**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Continuous Integration and Continuous Delivery

In

Web Application Development

by

Nguyen Thi Tuyet Trinh

A thesis submitted in conformity with the requirements  
for the degree of Master of Software Engineering

© Copyright by Nguyen Thi Tuyet Trinh - 2018

**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Continuous Integration and Continuous Delivery

In

Web Application Development

by

Nguyen Thi Tuyet Trinh

A thesis submitted in conformity with the requirements  
for the degree of Master of Software Engineering

Supervisor:

1. Dr. Phan Duy Hung PhD

© Copyright by Nguyen Thi Tuyet Trinh - 2018

Continuous Integration and Continuous Deployment

In

Web Application Development

Nguyen Thi Tuyet Trinh

Degree Master of Software Engineering

FPT University

2018

Abstract

The thesis focused on a web application development which using the continuous integration and continuous deployment methodology includes:

* The set of tools for the implementation of the continuous integration are introduced and reviewed.
* The development of the web application on the continuous integration; the web application’s automation testing and agile techniques are analyzed in detail reflecting the quality improvement.
* The implementation on a Vietnamese project with its difficulties and its solution which lead to the lesson learn.
* The expected evolution of the continuous integration and continuous development in the future is briefly described.

Acknowledgments

In order to finished this thesis, there are many people supporting me by giving their advice and knowledge. Therefore, the author, who conducted this research gratefully give acknowledgments to their support and motivation during the time of doing this thesis as the requirement of completing the Degree of Master Software Engineering.

First of all, I would like to express my thanks and gratefulness to my supervisor Dr. Phan Duy Hung, whose is the leader of MSE Program in FPT University. His kindly support and continuous advices went through the completion of my thesis. His encouragement and comments had significantly enriched and improved my work. Without his patient, motivation and introduction, the thesis would have been impossible to be done effetely.

Besides my advisor, I would like to thank “The Company” and my project managers who provided me an opportunity to join the projects, without they precious support, it would not be possible to conduct this thesis.

My sincere thanks also goes to Mr. Phung Khac Toan a friend who provide a lot of support during the thesis time and my colleagues in “The Company” for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspective. Their experiences in various projects help contributed ideas and reflection surface for this thesis.

Table of Contents

[Acknowledgments 2](#_Toc513984797)

[Table of Contents 4](#_Toc513984798)

[List of Tables (if any) 7](#_Toc513984799)

[List of Plates (if any) 8](#_Toc513984800)

[List of Figures (if any) 9](#_Toc513984801)

[List of Appendices (if any) 10](#_Toc513984802)

[Chapter 1 Introduction 1](#_Toc513984803)

[1 Introduction 1](#_Toc513984804)

[1.1 Mission of the thesis 1](#_Toc513984805)

[1.2 Outline of the thesis 1](#_Toc513984806)

[1.3 Introduction of the thesis 1](#_Toc513984807)

[Chapter 2 Research 2](#_Toc513984808)

[2 Research 2](#_Toc513984809)

[2.1 Iterative and Incremental development 2](#_Toc513984810)

[2.1.1 Rational Unified Process 3](#_Toc513984811)

[2.1.2 Agile Development 8](#_Toc513984812)

[2.1.3 Comparison of Rational Unified Process and Agile 13](#_Toc513984813)

[2.2 Continuous Integration 14](#_Toc513984814)

[2.2.1 Overview 14](#_Toc513984815)

[2.2.2 Key practices 14](#_Toc513984816)

[2.2.3 The benefit of Continuous Integration 16](#_Toc513984817)

[2.3 Continuous Delivery 17](#_Toc513984818)

[2.4 Automation Testing 17](#_Toc513984819)

[2.5 Cloud Infrastructure 18](#_Toc513984820)

[2.6 Configuration Management 18](#_Toc513984821)

[2.7 Common pitfall of CICD 18](#_Toc513984822)

[Chapter 3 Integration Process 20](#_Toc513984823)

[3 Integration Process 20](#_Toc513984824)

[3.1 Source code compilation 20](#_Toc513984825)

[3.2 Code inspections 21](#_Toc513984826)

[3.3 Deployment 21](#_Toc513984827)

[Chapter 4 Implementation 22](#_Toc513984828)

[4 Implementation 22](#_Toc513984829)

[4.1 Criteria for tool selection 22](#_Toc513984830)

[4.2 Final selection 22](#_Toc513984831)

[4.3 Supportive software 22](#_Toc513984832)

[Chapter 5 Case Study 23](#_Toc513984833)

[5 Case Study 23](#_Toc513984835)

[5.1 Brief Introduction 23](#_Toc513984836)

[5.1.1 Problem 23](#_Toc513984837)

[5.1.2 Project approach 24](#_Toc513984838)

[5.2 CICD tools selection 25](#_Toc513984842)

[5.2.1 Version control system 25](#_Toc513984843)

[5.2.2 Continuous Integration and Continuous Deployment server 26](#_Toc513984844)

[5.2.3 Automated function testing tool 27](#_Toc513984845)

[5.3 Specific the work 28](#_Toc513984846)

[5.3.1 Step by step 28](#_Toc513984847)

[5.3.2 Install Docker 29](#_Toc513984848)

[5.3.3 Install Jenkins 30](#_Toc513984849)

[5.3.4 Install Jenkins plugins 32](#_Toc513984850)

[5.3.5 Set up the first Pipeline 34](#_Toc513984851)

[5.3.6 Configure pipeline in Jenkins 35](#_Toc513984852)

[5.3.7 Configure pipeline in Jenkins 35](#_Toc513984853)

[5.3.8 Configure Webhook 36](#_Toc513984854)

[5.3.9 Maintain script 37](#_Toc513984860)

[5.4 Result 37](#_Toc513984861)

[5.4.1 Quality 37](#_Toc513984862)

[5.4.2 Time and Price 37](#_Toc513984863)

[5.5 Lesson learn 37](#_Toc513984864)

[5.5.1 The structure of branch should be… 38](#_Toc513984865)

[5.5.2 Do not merged code while CI failed 38](#_Toc513984866)

[5.5.3 CD 38](#_Toc513984867)

[Chapter 6 Conclusion 39](#_Toc513984868)

[6 Conclusion 39](#_Toc513984869)

[6.1 Before CICD 39](#_Toc513984870)

[6.2 After CICD 39](#_Toc513984871)

[References or Bibliography (if any) 40](#_Toc513984872)

[Appendices (if any) 41](#_Toc513984873)

[Copyright Acknowledgements (if any) 42](#_Toc513984874)

List of Tables (if any)

List of Plates (if any)

List of Figures (if any)

[Figure 1 - Iterative Development Process 3](#_Toc513911665)

[Figure 2 - Rational Unified Process 8](#_Toc513911666)

[Figure 3 - Values of Agile Manifesto 9](#_Toc513911667)

[Figure 4 - Subway Map to Agile Practices 11](#_Toc513911668)

[Figure 5 - SCRUM process 13](#_Toc513911669)

[Figure 6 - Comparison of some Automation Testing frameworks 18](#_Toc513911670)

List of Appendices (if any)

1. Introduction

The main body of your thesis begins here.

# Introduction

## Mission of the thesis

\*Chưa viết lại\*

My thesis focus on the use of continuous integration in web application development, oriented the management point of view. This is aim to bring a complex view on best practices of the continuous integration regarded as a part of a development life cycle and to analyze them based on a practical use with a strong focus on quality, but also with respect to price and time. The secondary aim is to introduce a set of tools which fits well into the continuous integration process.

## Outline of the thesis

\*Chưa viết, để sau \*

## Introduction of the thesis

\*Chưa viết, để hoàn thiện sau \*

1. Research

The Research chapter.

# Research

The continuous integration and continuous delivery came simultaneously with the evolution of a development life cycle. The classical development approaches are confronted with agile development techniques, which bring new lightweight point of view on software development. The best agile development practices are closely related to the continuous integration best practice and together can bring ideal way of how to develop applications. The first sub-chapter present iterative and incremental development which includes the classic software development lifecycle Rational Unified Process and the trending development method Agile Manifesto.

The second sub-chapter generally specifies Continuous Integration with key practices and its benefit. Continuous integration introduces its own history, the best practices and the general values of this approach to the development.

The third sub-chapter describe the understanding Continuous Delivery…

The fourth sub-chapter is written to bring the Automation testing ideal…

Cloud Infrastructure in fifth sub-chapter …

Configuration Management in sixth sub-chapter…

The last one I want to bring up the common pitfall of CICD and …

## Iterative and Incremental development

It’s no doubt that choosing a right process for a software development is a key lead to the project success. A project which is developed without a process or with non-effective process, can easily fail, facing with poor quality issues or be overpriced.

Iterative and Incremental development is a combination of Iterative method and Incremental development model. The core of this process is breaking down the software development into smaller chunks.

In Iterative development, features of software application will be designed, developed and tested in repeated cycles. With each iteration, additional features can be added, designed, developed and tested until fully functional of system finished and ready to be released.

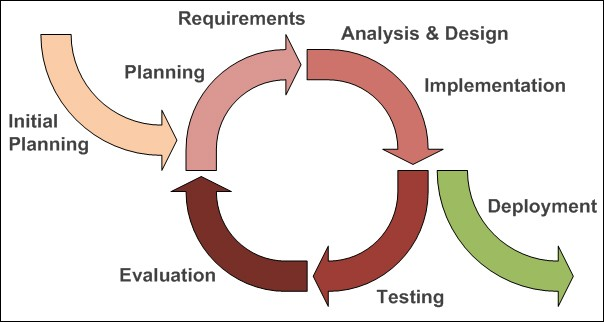


Figure 1 - Iterative Development Process

### Rational Unified Process

There are many methods and approaches for Iterative and Incremental development method and the best classic practices named Rational Unified Process – developed by Rational Software Corporation, division of IBM.

Rational Unified Process (RUF) is a Software Development process, which developed by Rational Software Corporation, a division of IBM. Rational Unified Process (RUF) defined process of assigning the tasks for both Software development lifecycle phases and each development team member. The main idea of Rational Unified Process is to ensure that the Software product meets the expectations of all client and end users, therefore ensure a high quality Software production.

The Time Phases:

RUP defined that one development lifecycle has 4 time phases:

* Inception phase
* Elaboration phase
* Construction phase
* Translation phase

#### Inception phase

Inception phase is the initial phase of Rational Unified Process. The main purpose of inception phase is: software developers bring up the plan for development and business cases, therefor they can describe the scope of development process and definition of business cases. After this phase, we should have:

* A vision document: a general vision of the core project’s requirements, key features, and main constraints.
* An initial use case model (10 % - 20 % complete)
* An initial project glossary (may be also partially expressed as a domain model)
* An initial business case, which includes business context, success criteria (revenue projection, market recognition etc.), and financial prognosis.
* An initial risk assessment
* A project plan, showing phases and iterations
* A business model (if necessary)
* One or several prototypes

#### Elaboration phase

Elaboration phase is the second phase. In this phase, software developer define all system features in detail. The output of this phase are:

* A use case model (at least 80% complete) - all use cases and actors have been identified, and most use case descriptions have been developed.
* Supplementary requirements capturing the non-functional requirements and other requirements that are not associated with a specific use case.
* A software architecture description
* An executable architectural prototype
* A revised risk list and a revised business case.
* A development plan for the overall project, including the coarse-grained project plan, showing iterations and evaluation criteria for each iteration.
* An updated development case specifying the process to be used. A preliminary user manual (optional)

#### Construction phase

Construction phase is the third phase. In this phase, software developers develop the Software product. The output of this phase are:

* Design documents
* Source code
* The software product
* The user manual

#### Transition phase

Transition phase is the last phase. In this phase, the software developers finish all last steps in development lifecycle and bring up the completed software product to client. The output of this phase are:

* Beta testing to validate the new system against user expectations
* Parallel operation with a legacy system that it is replacing
* Conversion of operational databases
* Training of users and maintainers
* Rolling-out the product to the marketing, distribution, and sales teams

**Core process:**

RUF defined that core process divides the Development phase into logical core phase (workflows). There are 6 core workflows:

* Business modeling
* Requirements
* Analysis & design
* Implementation
* Test
* Deployment

#### Business modeling

One of the biggest issue in development phase is the lack of communication/misunderstanding between Software development team and Business department. It will lead to an inaccurate interpretation of business requirements. To solve this problem, RUF provides common language/patterns understandable by both departments. That is business modeling.

#### Requirements

Software requirements is a field in software engineering that deals with establishing the needs of stakeholders that are to be solved by software.

The goal of Requirement workflow is to define all need by user. It should be all functional requirements and non-functional requirement. After this phase, software development team can have an overview of system behavior.

#### Analysis & design

In this phase, based of output of Requirement phase, the design model of project will be created. The software development team will have a detail view of how system will be and the way to develop the system.

#### Implementation

In this Implementation phase, the system will be created by Software development team. The goals are:

* All components that make up the system will be created
* Classes and objects are implemented and integrated
* The components will be unit tested

#### Test

In this Test phase:

* All components will be tested
* The interaction between objects will be verified
* All requirements (includes functional requirements and non-functional requirements) will be verified and fulfilled
* Defects are identified and tracking during this phase
* Correction process will be started if needed and verified

#### Deployment

In this Deployment phase, the system will be successfully deployed on customer environment. The deployment process includes:

* Packaging the production
* Deploying the production on customer environment
* Planning and conduction user support (if needed)
* Migration data or integrating existing software/system (if needed)
* Formal user acceptance

The whole Rational Unified Process is illustrated in figure 2.1

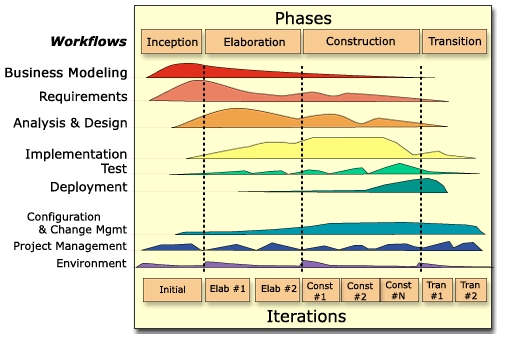


Figure 2 - Rational Unified Process

### Agile Development

Probably we are in the “Age of Agile”

The Agile Manifesto came as the result of a meeting in February 2001 that brought together a number of software and methodology experts, who then defined the Agile Manifesto and Agile Principle. The main purposes are:

* Individual and interactions over process and tools.
* Working software over comprehensive documentation.
* Customer collaboration over contract negotiation.
* Responding to change over following a plan.

The Agile Manifesto is not set a rules telling us to do one thing instead of another. It is subtler, and more powerful –it guides us to consider project from a value-based perspective.

The main benefit of Agile approach is helps team to be ready for dynamic and unpredictable project development scenarios.

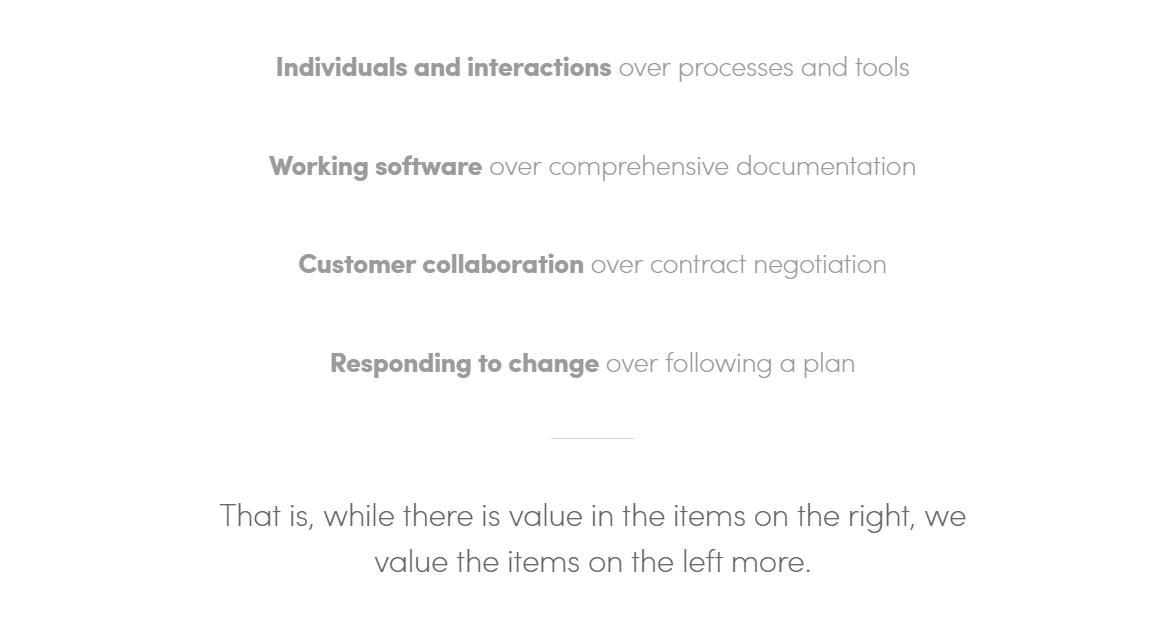


Figure 3 - Values of Agile Manifesto

#### Principles

In the Agile Manifesto, twelve principles of agile development were defined:

1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

#### Agile Practices

Started from 1986 with the “Conway’s Law”\*, thru 3 decades, Agile is practiced, grown and accretion as we see as below map.

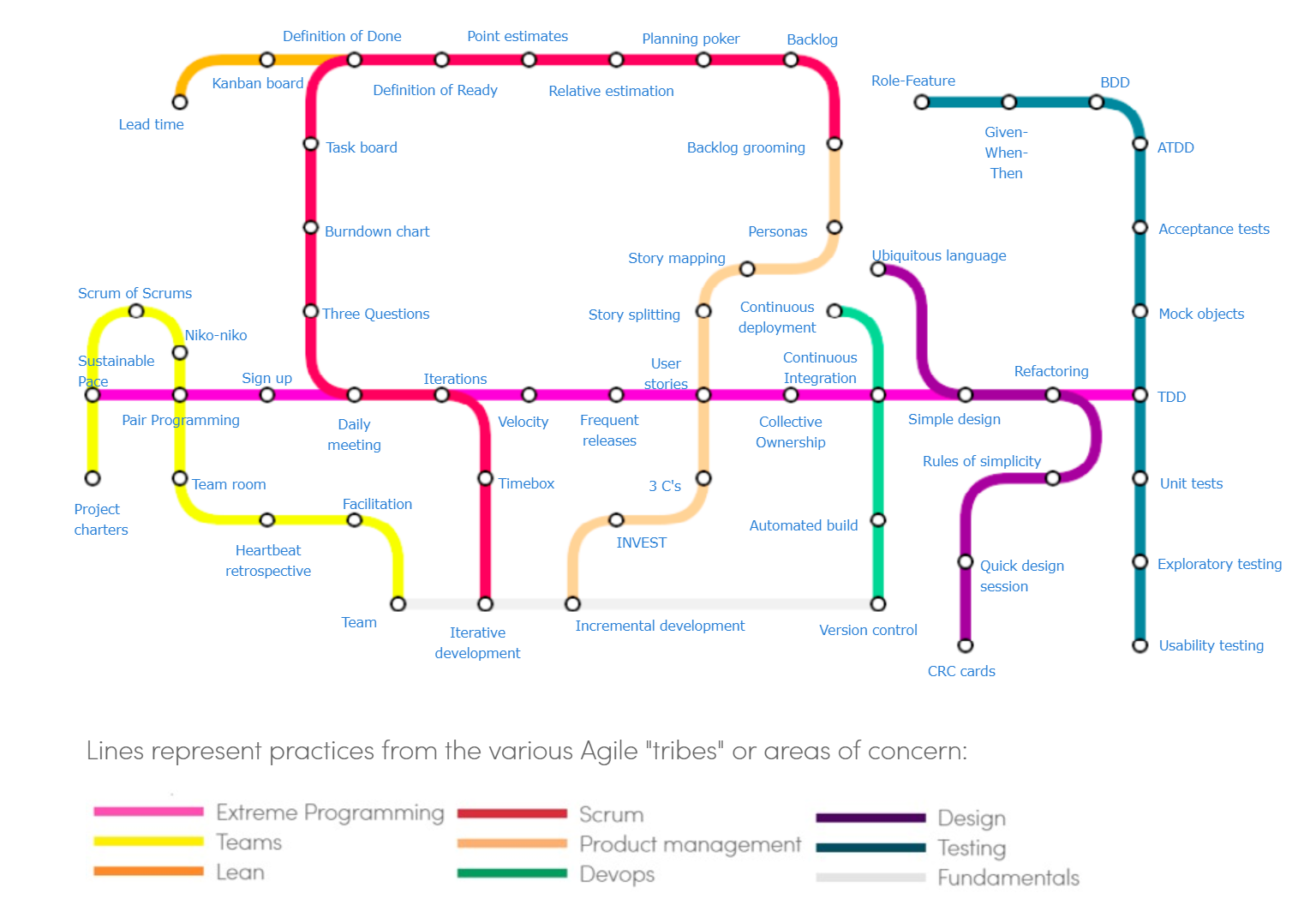


Figure 4 - Subway Map to Agile Practices

#### Extreme Programming

Described in Extreme Programming Explained by Kent Beck, Extreme Programming is a lightweight and low-risk way to develop a software based on an iterative and an incremental development.

Kent Beck summarizes the main difference from other procedures as following:

* Early, specific and continuing feedback from short cycles.
* Quick overall plan is set at the beginning and is expected to evolve through the project life.
* Development fully covered with tests.
* The possibility of flexible changes (responding to business needs in the moment).
* Oral communication, tests and source code to communicate system structure and intent.
* Evolutionary design process.
* Close collaboration of skilled programmers.

#### Feature-driven Development

In A Practical Guide to Feature Driven Development, Feature Driven Development is described as a process designed and proven to deliver frequent, tangible, working results repeatedly. It is a development approach, which brings systems that are based on easy-to-understand and easy-to implement methods, problem-solving techniques and reporting guidelines to keep stakeholders informed about the project in a way they need.

#### Scrum

Scrum is a process framework used to manage product development and other knowledge work. Scrum provides a means for teams to establish a hypothesis of how they think something works, try it out, reflect on the experience and make the appropriate adjustments. That is when the framework is used properly.

Scrum is best suited in the case where a cross functional team is work in a product development setting where there is a non-trivial amount of work that lends itself to being split into more than one 2-4 weeks’ iteration.

There are 3 essential roles in Scrum: Scrum Master, Product Owner and Development Team.

The Scrum Lifecycle starts with a prioritized backlog, but does not provide any guidance as to how that backlog is developed or prioritized. The Scrum Lifecycle consists of a series of Sprints, where the end result is a potentially shippable product increment. Inside these sprints, all of the activities necessary for the development of the product occur on a small subset of the overall product. Below is a description of the key steps in the Scrum Lifecycle.

1. Establish the Product Backlog.
2. The product owner and development team conduct Sprint Planning. Determine the scope of the Sprint in the first part of Sprint Planning and the plan for delivering that scope in the second half of Sprint Planning.
3. As the Sprint progresses, development team perform the work necessary to deliver the selected product backlog items.
4. On a daily basis, the development team coordinate their work in a Daily Scrum.
5. At the end of the Sprint the development team delivers the Product Backlog Items selected during Sprint Planning. The development team holds a Sprint Review to show the customer the increment and get feedback.  The development team and product owner also reflect on how the Sprint has proceeded so far and adapting their processes accordingly during a retrospective.
6. The Team repeats steps 2–5 until the desired outcome of the product have been met.

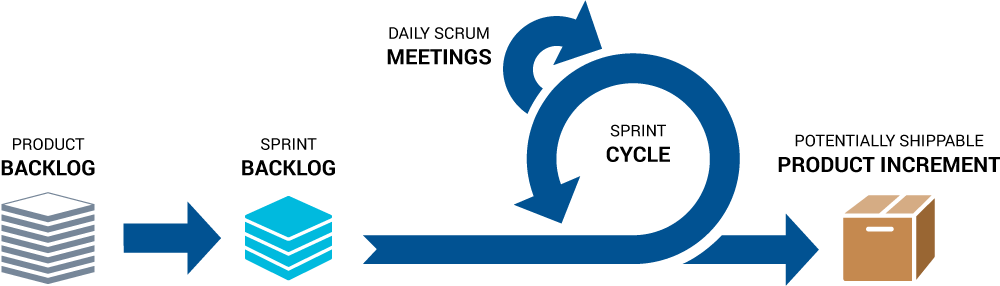


Figure 5 - SCRUM process

### Comparison of Rational Unified Process and Agile

.

They are both iterative models which seem similar, but both of them are vastly different. RUP is a framework for organizations and teams while Scrum is intended for a product team with stringent guidelines

Some diffs are: RUP uses 4 phases, project plan associated with 4 phases, scope is predefined, good for large, long term enterprise level projects. SCRUM uses sprints, no project plan, uses backlog instead of scope, and good for orgs not dependent on deadline.

Agile in an umbrella term for methods like XP, Scrum, Crystal, DSDM, FDD,... that share [common principles](http://agilemanifesto.org/principles.html). The [Unified Process](http://en.wikipedia.org/wiki/Unified_Process) is a framework that can be used to describe a development process, [RUP](http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process) being one instantiation of UP based on Rational's tools. UP predates most Agile methods and may or may not be considered as Agile. What they have in common is that both Agile methods and UP are **Iterative and Incremental Development** (IID) methods.

http://www.agilemodeling.com/essays/agileModelingRUP.htm

## Continuous Integration

### Overview

Continuous Integration is a concept, which first popular explained by Martin Fowler in 2006:

*Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. [STT]* Martin Fowler

Many teams find out that this approach leads to significantly reduced the integration problems therefor reduced the effort and allows team members to develop software more rapidly.

### Key practices

In Continuous Integration article, Martin Fowler defined 10 basic practices:

* Maintain a single source repository: all source of project need to be maintained and managed by source management tool – version control system. The main purpose of this practice is to manage whole project source in a repository. When a new member join team, he just need to clones this repository to his computer and work without any configuration else. The advantages of this step is reduce the effort to setup working environment for team member and unified source code in single place.
* Automate the build: Getting the sources turned into a running system can often be a complicated process – as Martin Fowler mentioned. To avoid risk for new members to running some new commands, clicking through some dialog boxes, there are some way to simplify this step like Ant for Java or MSBuild for .Net. The target is that everyone new using clean machine should be able to build the sources code into a running application.
* Make your build self-testing: Traditionally, a build includes compiling, linking and a ton of things required to get a program to execute. Although running through all step needed, it doesn’t mean that program working well. To reduce the effort to checking the system and make sure the system running well, these should have a process to test the source code checked in every time. And this step better should be automated trigger.
* Everyone commits to the mainline every day. This statement is self-describe. The mainline of repository better be as latest as possible. The advantages are helping new members have a latest repository, and reduce the risk to losing code.
* Every commit should build the mainline on an Integration machine. The mainline of repository should be consider like the mainline of project. The team member, who commits the last change need to be sure that the mainline on Integration machine was built successfully. Otherwise, this man should take responsibility to fix incase the build fail. The advantages of this statement is in any time, the project is ready for deploy (for Continuous Deployment) and there are always have an in charge person, who has responsibility for the successfully of the build.
* Fix broken build immediately: The broken build prevents project to have a successfully deployment. The one, who committed the change should fix the build right after received the broken build alert to make sure, there are always have a successfully build in system. Nowadays, team member can configure the CICD system (like Jenkin) to ignore the merge of broken build branch to master branch.
* Keep the build fast. The main idea of CICD is provide a rapid feedback system. If a build takes a long time, it’s not a rapid feedback system anymore. As Martin Fowler said: Nothing sucks the blood of a CI activity more than a build that takes a long time. Almost teams agreed that the build should finish around 10 minutes. It makes sure that every team member doesn’t need to wait a build for too long, and people can work on the latest code with confidence.
* Test in a clone of the Production environment. The point of testing is to find out the issues of system before deploy to the Production, and make sure the Production working well with new changes. Therefor this is necessary to test all functions in same environment as Production environment.
* Make it easy for anyone to get the latest executable. As Fowler mentioned, one of the hardest part in Software development is make sure the software released by development team is the right thing customer want. To achieved it, all people in team should be able to get the latest executable and run it for demonstration, testing or just viewing.
* Everyone can see what’s happening. Continuous Integration is just about communication. In another words, by Continuous Integration all software development team members can see what’s happening with project at this time. It can be the status of mainline, branches, or the changes that have been made to.
* Automate deployment. To do Continuous Integration, you will need multiple environments for: testing purpose, integration environment, automation testing purpose. Because you will need to change between these environments each day, so it’s important to have scripts to automate deploy the application easily.

### The benefit of Continuous Integration

* Better project visibility: This is the greatest benefit of Continuous Integration. By CI, software development team member can observe the status of mainline and branches, what works, what doesn’t … which cannot in the past.
* Reduce risks: All test scripts can be configured to run after the changes committed. The result after test script run will be sent to in charged person or all development team member. Therefore, the risk of all defined test case will be reduced.
* Reduction of repetitive manual processes: With Continuous Integration, the processed are automated and run without manually work. Software development team member can use scripts to deploy, Unit test scripts will be trigger right after the change committed… are one of many advantages come with Continuous Integration.
* Deployable build: With a token, it’s possible to release deployable software anytime.

## Continuous Delivery

Continuous Delivery is a familiar concept with Software development team using Agile methodology. Continuous Delivery builds on Continuous Integration and along with Continuous Integration. After the change is committed (Software development team checked in some change), Continuous Integration server (E.g.: Jenkins) runs several test process like Unit test or Integration test to ensure that the code is working well and align with coding rule/coding convention, defined by Software development team. After passed all existing testing process, the code change will be sent to staging environment for deployment.

## Automation Testing

Automation testing is a process in CICD. In this process, the changing code will be tested by several tools/frameworks like Jasmine/Karma Test (for JavaScript language), JUnit (for Java language), Watir (for Ruby language) … The Testing tool/framework depends on development language.

This process ensures that all changing source code will be working well with current system and follow all coding rule/coding convention defined by Software development team.



Figure 6 - Comparison of some Automation Testing frameworks

## Cloud Infrastructure

\* Chưa viết \*

## Configuration Management

\* Chưa viết \*

Configuration Management, or Software Configuration Management is a

## Common pitfall of CICD

\*Chưa viết \*

Five activities make DevOps are: continuous integration, continuous delivery, cloud infrastructure, test automation, and configuration management.

1. Integration Process

The Integration Process chapter is aim to …

# Integration Process

System integration is defined in engineering as the process of bringing together the component sub-systems into one system (an aggregation of subsystems cooperating so that the system is able to deliver the overarching functionality) and ensuring that the subsystems function together as a system, and in information technology as the process of linking together different computing systems and software applications physically or functionally, to act as a coordinated whole.

Continuous Integration process starts with a committing source code. Software development team member commits their source code to their Virtual SourceSafe server. Based on their changes, the Continuous Integration server will trigger a build. The process will be:

* Source code compilation
* Code inspections
  + Sonarqube scanning (if existed)
  + Unit test running (if existed)
  + Automation test (if existed)
  + Deployment (if existed)

## Source code compilation

Source code compilation is depending on the coding language.

In some languages (E.g.: Typescript – with Angular 2 and upper in Frontend project), source code need to be compiled to web running language (E.g.: Typescript to JavaScript).

However, some languages don’t need to be compiled (E.g.: PHP). Source files written in interpreted language and just need to be unzipped in the server.

## Code inspections

Code inspections is needed in model software development. It ensures that new code is written in right process and follow all rules pre-defined by Software development team.

Nowadays, there are some popular tools support to Code inspection process like:

* Sonarqube: SonarQube provides the capability to not only show health of an application but also to highlight issues newly introduced. With a Quality Gate in place, you can fix the leak and therefore improve code quality systematically.
* Jasmine/Karma test: Karma test support Frontend project (like Angular) to perform unit testing on CICD. Software development team can use Karma to support TDD process.
* JUnit test: JUnit test supports Software development team to write unit test for Java.
* Automation test: Some frameworks supports Software development team run Automation test like Selenium, Katalon Studio, TestComplete. The advantage of Automation test is: Reusable, Maximum code coverage, Low-cost maintenance, Minimal test effort…

## Deployment

The last step is deploying project to testing/developing/UAT environment. Depends on project configuration, the deployment process will be manual deploy by Software development team or automated deploy by system.

* Manual deployment: Software development team can observe and use the deployment version generated by CI server (like Jenkins) and manual deploy this version to corresponding environment.
* Automated deployment: Based on Project configuration, CI server can support to deploy a build version to corresponding environment (E.g.: deploy build version started with “DEV-” text in committed code branch to DEV environment)

After the deployment process, CI server can send an email/SMS to corresponding Software development team member or all Software development team to announce the status of the environment. The error log may have attached to that email (in case the build process is failed).

1. Implementation

The Implementation chapter.

# Implementation

## Criteria for tool selection

\*Chưa viết\*

## Final selection

\*Chưa viết \*

## Supportive software

\*Chưa viết \*

1. Case Study

# Case Study

After 4 years work in waterfall process model, June 2015, the author was started to contribute in a Scrum project which using CICD. This time build the curious and passion with CICD and the devOps in the author mind. At first, it was only a surprised about the advantage of CICD and the Scrum methodology, then, it turned in to the curious about how the development trending in the world run.

Jan 2018, the author joins the current project which will be described in detail in this chapter.

The first sub-chapter aims to introduction about the case study and its expectation. While in the second sub-chapter, the list of the tools which are selected for case study is described. The third sub-chapter help the reader has a clear look about what the project team do in the case study. The result of development CICD will be summarized in the sub-chapter number four. And last sub-chapter is written to show the lesson learn after working in this case study.

## Brief Introduction

### Problem

A set of websites for the large real-estate company are written by Java in a Content Management System (CMS) – Jasmine will be migrated into other CMS – Adobe Experience Management.

A part of business is new development cause of missing feature in current website.

Project development methodology must be Scrum as the requirement from customer.

**Difficulties:**

AEM is new and expensive system so that only a small group of people have experience about it.

A large number of page (more than 900 pages) of websites need to be migrated in to new system in a short period of time.

### Project approach

Content Management System: Adobe Experience Manager Site

Project development life cycle:

* Iterative and Incremental development - Agile

Project development methodology

* Scrum

Team size:

* Development team: 60 people

Location: 3 sites

* Singapore: the customer
* Ho Chi Minh: the development and migration team for the current product.
* Ha Noi: the development team for new product.

Timeline:

* Development: 10 months
* Migration: 5 months
* Maintenance: 2 months

Development approach:

* Front-end coding language: Bootstrap, …
* Back-end coding language: Java, …

Quality control approach:

* Requirement:
* Automation testing:
* Manual testing:

## CICD tools selection

To setup an CICD system, we used a set of tools:

* Version control system;
* Continuous Integration and Continuous Deployment server;
* And Automated function testing tool.

### Version control system

As known as revision control or source control is a component of software configuration management, version control system used to manage files change in project.

For now, we have several systems, which can support us to work like:

* Git – (brief introduction)
* SVN– (brief introduction)
* Mercurial– (brief introduction)
* Team Foundation Server– (brief introduction)
* Visual Studio Team Services– (brief introduction)
* …

Version control manages changes to set of data over time. The process firstly starts with *checking out* or *clone* repository to local repository. Project development team member can work with local repository by modifying or creating new files. The local repository after that need to be checked in or committed to existing branch or creating new branch. In some system like Git, the changes after committing is not available in remote repository until pushing the committed changes. When a group of people working in a same document or file, the conflict issue may happen when they push their code. In this case, they need to pull the newer code to local branch and merge this code to their local code before pushing to remote repository.

In this thesis, we choose GIT as version control system for project. – tại sao lại chọn GIT mà ko chọn những tool version control khác?

### Continuous Integration and Continuous Deployment server

Nowadays, there are many CI/CD systems presented to the world included open source or paid version:

* Jenkins– (brief introduction)
* TeamCity– (brief introduction)
* Travis CI– (brief introduction)
* Go CD– (brief introduction)
* Bamboo– (brief introduction)
* GitLab CI– (brief introduction)
* …

In this thesis, the author choose Jenkins as an CI/CD server due to the list reason below:

* **The fee:** Jenkins is a free open source written by Java. Every people can use Jenkins for their project without any fee. It’s published under MIT license, therefor everybody can free to use and distribute.
* **The flexibility:** Jenkins is very flexible because it has many extensions (through plugins). Until now, Jenkins has 1000+ plugins, which support people to build their own CI/CD server.
* **Support multi languages:** Jenkins supports almost every development language, no matter project using Java, C#, PHP, … or NodeJS.
* **The communities:** Jenkin is the free of charge tool, and everybody can distribute it, that is the reason why Jenkins has a large communities, which can support and answer every questions related to Jenkins. (Dẫn chứng)
* **Top CI/CD server:** Jenkins is one of biggest names in CI/CD world. (Dẫn chứng)

### Automated function testing tool

One of the biggest benefit when project team apply CICD is Automated Testing.

Automated Testing - as it self-describe can help team member easier to run automation test. Imagine that you just need to write Unit tests only one time and the unit test will be run every time whenever the new change checked in automatically. The result of Unit test was tracked in CICD system time by time. Moreover, project team member can configure CICD system to send test report to specified person. Last, CICD system can prevent new changed code to be merged into master branch if one of the Unit test case fail to run.

Not like Version Control System, Automation testing tool depends on programming language. Some support multiple programming languages like Selenium (support: Java, C#, Python, Javascript…), while the other only support one programming language like Watir (support Ruby only) Jasmine (support testing JavaScript code) or Unified Functional Testing (support VBScript only).



Figure 5.1.2.3: Some Automation Testing tool

Due to sample project written by Angular 5 with TypeScript, in this thesis, the author chose Jasmine as JavaScript automated testing tool and Karma as test runner.

## Specific the work

On this thesis, the author want to focus on CICD part on Web application development, so that this sub-chapter only shows step by step of the work while applying the CICD into case study.

### Step by step

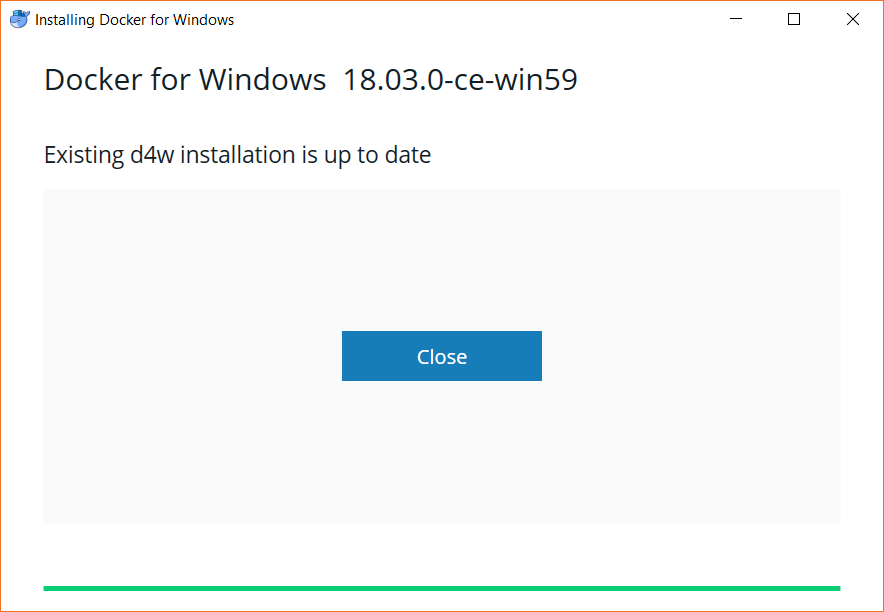
1. Install CICD server - Jenkins
2. Install - Sonar
3. Install deployment hosting - Apache
4. Configuration
5. Create Continuous Integration pipeline Jenkins
6. Create Continuous Deployment pipeline Jenkins
7. Maintain script.

### Install Docker

Docker is a computer program which performs operating system level virtualization, developed by Docker Inc. In order to set up Jenkins, we can use Docker as container platform, which contains Jenkins image – as a container

Step 1: Get latest Docker Community Edition (Docker CE) at: <https://www.docker.com/community-edition#/download>

Step 2: Install Docker

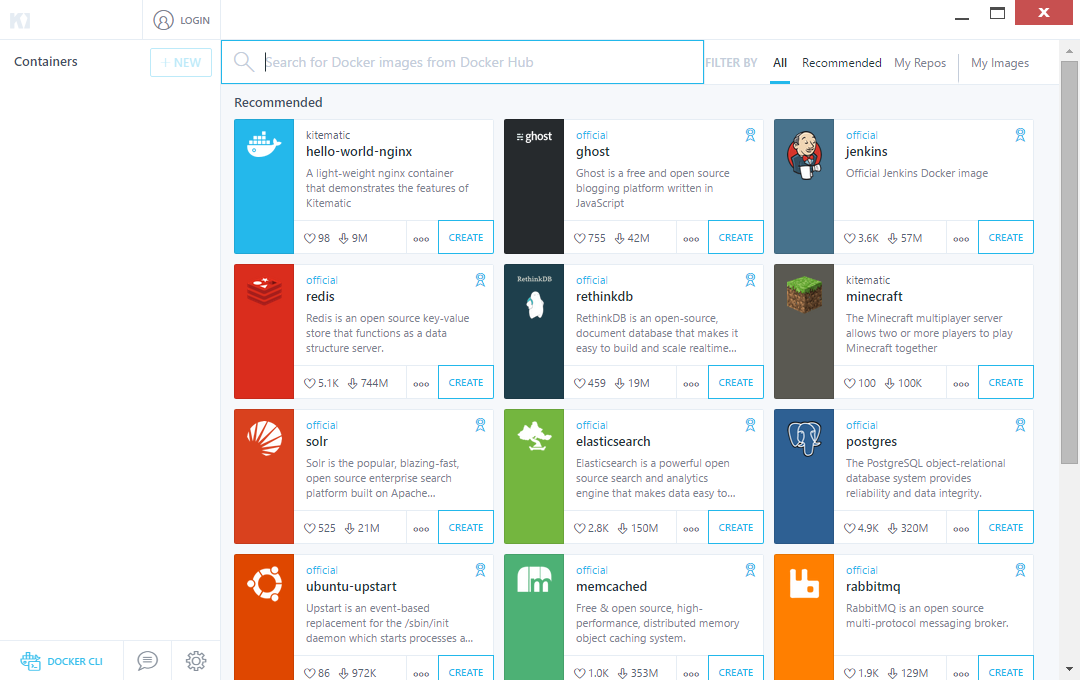


Step 3: Download and install Kitematic – a Graphical User Interface for Docker at <https://kitematic.com/>

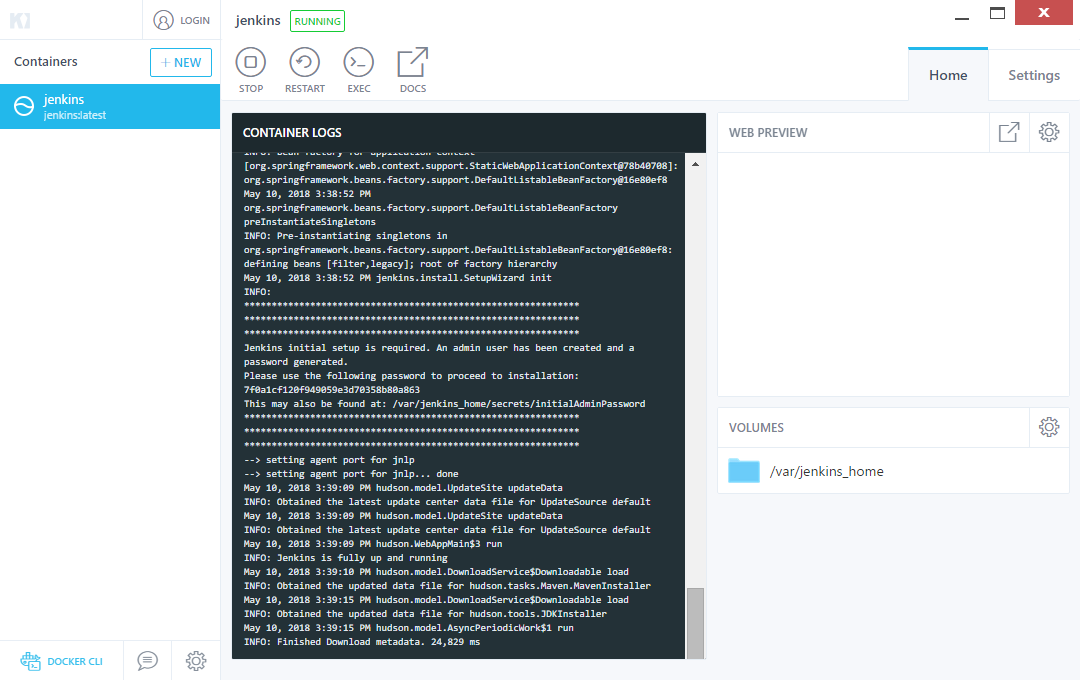
### Install Jenkins

With Docker, we can easily install and run Jenkins as a container via some steps:

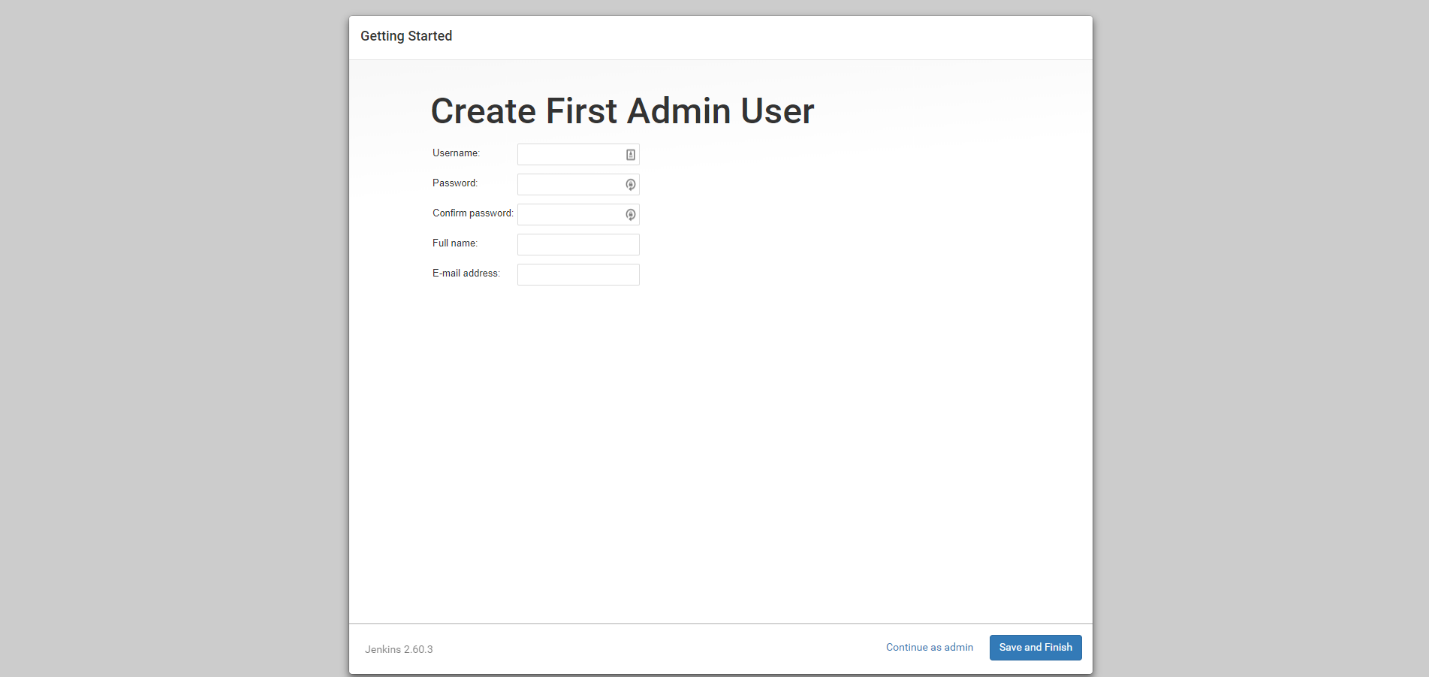
Step 1: Open Kitematic for Docker and create a Jenkins container



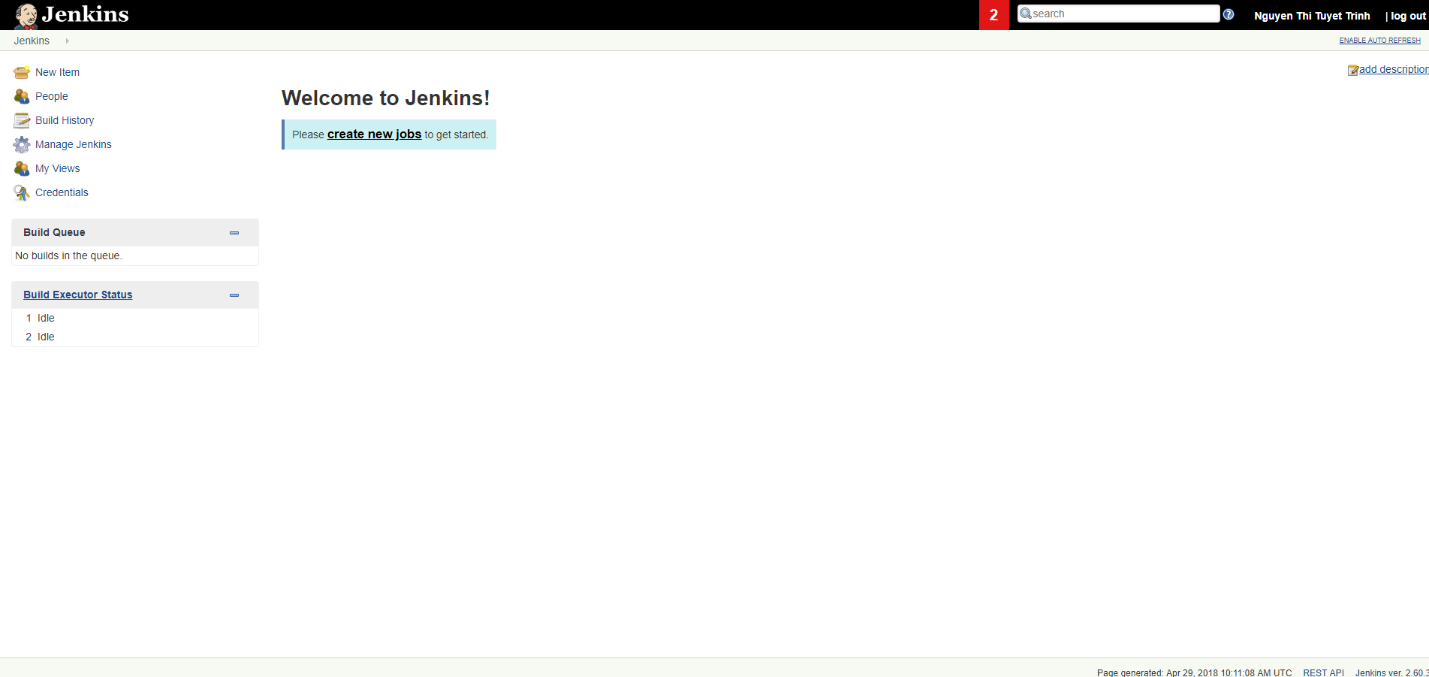
Step 2: Start Jenkins container on Docker



Step 3: Initial Jenkins and create first Admin user



Step 4: Login to Jenkins by username, which created in previous step



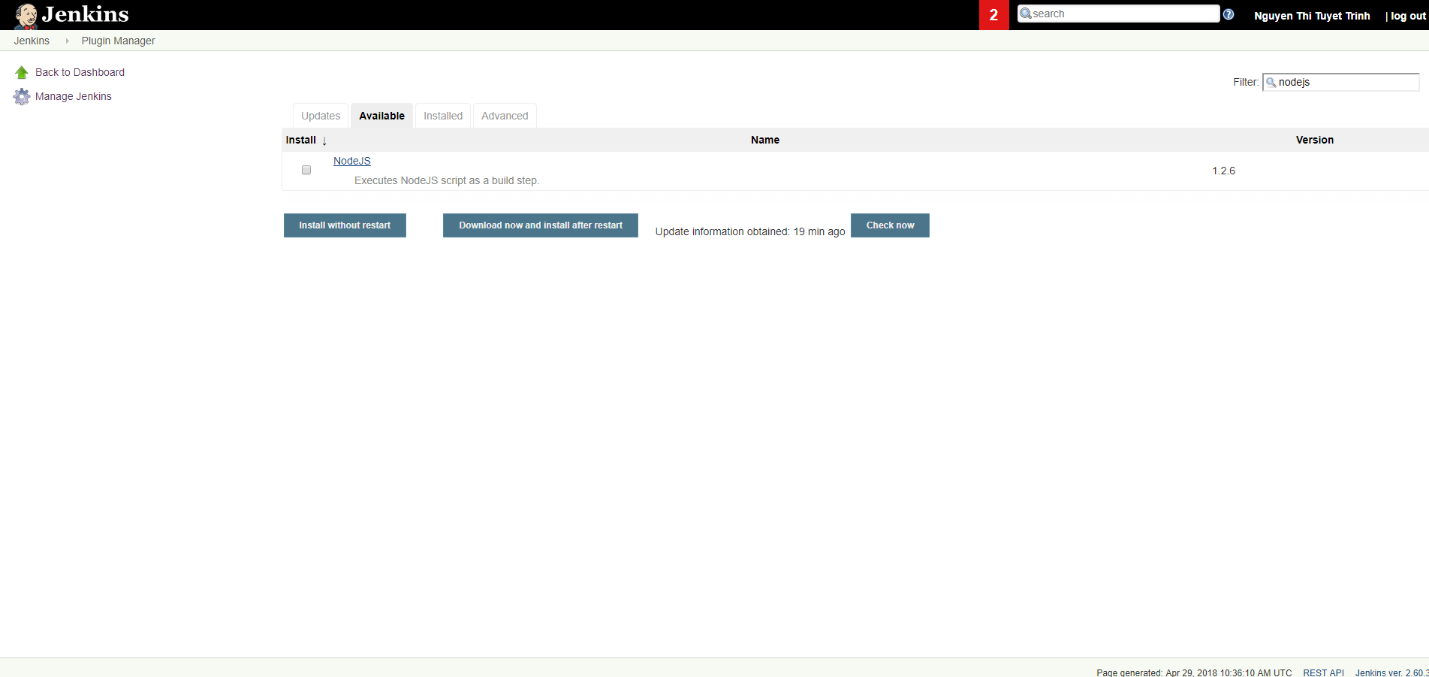
After this step, you have completed the initialization Jenkins on your Docker.

### Install Jenkins plugins

In this thesis statement, we are going to setup CICD environment for a NodeJS project. In order to setup Jenkins for NodeJS project, Jenkins need to be installed NodeJS plugin.

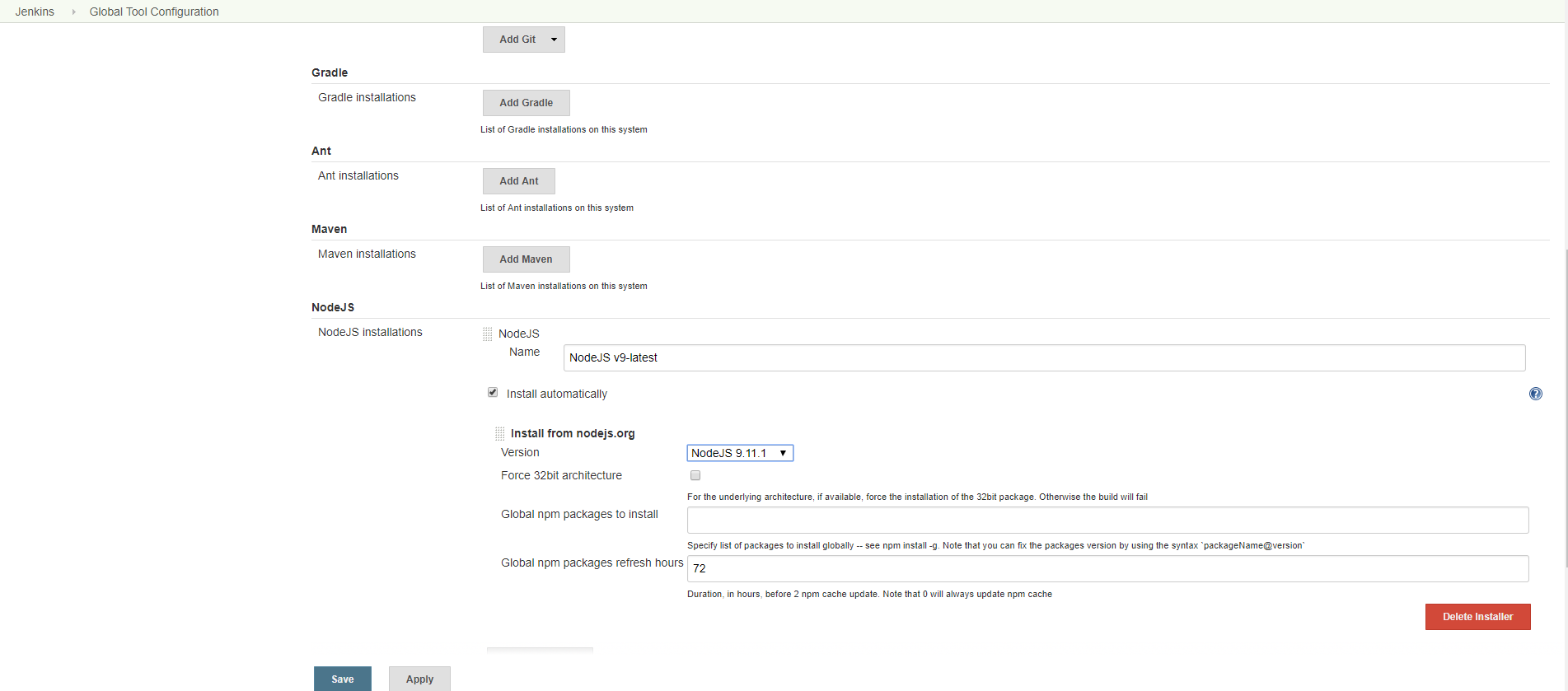
The steps to install NodeJS plugin on Jenkins:

Step 1: Find NodeJS plugin in tab Plugin Manager

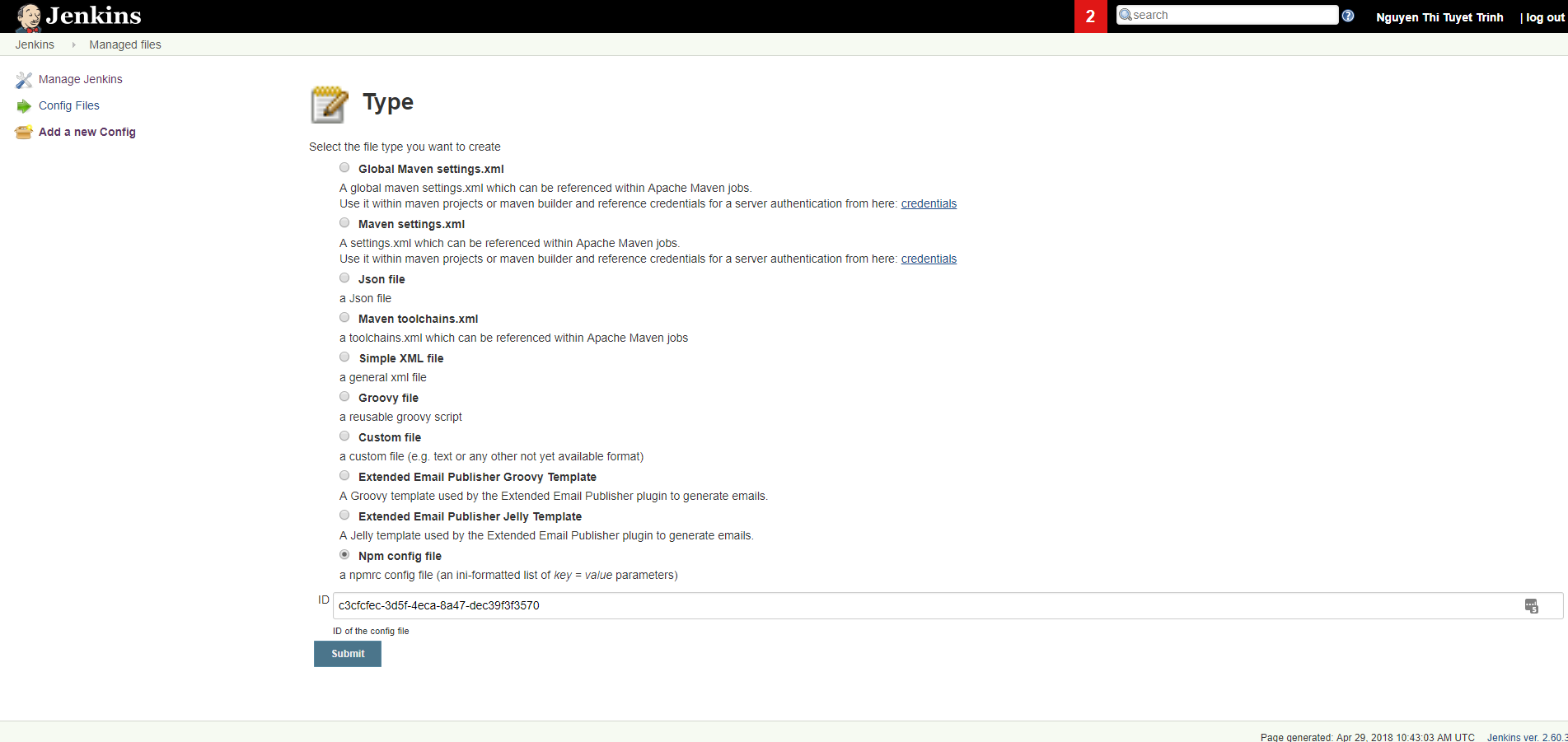


Install NodeJS plugin and restart Jenkins.

Step 2: Go to Global Tool Configuration, set up a NodeJS installation



Step 3: Go to Managed Files, add a new NPM configuration file



NPM means Node.js Package Manager, a software registry. Until now, NPM contains over 600.000 packages (building block of code) with approximately 3 billion download per week [“TODO: dẫn chứng ở tab reference”] NodeJS application uses npm to install all required libraries needed to initialize by reading the package.json file and build the product.

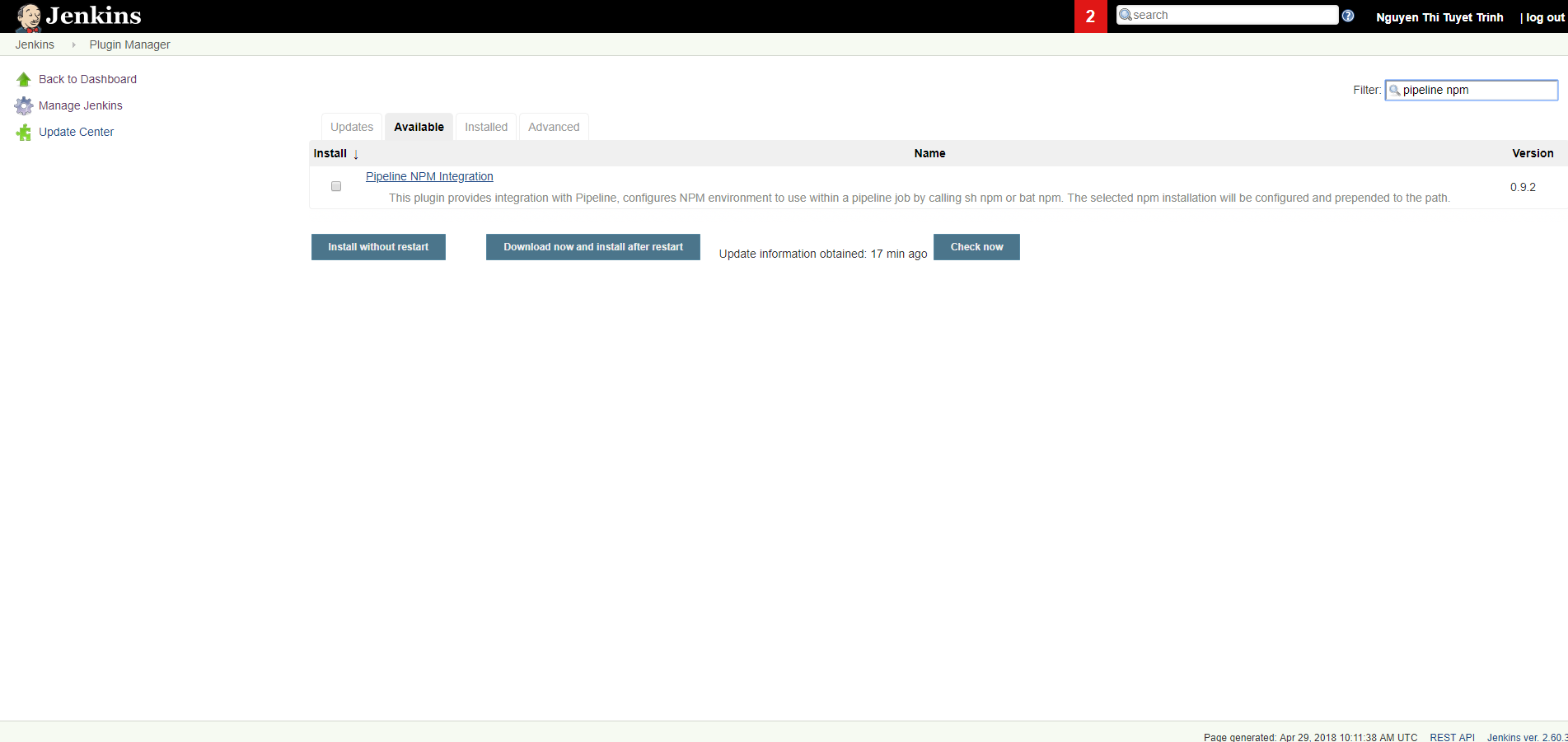
### Set up the first Pipeline

Jenkins Pipeline is a suite of plugins which support implementing and integrating continuous delivery pipelines into Jenkins “TODO: add vao reference”

In order to setup and run CICD in Jenkins, we will create a Pipeline Job. Because of we set up CICD for NodeJS project, thus we need to install Pipeline NPM integration plugin before set up Pipeline Job.

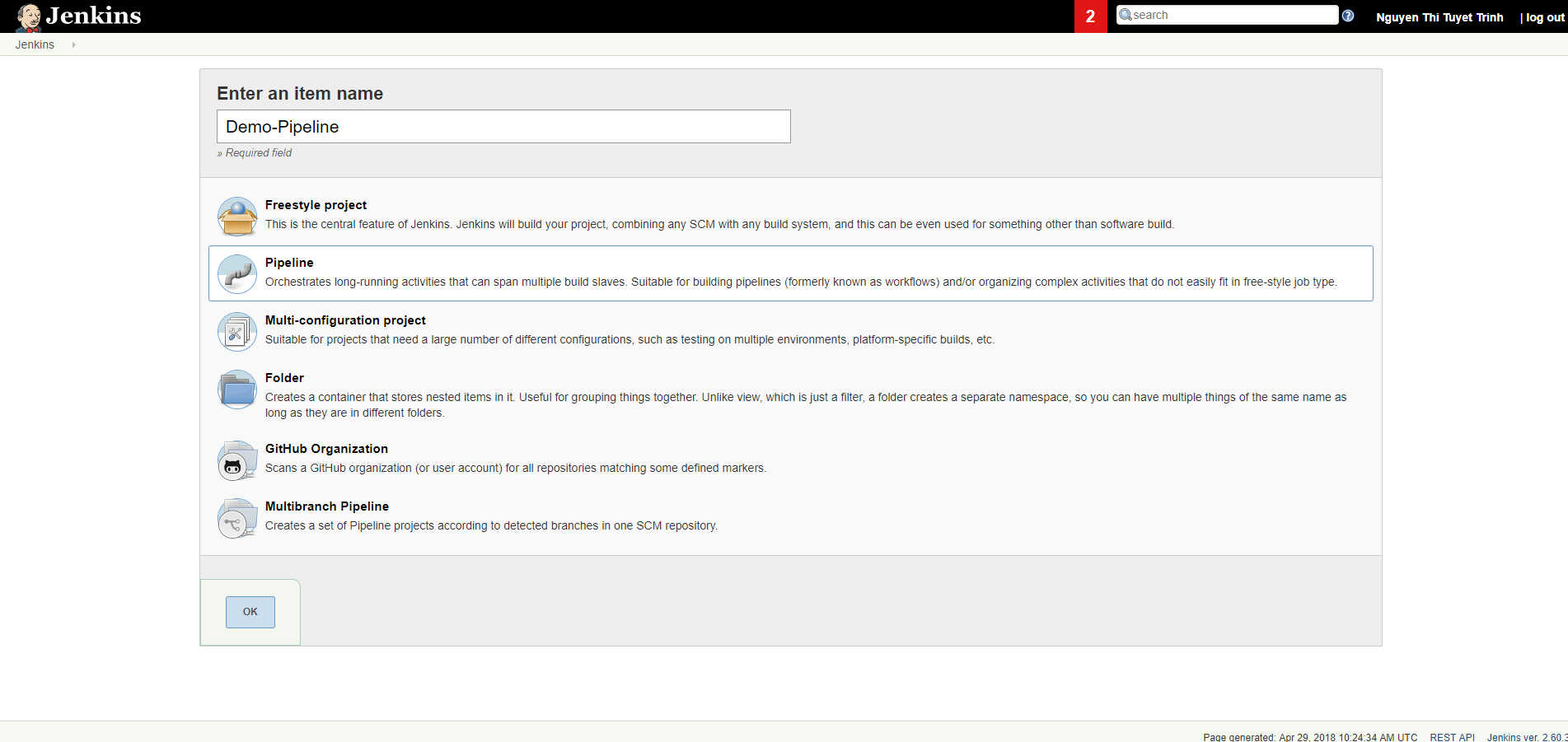
The steps to install Pipeline NPM integration:

Step 1: Go to plugin manager, install Pipeline NPM integration



After installed Pipeline NPM integration, we can create the first Pipeline Job as bellow:

Step 2: Create a new Pipeline Job



### Configure pipeline in Jenkins

In this step, we will configure the Demo-Pipeline from previous step.

To configure pipeline in Jenkins, we go to Pipeline Job and select Config

### Configure pipeline in Jenkins

In this step, we will configure the Demo-Pipeline from previous step.

Because of using Pipeline Job, we can able to define and use jenkinsfile, which help us to define all stages inside a job. The advantages of using jenkinsfile are:

* Centralized source code management.
* Jenkinsfile can be contained in project code as an independence file, and control by Git.
* Jenkinsfile can be review/edit/audit by any team member of Project.

Jenkinsfile using Pipeline script, which can be generated by Generate Pipeline Script function in Jenkins. To use Generate Pipeline Script function, we can go to Pipeline Job, choose Pipeline Syntax, after that go to tab Snippet Generator.

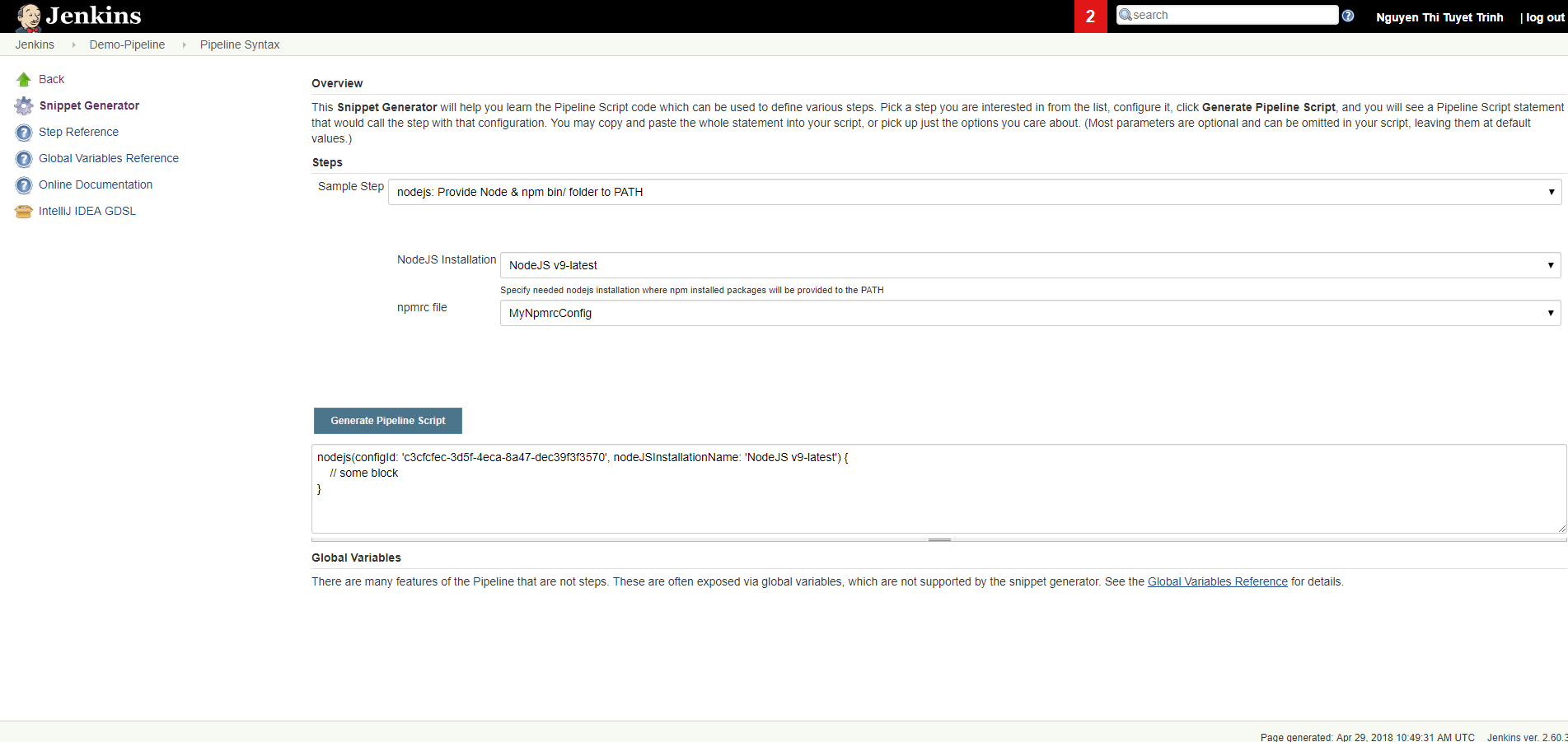


Figure 5.3.7: Snippet Generator screen

### Configure Webhook

The targets of this demo are:

* Using Jenkins to apply CICD for Project
* Using Git to manage source code
* Trigger build on Jenkins whenever new code change on branch master

To perform the third statement: Trigger build on Jenkins whenever new code change on branch master, we have to configure Git to notify an event to Jenkins when a new commit code on branch master. Git support us to do it by Webhook.

To set Webhook up, we:

* Step 1: Go to project repository on Git
* Step 2: Go to repository setting, tab webhook
* Step 3: Add new webhook (if does not exist) or edit current webhook. Because of Git webhook need a target URL of Jenkins to send s POST request, thus we used ngrok as a windows application tool to expose local Jenkins to a public URL

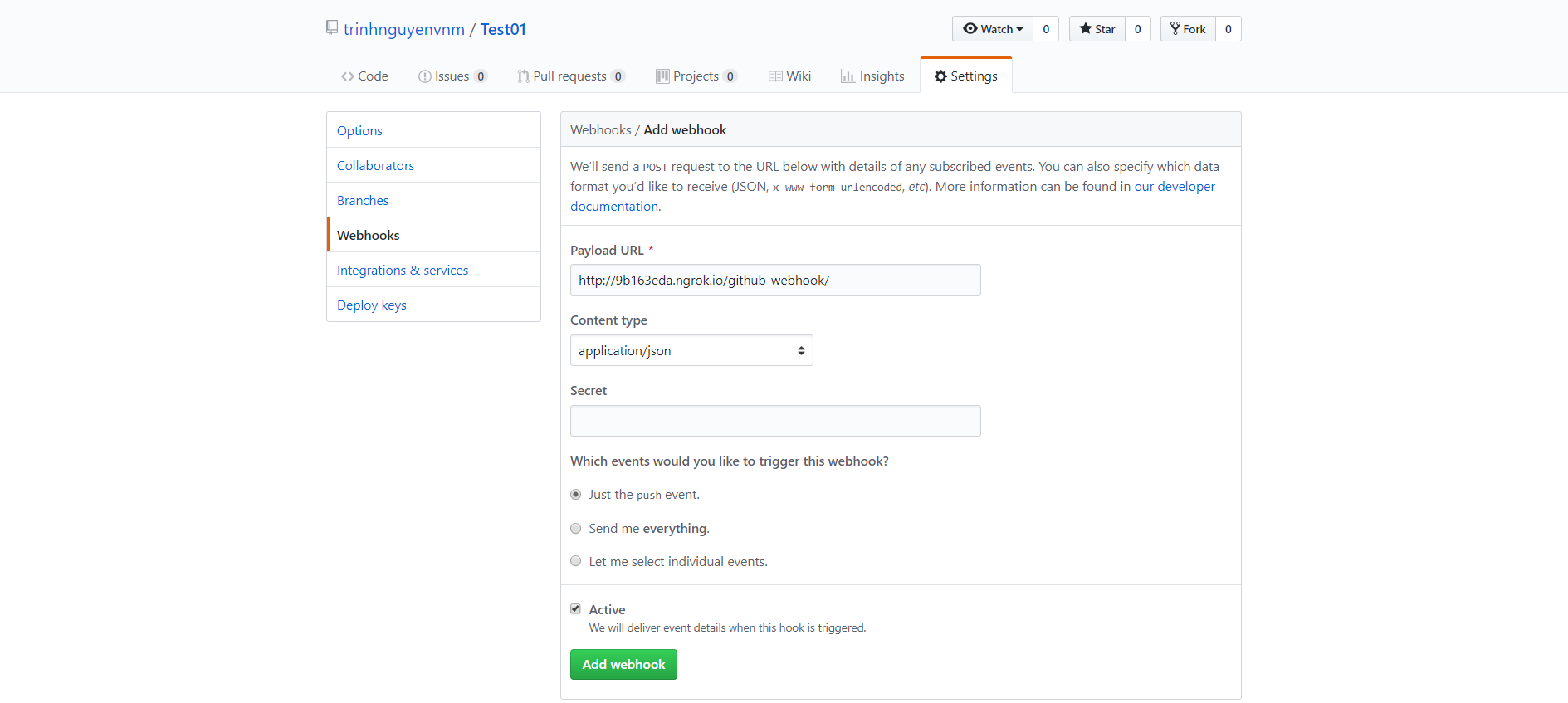


Figure 5.3.8: Configure webhook on Git

### Set up Apache HTTP Server

For Continuous Deployment, we use Apache HTTP Server image on Docker. The process is as diagram bellow

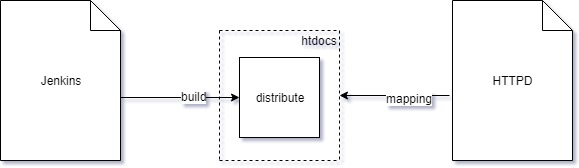


Figure 5.3.9 Deployment process diagram

In CI process, Jenkins will build product after finished all stages: checkout code, merge code (if any) run npm install to install all require libraries, run Unit test. The result of build process is a web deployable folder (folder distribute).

To doing CD, we use Apace HTTP Server (httpd). When create httpd container in Docker, we map folder htdocs (source folder) with distribute folder. Apache HTTP Server will use distribute folder to host a web application.

The result of this process is: whenever CI finished its process, there will be a new web application, which ready to test hosted by Apache HTTP Server.

### Configure jenkinsfile

Jenkinsfile is a text file, which contains the definition of a Jenkins Pipeline. Jenkinsfile always checked into source control.

Because of Jenkinsfile contains the definition of a Jenkins Pipeline, thus the structure of Jenkinsfile will be started with the definition of Pipeline

In Jenkinsfile, there are 2 ways to write down script: using “Declarative Pipeline” and “Scripted Pipeline”

1. Declarative Pipeline

Declarative Pipeline using some simple syntax based on some built in methods / functions to create stages inside. This is a new way to write script. It helps developer easily to write down Jenkinsfile, using some simple built in functions. Declarative Pipeline support well small project with basic stages.



Figure 5.3.10.1: Declarative Pipeline sample

1. Scripted Pipeline: Scripted pipeline is the traditional way to write Jenkinsfile. Scripted Pipeline using Groovy script – a technique advance, to define tasks or complicated logic

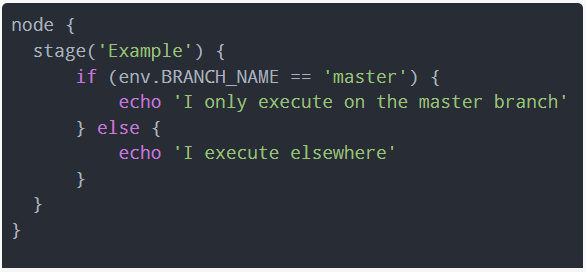


Figure 5.3.10.2: Scripted Pipeline sample

The advantages of Scripted Pipeline compare with Declarative Pipeline:

* + Most functionality provides by Groovy language are available to users of Scripted Pipeline. It means that users have a very expressive and flexible tool to use in Continuous Delivery pipelines
  + Scripted pipeline provides Flow control. Therefore, users can use if/else conditionals, such as:

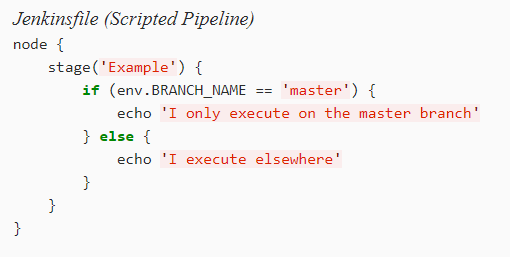


Figure 5.3.10.3: Scripted Pipeline Flow control sample

* + Scripted pipeline flow control can be managed with Groovy’s exception handling support. When steps fail, they can throw an exception, thus user can find the issues faster

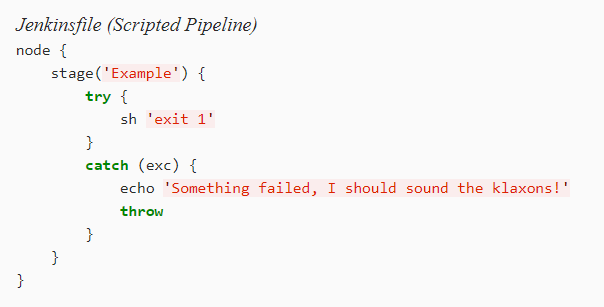


Figure 5.3.10.4: Scripted Pipeline Flow control exception handling sample

Because of these advantages, in this thesis, the author used Scripted pipeline in Jenkinsfile.

In Pipeline, step is the most fundamental part. Basically, steps tell Jenkins what to do for both Declarative and Scripted Pipeline. “Stages” is symbol of steps in Pipeline.

Stages: Containing a sequence of one or more stage directives, the stages section is where the bulk of the "work" described by a Pipeline will be located. (trích nguồn)

“Stages” contains a list of “stage”, which define what should Jenkins do, step by step. The recommendation is: “stages” contains at least one stage directive for each discrete part of the continuous delivery process, such as Build, Test, and Deploy. In this thesis demonstration, the author use 4 stage:

* Stage 1: “Clone code”: in this stage, Jenkins will check out the code of pre-defined url in master branch. The result of this stage will be updated to current build result.
* Stage 2: “npm install”. Because of demonstration project based on NodeJS, thus the “npm install” command need to be run at least at the first time to install all required libraries. This step can be ignored when the JSON package was stable.
* Stage 3: “Run Unit Test”. In this step, Unit Test will be run. The result of this step is the UT document, which contain all result of UT.
* Stage 4: “Deployment”. In this stage, the project will be built and deploy by Apache. The result of this step is new version of product, that deployed and ready use.

The Jenkinsfile will be checked in into source code.

## Result

This sub-chapter bring the result after applying the CICD into case study.

### Quality

\*Chưa viết \*

### Time and Price

\*Chưa viết \*

## Lesson learn

\*Chưa viết \*

### The structure of branch should be…

\*Chưa viết \*

### Do not merged code while CI failed

\*Chưa viết \*

### CD

\*Chưa viết \*

1. Conclusion

The Conclusion chapter.

# Conclusion

## Before CICD

\*Chưa viết\*

## After CICD

\*Chưa viết\*

References or Bibliography (if any)

Use the “biblio” style to format references.

[STT]: 2.2: Continuous Integration: FOWLER Martin, Continuous Integration. [online]. 2006. Available from: <https://www.martinfowler.com/articles/continuousIntegration.html>

[STT]: 5.3.4: Node.js Package Manager, what is NPM? [online]. Available from: <https://docs.npmjs.com/getting-started>

[STT]: 5.3.5: Jenkins Pipeline, what is Pipeline? [online]. Available from: <https://jenkins.io/doc/book/pipeline/>

[STT]: 5.3.10: Jenkins Agent, what is Jenkins Agent? [online]. Available from: <https://jenkins.io/doc/book/pipeline/syntax/#agent>

[STT]: 5.3.10: Jenkins Stages, what is Jenkins Stages? [online]. Available from:

<https://jenkins.io/doc/book/pipeline/syntax/#stages>

Appendices (if any)

Copyright Acknowledgements (if any)