



Getting Git

Chris Funk
Southeast Christian Church

#citrt #RefreshCache





What is Git?

Git is a distributed version control system.





What is Version Control?

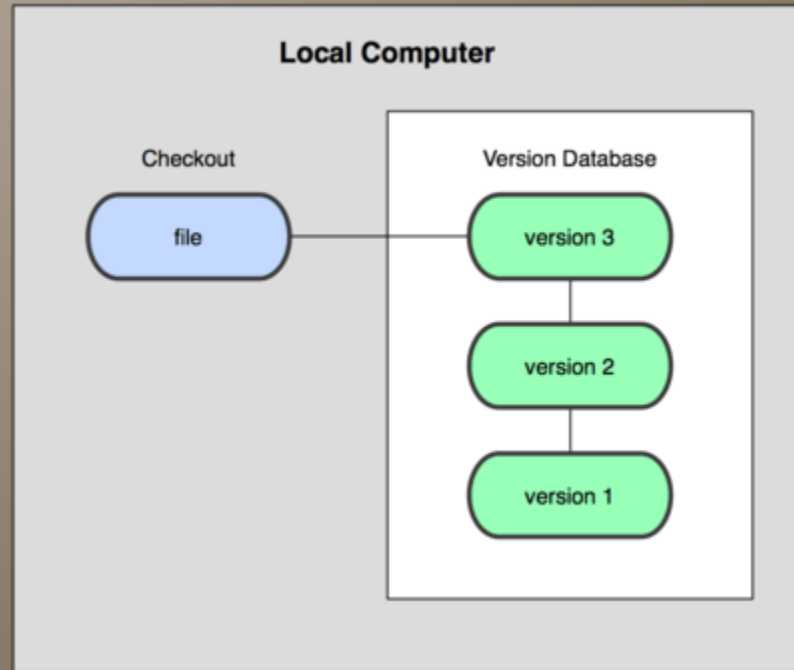
- A system that records changes to a file or a set of files over time, allowing you to recall specific versions later.
- Three types
 - Local
 - Centralized
 - Distributed





Localized

- Database local to a user or machine that keeps track of file changes/versions.
- Designed for single user.
- Single point of failure.
- Example rcs.

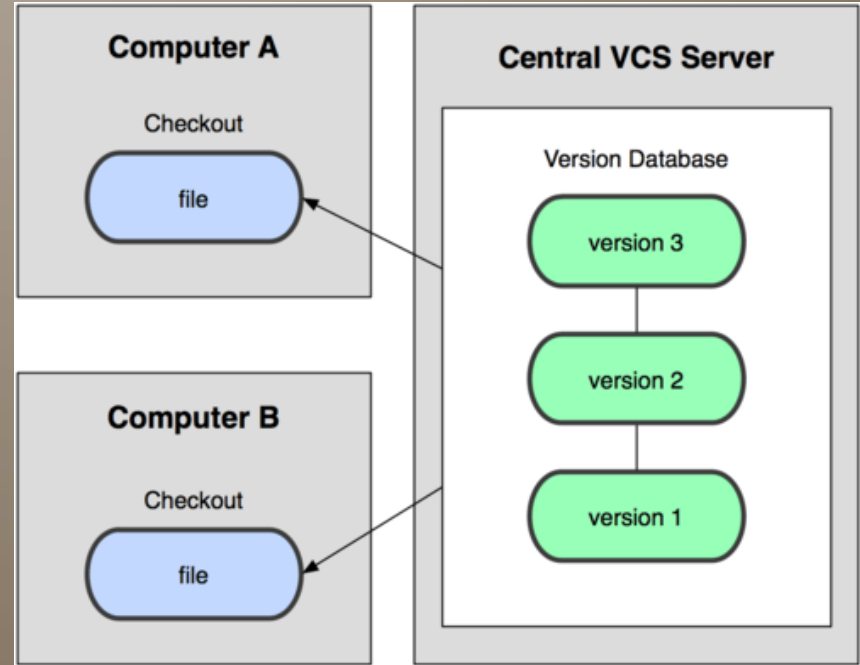


Source: <http://git-scm.com/book/en/Getting-Started-About-Version-Control>



Centralized

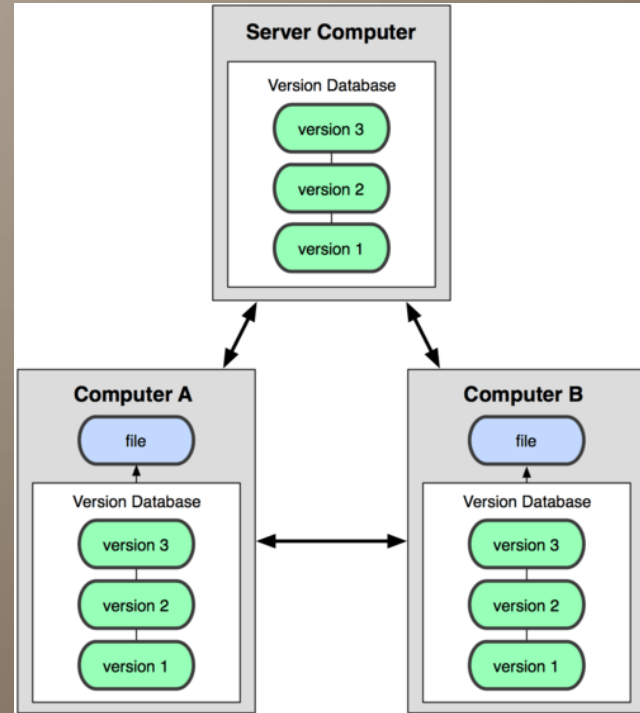
- Single server that contains the versioned files and a user checks out files from server.
- Downside – must be connected to server to checkout or commit.
- Examples: CVS, SVN, Visual Source Safe





Distributed

- Clients mirror/clone remote repository to their machines.
- Each “pull”/checkout is essentially a copy of the remote server.
- Clients merge their changes back the remote/origin
- Examples: Git, Mercurial



Source: <http://git-scm.com/book/en/Getting-Started-About-Version-Control>





Why do I need version control?

- Allows a user/team to see how a file or project has progressed over time
- Allows a team to easily collaborate on a project
 - Users can work on the same project or even file at the same time
- A user can rollback a file/project to a specific commit or date in time
- Features can be developed independently from each other and merged together.





Why get Git?

- Each user has a cloned copy of the repository
- Commits and checkouts are fast
- Can work disconnected from remote server
- Integrates with many Application Lifecycle Management (ALM) solutions
 - Github, BitBucket, Assemblia, Redmine (plugin)





Where to get it?

- Git (with Bash tools) – <http://git-scm.com>
- Gui Based tools - <http://git-scm.com/download/gui/>
 - SmartGit \$79 (commercial) / Free (non-commercial)
 - SourceTree Free
 - GitHub for Windows/Mac/Unix
- We will cover command line and SmartGit

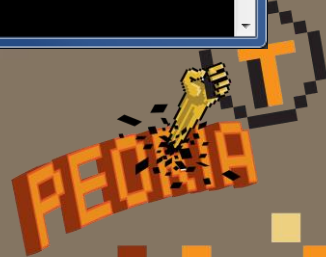




Creating a new repository

- Use git init to create a new repository
 - git init <repository name>

```
MINGW32:/c/_appdev/citrt
chrisf@FUNKMEISTER-PC /c/_appdev/citrt
$ git init ChrisF_CITRTDemo
Initialized empty Git repository in c:/_appdev/citrt/ChrisF_CITRTDemo/.git/
chrisf@FUNKMEISTER-PC /c/_appdev/citrt
$ _
```





What happened?

- A new (empty) repository was created with a single branch called “Master”
 - This is your main-line or Trunk branch
 - As a rule “Master” should always contain production code. (The code that is currently deployed).
 - It is highly discouraged to commit directly to the “Master” branch.





...there is an exception

- For your initial commit it is ok to commit your initial .gitignore and Readme and license files to master.
- .gitignore -> a file that contains files/directories that should be ignored by source control.
 - If you don't know what to ignore... there are templates.





Create Origin Repository

- Origin = the primary remote repository
- You can have an on premise Git server or use a hosting/ALM provider.
- Will use GitHub as our example
- GitHub Demo





Connecting to Origin

- Add origin bookmark
 - `git remote add <bookmark> <path>`
- Push master branch to origin
 - `git push -u <bookmark> <branch>`

```
MINGW32:/c/_appdev/citrt/ChrisF_CITRTSample

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/ChrisF_CITRTSample (master)
$ git remote add origin https://github.com/chrisfunk/CITRT_Sample.git

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/ChrisF_CITRTSample (master)
$ git push -u origin master
Username for 'https://github.com': chrisfunk
Password for 'https://chrisfunk@github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 205 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/chrisfunk/CITRT_Sample.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/ChrisF_CITRTSample (master)
$
```





Cloning an existing repository

- Clone = copy existing repository from Git server.
 - `git clone <url> <local path>`
 - Local path is optional; by default creates new folder with repository name.
- Depending on repository size it can take some time.

```
MINGW32:/c/_appdev/citrt

chrisf@FUNKMEISTER-PC /c/_appdev/citrt
$ git clone https://github.com/chrisfunk/CITRT_Sample2.git
Cloning into 'CITRT_Sample2'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
Checking connectivity... done.

chrisf@FUNKMEISTER-PC /c/_appdev/citrt
$ git clone https://github.com/chrisfunk/CITRT_Sample2.git csf1
Cloning into 'csf1'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
Checking connectivity... done.

chrisf@FUNKMEISTER-PC /c/_appdev/citrt
$
```





Staging and Committing Files

- `git status` – Lists files to be committed
- `git add <file>` – stages file for commit (all new files must be staged)
- `git reset <file>` cancels stage
- `git commit` commits staged/updated files

```
MINGW32:/c/_appdev/citrt/ChrisF_CITRTDemo

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/ChrisF_CITRTDemo (master)
$ git status
On branch master

Initial commit

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

        readme.md

nothing added to commit but untracked files present (use "git add" to track)

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/ChrisF_CITRTDemo (master)
$ git add readme.md

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/ChrisF_CITRTDemo (master)
$ git reset readme.md

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/ChrisF_CITRTDemo (master)
$ git add readme.md

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/ChrisF_CITRTDemo (master)
$ git commit -m "Initial Commit"
[master (root-commit) de8d5c0] Initial Commit
1 file changed, 1 insertion(+)
 create mode 100644 readme.md

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/ChrisF_CITRTDemo (master)
$
```





Branches

- Branch = A branch represents an independent line of development.
 - i.e. I am working on a check-in block and Maxim is working on refactoring a giving block
 - We can work independently on separate branches and then merge into a parent branch when complete.





Branching (cont)

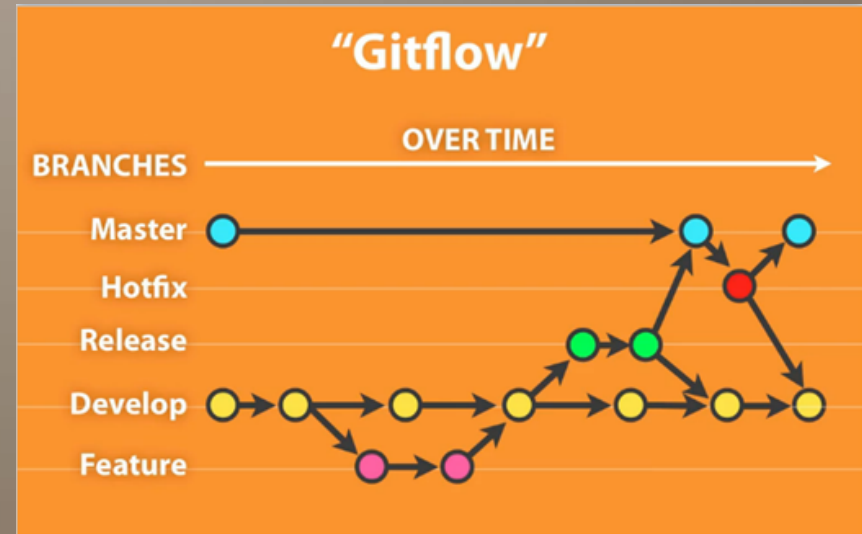
- There are many methodologies with branching, I recommend GitFlow.
- Two branches live in perpetuity (Master and Develop)
 - Master = what is currently in production.
 - Develop = In development/WIP branch
- Rest of branches have limited lifespan.





Branching - GitFlow

- Feature – feature that is in development (**feature-{identifier}**)
- Release – Release candidate (**rel-{identifier}**)
- Hotfix – Emergency update that can't wait for next release cycle (**hotfix-{identifier}**)





Working with Branches

- Create a new branch
 - `git branch <name>`
- Checkout – make head/working branch)
 - `git checkout <name>`
- Create & checkout
 - `git checkout <name> -b`

```
MINGW32:/c/_appdev/citrt/CITRT_Sample2

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/CITRT_Sample2 (master)
$ git branch feature-csf-test1

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/CITRT_Sample2 (master)
$ git branch
  feature-csf-test1
* master

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/CITRT_Sample2 (master)
$ git checkout feature-csf-test1
Switched to branch 'feature-csf-test1'

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/CITRT_Sample2 (feature-csf-test1)
$
```





Working with Branches

- Delete a branch (make sure head is a different branch)
 - `git branch -d <branch>`
- To delete from origin
 - `git push origin :<old branch name>`

```
MINGW32:/c/_appdev/citrt/CITRT_Sample2

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/CITRT_Sample2 (feature-csf-test)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/CITRT_Sample2 (master)
$ git branch -d feature-csf-test
Deleted branch feature-csf-test (was 77d225f).

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/CITRT_Sample2 (master)
$ git push origin :feature-csf-test
Username for 'https://github.com': chrisfunk
Password for 'https://chrisfunk@github.com':
To https://github.com/chrisfunk/CITRT_Sample2.git
 - [deleted]          feature-csf-test

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/CITRT_Sample2 (master)
$ git fetch -p
```





Merging

- Checkout the branch that you want to merge **to**
 - `git checkout <branch>`
- Perform the merge
 - `git merge <branch>`

```
MINGW32:/c/_appdev/citrt/CITRT_Sample2

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/CITRT_Sample2 (merge-example)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

chrisf@FUNKMEISTER-PC /c/_appdev/citrt/CITRT_Sample2 (master)
$ git merge merge-example
Updating 77d225f..5203033
Fast-forward
 SoutheastCampuses.txt | 15 ++++++
 1 file changed, 15 insertions(+)
 create mode 100644 SoutheastCampuses.txt
```





There's an easier way

- There are GUI based tools to work with Git.
- SmartGit demo





Using an ALM Tool

- There are several ALM tools on the market
 - ALM = Application Lifecycle Management
- They allow you to track issues/feature requests
- View commit history
- Some include Wiki functionality
- GitHub is the most popular for open source
- SECC is moving to Bitbucket b/c they allow free private repositories.





Learning tools

- Github provides a tutorial for learning Git
 - <https://help.github.com/articles/set-up-git/>
 - Training Kit - <https://training.github.com/kit/>
 - Try Git (online) – <https://try.github.io>
- Git command cheat sheet
 - <https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf>
- Git-SCM site <http://git-scm.org>

