Git - Version Control System

Ashik Salman

Beginner Workshop
.....
Backend Developer
Chillr, Backwater Technologies
Kochi, Kerala

September 20, 2016

What ?

Abstract

Git Workshop

- What is Git & Github.
- Git installation and creation of Github account.
- Git commands & Hands-on.
- Quick examples.

Introduction

What is "version control", and why should you care?

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Introduction

What is "version control", and why should you care?

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

- It allows you to revert files back to a previous state.
- Revert the entire project back to a previous state.
- Compare changes over time, see who last modified something that might be causing a problem.
- Who introduced an issue and when, and more.

Introduction

Distributed Version Control Systems

In a Distributed Version Control Systems, clients dont just check out the latest snapshot of the files: they fully mirror the repository. Thus if any server dies, and these systems were collaborating via it, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.

Introduction

Distributed Version Control Systems

In a Distributed Version Control Systems, clients dont just check out the latest snapshot of the files: they fully mirror the repository. Thus if any server dies, and these systems were collaborating via it, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.

- Git
- Mercurial
- Bazaar or Darcs

Getting started - Git

Installation

Note !

If you understand what Git is and the fundamentals of how it works, then using Git effectively will probably be much easier for you.

Customizations

Files: /etc/gitconfig, .git/config, /.config/git/config

Getting started - Git

Installation

Note !

If you understand what Git is and the fundamentals of how it works, then using Git effectively will probably be much easier for you.

Customizations

Files: /etc/gitconfig, .git/config, /.config/git/config

Samples

```
$ git config --global user.name "Ashik Salman"
$ git config --global user.email ashiksp@gmail.com
$ git config --global core.editor vim
$ git config --list , Explore more !

$ git config --global alias.co checkout
$ git config --global alias.unstage 'reset HEAD --'
```

Getting a Git Repository

Samples

```
$ git init
$ git add *.c
$ git add LICENSE
$ git commit -m 'initial project version'

$ git clone git@github.com:ashiksp/Workshops.git
$ git clone https://github.com/ashiksp/Workshops.git
```

File states

Recording Changes to the Repository (States)

- Committed, data is safely stored in your local database.
- Modified, changed the file but have not committed it to your database yet.
- Staged, marked a modified file in its current version to go into your next commit snapshot.

Samples

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean

$ git status -s  # Short Status
```

Ignoring Files

Need git to ignore specific file patterns - .gitignore

Samples

```
# no .a files
*.a

# but do track lib.a, even though you're ignoring
# .a files above
!lib.a

# ignore all files in the build/ directory
build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

# ignore all .pdf files in the doc/ directory
doc/**/*.pdf
```

Basic commands

Viewing Your Staged and Unstaged Changes

```
$ git status
$ git diff
$ git diff --cached | git diff --staged
```

Basic commands

Viewing Your Staged and Unstaged Changes

```
$ git status
$ git diff
$ git diff --cached | git diff --staged
```

Committing Your Changes

```
$ git commit
$ git commit -m "Commit Message"
$ git commit --amend # Undoing Things
```

Basic commands

Removing Files

```
$ git rm filename
$ git rm --cached README
$ git rm log/\*.log
```

Basic commands

Removing Files

```
$ git rm filename
$ git rm --cached README
$ git rm log/\*.log
```

Moving Files

```
$ git mv README.md README

$ mv README.md README
$ git rm README.md
$ git add README
```

Basic commands

Viewing the Commit History

```
$ git log
$ git log -p
$ git log --stat
$ git log --pretty=oneline
$ git log --pretty=format:"%h - %an, %ar : %s"
$ git log --graph
```

Basic commands

Viewing the Commit History

```
$ git log
$ git log -p
$ git log --stat
$ git log --pretty=oneline
$ git log --pretty=format:"%h - %an, %ar : %s"
$ git log --graph
```

Limiting Log Output

```
$ git log --since=2.weeks
$ git log --author 'ashiksp@gmail.com
$ git log -Sfunction_name'
```

Basic commands

Unstaging a Staged File

```
$ git reset HEAD
$ git reset HEAD CONTRIBUTING.md
```

Basic commands

Unstaging a Staged File

```
$ git reset HEAD
$ git reset HEAD CONTRIBUTING.md
```

Unmodifying a Modified File

```
$ git checkout .
$ git checkout -- CONTRIBUTING.md
```

Basic commands

Working with Remotes

```
$ git remote # Showing Your Remotes
$ git remote -v # Show with urls\
$ git remote show origin # Inspecting a Remote
$ Add remote reference:
$   git remote add remote-name https://url.git
$ Fetching and Pulling from Your Remotes:
$   git fetch [remote-name]
$ Pushing to Your Remotes:
$   git push origin master
$ Removing and Renaming Remotes:
$   git remote rename old-name new-name
$   git remote rm remote-name
```

Basic commands - Tagging

Types

Annotated

```
* stored as full objects in the Git database
* theyre checksummed
* contain the tagger name, email, and date
* have a tagging message
* git tag -a v1.4 -m "my version 1.4"
```

Basic commands - Tagging

Types

Annotated

* stored as full objects in the Git database
* theyre checksummed
* contain the tagger name, email, and date
* have a tagging message
* git tag -a v1.4 -m "my version 1.4"

Lightweight

* like a branch that doesnt change
* its just a pointer to a specific commit
* git tag v1.4

Basic commands - Tagging

Samples

```
$ git tag
$ git tag -l "v1.8.5*" # Pattern search
$ git push origin v1.5
$ git push origin --tags
$ git checkout -b version2 v2.0.0
```

Thank You

Questions?