GIT Basic Commands

VISHWANATH M S

VISHWACLOUDLAB.COM

Agenda

- Initial Git configuration
- Steps for Committing Files
- Clone the repo
- Undo Changes to file
- Viewing changes
- Branching and merging
- Git Commands Cheat Sheet

Initial Git configuration

- Set the name and email for Git to use when you commit:
  - **git config --global user.name “Vishwa Cloud"**
  - **git config --global user.email vishwacloudlab@gmail.com**
  - You can call **git config –list** to verify these are set.
- Set the editor that is used for writing commit messages:
  - **git config --global core.editor nano**
    - (it is vim by default)

Steps for Committing Files

- **Step 1 – Git Init**
- **Step 2 – Git Status**
- **Step 3 – Git Add**
- **Step 4 – Git Commit**
- **Step 5 – Git Ignore**

Step 1 - Git Init

- To store a directory under version control you need to create a repository. With Git you initialize a repository in the top-level directory for a project.
- As this is a new project, a new repository needs to be created.
- Use the **”git init”** command to create a repository.

ProTip

- After initializing a repository, a new hidden subdirectory called ***.git*** is created.
- This subdirectory contains the metadata that Git uses to store it's information.

Step 2 - Git Status

- When a directory is part of a repository it is called a ***Working Directory***.
- As you're working on a project, all changes are made in this working directory.
- You can view which files have changed between your working directory and what's been previously committed to the repository using the command **"git status"**
- **The output of this command is called the "working tree status"**

Protip

- All files are "**untracked**" by Git until it's been told otherwise. The details of how is covered in the next step

Output – GIT STATUS

```
Vishwanath@vishwa MINGW64 /d/repository/b27-git-repo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        File1.txt

nothing added to commit but untracked files present (use "git ad

Vishwanath@vishwa MINGW64 /d/repository/b27-git-repo (master)
$
```

Step 3 - Git Add

- To save, or commit, files into your Git repository you first need to add them to the ***staging area***.
- Git has three areas, a **working directory**, a **staging area** and the **repository** itself.
- One of the key approaches with Git is that commits are focused, small and frequent.
- Use the command **git add <file|directory>** to add to staging area.

  Or

- **git add . –** to add all the files in the folder.

Output

```
Vishwanath@vishwa MINGW64 /d/repository/b27-git-repo (master)
$ git add File1.txt

Vishwanath@vishwa MINGW64 /d/repository/b27-git-repo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   File1.txt


Vishwanath@vishwa MINGW64 /d/repository/b27-git-repo (master)
```

Step 4 - Git Commit

Once a file has been added to the staging area it needs to be committed to the repository.

The command **git commit -m "commit message“** moves files from staging to the repository

and records the below

- **time/date,**
- **author and**
- **a commit message**

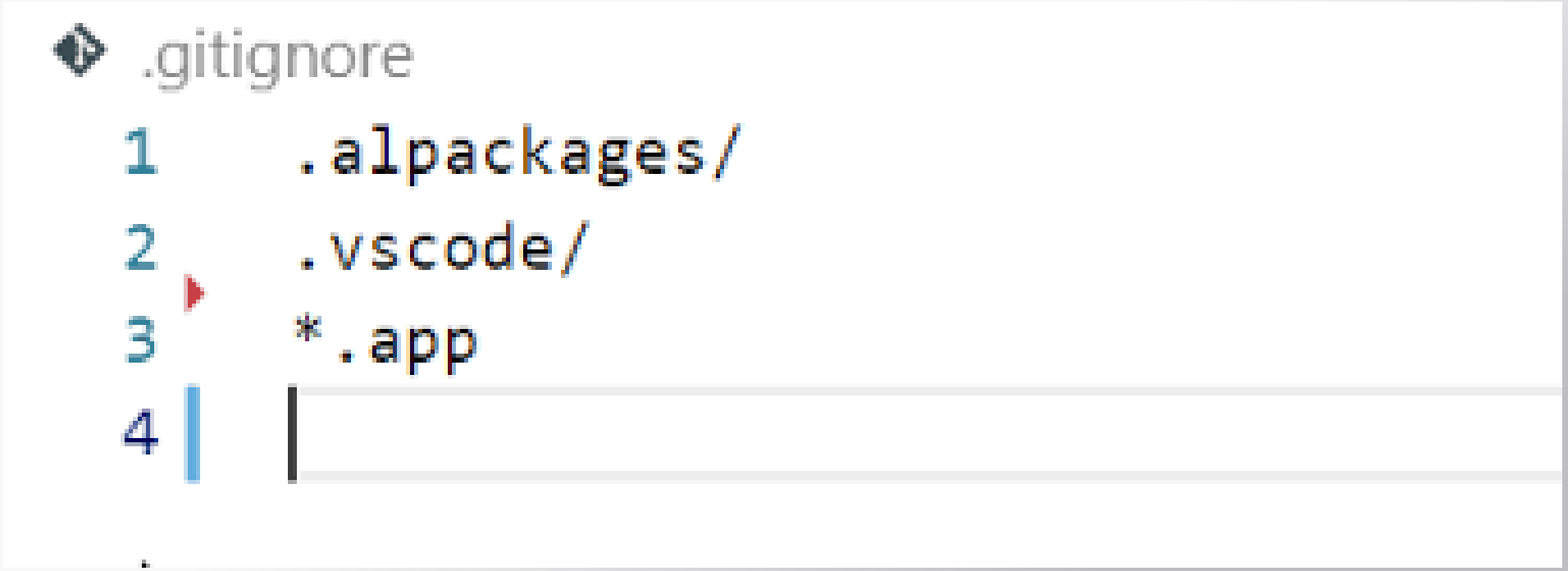
Step 5 - Git Ignore

- Git sees every file in your working copy.
- But some files may not be required to be commit like,
  - Build Architects
  - Machine generated files
  - Compiled code like, .class, .0, .pyc
  - Runtime files as, .log, .lock or .tmp

- Ignored file are tracked in a special file named **.gitignore**

Step 5 - Git Ignore

- Git Ignore Patterns
  - **.gitignore file in the root directory of the repository**
- Shared .gitignore files in your repo
  - **.gitignore file could be in each and every sub folders and in the root folder**



Step 5 - Git Ignore

- Personal Git Ignore Patterns
  - **.git/info/exclude file is used to ignore patterns in the local repo**
- Global Git Ignore rules
  - **.gitignore file would need to be declared globally for all the repos**

```
$ touch ~/.gitignore
$ git config --global core.excludesFile ~/.gitignore
```

- Ignoring a previously committed file

```
$ echo debug.log >> .gitignore
$ git rm --cached debug.log rm 'debug.log'
$ git commit -m "Start ignoring debug.log"
```

To clone a remote repo to your current directory

- **git clone** ***url*** **localDirectoryName**
  - This will create the given local directory, containing a working copy of the files from the repo, and a .git directory

Undo Changes to file

- To undo changes on a file before you have committed it:
  - **git reset HEAD --** *filename* (unstages the file)
  - **git checkout** -- *filename* (undoes your changes)
  - All these commands are acting on your local version of repo.

Viewing changes

- To see what is modified but unstaged:
  - **git diff**
- To see a list of staged changes:
  - **git diff --cached**
- To see a log of all changes in your local repo:
  - **git log** or **git log --oneline** (shorter version)
    - **1677b2d Edited first line of readme**
    - **258efa7 Added line to readme**
    - **0e52da7 Initial commit**
    - **git log -5 (to show only the 5 most recent updates), etc**

Branching and merging

Git uses branching heavily to switch between multiple tasks.

- To create a new local branch:
  - **git branch *name***
- To list all local branches: (* = current branch)
  - **git branch**
- To switch to a given local branch:
  - **git checkout master**
  - **git merge *branchname***

Git commands Cheat Sheet

Command	Description
git clone *url* [*dir*]	copy a Git repository so you can add to it
git add *file*	adds file contents to the staging area
git commit	records a snapshot of the staging area
git status	view the status of your files in the working directory and staging area
git diff	shows diff of what is staged and what is modified but unstaged
git help [command]	get help info about a particular command
git pull	fetch from a remote repo and try to merge into the current branch
git push	push your new branches and data to a remote repository
others: init, reset, branch, checkout, merge, log, tag, show	