GIT

Distributed revision control and source code management (SCM)

GIT

Introduction to GIT

Overview of Git

- Git is a distributed revision control and source code management (SCM) system with an emphasis on speed, data integrity and support for distributed, non-linear workflows.
  - Git was initially designed and developed by Linus Torvalds for Linux kernel development in 2005, and has since become the most widely adopted version control system for software development.
- As with most other distributed revision control systems, and unlike most client–server systems, every Git working directory is a full-fledged repository with complete history and full version-tracking capabilities, independent of network access or a central server. Like the Linux kernel, Git is free software distributed under the terms of the GNU General Public License version 2.

Getting Started - Installing Git

- The most official build is available for download on the Git website.
- Just go to *http://git-scm.com/download/win* and the download will start automatically.
- Note that this is a project called Git for Windows, which is separate from Git itself; for more information on it, go to *https://git-for-windows.github.io/*.
- Another easy way to get Git installed is by installing GitHub for Windows. The installer includes a command line version of Git as well as the GUI. It also works well with Powershell, and sets up solid credential caching and sane CRLF settings. You can download this from the GitHub for Windows website, at*http://windows.github.com*.

First-Time Git Setup

- Git comes with a tool called git config that lets you get and set configuration variables that control all aspects of how Git looks and operates.

- These variables can be stored in three different places:
  - /etc/gitconfig file: Contains values for every user on the system and all their repositories. If you pass the option --system to git config, it reads and writes from this file specifically.
  - ~/.gitconfig or ~/.config/git/config file: Specific to your user. You can make Git read and write to this file specifically by passing the --global option.
  - config file in the Git directory (that is, .git/config) of whatever repository you're currently using: Specific to that single repository.

- Each level overrides values in the previous level, so values in .git/config trump those in/etc/gitconfig.

- On Windows systems, Git looks for the .gitconfig file in the $HOME directory (C:\Users\$USER for most people). It also still looks for /etc/gitconfig. If you are using version 2.x or later of Git for Windows, there is also a system-level config file at C:\Documents and Settings\All Users\Application Data\Git\config .

First-Time Git Setup

Your Identity

- $ git config --global user.name “Anil"
- $ git config --global user.email anil@example.com

Your Editor

- $ git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -nosession"

Checking Your Settings

- $ git config –list
- You may see keys more than once, because Git reads the same key from different files (/etc/gitconfig and ~/.gitconfig, for example).
- You can also check what Git thinks a specific key’s value is by typing git config <key>:
- $ git config user.name

Getting Started - Getting Help

Getting Help

- **If you ever need help while using Git, there are three ways to get the manual page (manpage) help for any of the Git commands:**

  $ git help <verb>

  $ git <verb> --help

  $ man git-<verb>

- **For example, you can get the manpage help for the config command by running**

  $ git help config

GIT

Repositories and branches

Repositories and Branches

- Repositories:
  - It is a collection of refs together with an object database containing all objects which are reachable from the refs, possibly accompanied by meta data from one or more porcelains. A repository can share an object database with other repositories via alternates mechanism.

Git repositories

- Creating repositories :
  - at default location
    - git init
  - at particular location
    - git init c:/testGIT
  - Bare repository
    - git init --bare

Git Basics - Getting a Git Repository

- You can get a Git project using two main approaches.
  - The first takes an existing project or directory and imports it into Git.
  - The second clones an existing Git repository from another server.

Git Basics - Getting a Git Repository

- Initializing a Repository in an Existing Directory
  - If you're starting to track an existing project in Git, you need to go to the project's directory and type:

    $ git init
  - This creates a new subdirectory named .git that contains all of your necessary repository files – a Git repository skeleton.
- You can accomplish start version-controlling existing files that with a few git add commands that specify the files you want to track, followed by a git commit:

  $ git add *.c

  $ git add LICENSE

  $ git commit -m 'initial project version'

Git Basics - Getting a Git Repository

- Cloning an Existing Repository
  - If you want to get a copy of an existing Git repository – for example, a project you'd like to contribute to – the command you need is git clone [url] .

    $ git clone https://github.com/libgit2/libgit2

  - This creates a directory named "libgit2", initializes a .git directory inside it, pulls down all the data for that repository, and checks out a working copy of the latest version. If you go into the new libgit2 directory, you'll see the project files in there, ready to be worked on or used.

Recording Changes to the Repository

- Checking the Status of Your Files

```
$ git status
```

- Let’s say you add a new file to your project, a simple README file. If the file didn’t exist before, and you run git status, you see your untracked file like so:

```
$ echo 'My Project' > README
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  README
```

Recording Changes to the Repository

- Tracking New Files
  - In order to begin tracking a new file, you use the command git add. To begin tracking the README file, you can run this:

    $ git add README

  - If you run your status command again, you can see that your README file is now tracked and staged to be committed:

    $ git status
    On branch master
    Your branch is up-to-date with 'origin/master'.
    Changes to be committed:
     (use "git reset HEAD <file>..." to unstage)

     new file: README

Recording Changes to the Repository

- Staging Modified Files
  - Let's change a file that was already tracked. If you change a previously tracked file (say CONTRIBUTING.md) and then run your git status command again,

    $ git status

  - The CONTRIBUTING.md file appears under a section named "Changes not staged for commit" – which means that a file that is tracked has been modified in the working directory but not yet staged. To stage it, run the git add command.

    $ git add CONTRIBUTING.md

    $ git status

Working with Remotes

Showing Your Remotes

- To see which remote servers you have configured, you can run the git remote command.
- It lists the shortnames of each remote handle you've specified. If you've cloned your repository, you should at least see origin – that is the default name Git gives to the server you cloned from.

- You can also specify -v, which shows you the URLs that Git has stored for the shortname to be used when reading and writing to that remote.

Adding Remote Repositories

```
$ git remote add pb https://github.com/paulboone/ticgit
$ git remote -v
```

Working with Remotes

Pushing to Your Remotes

- If you want to push your master branch to your origin server (again, cloning generally sets up both of those names for you automatically), then you can run this to push any commits you've done back up to the server:

```
$ git push origin master
```

- This command works only if you cloned from a server to which you have write access and if nobody has pushed in the meantime. If you and someone else clone at the same time and they push upstream and then you push upstream, your push will rightly be rejected. You'll have to fetch their work first and incorporate it into yours before you'll be allowed to push.

Working with Remotes

- Inspecting a Remote
  - If you want to see more information about a particular remote, you can use the git remote show [remote-name] command. If you run this command with a particular shortname, such as origin, you get something like this:

```
$ git remote show origin
```

- Removing and Renaming Remotes
  - You can run git remote rename to change a remote's shortname. For instance, if you want to rename pb to paul, you can do so with git remote rename:

```
$ git remote rename pb paul
$ git remote
origin
paul
```

Git Pull

- git pull pull down from a remote whatever you ask (so, whatever trunk you're asking for) and instantly merge it into the branch you're in when you make the request.
- Pull is a high-level request that runs 'fetch' then a 'merge' by default, or a rebase with '–rebase'. You could do without it, it's just a convenience.

```
%> git checkout localBranch
%> git pull origin master
%> git branch
master
* localBranch
```

- The above will merge the remote "master" branch into the local "localBranch".

Git clone

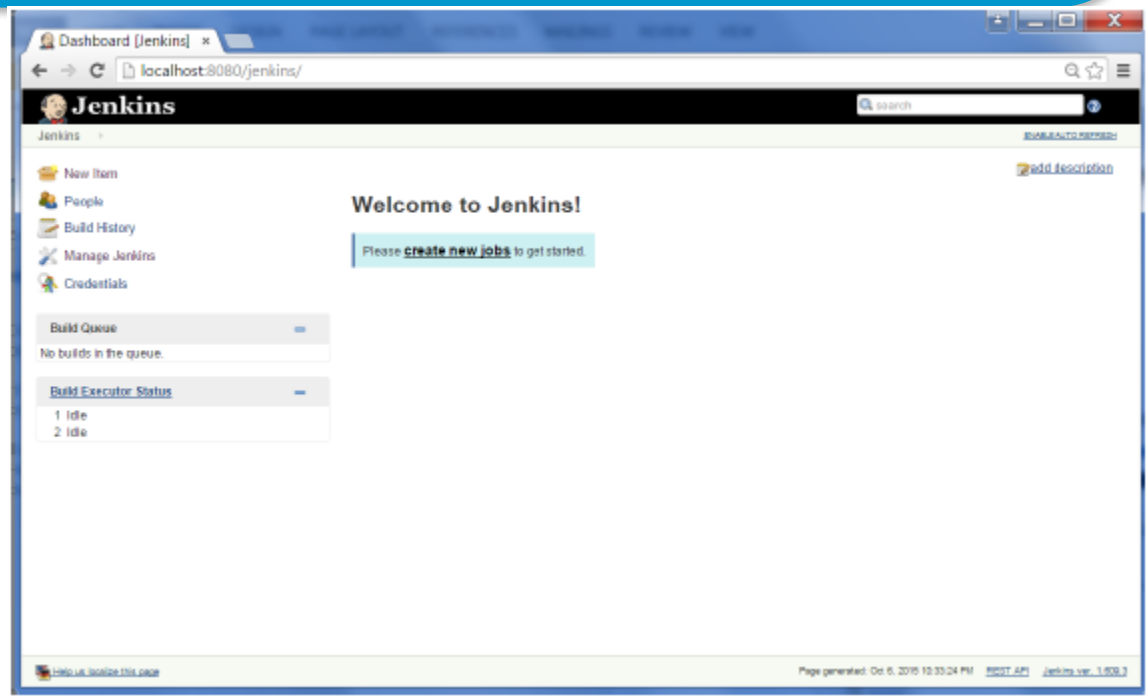
- Git clone will clone a repo int a newly created directory.

```
%> cd newfolder
%> git clone git@github.com:whatever/something.git
%> git branch
 * master
remoteBranch
```

- Git clone additionally creates a remote called ‘origin’ for the repo cloned from, sets up a local branch based on the remote’s active branch (generally master), and creates remote-tracking branches for all the branches in the repo

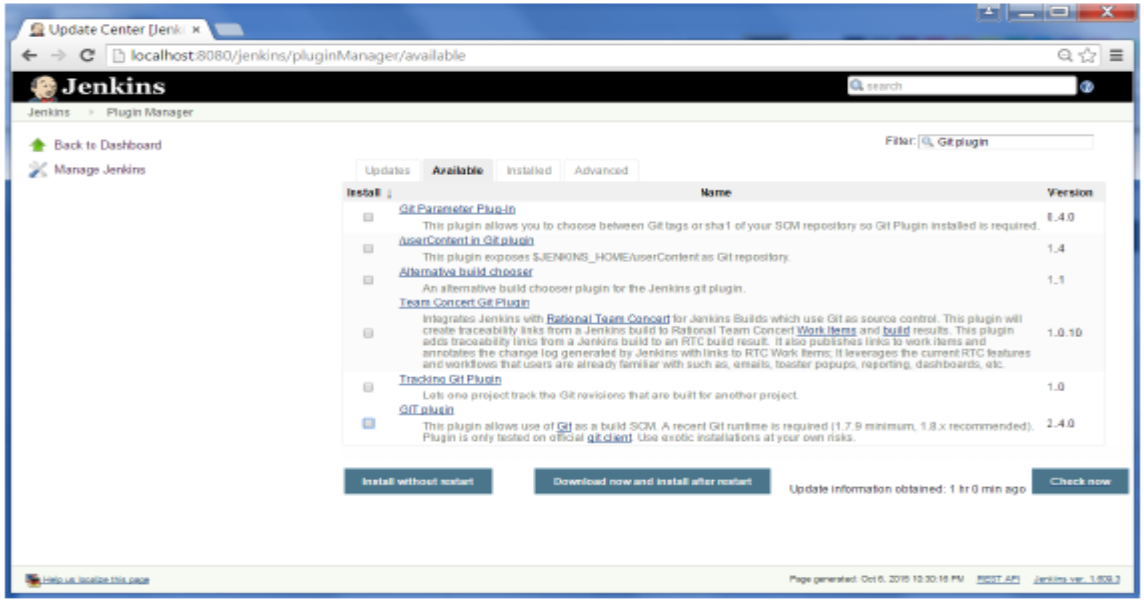
Jenkins - Git Setup

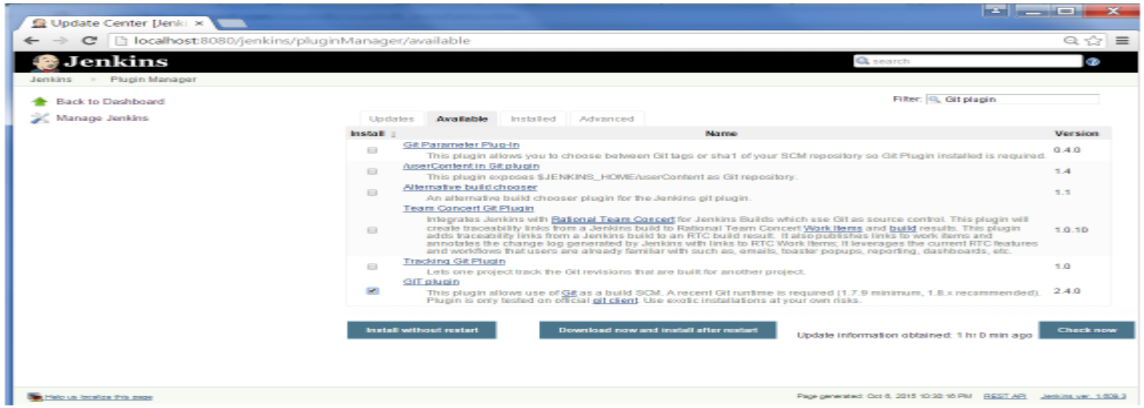
- In your Jenkins Dashboard (Home screen), click the Manage Jenkins option on the left hand side.
- In the next screen, click the ‘Manage Plugins’ option.



Jenkins - Git Setup

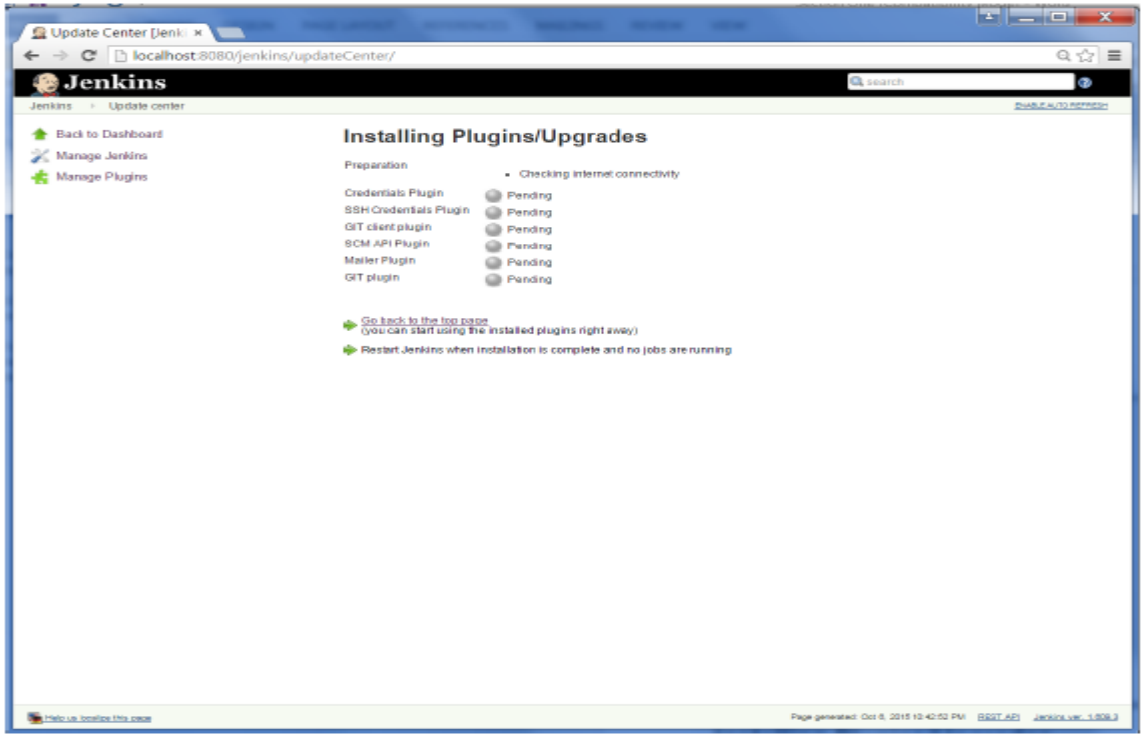
- In the next screen, click the Available tab. This tab will give a list of plugins which are available for downloading. In the ‘Filter’ tab type ‘Git plugin’.
- The list will then be filtered. Check the Git Plugin option and click on the button ‘Install without restart’.





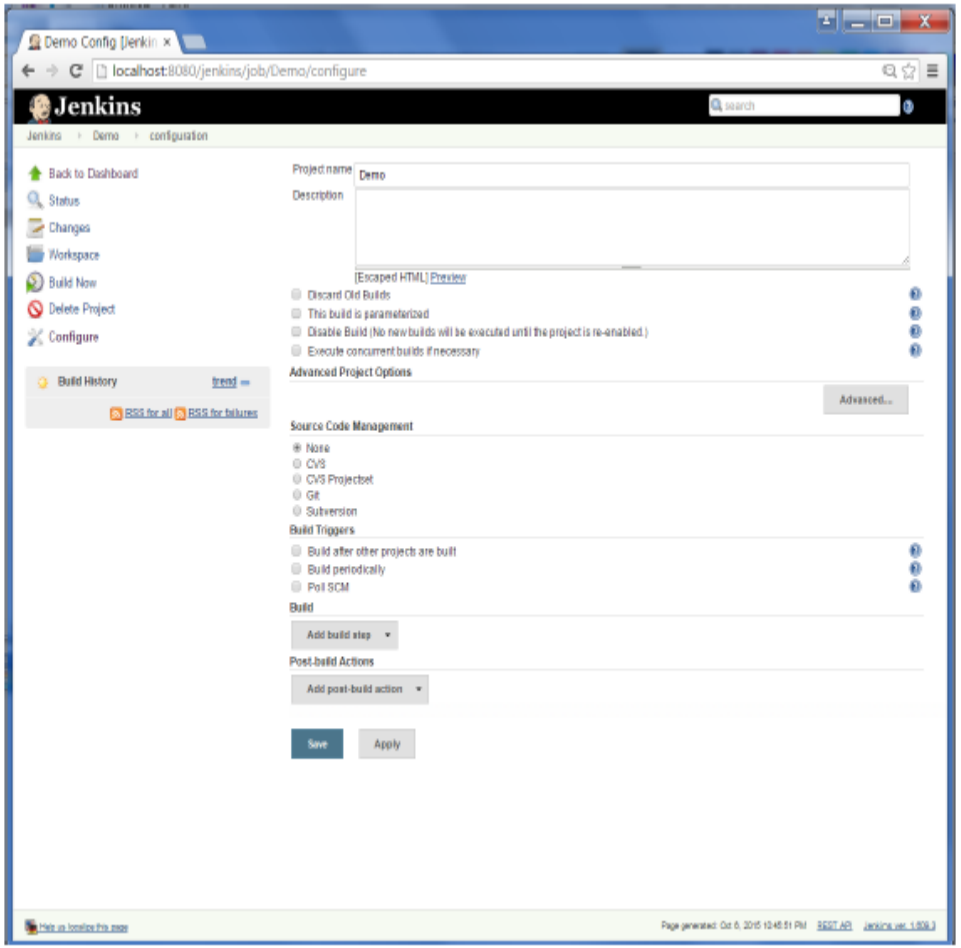
Jenkins - Git Setup

- The installation will then begin and the screen will be refreshed to show the status of the download..
- Once all installations are complete, restart Jenkins



Jenkins - Git Setup

- After Jenkins is restarted, Git will be available as an option whilst configuring jobs. To verify, click on New Item in the menu options for Jenkins. Then enter a name for a job, in the following case, the name entered is ‘Demo’. Select ‘Freestyle project’ as the item type. Click the Ok button.Once all installations are complete, restart Jenkins



Thanks