Project Versions

latest

RTD Search

Go

Full-text doc search.

Table Of Contents

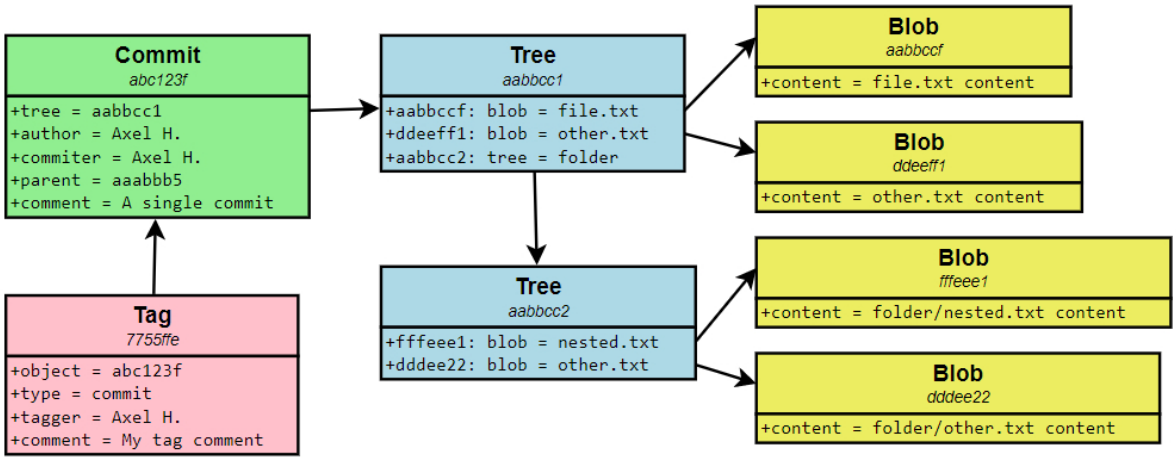
- GIT Illustrated Cheatsheet
  - GIT Model
    - Object & SHA1
    - Blob
    - Tree
    - Commit
    - Tags
    - Branch
  - Basic operations
    - Configuration
    - Staging
    - Commit
    - See your repository
    - Stash
    - Undoing
  - Branching and merging
    - Create a branch
    - Branche diverging
    - Merge
    - Rebase
    - Cherry Pick
  - Working with remote reposi
    - Add a remote
    - Diverging with remote
  - Sample workflows
  - ~/.gitconfig
  - Visual Tools
  - IDE Integration
  - References

This Page

Show Source

GIT Illustrated Cheatsheet

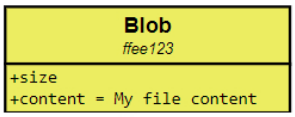
GIT Model



Object & SHA1

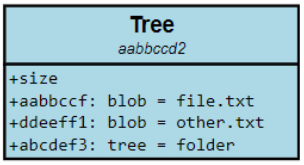
- Each object has a **type**, a **size** and a **content**
- Each object is identified by a 40-digit SHA1 hash of attributes
  - 6ff87c4664981e4397625791c8ea3bbb5f2279a3
- Each SHA1 can be shortened to the first digits
  - 6ff87c4664981e4397625
  - 6ff87c4
- Object type can be **blob**, **tree**, **commit** or **tag**

Blob



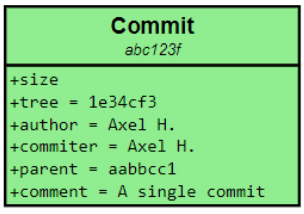
- Used to store file data
- Same content = same SHA1 = same blob
- See it as a file

Tree



- Associate names to blobs and other trees
- See it as a directory

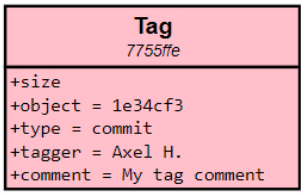
Commit



- Used to snapshot a tree state
- Has **tree**, **parent(s)**, **author**, **commiter** and **comment** attributes
- Not the same as SVN ones:
  - SVN stores diffs

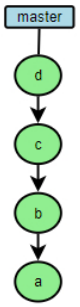
- SVN store diffs
- GIT store full state

Tags



- Reference an object
- Has **object**, **type**, **tagger** and **comment** attributes.
- Not used for lightweight tags
    - Simple pointer on a commit (like branches)

Branch



- A branch is simply a pointer to a certain commit.
- A branch is not aa GIT object (no SHA1)

Basic operations

Configuration

Set your name and email:

```
$ git config --global core.name "Me"
$ git config --global core.email "me@company.com"
```

Display your config:

```
$ git config --global
$ git config --local  # in a repository
```

Staging

Stage your changes:

```
# Add to index / stage
$ git add file.txt

# Add all modified and new files (tracked or not) to index
$ git add -A

# Partial staging
$ git add -p file.txt
```

Commit

Create a commit into the current branch:

```
# Commit from index
$ git commit

# Commit from tracked file list
$ git commit file1.txt file2.txt

# All modified tracked files
$ git commit -a

# Commit from pattern
$ git commit **/*.py
```

See your repository

See the current status:

```
$ git status
```

Retrieve your history:

```
# Log intégral
$ git log

# 5 dernier commits
$ git log -5

# Diff between two branches
$ git log origin/master..master
```

Stash

Discard your changes for later:

```
# Create a stash
$ git stash

# list stashes
$ git stash list

# Apply a stash
$ git stash apply

# Apply a stash and drop it
$ git stash pop

# Clear your stashes
$ git stash clear
```

Undoing

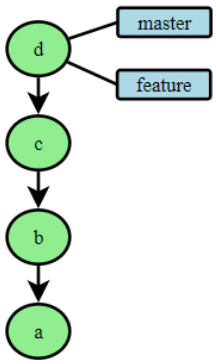
Revert back changes:

```
# Reverse commit
$ git revert {SHA1}

# Amend commit
$ git commit --amend

# Uncommit
$ git reset --mixed HEAD file

# Discard changes
$ git checkout file

# Reset branch to a given state
$ git reset --hard ref
```

Branching and merging

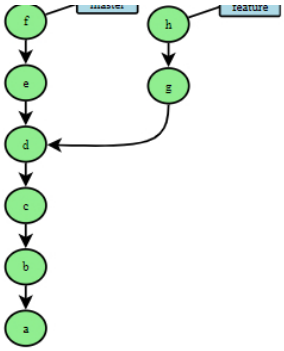
Create a branch

```
$ git branch feature     # Create the branch
$ git checkout feature   # Switch to the new branch
# or in a single command
$ git checkout -b feature
```



Branche diverging

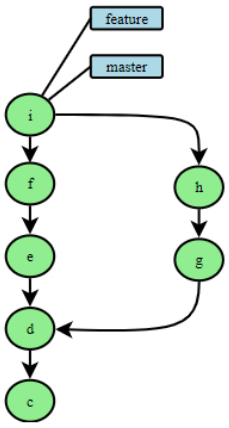
Branches diverge when they have different commits



Merge

Create a merge commit and keep your branch history:

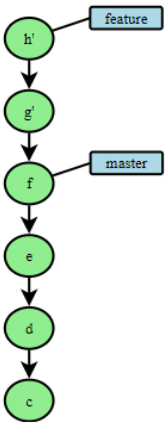
```
$ git merge feature
```



Rebase

Re-apply your commits and keeps your history linear:

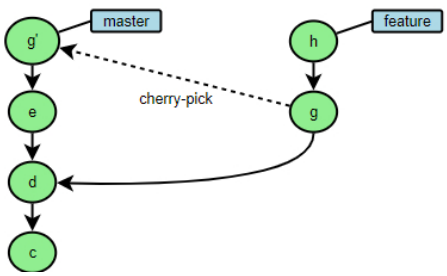
```
$ git rebase master
# or interactive version
$ git rebase -i master
```

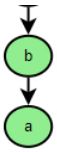


Cherry Pick

Pick a commit an apply it in the current branch as a new commit:

```
$ git cherry-pick {SHA1}
```

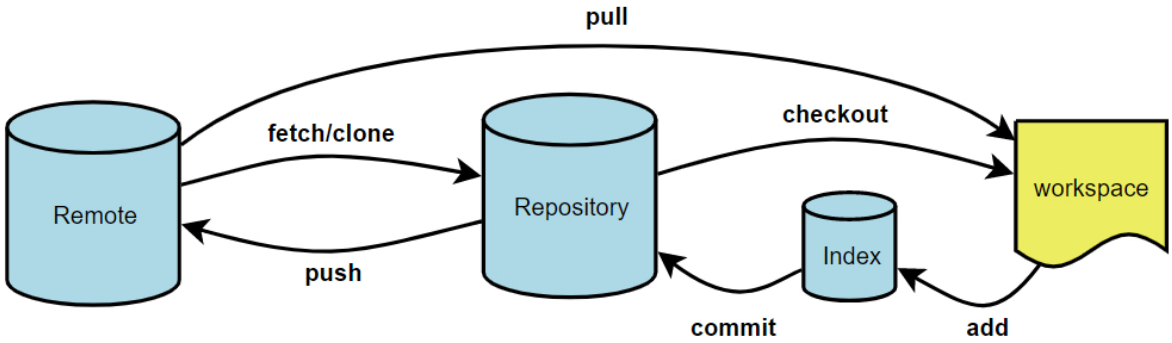




Working with remote repositories

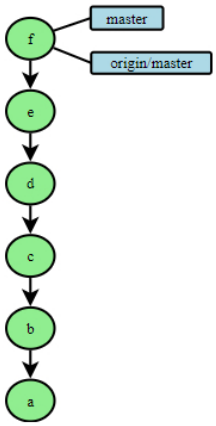
- It's only branches
- Repository synchronization operations:

```
$ git fetch
$ git push
$ git pull  # fetch + merge
$ git pull --rebase  # fetch + rebase
```



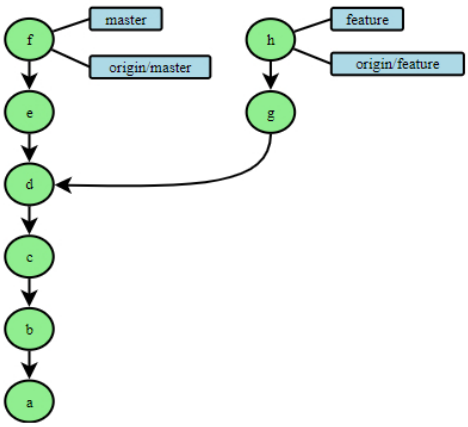
Add a remote

```
git remote add origin git://somewhere.git
git fetch
```



Diverging with remote

It's just more branches !



Sample workflows

TODO

TODO

~/.gitconfig

Tune your `~/.gitconfig` for comfort !

```
[alias]
   st = status
   ci = commit
   co = checkout
   br = branch
   amend = commit --amend
   rlog = log --pretty=oneline --abbrev-commit --graph --decorate
   plog = log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%
   unadd = reset --mixed HEAD
   uncommit = reset --soft HEAD^
[color]
   branch = auto
   diff = auto
   interactive = auto
   status = auto
```

Visual Tools

- Windows
    - Tortoise GIT
    - Git Extensions
- Mac OSX
    - GitX
    - Tower
- Linux
    - gitg
    - Giggle
- Multiplaform
    - SmartGit

IDE Integration

- *Eclipse*: EGit
- *IntelliJ*: Version Control Systems Integration
- *netbean*: GIT Support

References

- Git Community Book
- Pro GIT
- Github documentation