|  |
| --- |
| **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI**  **KHOA CÔNG NGHỆ THÔNG TIN**    01 - Logo - en - 3  **BÁO CÁO**  **HỌC PHẦN: QUY TRÌNH VÀ CÔNG CỤ PHÁT TRIỂN PHẦN MỀM**  **Công cụ phát triển phần mềm Git**  **Nhóm 15**  **Đoàn Thanh Tâm**  **Đỗ Trung Nghĩa**  **Phùng Quang Nhật**  **Nguyễn Văn Mạnh**  **Gvhd: Ths Nguyễn Văn Cường**  ***Hà Nội, tháng 08 năm 2022*** |

# CHƯƠNG I: GIT

1. **Git là gì:**

**Git** là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System - DVCS) được phát triển bởi [Linus Torvalds](https://vi.wikipedia.org/wiki/Linus_Torvalds) vào năm [2005](https://vi.wikipedia.org/wiki/2005), ban đầu dành cho việc phát triển [nhân Linux](https://vi.wikipedia.org/wiki/H%E1%BA%A1t_nh%C3%A2n_Linux). Hiện nay, Git trở thành một trong các phần mềm quản lý [mã nguồn](https://vi.wikipedia.org/wiki/M%C3%A3_ngu%E1%BB%93n) phổ biến nhất. Git là phần mềm mã nguồn mở được phân phối theo giấy phép công cộng [GPL2](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_C%C3%B4ng_c%E1%BB%99ng_GNU).

Git có khả năng chạy trên nhiều hệ điều hành khác nhau như Linux, Windows, Mac OS X …

Có nhiều website và phần mềm cho phép lưu trữ (miễn phí và có phí) Git.

* [GitHub](https://vi.wikipedia.org/wiki/GitHub)
* [GitLab](https://vi.wikipedia.org/w/index.php?title=GitLab&action=edit&redlink=1)
* [Gitorious](https://vi.wikipedia.org/w/index.php?title=Gitorious&action=edit&redlink=1)
* [Sourceforge](https://vi.wikipedia.org/wiki/SourceForge)
* [GNU Savannah](https://vi.wikipedia.org/w/index.php?title=GNU_Savannah&action=edit&redlink=1)
* [GitEnterprise](https://vi.wikipedia.org/w/index.php?title=GitEnterprise&action=edit&redlink=1)

# 1.2. Khái niệm cơ bản trong Git:

* Repository

Repository hay được gọi tắt là Repo là nơi chứa tất cả những thông tin cần thiết để duy trì và quản lý các sửa đổi và lịch sử của toàn bộ project.

Git có 2 loại là remote repository và local repository.

* Local repository

là repository bố trí trên máy local của lập trình viên, dành cho một người dùng sử dụng.

* Remote repository

là repository để chia sẻ giữa nhiều người và bố trí trên server chuyên dụng.

* Branch

Đây là thế mạnh của git. Với git, việc quản lý branch - nhánh rất dễ dàng. Mỗi nhánh trong Git gần giống như một workspace. Việc nhảy vào một nhánh để làm việc trong đó tương tự việc chuyển qua workspace mới và cũng có thể quay lại workspace cũ một cách nhanh chóng.

* Commit

Hành động chuyển trạng thái sang committed là commit. Hay commit là hành động record lại thay đổi trong repository của bạn.

committed: tức là git đã snapshot lại file ở Staging Area phía trên với 1 unique index trong Git directory.

* Clone

Clone là hành động thực hiện tải một bản sao có sẵn của một remote repository server nào đó (có thể là dự án bạn tham gia)

* Push

Push là hành động đưa nội dung của local\_repo lên remote\_repo. Push là cách bạn chuyển giao các commit từ local\_repo của bạn lên remote\_repo.

* Pull và Fetch

Git Pull và Git Fetch là có chức năng tương đồng với nhau. Cả hai đều được sử dụng để tải xuống dữ liệu mới từ một remote\_repo. Tuy nhiên, Git Fetch thường được coi là một phiên bản an toàn hơn của Git Pull.

Khi thực hiện fetch cùng với remote\_origin, Git sẽ tải về dữ liệu của tất cả các branch của repository trên remote server nằm tại địa chỉ quy định bởi remote\_origin và cập nhật dữ liệu này với dữ liệu của cách branch phía dưới máy local.

Tuy nhiên git fetch không cập nhật dữ liệu của working directory (Thư mục làm việc). Điều này có nghĩa là nếu như có bất cứ commit nào trên remote server thì chúng cũng không ảnh hưởng tới các tập tin, thư mục của bạn.

Khi thực hiện pull cùng với remote\_origin và branch\_name:

Git sẽ áp thực hiện việc fetch dữ liệu của Git repository tại nhánh branch\_name từ server nằm tại địa chỉ quy định bởi remote\_origin và áp dụng (merge) các thay đổi này vào thư mục và tập tin ở working directory. Git pull sẽ có thể gây ra xung đột (conflict) trong khi merge.

# 1.3. Các lệnh Git cơ bản lập trình viên cần nắm

Git được chia thành nhiều lệnh, từ đơn giản đến phức tạp. Vậy git là gì, các lệnh Git cơ bản bạn nên lưu ý là những lệnh nào?

* Câu lệnh cơ bản trong github

- Git config

Lệnh Git Config được sử dụng để thiết lập tên đăng nhập và email của người dùng trong main configuration file. Bạn có thể dễ dàng kiểm tra tên và kiểu email bằng cách dùng lệnh: **git config – global user.name và git config – global user.email.**

Để cập nhật một tên và email mới, bạn có thể dùng lệnh git config – global user.name/email = “tên/email”.

- Git init

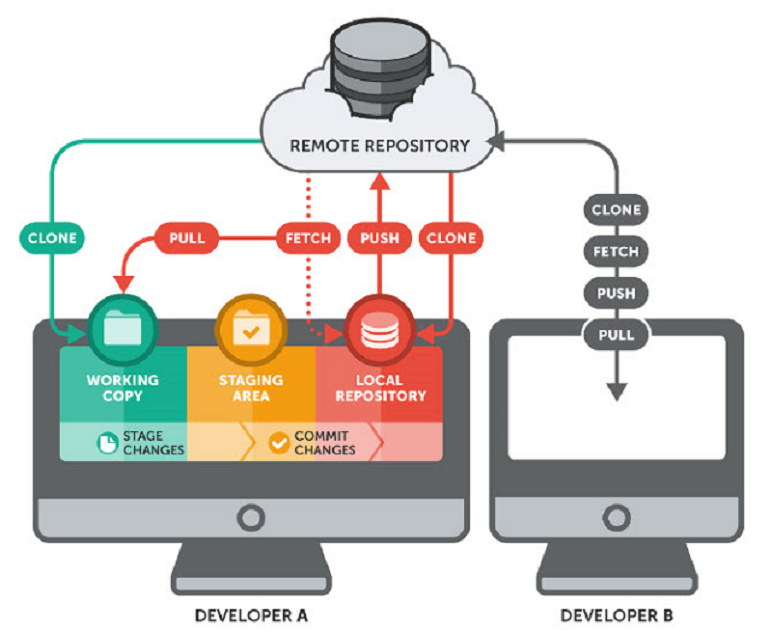
Bạn có thể dùng lệnh Git init để tạo ra một git repository trong 1 dự án mới hoặc đã có sẵn. Lệnh này được sử dụng trong thư mục gốc của dự án.

- Git clone

Lệnh Git clone cho phép người dùng Copy 1 git repository từ kho lưu trữ từ xa. Bạn chỉ cần dùng lệnh: **git clone <:clone git url:>.**

- Git status

Lệnh git status được sử dụng khi người dùng cần check trạng thái của các file đã được sửa đổi trong thư mục trong quá trình làm việc. Chẳng hạn như bạn có thể kiểm tra các thay đổi từ lần commit gần đây nhất.

* *Git Add được sử dụng khi cần đổi Stage hoặc index*

- Git add

Git Add được sử dụng khi cần đổi Stage hoặc index đến thư mục bạn đang làm việc.

- Git commit

Đây là lệnh cực kỳ phổ biến và bạn không thể bỏ qua khi tìm hiểu**Git là gì.** Git commit có tác dụng giúp Git lưu lại một ảnh chụp màn hình/snapshot đối với các thay đổi trong thư mục trong quá trình làm việc với Git.

Với Git, khi Commit, các thay đổi sẽ được tự động lưu lại và thường nằm trong mục Staging Area. Trong hệ thống cũng sẽ lưu lại tên người chỉnh sửa để người dùng có thể dễ dàng theo dõi. Hệ thống cũng sẽ lưu trữ tên và email của người thực hiện chính sửa. Bạn cũng có thể khôi phục lại các tệp tin và chuyển sang một nhánh khác.

- Git push/git pull

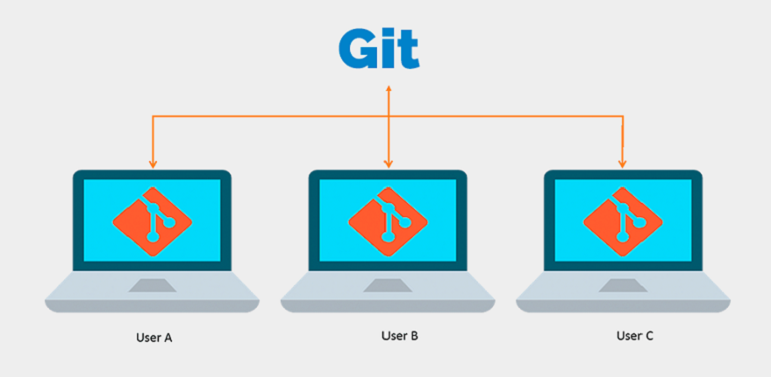
Lệnh Push hay pull cho phép người sử dụng có thể thay đổi tới các kho lưu trữ remote. Nếu bạn đã thêm các thay đổi và muốn đưa các thay đổi đó lên đầu, bạn có thể dùng lệnh này để cập nhật và apply các thay đổi đó thông qua việc code. Bạn có thể dùng bằng code: **git pull <:remote:> <:branch:> and git push <:remote:> <:branch:>.**

- Git branch

Bạn có thể dùng lệnh Git branch để liệt kê tất cả các nhánh trong hệ thống. Cách dùng rất đơn giản, bạn có thể dùng Git Branch hoặc Git Branch-a.

- Git checkout

Lệnh git checkout thường bị nhầm với lệnh git branch. Tuy nhiên, đây lại là 2 lệnh hoàn toàn khác nhau. Lệnh Checkout cho phép người dùng có thể chuyển sang một nhánh khác bằng cách dùng code: **git checkout <: branch:> hoặc \*\* \_ git checkout -b <: branch:>**.

* *Bạn có thể dùng lệnh Git branch để liệt kê tất cả các nhánh trong hệ thống*

- Git stash

Git stash cũng là lệnh được rất nhiều người quan tâm khi tìm hiểu **git là gì**. Đây là lệnh có tác dụng lưu các thay đổi trong trường hợp bạn không muốn commit ngay lúc đó. Bạn chỉ cần sử dụng lệnh này trong thư mục làm việc để thực hiện việc lưu thay đổi.

- Git merge

Lệnh git stash được sử dụng khi người dùng cần nối 2 nhánh với nhau. Bạn cần dùng code: **git merge <:branch\_ban\_muon\_merge:>**. Hãy lưu ý chọn nhánh cần chọn trước khi code nhé.

- Git reset

Nếu bạn đã đưa một tệp tin vào mục Staging Area và muốn loại bỏ tệp tin đó, bạn có thể dùng lệnh reset thông qua đường code: **git reset HEAD tên\_file**.

- Git remote

Git remote cho phép người dùng kiểm tra hoặc thêm các kho lưu trữ trực tuyến. Bạn có thể sử dụng lệnh này để liệt kê thông qua đường code: **git remote add <: remote\_url:>**.

- Git add

Thêm một lệnh Git add. Tuy nhiên, lệnh git add này sẽ được sử dụng để đưa tệp tin vào khu vực staging. Khi tìm hiểu **git là gì**, bạn cần lưu ý sự khác biệt giữa 2 lệnh git add để sử dụng chính xác.

Để thêm tệp tin, bạn có thể chọn git add tên file hoặc all nếu muốn thêm tất cả các file có trong thư mục.

# 1.4. Tại sao nên sử dụng Git:

- Lưu lại các phiên bản khác nhau của mã nguồn dự án phần mềm

- Khôi phục lại mã nguồn từ một phiên bản bất kỳ

- Dễ dàng so sánh giữa các phiên bản

- Phát hiện được ai đã sửa phần nào làm phát sinh lỗi

- Khôi phục lại tập tin bị mất

- Dễ dàng thử nghiệm, mở rộng tính năng của dự án mà không làm ảnh hưởng đến phiên bản chính (master branch)

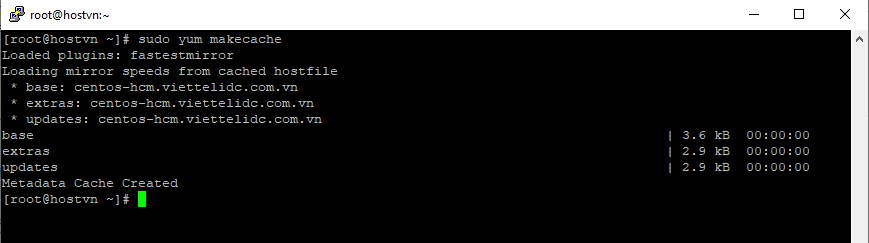
- Giúp phối hợp thực hiện dự án trong nhóm một cách hiệu quả

1.5. Cài Git trên CentOS 7

1.5.1. Cài đặt Git

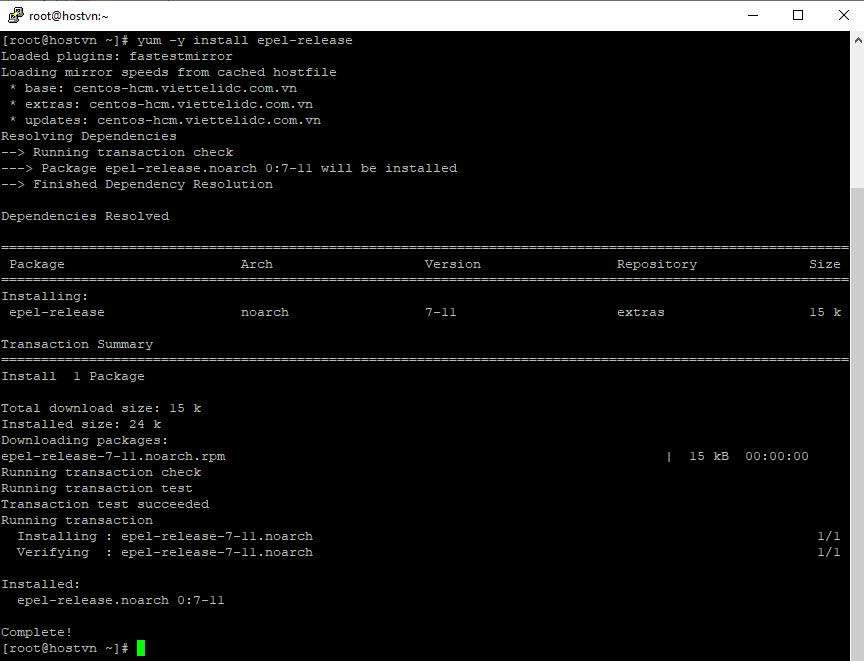
- Git có sẵn trong kho lưu trữ gói chính thức của CentOS 7. Đầu tiên bân cần cập nhật yum cache bằng lệnh sau:

*sudo yum makecache*



- Tiếp theo các bạn cài đặt epel-release bằng lệnh sau:

*yum -y install epel-release*



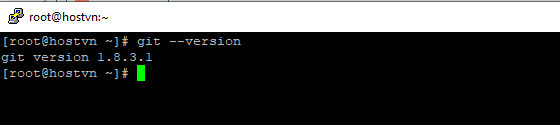
- Bây giờ hãy dùng lệnh dưới đây để cài đặt Git:

*sudo yum -y install git*



- Bạn có thể kiểm tra xem Git có hoạt động hay không với lệnh:

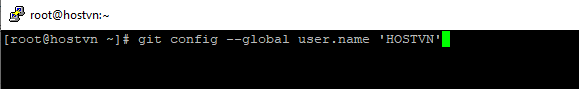
*git --version*



1.5.2. Cấu hình Git

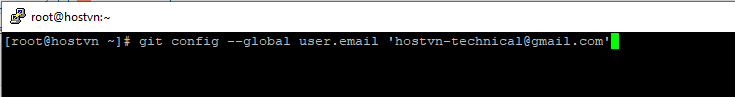
- Đầu tiên đặt User name của bạn bằng lệnh sau:

*git config --global user.name 'YOUR FULL NAME'*



- Tiếp theo các bạn sẽ cần cấu hình email (Lưu ý email này là email đăng ký tài khoản Github của các bạn):

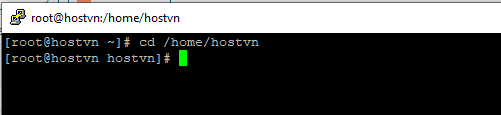
*git config --global user.email 'YOUR EMAIL'*



1.5.3. Khởi tạo Git Repository

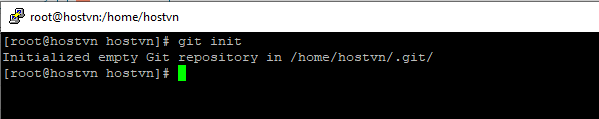
- Để sẵn sàng cho một dự án Git, trước tiên bạn phải khởi tạo nó. Đầu tiên hãy điều hướng vào thư mục dự án của bạn với lệnh sau:

*cd YOUR/PROJECT/DIRECTORY*



- Bây giờ hãy chạy lệnh sau để khởi tạo thư mục cho Git:

*git init*



# CHƯƠNG II: SVN

## 2.1. SVN là gì?

SVN (tên đầy đủ là Subversion), là một phần mềm mã nguồn mở dùng để quản lý và kiểm tra các phiên bản mã nguồn khác nhau trong quá trình phát triển phần mềm, được công ty CollabNet giới thiệu vào năm 2000. SVN hỗ trợ làm việc nhóm rất hiệu quả và được sử dụng phổ biến hiện nay. Hệ thống có khả năng tự cập nhật, so sánh và kết hợp tài nguyên mới vào tài nguyên cũ.

Subversion rất phổ biến trong giới mã nguồn mở và được dùng để quản lý mã nguồn cho nhiều ứng dụng Phần mềm miễn phí như: Apache Software Foundation, KDE, GNOME, Free Pascal, FreeBSD, GCC, Python, Django, Ruby, và Mono. SourceForge.net và Tigris.org, nơi chứa miễn phí chuyên nghiệp dành cho các ứng dụng tự do.

### **2.2.** **Thành phần Subversion**

### Subversion, sau khi cài đặt, có một số phần khác nhau. Sau đây là tổng quan của những gì bạn nhận được. Đừng lo lắng nếu phần mô tả làm cho bạn phải gãi đầu *rất nhiều* trang trong cuốn sách này được dành để làm giảm bớt sự rối rắm này.

Svn; Ứng dụng khách Subversion giao-diện-dòng-lệnh

Svnversion; Chương trình để báo cáo tình trạng (về các phiên bản của những mục hiện hữu) của một bản sao làm việc (working copy)

Svnlook; Công cụ để kiểm tra kho lưu trữ Subversion

Svnadmin; Công cụ để tạo, chỉnh, hoặc sửa chữa kho lưu trữ Subversion

mod\_dav\_svn; Thành phần plug-in dành cho Apache HTTP Server, được sử dụng để làm cho kho lưu trữ của bạn có sẵn cho những người khác trong một mạng lưới

Svnserver; Chương trình tuỳ biến độc lập chạy trên máy chủ, có thể được thực thi như là một tiến trình daemon hoặc kích hoạt thông qua SSH; một cách khác để làm cho kho lưu trữ của bạn có sẵn cho những người khác trong một mạng lưới

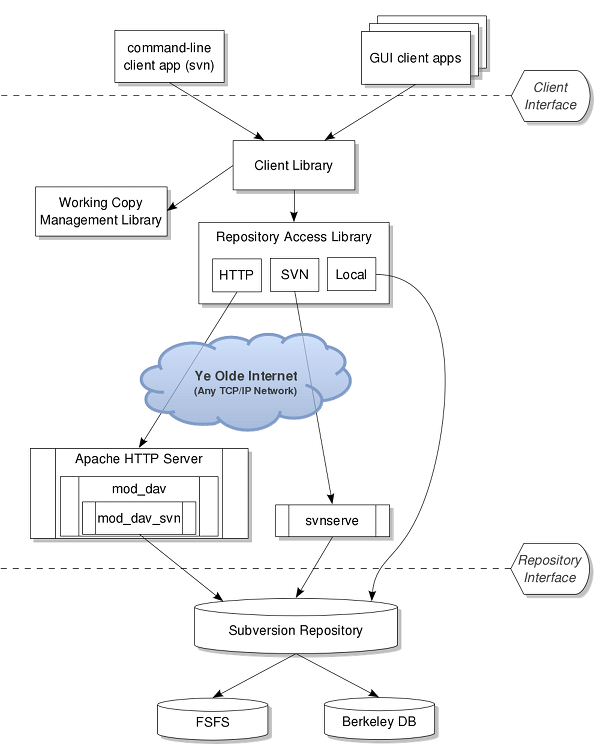
Svndumpfilter: Chương trình để lọc những luồng kết xuất của kho lưu trữ Subversion

Svnsync; Chương trình để đồng bộ từng bước một kho lưu trữ sang một kho khác trong một mạng lưới

Svnrdump; Chương trình để thực hiện việc kết xuất và nạp lịch sử của kho lưu trữ trong một mạng lưới

Svnmucc; Một chương trình để thực hiện nhiều thao tác kho lưu trữ dựa-trên-URL trong một lần chuyển giao (commit) và không cần bản sao làm việc (working copy)

## 2.3. Kiến trúc của SVN



Subversion gồm Repository và Client:

- **Repository** đóng vai trò làm server chứa dữ liệu, cung cấp giao thức cho phía client sử dụng giao thức tương tự.

- **Client** sẽ sử dụng giao diện GUI hay CLI tùy ý, thông qua thư viện Subversion để gửi và nhận dữ liệu.

Cấu trúc của Subversion thường được chia thành 3 folder chính, dĩ nhiên ta có thể thêm bớt tùy ý nhưng hầu hết các project lớn đều dùng cấu trúc này:

- **Thư mục trunk**: chứa code mới nhất theo đúng nghĩa đen, vì vậy đây thường là code dang dở, đang trong quá trình phát triển của dự án.

**- Thư mục tags**: thường chứa snapshot khi cần release cho dự án. Lấy ví dụ, Project A release tại version 1.0, toàn bộ code tại trunk sẽ được tạo thành tag 1.0 tại thời điểm đó, và như thế, khi cần build hoặc deploy hay review lại phiên bản 1.0 của project, ta chỉ cần lấy tag 1.0 về là xong, tương tự với các tag khác.

**- Thư mục branches**: như tên gọi, chứa các nhánh phát triển của dự án. Project A có thể song song phát triển tính năng X và tính năng Y mà không ảnh hưởng đến nhau, ta gọi đó là 2 branch của SVN, khi cần gộp lại cho official release, các branch có thể được gộp lại.

## 2.4. SVN hoạt động như thế nào?

Subversion ban đầu được thiết kế giống như giao diện dòng lệnh. Người dùng chỉ cần mở Terminal sau đó gõ các lệnh văn bản là được. Để SVN hoạt động, người dùng cần thiết lập nó dựa vào 2 yếu tố chính:

**Yếu tố 1:** Máy chủ có tất cả phiên bản của tệp nguồn

**Yếu tố 2:** Bản sao cục bộ của các tệp trên máy tính của người dùng

Các tệp trên máy tính người dùng được gọi là tệp tin làm việc. Đây là những tệp mà người dùng thực hiện việc chỉnh sửa và cam kết thay đổi với máy chủ SVN.

Mỗi khi người dùng thực hiện một thay đổi, SVN sẽ quản lý và ghi lại đồng thời tạo ra một phiên bản mới. Người dùng thường làm việc với phiên bản mới nhưng nếu cần đến phiên bản cũ họ vẫn có thể quay trở lại.

## 2.5. Những tính năng ưu điểm của SVN là gì?

Với SVN bạn sẽ được hưởng các tính năng ưu việt và cũng chính là những ưu điểm của hệ thống quản lý version - SVN như sau:

* Thứ nhất, Phiên bản được thực hiện cho các thư mục cũng bao gồm các tệp riêng lẻ.
* Thứ hai, có một mục tiêu chuẩn để sao chép, xóa, thêm và đổi tên các tệp tin trên các thư mục.
* Thứ ba, thuộc tính tệp và thư mục được đăng ký theo thời gian, theo đó hỗ trợ bạn xây dựng một lịch sử toàn diện.
* Thứ tư, giúp các nhà phát triển có thể thiết kế và dành các sửa đổi trong một phút, các đơn vị tự kiểm soát.
* Thứ năm, SVN đơn giản để thực hiện các chức năng mạng hiện tại.
* Thứ sáu, tệp nhị phân và văn bản được lưu và quản lý theo cách tương thích.
* Thứ bảy, các nhánh và thẻ được xây dựng hiệu quả và kịp thời.
* Thứ tám, SVN rất đơn giản để được sử dụng bởi các ngôn ngữ lập trình và thiết kế các phần mềm khác nhau.

## 2.6. Vai trò của SVN là gì?

SVN có khả năng thay thế hệ thống quản lý phiên bản CVS cũ (tồn tại nhiều nhược điểm). SVN có thể quản lý bất kỳ hệ thống phiên bản nào, được coi là hệ thống quản lý source code tập trung, mạnh mẽ, linh hoạt và hữu dụng.

SVN quản lý thư mục và tập tin theo thời gian. Nó giống như hệ thống file server mà máy khách có thể download file một cách bình thường. Đặc biệt, SVN có khả năng lưu lại tất cả những thay đổi trên hệ thống file (ai thay đổi, thay đổi lúc nào, thay đổi những gì).

SVN cho phép khôi phục lại những version cũ một cách nhanh chóng và chính xác. Do đó, làm việc nhóm cũng an toàn và hiệu quả hơn.

Máy chủ và máy khách có thể cùng chạy trên một SVN, có nhiệm vụ ghi lại lịch sử dự án.

SVN hỗ trợ rất tốt giao thức để kết nối giữa máy chủ và máy khách. Việc cho phép máy chủ hỗ trợ giao thức nào còn tùy thuộc vào cấu hình.

Trạng thái phát triển của 1 SVN, cùng với 1 số action đặc trưng của Version Control System.

1. Thư mục Trunks của SVN

2. Từ thư mục Trunks, project tạo ra Branch mới cho việc phát triển.

3. Làm việc với Branch mới.

4. Khi Branch mới đã hoàn thành feature, tiến hành gộp lại với branch chính, quá trình này gọi là Merge.

T1: tại đây, project đã stable và được tag lại thành T1.

5, 6: Tiếp tục tạo ra các branch mới cho feature độc lập.

8: Branch mới được phát triển.

10: Branch này bị dừng phát triển, development sẽ bị deactive từ đây và không còn ai làm việc trên branch này nữa.

6, 7: Branch mới được phát triển.

9: Branch mới hoàn thành feature, tiến hành Merge vào branch chính.

T2: project hoàn thành feature của giai đoạn tiếp theo, tạo ra tag T2 để lưu lại.

**Một số khái niệm chính trong SVN**

**Repository (Kho chứa):** Một repository là trái tim của bất cứ hệ thống quản lý phiên bản. Nó là khu vực trung tâm để các lập trình viên giữ tất cả công việc của họ. Kho lưu không chỉ lưu các file mà còn lưu giữ cả lịch sử làm việc. Kho lưu được truy cập thông qua mạng, thực hiện vai trò như một máy chủ và công cụ điều khiển phiên bản là một máy khách. Các máy khách có thể kết nối với repository và sau đó họ có thể giữ/lấy lại những thay đổi họ thực hiện tới/từ repository. Bằng cách giữ các thay đổi , một khách tạo các thay đổi này tới người khác và bằng cách lấy lại những thay đổi, một khách có thể nhận được các thay đổi được thực hiện bởi người khác như là một bản sao công việc.

**Trunk (Thân):** Một thân là một thư mục tại đó tất cả các phát triển chính xảy ra và thường được kiểm tra bởi các lập trình viên làm việc trên dự án.

**Tags (Thẻ):** Thư mục các thẻ được sử dụng để giữ các ảnh chụp nhanh (snapshot) được đặt tên của dự án. Hoạt động thẻ cho phép đặt các tên để mô tả và dễ ghi nhớ cho các phiên bản cụ thể trong repository.

**Branches (Nhánh):** Hoạt động nhánh được sử dụng để tạo một tiến trình khác của sự phát triển. Nó hữu ích khi bạn muốn tiến trình phát triển của bạn phân theo hai hướng khác nhau. Ví dụ, khi bạn công bố phiên bản 5.0, bạn có thể muốn tạo ra một nhánh để phát triển các tính năng phiên bản 6.0 mà được giữ riêng rẽ với phiên bản 5.0.

**HEAD**: đỉnh của cấu trúc SVN

**master**: thông thường khi tạo SVN, có một branch chính được tạo ra và gọi là master, các branch phụ thường được đặt theo tên feature mà branch đó được tạo. Với hình trên, đường nối các ô màu xanh lá thể hiện master, các đường nối ô màu vàng thể hiện branch được tạo ra

**change:** mô tả sự thay đổi cụ thể của 1 revision so với revision trước đó

**working copy:** Bản sao công việc là hình chụp nhanh (snapshot) của repository. Kho lưu được chia sẻ bởi cả team, nhưng người ta không thể sửa nó một cách trực tiếp. Thay vào đó mỗi nhà lập trình kiểm tra bản sao công việc. Bản sao công việc là khu vực làm việc riêng, tại đó các nhà lập trình có thể thực hiện công việc của họ riêng rẽ một mình, cách biệt với phần còn lại của team.

**conflict**: xung đột xảy ra khi có nhiều developer cùng làm việc với 1 working copy, VD A và B cùng checkout rev 40 của file config.js, sau đó A sửa function update() rồi commit lên rev 41, lúc này B cũng sửa function đó, khi B checkout về sẽ xảy ra conflict do SVN không biết được phiên bản của A hay B mới là latest

**resolve:** lúc này, B sẽ xem xét lại code trong function, giữ lại đoạn code của A hoặc sửa nó, đánh dấu conflict đã được resolve

**Một số action quan trọng khi sử dụng và triển khai SVN**

**commit:** khi developer tiến hành thay đổi code, architecture, structure, documentation,… và đưa nó lên server SVN, việc này được gọi là commit, các commit sẽ buộc phải kèm theo message mô tả đầy đủ sự thay đổi của lần commit đó, mỗi commit sẽ tăng số revision của SVN lên 1

**checkout**: user với credential truy cập vào SVN và lấy nội dung của SVN tại 1 revision nhất định về, thông thường có thể là revision mới nhất, revision này sẽ được gọi là HEAD – đỉnh của cấu trúc SVN

**revert:** sau khi tiến hành thay đổi, developer cảm thấy họ đã sai và muốn xóa đi làm lại từ đầu, việc revert là hành động để khôi phục trạng thái của 1 hay nhiều document về 1 revision nào đó, thường thì sẽ revert về revision hiện tại mà user đang làm việc

**merge:** việc gộp code từ nhiều branch khác lại làm một được gọi là Merge, branch thứ hai sẽ chuyển hướng về branch đầu tiên

**update:** việc update working copy sẽ chuyển working copy sang revision mà user lựa chọn

3. Cài đặt SVN server trên CentOS 7

*yum install httpd subversion mod\_dav\_svn*

*vi /etc/httpd/conf.modules.d/10-subversion.conf*

*<Location /svn>*

*DAV svn*

*SVNParentPath /var/www/svn*

*AuthType Basic*

*AuthName "SVN Repositoty"*

*AuthUserFile /etc/svn-auth-accounts*

*Require valid-user*

*</Location>*

*htpasswd -cm /etc/svn-auth-accounts admin*

*mkdir /var/www/svn*

*cd /var/www/svn*

*svnadmin create repo*

*chown apache.apache repo/*

*chcon -R -t httpd\_sys\_content\_t /var/www/svn/repo*

*chcon -R -t httpd\_sys\_rw\_content\_t /var/www/svn/repo*

*systemctl status firewalld*

*firewall-cmd --permanent --zone=public --add-service=http*

*firewall-cmd --permanent --zone=public --add-service=https*

*firewall-cmd --reload*

*systemctl restart httpd*

*systemctl enable httpd*

*chown -R apache:apache /var/www/svn/repo*

*sudo systemctl restart httpd.service*

**Chương III. CentOS**

1. CentOS là gì?

CentOS là một hệ điều hành miễn phí được xây dựng và phát triển dựa trên hệ điều hành mã nguồn mở Linux. CentOS là chữ viết tắt của “Community Enterprise Operating System”. CentOS ra mắt công chúng vào tháng 5 năm 2004 và được phát triển dựa trên bản phân phối của Red Hat Enterprise Linux (RHEL).

2. Ưu điểm của CentOS

Hệ điều hành CentOS hiện đang thuộc top những bản phân phối Linux rộng rãi nhất hiện nay trên thế giới. Nó được giới IT đánh giá rất cao nhờ những ưu điểm sau đây:

- CentOS có rất nhiều tính năng hỗ trợ được phát triển và xây dựng từ chính cộng đồng đã tạo ra nó.

- Do có tính tương đồng với RHEL nên CentOS là một môi trường rất tốt để lập trình.

- CentOS có khả năng vận hành tốt, mượt mà trên các mainframe, đặc biệt là GUI, KDE, GNOME,…

- CentOS tạo ra một môi trường giống như trên desktop, rất đơn giản . Bạn có thể sử dụng, tùy chỉnh dễ dàng với sự hỗ trợ của cộng đồng mạng và Red Hat.

- Môi trường mà CentOS tạo ra có tính ổn định cao nên rất được ưa chuộng bởi các doanh nghiệp.

- CentOS được Red Hat cung cấp nhiều mã nguồn và trình quản lý gói RPM.

- CentOS có nhiều cấp độ bảo mật khác nhau, vì vậy nó là sự lựa chọn an toàn nhất để bảo mật thông tin kinh doanh cho doanh nghiệp.

3. Các thông tin cơ bản về CentOS

Hình thức cập nhật hệ thống CentOS: Thông qua câu lệnh Yum.

Mã nguồn mà CentOS sử dụng là mã nguồn mở tương tự Red Hat. CentOS có thể tương thích hoàn toàn với các phần mềm chạy trên Red Hat. Đặc biệt là với các phiên bản CentOS 5.0 trở về sau.

Người dùng CentOS có thể nhờ đến sự hỗ trợ kỹ thuật của cộng đồng lập trình thông qua các kênh social như diễn đàn, chat room hoặc list chính thức,…

Tuy nhiên, hệ điều hành CentOS chỉ hỗ trợ các kiến trúc x86 (kiến trúc tập lệnh được xây dựng dựa trên bộ vi xử lý 8086 của Intel). Cụ thể là:

- Kiến trúc tập lệnh x86 32 bit

- Kiến trúc tập lệnh x86-64 (hay còn gọi là x64, AMD64 hoặc Intel64)

4. Lịch sử hình thành và phát triển của CentOS

Nhắc đến lịch sử hình thành và phát triển của hệ điều hành CentOS, chúng ta nhất định phải biết về Tao Linux. Tao Linux là một bản sao của hệ điều hành RHEL.

- 06/2006, David Parsley – cha đẻ của Tao Linux tuyên bố sẽ dừng dự án này vô thời hạn. Trong lúc đó, sau 2 năm xây dựng, CentOS lại đang trên đà phát triển. Hầu hết người dùng Tao Linux đều quyết định chuyển sang CentOS bằng câu lệnh “Yum”.

- 07/2009, CentOS đột ngột thông báo người sáng lập của hệ điều hành này – Lance Davis đã mất tích trong suốt năm 2008 trong khi vẫn đang giữ tên miền đăng ký của CentOS và tài khoản Paypal, khiến hoạt động của CentOS tạm thời bị gián đoạn.

- 08/2009, nhóm CentOS liên lạc thành công với Lance Davis và nhận lại được tên miền centos.info, centos.org.

- 07/2010, hệ điều hành CentOS chính thức trở lại “đường đua”. CentOS sau đó trở thành bản phân phối Linux phổ biến nhất. Nó chiếm khoảng 30% trong tổng số tất cả các server Linux trên thế giới.

- 01/2014, Red Hat đưa ra tuyên bố rằng sẽ tài trợ cho CentOS để tạo ra môi trường lập trình tốt nhất cho các nhà phát triển mã code. Kể từ đó, Red Hat chính thức trở thành đơn vị sở hữu CentOS.

5. Các phiên bản của CentOS

Các phiên bản của hệ điều hành CentOS được đánh số và chia làm 2 phần. Một phần là phiên bản chính và một phần là phiên bản nhỏ, tương ứng với phiên bản và cập nhật của RHEL. Ví dụ, CentOS 4.4 được tạo ra từ RHEL 4, cập nhật 4.

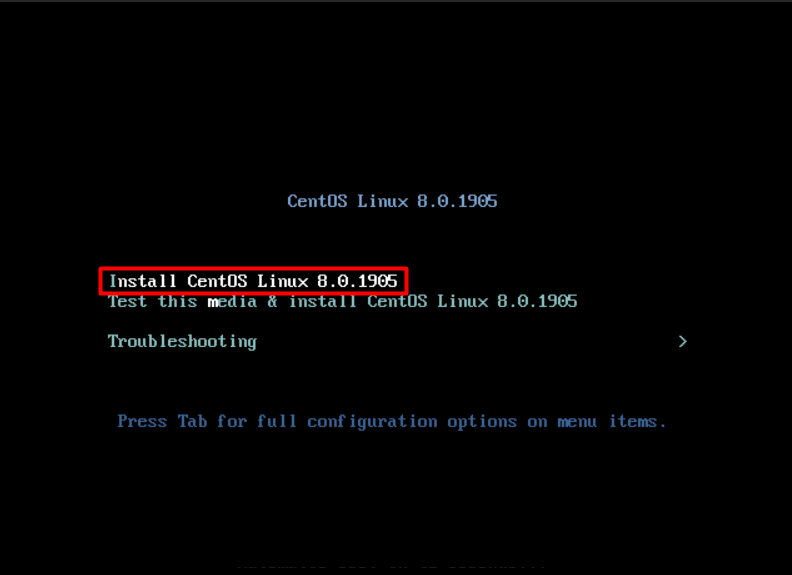
Bắt đầu từ CentOS 7.0 trở đi, các phiên bản cập nhật sẽ có thêm 1 dãy số cho biết ngày tháng phát hành. Ví dụ, phiên bản CentOS 7.0-1406 được phát hành tháng 6 năm 2014. Phiên bản mới nhất của CentOS hiện nay là CentOS 7.7, được phát hành ngày 06/08/2019.

6. Cách cài đặt hệ điều hành CentOS

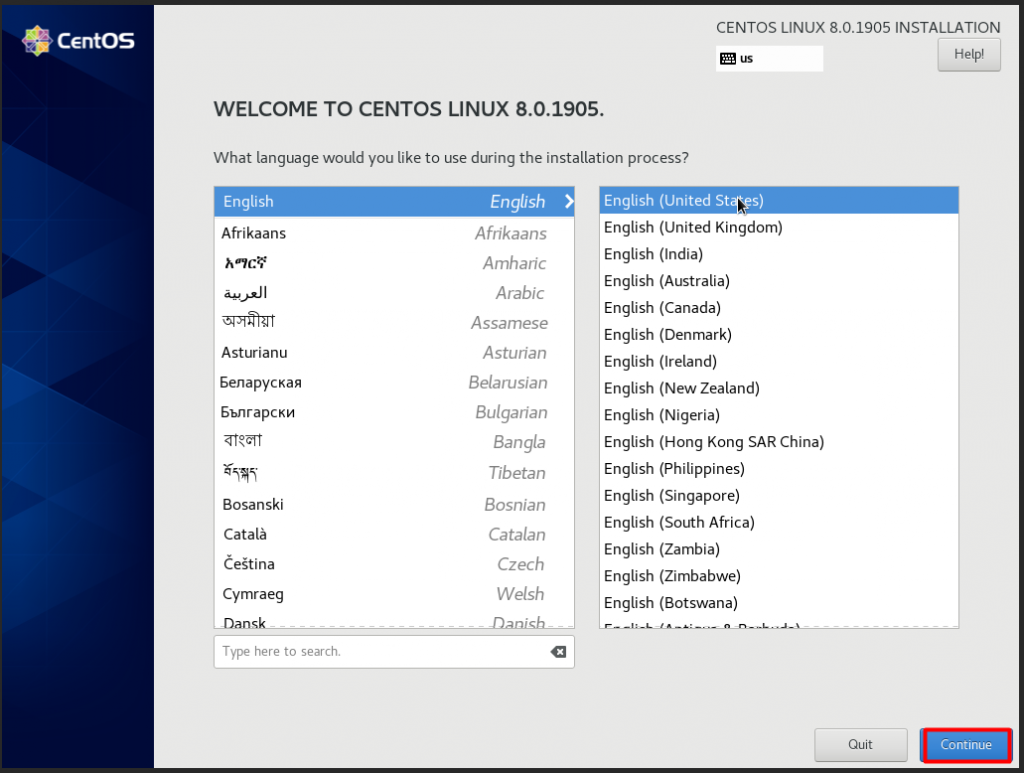
6.1. Cài đặt

- Tạo máy ảo hoặc đưa đĩa vào ổ đĩa của máy cần cài. Tùy thuộc vào nhu cầu bạn cài CentOS cho máy chủ ảo hay máy tính cá nhân.

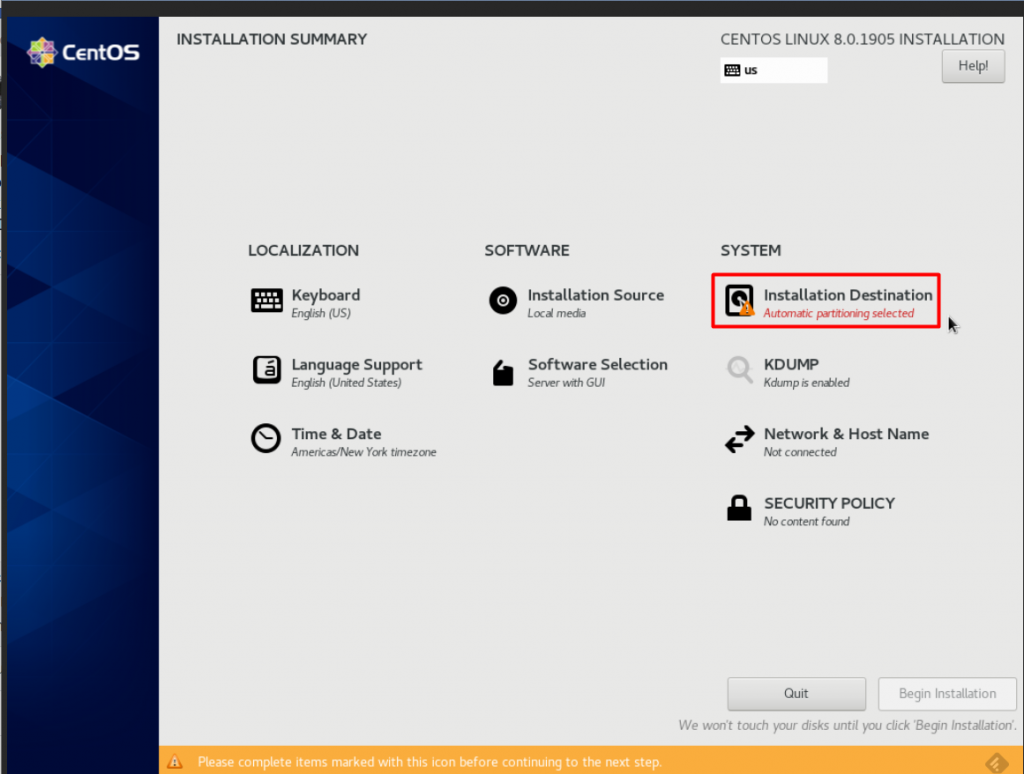
- Khởi động máy và chọn boot vào CentOS. Ấn Enter



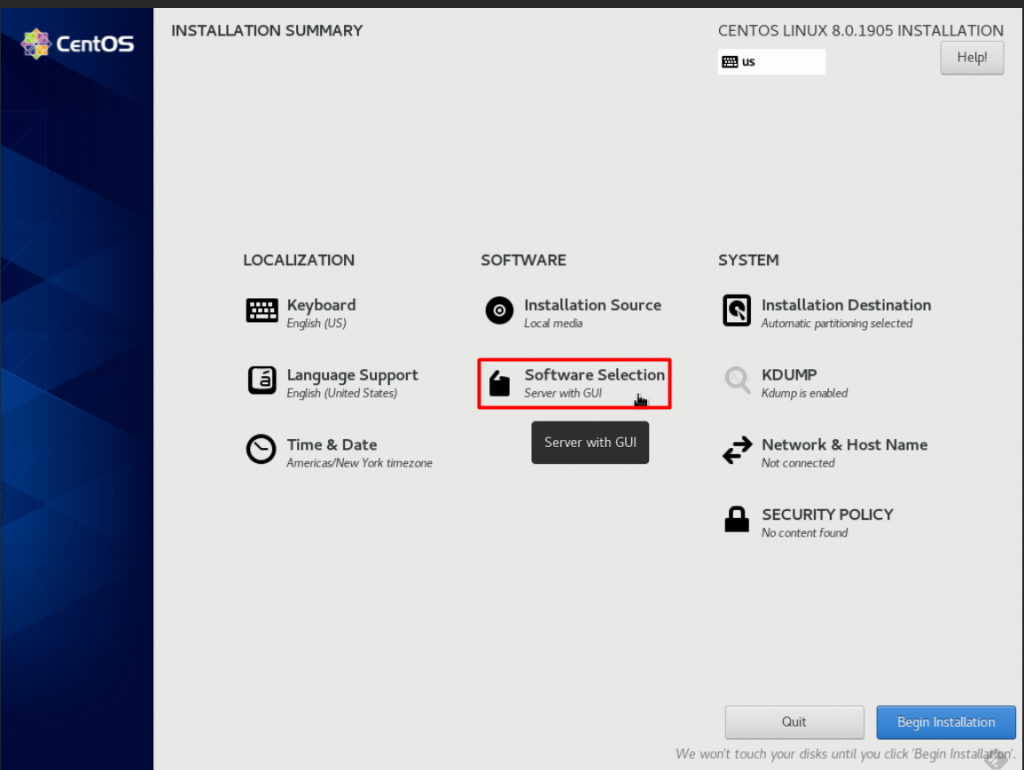
- Chọn ngôn ngữ cho hệ điều hành



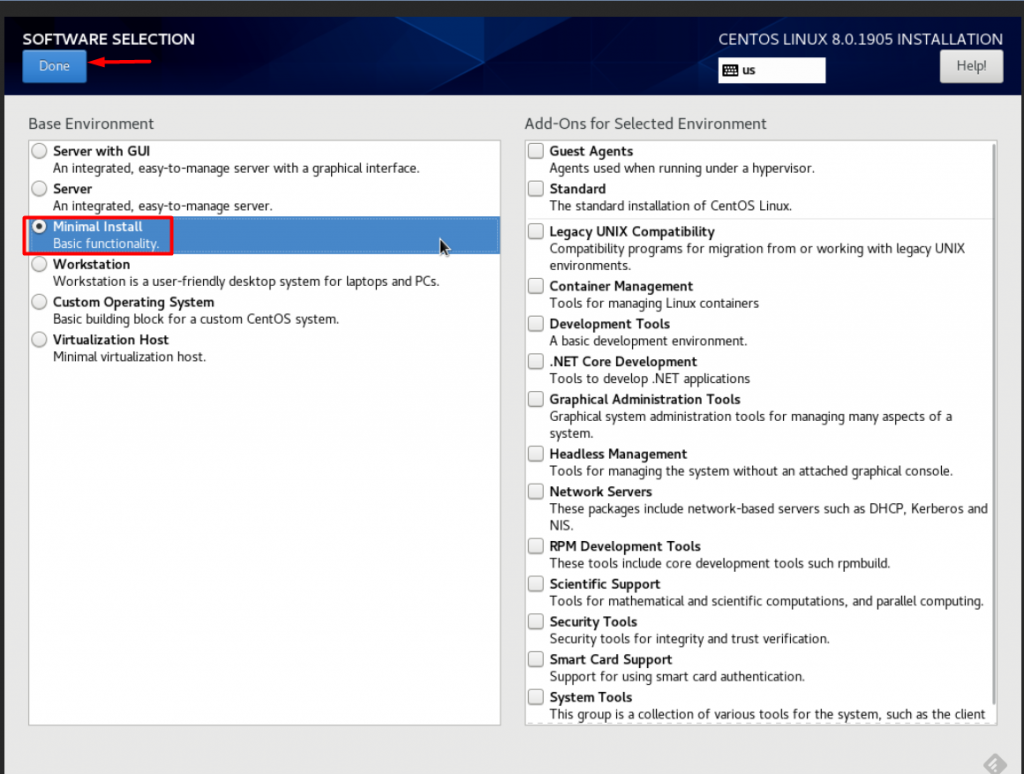
- Chọn ổ cứng để cài CentOS trong Installation Destination.



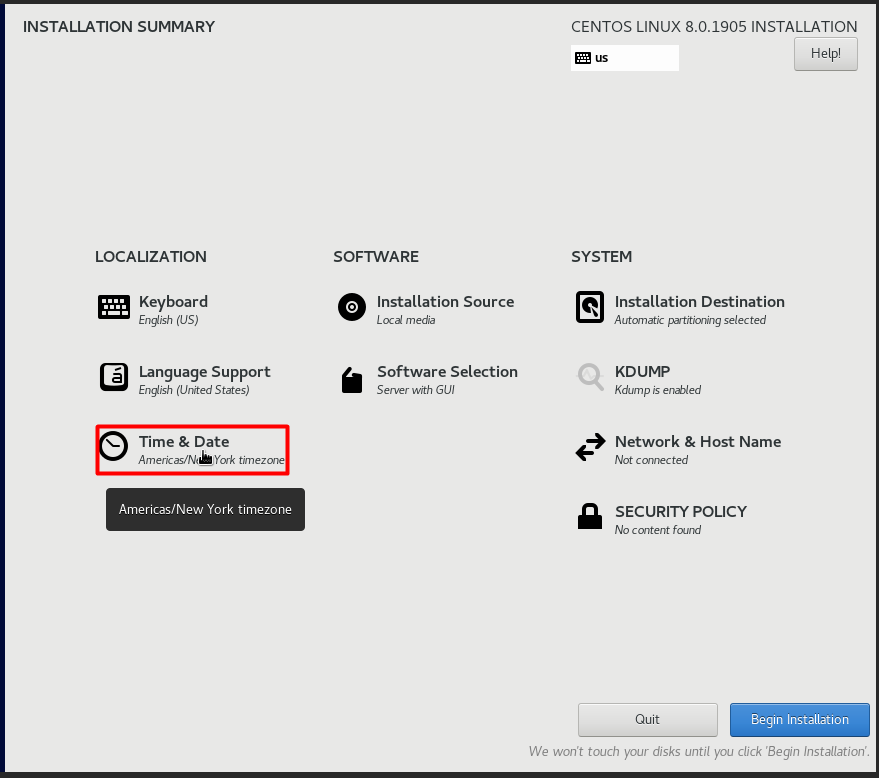
- Sau khi chọn Done, ta sẽ quay về giao diện tiếp và chọn mục Software Selection để lựa chọn chế độ không dùng GUI.



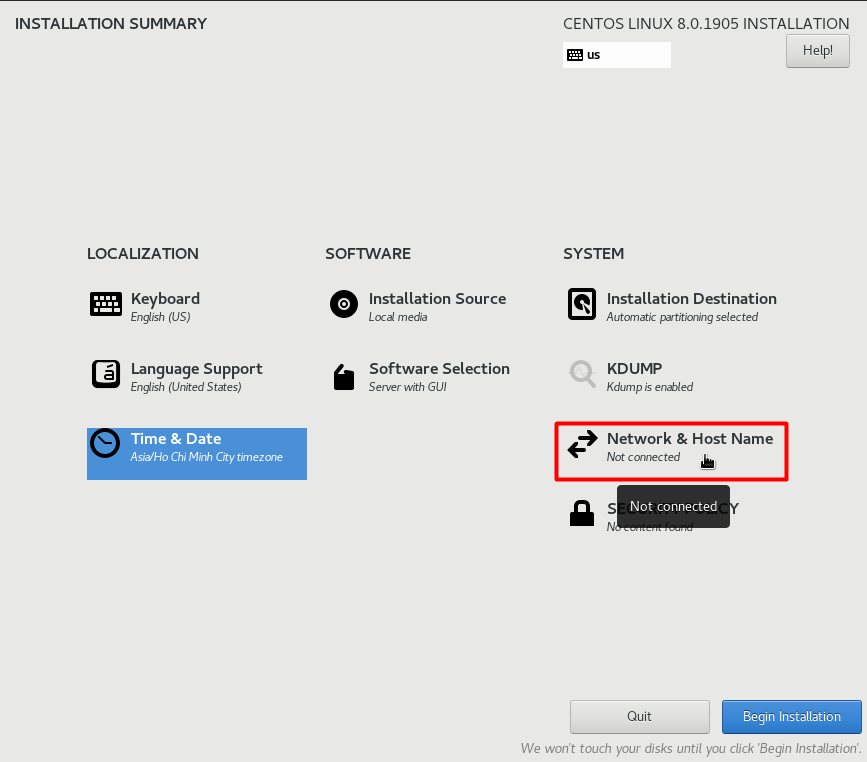
- Chọn chế độ Minimal Install để không sử dụng GUI cho hệ điều hành sau này. Sau đó chọn Done.



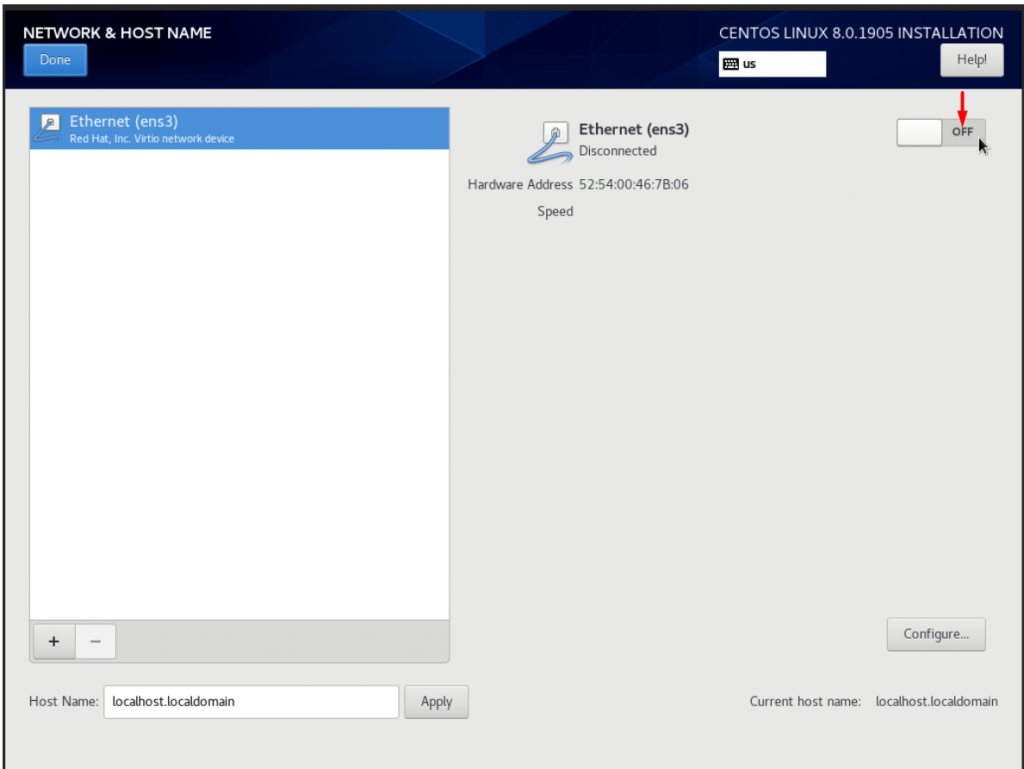
- Nên chọn timezone chuẩn với giờ Việt Nam để dễ sử dụng hơn.



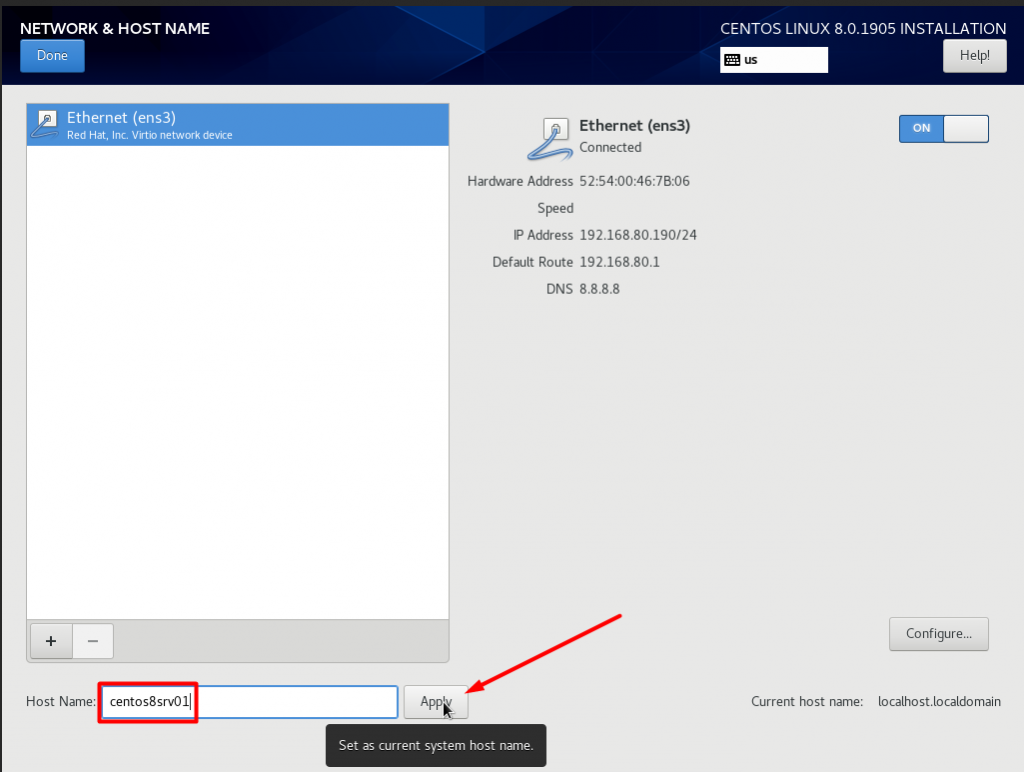
- Chọn Network & Host Name để cấu hình IP và hostname.



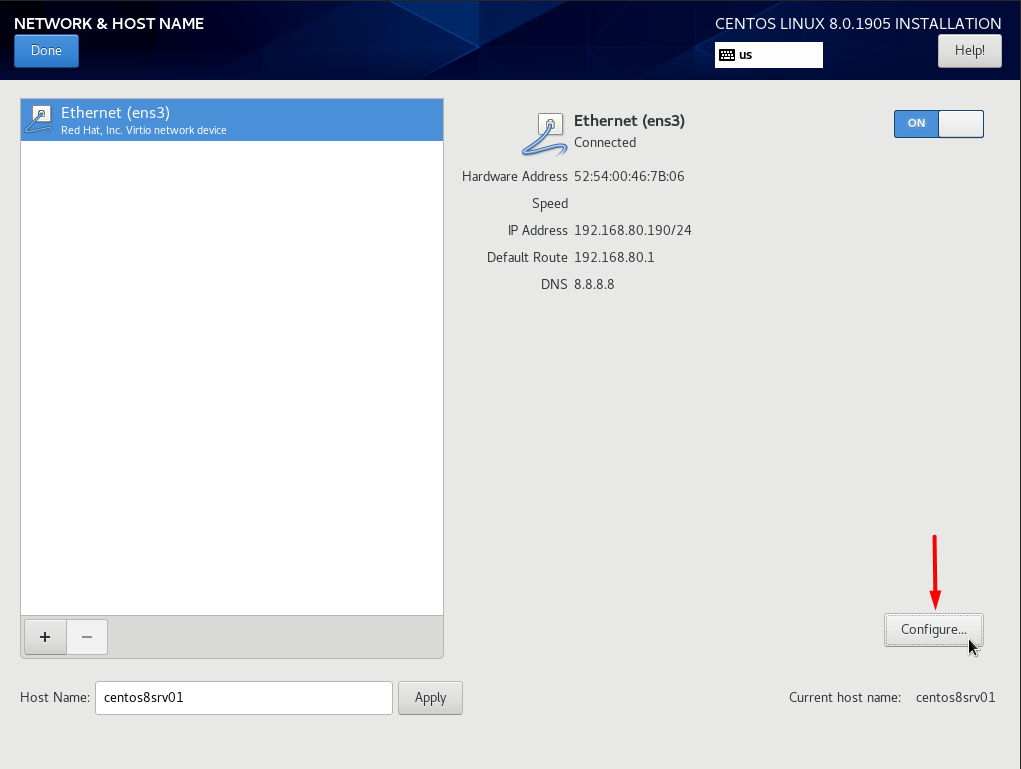
- Ta sẽ có giao diện chọn hostname và IP bên dưới. Ta click vào nút OFF để máy có kết nối về network.



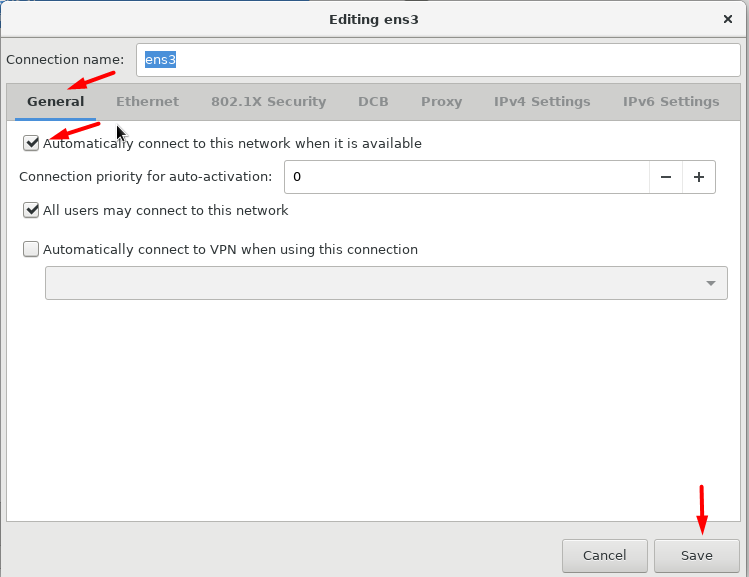
- Sau đó nhập hostame tùy thích và chọn Apply.



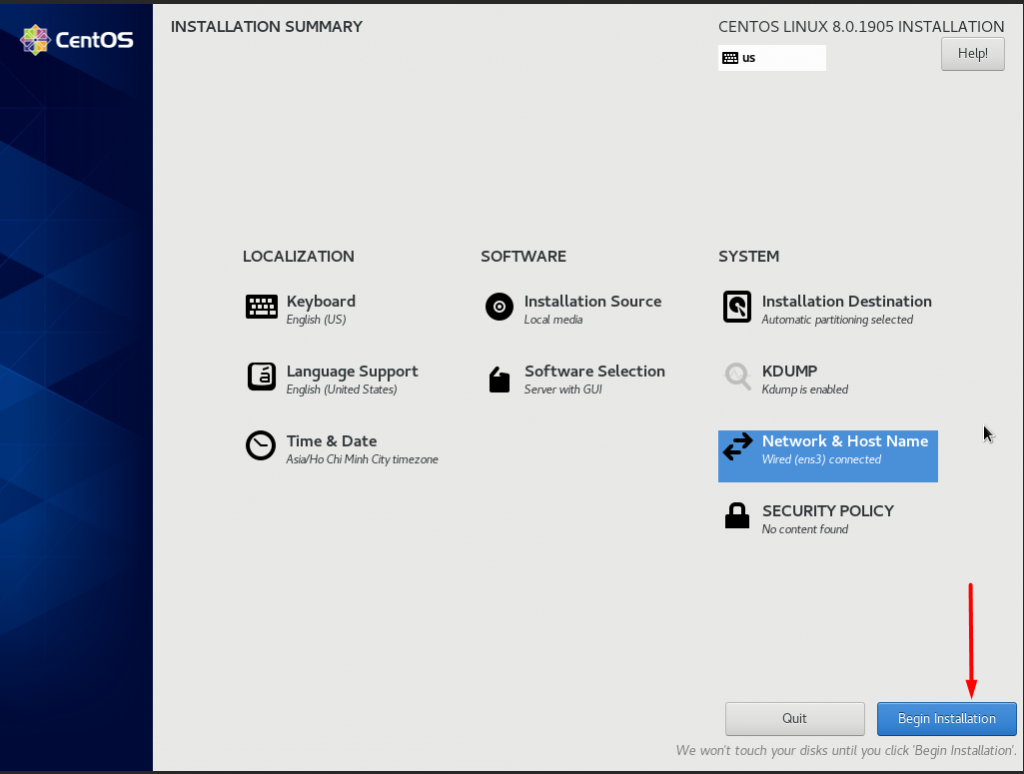
- Chọn mục Configure để cấu hình card mạng ở chế độ start cùng OS.



- Chọn tab Genaral và tích vào nút Automaticaly Connect to this network when it is avaible



- Sau khi về menu chính, ta chọn Begin Instalation để thực hiện cài đặt.



Sau đó nhập mật khẩu cho tài khoản root. Lưu ý ghi nhớ mật khẩu này để sử dụng sau này.

Chọn nút Reboot để máy khởi động lại. Bắt đầu đăng nhập vào OS với tài khoản root và mật khẩu nhập trước đó.

6.2. Cấu hình CentOS 7

- Khởi tạo một user mới:

*useradd ten-user*

- Tạo một nhóm người dùng (Group) mới:

*groupadd ten-group*

**-** Đổi mật khẩu cho user:

*passwd ten-user*

- Xóa user

*userdel ten-user*

- Xóa Group

*groupdel ten-group*

- Hiển thị danh sách user

*vi /etc/passwd*

- Hiển thị danh sách Group

*vi /etc/group*

- Thêm một user vào một group

*usermod -g ten-group ten-user*

*-* Chuyển đổi giữa các user

*su ten-user*

**Chương IV. SSH**

1. SSH là gì?

SSH là viết tắt của Secure Shell. Là một giao thức mạng dùng để thiết lập kết nối giữa Client và Server. Thường được sử dụng để kết nối, làm việc từ xa thông qua internet.

Thông thường các server được cung cấp bởi AWS hay Google đề được cài đặt sẵn SSH giúp cho người dùng dễ dàng kết nối. Đối với các nhà cung cấp VPS thì họ cũng sẽ cài đặt sẵn. Hay cụ thể hơn là khi cài đặt Ubuntu, Debian từ file ISO sẽ có bước chọn các phần mềm cài đặt cùng với OS. Ví dụ như giao diện Gnome, SSH Server, Web server,…

SSH là phần mềm cực kì hữu ích khi làm việc với server Linux. Có thể nói SSH là phần mềm phải cài đặt ngay sau khi cài OS.

1. Cài đặt SSH trên CentOS 7

- Update server CentOS 7:

*yum update -y*

- Cài đặt SSH Server bằng lệnh sau:

*yum install -y openssh openssh-server openssh-clients openssl-libs*

- Khởi động dịch vụ SSH để kết nối

*systemctl start sshd*

- Mở firewalld port 22 để bên ngoài có thể kết nối đến server

*firewall-cmd --zone=public --add-port=22/tcp –permanent*

*firewall-cmd –reload*

1. Đổi port SSH trên CentOS 7

- Update các bản vá cho hệ thống

*[root@localhost ~]# yum update -y​*

- Cấu hình SSH đổi port mong muốn

*[root@localhost ~]# vi /etc/ssh/sshd\_config*

Mặc định là port 22 bạn sửa thành port mong muốn:

VD: ở đây muốn mở port 2220 cho dịch vụ SSH



- Lưu lại cấu hình và khởi động lại dịch vụ SSH để nhận port mới.

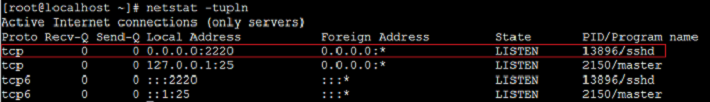
*[root@localhost ~]# systemctl restart sshd.service*

- Bạn có thể kiểm tra lại dịch vụ SSH đã nhận chính xác port mới chưa, bằng công cụ netstat. Nếu chưa có Netstat bạn có thể cài đặt bằng lệnh:

*[root@localhost ~]# yum install net-tools -y*

- Sau đó kiểm tra các cổng đang mở.

*[root@localhost ~]# netstat -tupln*



- Trường hợp CentOS của bạn có đang chạy SElinux bạn sẽ cần phải. Hoặc tắt SELinux, hoặc nếu vẫn muốn chạy SELinux đồng thời thì bạn cần cho phép các port mới được sử dụng cho ssh. Trước tiên cài đặt Semanage:

*[root@localhost ~]# yum install policycoreutils-python​*

- Để xem các cổng SELinux đang được sử dụng cho chúng ta có thể chạy các lệnh tùy chọn:

*[root@localhost ~]# semanage -l | grep ssh​*

Trường hợp bạn thấy cổng 22 được liệt kê thì để thêm một cổng mới gõ lệnh sau:

Đầu tiên, xóa port 22:

*[root@localhost ~]# semanage port -d -t ssh\_port\_t -p tcp 22*

- Và mở port 2220 cho dịch vụ SSH trên SELinux:

*[root@localhost ~]# semanage port -a -t ssh\_port\_t -p tcp 2220*

- Bây giờ cho phép kiểm tra xem CentOS của bạn đang lắng nghe dịch vụ SSH trên cổng mới. Gõ lệnh lệnh netstat để kiểm tra:

*[root@localhost ~]# netstat -tupln*

- Cuối cùng mở port mới cho dịch vụ SSH trên Firewall

*[root@localhost ~]# firewall-cmd --permanent --add-port=2220/tcp*

- Khởi động lại Firewall để có hiệu lực

*[root@localhost ~]# firewall-cmd --reload*

**Chương V. FTP**

1. Vsftpd

vsftpd là một FTP Server Stand Alone được phân phối bởi Red Hat Enterprise Linux. Mô hình vsftpd có 3 thuộc tính chính là:

- Sự phân chia rõ rệt giữa tiến trình mang đặc quyền và tiến trình không mang đặc quyền.

- Các task cần các đặc quyền được xử lý bởi các tiến trình có ít các đặc quyền nhất

- Hầu hết các tiến trình được chạy trong chroot jail để nâng cao tính bảo mật.

2. Cài đặt vsftpd trên CentOS 7

- Cài đặt gói vsftpd:

*# yum install vsftpd*

- Copy file cấu hình để backups: **/etc/vsftpd/vsftpd.conf**

*# cp /etc/vsftpd/vsftpd.conf /etc/vsftpd/vsftpd.conf.example*

- Cấu hình FTP server, chỉnh sửa file cấu hình

*# vi /etc/vsftpd/vsftpd.conf*

*# không cho user nặc danh truy cập ftp server*

*anonymous\_enable=NO*

*# sử dụng local user*

*local\_enable=YES*

*# cho phép local user upload, delete file*

*write\_enable=YES*

*# chroot và tạo 1 file /etc/vsftpd/chroot\_list chứa danh sách các local user không bị giới hạn bởi chroot*

*chroot\_local\_user=YES*

*allow\_writeable\_chroot=YES*

*chroot\_list\_enable=YES*

*chroot\_list\_file=/etc/vsftpd/chroot\_list*

*# Ta chỉ sử dụng IPv4*

*listen=YES*

*# Không sử dụng IPv6*

*listen\_ipv6=NO*

*# Chỉ cho các user có trong file /etc/vsftpd/user\_list được truy cập FTP server*

*userlist\_deny=NO*

*userlist\_enable=YES*

*userlist\_file=/etc/vsftpd/user\_list*

*# Thư mục home của FTP Server*

*local\_root=/var/ftp/*

*# Sử dụng thời gian của hệ thống*

*use\_localtime=YES*

- Khởi động vsftp:

*# systemctl start vsftpd*

*# systemctl enable vsftpd*

- Cấu hình tường lửa:

*# firewall-cmd --permanent --add-port=21/tcp*

*# firewall-cmd --permanent --add-service=ftp*

*# firewall-cmd –reload*

- SELinux:

*# setsebool -P ftpd\_full\_access on*

3. Tạo user và phân quyền

3.1. Tạo các user và group

- Tạo 3 user: user1, user2, user3 và thay đổi Home directory của chúng như sau:

*# useradd -d /var/ftp/user1 user1*

*# useradd -d /var/ftp/user2 user2*

*# useradd -d /var/ftp/user3 user3*

- Đặt mật khẩu cho các user vừa tạo. Ở đây, mình đặt mật khẩu của 3 user giống nhau là 123456:

*# passwd user1*

*# passwd user2*

*# passwd user3*

- Tạo 2 group:

* **ftp\_basic**: user có quyền bình thường đối với thư mục của mình và thư mục dùng chung. Dùng cho user1 và user2.
* **ftp\_onlyread**: user chỉ có quyền đọc. Dùng cho user3.

*# groupadd ftp\_basic*

*# groupadd ftp\_onlyread*

- Thêm các user vào 2 group vừa tạo:

*# usermod -g ftp\_basic user1*

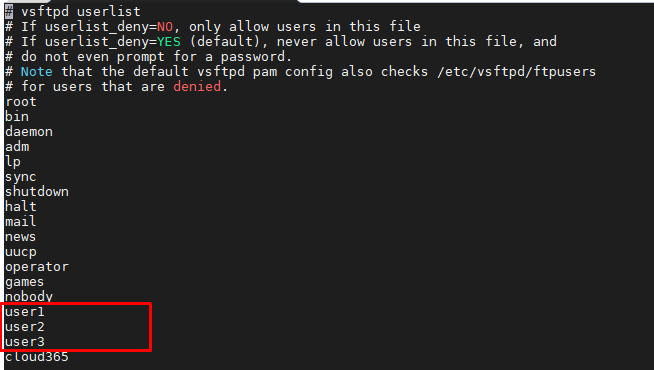
*# usermod -g ftp\_basic user2*

*# usermod -g ftp\_onlyread user3*

3.2. Cấp quyền truy cập và phân quyền các user:

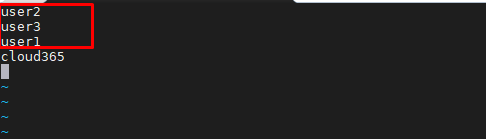
- Cấp quyền truy cập FTP server cho các user vừa tạo bằng cách thêm vào file **/etc/vsftpd/user\_list**, mỗi user trên một dòng và lưu lại.

*# vi /etc/vsftpd/user\_list*



- Thêm vào danh sách không bị chroot giới hạn

*# vi /etc/vsftpd/chroot\_list*



- Tạo 1 thư mục dùng chung cho user1 và user2 tên là **/user12**:

*# mkdir /var/ftp/user12*

- Thay đổi quyền sở hữu thư mục:

*# chown -R user1:ftp\_basic /var/ftp/user1*

*# chown -R user2:ftp\_basic /var/ftp/user2*

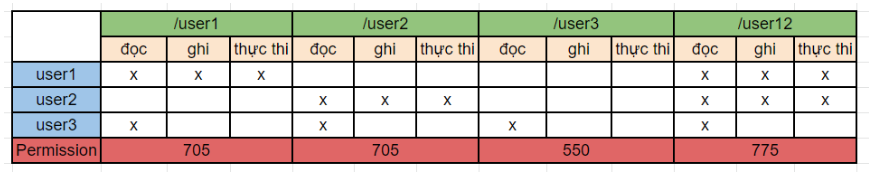
*# chown -R user3:ftp\_onlyread /var/ftp/user3*

*# chown -R :ftp\_basic /var/ftp/user12*

- Thay đổi permission để kiểm soát truy cập của từng user đối với các thư mục



Quyền và permission của các user với các thư mục:



*# chmod 705 /var/ftp/user1*

*# chmod 705 /var/ftp/user2*

*# chmod 550 /var/ftp/user3*

*# chmod 775 /var/ftp/user12*

- Tiến hành restart lại dịch vụ vsftpd:

*# systemctl restart vsftpd*

4. Đổi port vsftpd

- Truy cập file config

*vi /etc/vsftpd/vsftpd.conf*

- Đổi port ftp

*# vi /etc/vsftpd/vsftpd.conf*

*listen\_port=2020*

*ftp\_data\_port=2121*