --force --progress --
https://github.com/Tjatur-K/coba-jenkins.git
+refs/heads/main:refs/remotes/origin/main #
timeout=10
Checking out Revision
5c783b7f2ea9828f04296c415656a6ecaf9f46c5 (main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f
5c783b7f2ea9828f04296c415656a6ecaf9f46c5 # timeout=10
Commit message: "Ubah instruksi"
 > git rev-list --no-walk
142f6c0dbbee3f2d9f87e6e80a938d978948101a # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (build)
[Pipeline] sh+ ls -l
total 64
-rw-r--r-- 1 tjatur tjatur   453 Feb 28 01:35
IMobil.java
-rw-r--r-- 1 tjatur tjatur   140 Feb 28 01:35
IState.java
-rw-r--r-- 1 tjatur tjatur   240 Feb 28 01:35
Jenkinsfile
-rw-r--r-- 1 tjatur tjatur 11357 Feb 28 00:51 LICENSE
-rw-r--r-- 1 tjatur tjatur  1308 Feb 28 01:35
Mobil.java
-rw-r--r-- 1 tjatur tjatur    43 Feb 28 00:51
README.md
-rw-r--r-- 1 tjatur tjatur   586 Feb 28 01:35
StateBengkel.java
-rw-r--r-- 1 tjatur tjatur   613 Feb 28 01:35
StateGedungOlahraga.java
-rw-r--r-- 1 tjatur tjatur   544 Feb 28 01:35
StateKostTeman.java
-rw-r--r-- 1 tjatur tjatur   555 Feb 28 01:35
StateMall.java
-rw-r--r-- 1 tjatur tjatur   611 Feb 28 01:35
StateMaranatha.java
-rw-r--r-- 1 tjatur tjatur   584 Feb 28 01:35
StatePomBensin.java
-rw-r--r-- 1 tjatur tjatur   638 Feb 28 01:35
StateRumah.java
-rw-r--r-- 1 tjatur tjatur   493 Feb 28 01:35
TestMobil.java
[Pipeline] sh
+ javac IMobil.java IState.java Mobil.java
StateBengkel.java StateGedungOlahraga.java
StateKostTeman.java StateMall.java
StateMaranatha.java StatePomBensin.java
StateRumah.java TestMobil.java
[Pipeline] sh
+ java TestMobil
S: dari rumah ke kost teman
U: dari kost teman ke rumah
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Could not update commit status, please check if your
scan credentials belong to a member of the
organization or a collaborator of the repository and
repo:status scope is selected

GitHub has been notified of this commit’s build
result

Finished: SUCCESS
```

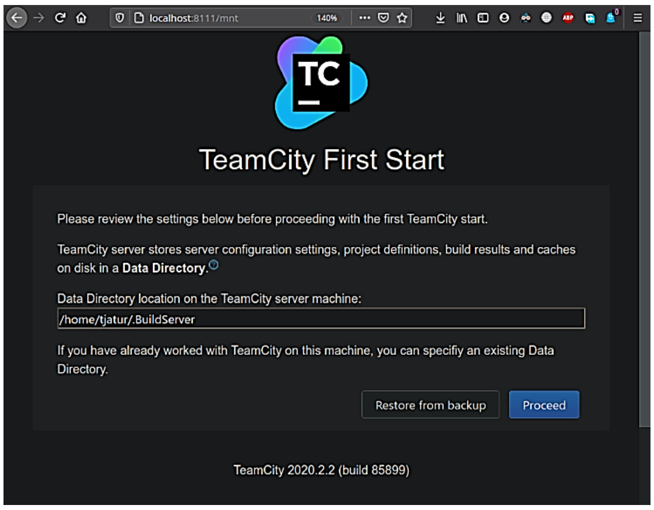
Figure 20. Actual console output (part 2)

Here is a presentation of an example of TeamCity experimentation: Firstly, TeamCity is invoked on the command line terminal, see Figure 21, when successful it shows start screen (Figure 22). In Figure 23 TeamCity shows the screen to connect to a database. In the next Figure 24, a TeamCity administrator account is created.



```
tjatur@tjatur-PC:~/prg/TeamCity$ ./bin/runAll.sh start
Spawning TeamCity restarter in separate process
TeamCity restarter running with PID 30559
Starting TeamCity build agent...
Java executable is found: '/home/tjatur/prg/jdk-11/bin/java'
Starting TeamCity Build Agent Launcher...
Agent home directory is /home/tjatur/prg/TeamCity/buildAgent
Agent Launcher Java runtime version is 11
Lock file: /home/tjatur/prg/TeamCity/buildAgent/logs/buildAgent.properties.lock
Using no lock
Done [30885], see log at /home/tjatur/prg/TeamCity/buildAgent/logs/teamcity-agent.log
tjatur@tjatur-PC:~/prg/TeamCity$
```

Figure 21. Invoke TeamCity via command line terminal



TeamCity First Start

Please review the settings below before proceeding with the first TeamCity start.

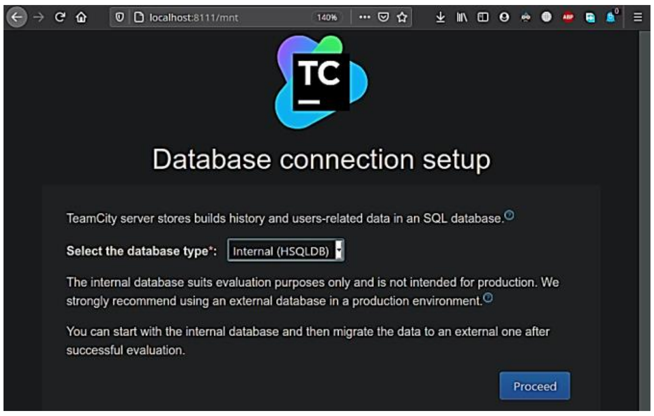
TeamCity server stores server configuration settings, project definitions, build results and caches on disk in a **Data Directory**.

Data Directory location on the TeamCity server machine: /home/tjatur/.BuildServer

If you have already worked with TeamCity on this machine, you can specifiy an existing Data Directory.

TeamCity 2020.2.2 (build 85899)

Figure 22. TeamCity start screen



Database connection setup

TeamCity server stores builds history and users-related data in an SQL database.

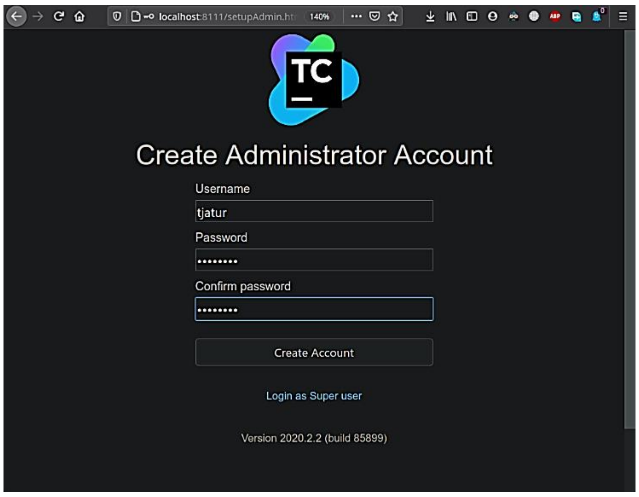
Select the database type*: Internal (HSQLDB)

The internal database suits evaluation purposes only and is not intended for production. We strongly recommend using an external database in a production environment.

You can start with the internal database and then migrate the data to an external one after successful evaluation.

Figure 23. TeamCity database connection setup screen

When creating account is successful, then TeamCity is ready, as shown in Figure 25. A GitHub repository is used in this experiment, see Figure 26. Next step is to create project on TeamCity, see Figure 27, then setup build configuration properly on general setting (Figure 28). Version control system (VCS) setting is also configured, see Figure 29. In Figure 30, we define the sequence of build steps to be executed. When everything is set, the build step sequence is executed, and Figure 31 shows the result of successful execution. Detailed statistics of the successful build is shown in Figure 32, and the overview of the successful build process is shown in Figure 33. A complete detailed log of the build process is created along the way of the process, see Figure 34.



Create Administrator Account

Username: tjatur

Password: ••••••••

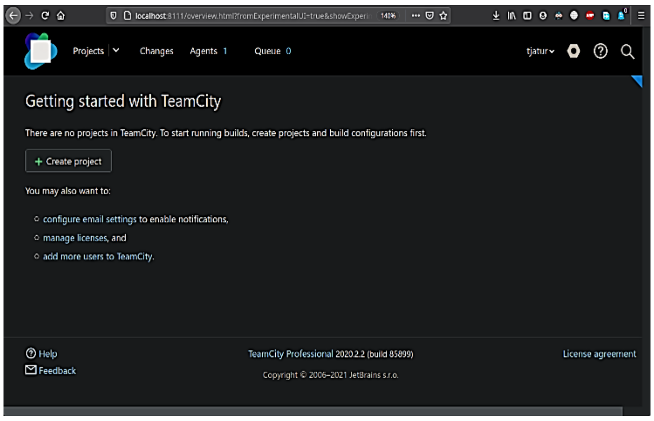
Confirm password: ••••••••

Create Account

Login as Super user

Version 2020.2.2 (build 85899)

Figure 24. TeamCity screen to create an administrator account


localhost:8111/overview.html?fromExperimentalUI=true&showExperi
Projects
Changes
Agents 1
Queue 0
tjatur

Getting started with TeamCity

There are no projects in TeamCity. To start running builds, create projects and build configurations first.

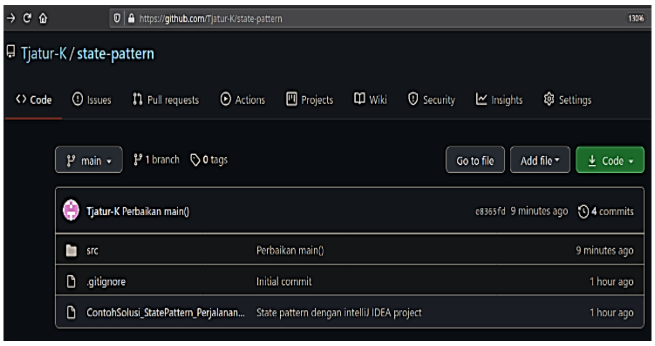
+ Create project

You may also want to:

- configure email settings to enable notifications,
- manage licenses, and
- add more users to TeamCity.

Help | TeamCity Professional 2020.2.2 (build 85899) | License agreement
Feedback | Copyright © 2006–2021 JetBrains s.r.o.

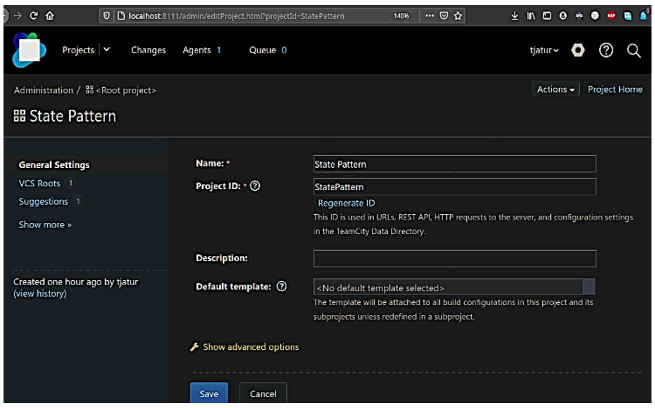
Figure 25. TeamCity is ready



Tjatur-K Perbaikan main() e8365fd 9 minutes ago 4 commits

src	Perbaikan main()	9 minutes ago
.gitignore	Initial commit	1 hour ago
ContohSolusi_StatePattern_Perjalanan...	State pattern dengan intelliJ IDEA project	1 hour ago

Figure 26. GitHub repository



State Pattern

Name: * State Pattern

Project ID: * StatePattern

Regenerate ID

This ID is used in URLs, REST API, HTTP requests to the server, and configuration settings in the TeamCity Data Directory.

Description:

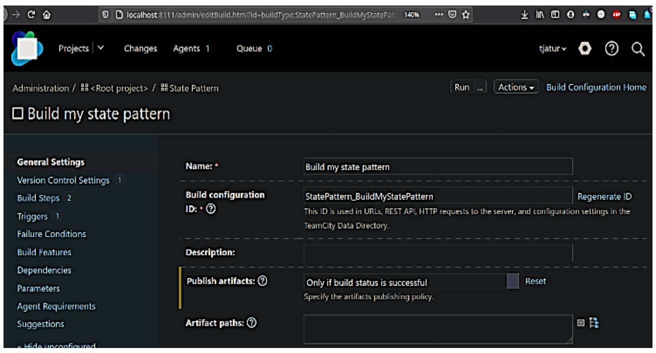
Default template: <No default template selected>

The template will be attached to all build configurations in this project and its subprojects unless redefined in a subproject.

Show advanced options

Created one hour ago by tjatur (view history)

Figure 27. TeamCity screen to create project



Build my state pattern

Name: * Build my state pattern

Build configuration ID: * StatePattern_BuildMyStatePattern

This ID is used in URLs, REST API, HTTP requests to the server, and configuration settings in the TeamCity Data Directory.

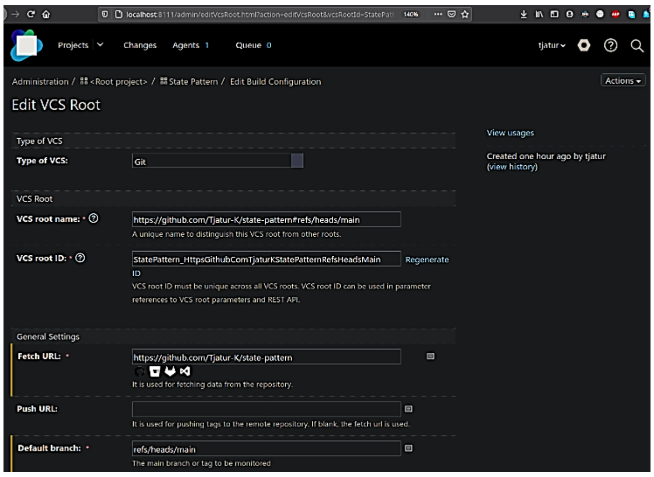
Description:

Publish artifacts: Only if build status is successful

Specify the artifacts publishing policy.

Artifact paths:

Figure 28. Build configuration setup on general setting


localhost:8111/admin/editVcsRoot.html?action=editVcsRoot&vcsRootId=StatePatt
Projects
Changes
Agents 1
Queue 0
tjatur
Actions
View usages

Administration / <Root project> / State Pattern / Edit Build Configuration

Edit VCS Root

Created one hour ago by tjatur (view history)

Type of VCS

Type of VCS: Git

VCS Root

VCS root name: https://github.com/Tjatur-K/state-pattern#refs/heads/main
A unique name to distinguish this VCS root from other roots.

VCS root ID: StatePattern_HttpsGithubComTjaturKStatePatternRefsHeadsMain Regenerate ID
VCS root ID must be unique across all VCS roots. VCS root ID can be used in parameter references to VCS root parameters and REST API.

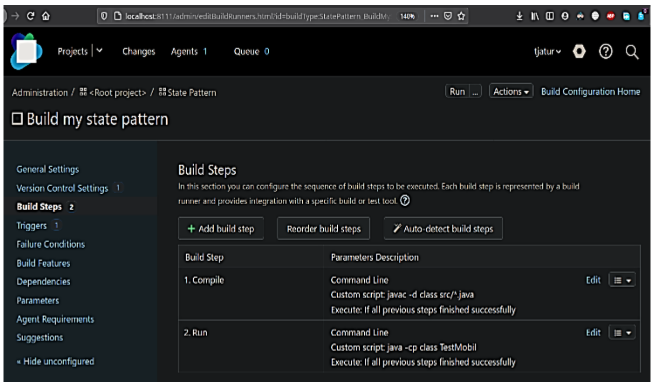
General Settings

Fetch URL: https://github.com/Tjatur-K/state-pattern
It is used for fetching data from the repository.

Push URL:
It is used for pushing tags to the remote repository. If blank, the fetch url is used.

Default branch: refs/heads/main
The main branch or tag to be monitored

Figure 29. TeamCity build configuration version control system (VCS)



Administration / <Root project> / State Pattern

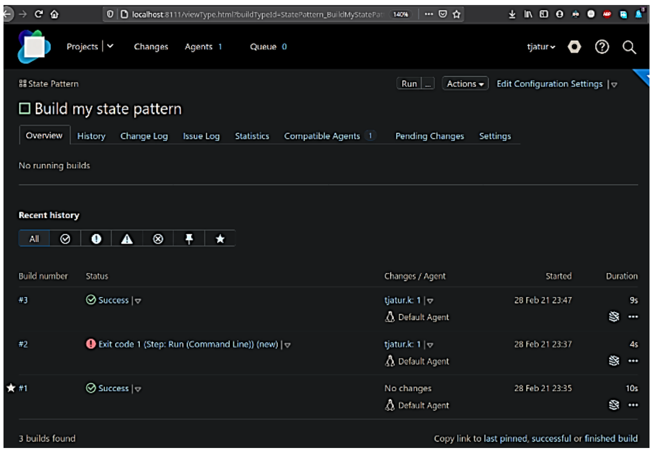
Build my state pattern

Build Steps

In this section you can configure the sequence of build steps to be executed. Each build step is represented by a build runner and provides integration with a specific build or test tool.

Build Step	Parameters Description
1. Compile	Command Line Custom script: javac -d class src/*.java Execute: If all previous steps finished successfully
2. Run	Command Line Custom script: java -cp class TestMobil Execute: If all previous steps finished successfully

Figure 30. TeamCity configuration of sequence build steps to be executed



State Pattern

Build my state pattern

No running builds

Recent history

Build number	Status	Changes / Agent	Started	Duration
#3	Success	tjatur.k: 1 Default Agent	28 Feb 21 23:47	9s
#2	Exit code 1 (Step: Run (Command Line)) (new)	tjatur.k: 1 Default Agent	28 Feb 21 23:37	4s
★ #1	Success	No changes Default Agent	28 Feb 21 23:35	10s

3 builds found

Copy link to last pinned, successful or finished build

Figure 31. TeamCity build step sequence execution result overview

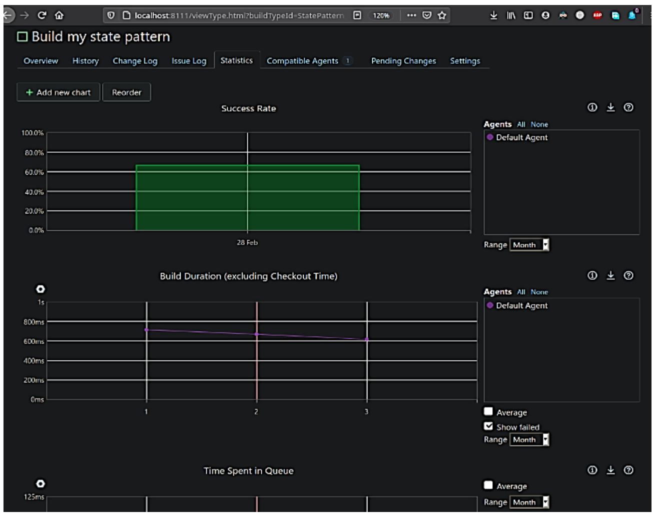
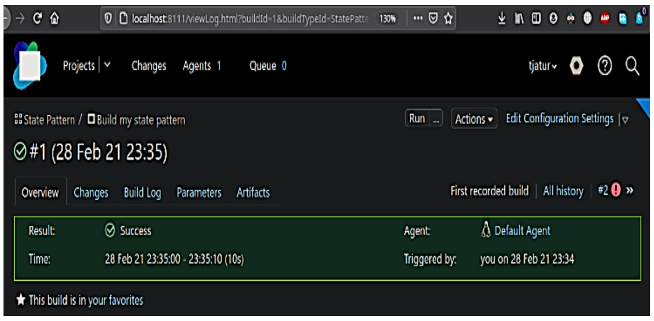


Figure 32. TeamCity shows statistics of build results



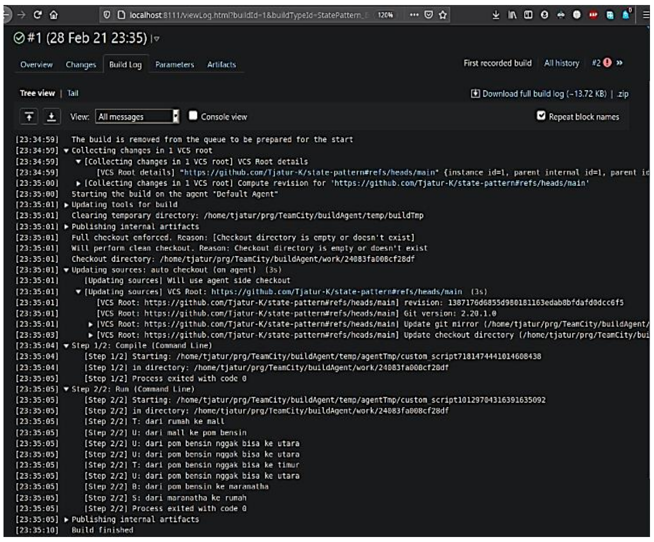
State Pattern / Build my state pattern

#1 (28 Feb 21 23:35)

Result:	Success	Agent:	Default Agent
Time:	28 Feb 21 23:35:00 - 23:35:10 (10s)	Triggered by:	you on 28 Feb 21 23:34

This build is in your favorites

Figure 33. Overview of TeamCity success build process



#1 (28 Feb 21 23:35)

```
[23:34:59]   The build is removed from the queue to be prepared for the start
[23:34:59] ▼ Collecting changes in 1 VCS root
[23:34:59]   ▼ [Collecting changes in 1 VCS root] VCS Root details
[23:34:59]       [VCS Root details] "https://github.com/Tjatur-K/state-pattern#refs/heads/main" {instance id=1, parent internal id=1, parent id
[23:35:00]   ▶ [Collecting changes in 1 VCS root] Compute revision for 'https://github.com/Tjatur-K/state-pattern#refs/heads/main'
[23:35:00]   Starting the build on the agent "Default Agent"
[23:35:01] ▶ Updating tools for build
[23:35:01]   Clearing temporary directory: /home/tjatur/prg/TeamCity/buildAgent/temp/buildTmp
[23:35:01] ▶ Publishing internal artifacts
[23:35:01]   Full checkout enforced. Reason: [Checkout directory is empty or doesn't exist]
[23:35:01]   Will perform clean checkout. Reason: Checkout directory is empty or doesn't exist
[23:35:01]   Checkout directory: /home/tjatur/prg/TeamCity/buildAgent/work/24083fa008cf28df
[23:35:01] ▼ Updating sources: auto checkout (on agent)  (3s)
[23:35:01]     [Updating sources] Will use agent side checkout
[23:35:01]   ▼ [Updating sources] VCS Root: https://github.com/Tjatur-K/state-pattern#refs/heads/main  (3s)
[23:35:01]       [VCS Root: https://github.com/Tjatur-K/state-pattern#refs/heads/main] revision: 1387176d6855d980181163edab8bfdafd0dcc6f5
[23:35:01]       [VCS Root: https://github.com/Tjatur-K/state-pattern#refs/heads/main] Git version: 2.20.1.0
[23:35:01]     ▶ [VCS Root: https://github.com/Tjatur-K/state-pattern#refs/heads/main] Update git mirror (/home/tjatur/prg/TeamCity/buildAgent/
[23:35:03]     ▶ [VCS Root: https://github.com/Tjatur-K/state-pattern#refs/heads/main] Update checkout directory (/home/tjatur/prg/TeamCity/bui
[23:35:04] ▼ Step 1/2: Compile (Command Line)
[23:35:04]     [Step 1/2] Starting: /home/tjatur/prg/TeamCity/buildAgent/temp/agentTmp/custom_script7181474441014608438
[23:35:04]     [Step 1/2] in directory: /home/tjatur/prg/TeamCity/buildAgent/work/24083fa008cf28df
[23:35:05]     [Step 1/2] Process exited with code 0
[23:35:05] ▼ Step 2/2: Run (Command Line)
[23:35:05]     [Step 2/2] Starting: /home/tjatur/prg/TeamCity/buildAgent/temp/agentTmp/custom_script10129704316391635092
[23:35:05]     [Step 2/2] in directory: /home/tjatur/prg/TeamCity/buildAgent/work/24083fa008cf28df
[23:35:05]     [Step 2/2] T: dari rumah ke mall
[23:35:05]     [Step 2/2] U: dari mall ke pom bensin
[23:35:05]     [Step 2/2] U: dari pom bensin nggak bisa ke utara
[23:35:05]     [Step 2/2] U: dari pom bensin nggak bisa ke utara
[23:35:05]     [Step 2/2] T: dari pom bensin nggak bisa ke timur
[23:35:05]     [Step 2/2] U: dari pom bensin nggak bisa ke utara
[23:35:05]     [Step 2/2] B: dari pom bensin ke maranatha
[23:35:05]     [Step 2/2] S: dari maranatha ke rumah
[23:35:05]     [Step 2/2] Process exited with code 0
[23:35:05] ▶ Publishing internal artifacts
[23:35:10]   Build finished
```

Figure 34. Build process log created by TeamCity

V. DISCUSSION

CI/CD platform of software engineering and software project management using Jenkins and TeamCity is managed to work without any major problems. Experimentation was done for various cases including student projects. We managed to do a small survey which gives results of positive acceptance of this platform by users and students.

The results of our experimentations using Jenkins show that Jenkins significantly improves the performance of the CI/CD pipeline. Jenkins greatly reduces the time necessary to do the process. So far, we don't experience a slowdown caused by bottlenecking of the hardware, since our hardware is relatively powerful enough to handle the job. Furthermore, we experience similar performance and ease of TeamCity. There is no significant performance difference, in term of build process speed between Jenkins and TeamCity.

VI. CONCLUSION

We managed to build a CI/CD platform of software engineering and software project management using Jenkins and TeamCity. The CI/CD platform provides pipelined processes necessary to a software development from source code coding process until product deployment, all in automatic fashion. Both tools feature product testing stages which assure the quality of the product.

Through our experimentations, we can conclude that Jenkins and TeamCity are similar in performance and ease, with assumption that both are installed in a "decent" good performance hardware. Therefore, the CI/CD platform using Jenkins and TeamCity is suitable for software project management in higher education, in particular for students majoring information technology.

ACKNOWLEDGEMENT

We thank the Faculty of Information Technology, Universitas Kristen Maranatha for funding this research. Lembaga Penelitian dan Pengabdian Kepada Masyarakat (LPPM) Universitas Kristen Maranatha is also acknowledged for facilitating the administrative process necessary for this research. User and students' positive acceptance is highly appreciated.

REFERENCES

[1] R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach, 9th edition, New York: McGraw-Hill Education, 2019.

[2] B. Fitzgerald and K.-J. Stol, "Continuous software engineering: A roadmap and agenda," *Journal of Systems and Software,* vol. 123, pp. 176-189, 2017.

[3] L. E. Lwakatare, T. Kilamo,, T. Karvonen, T. Sauvola, V. Heikkilä, J. Itkonen, P. Kuvaja, T. Mikkonen, M. Oivo and C. Lassenius, "DevOps in practice: A multiple case study of five companies," *Information and Software Technology,* vol. 114, pp. 217-230, 2019.

[4] A. Mishra and Z. Otaiwi, "DevOps and software quality: A systematic mapping," *Computer Science Review,* vol. 38, pp. 1-14, 2020.

[5] D. J. Mala, "Integrating the Internet of Things into Software Engineering Practices," in *Advances in Systems Analysis, Software Engineering, and High Performance Computing,* Hershey, Pennsylvania, USA, IGI Global, 2019, p. 16.

[6] D. Ståhl, T. Mårtensson and J. Bosch, "The continuity of continuous integration: Correlations and consequences," *Journal of Systems and Software,* vol. 127, pp. 150-167, 2017.

[7] J. Humble and D. Farley, Continuous delivery : reliable software releases through build, test, and deployment automation, Boston: Addison-Wesley Professional, 2010.

[8] L. Chen, "Continuous Delivery: Overcoming adoption challenges," *Journal of Systems and Software,* vol. 128, pp. 72-86, 2017.

[9] M. Fowler, "Continuous Integration," [Online]. Available: http://www.martinfowler.com/articles/continuousIntegration.html. [Accessed 23 November 2019].

[10] M. Fowler, "Continuous Delivery," [Online]. Available: http://martinfowler.com/bliki/ContinuousDelivery.html. [Accessed 23 November 2019].

[11] I. Zubin,, "5 common pitfalls of CI/CD -- and how to avoid them," *InfoWorld,* 28 March 2018.

[12] H. Martin, "Continuous integration is not always the right answer. Here's why.," *TechBeacon,* 20 July 2015.

[13] Atlassian, "Continuous integration vs. continuous delivery vs. continuous deployment.," *Atlassian,* 14 April 2017.

[14] M. Manturewicz, "What is CI/CD," [Online]. Available: https://codilime.com/what-is-ci-cd-all-you-need-to-know. [Accessed 23 November 2019].

[15] T. Lehtonen, S. Suonsyrjä, T. Kilamo and T. Mikko, "Defining metrics for continuous delivery and deployment pipeline," in *Proceeding of 14th Symposium on Programming Languages and Software Tools,* University of Tampere, Finland, October 9-10, 2015.

[16] S. Arachchi and I. Perera, "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management," in *MERCon 2018,* Moratuwa, Sri Lanka, May 2018.

[17] S. Krusche and L. Alperowitz, "Introduction of Continuous Delivery in Multi-Customer Project Courses," in *ICSE Companion 2014: Companion Proceedings of the 36th International Conference on Software Engineering. ACM.,* Hyderabad, India, May 2014.

[18] L. B. Torvalds, "GIT," Software Freedom Conservancy, [Online]. Available: https://git-scm.com. [Accessed 1 February 2020].

[19] "GitHub," GitHub, Inc., [Online]. Available: https://github.com. [Accessed 1 May 2020].

[20] A. V. Aho, M. S. Lam, R. Sethi and U. D. Jeffrey, Compilers-Principles, Techniques, and Tools, 2nd edition., Boston: Pearson Education, 2007.

[21] K. Naik and P. Tripathy , Software Testing and Quality Assurance, Theory and Practice, Hoboken: Wiley, 2008.

[22] M. Aggarwal, TeamCity: continuous integration & DevOps with Java and .NET, Birmingham: Packt Publishing, 2018.

[23] V. Melymuka, "Continuous Delivery with TeamCity," in *XPDays* , Kiev, Ukraine , 2012.

[24] S. Machiraju and S. Gaurav, "Deployment via TeamCity and Octopus Deploy," in *DevOps for Azure Applications,* Berkeley, CA, USA, Apress, 2018, pp. 11-38.

[25] J. Lee, Master Jenkins Course For Developers and DevOps, Birmingham: Packt Publishing, 2017.

[26] V. Armenise, "Continuous Delivery with Jenkins: Jenkins Solutions to Implement Continuous Delivery.," in *2015 IEEE/ACM 3rd International Workshop on Release Engineering (RELENG),* Florence,

p-ISSN : 2443-2210
e-ISSN : 2443-2229

Italy, 2015.

[27] P. Rai, Madhurima, S. Dhir, Madhulika and A. Garg, "A prologue of JENKINS with comparative scrutiny of various software integration tools," in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2015.

[28] R. Arpitha and S. N. Kavitha, "Automation Using Jenkins: Plugins,Test, Design, Test Execution and Reporting," *Imperial Journal of Interdisciplinary Research,* vol. 3, no. 5, pp. 1171-1174, 2017.

[29] I. K. Moutsatsos et al., "Jenkins-CI, an Open-Source Continuous Integration System, as a Scientific Data and Image-Processing Platform," *SLAS DISCOVERY: Advancing the Science of Drug Discovery,* vol. 22, no. 3, p. 238–249, 2017.

[30] "Continuous Integration and Delivery - CircleCI," Circle Internet Services, Inc., [Online]. Available: https://circleci.com. [Accessed 1 May 2020].

[31] "GitLab," GitLab Inc., [Online]. Available: https://about.gitlab.com. [Accessed 1 May 2020].

[32] "Azure DevOps Server," Microsoft, [Online]. Available: https://azure.microsoft.com/en-us/services/devops/server/. [Accessed 1 Nov 2020].

[33] "Travis CI," TRAVIS CI, GMBH, [Online]. Available: https://www.travis-ci.com. [Accessed 1 Nov 2020].

[34] "Go CD," ThoughtWorks Inc., [Online]. Available: https://www.gocd.org. [Accessed 1 Nov 2020].

[35] "Bamboo," Atlassian, [Online]. Available: https://www.atlassian.com/software/bamboo. [Accessed 1 Nov 2020].

[36] "CloudBees CodeShip," CloudBees, Inc., [Online]. Available: https://www.cloudbees.com/products/codeship. [Accessed 1 Nov 2020].

[37] "Buddybuild," Doe Pics Hit Holdings ULC., [Online]. Available: https://buddybuild.com. [Accessed 1 Nov 2020].

[38] "AWS CodePipeline," Amazon Web Services, Inc., [Online]. Available: https://aws.amazon.com/codepipeline/. [Accessed 1 Nov 2020].

[39] "Jenkins," CloudBees, [Online]. Available: https://jenkins.io. [Accessed 1 July 2019].

[40] J. Brains, "TeamCity," Jet Brains, [Online]. Available: https://www.jetbrains.com/teamcity. [Accessed 1 July 2019].

[41] "TeamCity Plugins," JetBrains, [Online]. Available: https://plugins.jetbrains.com/teamcity. [Accessed 1 Nov 2020].

[42] S. Labourey, "Enterprise DevOps: The Spine is Critical," CloudBees, 20 March 2020. [Online]. Available: https://www.cloudbees.com/blog/enterprise-devops-spine-critical. [Accessed 1 April 2020].