

**Môn: CÔNG CỤ PHÁT TRIỂN PHẦN MỀM**

**Đề tài : Tìm hiểu về Git, SVN, SSH, FTP**

**Trường đại học công nghệ GTVT**

**Khoa công nghệ thông tin**

GTVT GTVT



GVHD: Nguyễn Văn Cường

Nhóm: Nhóm 1

Thành viên: Phạm Tiến Thành

Nguyễn Thu Thảo

Nguyễn Thị Nhung

Dương Thị Huệ

**Hà Nội**

Contents

[**LỜI NÓI ĐẦU** 3](#_Toc119448666)

[**Chương 1: Hệ thống quản lí mã nguồn GIT** 4](#_Toc119448667)

[1.1. Giới thiệu tổng quan về GIT 4](#_Toc119448668)

[1.1.1. Khái niệm GIT 4](#_Toc119448669)

[**1.1.2. Sơ lược lịch sử phát triển** 5](#_Toc119448670)

[1.1.3. Lợi ích của Git 6](#_Toc119448671)

[1.2. Kiến thức cơ bản về GIT 6](#_Toc119448672)

[1.2.1. Cài đặt GIT 6](#_Toc119448673)

[1.2.2. Một số lưu ý khi làm việc với Git 17](#_Toc119448675)

[1.2.3. Một số giao diện GUI của GIT 17](#_Toc119448676)

[**Chương 2: Hệ thống quản lí mã nguồn SVN** 22](#_Toc119448677)

[2.1. Giới thiệu tổng quan về SVN 22](#_Toc119448678)

[2.1.1. Khái niệm SVN 22](#_Toc119448679)

[2.1.2 Các thành phần của SVN 23](#_Toc119448680)

[2.2. Kiến thức cơ bản về SVN 24](#_Toc119448681)

[2.2.1. Cài đặt SVN 24](#_Toc119448682)

[2.2.2. Bắt đầu với Tortoise SVN 30](#_Toc119448684)

[2.2.3 So sánh SVN với Git 40](#_Toc119448685)

[**Chương 3: Hệ thống quản lí mã nguồn SSH** 42](#_Toc119448686)

[3.1. Giới thiệu tổng quan về SSH 42](#_Toc119448687)

[3.1.1. Khái niệm SSH 42](#_Toc119448688)

[3.1.2. Cơ chế hoạt động 42](#_Toc119448692)

[3.1.3 Chức năng chính 43](#_Toc119448699)

[3.2. Kiến thức cơ bản về SSH 44](#_Toc119448706)

[3.2.1. Cài đặt SSH 44](#_Toc119448707)

# **LỜI NÓI ĐẦU**

Git là hệ thống **điều khiển phiên bản phân tán** hiện đại và được sử dụng rộng rãi trên thế giới. Nó được phát triển để quản lý các dự án với tốc độ và hiệu quả cao. Hệ thống kiểm soát phiên bản cho phép chúng ta giám sát và làm việc cùng với các thành viên trong nhóm của chúng ta tại cùng một không gian làm việc.

Hướng dẫn này sẽ giúp bạn hiểu hệ thống điều khiển phiên bản phân tán Git thông qua dòng lệnh cũng như với GitHub. Các ví dụ trong hướng dẫn này được thực hiện trên **Windows** , nhưng chúng ta cũng có thể thực hiện các thao tác tương tự trên các hệ điều hành khác như **Linux (Ubuntu)** và **MacOS**.

# **Chương 1: Hệ thống quản lí mã nguồn GIT**

## 1.1. Giới thiệu tổng quan về GIT

### 1.1.1. Khái niệm GIT

**Git** là một **hệ thống kiểm soát phiên bản phân tán code nguồn mở**. Nó được thiết kế để xử lý các dự án từ nhỏ đến lớn với tốc độ và hiệu quả cao. Nó được phát triển để điều phối công việc giữa các developer. Kiểm soát phiên bản cho phép chúng ta theo dõi và làm việc cùng với các thành viên trong nhóm của chúng ta tại cùng một không gian làm việc..

Git là nền tảng của nhiều dịch vụ như **GitHub** và **GitLab** , nhưng chúng ta có thể sử dụng Git mà không cần sử dụng bất kỳ dịch vụ Git nào khác. Git có thể được sử dụng **riêng tư** và **công khai** .

Git được **Linus Torvalds** tạo ra vào **năm 2005** để phát triển Nhân Linux. Nó cũng được sử dụng như một công cụ kiểm soát phiên bản phân tán quan trọng cho [**DevOps**](https://cafedev.vn/mien-phi-100-series-tu-hoc-devops-tu-co-ban-toi-nang-cao/) .

Git rất dễ học và có hiệu suất cao trong công việc. Nó vượt trội hơn các công cụ SCM khác như Subversion, CVS, Perforce và ClearCase.

\* Ba trạng thái với Git

Với Git, các file của bạn có thể nằm ở một trong ba trạng thái: committed, modified, staged

- **committed** có nghĩa là dữ liệu đã lưu trữ an toàn trong Database (local)

- **modified** có nghĩa là dữ liệu (file) có thay đổi nhưng chưa lưu trong Database (local)

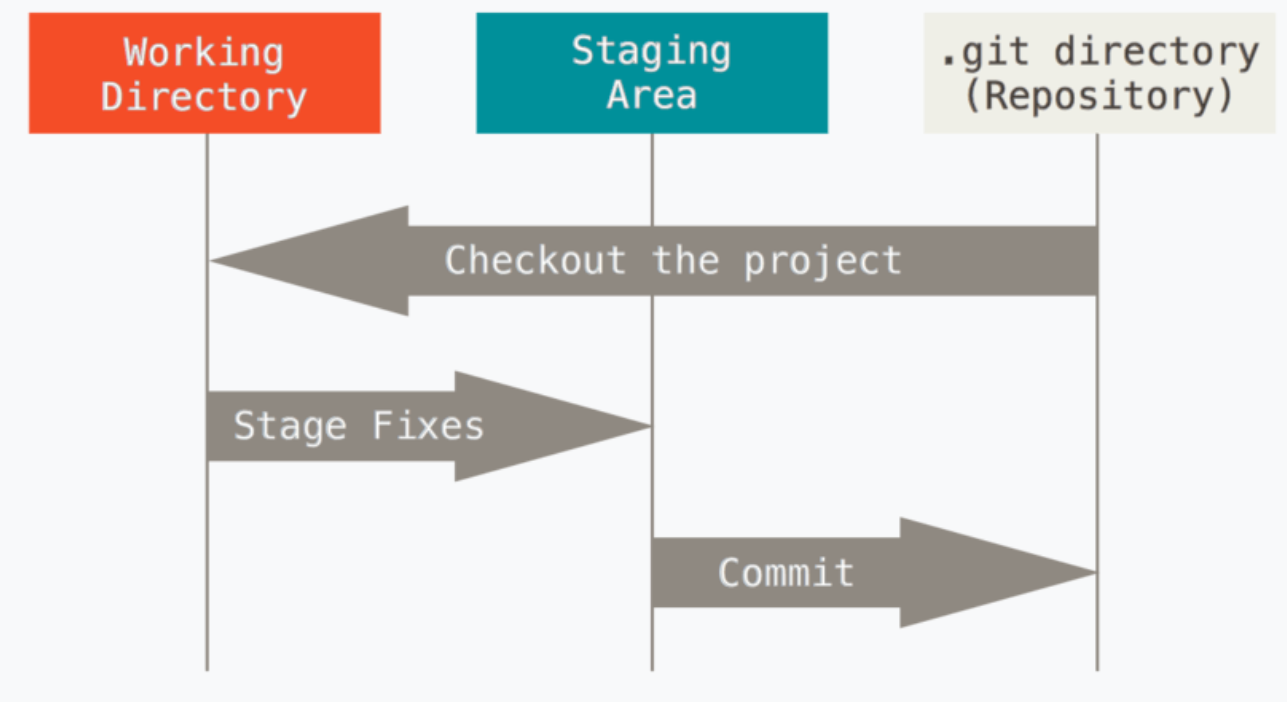
- **staged** có nghĩa đánh dấu các file sửa đổi modified chuẩn bị được commit tiếp theo

\* Các khái niệm trên dẫn tới Git tổ chức một dự án ra ba khu vực:

- Thư mục làm việc (Working tree) - Chứa bản sao phiên bản cụ thể của dự án

- Khu vực sắp xếp (staging) - là một file, nằm trong thư mục Git chứa thông tin sẽ commit

- Thư mục Git (Git directory) - nơi Git lưu Database các file Metadata



- Từ hình trên, khi làm việc với Git bạn theo tiến trình như sau:

+ Bạn code, sửa đổi các file trong Thư mục làm việc (Working tree)

+ Bạn tổ chức những sự thay đổi nào muốn commit tiếp theo, cơ bản là việc đánh dấu sự thay đổi vào khu vực Khu vực sắp xếp (staging)

+ Bạn thực hiện commit để các file đánh dấu trong staged lưu vào Git (database) như nhưng snapshot (chụp file)

**1.1.2. Sơ lược lịch sử phát triển**

Cũng như nhiều thứ tuyệt vời khác trong cuộc sống, Git ra đời từ một chút của sự huỷ diệt/phá sản/kết thúc có tính sáng tạo và sự tranh cãi nảy lửa. Nhân của Linux là một dự án phần mềm mã nguồn mở của một phạm vi khá lớn. Trong phần lớn thời gian bảo trì của nhân Linux (1991-2002), các thay đổi của phần mềm được truyền đi dưới dạng các bản vá và các tập tin lưu trữ. Vào năm 2002, dự án nhân Linux bắt đầu sử dụng một DVCS độc quyền có tên là BitKeeper.

Vào năm 2005, sự hợp tác giữa cộng đồng phát triển nhân Linux và công ty thương mại phát triển BitKeeper bị phá vỡ, và công cụ đó không còn được cung cấp miễn phí nữa. Chính điều này đã thúc đẩy cộng đồng phát triển Linux (chính xác hơn là Linus Torvalds, người sáng lập ra Linux) phát triển công cụ của riêng họ dựa trên những bài học từ việc sử dụng BitKeeper. Một số mục tiêu của hệ thống mới được vạch ra như sau:

* Nhanh
* Thiết kế đơn giản
* Hỗ trợ tốt cho "phát triển phi tuyến tính" (non-linear development) - (hàng ngàn nhánh song song)
* Phân tán toàn diện
* Có khả năng xử lý các dự án lớn giống như nhân Linux một cách hiệu quả (về mặt tốc độ và khối lượng dữ liệu)

Kể từ khi ra đời năm 2005, Git đã tiến hoá và phát triển toàn diện để dễ dàng sử dụng hơn, tuy thế các tiêu chí ban đầu vẫn được đảm bảo. Nó nhanh một cách đáng kinh ngạc, vô cùng hiệu quả với các dự án lớn, và một hệ thống phân nhánh không thể tin được cho phát triển phi tuyến tính

### 1.1.3. Lợi ích của Git

Thì thường các lập trình viên sẽ làm các dự ấn song song, nên sử Git là cần thiết để đảm bảo việc quản lý và không bị xung đột code giữa các thành viên khác. Và trong các dự án cũng thay đổi thường xuyên, vì vậy nên một hệ thống kiểm soát phiên bản cho phép các nhà phát triển có thể quay lại khôi phục các thay đổi trước đó.

* Dễ sử dụng, thao tác dễ dàng và đơn giản, đảm bảo an toàn.
* Dễ phân nhánh các branch, có thể giúp quy trình làm việc đucợ hiệu quả hơn.
* Bạn có thể clone mã nguồn từ kho lưu trữ hoặc có thể clone một flie thay đổi nào đó, hay các nhánh nào đó.
* Deloyment sản phẩm của bạn một cách dễ dàng.

## 1.2. Kiến thức cơ bản về GIT

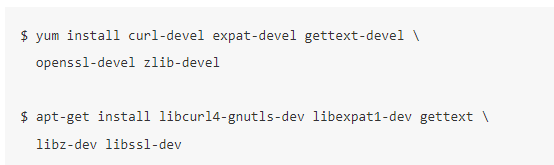
### 1.2.1. Cài đặt GIT

Để sử dụng Git đương nhiên chúng ta sẽ phải cài đặt Git lên máy tính cá nhân của chúng ta để thao tác. Bản chất của việc cài đặt Git là chúng ta sẽ có một Server Git Local để thao tác.

1.2.1.1. Cài đặt từ mã nguồn

Sẽ hữu ích hơn nếu bạn có thể cài đặt Git từ mã nguồn, vì bạn sẽ có được phiên bản mới nhất. Mỗi phiên bản của Git thường bao gồm nhiều cải tiến hữu ích về giao diện người dùng, vì thế cài đặt phiên bản mới nhất luôn là cách tốt nhất nếu như bạn quen thuộc với việc biên dịch phần mềm từ mã nguồn. Đôi khi nhiều phiên bản của Linux sử dụng các gói (package) rất cũ; vì thế trừ khi bạn đang sử dụng phiên bản mới nhất của Linux hoặc thường xuyên cập nhật, cài đặt từ mã nguồn có thể nói là sự lựa chọn tốt nhất.

Để cài đặt được Git, bạn cần có các thư viện mà Git sử dụng như sau: curl, zlib, openssl, expat, và libiconv. Ví dụ như bạn đang sử dụng một hệ điều hành có sử dụng yum (như Fedora) hoặc apt-get (như các hệ điều hành xây dựng trên nền Debian), bạn có thể sử dụng một trong các lệnh sau để cài đặt tất cả các thư viện cần thiết:



Khi đã cài đặt xong tất cả các thư viện cần thiết, bước tiếp theo là tải về phiên bản mới nhất của Git từ website của nó:



Sau đó, dịch và cài đặt:



Sau khi thực hiện xong các bước trên, bạn cũng có thể tải về các bản cập nhật của Git dùng chính nó như sau:



1.2.1.2. Cài Đặt Trên Linux

Nếu như bạn muốn cài đặt Git trên Linux thông qua một chương trình cài đặt, bạn có thể làm việc này thông qua phần mềm quản lý các gói cơ bản đi kèm với hệ điều hành của bạn. Nếu bạn đang sử dụng Fedora, bạn có thể dùng yum:



Còn nếu bạn đang sử dụng một hệ điều hành dựa trên nhân Debian như Ubuntu, hãy dùng apt-get:



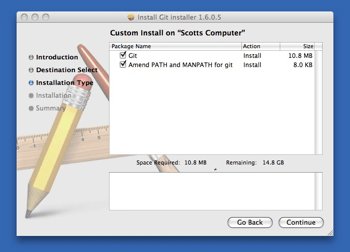
Với CentOS



1.2.1.3. Cài Đặt Trên Mac

Có hai cách đơn giản để cài đặt Git trên Mac. Cách đơn giản nhất là sử dụng chương trình cài đặt có hỗ trợ giao diện, bạn có thể tải về từ trang web của Google Code





Cách khác để cài đặt Git là thông qua MacPorts. Nếu như bạn đã cài đặt MacPorts, Git có thể được cài đặt sử dụng lệnh sau:



1.2.1.4. Cài Đặt Trên Windows

Cài đặt Git trên Windows rất đơn giản. Mặc định bạn sẽ làm việc với Git ở giao diện dòng lệnh cmd, đối với Git trên Windows mà bạn cài như trên bạn cũng có thể sử dụng Git Bash - git-bash.exe trong thư mục C:\Program Files\Git, để mở giao diện dòng lệnh tiện ích hơn là cmd mặc định của Windows. Nó có giao diện như dưới

Shape, rectangle

Description automatically generated with medium confidence

Các tác vụ với Git thực hiện bằng lệnh git với tham số thích hợp.

Lệnh đầu tiên là hãy kiểm tra Version của Git đang dùng:



Hiện thị các lệnh Git bằng cách gõ:



**\* Cấu hình Git**

Thiết lâp các tham số chung ban đầu:

- Tên trên hệ thống Git



- Email của bạn trên Git



- Hiển thị các tham số thiết lập Git



## \* Tạo Repo mới trên Local

Để tạo 1 repo mới bạn hãy tạo 1 folder. Di chuyển đến Folder này và gõ dòng lệnh:

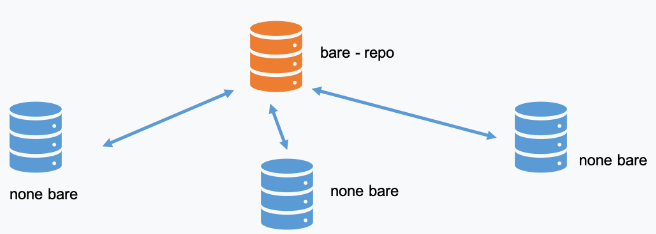


Sau lệnh này bạn có một Repo ở local và bắt đầu thực thi được các lệnh khác của Git.

- Khi bạn cần tạo ra một Repo Git mà nó chỉ có chức năng lưu trữ - không có thư mục làm việc thì thực hiện lệnh:



Loại dự án Git này thì bạn có thể truy cập, lưu trữ, nhưng không soạn thảo, sửa file, thực hiện commit trực tiếp tại dự án. Thường tạo loại dự án này để lưu trữ như là Remote Repo (Tạo Repo git trên Server), từ đó lấy về Local (lệnh git clone), và để local đẩy dữ liệu Git lên



**\* Mô hình hoạt động cơ bản trên Git Local**

Thường khi làm việc với thư mục Repo Local của bạn, có 3 trees được duy trì bởi Git (Toàn bộ được lưu trữ trong thư mục .git trên Folder của bạn).

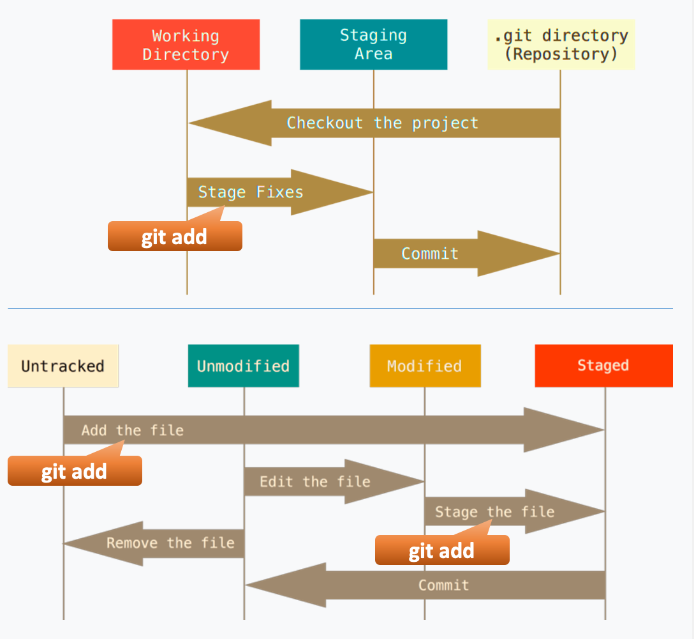
* Thư mục đang làm việc (Working Directory) có chứa folder .git và các tập tin hiện tại
* Chỉ mục (Index) đóng vai trò Staging area
* Và HEAD trỏ đến commit gần nhất của bạn



**\* Thêm (add) && commit**

Lệnh git add sử dụng để đánh chỉ mục (index) các nội dung mới, mới cập nhật trong thư mục làm việc, nó chuẩn bị nội dung sắp xếp cho lần commit tiếp theo.

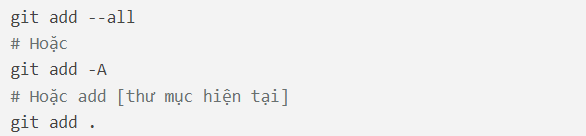
Hình ảnh sau cho biết những ảnh hưởng của lệnh git add



- Đưa vào vùng staging file, thư mục cụ thẻ:



- Đưa vào vùng staging toàn bộ thư mục làm việc:



Sau khi đưa vào vùng staging, vùng này có snapshot thì bạn đã có thể sẵn sàng để thực hiện lệnh git commit để lưu sự thay đổi vào CSDL của Git

**\* Xem trạng thái**

Sử dụng lệnh git để biết thông tin trạng thái sửa đổi, thêm mới, xóa các file trước khi thực hiện commit. Thông thường thì có thể thi hành ngay lệnh với cú pháp đơn giản sau để có thông tin trạng thái đầy đủ, chỉ tiết



**\* Lưu dữ liệu**

Lệnh git commit thực hiện lưu vào CSDL Git toàn bộ nội dung chứa trong index (vùng staging) và kèm theo nó là một đoạn text thông tin (log) mô tả sự thay đổi của của commit này so với commit trước. Sau khi commit con trỏ HEAD tự động dịch chuyển đến commit này (Trong nhánh hiện tại)

- Lệnh commit cơ bản, đơn giản nhất là thực hiện với tham số -m để kèm dòng thông tin về commit



- Khi cho tham số -a thì nó tương đương thực hiện lệnh git add để đưa các file đang được giám sát có sự thay đổi vào staging rồi tự động chạy git commit



- Nếu commit đã được tạo ra nhưng chưa thực hiện push lên remote (khi có làm việc với Remote Repo - nói ở các phần sau) thì bạn có thể tạo ra commit mới thay thế cho commit cuối cùng đó. Dùng trong trường hợp không muốn tạo ra nhiều commit trong lịch sử commit thì cho vào lệnh tham số --amend



**\* Hủy commit cuối**

- Trường hợp này sẽ hủy commit cuối, con trỏ HEAD sẽ chuyển về commit cha. Đồng thời những thay đổi của commit cuối được chuyển vào vùng staging nhằm để có cơ hội commit lại hoặc sửa đổi, cú pháp lệnh như sau:



- Hủy git add: nếu bạn đã dùng lệnh git add để cập nhật thay đổi vào vùng staging, bạn có thể hủy thao tác này bằng cách thực hiện lệnh:



**-** Hủy đưa một file vào stagging: nếu muốn hủy một file nào đó trong vùng staging chứ không phải toàn bộ thì dùng lệnh



**\* Kiểm tra thay đổi**

Lệnh git diff hiện thị thông tin thay đổi giữa thư mục làm việc và vùng index (staging) hoặc với commit cũ, thông tin thay đổi giữa index(staging) và commit, thông tin thay đổi giữa hai nhánh …



- Kiểm tra sự thay đổi index với commit cuối



- Kiểm tra sự thay đổi giữa 2 commit



- Kiểm tra sự thay đổi giữa 2 nhánh



**\* Sao chép dữ liệu**

Lệnh git clone để sao chep, copy một Git Repo (kho chứa dự án Git) về máy đang local. Một số trường hợp sử dụng git clone như:

- Copy một Repo từ máy Remote về Local

- Copy một Repo từ thư mục này sang một thư mục khác

- Copy một Repo từ một Url (https) ví dụ GitHub

- Copy Repo từ thư mục này sang thư mục khác



Có thể chỉ rõ thư mục cần copy về thay vì tại thư mục hiện tại



- Copy Repo bằng giao thức https



- Mặc định nó sẽ sao chép về nhánh hoạt động, để xem tất cả các nhánh có trên Remote dùng lệnh



**\* Chuyển nhánh**

Giả sử đang ở nhánh nào đó, muốn chuyển sang nhánh master thì thực hiện lệnh:



- Phục hồi file từ phiên bản cũ

Giả sử có file index.html, muốn phục hồi nó về phiên bản ở commit có mã hash là HASH, thì thực hiện:



\* **Làm việc với nhánh (tạo nhanh, gộp nhánh, xóa nhánh, xử lí xung đột)**

- Tạo nhánh mới:

***$ git branch tennhanh***

- Chuyển nhánh làm việc

***$ git checkout tennhanh***

- Gộp nhánh

***$ git merge***

**\* Đấy dữ liệu từ local lẻn server**

Lệnh git push được sử dụng để đẩy các commit mới ở máy trạm (local repo) lên server (remote repo). Nguồn để đẩy lên là nhánh mà con trỏ HEAD đang trỏ tới (nhánh làm việc).

Nếu là lần đầu tiên đẩy Local Repo lên Remote Repo mới khởi tạo thì cần tạo ra một theo dõi kết nối, upstream giữa local và remote, vậy hãy dùng tham số -u. Ví dụ đẩy lên remote có tên origin và tạo upstream cho nhánh master



Sau khi có upstream, mỗi lần cần đẩy dữ liệu lên remote của nhánh master, chỉ việc thực hiện lệnh



Hoặc có thể đẩy một nhánh cụ thể, ví dụ đẩy nhánh beta lên remote



- Xóa một nhánh trên remote

Ví dụ bạn cần xóa nhánh beta, trên remote có tên origin



Bạn có thể kiểm tra các nhánh có trên remote bằng lệnh

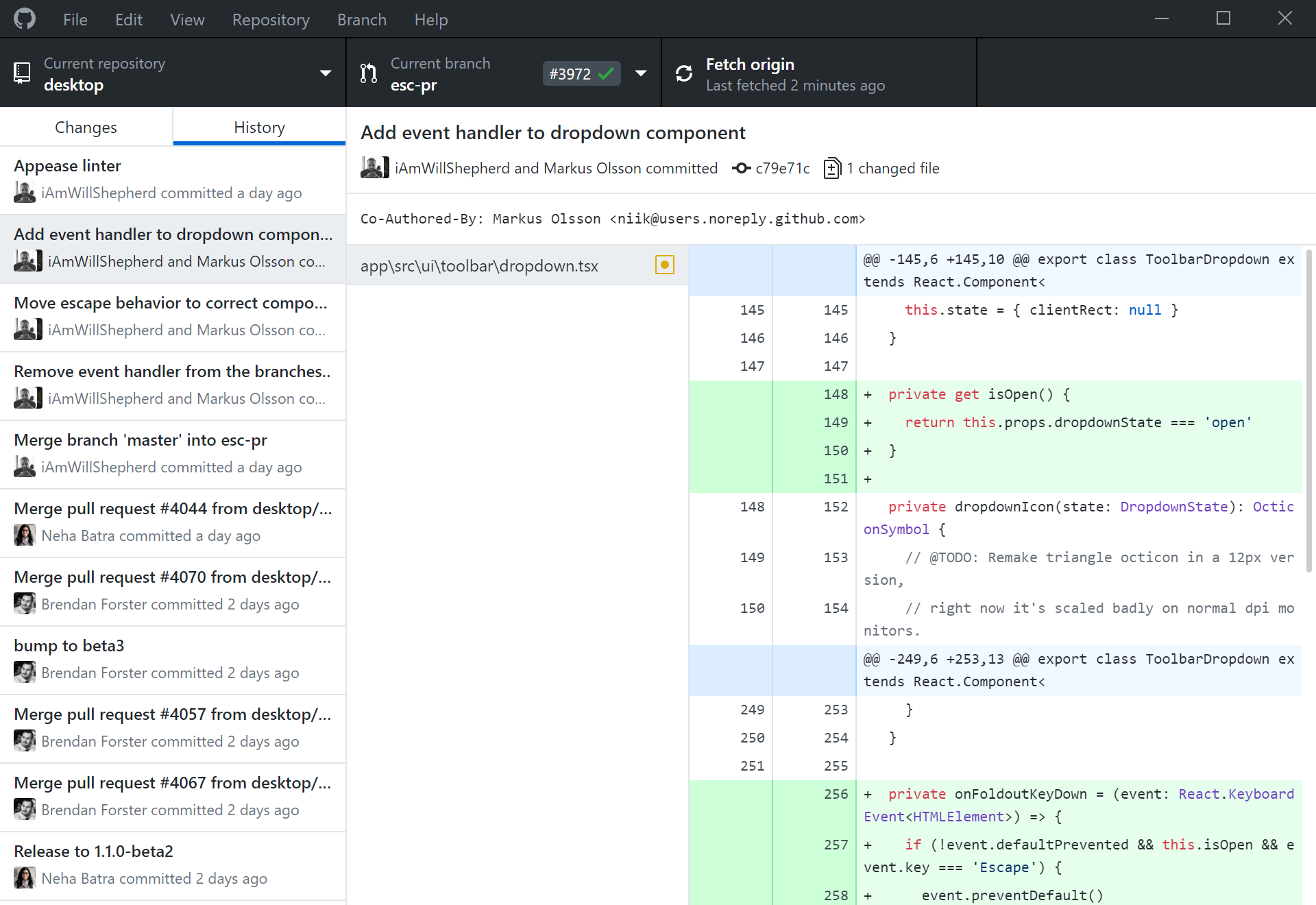


### 1.1.2. Một số lưu ý khi làm việc với Git

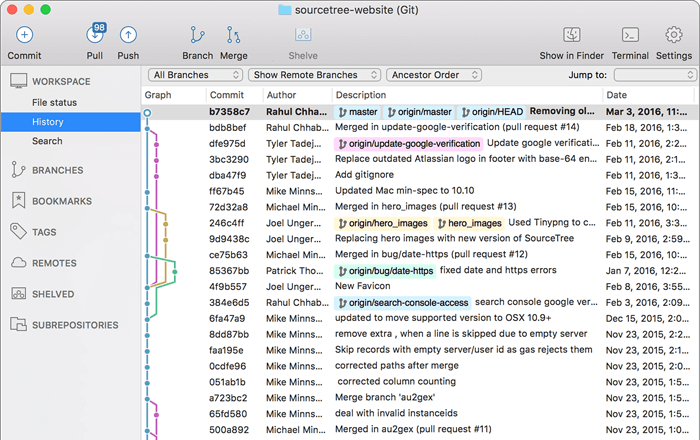
* Hãy lưu ý không nên commit nếu chưa hoàn tất process. Bạn nên test các thay đổi của bạn trước khi public.
* Nên commit thường xuyên bằng cách tách nhỏ từng commit của bạn và commit thường xuyên. Điều này sẽ hỗ trợ cho các thành viên trong nhóm dễ dàng tích hợp các công việc của họ hơn.
* Viết các ghi chú commit để cho các thành viên khác trong nhóm biết các thay đổi của bạn đã thực hiện. Nếu thực hiện càng cụ thể và chi tiết thì càng tốt.
* Bạn cần tận dụng các lợi ích của branch để giúp bạn dễ dàng theo dõi các dòng phát triển khác nhau.
* Bạn cũng nên theo một Git Workflow để đảm bảo sự đồng nhất giữa các thành viên trong nhóm.

### 1.2.3. Một số giao diện GUI của GIT

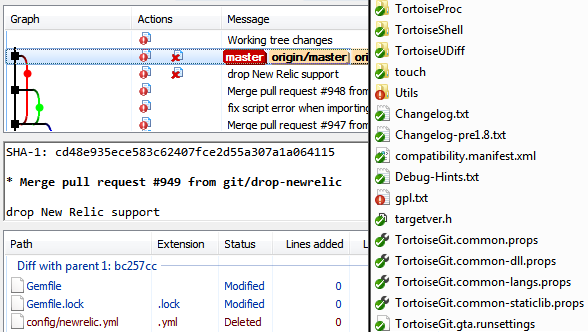
***- GitHub Desktop***



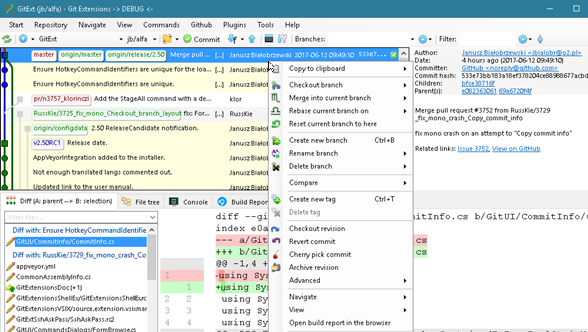
- ***SouceTree***



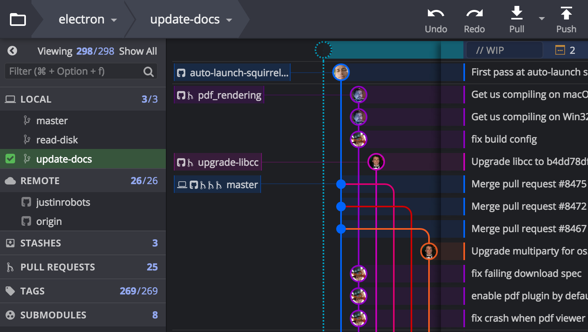
***- TortoiseGit***



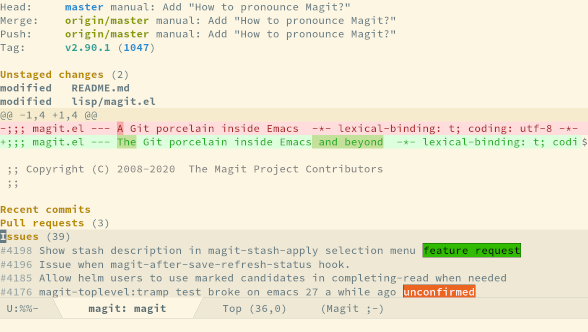
***- Git Extensions***



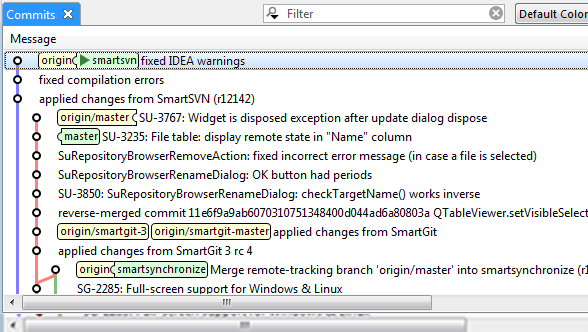
***- Git Kraden***



***- Magit***



***- SmartGit***



# **Chương 2: Hệ thống quản lí mã nguồn SVN**

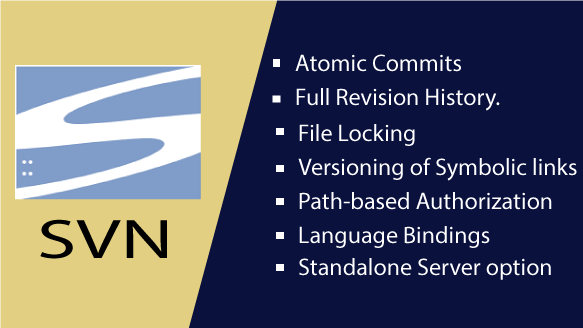
## 2.1. Giới thiệu tổng quan về SVN

### 2.1.1. Khái niệm SVN

SVN là viết tắt của Subversion . Nó được gọi là SVN vì các lệnh của nó (tên lệnh của nó là svn). Nó là một hệ thống kiểm soát phiên bản tập trung . Nó là một công cụ mã nguồn mở để kiểm soát phiên bản.

SVN được sử dụng để quản lý các phiên bản hiện tại và trước đó của các tệp như mã nguồn, tài liệu và tệp. Nó hoạt động như một cỗ máy thời gian cho các nhà phát triển và cho phép họ quay lại và duyệt qua lịch sử của dự án.

Subversion là mã nguồn mở và có Giấy phép Apache , và nó được phát triển bởi CollabNet Inc vào năm 2000. Nó được vận hành giống như CVS (Hệ thống các phiên bản đồng thời). Vào năm 2009, Apache Incubator đã chấp nhận nó và biến nó thành một sản phẩm apache cấp cao nhất.



**\* Các tính năng của SVN**

- Hỗ trợ các cam kết nguyên tử (Có thể xảy ra hoặc không).

- Lưu giữ một lịch sử sửa đổi đầy đủ.

- Hỗ trợ tin nhắn chương trình quốc tế hóa.

- Cung cấp khóa tệp cho các tệp không thể được hợp nhất.

- Cung cấp phiên bản của các liên kết tượng trưng.

- Hỗ trợ ủy quyền dựa trên đường dẫn.

- Có các ràng buộc ngôn ngữ cho một số ngôn ngữ lập trình như C #, Perl, Ruby, PHP, Python và Java.

- Có tùy chọn máy chủ độc lập (svnserver).

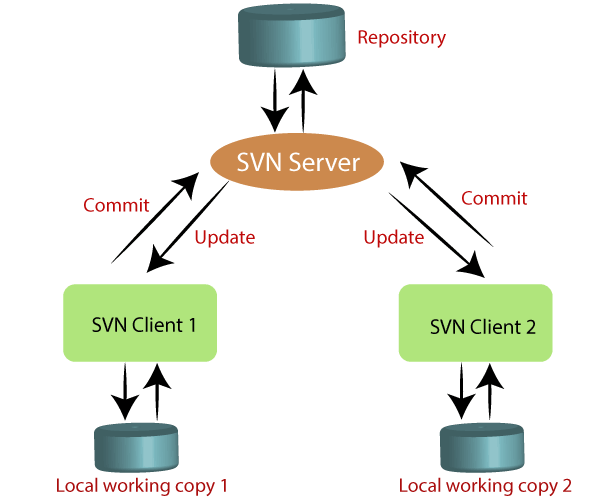
### 2.1.2 Các thành phần của SVN

Có hai thành phần cốt lõi của SVN; chúng như sau:

- Máy chủ SVN

- Khách hàng SVN

Các thành phần này có thể được coi là máy chủ web và trình duyệt web. Máy khách đóng vai trò là trình duyệt truy cập dữ liệu trên máy chủ



## 2.2. Kiến thức cơ bản về SVN

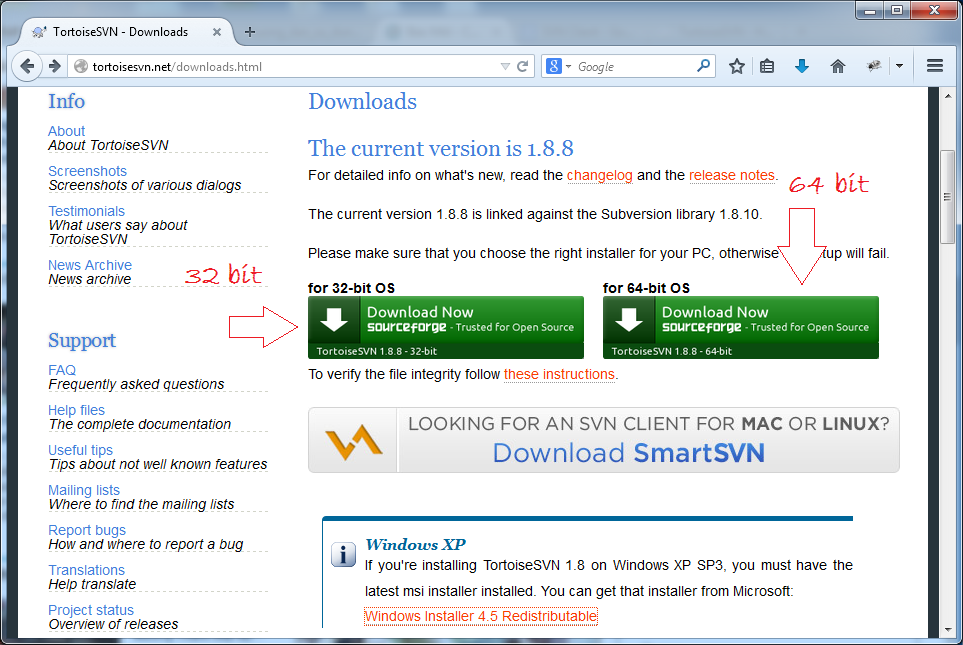
### 2.2.1. Cài đặt SVN

Để cộng tác trong dự án sử dụng SVN, chúng tôi phải cài đặt ứng dụng khách SVN. Sự lật đổ là một sản phẩm của Apache. Vì vậy, để hoàn tất cài đặt SVN, chúng ta phải cài đặt SVN server và SVN client.

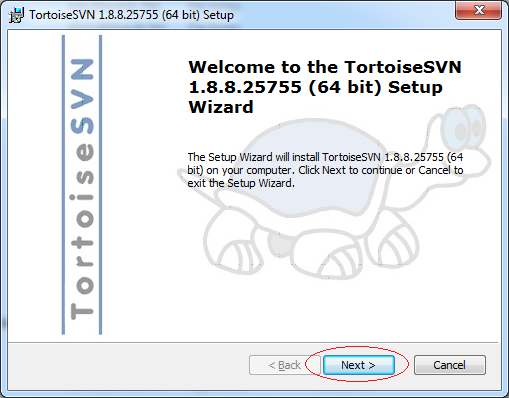
2.2.1.1. Cài đặt Tortoise SVN trên Windows

VisualSVN Server dễ cài đặt vì nó được phân phối dưới một trình cài đặt gói duy nhất. Nó chứa tất cả các thành phần cần thiết cho phiên bản mới nhất. Thật dễ dàng để chạy thiết lập và cấu hình máy chủ SVN trên máy của chúng tôi; nó chỉ là một vài quy trình bước. VisualSVN Server là miễn phí và mã nguồn mở để sử dụng thương mại theo giấy phép của Cộng đồng. Loại giấy phép này không cần đăng ký. Nó cho phép không giới hạn số lượng kho lưu trữ và tối đa 15 người dùng.

Nhấp vào tùy chọn tải xuống ở góc trên bên phải của trang chính thức. Theo tùy chọn này, chúng tôi sẽ tìm thấy tùy chọn cấu hình cửa sổ cho 32 bit và 64 bit.



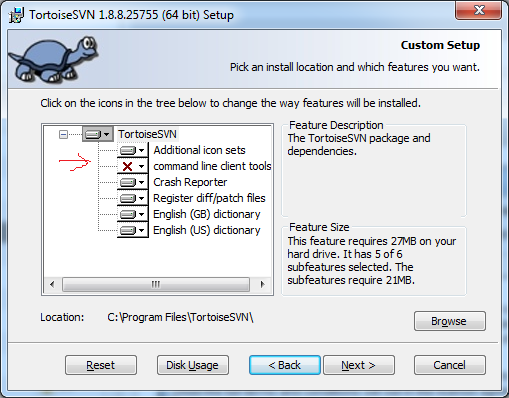
Chạy tệp trình cài đặt. Nó sẽ nhắc tùy chọn cài đặt. Nhấn Next để tiếp tục.



Bây giờ, nó sẽ yêu cầu chúng tôi cung cấp Thỏa thuận PHÉP CHO Người dùng Cuối. Chấp nhận các điều khoản và điều kiện bằng cách đánh dấu vào ô đã cho và nhấp vào tiếp theo để tiếp tục

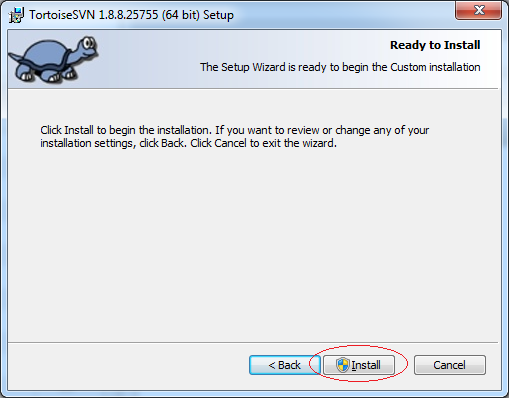


Bước tiếp theo là chọn và cài đặt vị trí và các tính năng. Chọn vị trí và chọn các tính năng bạn muốn cài đặt. Nhấn Next để tiếp tục cài đặt.

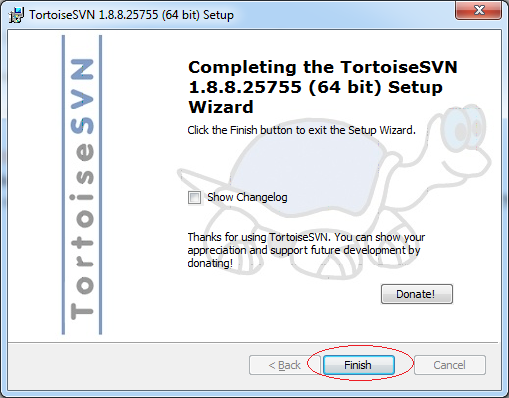




Đến đây chúng tôi đã hoàn thành tất cả các bước cần thiết để cài đặt TortoiseSVN. Nhấp vào tùy chọn cài đặt để cài đặt nó trên máy của chúng tôi. Hãy xem xét hình ảnh dưới đây:



Đã hoàn thành quá trình cài đặt. Nhấp vào **Kết thúc** để khám phá TortoiseSVN.

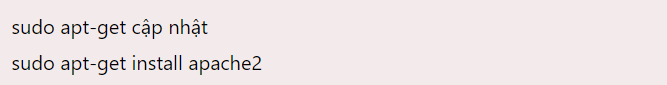


Đến đây chúng ta đã cài đặt thành công máy chủ SVN

**2.2.1.2. Cài đặt Tortoise SVN trên Windows**

Để truy cập máy chủ SVN bằng URL HTTP, chúng ta cần cài đặt máy chủ web Apache. Nếu chúng ta đã có máy chủ web Apache trên máy của mình, hãy bỏ qua bước này.

Để cài đặt máy chủ web Apache, hãy chạy lệnh dưới đây:



Lệnh trên sẽ cấu hình máy chủ web Apache trên máy của bạn. Bước tiếp theo là cài đặt máy chủ SVN. Hãy cài đặt máy chủ SVN.

**\* Cài đặt Máy chủ SVN**

SVN có mặc định trên hầu hết hệ điều hành GNU / Linux. Nếu nó không tồn tại trên máy của bạn, hãy cài đặt nó.

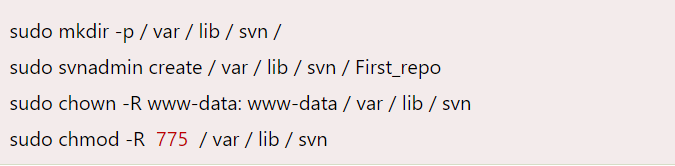
Để cài đặt gói máy chủ SVN và các gói phụ thuộc của chúng vào Linux, hãy chạy lệnh dưới đây:



Lệnh trên cũng sẽ cài đặt mô-đun SVN trên hệ thống của bạn. Sau khi cài đặt, hãy chạy lệnh trạng thái svn để đảm bảo rằng SVN đã được cài đặt thành công hay chưa. Sau khi cài đặt thành công, hãy tạo kho lưu trữ và người dùng đầu tiên cho máy chủ SVN.

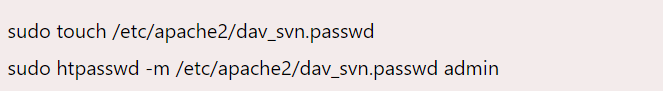
- Tạo kho lưu trữ đầu tiên

Các lệnh dưới đây sẽ được sử dụng để tạo kho lưu trữ SVN đầu tiên của bạn với tên First\_repo.

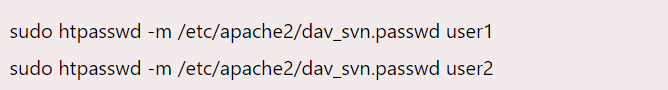


- Tạo người dùng cho SVN

Bây giờ tạo người dùng svn đầu tiên trong tệp /etc/apache2/dav\_svn.passwd . Những thông tin người dùng này sẽ được sử dụng để xác thực kho svn cho hoạt động kiểm tra, quy trình cam kết.



Để tạo thêm người dùng, hãy sử dụng các lệnh sau.



**\* Cài đặt SVN trên MAC**

Để cài đặt ứng dụng dòng lệnh SVN trên MacOS, hãy làm theo các bước sau:

- Bước 1: Tải xuống gói nhị phân MacOS được yêu cầu cho phiên bản macOS của bạn. Cài đặt tệp và làm theo các bước tiếp theo.

- Bước 2: Khi gói Binary được cài đặt, hãy chạy lệnh dưới đây để thiết lập đường dẫn



Bước 3: Xác nhận cài đặt và kiểm tra phiên bản, chạy lệnh svn version :

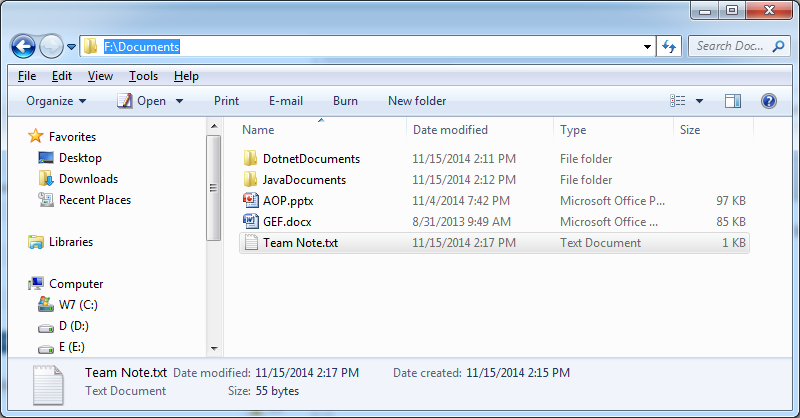


### 2.2.2. Bắt đầu với Tortoise SVN

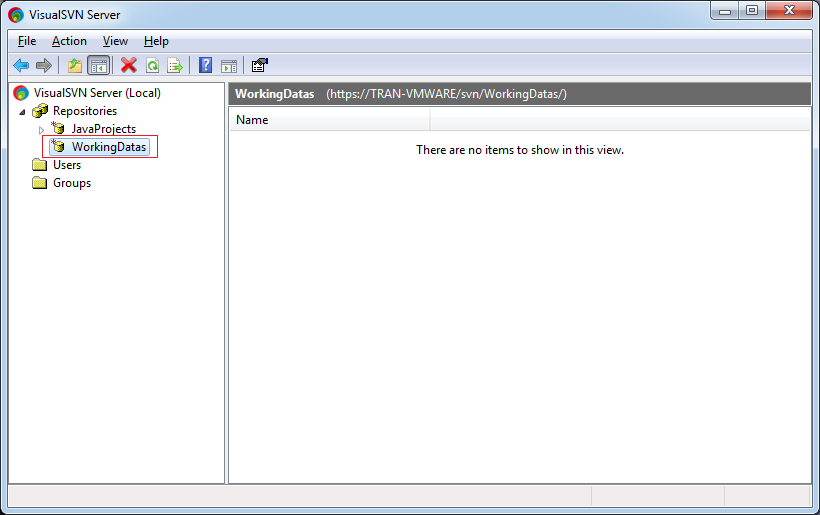
**2.2.2.1. Import dữ liệu lên Repository**

Đầu tiên một thành viên nhóm chia sẻ dữ liệu ban đầu lên SVN Repository. Nếu không có dữ liệu có thể bỏ qua.

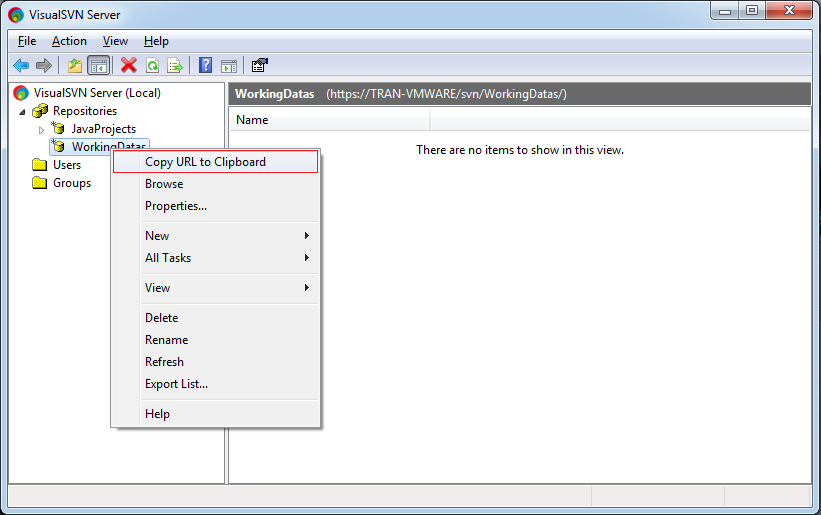
Import các file dữ liệu trong thư mục Documents vào SVN Repository:



Trên Visual SVN đã tạo sẵn một Repository có tên WorkingDatas



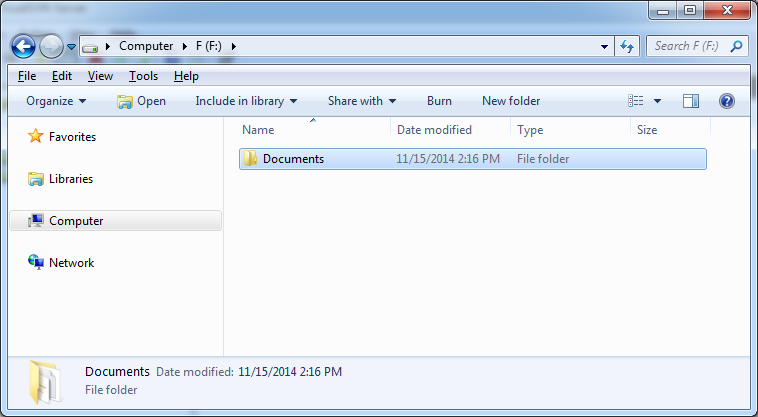
Copy URL:



URL copy được có thể chứa tên của máy chủ SVN

<https://TRAN-VMWARE/svn/WorkingDatas/>

Có thể thay đổi tên máy chủ bằng địa chỉ IP



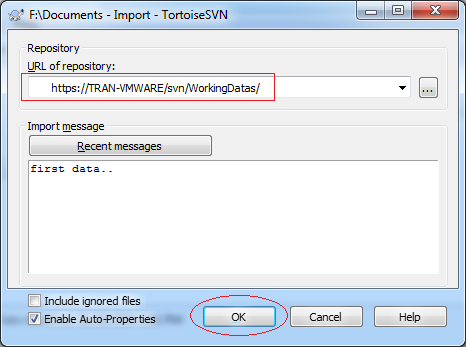
Nhấp chuột phải vào thư mục Documents, chọn:

TortoiseSVN/Import…

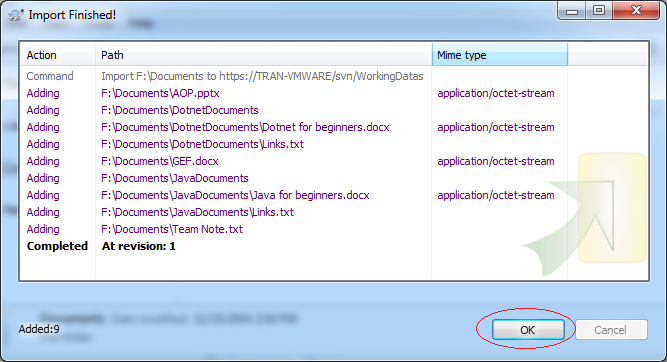


Nhập vào URL của SVN Repository:

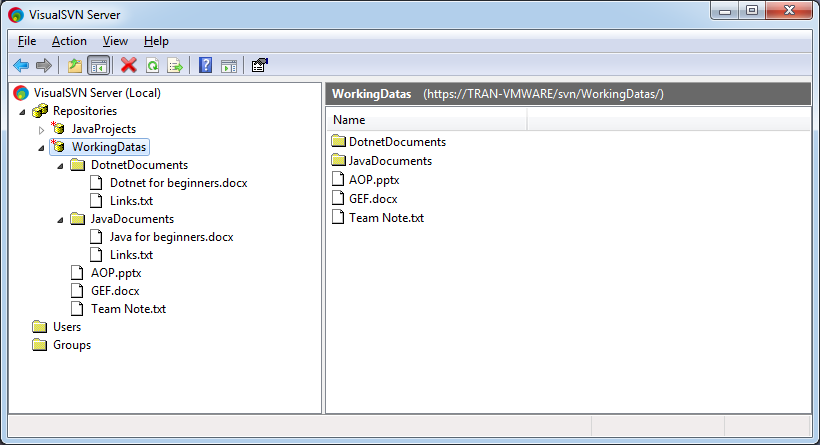
<https://TRAN-VMWARE/svn/WorkingDatas/>



Dữ liệu đã được Import vào SVN Repository



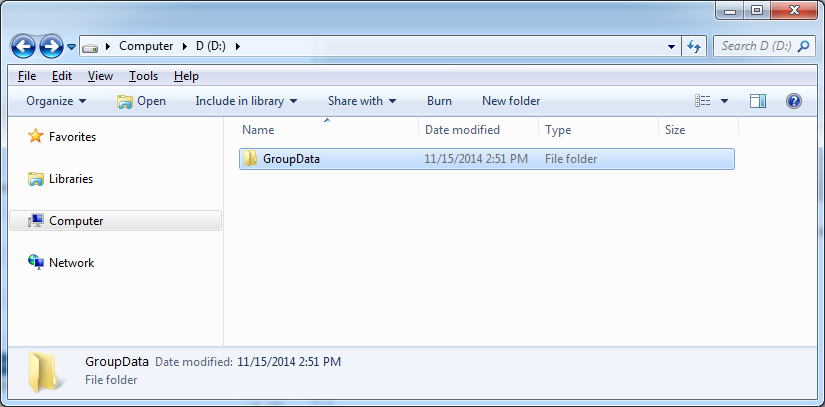
Nhìn trên Visual SVN:

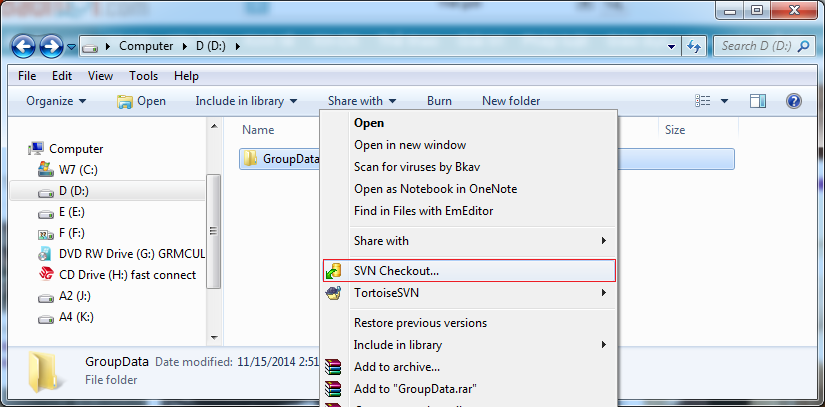


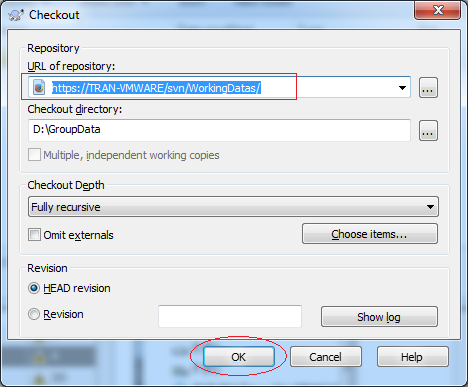
**2.2.2.2. Checkout**

Tất cả các thành viên nhóm sẽ checkout các dữ liệu từ trên SVN Repository về máy tính của mình, bao gồm cả người inport dữ liệu ban đầu vào SVN Repository.

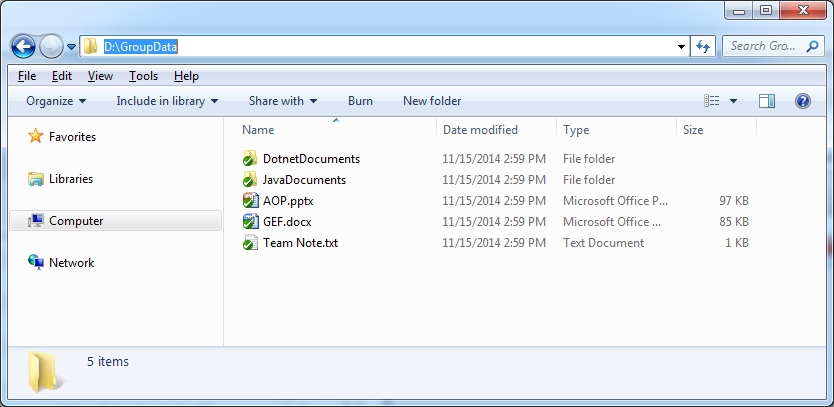
Tại các máy tính của các thành viên, tạo mới một thư mục rỗng nào đó, và checkout dữ liệu từ SVN Repository về đó. Thư mục này sẽ là thư mục làm việc của mỗi thành viên trên máy tính của mình.







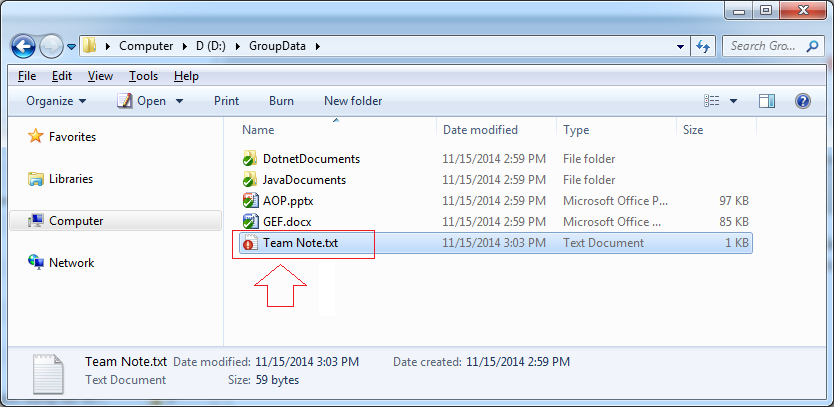
Dữ liệu đã được lấy về.



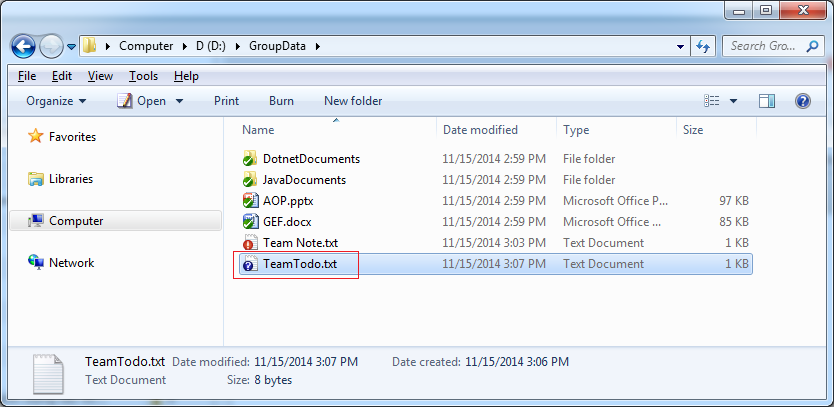
**2.2.2.3. Commit và Update**

Checkout là hành động lấy toàn bộ dữ liệu được chia sẻ trên Repository xuống máy địa phương lần đầu tiên. Sau đó, dữ liệu được sửa đổi tại địa phương. Việc đưa dữ liệu đã thay đổi lên Repository được gọi là Commit. Và được cập nhật các dữ liệu đã thay đổi về là hành động Update.

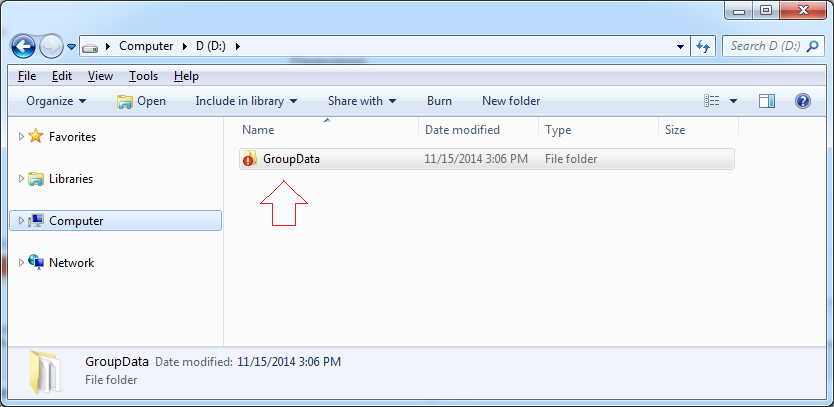
Sửa (hoặc thêm ) file trên các thư mục rồi Commit lên, thay đổi file TeamNote.txt và lưu lại, khi đó biểu tượng SVN của file này thay đổi, thông báo rằng file này đã bị sửa đổi.



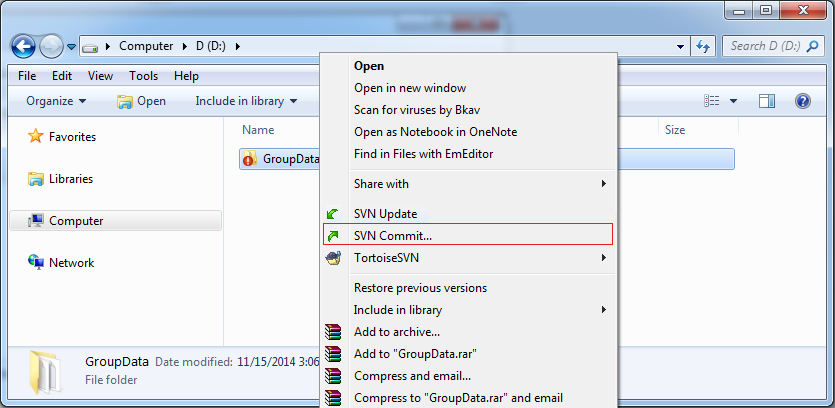
Tiếp theo tạo mới một file TeamTodo.txt. Biểu tượng của file mới được tạo ra có dấu ?, thông báo rằng file này mới, và chưa được quản lý bởi SVN.



Thư mục SVN chứa file có dữ liệu bị sửa đổi cũng thay đổi biểu tượng

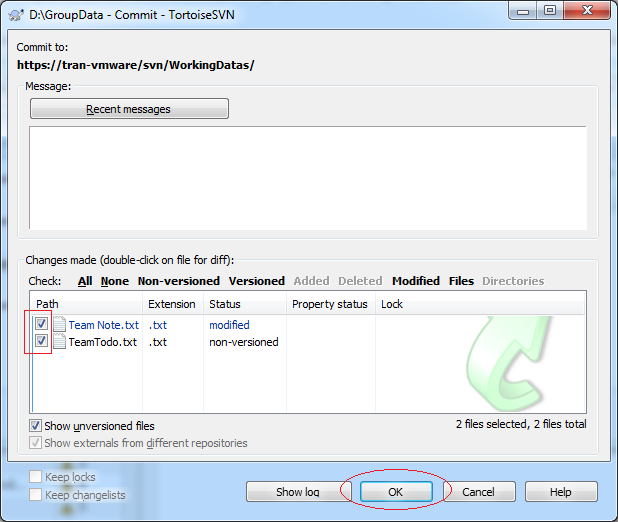


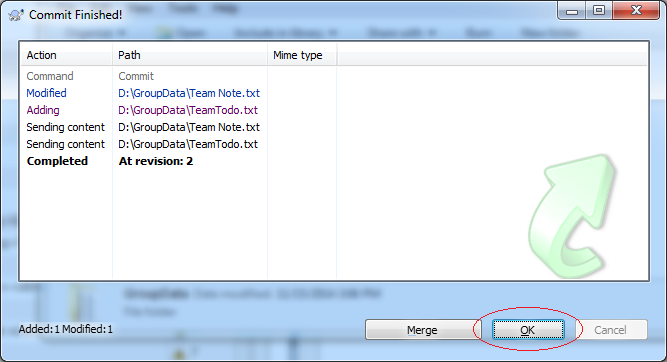
Để Commit dữ liệu lên SVN Repository, nhấn phải chuột vào file hoặc thư mục cần Commit, chọn SVN Commit …



Check chọn các file cần commit trên danh sách

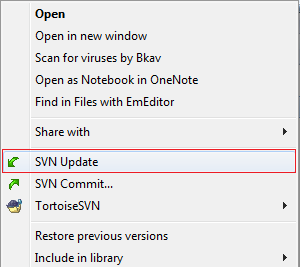
Mặc định Tortoise SVN tự động check vào các file sửa đổi





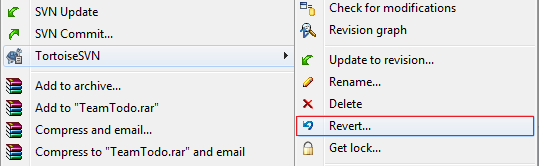
Update:

Thành viên khác trong nhóm có thể Update(Cập nhập) các dữ liệu mới nhất từ SVN Repository



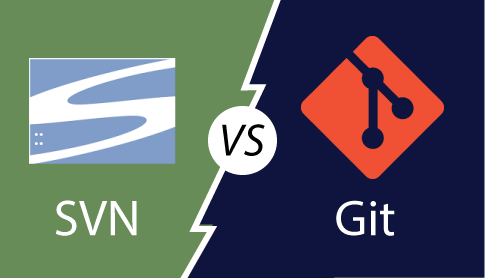
**2.2.2.4. Revert**

Chức năng Revert được sử dụng để trả dữ liệu về ban đầu. Chẳng hạn khi bạn cần sửa đổi dữ liệu của một vài file, revert giúp cho file trở lại như trước khi thay đổi.



### 2.2.3 So sánh SVN với Git

Apache Subversion hay SVN là một trong những hệ thống kiểm soát phiên bản tập trung phổ biến nhất . Bây giờ, mức độ phổ biến của SVN giảm đi, nhưng vẫn có hàng triệu dự án được lưu trữ trong đó. Nó có thể tiếp tục được duy trì tích cực bởi một cộng đồng nguồn mở. Trong SVN, bạn có thể kiểm tra một phiên bản duy nhất của kho lưu trữ. Nó lưu trữ dữ liệu trong một máy chủ trung tâm. Hạn chế của SVN là, nó có toàn bộ lịch sử trên một kho lưu trữ cục bộ, điều này sẽ giới hạn bạn. Bạn chỉ có thể thực hiện cam kết, khác biệt, nhật ký, chi nhánh, hợp nhất, chú thích tệp, v.v.



Mặc dù Git là một hệ thống kiểm soát phiên bản phân tán phổ biến , có nghĩa là bạn có thể sao chép kho lưu trữ của mình, do đó bạn có thể nhận được bản sao hoàn chỉnh của toàn bộ lịch sử của dự án đó. Nó có nghĩa là bạn có thể truy cập tất cả các cam kết của mình.

Git có nhiều ưu điểm hơn SVN. Sẽ tốt hơn nhiều cho những nhà phát triển không phải lúc nào cũng được kết nối với kho lưu trữ chính. Ngoài ra, nó nhanh hơn nhiều so với SVN. Để hiểu rõ hơn về sự khác biệt giữa Git và Subversion, chúng ta hãy xem xét các điểm quan trọng sau đây.

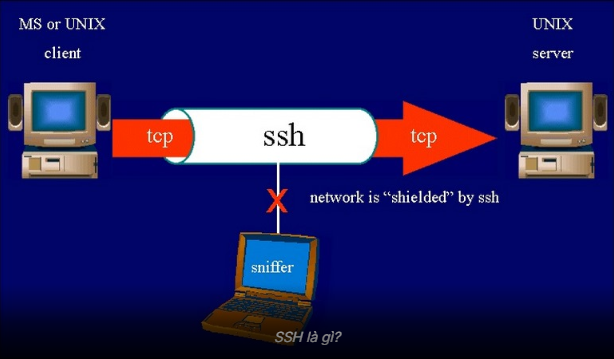
|  |  |
| --- | --- |
| SVN | Git |
| Đó là một hệ thống kiểm soát phiên bản tập trung | Đó là một hệ thống kiểm soát phiên bản phân tán. |
| Đó là kiểm soát sửa đổi. | Git là một SCM (quản lý mã nguồn). |
| Nó không giữ một kho lưu trữ nhân bản. | Nó có một kho lưu trữ nhân bản. |
| Các nhánh trong SVN là một thư mục diễn ra trong kho lưu trữ. Một số lệnh đặc biệt được yêu cầu Để hợp nhất các nhánh. | Các nhánh Git quen thuộc để làm việc. Hệ thống Git giúp hợp nhất các tệp một cách nhanh chóng và cũng hỗ trợ tìm kiếm những tệp chưa được hợp nhất. |
| Nó có một số sửa đổi quốc tế. | Nó không có số sửa đổi Toàn cầu. |
| SVN không chứa bất kỳ nội dung được băm bằng mật mã nào. | Nó có nội dung được băm bằng mật mã để bảo vệ nội dung khỏi bị hỏng kho lưu trữ do sự cố mạng hoặc lỗi đĩa. |
| SVN lưu trữ nội dung dưới dạng tệp. | Git nội dung được lưu trữ dưới dạng siêu dữ liệu. |
| Nội dung của SVN kém an toàn hơn Git. | Git có nhiều tính năng bảo vệ nội dung hơn SVN |
| CollabNet, Inc đã phát triển SVN. | Linus Torvalds đã phát triển git cho nhân Linux. |
| SVN được phân phối theo giấy phép nguồn mở. | Git được phân phối theo GNU (Giấy phép công cộng chung). |

# **Chương 3: Hệ thống quản lí mã nguồn SSH**

## 3.1. Giới thiệu tổng quan về SSH

### 3.1.1. Khái niệm SSH

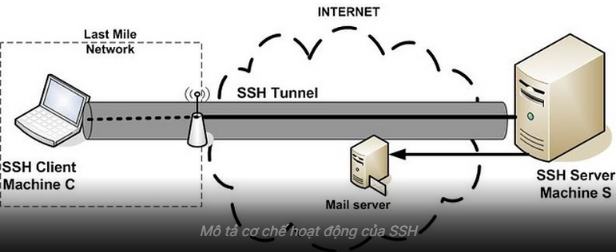
SSH viết đầy đủ là Secure Shell, đây là một giao thức hỗ trợ các nhà quản trị mạng truy cập vào máy chủ từ xa thông qua mạng internet không bảo mật. Ngoài ra, SSH còn cung cấp các bộ tiện ích phục vụ phát triển chính giao thức SSH.



SSH tạo ra cơ chế xác thực qua mật khẩu mạnh, hình thành mối liên kết giao tiếp dữ liệu mã hóa ra giữa hai máy qua môi trường internet. Ngày nay giao thức SSH được giấy quản trị mạng sử dụng phổ biến trong quá trình quản lý, điều chỉnh ứng dụng từ xa. Nó cho phép vị tự đăng nhập vào mạng máy tính và thực hiện một số tác vụ cơ bản như dịch chuyển file.

### 3.1.2. Cơ chế hoạt động

Bạn sẽ chưa thể hiểu rõ SSH là gì nếu chưa nắm rõ cơ chế hoạt động của giao thức này. SSH gồm cả giao thức mạng và bộ tiện ích cơ bản để triển khai chính giao thức đó. Cụ thể, SSH ứng dụng ảnh mô hình client-server, kết nối với vùng hiển thị Session và vùng Session chạy.



Mô tả cơ chế hoạt động của SSHMô tả cơ chế hoạt động của SSH

Khi triển khai SSH hệ thống sẽ hỗ trợ cả giao thức ứng dụng, sử dụng cho trình giả lập Terminal hoặc truyền file. Trong thực tế, người ta còn sử dụng SSH để phát triển tunnel bảo mật cho các giao thức ứng dụng.

Mục đích SSH được tạo ra là để thay thế cho trình giả lập Terminal, cơ chế đăng nhập không an toàn (Telnet, Rlogin). Giao thức SSH hỗ trợ tính năng đăng nhập, khởi chạy Terminal Session thông qua hệ thống điều khiển từ xa.

Chức năng cơ bản nhất của giao thức SSH là liên kết với một host từ xa, ứng với một phiên Terminal bằng dòng lệnh “ssh server.example.org”. Dòng lệnh này có thể liên kết Client với một máy chủ server.example.com thông qua ID người dùng UserName.

### 3.1.3 Chức năng chính

Giao thức đảm nhiệm khá nhiều chức năng trong hệ thống điều khiển, liên kết máy chủ. Các chức năng cơ bản phải kể đến như:

- Hỗ trợ truy cập từ xa vào những hệ thống, thiết bị ứng dụng giao thức SSH.

- Cho phép dịch chuyển file an toàn.

- Thực thi lệnh bảo mật, an toàn trên hệ thống điều khiển từ xa.

- Quản lý an toàn và hiệu quả thành phần hạ tầng mạng.

SSH có thể kết hợp với Terminal Session thay thế cho những chương trình Telnet có tính bảo mật thấp.

## 3.2. Kiến thức cơ bản về SSH

### 3.2.1. Cài đặt SSH

Cài đặt OpenSSH Server trên các hệ thống Linux như CentOS, Ubuntu và Windows và cấu hình cơ bản để có dịch vụ SSH cho phép các máy Client kết nối đến

**3.2.1.1. Cài đặt OpenSSH Server trên Linux**

OpenSSH là phần mềm nguồn mở, nó cho phép tạo dịch vụ SSH Server (máy chủ SSH) và các chương trình SSH CLient. Cài đặt mặc định trong các máy chủ CentOS, Ubuntu thường là có sẵn OpenSSH Server, nếu chưa có thì cài đặt như sau (gồm cả trình client và server):

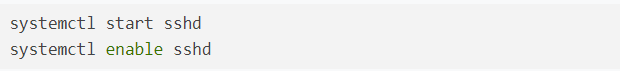
- Cài đặt OpenSSH trên CentOS/RHEL



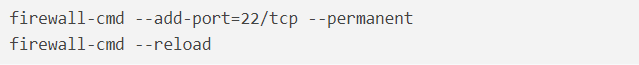
- Cài đặt OpenSSH Server trên Ubuntu



- Chạy dịch ụ SSH Server, thiết lập chạy ngay khi khởi động



- Mặc định SSH dùng cổng 22 nên bạn phải mở cổng này nếu có firewall



- File cấu hình của OpenSSH Server tại /etc/ssh/sshd\_config, bạn có thể mở và chỉnh sửa nó theo nhu cầu (có thể dùng Vim, Nano để soạn thảo). Sau khi sửa Config thì cần khởi động lại SSH Server.

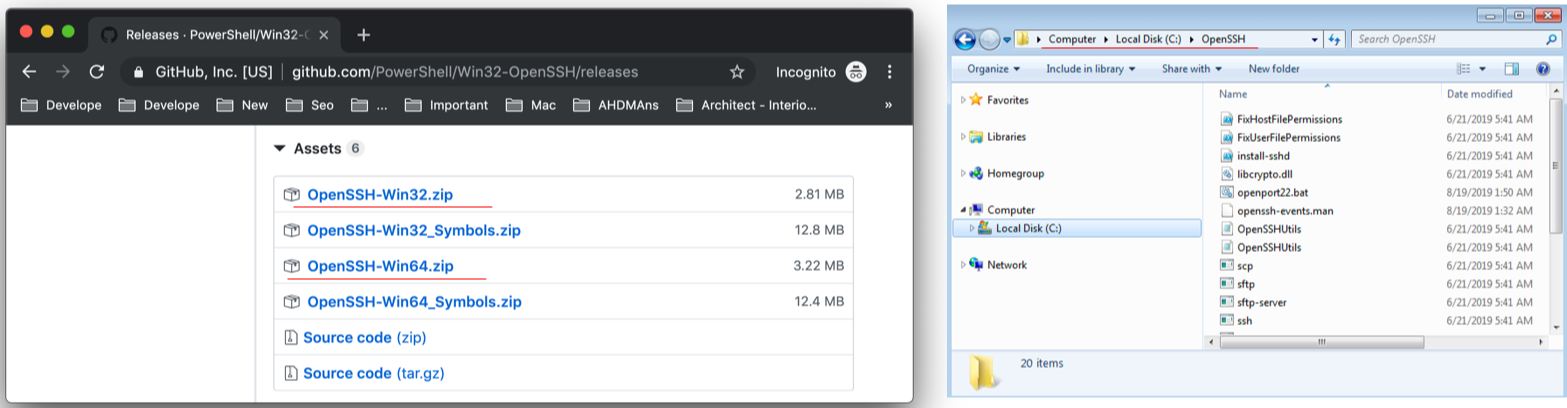


- Kiểm tra trạng thái SSH Server



**3.2.1.2. Cài đặt OpenSSH Server trên Windows**

* Bước 1: Tải về OpenSSH - vào đường link Win OpenSSH, chọn đúng phiên bản phù hợp với hệ thống của bạn (chọn OpenSSH-Win32.zip hay OpenSSH-Win64.zip theo kiến trúc Windows của bạn), File tải về giải nén vào thư mục muốn cài đặt, ví dụ chọn giải nén vào thư mục C:\OpenSSH

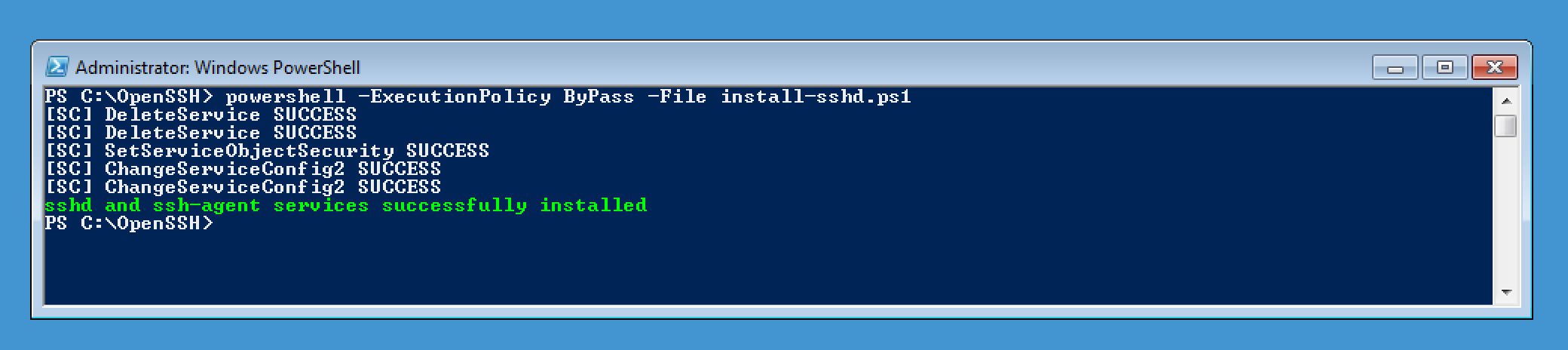


* Bước 2: Chạy PowserShell hoặc (Cmd) với quyền Administrator (Nhấn phải chuột vào biểu tượng PowerShell hoặc Cmd và chọn Run as Administrator ...), sau đó chuyển đến thư mục C:\OpenSSH bằng lệnh

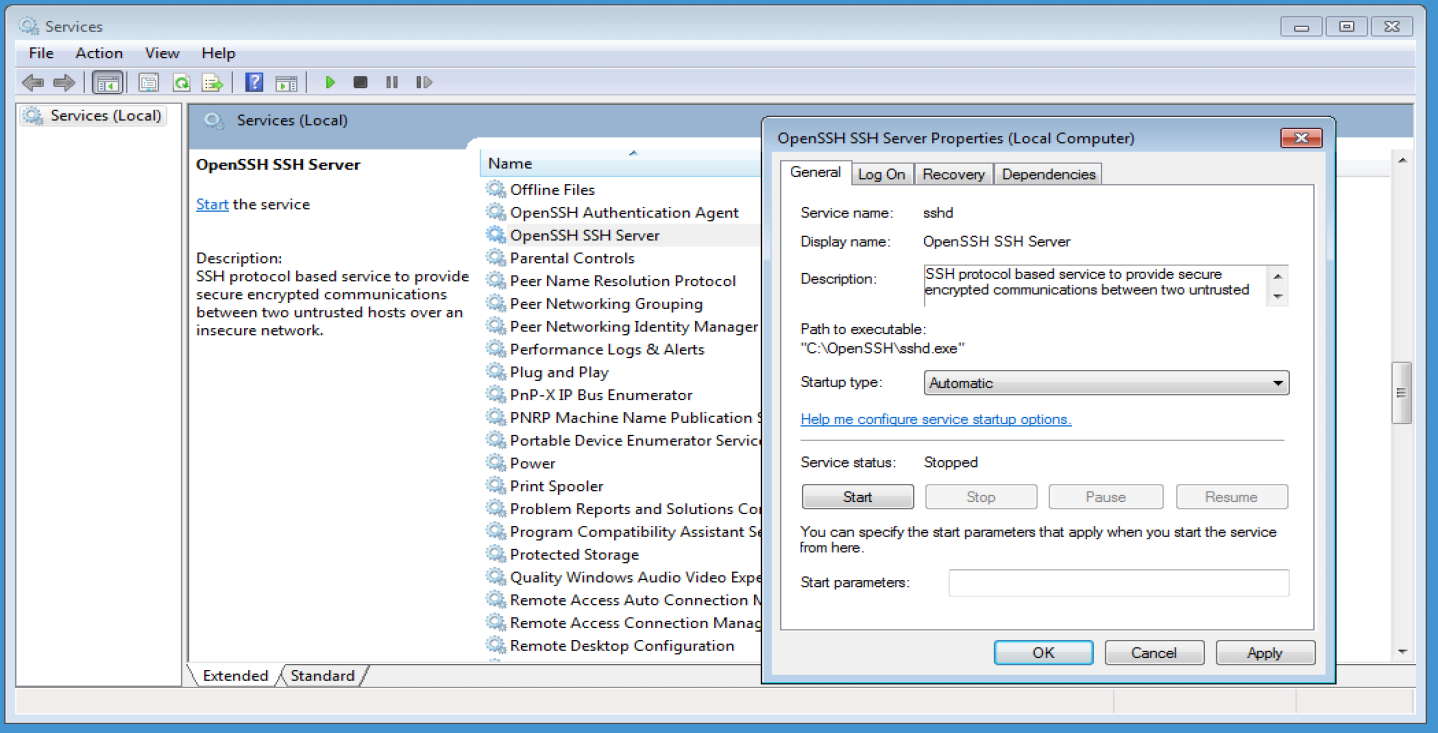


Đang ở thư mục C:\OpenSSH, gõ lệnh sau để cài đặt dịch vụ OpenSSH Server vào Windows

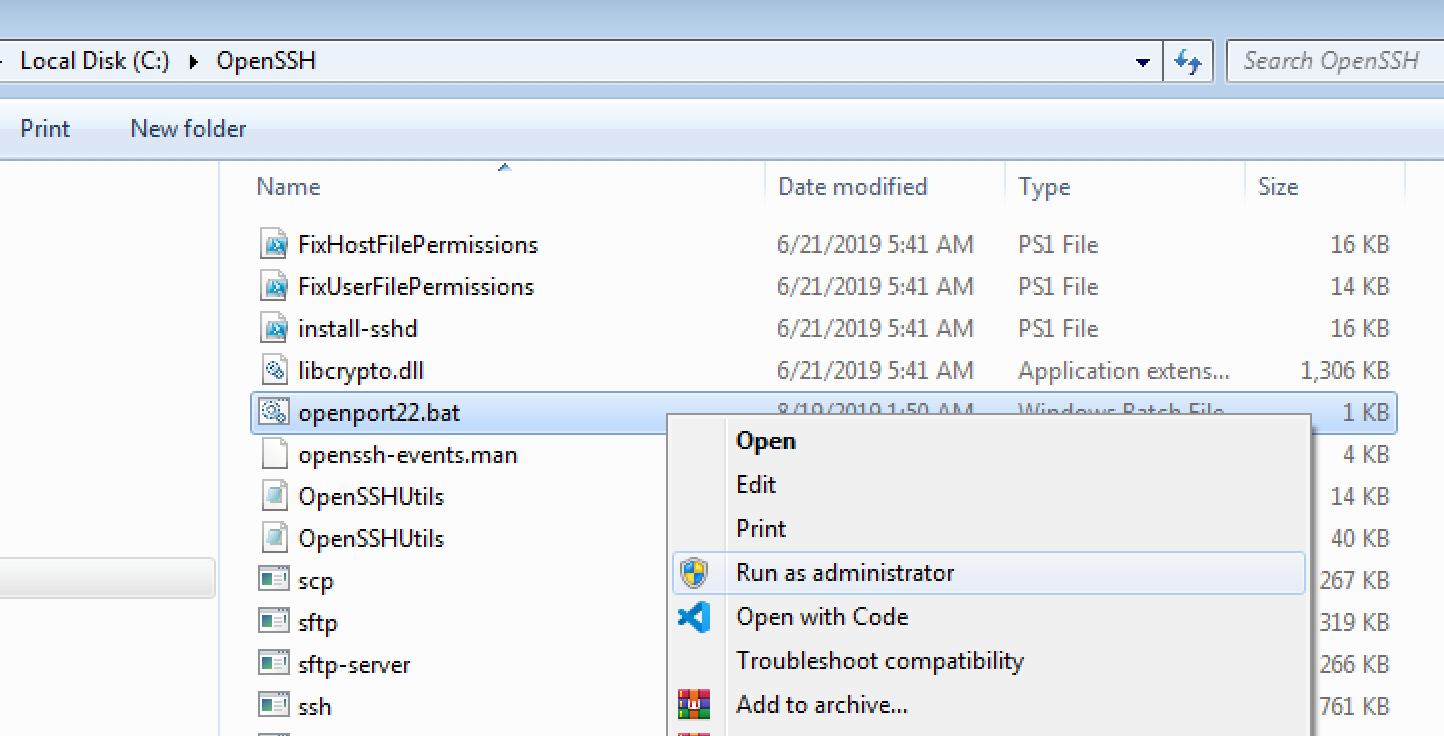




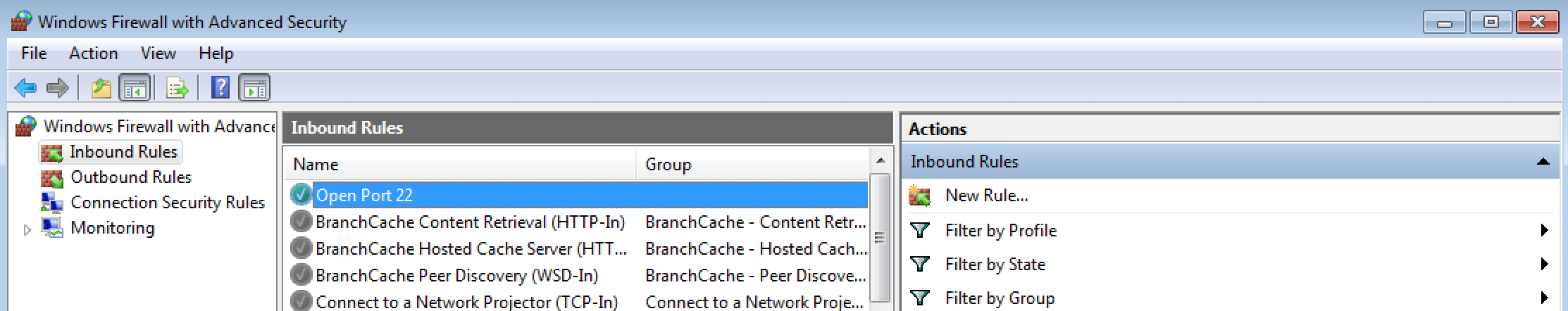
* Bước 3: Gõ services.msc vào Menu Start, chạy và mở Windows Services Manager, tìm đến OpenSSH SSH Server và dịch vụ OpenSSH Authetication Agent. Thiết lập cả hai dịch vụ có Startup type là Automatic (khởi chạy cùng hệ thống), cuối cùng hãy chạy 2 dịch vụ đó (nhấn phải chuột vào nó, chọn start)



* Bước 4: Bước 4 Mở cổng 22, vì SSH Server làm việc trên cổng 22 nên cần thiết lập Firewall cho phép kết nối đến cổng này. Để tự động tạo rule cho phép kết nối đến cổng 22 trong Windows Firewall, bạn có thể tải file .bat sau về Open Port 22 , lưu vào đĩa, ví dụ lưu tại C:\OpenSSH\openport22.bat, sau đó chạy file bat này với quyền Administrator.

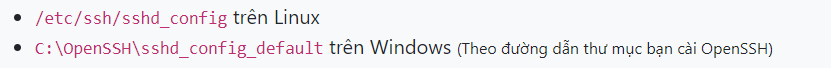


Sau khi chạy, nó sẽ tạo một Rule có tên Open Port 22 trong Windows Firewall là thành công



### 3.2.2. Cấu hình dịch vụ OpenSSH Server

Các cấu hình cho dịch vụ SSH trên Server giống nhau giữa Windows và Linux, các thiết lập thực hiện bằng việc chỉnh sửa file cấu hình, sau khi cập nhật thì khởi động lại dịch vụ SSH. Vị trí file cấu hình tại:



- Đổi cổng mặ định: nếu muốn đổi cổng kết nối SSH



- Chỉ cho phép một số User kết nối SSH: nếu chỉ muốn một số User được chỉ ra được phép kết nối SSH, ví dụ chỉ cho phép user1, user2 thì thêm vào



- Cấm một số User ngược lại với AllowUsers, muốn cấm User nào, ví dụ user3, user4 thì thêm vào



- Bật chế độ xác thực SSH Key



- Chỉ ra đường dẫn lưu Public key ví sụ lưu tại .ssh/authorized\_keys



- Tắt xác thực bằng Password nếu muốn tắt chế độ cho phép nhập Password để kết nối SSH thì thêm vào



- Không cho xác thực bằng password rỗng



- Không cho tài khoản rôt đăng nhập



Vậy là đã có dịch vụ SSH chạy ở Server - với cấu hình cơ bản, giờ đến lúc từ máy trạm kết nối vào Shell của Server bằng SSH tại Kết nối vào Server bằng SSH

### 3.2.3. Thực hiệ kết nối SSH đến Server

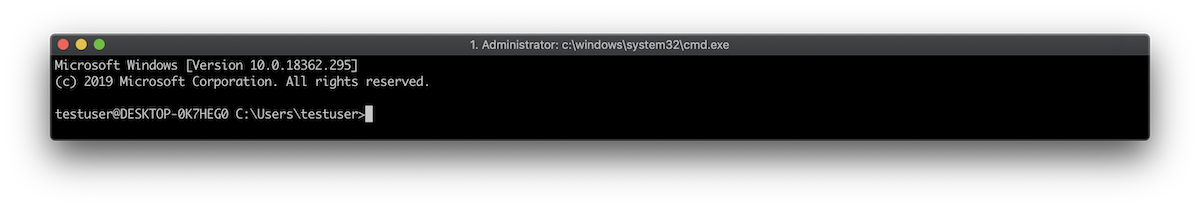


username@remoteserver nghĩa là tài khoản username tại máy chủ remoteserver. Lệnh này sẽ bắt đầu thực hiện kết nối SSH đến server có địa chỉ remoteserver, mặc định sẽ dùng cổng 22, nó sẽ hỏi password - bạn nhập password vào, nếu thành công - bạn sẽ ở dòng lệnh của máy Server.

Sau khi nhập lệnh, do lần đầu kết nối SSH đến IP này, nên ssh có hỏi về ECDSA key fingerprint- gõ yes, rồi nhập password theo yêu cầu, như hình dưới



Sau khi nhập password - kết nối thành công tới dòng lệnh của máy Windows (Remote Server)



Khi đã kết nối SSH tới Server Windows rồi, có thể gõ powershell.exe để chuyển sử dụng PowerShell, từ đó thực hiện các tác vụ trên Windows.

Tương tự, nếu Server là Linux CentOS hay Ubuntu thì lệnh kết nối ssh như trên sẽ giúp bạn vào thẳng terminal của máy Server - từ đó thi hành các lệnh của Server

Nếu cổng kết nối khác 22 ví dụ cổng 2222 thì cho vào tham số -p 2222



Ngoài ra để hiện một số thông tin thì cho vào tham số -v, hiện thị thông tin thật chi tiết quá trình kết nối (giúp gõ rối nếu lỗi) thì thâm vào tham số -vvv, nếu muốn nén dữ liệu khi truyền qua SSH thì thêm vào tham số -C



### 3.2.4. File cấu hình SSH Client

ssh (ssh client) khi chạy, thực hiện các kết nối - nó sẽ tìm file cấu hình tại đường dẫn ~/.ssh/config, nếu có file này nó dùng file đó thiết lập các thông tin bổ sung kết nối.

Trong đường dẫn file trên, ~ cho biết đó là thư mục của User hiện tại, như vậy mỗi user của máy có thể có file config riêng. Như vậy, nếu file config cho user abc trên hệ thống chưa có, thì cần tạo file này, nằm đúng ở đường dẫn như trên.

\* Chú ý: trên Windows, để tạo được thư mục .ssh thì bạn dùng PowerShell (không dùng Cmd), và gõ lệnh mkdir để tạo thư mục



# **Chương 4: Hệ thống quản lí mã nguồn FTP và SFTP**

## 4.1. Giới thiệu tổng quan về FTP

### 4.1.1. Sơ lược về FTP



Giao thức FTP là từ viết tắt bởi File Transfer Protocol. Là một trong những giao thức phát triển và được sử dụng rất phổ biến vào những năm 1970. Mục đích của FTP là dùng để trao đổi tệp, dữ liệu giữa máy khách và máy chủ. Và tệp sẽ được chuyển gián tiếp qua mạng máy tính. Có một điều đặc biệt mà chắc hẳn ít người biết đến. Đó là giao thức này có 2 kênh riêng biệt dùng để trao đổi các tệp giữa các máy. Đó là:

- Kênh điều khiển

- Kênh dữ liệu

### 4.1.2. Ưu điểm đặc biệt của FTP

Để biết được lý do tại sao giao thức FTP được sử dụng phổ biến, bạn cần biết được những ưu điểm cơ bản sau:

- Là giao thức có thời gian hoạt động lâu năm nên đa phần mọi người đều không còn xa lạ với nó.

- Với FTP, bạn có thể chuyển nhiều tệp khác nhau vào cùng một thời điểm. Và nó sẽ tiếp tục quá trình chuyển tệp nếu bị mất kết nối. Có thể thêm tệp chờ chuyển vào khung chờ download hay upload và để lịch truyền.

- Chức năng của các script sẽ giúp các tệp tin được chuyển đi tự động.

- Các máy khách FTP có thể đồng bộ hoá được các tệp tin.

### 4.1.3. Nhược điểm của FTP

Bên cạnh những ưu điểm đặc biệt hữu ích đó, bạn cũng cần biết một số nhược điểm sau:

- Tính bảo mật kém

- Tin tặc có thể truy cập những thông tin cá nhân của bạn một cách dễ dàng.

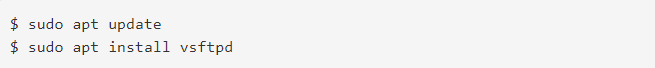
- Nếu không cẩn thận, máy chủ có thể bị lừa và sau đó, dữ liệu sẽ bị gửi đến máy tính nào đó không xác định.

- Đặc biệt, để kênh dữ liệu được tạo, giao thức FTP thường yêu cầu tập hợp port. Nhưng vấn đề này lại khá nan giải.

## 4.2 Kiến thức cơ bản về FTP

### 4.2.1. Cài đặt và cấu hình VSFTP làm FTP Server trên UBUNTU 20.04

- Cài đặt vsftp

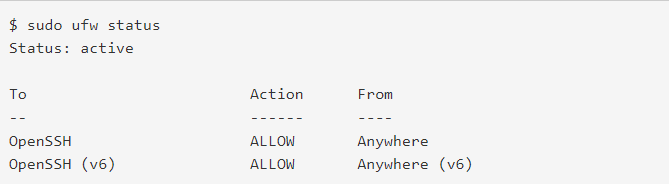


Tiếp theo, chúng ta cần sao lưu lại cài đặt vsfptd cũ nếu bạn muốn cài đặt vsftpd mới ghi đè lên version cũ:

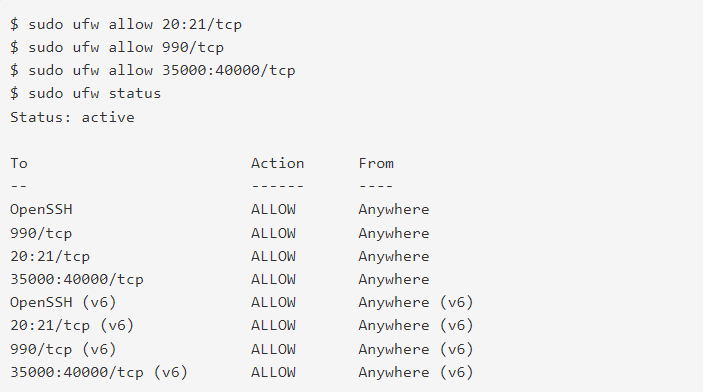


- Cấu hình tưởng lửa cho vsftpd

Bạn cần phải cấu hình UFW (công cụ tường lửa trên Ubuntu 20.04) để mở port cho FTP nên trước hết bạn hãy kiểm tra trạng thái hiện tại:



Thực hiện mở các port 20 (FTP command port), 21 (FTP data port), 990 (TLS FTP data port) và dải port 35000-40000:



- Cấu hình vsftpd

Tiếp theo bạn cần cấu hình vsftpd bằng cách mở và chỉnh sửa file cấu hình:

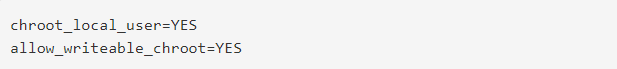


Để giới hạn chỉ cho người dùng nội bộ truy cập vào FTP thì bạn thêm dòng cấu hình sau:

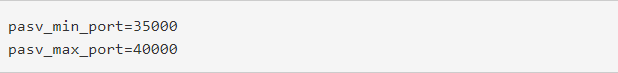
Bạn cần cho phép quyền ghi để có thể kích hoạt chức năng upload trên FTP Server. Để làm điều đó, uncomment dòng sau:



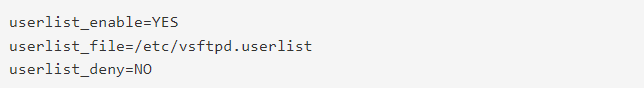
Tiếp theo cần giới hạn người dùng chỉ có thể thao tác trên thư mục cụ thể. Để làm điều đó, bạn cần uncomment dòng sau:



vsftpd có thể sử dụng bất kì port nào cho các kết nối passive FTP. Vi vậy nên chúng ta thực hiện cấu hình minimum port và maximum port với dòng cấu hình sau:



Cuối cùng là để giới hạn những người dùng nào có thể đăng nhập vào FTP Server, thêm đoạn cấu hình sau:



### 4.2.2 Cấu hình thư mục người dùng

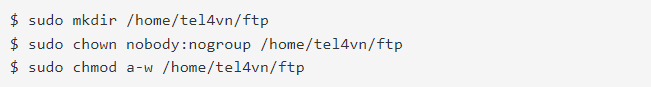
**-** Để thêm người dùng mới vào FTP Server, trong bài viết mình sẽ thực hiện thêm mới người dùng. Đầu tiên là tạo người dùng mới:



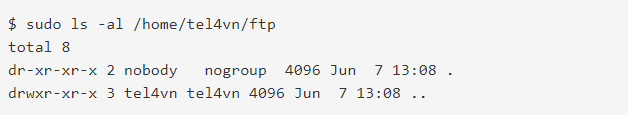
Tiếp theo bạn cần thêm người dùng mới tạo vào danh sách người dùng của FTP:



Sau khi thêm vào danh sách, bạn hãy tạo thư mục cho người dùng đó:



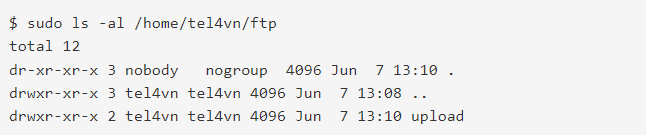
Sau khi tạo xong thư mục, kiểm tra lại quyền thư mục như sau:



Tiếp theo, bạn cần tạo thư mục có quyền write để có thể lưu các file tải lên:



Lúc đó, thư mục dành cho việc tải lên sẽ có quyền như bên dưới:

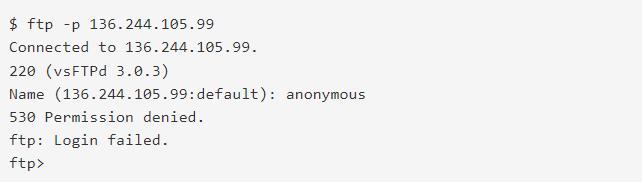


Để tiện cho việc test thử, bạn nên tạo một file test.txt ttrong thư mục upload:

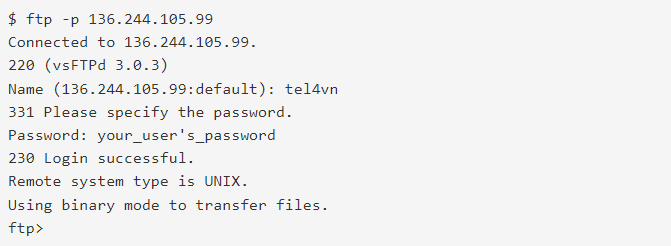


### 4.2.3 Kiểm tra kết nối FTP

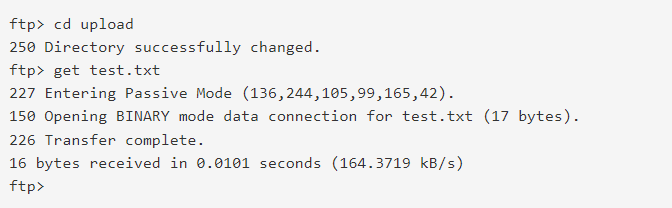
Sau khi cấu hình xong, bạn hãy thử kiểm tra kết nối FTP như sau:



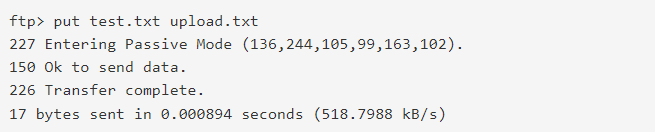
Như bên trên, người dùng bất kì không thể đăng nhập vào FTP Server. Ta thử lại với người dùng mới tạo:



Với người dùng được cấu hình thì bạn có thể đăng nhập thành công. Thực hiện tải xuống file test.txt vừa tạo:



Tiếp tục thực hiện đổi tên file test.txt để xác nhận người dùng được tạo có quyền write:



Cuối cùng đóng kết nối:



**TÀI LIỆU THAM KHẢO**

1. <https://backlog.com/git-tutorial/vn/intro/intro1_1.html>
2. <https://cafedev.vn/mien-phi-100-series-tu-hoc-git-tu-co-ban-toi-nang-cao/>
3. <https://git-scm.com/downloads/guis/>