TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN - BỘ MÔN CÔNG NGHỆ PHẦN MỀM

Topic 2:

DevOps - Frameworks

Giảng viên: Phạm Thị Thương – Bộ môn CNPM – Khoa CNTT
Email: ptthuong@ictu.edu.vn

Mục tiêu

- Trình bày về các DevOps Framework
- Các cách tiếp cận thực tiễn tốt nhất được sử dụng trong DevOps.

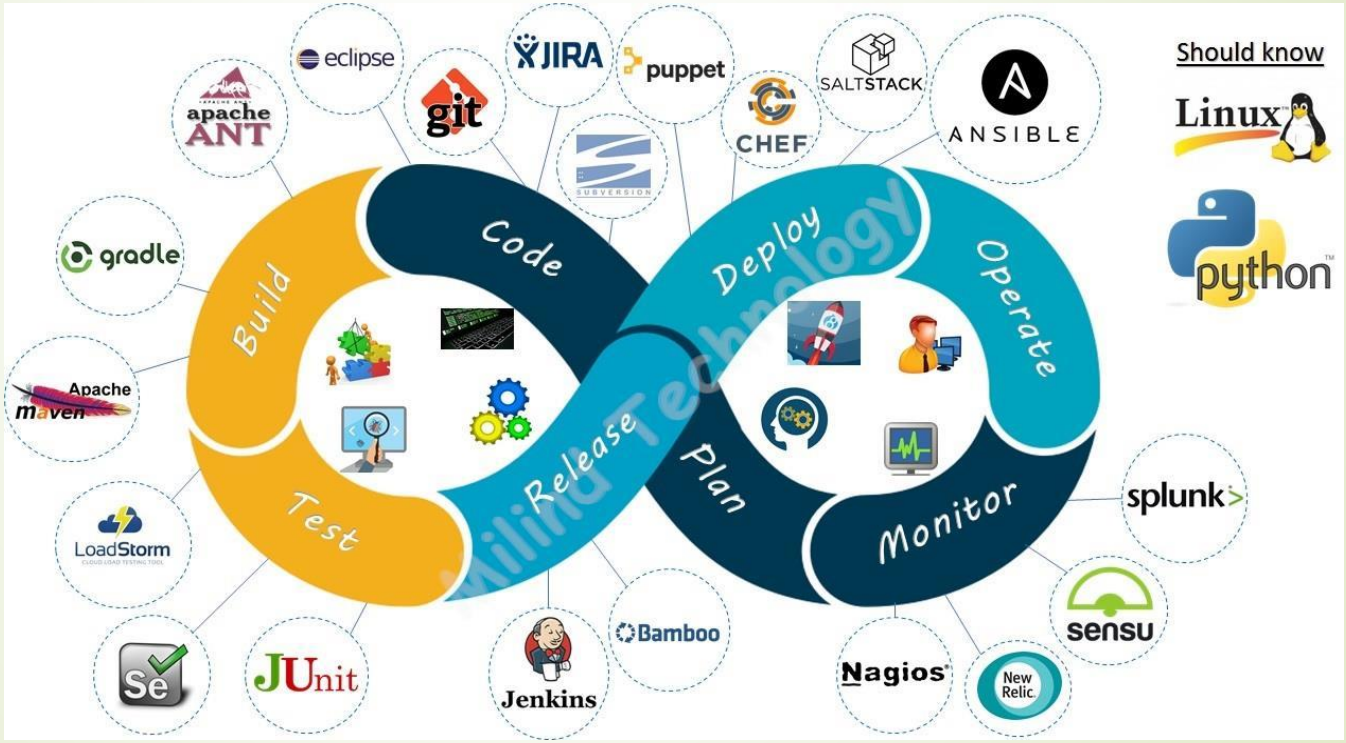
Nội dung chính

I. **(Full) DevOps framework**
II. (Core) DevOps framework
III. (Core) DevOps processes

I. (Full) DevOps Framework

1. Source code management
2. Build management
3. Test
4. Repository management
5. Release management
6. Configure Management - Cấu hình cơ sở hạ tầng (IaC - Infrastructure and Configuration as code)
7. Monitoring, Loging - Giám sát quá trình vận hành app
8. Lập kế hoạch – Lập kế hoạch cải tiến và quay vòng

I. (Full) DevOps Framework



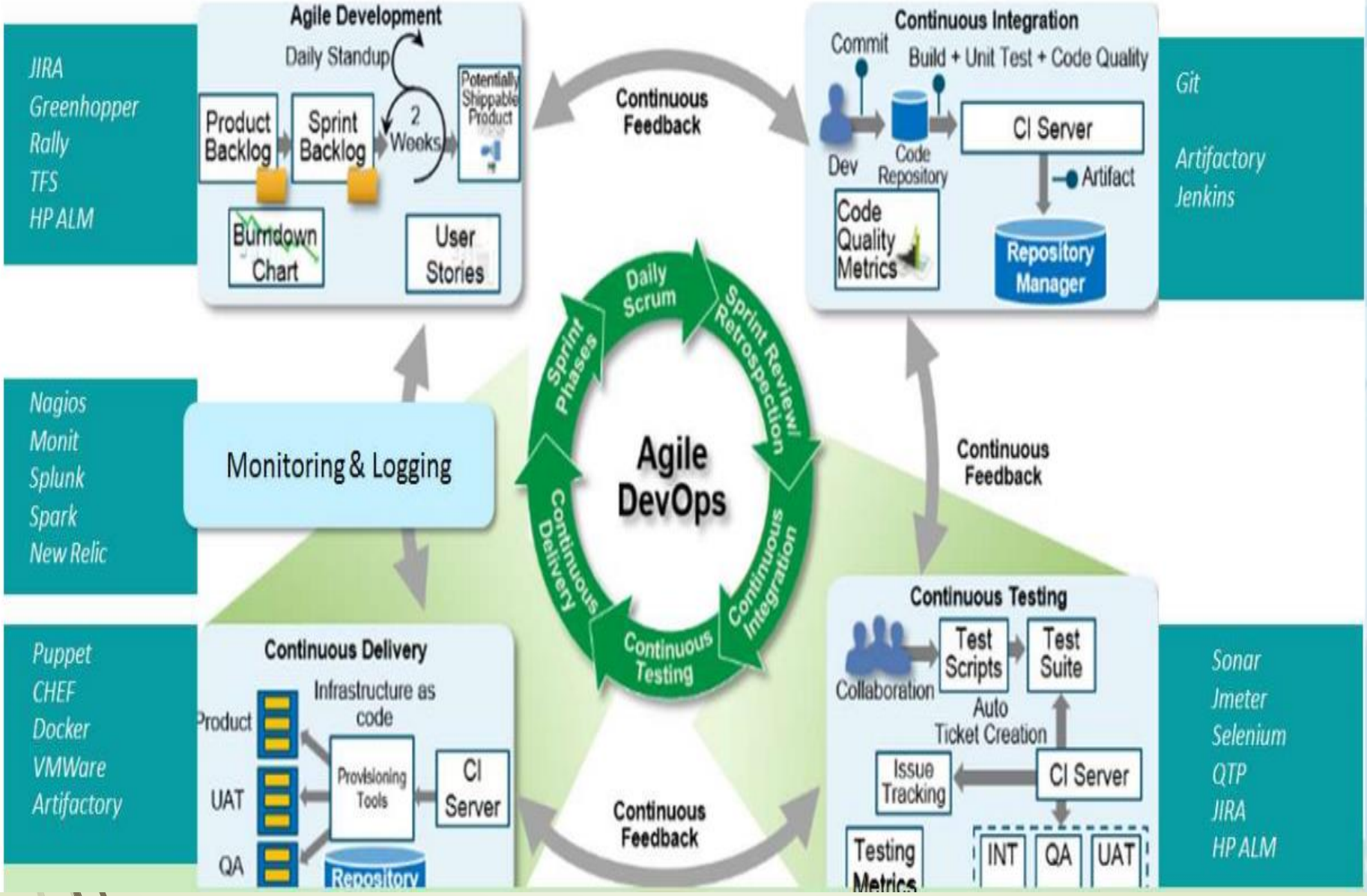
I. (Full) DevOps Framework

- DevOps: tools
  - **Source code management**: Git, GitHub, Subversion, and Bitbucket
  - **Build management**: Maven, Ant, Make, and MSBuild
  - **Testing tools**: JUnit, Selenium, Cucumber, and QUnit
  - **Repository management**: Nexus, Artifactory, and Docker hub
  - **Continuous integration**: Jenkins, Bamboo, TeamCity, and Visual Studio
  - **Configuration provisioning**: Chef, Puppet, Ansible, and Salt
  - **Release management**: Visual Studio, Serena Release, and StackStorm
  - **Cloud**: AWS, Azure, OpenShift, and Rackspace
  - **Deployment management**: Rapid Deploy, Code Deploy, and Elastic box
  - **Collaboration**: Jira, Team Foundation, and Slack
  - **BI/Monitoring**: Kibana, Elasticsearch, and Nagios
  - **Logging**: Splunk, Logentries, and Logstash
  - **Container**: Linux, Docker, Kubernetes, Swam, AWS, and Azure

I. (Full) DevOps Framework

- Ví dụ:
  - DevOps – A Representative Reference Architecture
    - Xem hình (dưới)

Agile Methodology



Nội dung chính

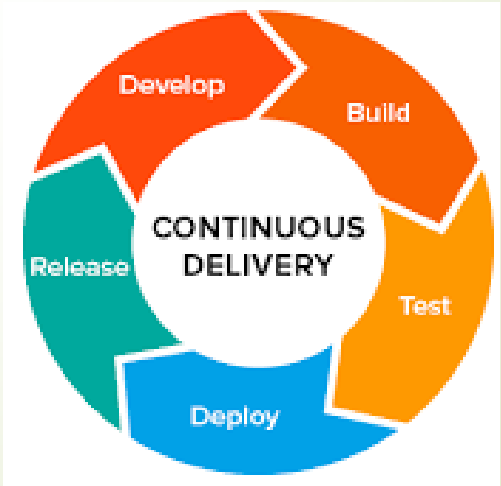
I. (Full) DevOps framework

II. **Continuous Integration (CI) & Continuous Delivery (CD)**

III. (Core) DevOps processes

II. CI&CD

- Khách hàng tự vận hành sản phẩm
  - Software not is a service
- Các giai đoạn DevOps được hợp nhất qua tiến trình CI & CD sử dụng Jenkins pipeline
  - CI, CD?

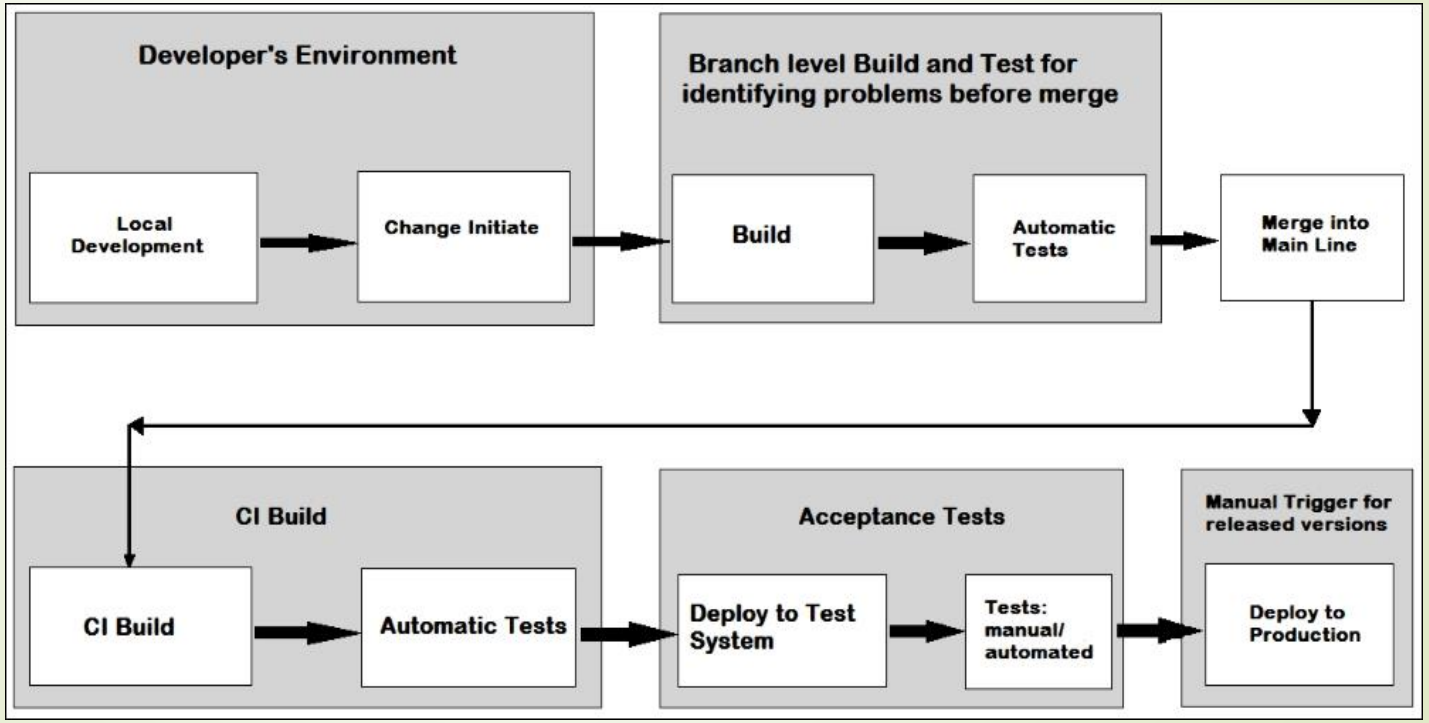


II. Continuous Integration (CI) & Continuous Delivery (CD)

- CI/CD: các tiến trình quan trọng của DevOps
  - Nhằm phát hành phần mềm đến người dùng một cách nhanh chóng và hiệu quả.
  - CD: mở rộng CI, đảm bảo:
    - Phiên bản mới nhất của phần mềm là sẵn sàng cho việc chuyển giao
    - Tự động chuyển dịch phiên bản đến môi trường QA, Product
      - Xem hình (dưới)

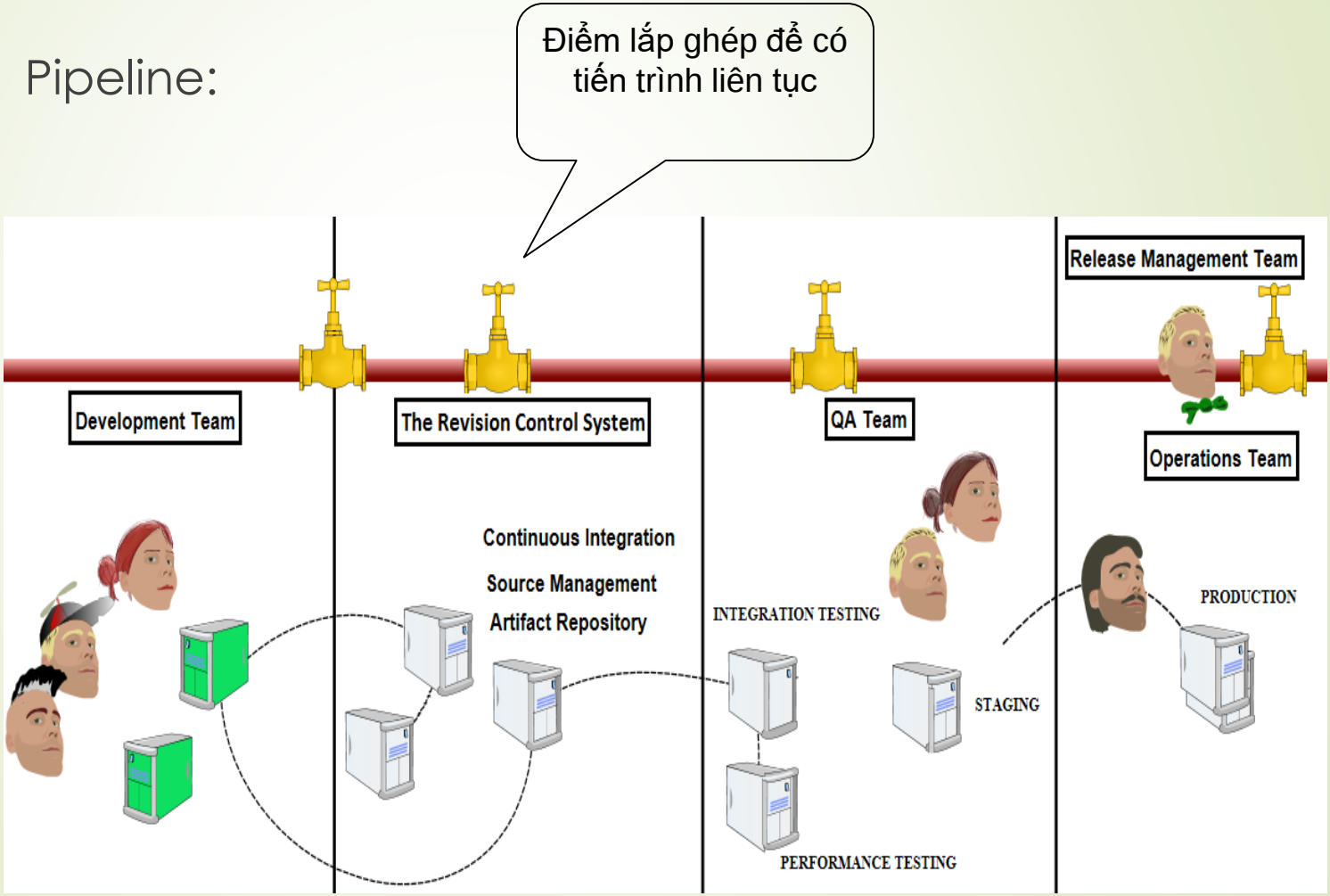
II. CI &CD

- Tiến trình tích hợp (CD):



II. CI&CD

- Pipeline:



Nội dung chính

I. (Full) DevOps framework

II. Continuous Integration (CI) & Continuous Delivery (CD)

III. **DevOps processes**

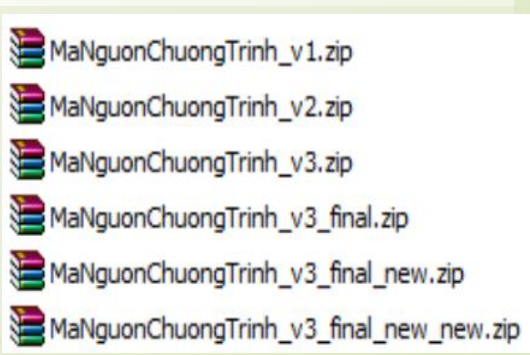
III. (Core) DevOps Process

1. Source code management – **Git&GitHub**
2. Build management – **Maven**
3. Source code review **– Gerrit**
4. Repository management – **Nexus**
5. Test Automation – **Selenium**
6. Continuous deployment – **Pipelines**
7. Continuous integration - **Jenkins**

Mục đích phần này sẽ giới thiệu sơ lược từng giai đoạn (chủ đề)/tiến trình và chuẩn bị công cụ/giai đoạn.

1. Source Code Management

- Why?
  - Qua (t) **mã nguồn** của **c.trình ngày càng nhiều** => nhiều version:
    - **Lập trình cá nhân**: => Vấn đề nảy sinh:
      - Lưu trữ chồng chéo, dư thừa, tốn không gian
      - Khó kiểm soát, không xác định được những thay đổi mã nguồn trong từng version
      - Khó tìm kiếm, so khớp xác định những thay đổi đã tạo với version trước đó
      - Đặt tên không đúng định dạng, ..
    - Ví dụ: xem hình



1. Source Code Management

- Why?
  - **Lập trình nhóm**:
    - Gặp hạn chế của lập trình đơn lẻ (*x lần) +
      - Phải copy, gửi **mã nguồn** cho nhau (thủ công)
      - Phải **thông báo** những update thường xuyên cho nhau
      - Dễ **nhầm lẫn, sai sót, chỉnh sửa trùng**, viết đè, **xóa** những **phần** code **chỉnh sửa của** người khác nếu việc quản lý, giám sát không tốt
      - Vấn đề đồng bộ mã, tích hợp (thủ công) → khó khăn, mất (t) đặc biệt với p.mềm lớn, phức tạp,…

    ⇒ Thách thức: khi cộng tác ↑, nhiều tổ chức gồm 10, 100, … teams, phân bố khắp nơi trên TG.

1. Source Code Management

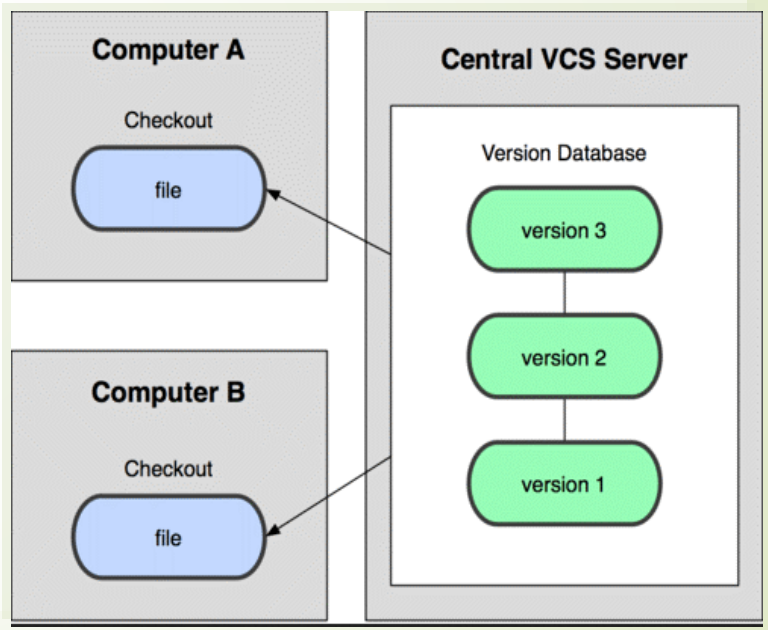
- Why?
  - ⇒ Sử dụng Hệ điều khiển phiên bản (VCS) để quản lý mã nguồn của dự án
    - ~ giải pháp cho những vấn đề nêu trên!

1. Source Code Management

- Why?
  - Version Control System (VCS):
    - Lưu trữ và ghi nhận sự thay đổi của các tập tin, mã nguồn theo thời gian,
      - ⇒ Giúp theo dõi mọi cập nhật sửa đổi mã nguồn của các thành viên trong suốt quá trình làm dự án.
    - 2 loại mô hình VCS:
      - ⇒ **Mô hình tập trung (Centralized version control - CVC)**
      - ⇒ **Mô hình phân tán (Distributed version control - DVC)**

Mô hình tập trung (CVC)

- Ví dụ: SVN (Subversion)
  - Sử dụng kiến trúc Client-Server.
    - **Server**: lưu kho dữ liệu (Repository) chung: lưu trữ tất cả các tập tin, mã nguồn, các thay đổi mã nguồn, …
    - **Các client**: kết nối đến server để lấy phiên bản mới nhất của mã nguồn về máy (check out) và cập nhật thay đổi, sau đó commit lên server.

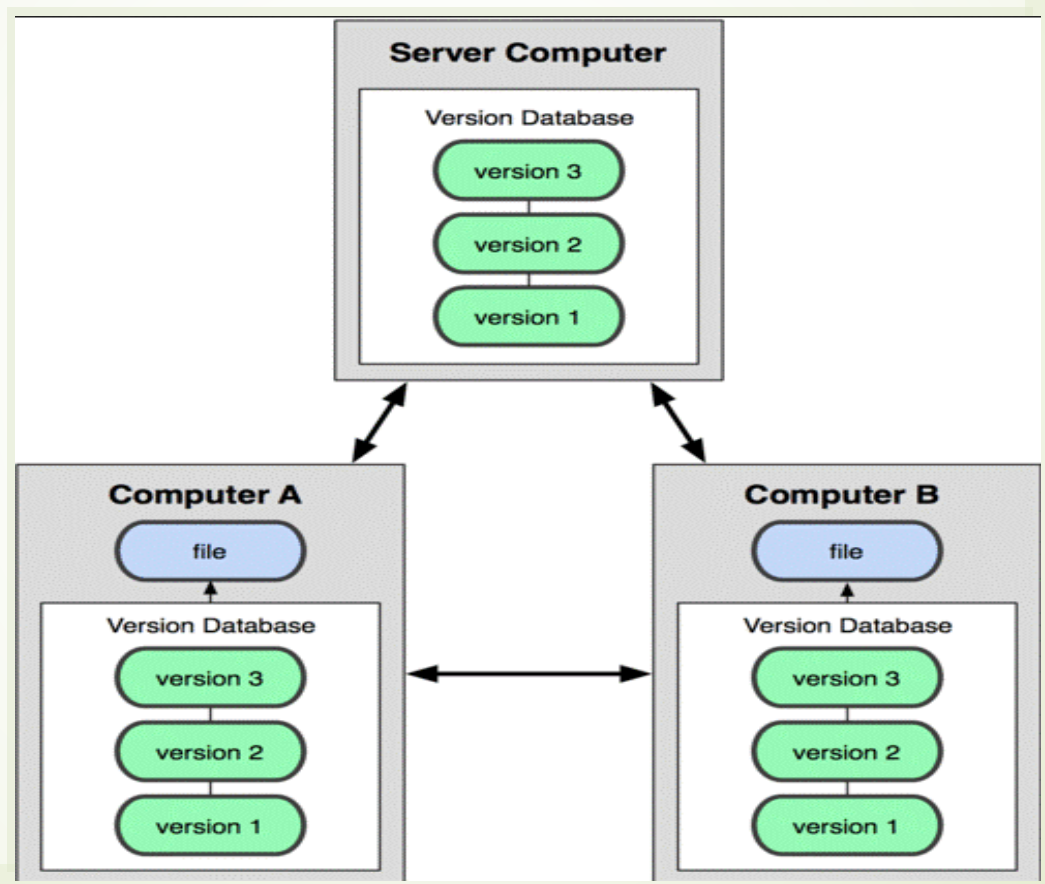


Mô hình phân tán (DVC)

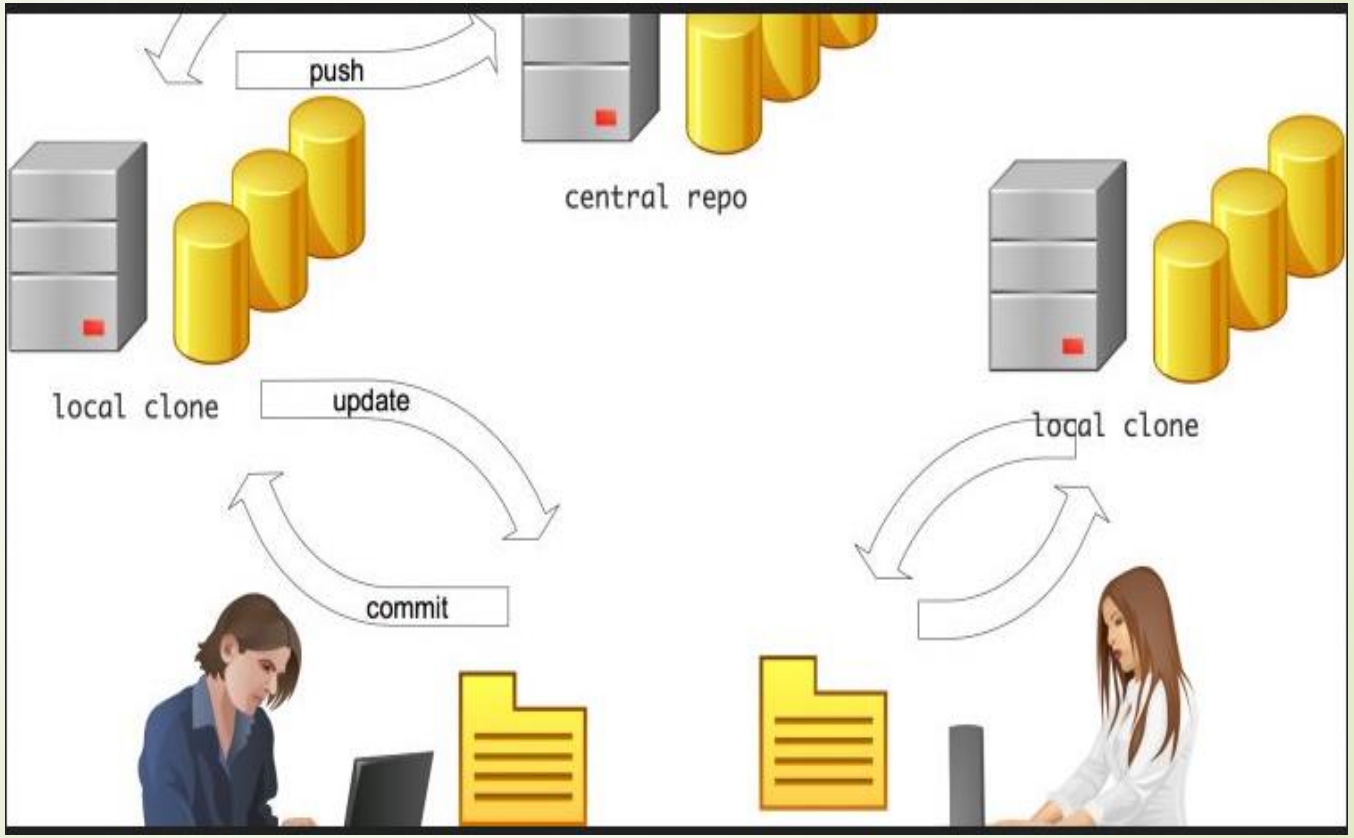
- Ví dụ: Git
  - Khi client lấy mã nguồn phiên bản mới nhất từ server về thì hệ thống sẽ sao chép (clone) cả kho mã nguồn (repository) đến client
    - Client có thể trực tiếp làm việc trên kho cục bộ này bất cứ lúc nào mà ko cần connect đến server

      => Linh hoạt hơn khi cập nhật mã nguồn lên server
    - Khi commit, mọi thay đổi từ kho cục bộ sẽ được đồng bộ đến kho chính (remote repository).

Mô hình phân tán (DVC)



Mô hình phân tán (DVC)



1. Source Code Management

- Lợi thế:
  - Developers
    - **Tương tác, phối hợp**, chia sẻ mã nguồn cho **nhau** dễ dàng
  - Lưu mã nguồn tập trung:
    - Kiểm soát tốt mã nguồn, tìm kiếm dễ dàng
    - Tránh lưu trữ dư thừa, chồng chéo, viết đè
    - **Lưu trữ** mọi **thay đổi** trong các version, xác định các thay đổi một cách dễ dàng
      - Ví dụ: lịch sử thay đổi: xem hình (dưới)

Commits

All branches ▾

Find commits

Author	Commit	Message	Date	Builds
Emma Paris	48382de	updated menu my-updates	3 minutes ago	
Emma Paris	94e226b	team_contact_info edited online with Bitbucket	an hour ago	
Emma Paris	6f19a87 M	Merged in my-updates (pull request #1) updating menu	an hour ago	
Emma Paris	635e8ae	adding data gathered	3 hours ago	
Emma Paris	7eeea78	team_contact_info edited online with Bitbucket	3 hours ago	
Emma Paris	7324cdd	team_contact_info created online with Bitbucket	3 hours ago	
Emma Paris	ab11dca	updating menu	yesterday	
Emma Paris	1bf69bf	adding team photo	yesterday	

1. Source Code Management

- Kỹ thuật phân nhánh:
  - Quan trọng:
    - Dùng để quản lý các phiên bản mã nguồn, cho phép lập trình phân tán, lập trình // một cách độc lập trên nhánh tương ứng được phân công.
  - Dự án: Nhiều nhánh:
    - **Nhánh chính:**
      - Kiểm duyệt code, lưu trữ versions sẵn sàng phát hành.
    - **Các nhánh phụ:**
      - Lưu các sửa đổi của version tương ứng.
        - Mỗi loại bảo trì sửa đổi ~ 1 loại nhánh

      => Những loại bảo trì sửa đổi gì?

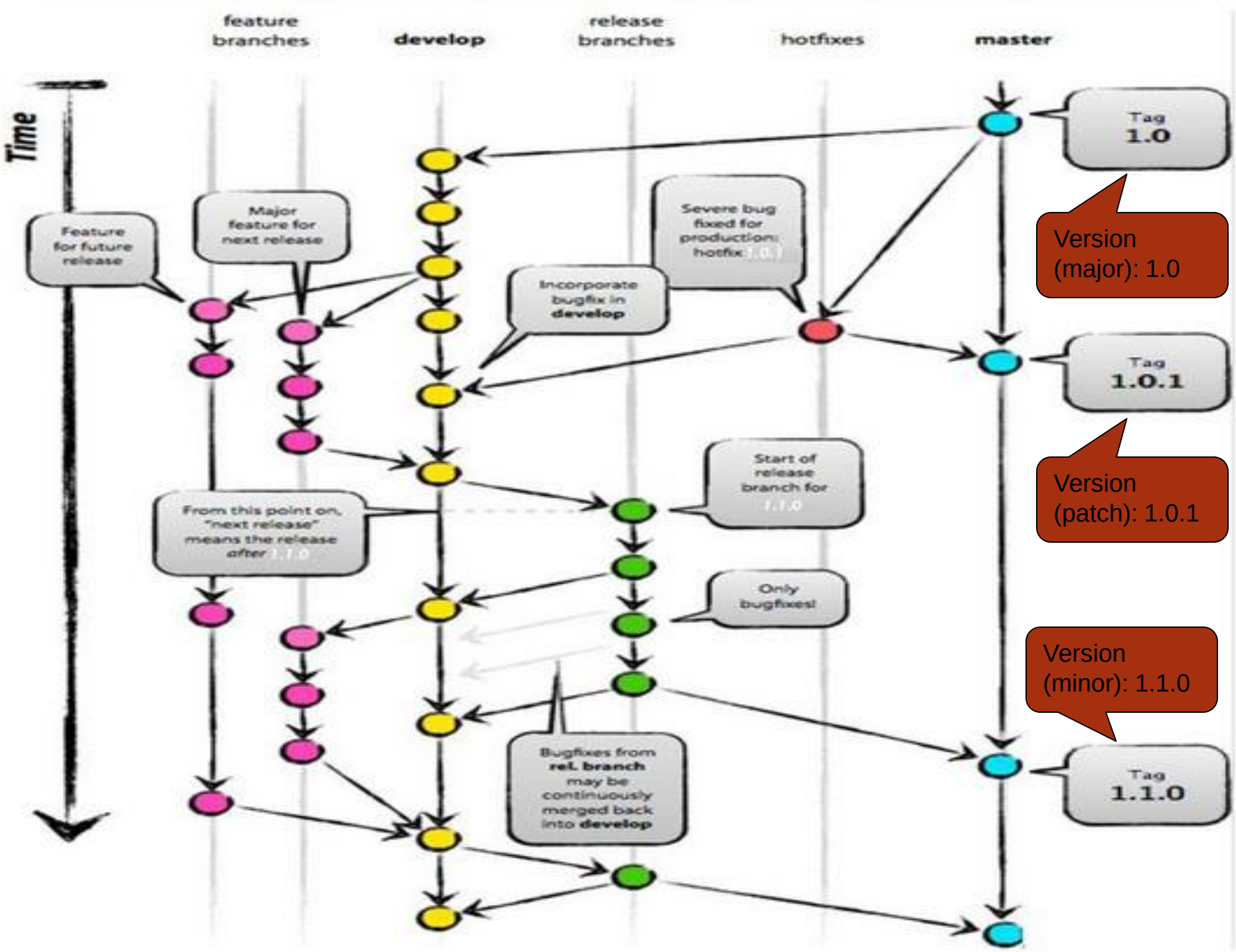
1. Source Code Management

- Kỹ thuật phân nhánh:
  - Những loại bảo trì sửa đổi?
    - Nâng cấp (minor): sửa đổi, bổ sung new feature
    - Thích nghi (major)
    - Vá lỗi (patch) hoặc fix lỗi khẩn cấp (hotfixes),
    - …
  - Tương ứng sẽ có các loại nhánh khác nhau.

1. Source Code Management

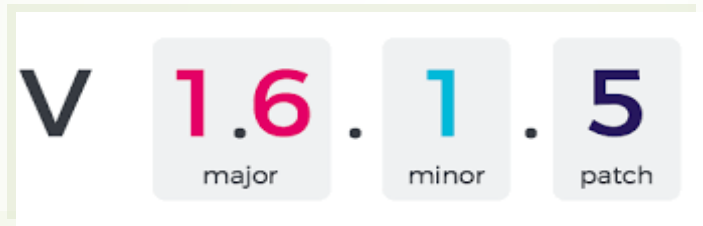
- Kỹ thuật phân nhánh
  - Ví dụ:
    - Mô hình phân nhánh được đề xuất bởi Vincent Driessen, 2012
      - Kho code của OpenCPS cũng sử dụng mô hình này
      - Xem hình (dưới)

Vincent Driessen (a software engineer, at Netherlands)


feature branches
develop
release branches
hotfixes
master
Time
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix
Incorporate bugfix in develop
Tag 1.0
Version (major): 1.0
Tag 1.0.1
Version (patch): 1.0.1
Start of release branch for
From this point on, "next release" means the release after
Only bugfixes!
Version (minor): 1.1.0
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.1.0

Mô hình phân nhánh - Vincent Driessen, 2012

- Cách đánh số hiệu phiên bản



Mô hình phân nhánh - Vincent Driessen, 2012

1. Các nhánh chính
   - **Master (origin/master),**
     - Lưu các phát hành/version ổn định của s.phẩm.
       - Còn gọi là nhánh chính (core)
   - **Develop (origin/develop)**
     - Lưu các thay đổi (major, minor, patch, hotfixes) cuối cùng cho phát hành tiếp theo.
       - Còn gọi là "integration branch".

Mô hình phân nhánh - Vincent Driessen, 2012

2. Các nhánh phụ (hỗ trợ)

i. **Feature branches**

ii. **Release branches**

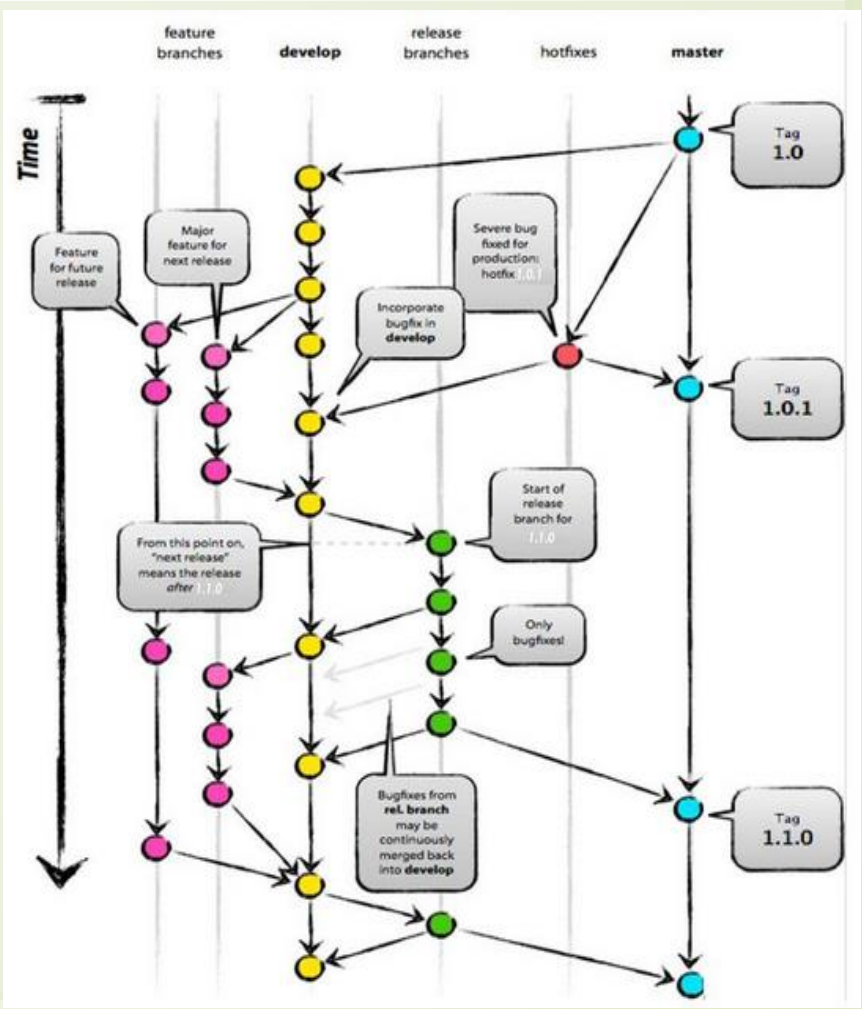
iii. **Hotfix branches**

⇒ Mỗi nhánh có một mục đích cụ thể và tuân theo các "strict rules" gồm:

- Nhánh gốc của chúng là gì?
- Chúng phải nối vào những nhánh nào?
- Cách đặt tên các update code trên nhánh?

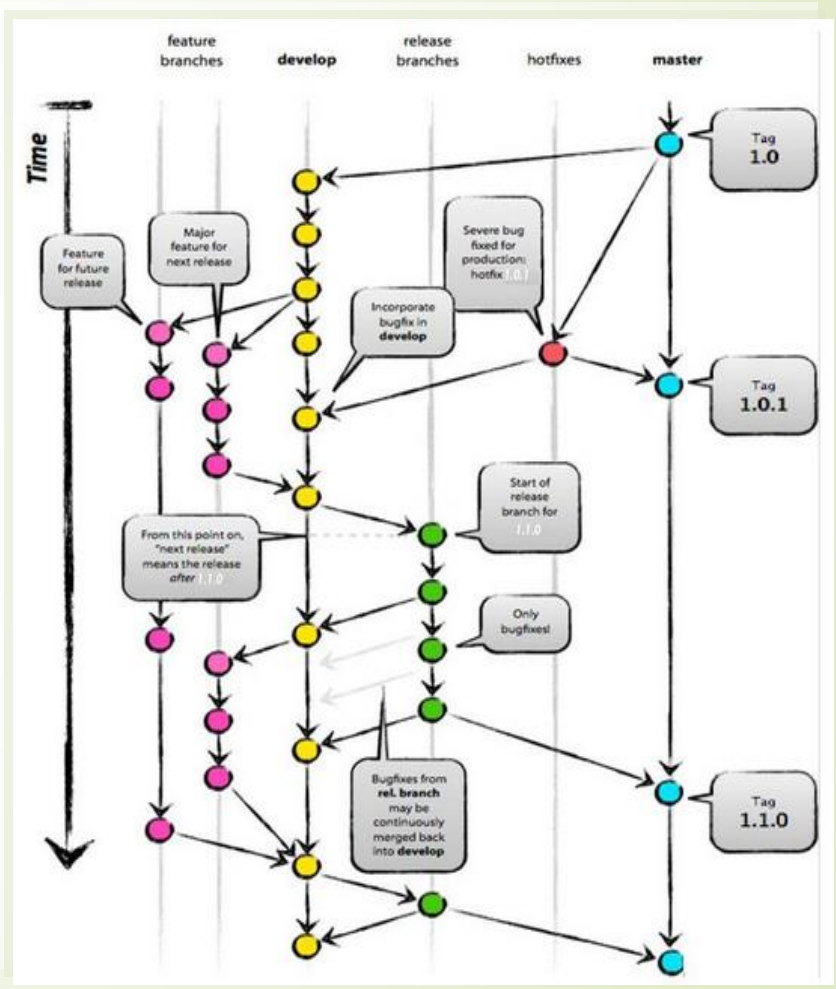
i. Feature branches

- Rules:
  - Original branch:
    - develop
  - Must merge back into:
    - develop
  - Branch naming convention
    - anything except
      - master, develop, release-*, or hotfix-*



i. Feature branches

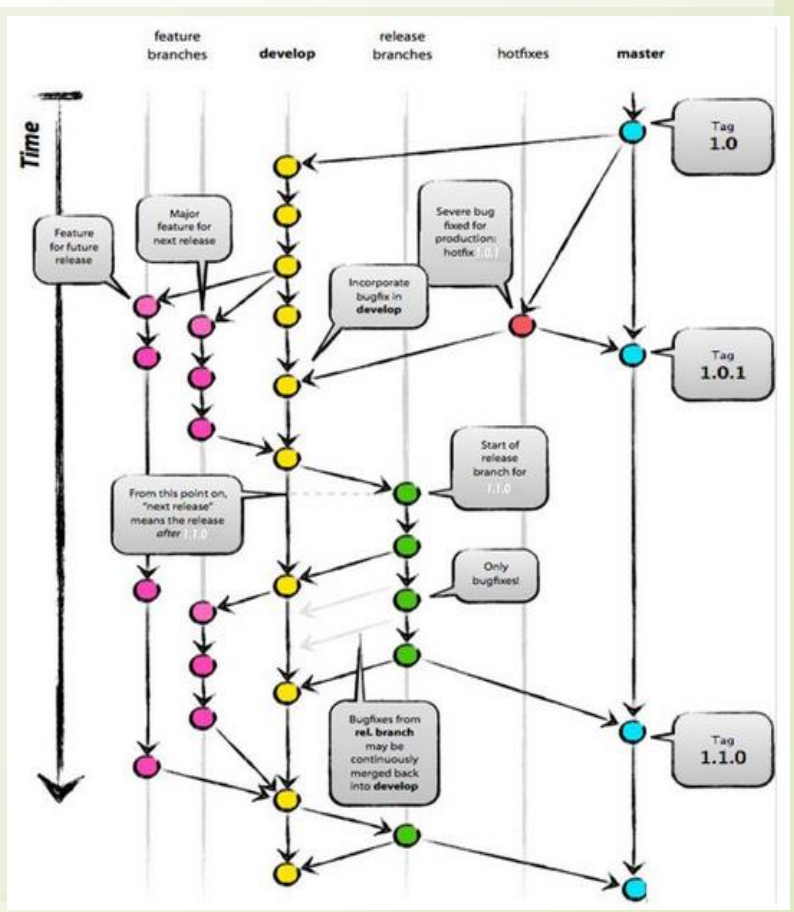
- Are used to:
  - Phát triển các new features cho phát hành sắp tới (minor), hoặc nâng cấp phiên bản chính (major)
    - Còn gọi là các topic branches



ii. Release branches

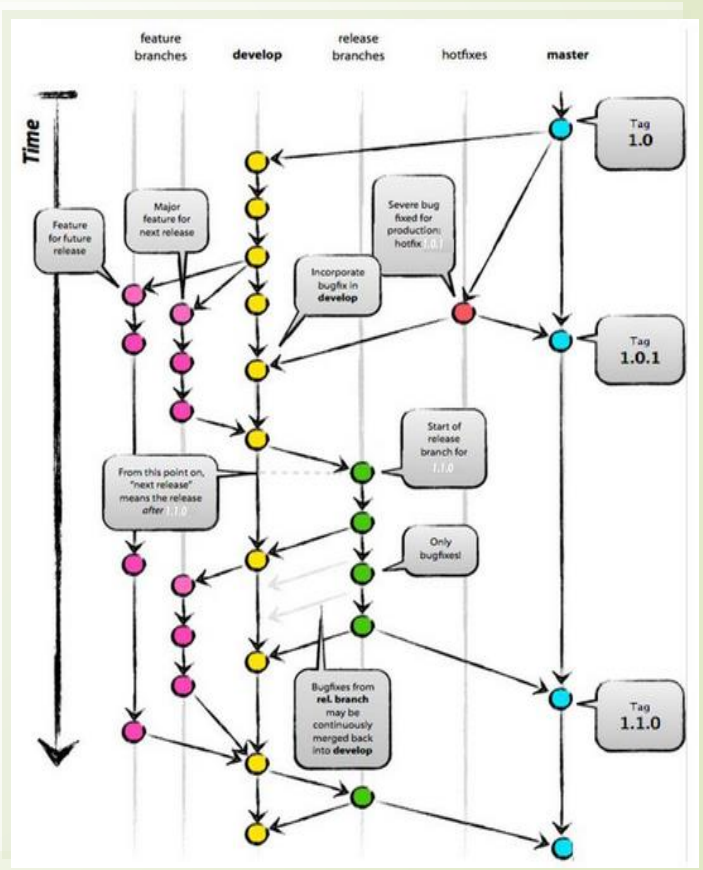
- Rules
  - Original branch:
    - develop
  - Must merge back into:
    - develop and master
  - Branch naming convention:
    - release-*

      (-* : số hiệu phát hành)



ii. Release branches

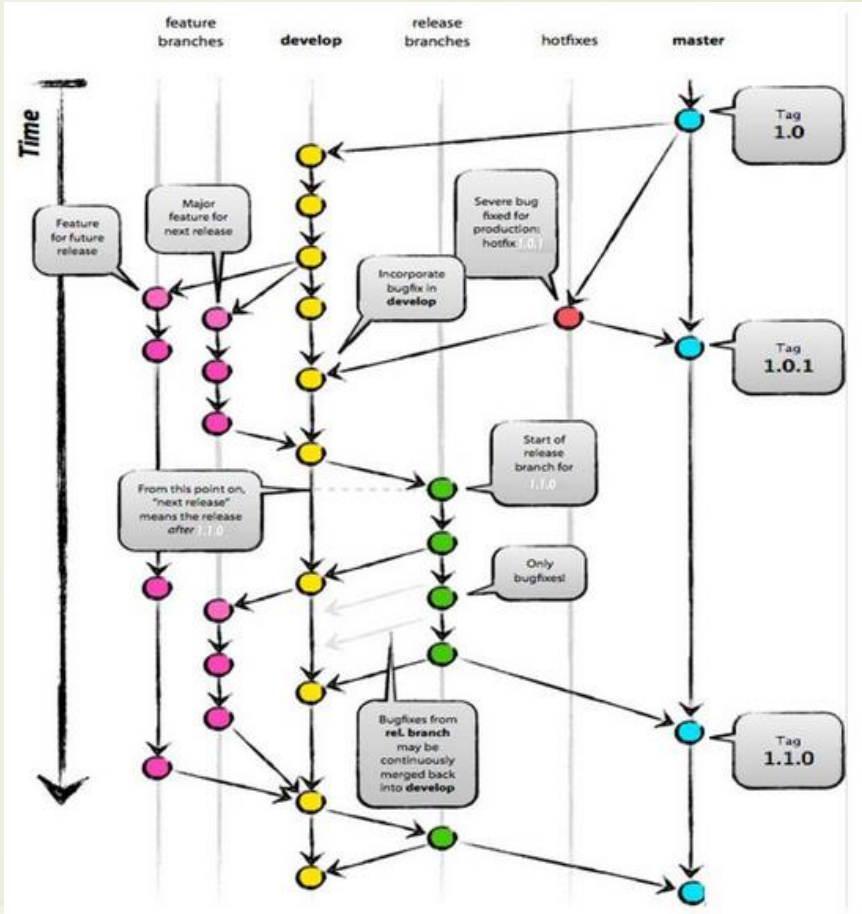
- Are used to:
  - Chuẩn bị new production release:
    - Thực hiện các fix minor bug (patch), phê chuẩn mã
    - Chuẩn bị dữ liệu cho release *(version number, build dates, etc.)*.
    - => release branch giúp giảm nhẹ công việc của develop branch, tạo đk cho developer tập trung vào big release tiếp theo



iii. Hot-fix branches

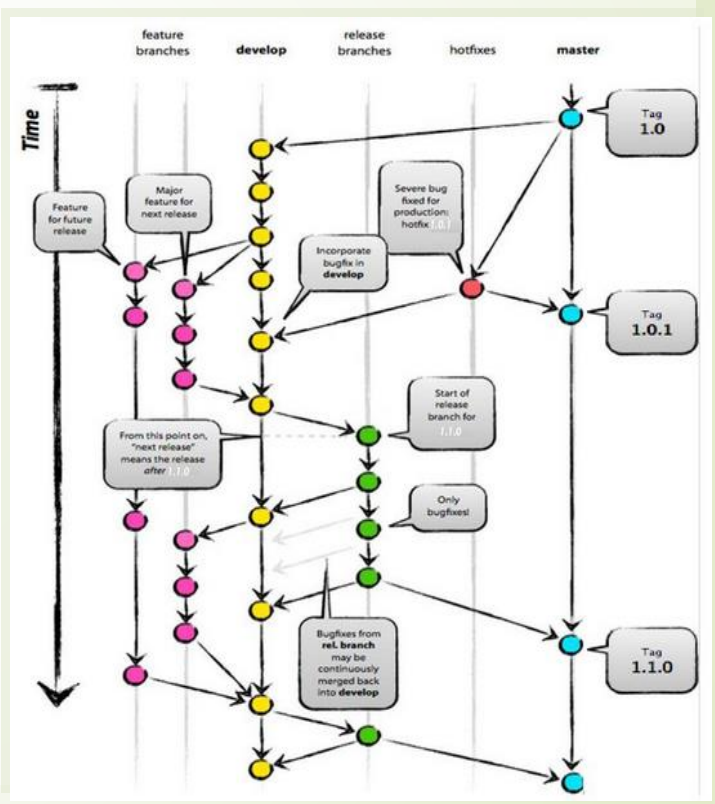
- Rules:
  - Original branch:
    - Master
  - Must merge back into:
    - develop and master
  - Branch naming convention:
    - hotfix-*

(-* : số hiệu phát hành)



iii. Hot-fix branches

- Are used to:
  - prepare for a new production release
    - Similar to release branches, but unplanned
      - Applied for immediately changes



1. Source Code Management

- Kỹ thuật phân nhánh:
  - Developers làm việc tại nhánh tương ứng của họ, sau đó nối (merge) mã nguồn đã được update vào nhánh chính để được theo dõi (tracked), kiểm toán (audited), xét duyệt và rolled back (nếu cần).

1. Source Code Management

- **Benefits:**
  - **Work together in teams.**
    - Allows multiple developers to access and change different areas of the code, without interfering with each other's work. This is a big benefit for large teams of developers.
  - **Version history.**
    - The system will keep a history of all saved changes to the code. => allows to see what has changed in a file, files, compare changes and define any problems that may arise
  - **Generate release notes.**
    - Code can be linked to a release, release notes can be generated

    => save time in generating it manually and searching for the changes.
  - **Backup of code.**
    - Easy backup of the code, depending on your server setup.

1. Source Code Management

- SCM Tools:
  - **Mất phí:**
    1. IBM ClearCase
    2. Perforce
    3. PVCS
    4. Team Foundation Server
    5. Visual Studio Team Services
    6. Visual SourceSafe
  - **Nguồn mở:**
    1. Subversion (SVN)
    2. Concurrent Version System (CVS)
    3. **Git & GitHub**
    4. SCCS
    5. Revision control systems
    6. Bitbucket

1. Source Code Management

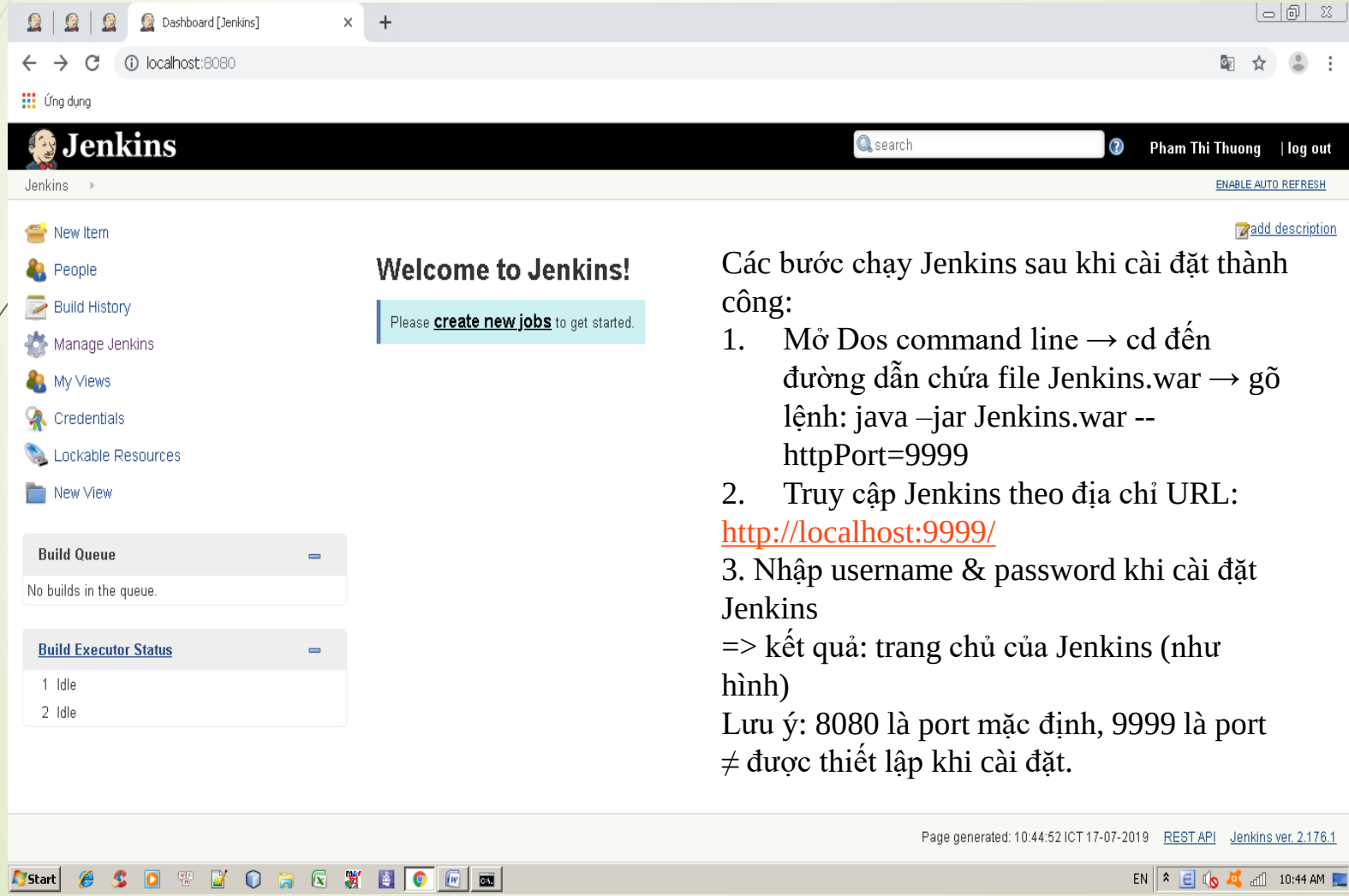
- SCM: Cài đặt và thiết lập
  a. Cài Jenkins
  b. Tải, cài Git, thiết lập chứng thực cá nhân
  c. Đăng ký tài khoản GitHub
  d. Tích hợp Git vào DevOps thông qua Jenkins

a. Cài đặt Jenkins

- Link: https://jenkins.io/index.html
  - Xem hướng dẫn cài đặt on Window 10:
    - https://www.youtube.com/watch?v=z0WDNPX9ZFA
- Kết quả cài đặt:
  - Truy cập Jenkins tại địa chỉ:
    - http://localhost:9999/
      - Kết quả: Bảng điều khiển (dashboard) Jenkins:

⇒ Xem màn hình (dưới):

a. Cài đặt Jenkins



Các bước chạy Jenkins sau khi cài đặt thành công:

1. Mở Dos command line → cd đến đường dẫn chứa file Jenkins.war → gõ lệnh: java –jar Jenkins.war --httpPort=9999
2. Truy cập Jenkins theo địa chỉ URL: http://localhost:9999/

3. Nhập username & password khi cài đặt Jenkins

=> kết quả: trang chủ của Jenkins (như hình)

Lưu ý: 8080 là port mặc định, 9999 là port ≠ được thiết lập khi cài đặt.

b. Tải, cài Git, thiết lập chứng thực cá nhân

- Link: **Git** for win: http://git-scm.com/download/win.
- Lưu ý:
  - Để làm việc với Git, trước hết phải thiết lập chứng thực cá nhân sau cài đặt:
  - Ví dụ:
    - **$** git config --global user.name "Pham Thuong"
    - **$** git config --global user.email ptthuong@ictu.edu.vn

    ⇒ Kiểm tra chứng thực:

    $ cat ~/.gitconfig

    [user]

        name = Pham Thuong

        email = ptthuong@ictu.edu.vn

c. Đăng ký tài khoản GitHub

- Đăng ký tài khoản, đăng nhập vào Github
  - Link:
    - https://github.com/join?source=prompt-code
      - Xem hình (dưới)
    - Sử dụng tài khoản đã đăng ký để Làm việc với GitHub

Join GitHub

The best way to design, build, and ship software.

Step 1: Set up your account

Step 2: Choose your subscription

Step 3: Tailor your experience

Create your personal account

Username *

This will be your username. You can add the name of your organization later.

Email address *

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password *

You'll love GitHub

Unlimited public repositories

Unlimited private repositories

- ✓ Limitless collaboration
- ✓ Frictionless development
- ✓ Open source community

https://github.com/PhamThuongBlog

Search or jump to... / Pull requests Issues Marketplace Explore

Overview Repositories 10 Projects Packages

PhamThuongBlog

Edit profile

ProTip! Updating your profile with your name, location, and a profile picture helps other GitHub users get to know you. Edit profile X

Popular repositories Customize your pins

Demo_Maven_Selenium

hoc_git_basic

Đăng nhập thành công

PhamThuongBlog.github.io

Forked from ndrewtl/airspace-jekyll

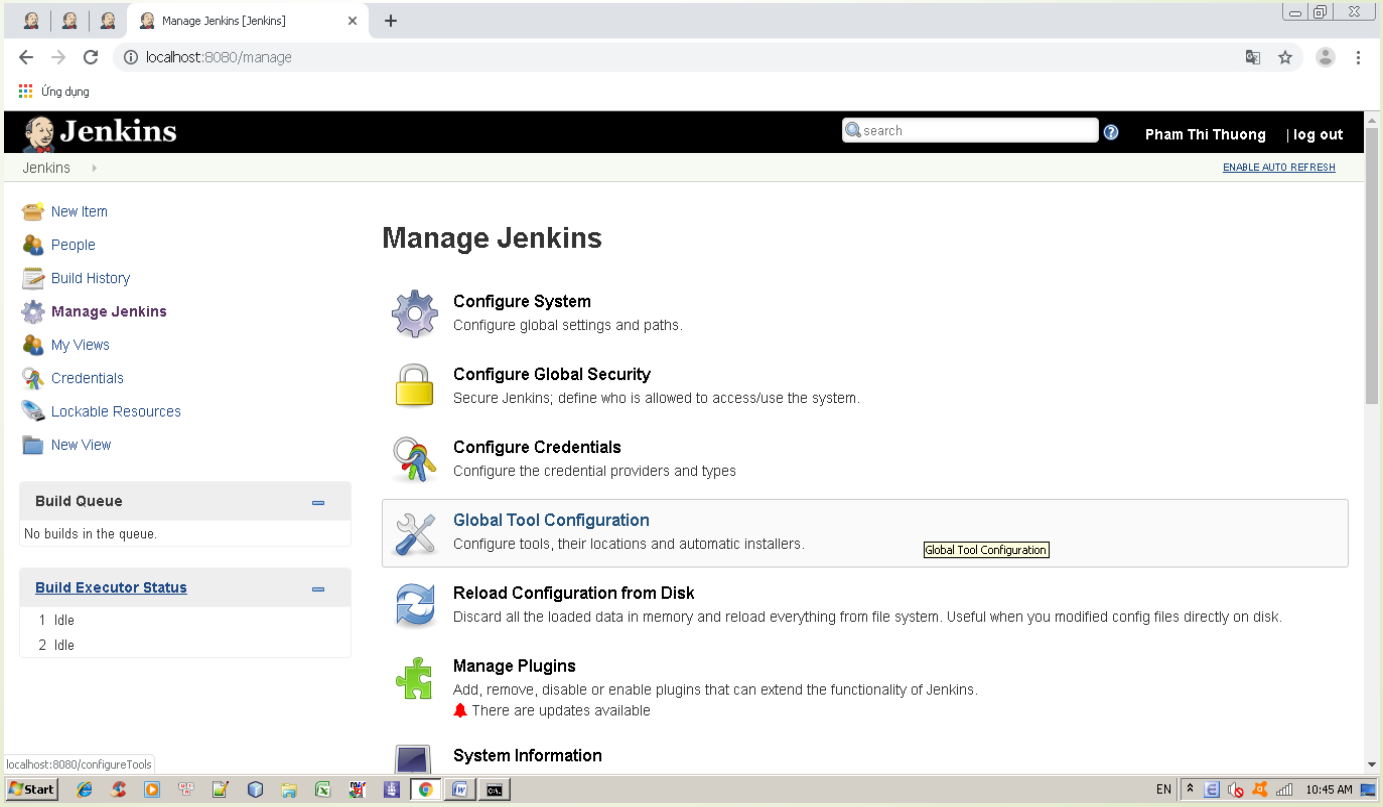
A port of the Airspace theme by ThemeFisher to Jekyll.

CSS

Demo_NetbeanProject

d. Tích hợp Git vào DevOps thông qua Jenkins

1. Chọn: **Manage Jenkins**



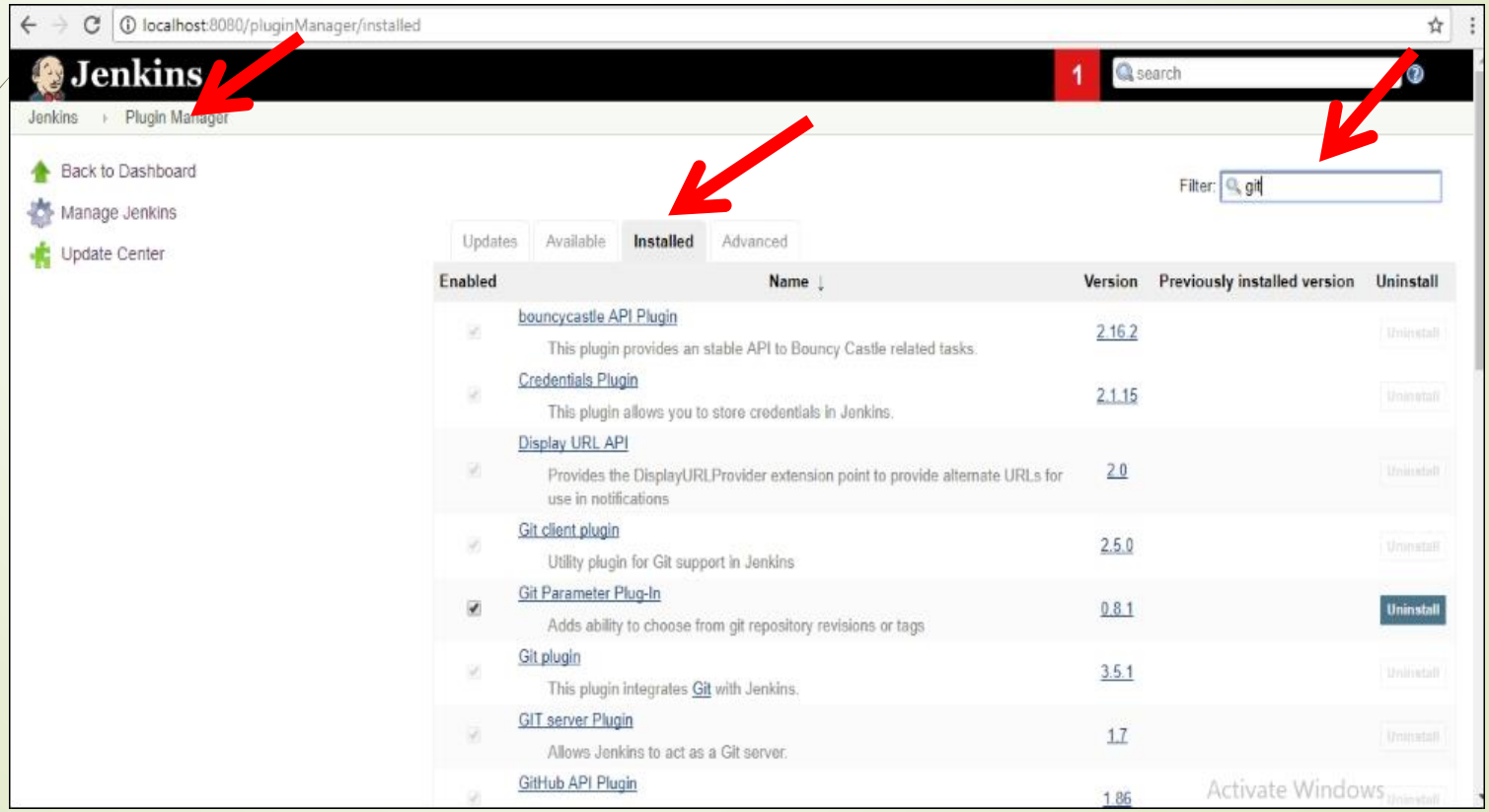
d. Tích hợp Git vào DevOps thông qua Jenkins

2. Chọn: **Manage Plugins**

- Tìm Git plugin sử dụng bộ lọc (filter) và cài đặt.
  - Kết quả: Xem hình (dưới)

d. Tích hợp Git vào DevOps thông qua Jenkins

- Kết quả:



1. Source Code Management

- DevOps framework kết hợp với SCM:

  => Lợi ích:

  - Cộng tác thành viên trong nhóm ↑ với các nhóm ≠
  - Phê chuẩn các thay đổi trước khi phát hành
  - Theo dõi đóng góp của ∀ cá nhân/tổ chức cộng tác ↑
  - Kiểm toán các thay đổi mã
  - Dễ dàng rollback, sao lưu và phục hồi dữ liệu.

III. (Core) DevOps Processes

1. Source code management – Git&GitHub
2. **Build management – Maven**
3. Source code review – Gerrit
4. Repository management – Nexus
5. Test Automation – Selenium
6. Continuous deployment – Pipelines
7. Continuous integration - Jenkins

2. Build management – Maven

- Build management:
  - An important part of the software development and testing lifecycle.
    - Is the process of collecting all of the assets to be included in a software release, performing all the automated tasks to **compile, build and test the system and then deploy** onto the development and testing environments in preparation for staging.

2. Build management – Maven

- Why?
  1. Building 1 dự án p.mềm gồm nhiều nhiệm vụ:
     i. Downloading dependencies,
     ii. putting additional jars on a classpath,
     iii. compiling source code into binary code,
     iv. running tests,
     v. packaging compiled code into deployable artifacts such as JAR, WAR, and ZIP files, and
     vi. deploying these artifacts onto the development and testing environments in preparation for staging

2. Build management – Maven

- Why?
  - Developers có thể tạo lỗi khi build phần mềm thủ công, hoặc tách rời giữa biên dịch và đóng gói. Lost time and making errors while download and setup dependences for each task
  - Need the build automation:
    - To ensure optimized building

      => Build automation?

2. Build management – Maven

- Why?
  - Cần dùng công cụ trợ giúp build tự động:
    - Có thể: tùy chỉnh các hoạt động trong tiến trình build & thiết lập thời gian build tự động, ví dụ:
      - Tích hợp và Build hàng đêm or anytime
      - Tích hợp và Build liên tục mỗi khi merge
      - Trigger tích hợp và Build thủ công.

2. Build management – Maven

- Build Tools:
  - GNU make for C/C++
  - Ant, **Maven**, Gradle, or GNU make for Java
  - MS Build or GNU Make for .NET (e.g., C#)

2. Build management – Maven

- Apache Maven:
  - Là công cụ build tự động, giúp tự động hóa toàn bộ các nhiệm vụ trên
    - Giảm thiểu rủi ro do build, test & đóng gói thủ công
    - It makes the day-to-day work of Java developers easier by making the build process easy, providing a uniform build system
  - Giải pháp cho việc quản lý các thành phẩm dự án (artifacts)

2. Build management – Maven

- Maven – POM file
  - Toàn bộ cấu trúc dự án Maven, nội dung được khai báo trong file pom.xml **(Project Object Model)**
    - POM.xml: là đơn vị nền tảng của toàn bộ hệ thống Maven.
      - ~ the XML file that contains information related to the project and configuration information such as dependencies, source directory, plugin, and goals etc. used by Maven to build the project.

2. Build management – Maven

- Maven – Dependences
  - Dependencies: ~ Java Archive (JAR) files required by the project
    - To specify a dependency in the POM file, we have to supply the following information:
      - groupID: should be the domain name of the JAR provider
      - artifactId: name of the JAR file without version
      - version: is the version of the Jar file
      - scope (option): determine when the jar file is used in the project, default is compile (~dependency is used while compilation & runtime)

2. Build management – Maven

- Maven – Dependences
  - For example:
    - Cấu hình file POM được sử dụng trong dự án xây dựng Website Evergreen BookStore sử dụng các công nghệ JSP, Servlet and Hibernate
      - See link

2. Build management – Maven

- Maven - Build Life Cycle
  - Vòng đời build là một chuỗi các giai đoạn được định nghĩa trước nhằm xác định trình tự các goals sẽ được thực thi.
    - Ví dụ: **Maven Build Lifecycle:**
      - bao gồm 6 giai đoạn chính sau:

2. Build management – Maven

1. Compile

- ~ Command: ***mvn compile***.
  - If this phase is executed, the source code is compiled.

2. Test

- ~ Command: ***mvn test***

  => Run unit tests placed at src/test/java folder use the resources placed at src/test/resources if there are any resources specified in this folder. be done automatically by Maven.

2. Build management – Maven

3. Validate

- The command: ***mvn validate***

  => Validate the POM if it is correctly formed according to model version definition.

4. Package

- Command ***mvn package***.

  ⇒ Group the compiled code in the specified distributable format (jar, war, etc.).

  ⇒ The packaged code is placed in the target folder.

2. Build management – Maven

5. Install

- Running ***mvn install***

  => Installs the packaged project into the local repository. That way, it can be used by other projects.

6. Deploy

- Command ***mvn deploy***
  - Similar to *install*; only it puts the final package on the shared repository so that any developers that have access to and used package in the common repository

2. Build management – Maven

- Maven có 3 build lifecycles & order of phases:
  - clean: pre-clean, **clean**, post-clean
    - xóa output của 1 build bằng cách xóa thư mục build của dự án
  - default: **validate**, initialize, generate-sources, process-sources, generate-resources, process-resources, **compile**, process-classes, generate-test-resources, process-test-resources, test-compile, process-test-classes, **test**, prepare-package, **package**, pre-integration-test, **integration-test**, post-integration-test, **verify**, **install**, **deploy**
  - site: pre-site, **site**, post-site, site-deploy
    - Tạo các báo cáo (reports), deploy site,…

2. Build management – Maven

- **Lưu ý:**
  - Có thể sửa file pom.xml của dự án maven để tùy chọn hành vi của các giai đoạn/vòng đời build.
  - Khi một giai đoạn được gọi = lệnh Maven,
    - => chỉ các giai đoạn trước đó đến giai đoạn đó được thực thi.
    - Ví dụ: **mvn clean**

      => pre-clean, **clean** được thực thi, post-clean không được thực thi.

2. Build management – Maven

- Maven – Lợi ích
  - Add all the dependencies, plug-ins, … required for the project automatically by reading the POM file.
    - Save a lot of time of programmers
  - Easily build project to jar, war etc…
  - Makes easy to start project in different environments that doesn't needs to handle the dependencies injection, builds, processing, etc.
  - Adding a new dependency is very easy - just by declaring the dependency in POM file.

2. Build management – Maven

- Maven – benefits
  - So, the greatest benefit of Maven is making the build process easy with automatic dependency management and helping developing understand Java projects by looking the POM file.

2. Build management – Maven

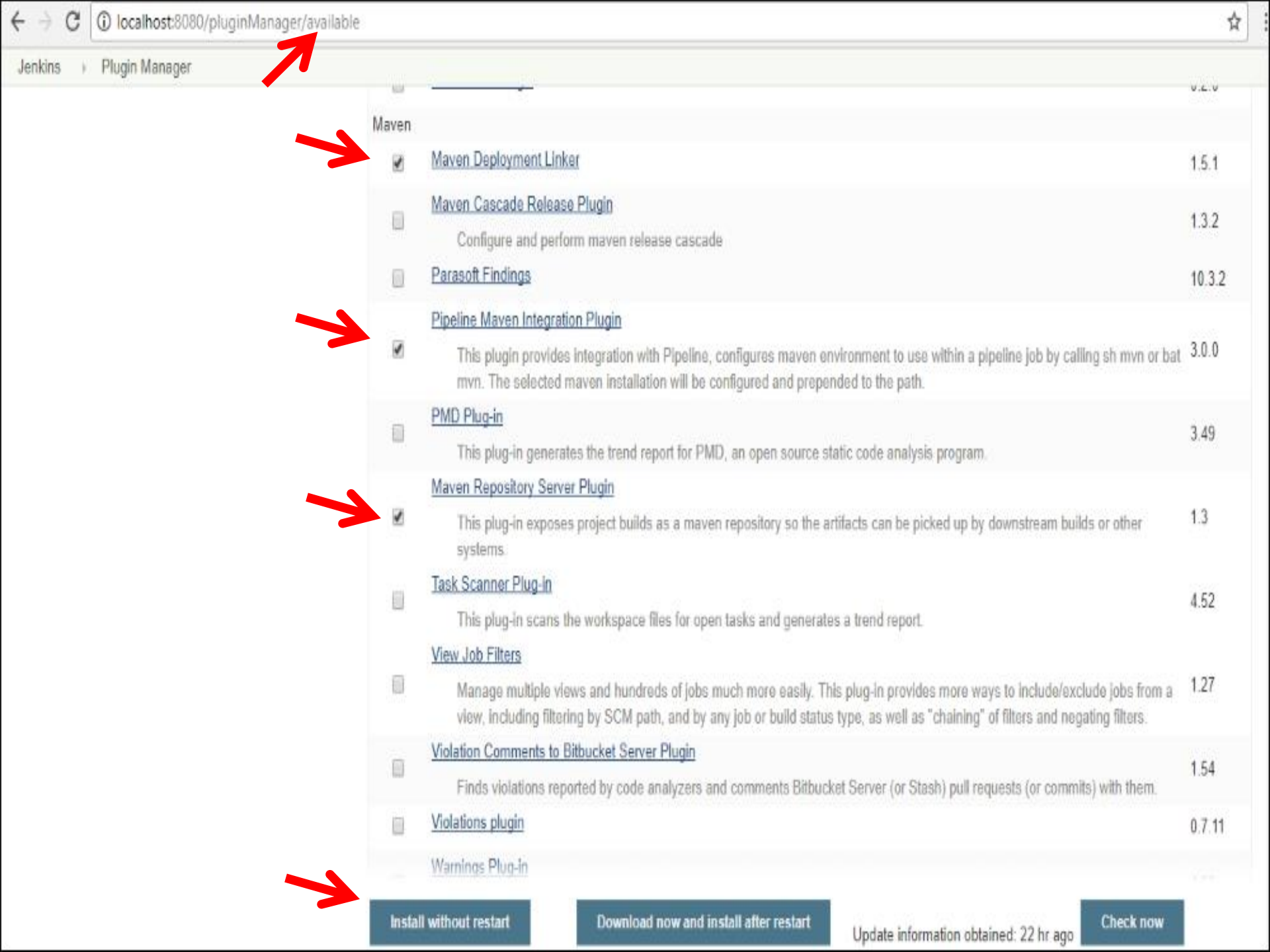
- Download: Apache Maven (phát hành mới nhất)
  - Link: https://maven.apache.org/download.cgi
- Installing Apache Maven:
  - Extracting the archive
  - Adding the **bin** folder into the PATH environment variable

2. Build management – Maven

- Tích hợp Maven (Build) với Jenkins

  Bước 1: Mở **Manage Jenkins**:

  - Lựa chọn Maven Plugins, tìm Maven và click: install them without the restart option
    - Màn hình (dưới)

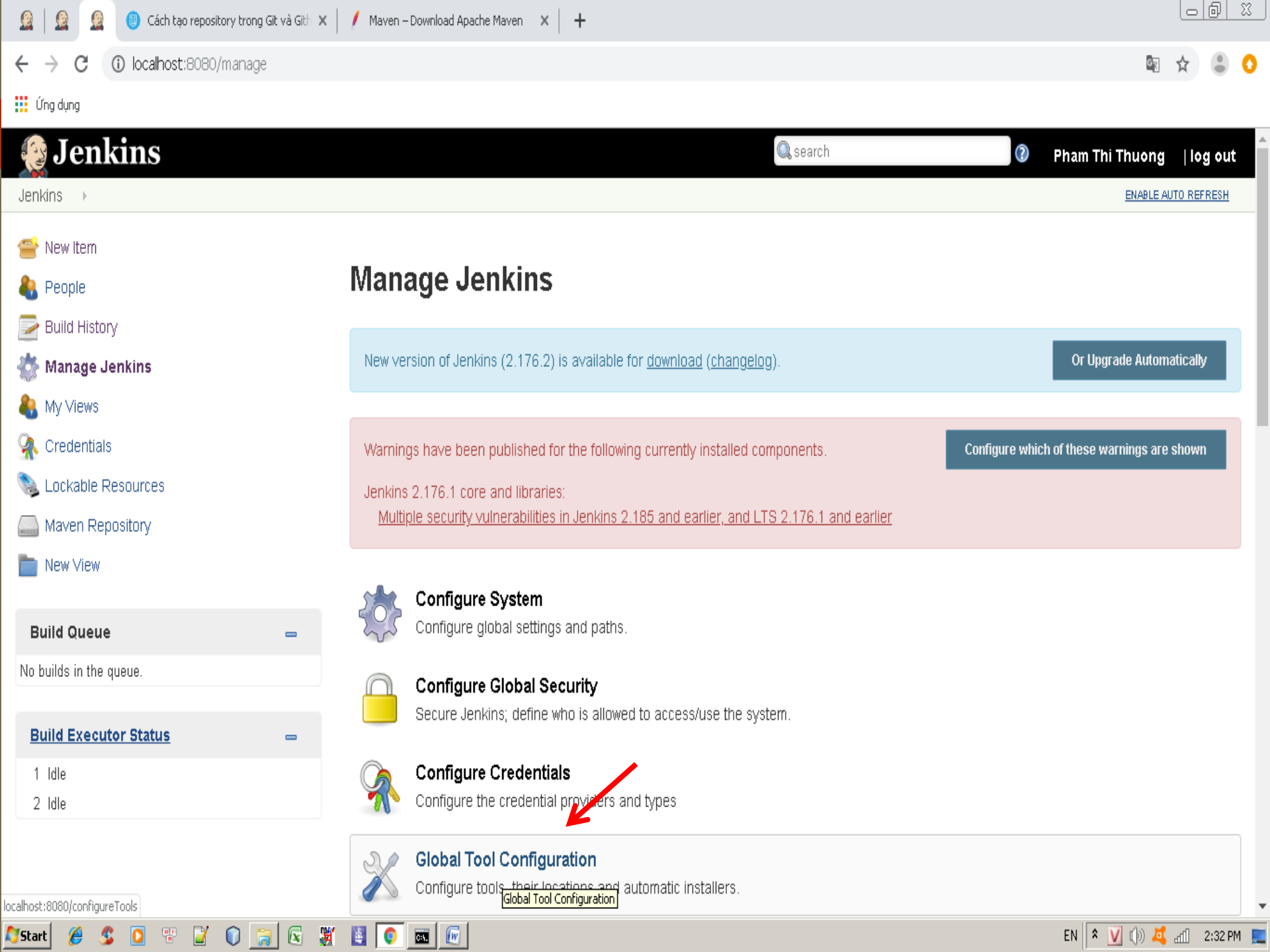

localhost:8080/pluginManager/available
Jenkins › Plugin Manager
Maven
Maven Deployment Linker
1.5.1
Maven Cascade Release Plugin
Configure and perform maven release cascade
1.3.2
Parasoft Findings
10.3.2
Pipeline Maven Integration Plugin
This plugin provides integration with Pipeline, configures maven environment to use within a pipeline job by calling sh mvn or bat mvn. The selected maven installation will be configured and prepended to the path.
3.0.0
PMD Plug-in
This plug-in generates the trend report for PMD, an open source static code analysis program.
3.49
Maven Repository Server Plugin
This plug-in exposes project builds as a maven repository so the artifacts can be picked up by downstream builds or other systems.
1.3
Task Scanner Plug-in
This plug-in scans the workspace files for open tasks and generates a trend report.
4.52
View Job Filters
Manage multiple views and hundreds of jobs much more easily. This plug-in provides more ways to include/exclude jobs from a view, including filtering by SCM path, and by any job or build status type, as well as "chaining" of filters and negating filters.
1.27
Violation Comments to Bitbucket Server Plugin
Finds violations reported by code analyzers and comments Bitbucket Server (or Stash) pull requests (or commits) with them.
1.54
Violations plugin
0.7.11
Warnings Plug-in
Install without restart
Download now and install after restart
Update information obtained: 22 hr ago
Check now

2. Build management – Maven

- Tích hợp Maven (Build) với Jenkins

  Bước 2. Chọn Global Tool **Configure**

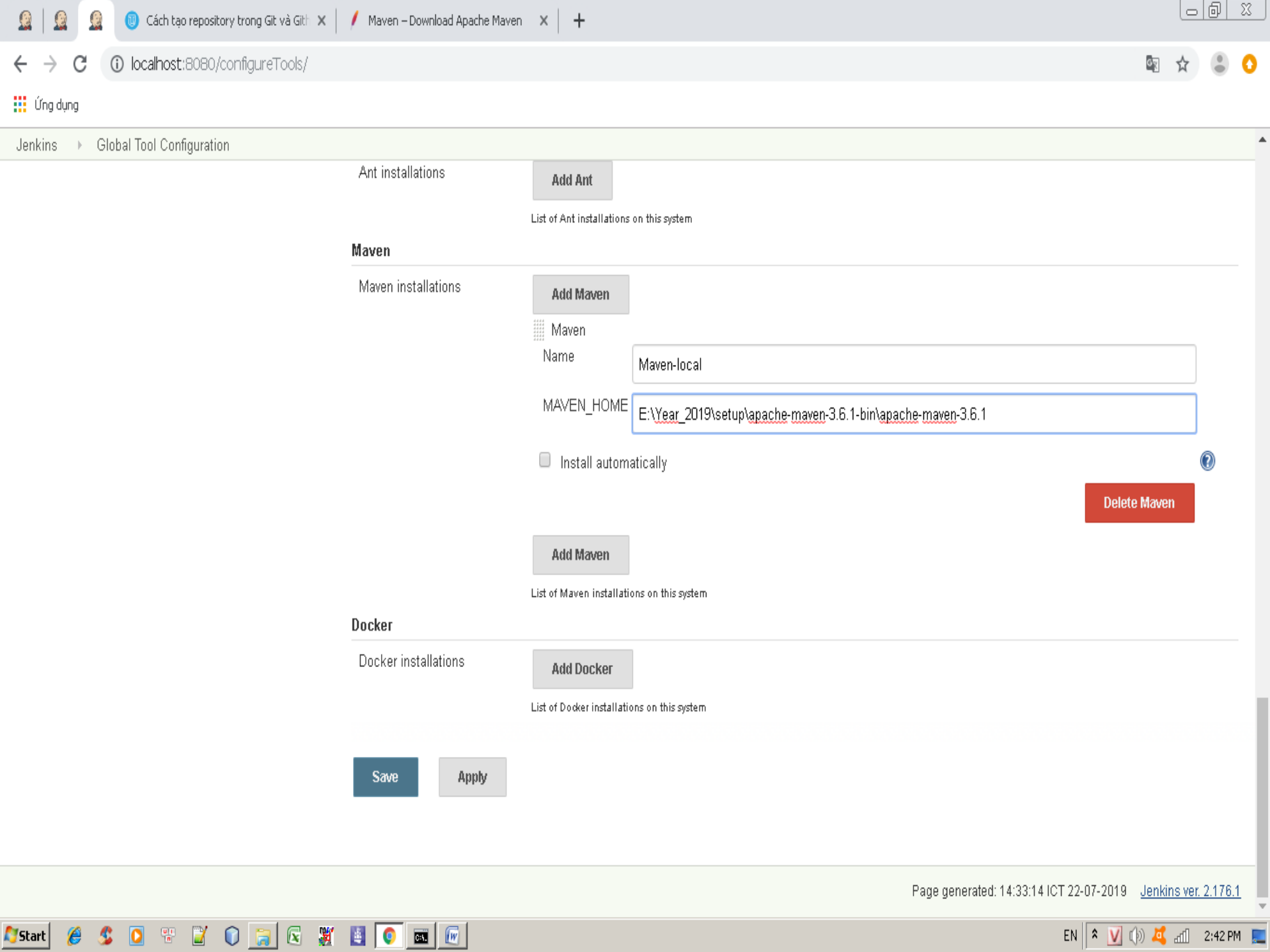
  - Xem hình (dưới)


Cách tạo repository trong Git và Gith
Maven – Download Apache Maven
localhost:8080/manage
Ứng dụng
Jenkins
search
Pham Thi Thuong
| log out
Jenkins
ENABLE AUTO REFRESH
New Item
People
Build History
Manage Jenkins
My Views
Credentials
Lockable Resources
Maven Repository
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Manage Jenkins
New version of Jenkins (2.176.2) is available for download (changelog).
Or Upgrade Automatically
Warnings have been published for the following currently installed components.
Configure which of these warnings are shown
Jenkins 2.176.1 core and libraries:
Multiple security vulnerabilities in Jenkins 2.185 and earlier, and LTS 2.176.1 and earlier
Configure System
Configure global settings and paths.
Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.
Configure Credentials
Configure the credential providers and types
Global Tool Configuration
Configure tools, their locations and automatic installers.
Global Tool Configuration
localhost:8080/configureTools
Start
EN
2:32 PM

2. Build management – Maven

- Tích hợp Maven (Build) với Jenkins
  - Bước 2. Chọn Global Tool **Configure**

    => Add Maven như hình (dưới):


Cách tạo repository trong Git và Gith
Maven – Download Apache Maven
localhost:8080/configureTools/
Ứng dụng
Jenkins
Global Tool Configuration
Ant installations
Add Ant
List of Ant installations on this system
Maven
Maven installations
Add Maven
Maven
Name
Maven-local
MAVEN_HOME
E:\Year_2019\setup\apache-maven-3.6.1-bin\apache-maven-3.6.1
Install automatically
Delete Maven
Add Maven
List of Maven installations on this system
Docker
Docker installations
Add Docker
List of Docker installations on this system
Save
Apply
Page generated: 14:33:14 ICT 22-07-2019
Jenkins ver. 2.176.1
Start
EN
2:42 PM

III. (Core) DevOps Processes

1. Source code management – Git&GitHub
2. Build management – Maven
3. **Source code review – Gerrit**
4. Repository management – Nexus
5. Test Automation – Selenium
6. Continuous deployment – Pipelines
7. Continuous integration - Jenkins

3. Source code review – Gerrit

- Why?
  - Chức năng quan trọng trong tiến trình phẩm mềm để cải tiến chất lượng của code updates, tìm lỗi, loại bỏ errors trước khi tích hợp vào luồng chính nhằm nâng cao chất lượng tổng thể của phần mềm
    - Nhằm cho ra phiên bản tốt nhất của mã nguồn.
      - **Use Gerrit to Be a Rockstar Programmer**
        - ~ talented programmers

3. Source code review – Gerrit

- Why? Review to:
  - Fix compilation errors; Avoid duplicate code
  - Use a better algorithm, to make it faster
  - Handle error conditions, to make it more robust
  - Add tests, to prevent a bug
  - Adapt tests, to reflect changed behavior
  - Polish/đánh bóng code, to make it easier to read
  - Improve the commit message to explain why a change was made
  - Check if code is follows the coding standards, named conventions, …with clear, logical structures?

3. Source code review – Gerrit

- Các lợi ích:
  - Mọi thay đổi code được shared đến cộng sự để nhận được các góp ý
    - Việc share được tiến hành theo 2 hướng:
      - Reviewers học được từ developers, và ngược lại developers học từ reviewers về best practices được sử dụng trong dự án.
      - Share cho càng nhiều người, càng tăng cơ hội tìm được nhiều bugs và nhiều gợi ý cải tiến.

3. Source code review – Gerrit

- Buộc team và company phải có chính sách và cơ chế cho review
  - Ví dụ:
    - Tất cả các tests cần "pass" trên mọi nền tảng hoặc ít nhất 2 người ký vào code trước khi chuyển đến production evr.
- Nhiều công ty phát triển p.mềm
  - Ví dụ: Google
    - Google sử dụng code review như 1 tiêu chuẩn, một giai đoạn không thể thiếu trong tiến trình ↑ p.mềm.

3. Source code review – Gerrit

- Tools mã nguồn mở:
  - Review board
  - Phabricator
  - **Gerrit**
  - GitLab

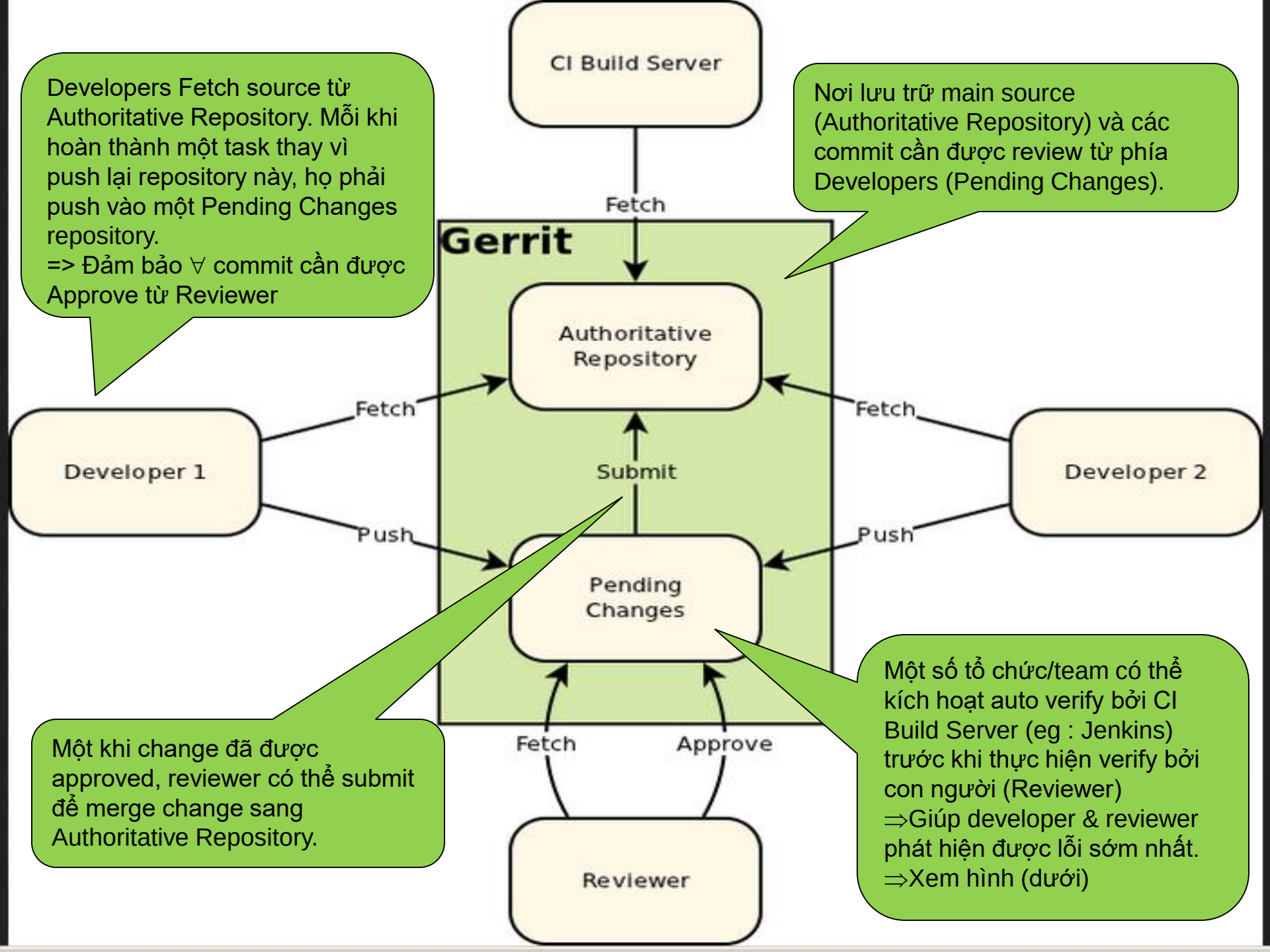
3. Source code review – Gerrit

- Gerrit:
  - Hỗ trợ quản lý & review source code dựa trên nền tảng web, sử dụng git làm version control
  - Được phát triển tại Google bởi Shawn Pearce trong quá trình phát triển dự án Android.
    - Ban đầu viết bằng python, từ version 2.x chuyển sang Java trên nền tảng J2EE servlet container.

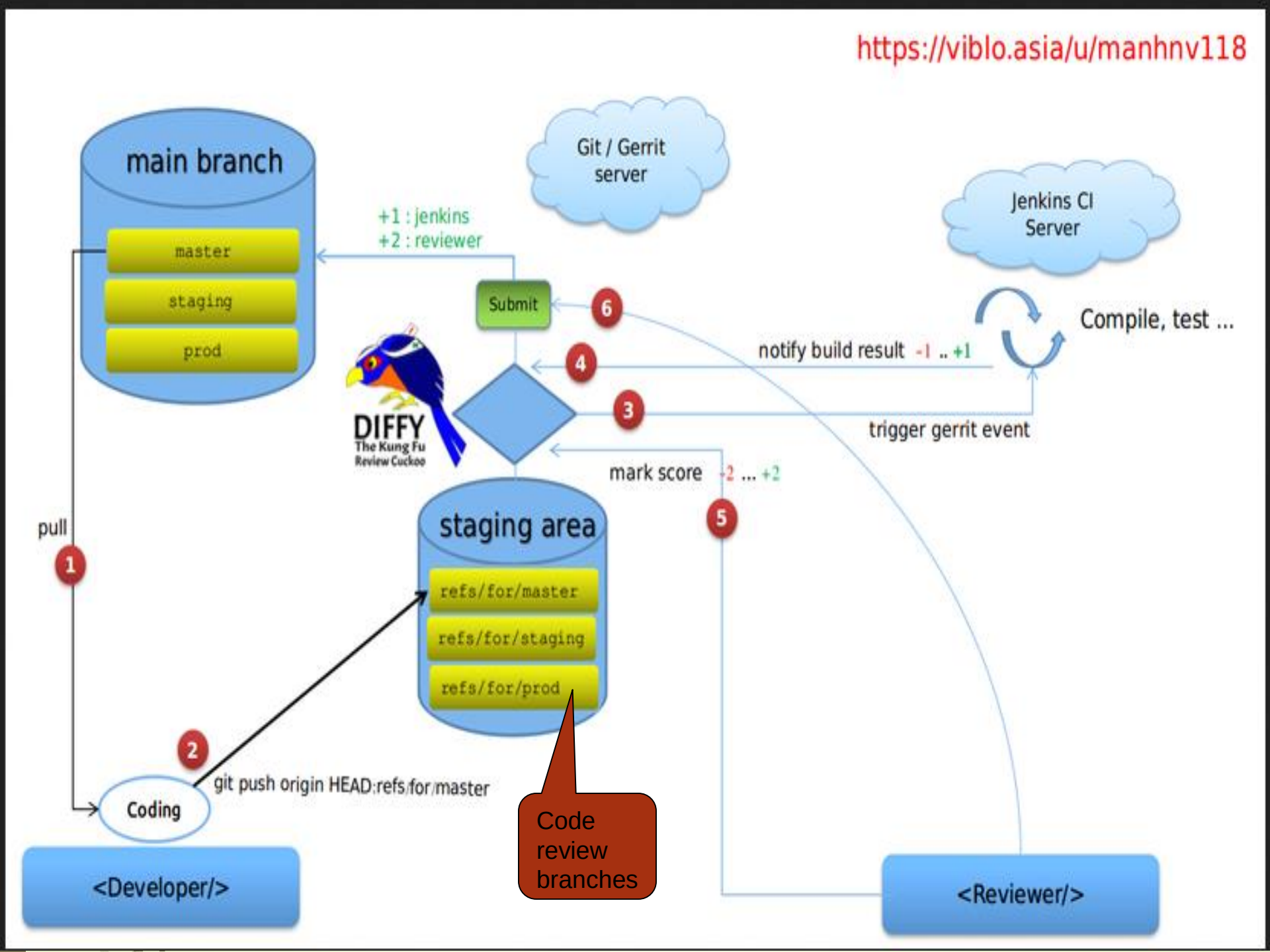
      => *Gerrit không đơn thuần chỉ là công cụ review mà nó có thể được sử dụng nó để xây dựng một server quản lý source code, giống như những gì mà github, gitlab ..vv đang làm.*

3. Source code review – Gerrit

- Gerrit:
  - Hiện nay, Android Open Source Project (AOSP) cũng sử dụng Gerrit làm công cụ quản lý version chính
    - Một số tổ chức khác: CyanogenMod, Eclipse Foundation, LibreOffice, OpenStack,Qt … cũng dùng Gerrit.
  - Gerrit tổ chức như thế nào?
    - Xem hình (dưới)


CI Build Server
Fetch
Gerrit
Authoritative Repository
Fetch
Fetch
Developer 1
Submit
Developer 2
Push
Push
Pending Changes
Fetch
Approve
Reviewer
Developers Fetch source từ Authoritative Repository. Mỗi khi hoàn thành một task thay vì push lại repository này, họ phải push vào một Pending Changes repository.
=> Đảm bảo ∀ commit cần được Approve từ Reviewer
Nơi lưu trữ main source (Authoritative Repository) và các commit cần được review từ phía Developers (Pending Changes).
Một khi change đã được approved, reviewer có thể submit để merge change sang Authoritative Repository.
Một số tổ chức/team có thể kích hoạt auto verify bởi CI Build Server (eg : Jenkins) trước khi thực hiện verify bởi con người (Reviewer)
⇒Giúp developer & reviewer phát hiện được lỗi sớm nhất.
⇒Xem hình (dưới)

https://viblo.asia/u/manhnv118

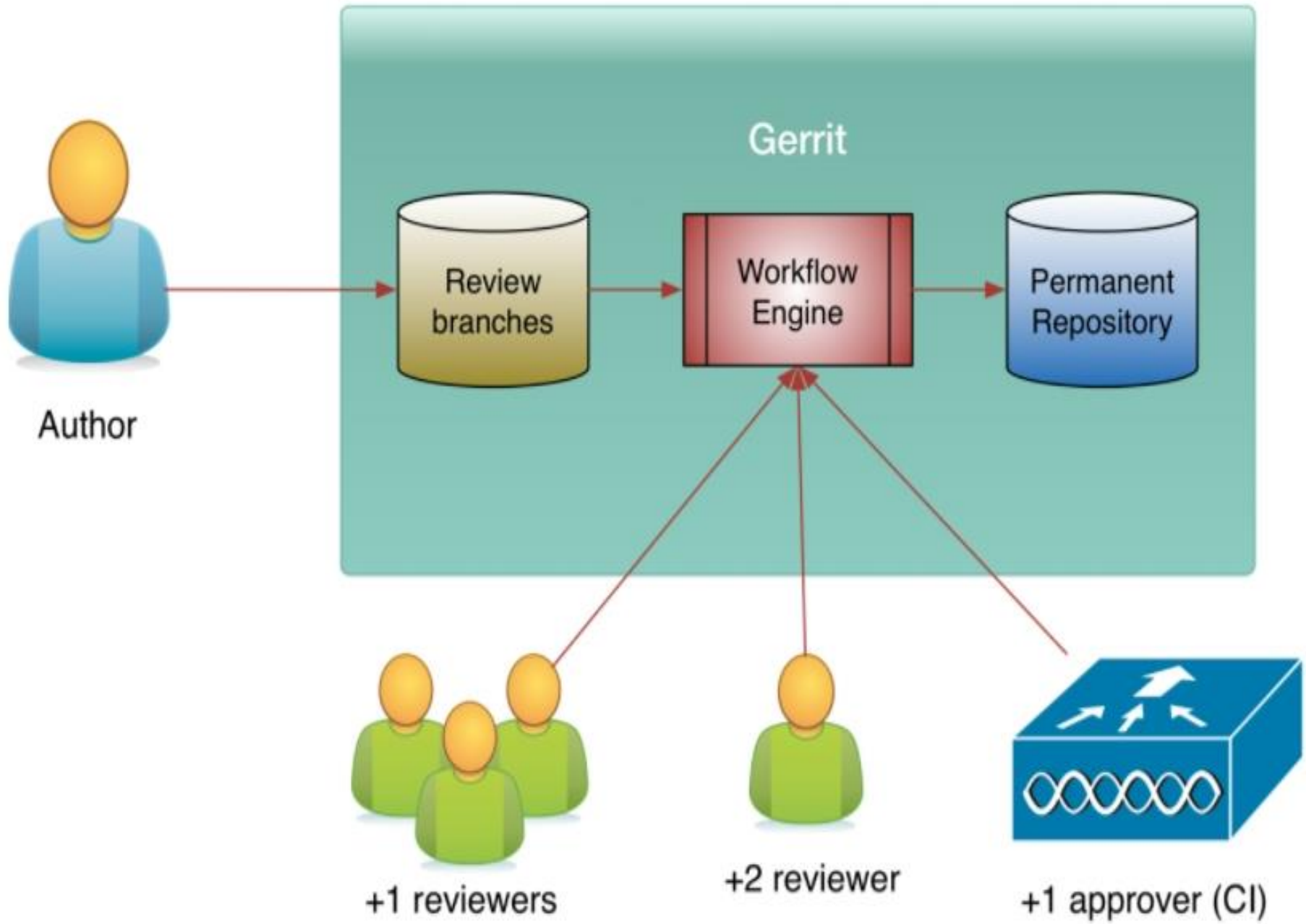


3. Source code review – Gerrit

- Chất lượng source của commit được đánh giá thể hiện theo mức điểm -2 to +2:
  1. Label verify (áp dụng cho auto verify - compile, passes basic unit tests):
     - -1 : Fails - compile error hoặc unit tests fail
     - 0 : No score - việc verification không được thực hiện.
     - +1: pass all tests

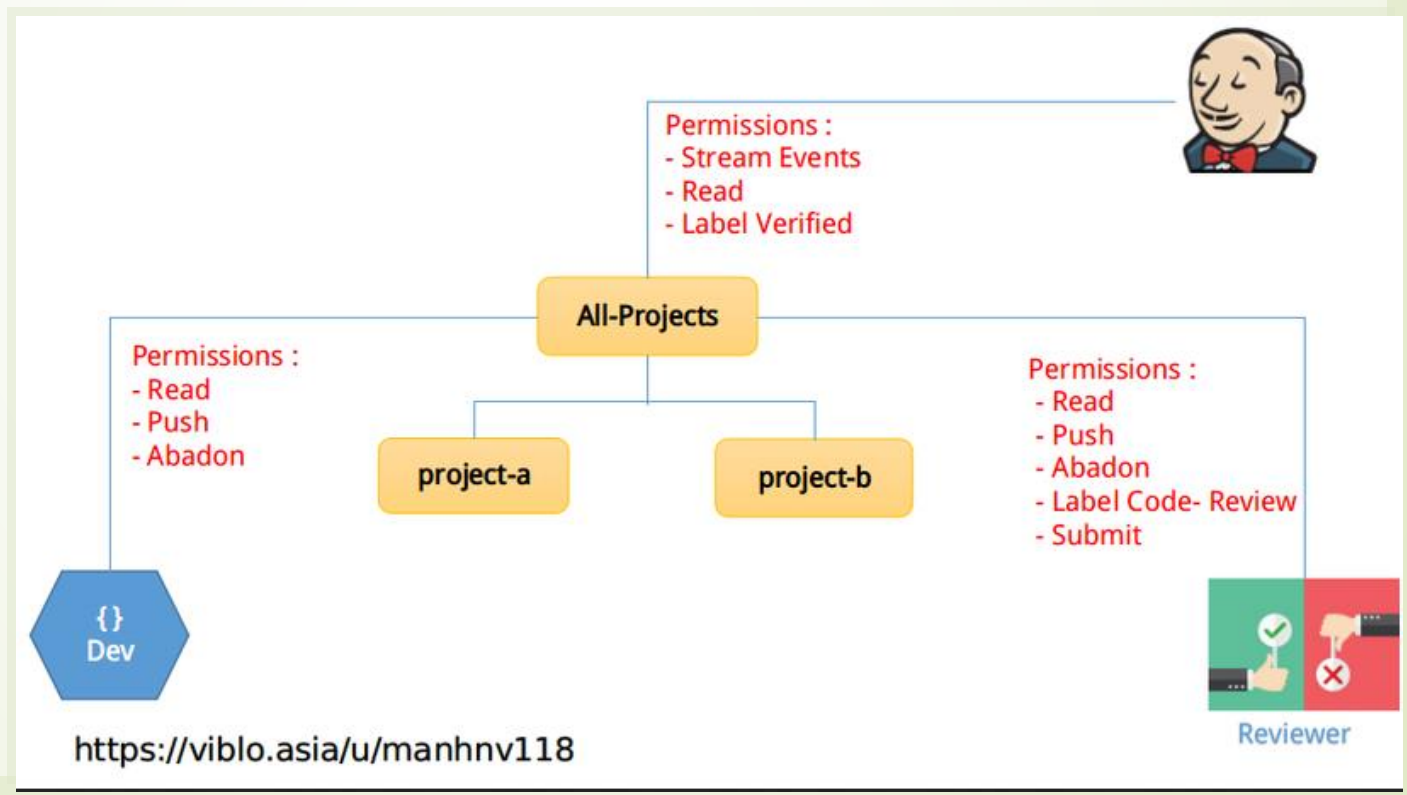
3. Source code review – Gerrit

- Chất lượng source của commit được đánh giá thể hiện theo mức điểm -2 to +2:
  - 2. Code Review (reviewers):
    - 1 reviewer có thể vote code bởi 1 trong 5 kết luận:
      - **+2**: Looks good, approved
      - **+1**: Looks good, but needs additional approval
      - **0**: No comments
      - **-1**: Suggest not submit this
      - **-2**: Block the submit
    - Ví dụ: xem hình (dưới)


Gerrit
Review branches
Workflow Engine
Permanent Repository
Author
+1 reviewers
+2 reviewer
+1 approver (CI)

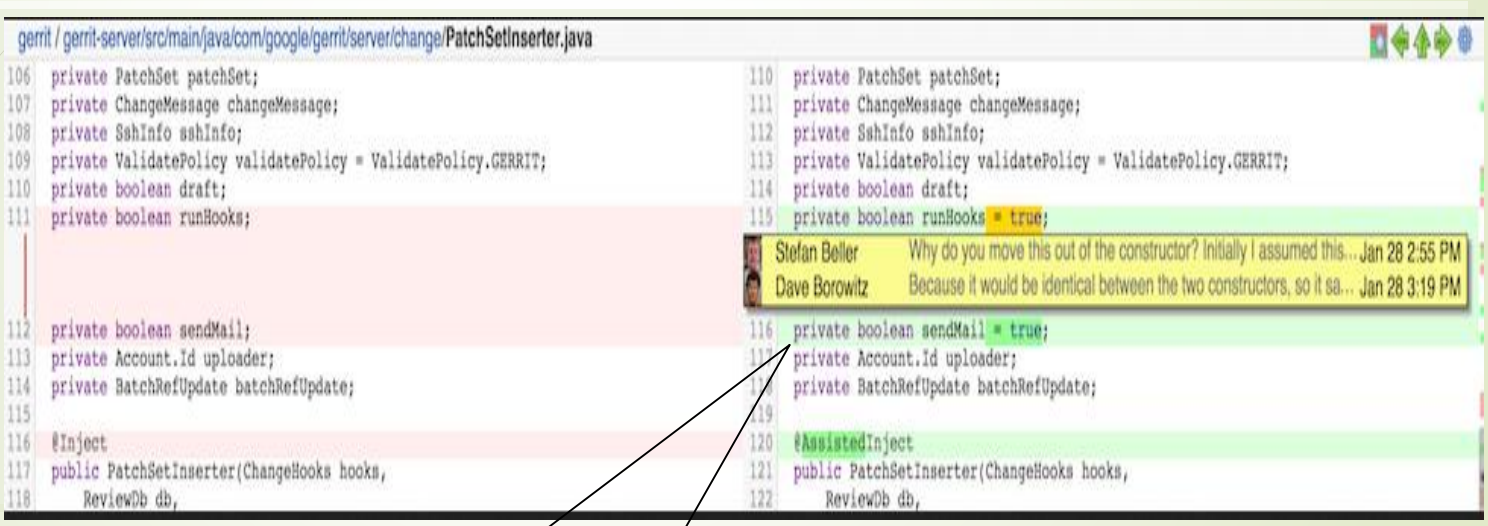
3. Source code review – Gerrit

- Cấp phép trên gerrit:



3. Source code review – Gerrit

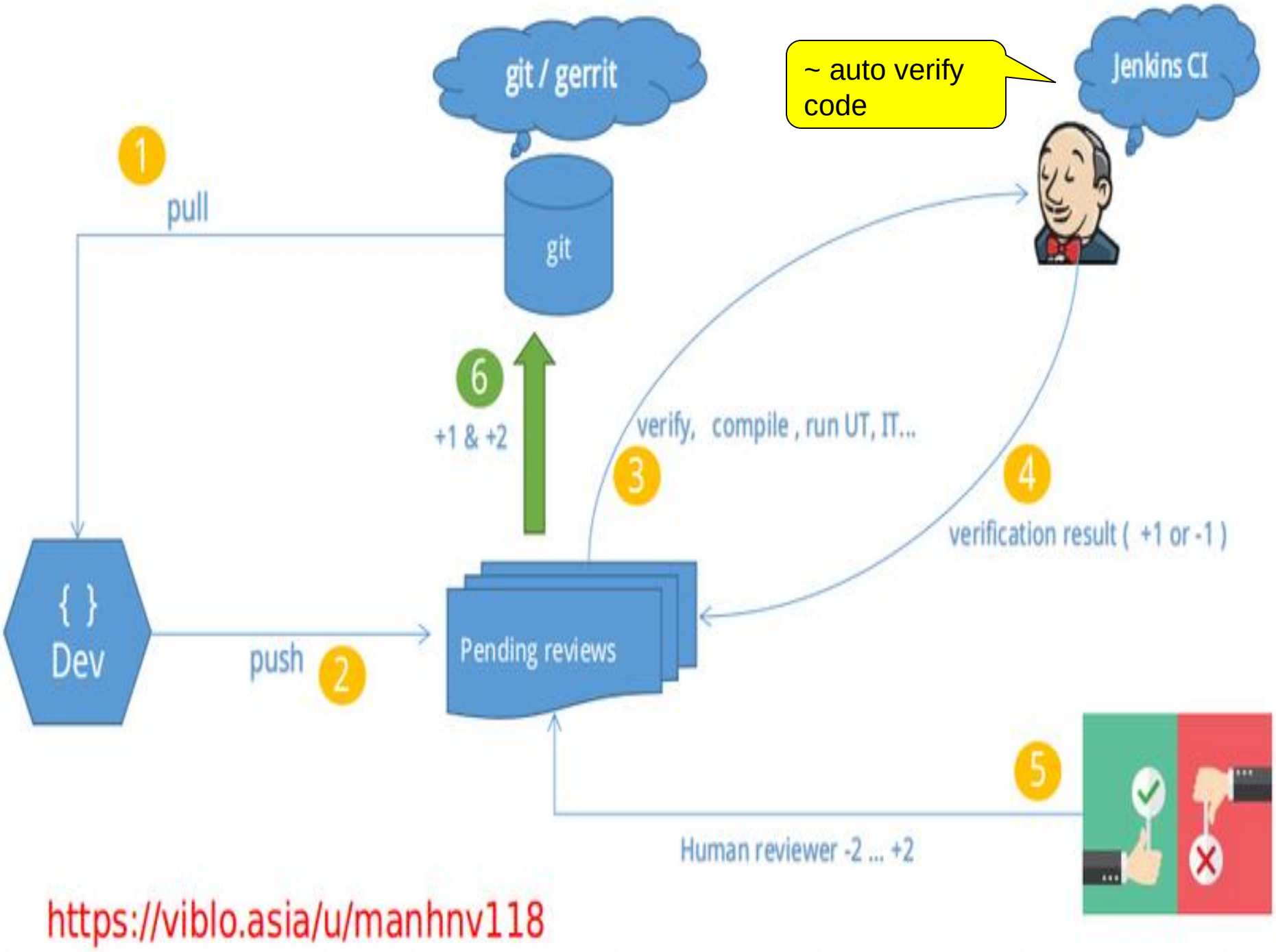
- Review trong Gerrit: Thảo luận mã
  - Ví dụ:



Cơ chế review rất trực quan: Comment và Reply comment được insert trên từng dòng code của files => Giúp phân tích và khả năng communicate tốt hơn.

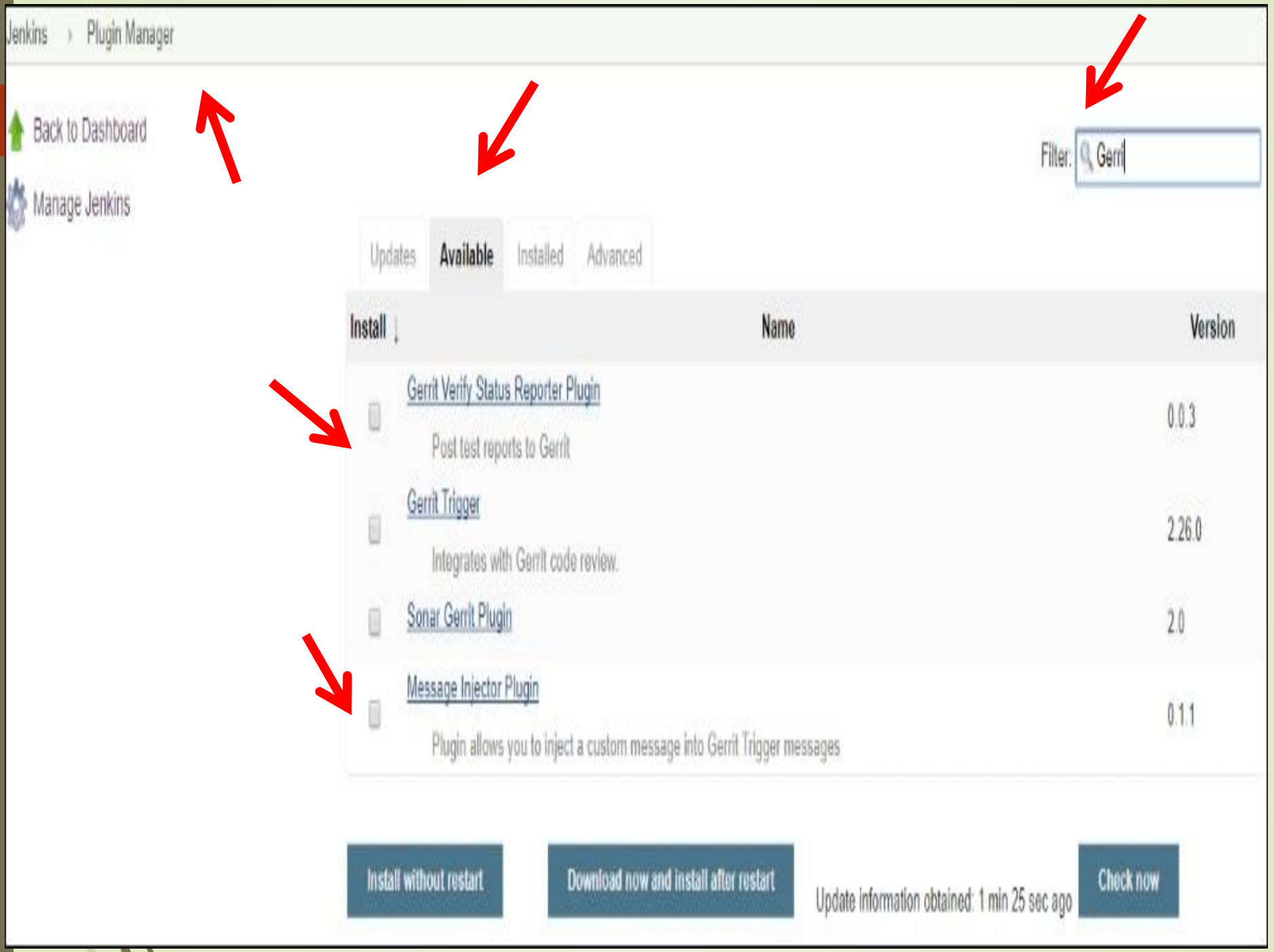
3. Source code review – Gerrit

- Cài đặt Gerrit và tích hợp với Jenkins
  - Cộng tác hoạt động sau tích hợp
    - Xem hình (dưới)


git / gerrit
Jenkins CI
~ auto verify code
1
pull
git
6
+1 & +2
verify, compile , run UT, IT...
3
4
verification result (+1 or -1)
{ }
Dev
push
2
Pending reviews
5
Human reviewer -2 ... +2
https://viblo.asia/u/manhnv118

3. Source code review – Gerrit

- Cài đặt Gerrit và tích hợp với Jenkins
  - Download và cài đặt Gerrit:
    - Link: from https://www.gerritcodereview.com/
    - Bản mới nhất**: gerrit-3.0.1.war**
  - Tích hợp Gerrit với Jenkins
    - Xem hình (dưới)


Jenkins › Plugin Manager
Back to Dashboard
Manage Jenkins
Filter: Gerri
Updates
Available
Installed
Advanced
Install
Name
Version
Gerrit Verify Status Reporter Plugin
Post test reports to Gerrit
0.0.3
Gerrit Trigger
Integrates with Gerrit code review.
2.26.0
Sonar Gerrit Plugin
2.0
Message Injector Plugin
Plugin allows you to inject a custom message into Gerrit Trigger messages
0.1.1
Install without restart
Download now and install after restart
Update information obtained: 1 min 25 sec ago
Check now

III. (Core) DevOps Processes

1. Source code management – Git&GitHub
2. Build management – Maven
3. Source code review – Gerrit
4. **Repository management – Nexus**
5. Test Automation – Selenium
6. Continuous deployment – Pipelines
7. Continuous integration - Jenkins

4. Artifacts Repository Management – Nexus

- Why?
  - Một sản phẩm p.mềm có thể gồm 10, 100, …1000 các individual binary and build artifacts
    - Artifacts:
      - Packages, build metadata scripts, …
      - Product releases (*.com, *.exe, …)
    - Ví dụ: Sun's artifacts (Sun's Jar - xem bảng)
      - Xem bảng (dưới)

Product artifact	Group ID	Artifact ID
Java Activation Framework	javax.activation	activation
J2EE	javax.j2ee	j2ee
Java Data Object (JDO)	javax.jdo	jdo
Java Message Service (JMS)	javax.jms	jms
JavaMail	javax.mail	mail
Java Persistence API (JPA) / EJB 3	javax.persistence	persistence-api
J2EE Connector Architecture	javax.resource	connector
J2EE Connector Architecture API	javax.resource	connector-api
Java Authentication and Authorization Service (JAAS)	javax.security	jaas
Java Authorization Contract for Containers	javax.security	jacc
Servlet API	javax.servlet	servlet-api
Servlet JavaServer Pages (JSP)	javax.servlet	jsp-api
Servlet JavaServer Pages Standard Tag Library (JSTL)	javax.servlet	jstl
JDBC 2.0 Optional Package	javax.sql	jdbc-stdext
Java Transaction API (JTA)	javax.transaction	jta
Java XML RPC	javax.xml	jaxrpc
Portlet	javax.portlet	portlet-api
Java Naming and Directory Interface (JNDI)	javax.naming	jndi

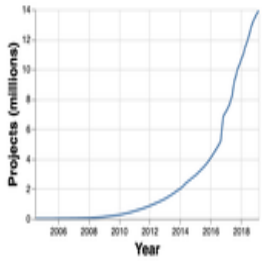
4. Artifacts Repository Management – Nexus

- Why?
  - Artifacts cần được quản lý một cách hiệu quả (quản lý tập trung mọi binary artifacts) để:
    - Duy trì tính nguyên vẹn của sản phẩm.
    - Chia sẻ, hỗ trợ trong phát triển p.mềm cộng tác
      - ∀ artifacts là sẵn dùng trên kho trung tâm
      - Ví dụ: kho artifact của Maven (MVNRepository)

4. Artifacts Repository Management – Nexus

- MVNRepository (∀ artifacts: *.jar)
  1. **Kho center:**
     - Jar files được published hàng năm
       - Link: https://mvnrepository.com/repos/central
       - Xem hình (dưới)

Indexed Artifacts (15.0M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities

Central Repository

https://repo1.maven.org/maven2/

URL	https://repo1.maven.org/maven2/
Jars	4,158,569 indexed jars

Published Jars by Year

2019	796,515
2018	924,592
2017	710,220
2016	540,051
2015	362,997
2014	229,888
2013	170,787
2012	126,711
2011	106,285
2010	86,191

Indexed Repositories (1192)

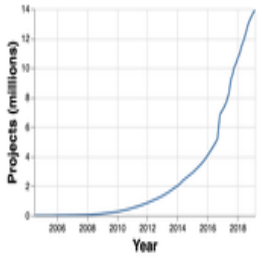
- Central
- Sonatype
- Spring Plugins
- Spring Lib M
- Hortonworks
- Atlassian
- JCenter
- JBoss Releases
- JBossEA
- Spring Lib Release

4. Artifacts Repository Management – Nexus

- MVNRepository

  2. Kho Spring Plugins

  1. Jar files được published hàng năm
     - Link: https://repo.spring.io/plugins-release/
     - Xem hình (dưới)



Spring Plugins Repository

https://repo.spring.io/plugins-release/

URL	https://repo.spring.io/plugins-release/
Jars	1,436,412 indexed jars

Published Jars by Year

2019	3,159
2018	153,851
2017	496,368
2016	714,630
2015	23,643
2014	15,306
2013	12,269
2012	4,772
2011	3,933
2010	2,867
2009	2,983

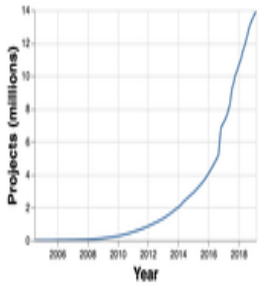
4. Artifacts Repository Management – Nexus

- MVNRepository

  => Phân loại artifacts

  - Xem hình (dưới)

Indexed Artifacts (15.0M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities

Aspect Oriented

Sort: **popular** | newest

1. **Spring AOP** — 2,359 usages
org.springframework » spring-aop — Apache
Spring AOP
Last Release on Aug 2, 2019

2. **AspectJ Runtime** — 2,130 usages
org.aspectj » aspectjrt — EPL
The runtime needed to execute a program using AspectJ
Last Release on May 11, 2019

3. **AspectJ Weaver** — 1,966 usages
org.aspectj » aspectjweaver — EPL
The AspectJ weaver introduces advices to java classes
Last Release on May 11, 2019

4. **Spring Boot AOP Starter** — 886 usages

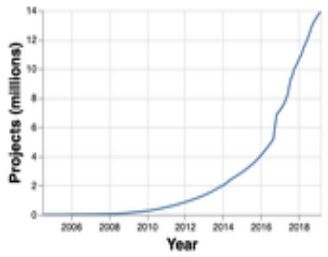
Indexed Repositories (1192)

- Central
- Sonatype
- Spring Plugins
- Spring Lib M
- Hortonworks
- Atlassian
- JCenter
- JBoss Releases
- JBossEA
- Spring Lib Release

4. Artifacts Repository Management – Nexus

- Why?
  - Artifacts trong repository có thể được điều khiển qua phiên bản
    - Ví dụ: Spring AOP artifacts
      - See link:
        - https://mvnrepository.com/artifact/org.springframework/spring-aop
      - See hình (dưới)

Indexed Artifacts (17.8M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases

Home » org.aspectj » aspectjrt

aspectj

AspectJ Runtime

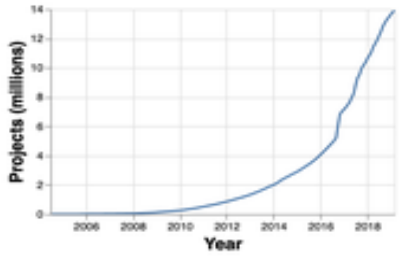
The runtime needed to execute a program using AspectJ

License	EPL 1.0
Categories	Aspect Oriented
Tags	aspect aop
Used By	2,365 artifacts

Central (42) | Spring Releases (10) | Spring Milestones (30) | ICM (3) | Alfresco (2)

	Version	Repository	Usages	Date
1.9.x	1.9.6	Central	400	Jul, 2020
	1.9.5	Central	492	Nov, 2019
	1.9.4	Central	480	May, 2019
	1.9.3	Central	373	Apr, 2019
	1.9.3.RC1	Central	1	Mar, 2019
	1.9.2	Central	515	Oct, 2018
	1.9.1	Central	411	Apr, 2018
	1.9.0	Central	21	Apr, 2018

Indexed Artifacts (17.8M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases

Home » org.aspectj » aspectjrt » 1.9.6

AspectJ Runtime » 1.9.6

The runtime needed to execute a program using AspectJ

License	EPL 1.0
Categories	Aspect Oriented
HomePage	https://www.eclipse.org/aspectj/
Date	(Jul 22, 2020)
Files	jar (118 KB) View All
Repositories	Central
Used By	2,365 artifacts

Maven | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjrt</artifactId>
    <version>1.9.6</version>
</dependency>
```

☑ Include comment with link to declaration

4. Artifacts Repository Management – Nexus

- MVNRepository
  - Các quyền hạn sử dụng kho:
    1. Truy cập: **file://** (kho cục bộ/internal) và **http://** (center/remote Repository)
    2. Download,
    3. Publishing your artifacts to the Central Repository
       - Need to use an approved repository hosting
    4. Make change

    …

4. Artifacts Repository Management – Nexus

- **Why:**
  - Using Nexus?

THE 2014 LEADERBOARD
OF JAVA TOOLS & TECHNOLOGIES

REBELLABS

82.5%
JUnit*
TOP TESTING FRAMEWORK USED BY DEVELOPERS

70%
Jenkins°
MOST USED CI SERVER IN THE INDUSTRY

64%
Maven
MOST USED BUILD TOOL IN JAVA

64%
Nexus°
THE MAIN REPOSITORY USED BY DEVELOPERS

67.5%
Hibernate*/°
THE TOP ORM FRAMEWORK USED

65% Java 7
THE INDUSTRY LEADER FOR SE DEVELOPMENT

56%
MongoDB*
THE NOSQL TECHNOLOGY OF CHOICE

69% Git*
#1 VERSION CONTROL TECHNOLOGY OUT THERE

55%
FindBugs*/°
MOST-USED STATIC CODE ANALYSIS TOOL

50%
Tomcat*
THE MOST POPULAR APPLICATION SERVER

49%
Java EE 6*
FOUND IN THE MOST ENTERPRISES

48%
Eclipse
THE IDE USED MORE THAN ANY OTHER

40% **Spring MVC*/°**
MOST COMMONLY USED WEB FRAMEWORK

32% **MySQL***
THE MOST POPULAR SQL TECHNOLOGY

* *Multiple selections possible* ° *Normalized to exclude non-user base*

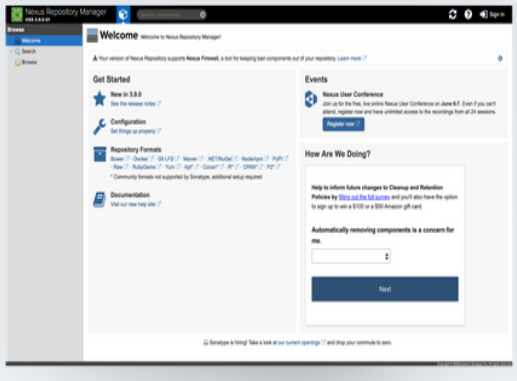
Sample population of 2164 Java professionals, sample error 2.1%

Nexus: Why?

10 million developers trust Nexus.

The world's most popular repository

- Centralized repository for managing all popular component formats
- Single source of truth for all binaries and build artifacts.
- Gain insight into component security, license, and quality issues.

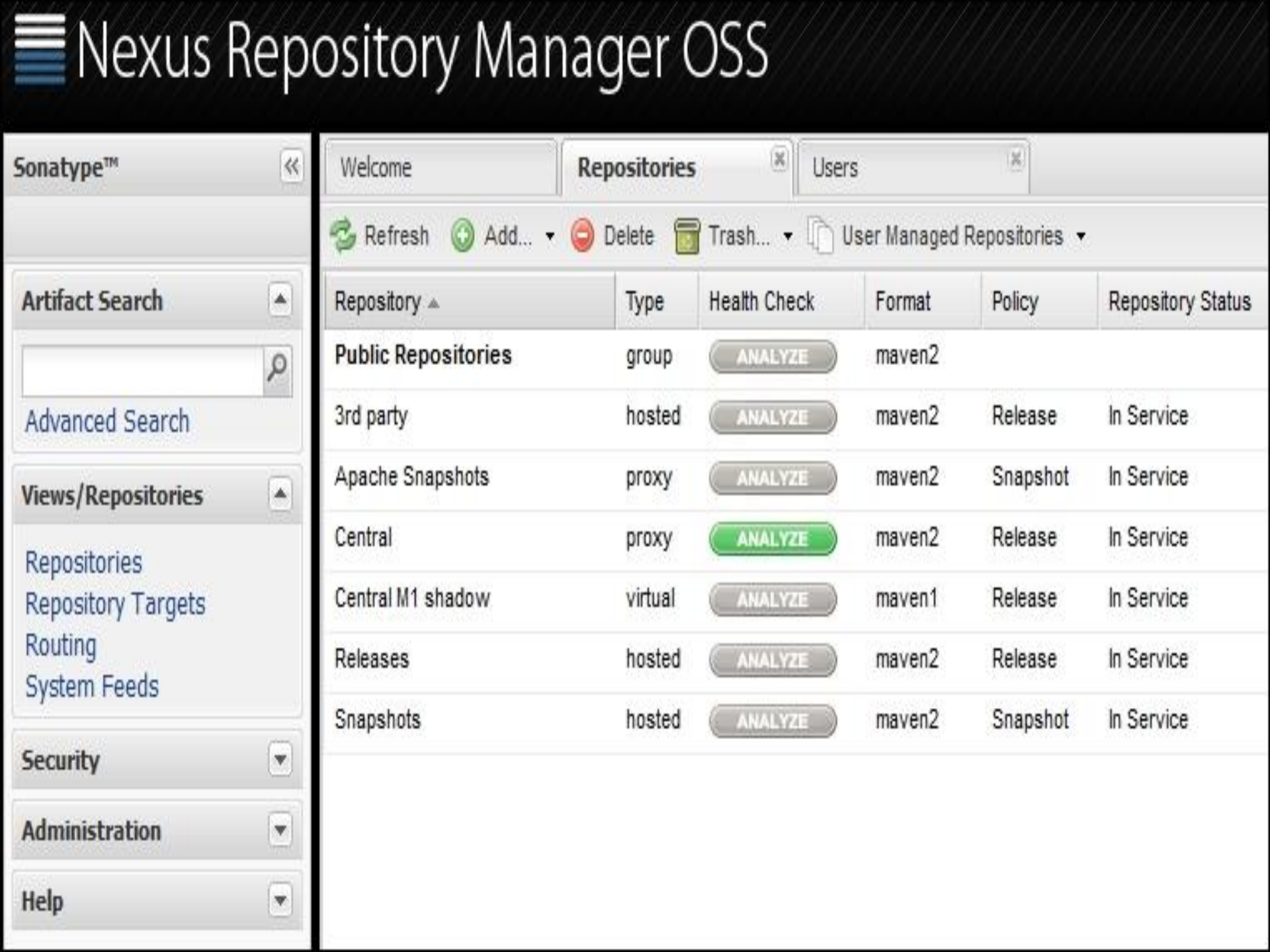


Universal support for all popular formats

- Store and distribute Maven/Java, npm, NuGet, RubyGems, Docker, P2, OBR, APT and YUM and more.
- Manage components from dev through delivery: binaries, containers, assemblies, and finished goods.
- Awesome support for the Java Virtual Machine (JVM) ecosystem, including Gradle, Ant, Maven, and Ivy.
- Compatible with popular tools like Eclipse, IntelliJ, Hudson, Jenkins, Puppet, Chef, Docker, and more.

4. Artifacts Repository Management – Nexus

- Download Nexus:
  - https://help.sonatype.com/repomanager3/download
- Sau khi cài đặt, Nexus có thể được truy cập từ địa chỉ:
  - http://localhost:8081/nexus
    - Xem hình (dưới)

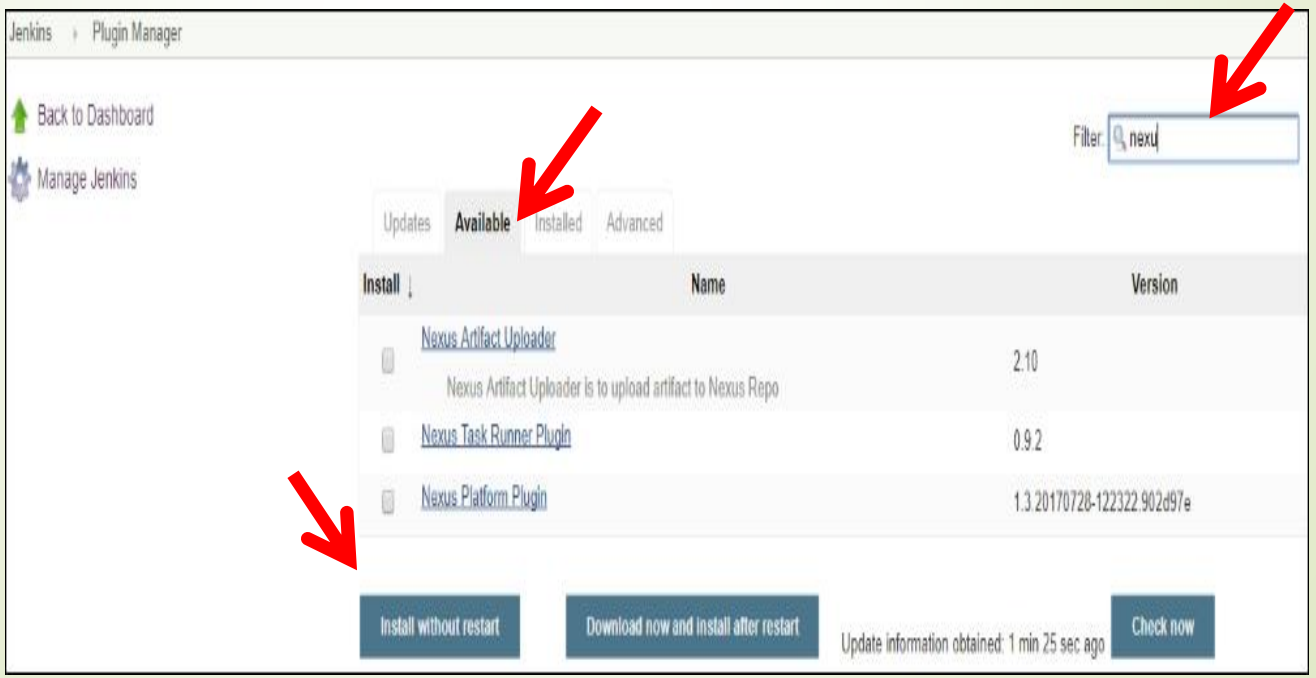

Nexus Repository Manager OSS
Sonatype™
Artifact Search
Advanced Search
Views/Repositories
Repositories
Repository Targets
Routing
System Feeds
Security
Administration
Help
Welcome
Repositories
Users
Refresh
Add...
Delete
Trash...
User Managed Repositories
Repository
Type
Health Check
Format
Policy
Repository Status
Public Repositories
group
ANALYZE
maven2
3rd party
hosted
ANALYZE
maven2
Release
In Service
Apache Snapshots
proxy
ANALYZE
maven2
Snapshot
In Service
Central
proxy
ANALYZE
maven2
Release
In Service
Central M1 shadow
virtual
ANALYZE
maven1
Release
In Service
Releases
hosted
ANALYZE
maven2
Release
In Service
Snapshots
hosted
ANALYZE
maven2
Snapshot
In Service

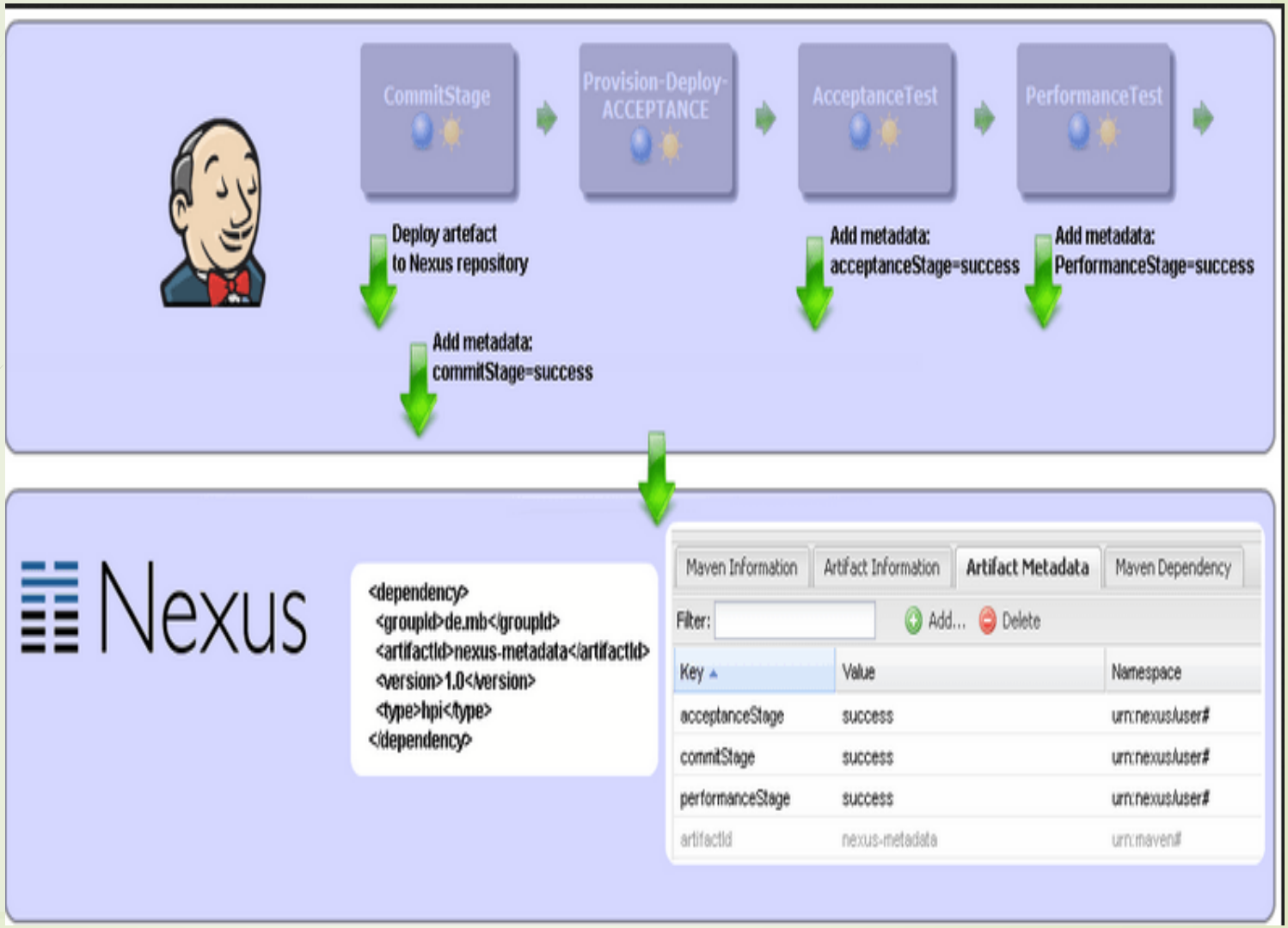
4. Artifacts Repository Management – Nexus

- Hướng dẫn sử dụng Nexus:
  - Xem link:
    - https://huongdanjava.com/vi/nexus-repository-manager

4. Artifacts Repository Management – Nexus

- Tích hợp Nexus với Jenkins:




CommitStage
Provision-Deploy-ACCEPTANCE
AcceptanceTest
PerformanceTest
Deploy artefact to Nexus repository
Add metadata: commitStage=success
Add metadata: acceptanceStage=success
Add metadata: PerformanceStage=success
Nexus
<dependency>
<groupId>de.mb</groupId>
<artifactId>nexus-metadata</artifactId>
<version>1.0</version>
<type>hpi</type>
</dependency>
Maven Information
Artifact Information
Artifact Metadata
Maven Dependency
Filter:
Add...
Delete
Key
Value
Namespace
acceptanceStage
success
urn:nexus/user#
commitStage
success
urn:nexus/user#
performanceStage
success
urn:nexus/user#
artifactId
nexus-metadata
urn:maven#

III. (Core) DevOps Processes

1. Source code management – Git&GitHub
2. Build management – Maven
3. Source code review – Gerrit
4. Repository management – Nexus
5. **Test Automation – Selenium**
6. Continuous deployment – Pipelines
7. Continuous integration - Jenkins

5. Test Automation – Selenium

- Jenkins cung cấp nhiều chức năng và plugins cho testing.
  - List of Unit testing plugins sẵn dùng:

- JUnit itself
- AUnit
- MSTest (imported from MSTest Plugin)
- NUnit (imported from NUnit Plugin)
- UnitTest++
- Boost Test Library
- PHPUnit
- Free Pascal Unit
- CppUnit
- MbUnit
- Google test
- EmbUnit
- gtester/glib
- QTestLib

5. Test Automation – Selenium

i. Setting up Unit Testing cho dự án trong Jenkins

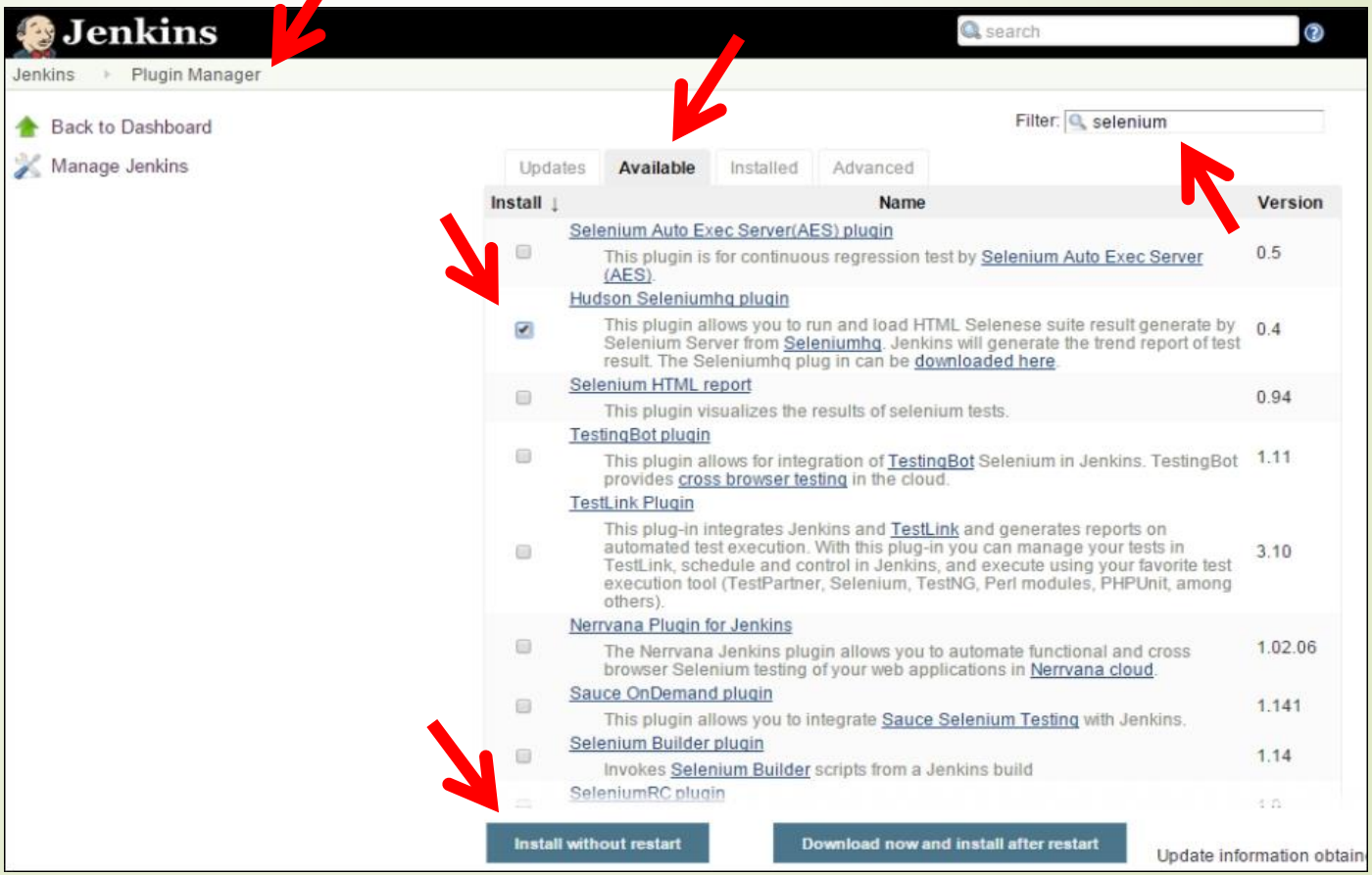
ii. Thiết lập Automated Test Suite sử dụng Selenium

- Download Selenium:

  Link: http://selenium-release.storage.googleapis.com/index.html?path=2.48/

- Tích hợp Selenium với Jenkins
  - Xem hình (dưới)

5. Test Automation – Selenium

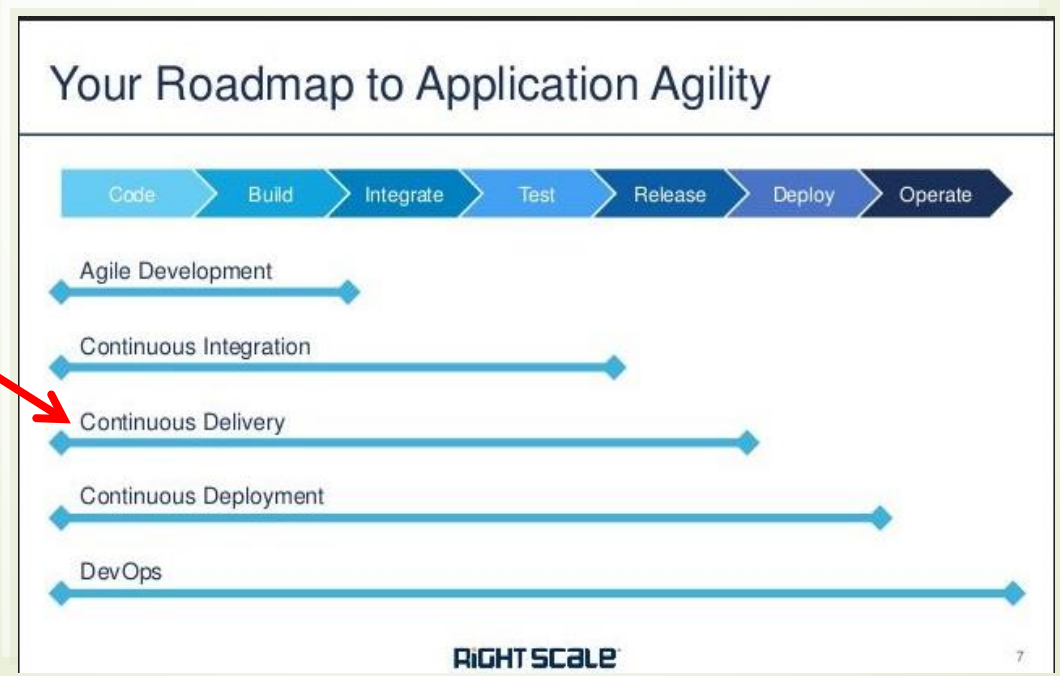


III. (Core) DevOps Processes

1. Source code management – Git&GitHub
2. Build management – Maven
3. Source code review – Gerrit
4. Repository management – Nexus
5. Test Automation – Selenium
6. **Continuous deployment – Pipelines**
7. Continuous integration - Jenkins

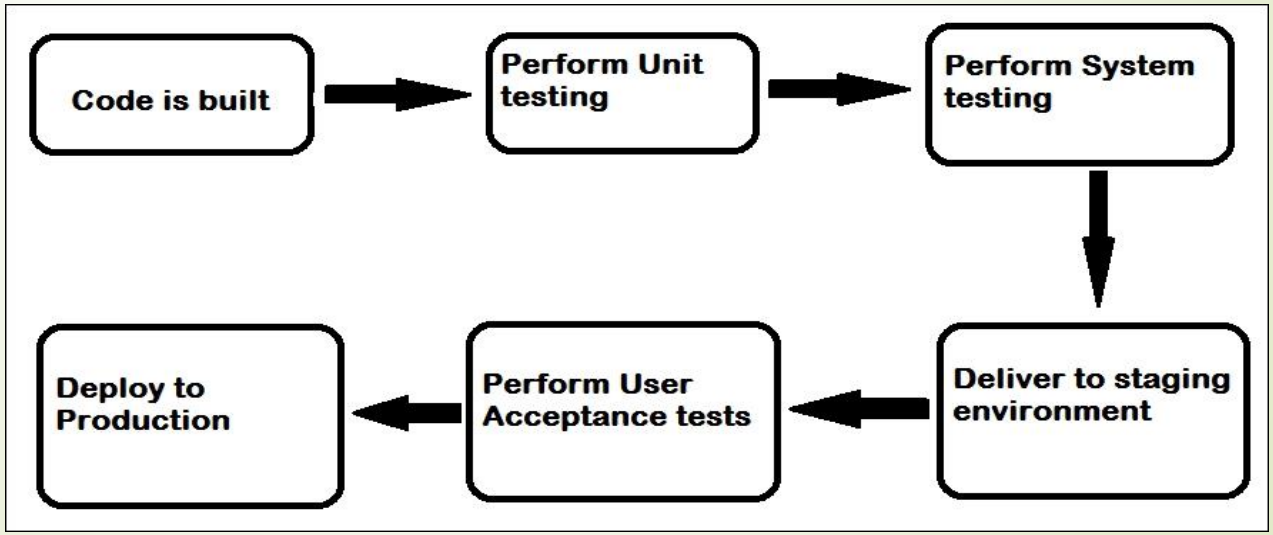
6. Continuous deployment – Pipelines

- Why?



6. Continuous deployment – Pipelines

- Why?
  - Continuous delivery (CD) ~ pipeline: code → deployment



6. Continuous deployment – Pipelines

- Jenkins Pipeline (or Pipeline):
  - ~ Tích hợp bộ plugins cho CD để mô hình hóa các CD pipeline từ đơn giản đến phức tạp
    - "Pipeline as code"
    - Ngôn ngữ sử dụng DSL: Pipeline domain-specific language (DSL) syntax.

6. Continuous deployment – Pipelines

- Định nghĩa 1 Jenkins Pipeline được viết trong 1 text file (gọi là Jenkinsfile)
  - **Jenkinsfile**: có thể được commit to a project's source control repository và được versioned và reviewed giống như các code khác.
    - Đây là nền tảng của "Pipeline-as-code";
      - ~ xem CD pipeline như một phần code của ứng dụng

6. Continuous deployment – Pipelines

- A Pipeline:
  - is a user-defined model of a CD pipeline.
    - A Pipeline's code defines your entire build process: gồm các giai đoạn: building an application, testing it and then delivering it.
  - Có 2 Pipeline syntax
    i. **Declarative Pipeline syntax, và**
    ii. **Scripted Pipeline syntax.**

i. Declarative Pipeline fundamentals

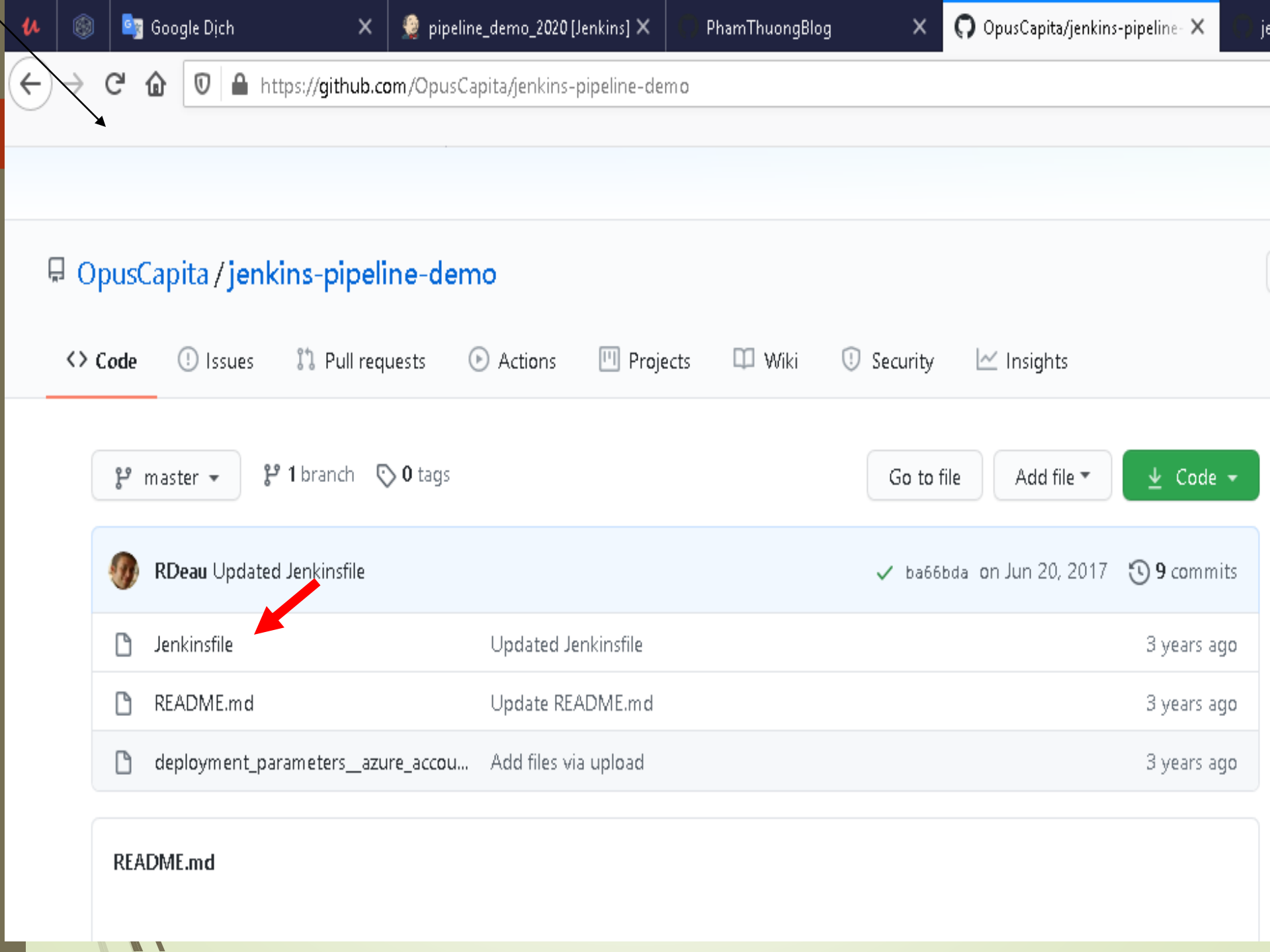
Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent any (1)
    stages {
        stage('Build') { (2)
            steps {
                // (3)
            }
        }
        stage('Test') { (4)
            steps {
                // (5)
            }
        }
        stage('Deploy') { (6)
            steps {
                // (7)
            }
        }
    }
}
```

1. Execute this Pipeline or any of its stages, on any available agent.
2. Defines the "Build" stage.
3. Perform some steps related to the "Build" stage.
4. Defines the "Test" stage.
5. Perform some steps related to the "Test" stage.
6. Defines the "Deploy" stage.
7. Perform some steps related to the "Deploy" stage.

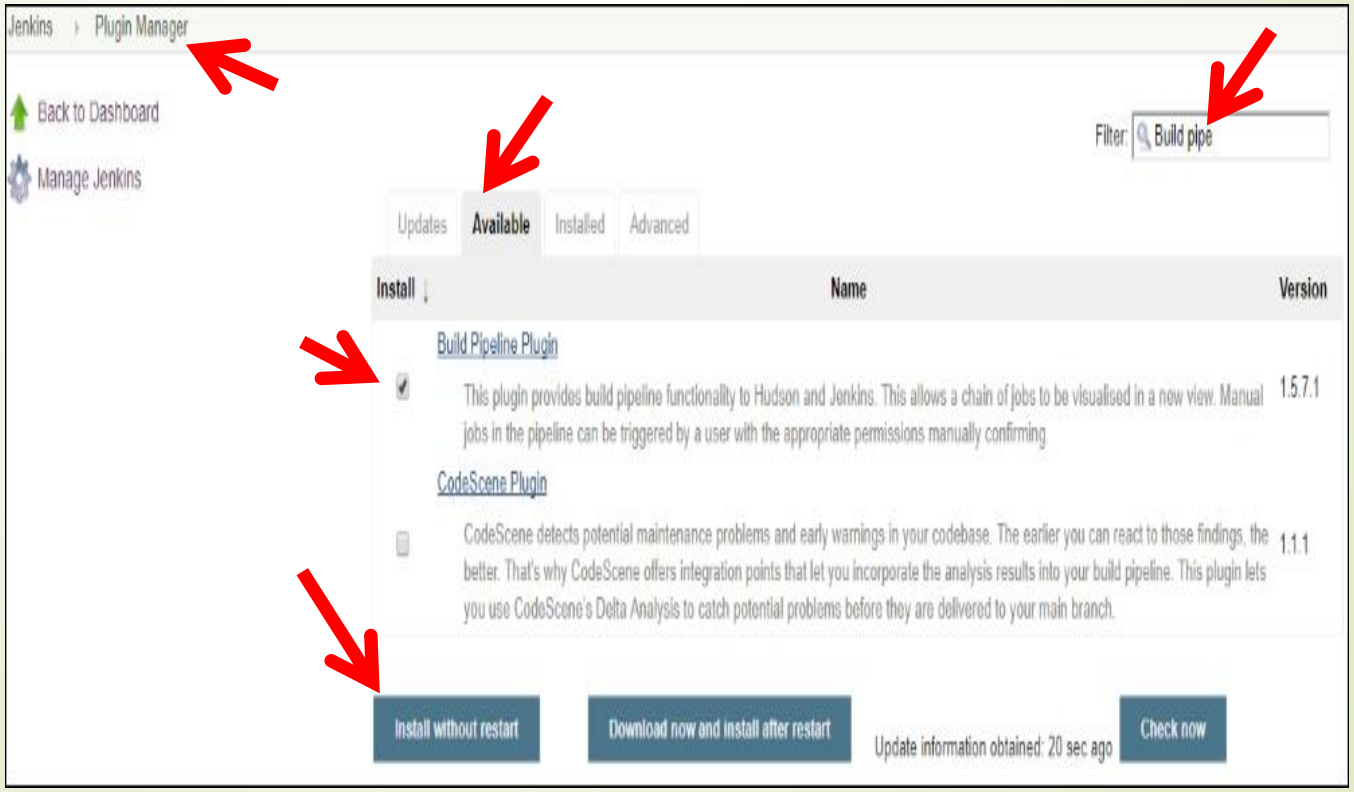
i. Declarative Pipeline fundamentals

- Ví dụ:
  - Thực hành trên tạo Job-pipeline: pipeline_demo_2020
    - Link GitHub của dự án demo:
      - https://github.com/OpusCapita/jenkins-pipeline-demo
    - Jenkinsfile của dự án:
      - https://github.com/OpusCapita/jenkins-pipeline-demo/blob/master/Jenkinsfile.
      - Click here to see file content
    - Địa chỉ kho GitHub của dự án:
      - https://github.com/OpusCapita/jenkins-pipeline-demo.git


Google Dịch
pipeline_demo_2020 [Jenkins]
PhamThuongBlog
OpusCapita/jenkins-pipeline-
https://github.com/OpusCapita/jenkins-pipeline-demo
OpusCapita / jenkins-pipeline-demo
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
master
1 branch
0 tags
Go to file
Add file
Code
RDeau Updated Jenkinsfile
ba66bda on Jun 20, 2017
9 commits
Jenkinsfile
Updated Jenkinsfile
3 years ago
README.md
Update README.md
3 years ago
deployment_parameters__azure_accou...
Add files via upload
3 years ago
README.md

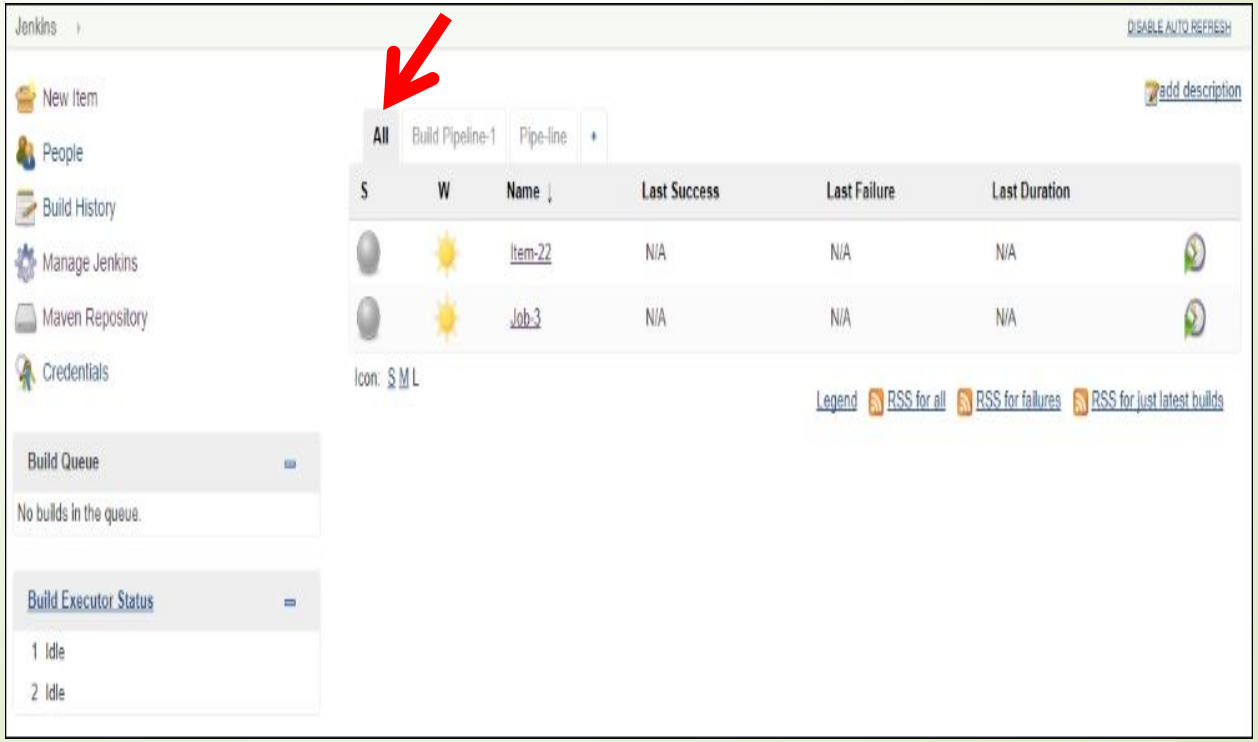
6. Continuous deployment – Pipelines

- Cài đặt Build Pipeline với Jenkins:



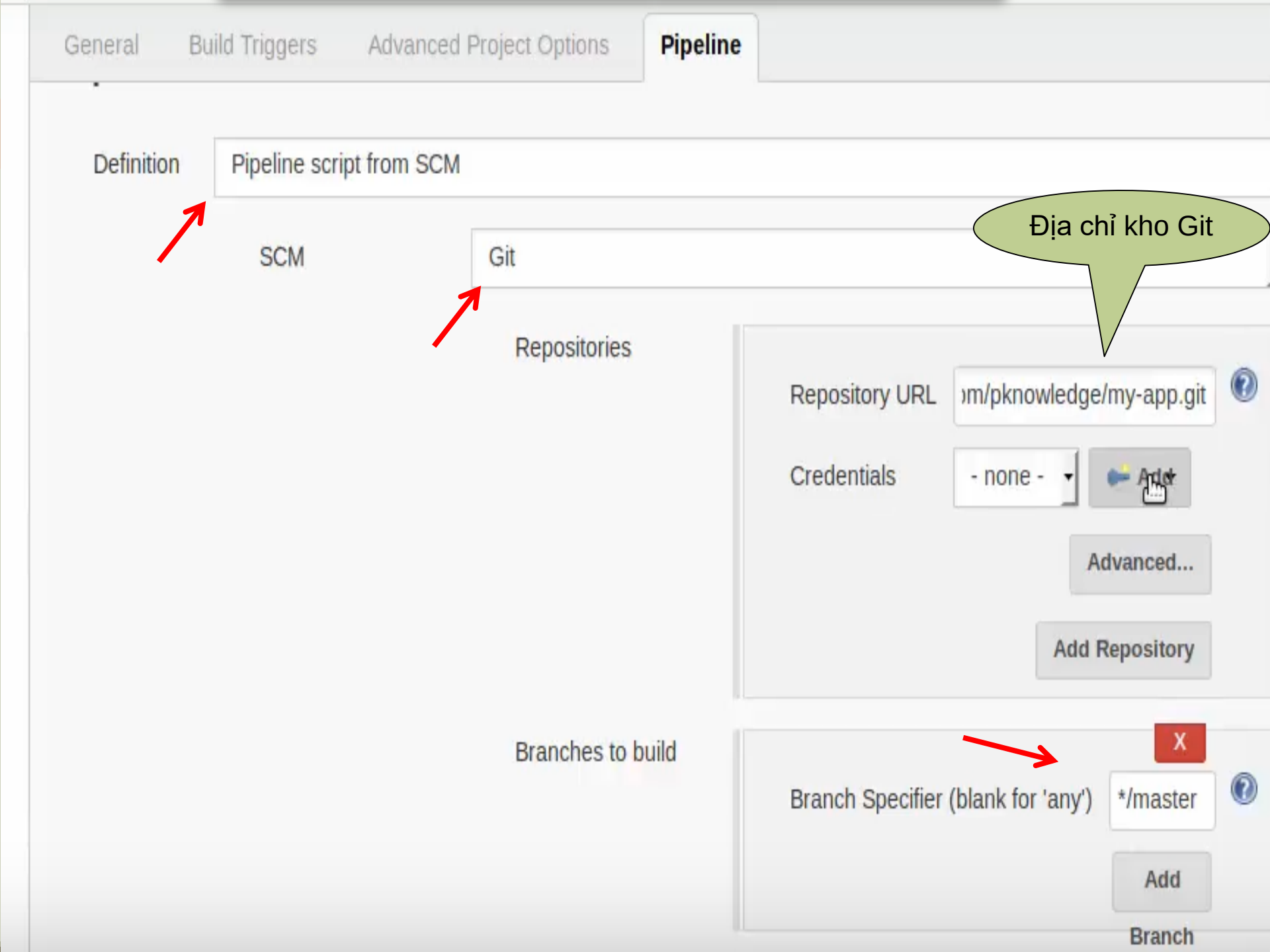
6. Continuous deployment – Pipelines

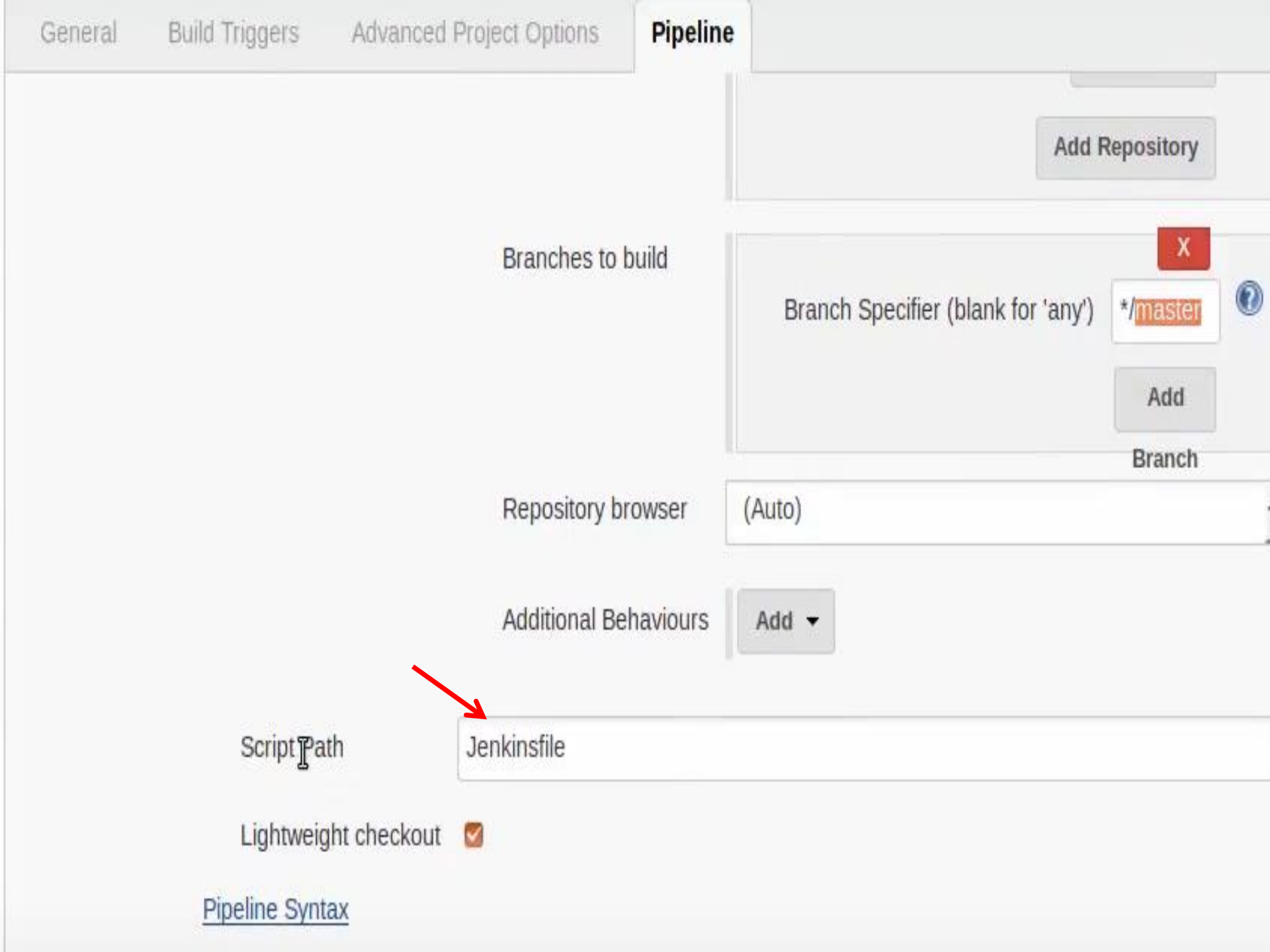
- Thiết lập Build Pipeline cho dự án cụ thể để làm việc với nó:

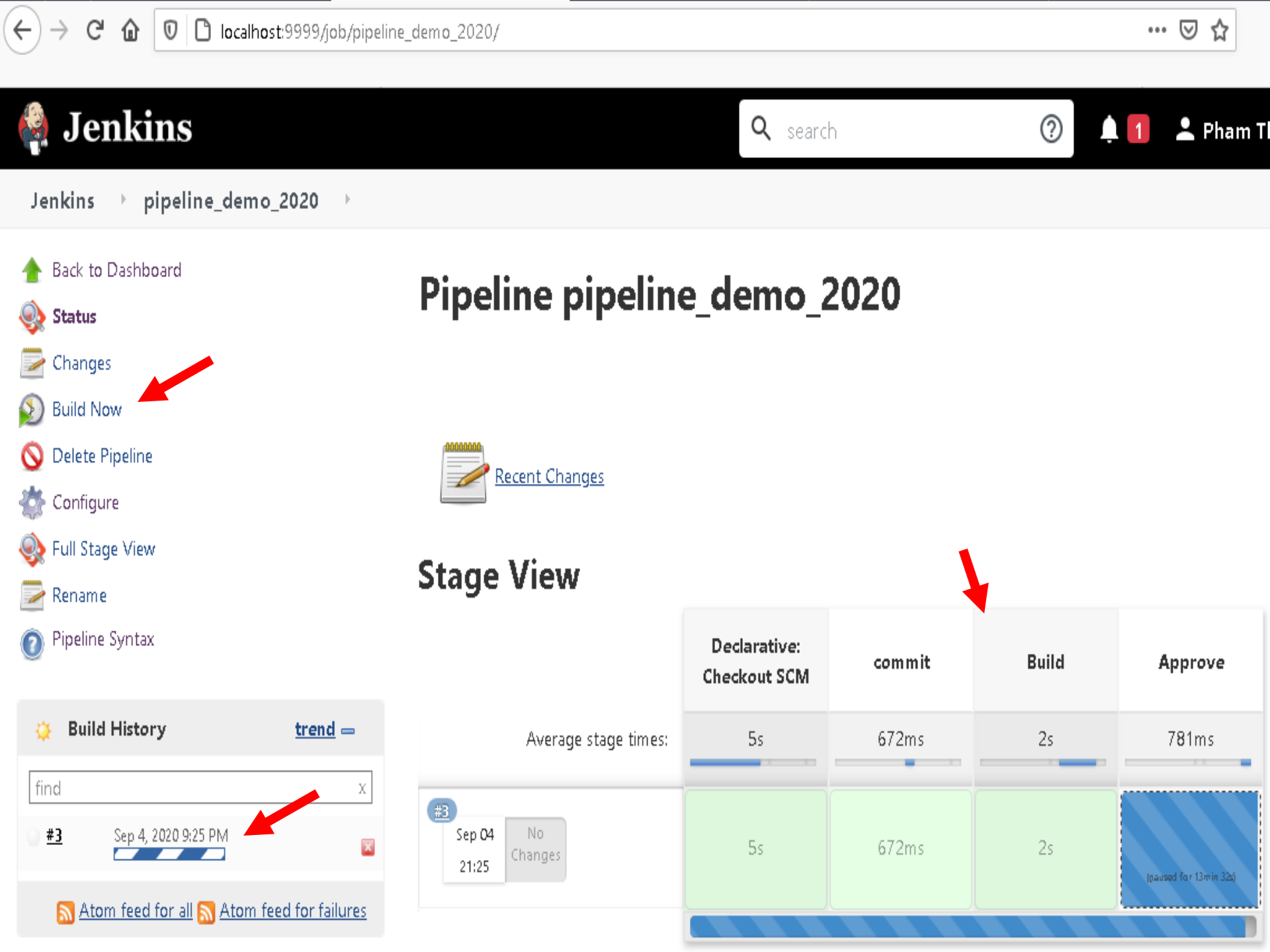


6. Continuous deployment – Pipelines

- Tạo Job trong Jenkins:
  - Tên: Pileline_Demo_2020
  - Kiểu dự án: Pipeline
  - Thiết lập cấu hình Pipeline cho dự án:
    - Xem hình (dưới)


General
Build Triggers
Advanced Project Options
Pipeline
Definition
Pipeline script from SCM
SCM
Git
Địa chỉ kho Git
Repositories
Repository URL
m/pknowledge/my-app.git
Credentials
- none -
Add
Advanced...
Add Repository
Branches to build
X
Branch Specifier (blank for 'any')
*/master
Add
Branch


General
Build Triggers
Advanced Project Options
Pipeline
Add Repository
Branches to build
X
Branch Specifier (blank for 'any')
*/master
Add
Branch
Repository browser
(Auto)
Additional Behaviours
Add
Script Path
Jenkinsfile
Lightweight checkout
Pipeline Syntax


localhost:9999/job/pipeline_demo_2020/
Jenkins
search
1
Pham T
Jenkins
pipeline_demo_2020
Back to Dashboard
Status
Changes
Build Now
Delete Pipeline
Configure
Full Stage View
Rename
Pipeline Syntax
Pipeline pipeline_demo_2020
Recent Changes
Stage View
Declarative: Checkout SCM
commit
Build
Approve
Average stage times:
5s
672ms
2s
781ms
#3
Sep 04 21:25
No Changes
5s
672ms
2s
Build History
trend
find
#3
Sep 4, 2020 9:25 PM
Atom feed for all
Atom feed for failures

6. Continuous deployment – Pipelines

- Tóm lại: Jenkins Pipeline

- **Jenkins pipeline** is a single platform that runs the entire *pipeline as code*
- All the standard jobs defined by Jenkins are manually written in one script and they can be stored in a VCS
- Instead of building several jobs for each phase, you can now code the entire workflow and put it in a Jenkinsfile

Hoàn toàn có thể tùy biến các stage để code một Pipeline DevOps by Jenkins đơn giản hay phức tạp cho ứng dụng cụ thể.

III. (Core) DevOps Processes

1. Source code management – Git&GitHub
2. Build management – Maven
3. Source code review – Gerrit
4. Repository management – Nexus
5. Test Automation – Selenium
6. Continuous deployment – Pipelines
7. **Continuous integration - Jenkins**

7. Continuous integration - Jenkins

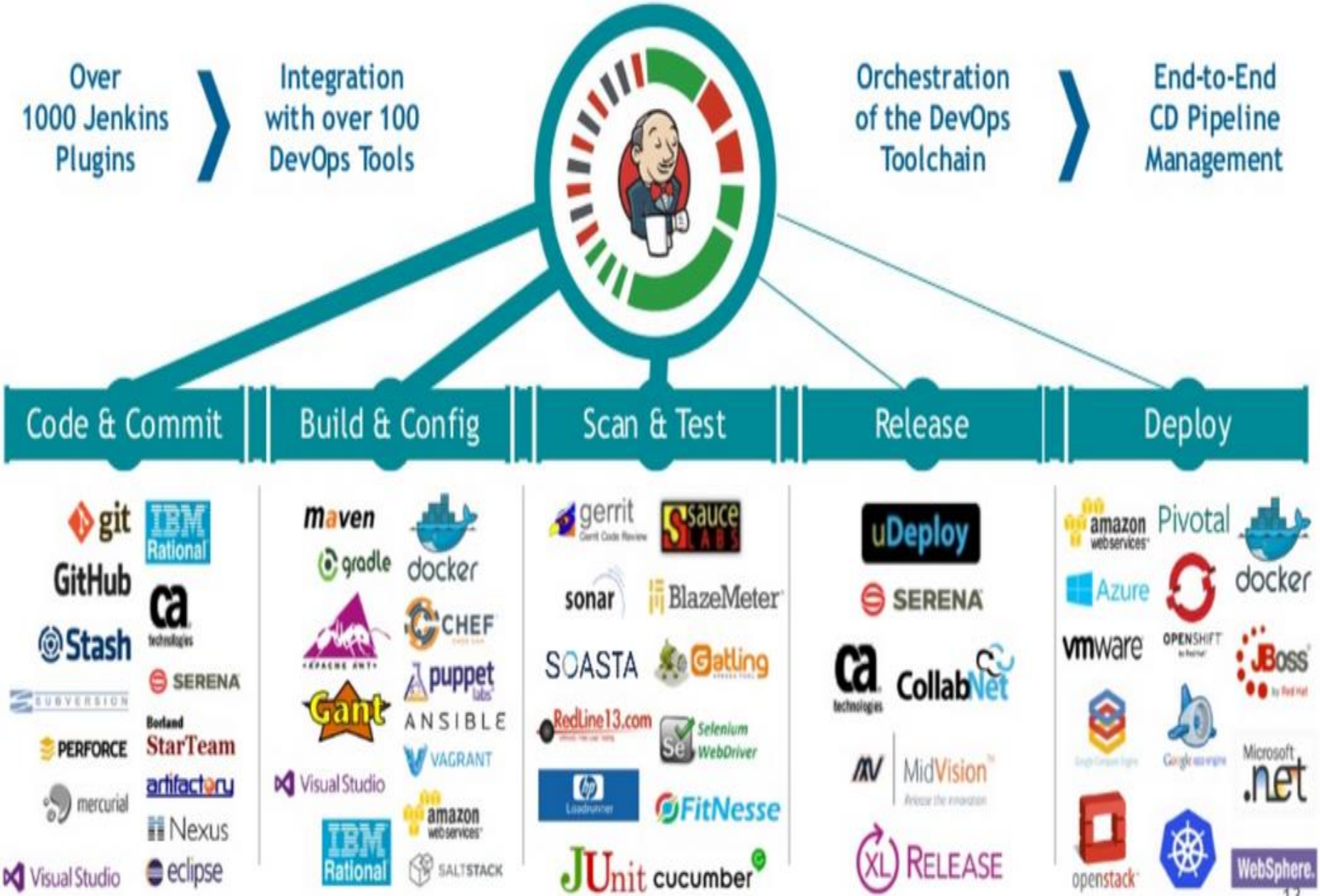
Jenkins

- ~ web application chạy trên nền J2EE
  - Đóng vai trò là một máy chủ build & test hệ thống tích hợp liên tục (Continuous Integration) .
    - Jenkins được viết bởi Kosuke Kawaguchi (Sun), Hudson là tiền thân của nó
      - Phát triển trên nền tảng Java.
    - Khi Oracle mua lại Sun (2010) => Hudson đã tách ra phát triển riêng và đặt tên là Jenkins.

Jenkins

- Đặc điểm nổi trội:
  - Đa nền tảng.
  - Hỗ trợ hầu hết các công nghệ phần mềm.
  - Hỗ trợ đầy đủ với các hệ thống tích hợp dễ dàng thông qua plugin.
  - Đóng gói quy trình phát triển phần mềm một cách tự động.

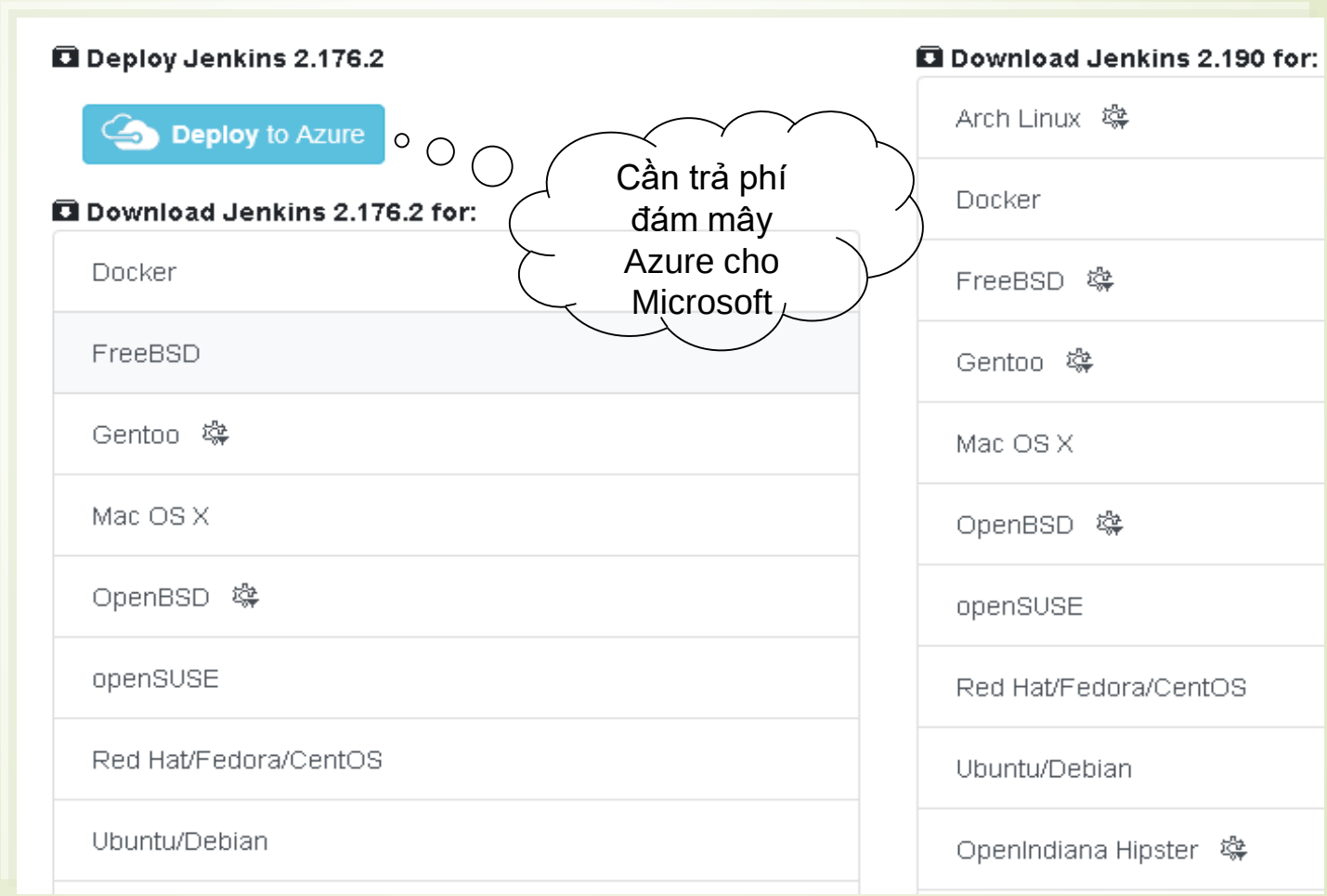
Jenkins is the Hub of the CD/DevOps Ecosystem



Jenkins

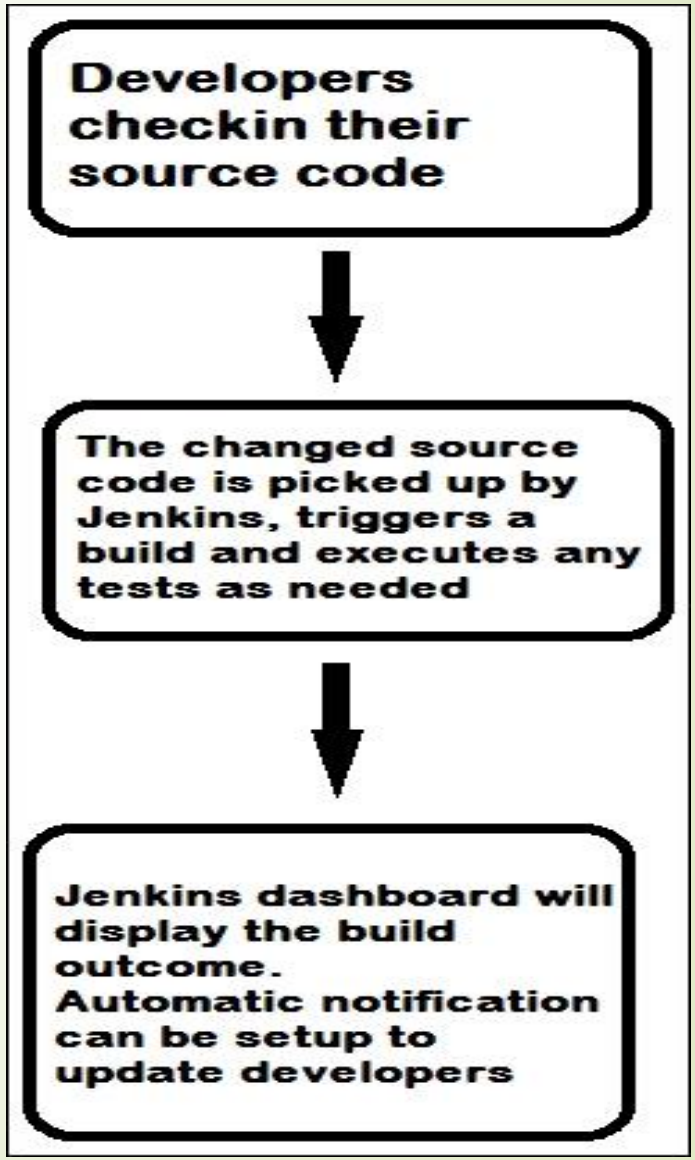
- Modes of installing:
  - **Standalone:**
    - Run standalone in its own process using its own built-in web server (Jetty) for experimentation and small projects
  - Servlet-based:
    - Run as one servlet framework for development projects
  - Multi-node setup for staging or production:
    - Distributed client-server setup; the Jenkins advanced installation procedure is recommended

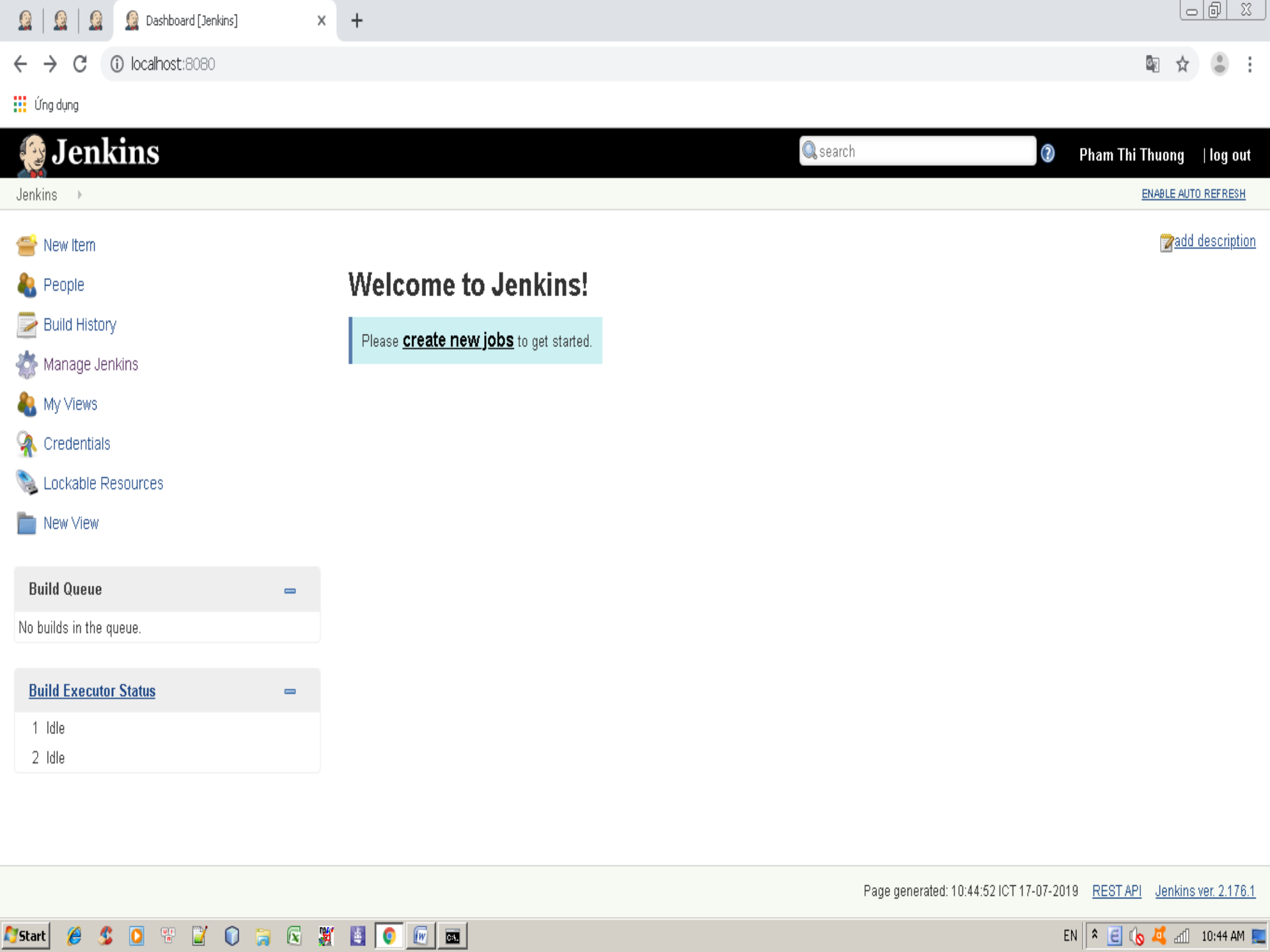
Jenkins


Deploy Jenkins 2.176.2
Deploy to Azure
Download Jenkins 2.176.2 for:
Docker
FreeBSD
Gentoo
Mac OS X
OpenBSD
openSUSE
Red Hat/Fedora/CentOS
Ubuntu/Debian
Cần trả phí đám mây Azure cho Microsoft
Download Jenkins 2.190 for:
Arch Linux
Docker
FreeBSD
Gentoo
Mac OS X
OpenBSD
openSUSE
Red Hat/Fedora/CentOS
Ubuntu/Debian
OpenIndiana Hipster

Jenkins

- Luồng tiến trình CI
  - Hình (kế)
- Cài đặt Jenkins thành công
  - KQ: màn hình (dưới)




Dashboard [Jenkins]
localhost:8080
Ứng dụng
Jenkins
search
Pham Thi Thuong
log out
Jenkins
ENABLE AUTO REFRESH
New Item
People
Build History
Manage Jenkins
My Views
Credentials
Lockable Resources
New View
add description
Welcome to Jenkins!
Please create new jobs to get started.
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Page generated: 10:44:52 ICT 17-07-2019
REST API
Jenkins ver. 2.176.1
Start
EN
10:44 AM

Manage Jenkins [Jenkins]

localhost:8080/manage

Ứng dụng

Jenkins

search

Pham Thi Thuong | log out

Jenkins

ENABLE AUTO REFRESH

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Credentials
- Lockable Resources
- New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Manage Jenkins

Màn hình thiết lập cấu hình các dự án Jenkin

Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials
Configure the credential providers and types

Global Tool Configuration
Configure tools, their locations and automatic installers.

Global Tool Configuration

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

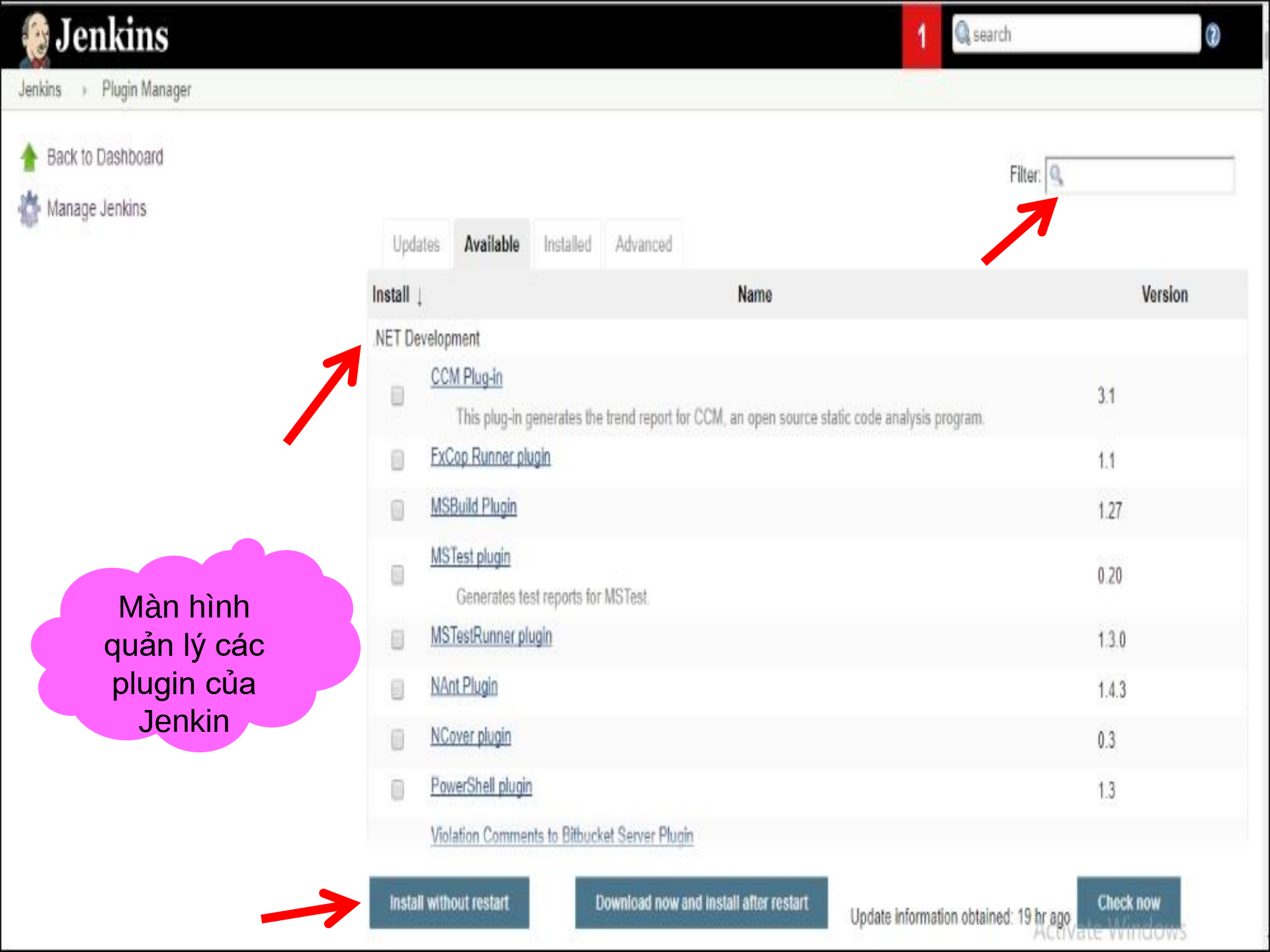
Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
There are updates available

System Information

localhost:8080/configureTools

Start

EN 10:45 AM


Jenkins
1
search
Jenkins › Plugin Manager
Back to Dashboard
Manage Jenkins
Filter:
Updates
Available
Installed
Advanced
Install
Name
Version
.NET Development
CCM Plug-in
This plug-in generates the trend report for CCM, an open source static code analysis program.
3.1
FxCop Runner plugin
1.1
MSBuild Plugin
1.27
MSTest plugin
Generates test reports for MSTest.
0.20
MSTestRunner plugin
1.3.0
NAnt Plugin
1.4.3
NCover plugin
0.3
PowerShell plugin
1.3
Violation Comments to Bitbucket Server Plugin
Install without restart
Download now and install after restart
Update information obtained: 19 hr ago
Check now
Màn hình quản lý các plugin của Jenkin

Tóm lại

I. (Full) DevOps framework
II. (Core) DevOps framework
III. Processes of (core) DevOps framework:
    1. Source code management – Git&GitHub
    2. Build management – Maven
    3. Source code review – Gerrit
    4. Repository management – Nexus
    5. Test Automation – Selenium
    6. Continuous deployment – Pipelines
    7. Continuous integration - Jenkins

Discussion

Preparing for next lesson

- Topic 3: Đóng gói, phát hành phần mềm

Tài liệu tham khảo

- https://viblo.asia/p/part-1-gerrit-code-review-with-jenkins-ci-introduction-XQZGxoMmvwA
  - Part 1 → Part 4
- https://www.javaguides.net/2019/06/jsp-servlet-projects-with-source-code-free-download.html
- https://stackoverflow.com/questions/60031044/how-to-change-mavens-remote-repository-url-in-the-netbeans-ide-from-http-to-ht