Git Essentials [Intro]

Haggai Philip Zagury,
Q1 2013

whoami

Haggai Philip Zagury
CM / DevOps Engineer
Over 5 years of CM/ALM/DevOps expertise

"I am a member of Tikal's ALM group. With over 12 members, we meet, share, contribute and code together on a bi-weekly basis. "

We help companies build, deliver,
deploy, manage and optimize their products.

ALM	JAVA	JS	.NET	RoR

"Today we are SURE that we made the right decision, choosing Tikal"
Guy Ben-Porat - Development Manager "ExLibris"

Tikal by Numbers

1600+ Community Members	150+ Blog Posts Last Year	460+ Meet up Members
12+ Years old	90+ Tikal's Experts Team	100+ Projects Last Year

"Actions speak louder than words"
Tikal's motto

Agenda

- Some history ...
- Dvcs .vs Cvcs
- Installing Git
- Everyday Git Workflow
- Git Internals
- Branching & Tagging
- Merging
- Remotes
- Extra's [on a Free time basis]

HISTORY

History

> "I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git." – Linus

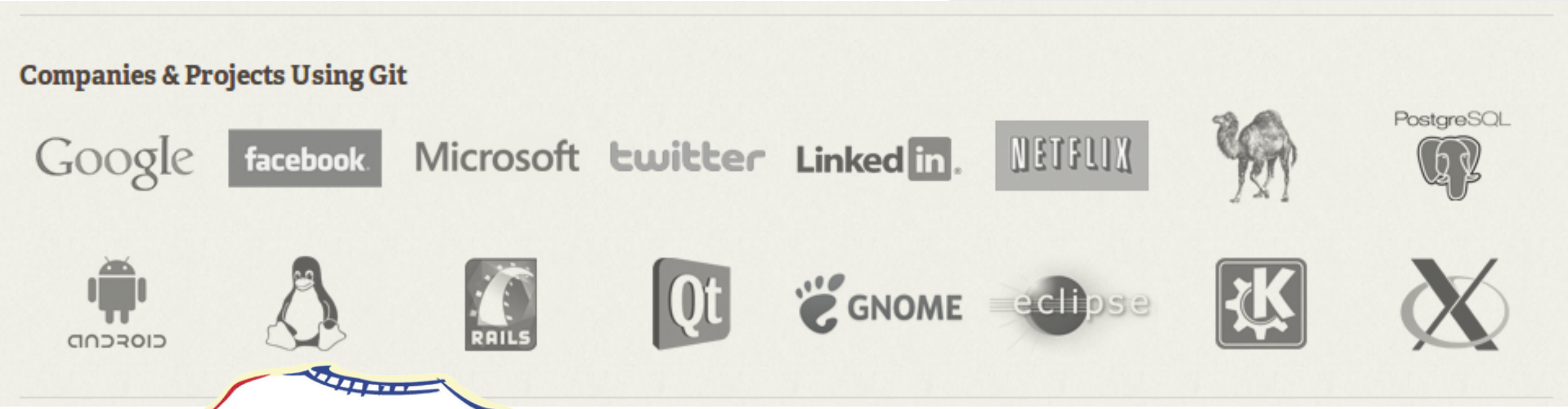
Linus Torvald - hated(s) almost any VCS out there ...

In 2005 after being blown by Bitkeeper he started his own VCS project !

(like the history of unix=>linux)

source : http://finland.fi/public/default.aspx?contentid=251229&contentlan=2&culture=en-US

The rest is history …


Companies & Projects Using Git

source: http://git-scm.com/

heroku

CloudBees

beanstalk

appfog

@bout

Git is a **free** and **open source** **distributed** **V**ersion **C**ontrol **S**ystem **designed** to handle everything from **small** to **very large** projects with **speed** and efficiency.

Git is easy to learn and has a tiny footprint with **lightning fast performance**. It **outclasses** SCM tools like **Subversion**, **CVS**, **Perforce**, and **ClearCase** with features like cheap local branching, convenient staging areas, and multiple workflows.

source: git-scm.org

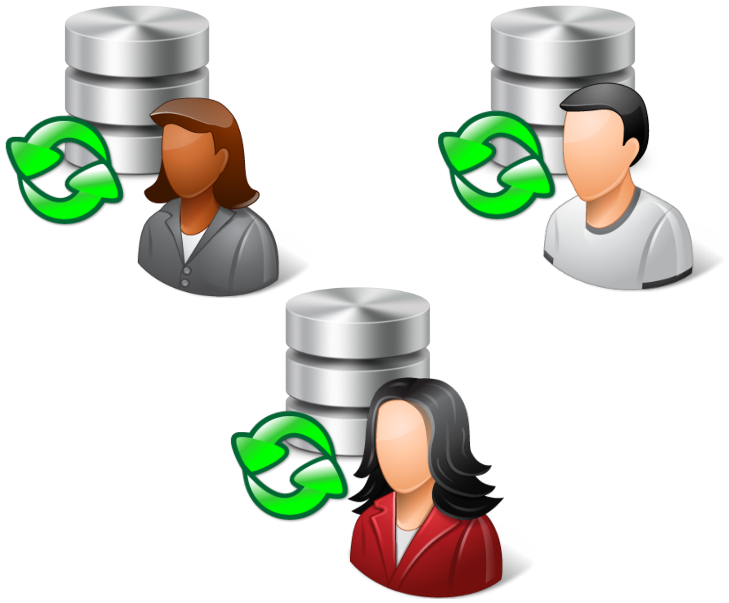
Dvcs .vs Cvcs

DVCS key concepts

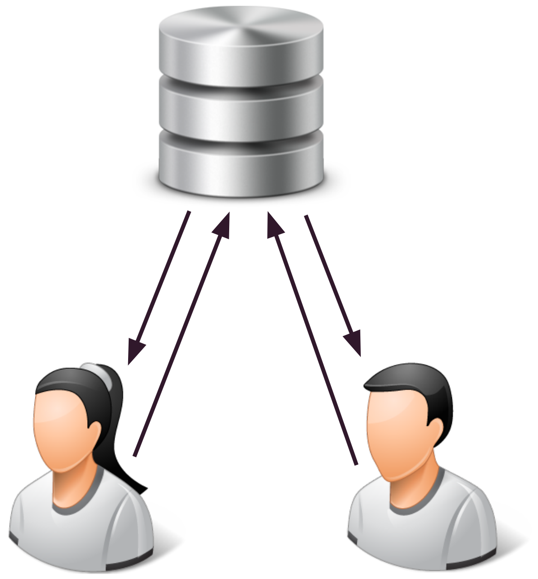
- **Everything** is done locally
  - check-in / checkout / commit / branch / merge
- **Collaboration** via repository sync
- **Peer-to-Peer** approach (All repo are equal).
- **Change** & **Share** [vs. commit / merge commit …]
- **Branch** & **Merge** - then share
- **Control** your **History** [commit history]

Dvcs vs. Cvcs

Distributed

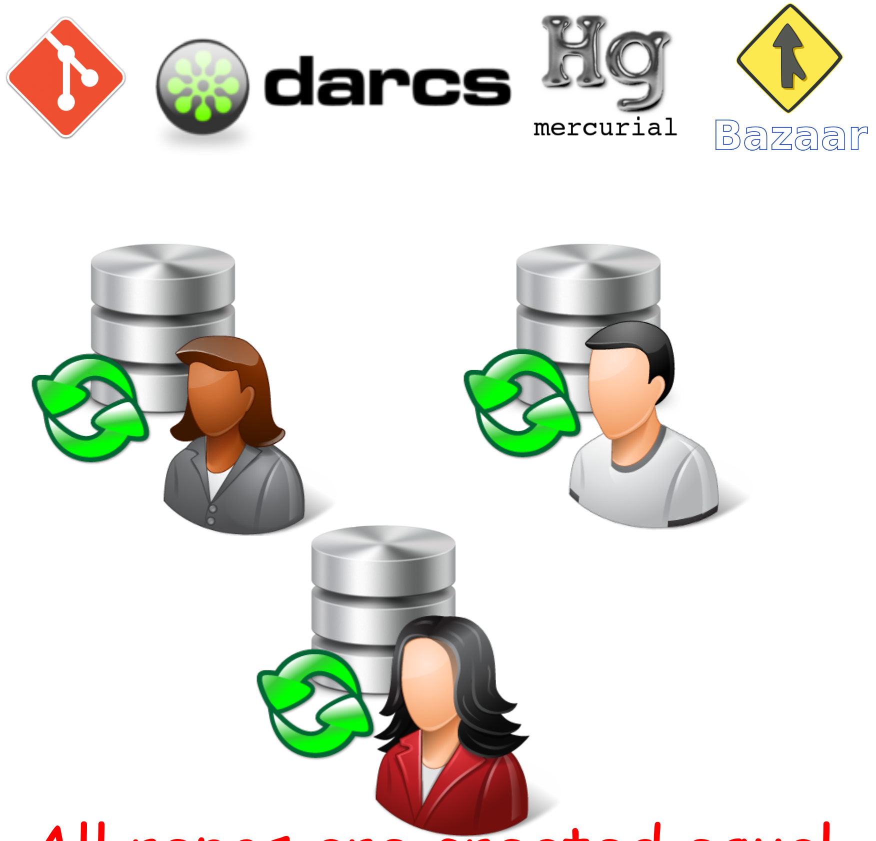


Centralized



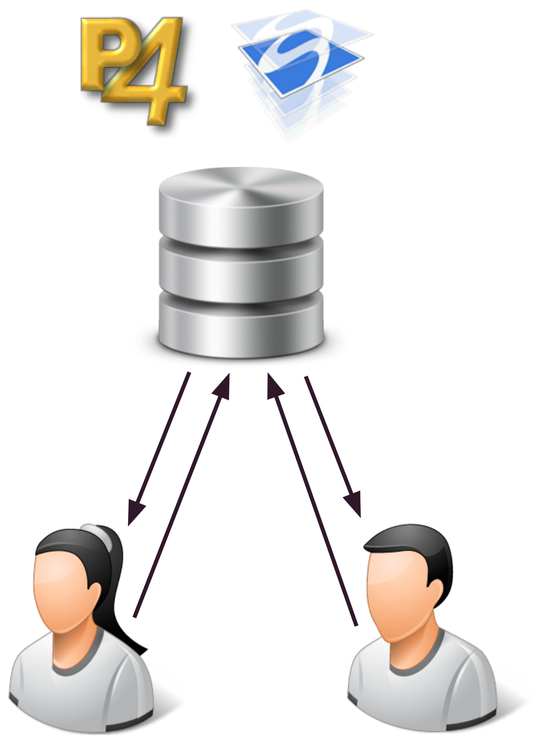
Dvcs vs. Cvcs

Distributed



All repos are created equal

Centralized

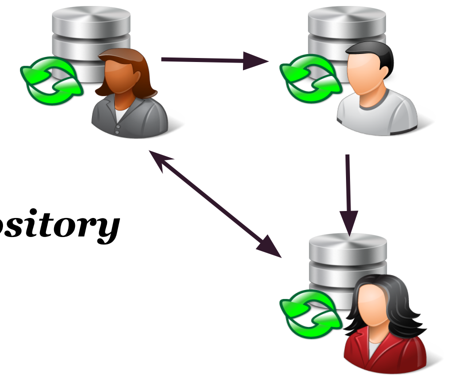


1 "master" repository

Dvcs vs. Cvcs

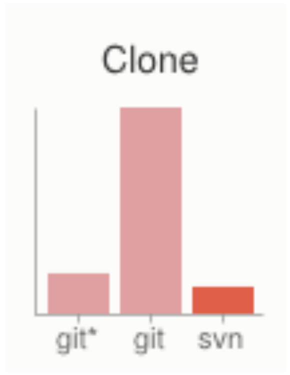
Distributed - pros

- ***Extremely* fast** because the tool only needs to access the local **hard drive**
- **Commits are local** without anyone else seeing them. **when you are ready to share => share**
- You can work offline ! [no internet / vpn etc]
- Share between developers before sharing with everyone



DVCS Can do everything a Centralized repository can do and much more ...

cons



- large binary sync might take a lot of ***disk space***
- Initial sync will take longer for all history to download

Installing & Configuring Git

Oh well, If you can't beat them join them ...

Getting git [installing]

Install Git

- From source :(
- From your favorite package manager / Installer
- And there are many ports / front-ends out there

http://sourceforge.net/projects/gitextensions/
https://code.google.com/p/tortoisegit/downloads/list
http://windows.github.com/

PLEASE NOTE: I will be covering Git from the Command Line - each of the "clients" mentioned above implement the exact same commands, so once you know the CLI you know them all !

Configure your environment

Per repo config - stored in *your_repo./git/config*

```
$> git config user.name "Haggai Philip Zagury"
$> git config user.email "hagzag@tikalk.com"
```

3	/etc/gitconfig (***system wide***)
2	~/.gitconfig (***user all repos***)
1	git_repo/.git/.config (***repo***)

Configure all your repositories [--global]
- stored in ~/.gitconfig or $USER/.gitconfig on windows

```
$> git config --global user.name "Haggai Philip Zagury"
$> git config --global user.email "hagzag@tikalk.com"
```

Results in:

```
[user]
        name = Haggai Philip Zagury
        email = hagzag@tikalk.com
```

git config http://git-scm.com/docs/git-config

TIP

```
git config --list
git config --global core.editor vim
```

Ignoring Files

- Create a .gitignore at the root of your git repository.
- Create a .gitignore at any directory level - the deeper the stronger.
- .git/info/exclude
- ***git config*** core.excludesfile

TIP

```
git config --global core.excludesfile ~/.gitignore
```

Everyday Git Workflow

Creating a repository

- Bob is a developer …
- Starting a new project on his personal laptop

```
~/ $> cd ~/Projects/git/git_intro/

~/Projects/git/git_intro/ $> git init
Initialized empty Git repository in /home/hagzag/Projects/git/git_intro/.git/
~/Projects/git/git_intro/(master) $>
```

git init http://git-scm.com/docs/git-init.html

Cloning a repository

- Cloning an existing repository

```
$> git clone git@github.com:jenkinsci/tikal-multijob-plugin.git
Cloning into 'tikal-multijob-plugin'...
remote: Counting objects: 1872, done.
remote: Compressing objects: 100% (661/661), done.
remote: Total 1872 (delta 601), reused 1745 (delta 476)
Receiving objects: 100% (1872/1872), 199.15 KiB | 195 KiB/s, done.
Resolving deltas: 100% (601/601), done.
```

- ***git clone repo_url dest_dir*** would yield the content into dest_dir

git clone http://git-scm.com/docs/git-clone

Getting help

```
~/ $> git help
usage: git [--version] [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [-c name=value] [--help]
           <command> [<args>]

The most commonly used git commands are:
   add        Add file contents to the index
   bisect     Find by binary search the change that introduced a bug
   branch     List, create, or delete branches
```

```
See 'git help <command>' for more information on a specific command.
~/ $> git help init
~/ $> git help config
~/ $> git help commit
~/ $> git help branch
```

git help http://git-scm.com/docs/git-help

Working with files

Creating our first file

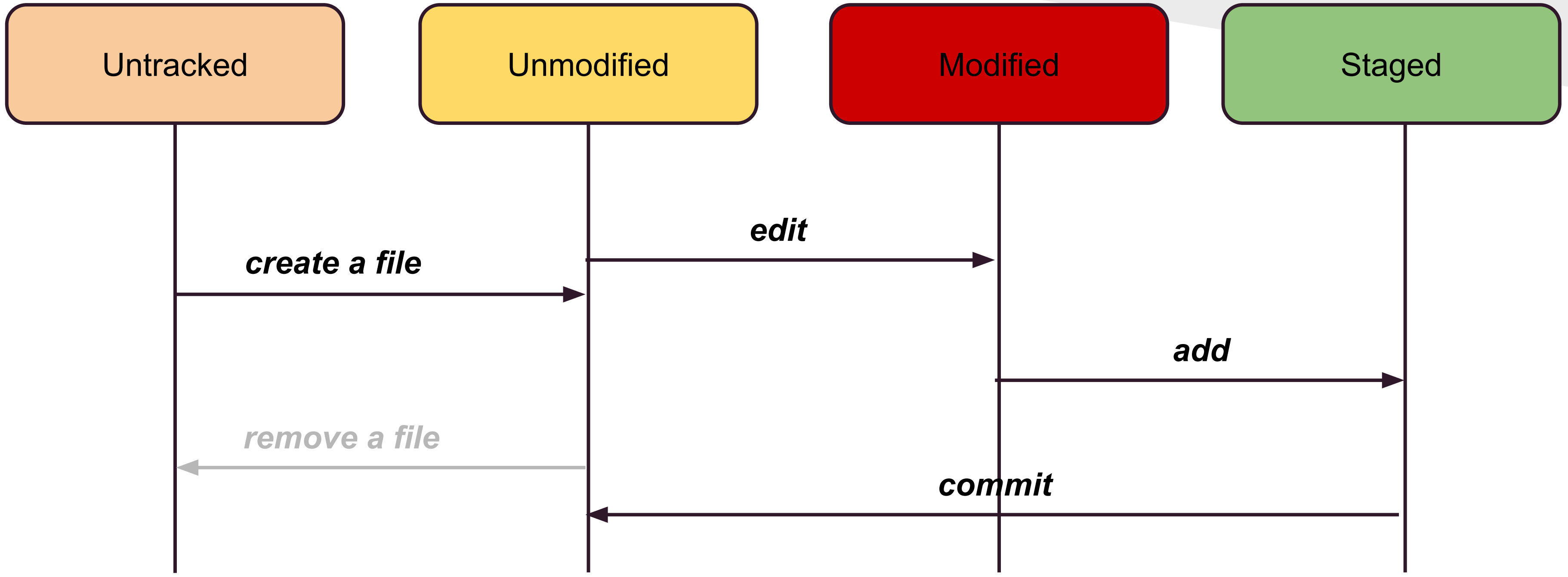
```
~/Projects/git/git_intro/(master) $> echo -e "=== README FILE for git_into ===\n Version 1.0" > README
```

File status ?

```
~/Projects/git/git_intro/(master) $> git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       README
nothing added to commit but untracked files present (use "git add" to track)
```

git status http://git-scm.com/docs/git-status

File status lifecycle



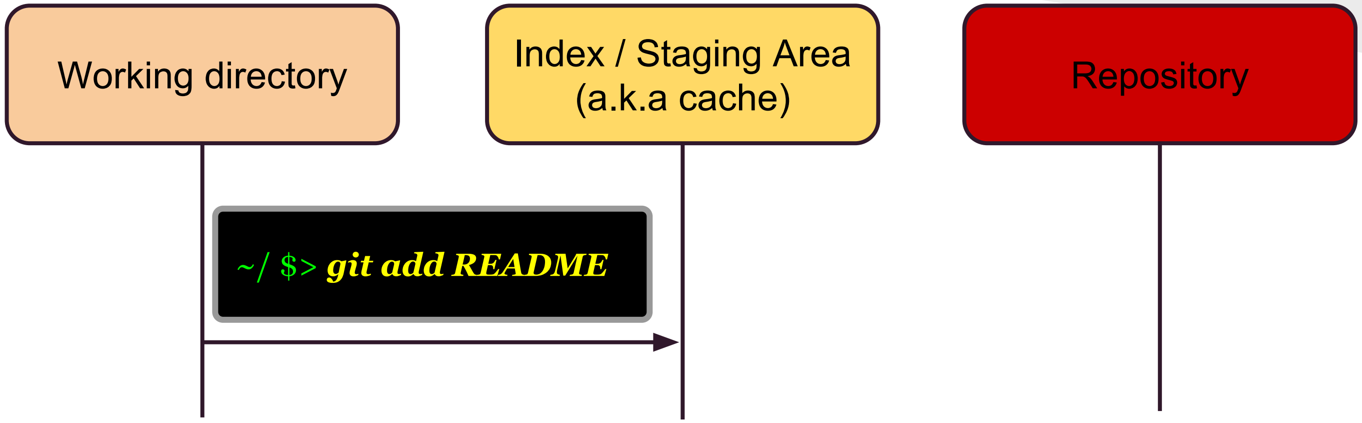
A file is Untracked until it is added with ***git add***

Unmodified as long as it's committed and its SHA-1 is equal to the checksum it had last time it was committed. (The SHA-1 used by git is not used only for file integrity - but as a hash function)

Modified once its SHA-1 differs from previous commit.

Staged once it was "git added"

File status lifecycle - add / status

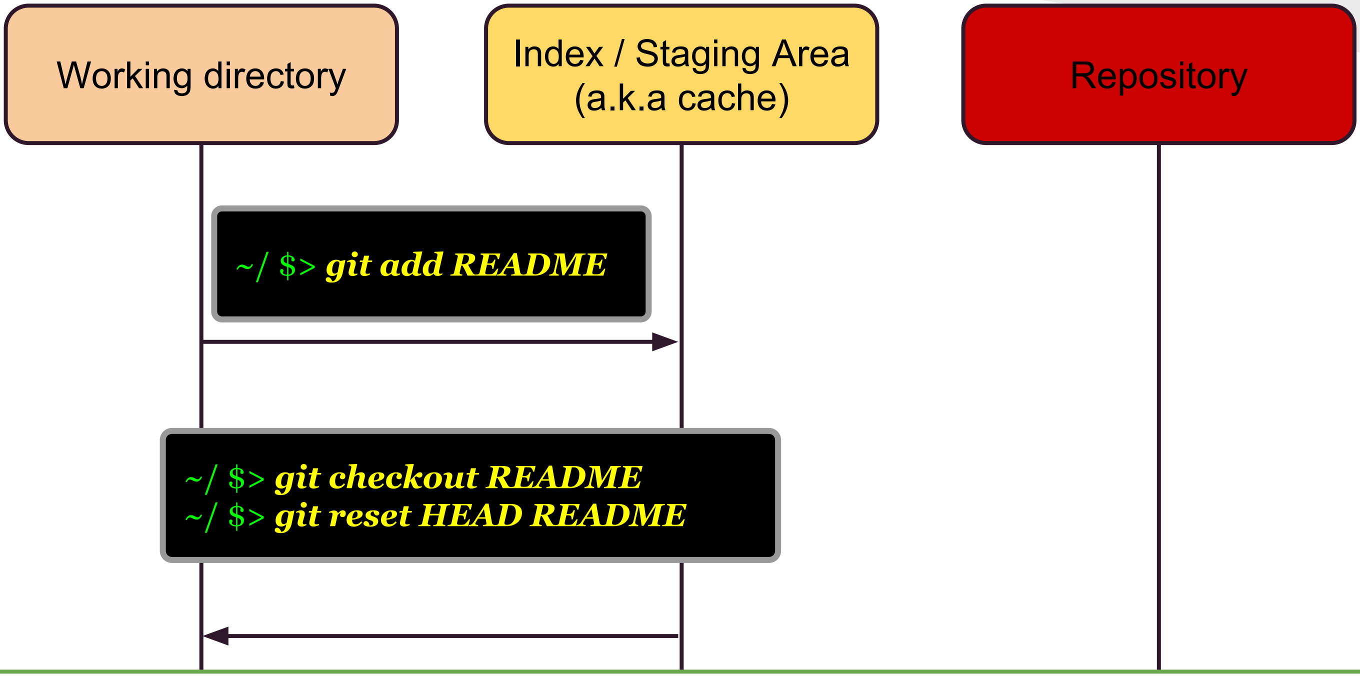


```
~/ $> git add README
```

Adding our README file will moving from the untracked state to the *staging* area.

```
~/Projects/git/git_intro/(master) $> git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   README
#
```

File status lifecycle - add / status



```
~/ $> git add README
```

```
~/ $> git checkout README
~/ $> git reset HEAD README
```

TIP

git checkout will undo any local changes [don't mixup with revert] (index untouched)
git reset **HEAD README** => Remove from staging area (local copy still modified).
git reset --hard will undo both the index and the working copy

git add [stage]	http://git-scm.com/docs/git-add
git checkout	http://git-scm.com/docs/git-checkout
git reset	http://git-scm.com/docs/git-reset

File status lifecycle - rm

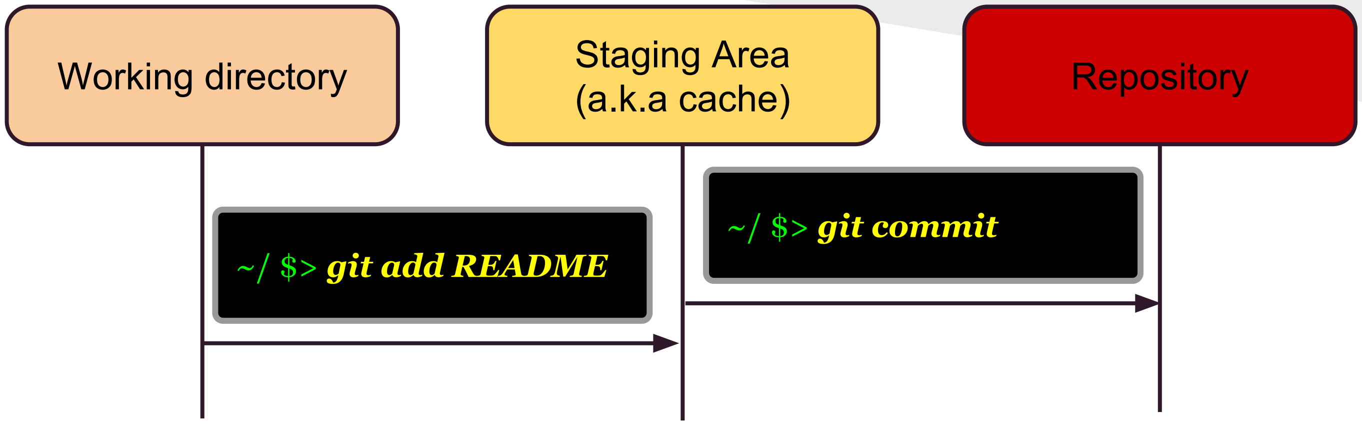
```
~/ $> git rm README
~/ $> git rm --cached README
```

git rm: Remove files from the **index,** or from the **working tree** **and** **the index**
git rm --cached: ***unstage*** and remove paths **only from the index**

git rm - adds the file to the index to be removed [the opposite of ***git add***]

git rm http://git-scm.com/docs/git-rm

File status lifecycle - commit



```
~/ $> git add README
```

```
~/ $> git commit
```

Adding our README file will moving from the **untracked** state to the **staging** area.

```
~/Projects/git/git_intro/(master) $> git commit -m "Adding README file"
[master (root-commit) 38a5307] Adding README file
 1 file changed, 2 insertions(+)
 create mode 100644 README
```

TIP

1. ***git commit -a*** will add any modified / deleted files to the Staging (Index) and commit them
2. **-m** "your commit message"

=> ***git commit -a -m*** "your commit message" <=

git commit http://git-scm.com/docs/git-commit

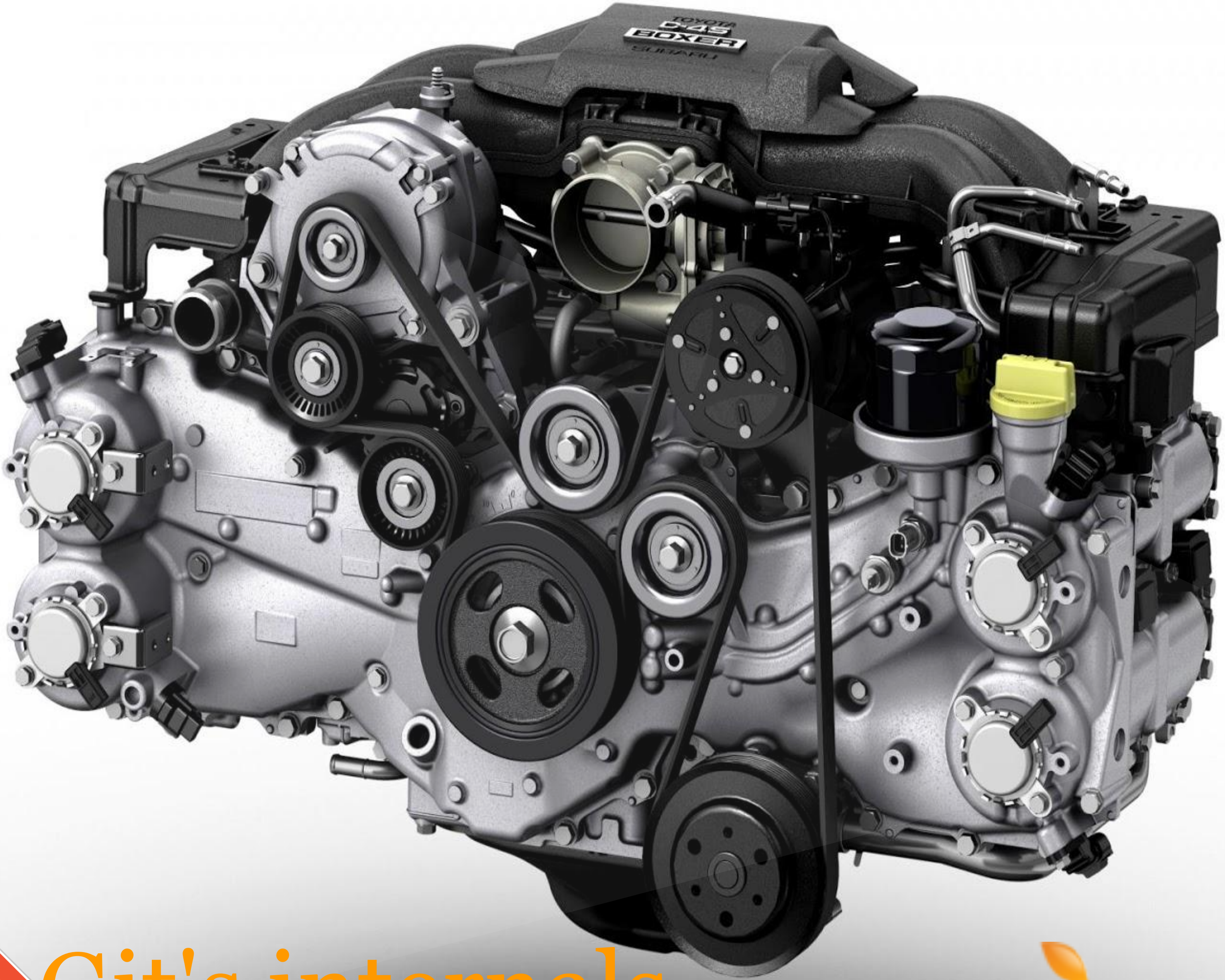
Buckets

- Working copy
- Index / Cache/ Stage
- Repository

Commands

- git help [cmd]
- git config
- git init
- git clone
- git add (stage)
- git status
- git reset
- git checkout
- git rm
- git commit

So far so **good** …



Git's internals

TIKAL

The ${GIT_DIR} .git directory

Before the commit:

```
~/Projects/git/git_intro/(master) $> find .git
.git
.git/refs
.git/refs/heads
.git/refs/tags
.git/description
.git/hooks/...
.git/config
.git/info
.git/info/exclude
.git/branches
.git/objects
.git/objects/pack
.git/objects/info
.git/HEAD
```

```
~/Projects/git/git_intro/(master) $> git commit -m
"Adding README file"
[master (root-commit) 38a5307] Adding README file
 1 file changed, 2 insertions(+)
 create mode 100644 README
```

After the commit:

```
~/Projects/git/git_intro/(master) $> find .git
.git
.git/COMMIT_EDITMSG
.git/refs
.git/refs/heads
.git/refs/heads/master
.git/refs/tags
.git/description
.git/hooks/...
.git/index
.git/config
.git/info
.git/info/exclude
.git/branches
.git/objects
.git/objects/bd
.git/objects/bd/2510ea0000fa2294947172f6f450bd0272fdab
.git/objects/38
.git/objects/38/a5307967fe2c9f92eb3c5a46ccdcc18410b4f3
.git/objects/pack
.git/objects/info
.git/objects/43
.git/objects/43/841a2f87570c9e458ab1da83396e0a5563ff36
.git/logs
.git/logs/refs
.git/logs/refs/heads
.git/logs/refs/heads/master
.git/logs/HEAD
.git/HEAD
```

What happened ? Git Objects

Every ***commit*** consists of objects of three ***types***:

[To be precise the ***tree*** & ***blob*** are created when you **add/stage** the commit is created when you -> commit], more about that in a few ...

commit -> a snapshot in time

tree -> represent directory

blob -> file content

DAG - Directed acyclic graph

A **directed acyclic graph** (**DAG** /ˈdæɡ/), is a directed graph with no directed cycles. That is, it is formed by a collection of vertices and directed edges, each edge connecting one vertex to another, such that there is no way to start at some vertex *v* and follow a sequence of edges that eventually loops back to *v* again.

Git uses DAG and a hash mechanism to redirect / map the repository.

Our README as object(s)

git log ->

```
commit 38a5307967fe2c9f92eb3c5a46ccdcc18410b4f3
Author: Haggai Philip Zagury <hagzag@tikalk.com>
Date:   Sat Apr 20 18:27:02 2013 +0300
```

commit ->

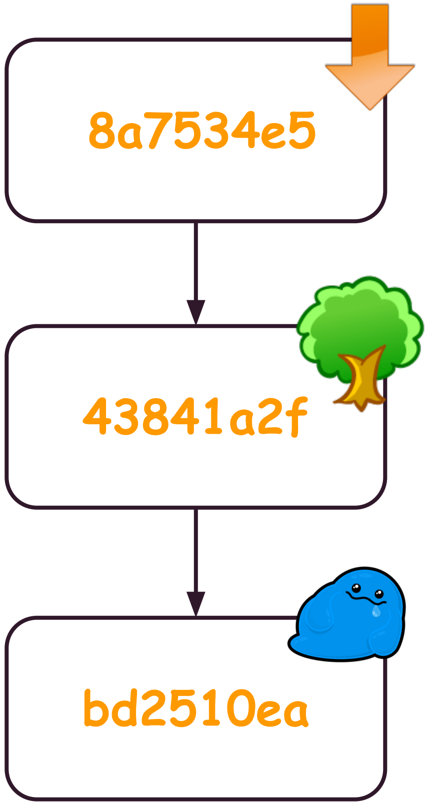
```
tree 43841a2f87570c9e458ab1da83396e0a5563ff36
author Haggai Philip Zagury <hagzag@tikalk.com> 1366471622 +0300
committer Haggai Philip Zagury <hagzag@tikalk.com> 1366471622 +0300
```

Tree ->

```
100644 blob bd2510ea0000fa2294947172f6f450bd0272fdab README
```

Blob ->

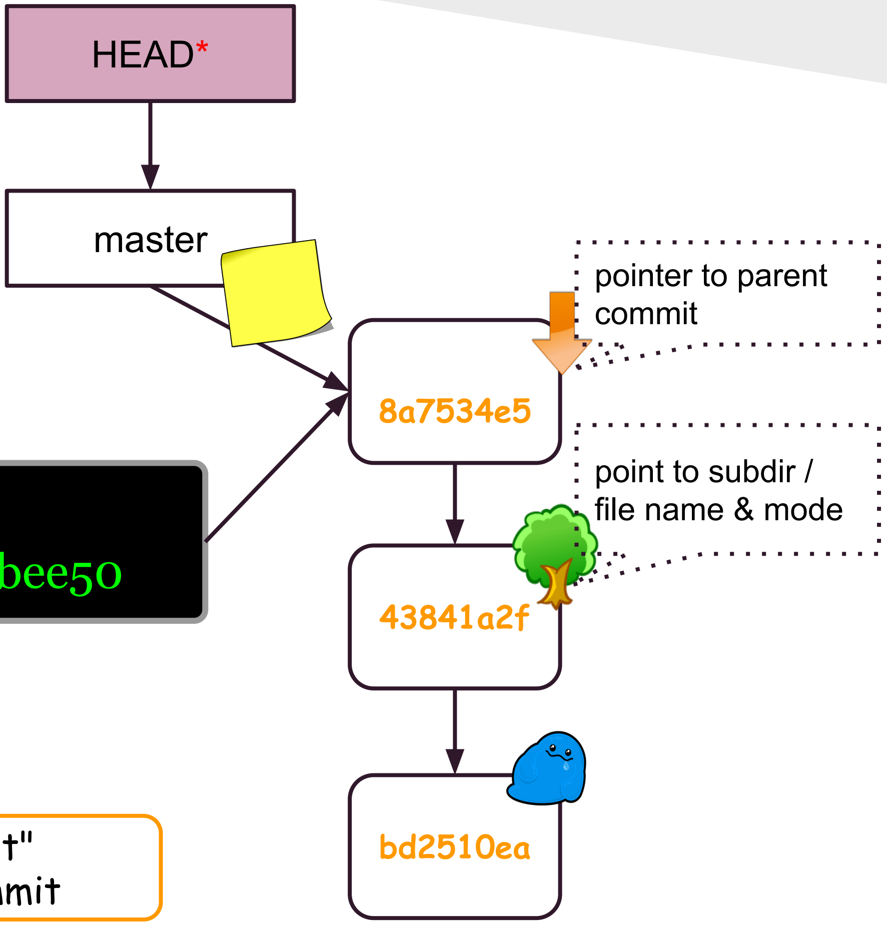
```
=== README FILE for git_intro ===
 Version 1.0
```



refs [references]

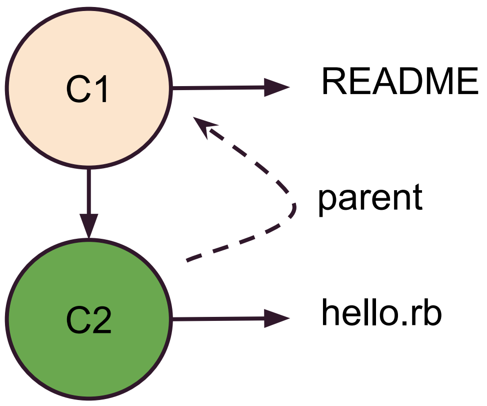
```
$> find .git/refs/
.git/refs/
.git/refs/heads
.git/refs/heads/master
.git/refs/tags
```

```
$> cat .git/refs/heads/master
8a7534e5ac1eb36ef21b8c4a06b8af5d59abee50
```



The "master" branch is just like a "post-it" reference to the SHA1 of the latest commit

Probing Git Objects



```
$> git cat-file -p 38a5307967fe2c9f92eb3c5a46ccdcc18410b4f3
tree 43841a2f87570c9e458ab1da83396e0a5563ff36
author Haggai Philip Zagury <hagzag@tikalk.com> 1366471622 +0300
committer Haggai Philip Zagury <hagzag@tikalk.com> 1366471622
+0300

Adding README file
```

```
$> git cat-file -p 8a7534e5ac1eb36ef21b8c4a06b8af5d59abee50
tree ea94fb0f34ca7dbcfc6ecaf7077dfe4b12725068
parent 38a5307967fe2c9f92eb3c5a46ccdcc18410b4f3
author Haggai Philip Zagury <hagzag@tikalk.com> 1366488967 +0300
committer Haggai Philip Zagury <hagzag@tikalk.com> 1366488967
+0300

Adding hello.rb to repo
```

In every repository there is at least one "parent-less" commit

git log http://git-scm.com/docs/git-log

TIP

The commands are presented for educational purposes and are rarely used by the common developer ...

It's a blob more probing...

```
$> git log
commit 8a7534e5ac1eb36ef21b8c4a06b8af5d59abee50
Author: Haggai Philip Zagury <hagzag@tikalk.com>
Date:   Sat Apr 20 23:16:07 2013 +0300

    Adding hello.rb to repo
```

```
$> git cat-file -t 8a7534e5ac1eb36ef21b8c4a06b8af5d59abee50
commit
$> git cat-file -t bd2510ea0000fa2294947172f6f450bd0272fdab
blob
```

```
$> find .git/objects/ -type f
.
git/objects/bd/2510ea0000fa2294947172f6f450bd0272fda
b
.git/objects/5e/b56f99ad91c6e8933c3e06593a66a09e3a1b91
.git/objects/38/a5307967fe2c9f92eb3c5a46ccdcc18410b4f3
.git/objects/ea/94fb0f34ca7dbcfc6ecaf7077dfe4b12725068
.git/objects/43/841a2f87570c9e458ab1da83396e0a5563ff36
.git/objects/8a/7534e5ac1eb36ef21b8c4a06b8af5d59abee50
```

```
bd + 2510ea0000fa2294947172f6f450bd0272fdab = README

$> git cat-file -p bd2510ea0000fa2294947172f6f450bd0272fdab
=== README FILE for git_into ===
 Version 1.0
```

Git stores a single file per piece of content, named with the **SHA-1 checksum** of the content and its header.

The subdirectory is named with **the first 2 characters of the SHA**, and the **filename** is the **remaining 38 characters**

Branching & tagging

The Idea

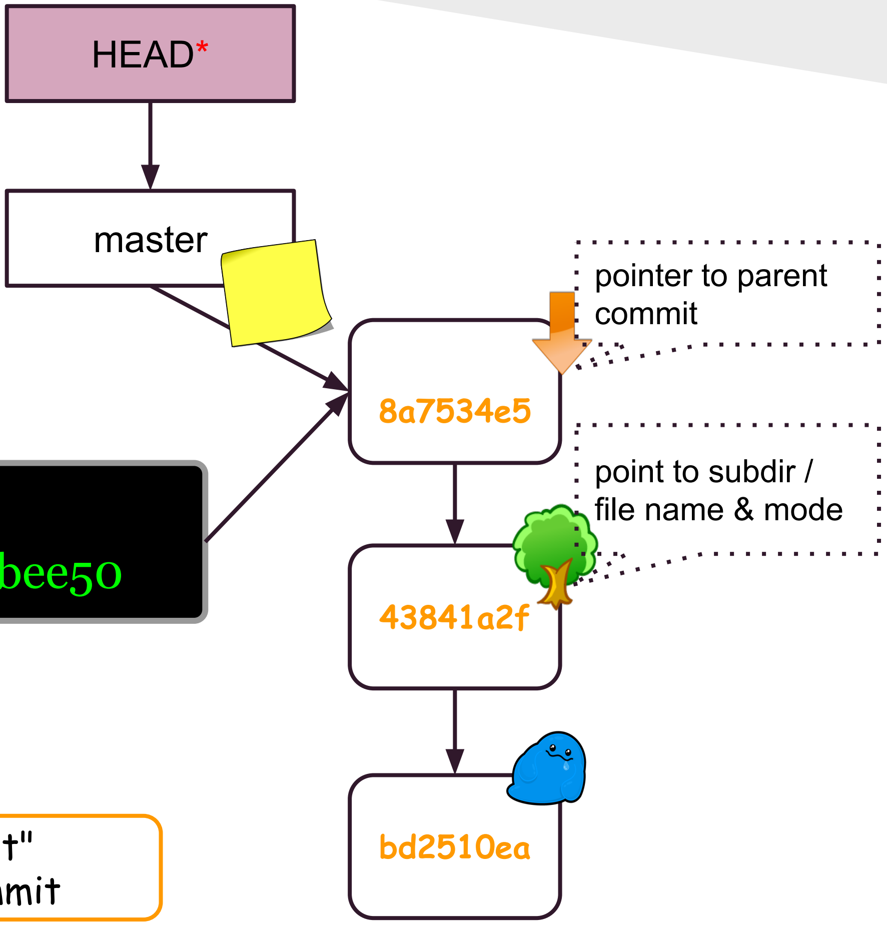
We already know branches :)

master == branch

refs [references]

```
$> find .git/refs/
.git/refs/
.git/refs/heads
.git/refs/heads/master
.git/refs/tags
```

```
$> cat .git/refs/heads/master
8a7534e5ac1eb36ef21b8c4a06b8af5d59abee50
```

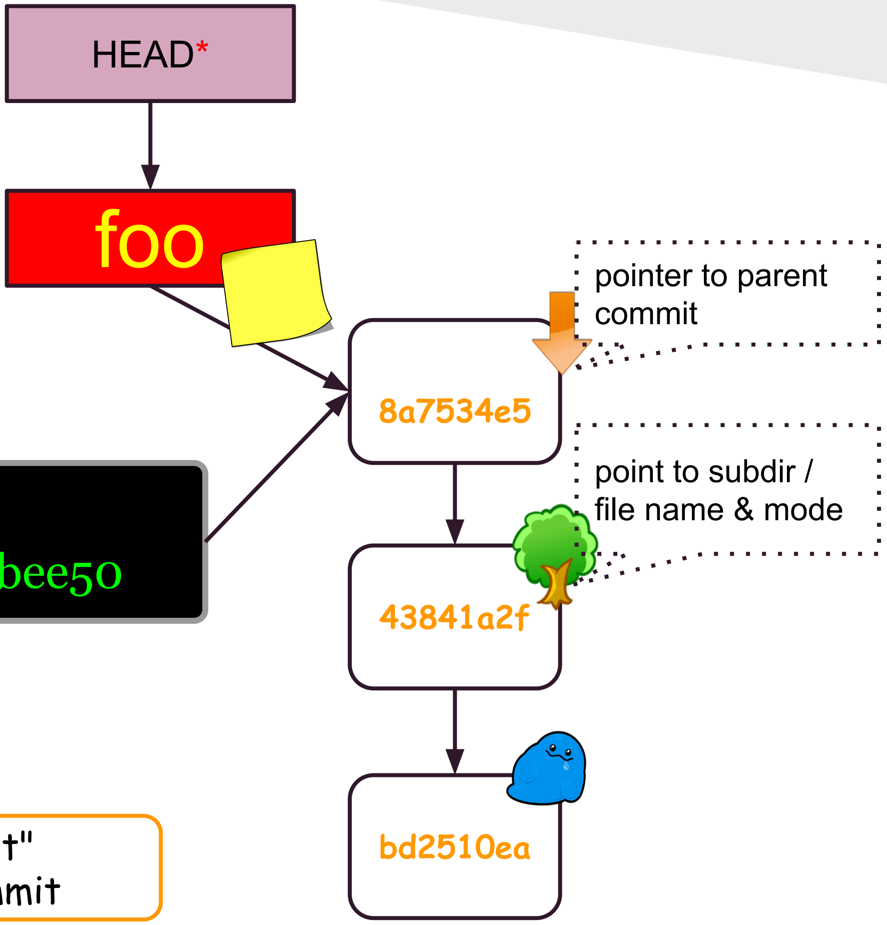


The "master" branch is just like a "post-it" reference to the SHA1 of the latest commit

refs [references]

```
$> find .git/refs/
.git/refs/
.git/refs/heads
.git/refs/heads/master
.git/refs/tags
```

```
$> cat .git/refs/heads/master
8a7534e5ac1eb36ef21b8c4a06b8af5d59abee50
```



The "master" branch is just like a "post-it" reference to the SHA1 of the latest commit

Context based development

```
$> git branch
* master
$> git branch the-idea
$> git branch*
 master
  the-idea
$> git checkout the-idea
Switched to branch 'the-idea'
```

```
$> ls .git/refs/heads/
master  the-idea
```

TIP

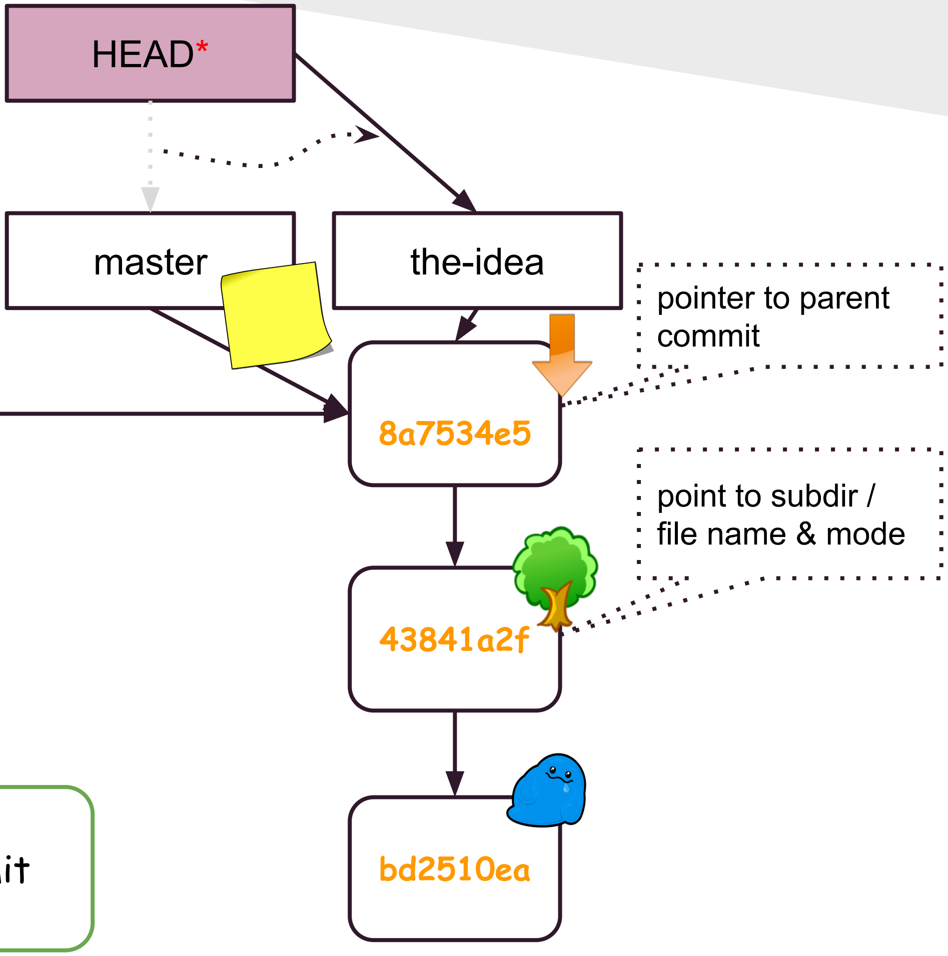
`$> git checkout -b second-idea`
will switch and create a new branch in that name in one command [like executing:
"git branch the-idea && git checkcout the-idea"]

git branch http://git-scm.com/docs/git-branch

refs [references] - branches

```
$> git checkout the-idea
Switched to branch 'the-idea'
```

```
$> cat .git/refs/heads/master
8a7534e5ac1eb36ef21b8c4a06b8af5d59abee50
$> cat .git/refs/heads/the-idea
8a7534e5ac1eb36ef21b8c4a06b8af5d59abee50
```



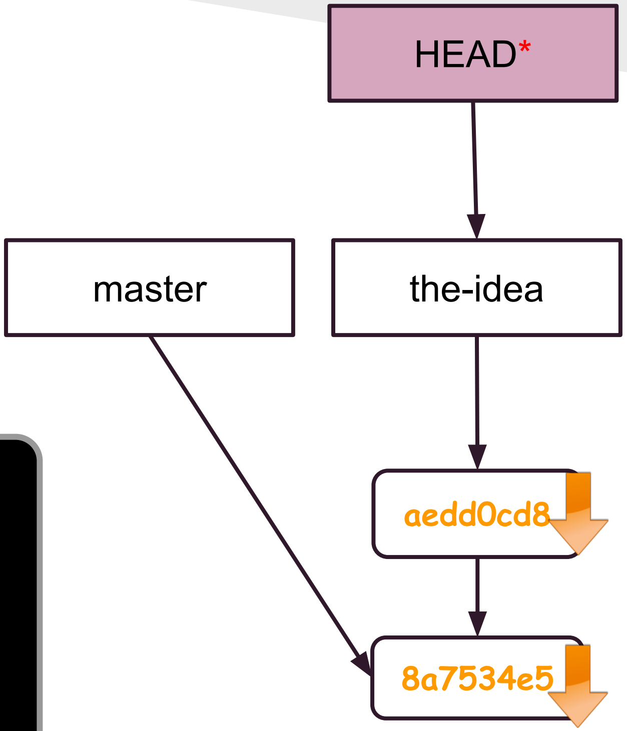
$> as long as I **haven't added** anything to the new branch, the pointer's **content** is on the same commit as "master" branch - Remember DAG ?!

refs change

```
$> sed -i s/1\.0/2\.0/g README
$> git commit -a -m "Bumping version to 2.0"
[the-idea aedd0cd] Bumping version to 2.0
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
 $> cat .git/refs/heads/master .git/refs/heads/the-idea
8a7534e5ac1eb36ef21b8c4a06b8af5d59abee50
aedd0cd8ba404f292bdf3f9542d67285c489a143
```

$>The reference to the new object has ***changed*** the parent object [DAG ...] is the same



Deleting branches

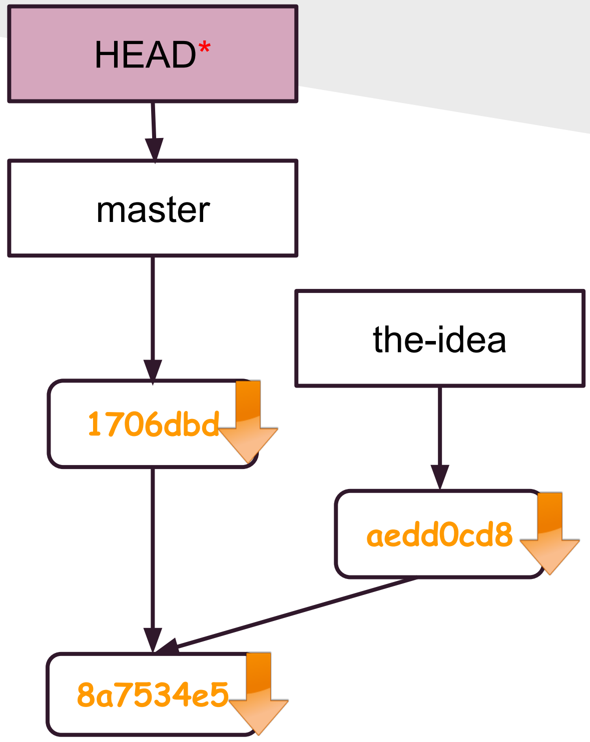
```
$> git branch -d the-idea
error: Cannot delete the branch 'the-idea' which you are currently on.
$> git checkout master
Switched to branch 'master'
$> git branch -d the-idea
error: The branch 'the-idea' is not fully merged.
If you are sure you want to delete it, run 'git branch -D the-idea'.
```

Why delete you say ?! [we never used to …]

More about why when we discuss implementation / methodology

Let's create a conflict (on master)

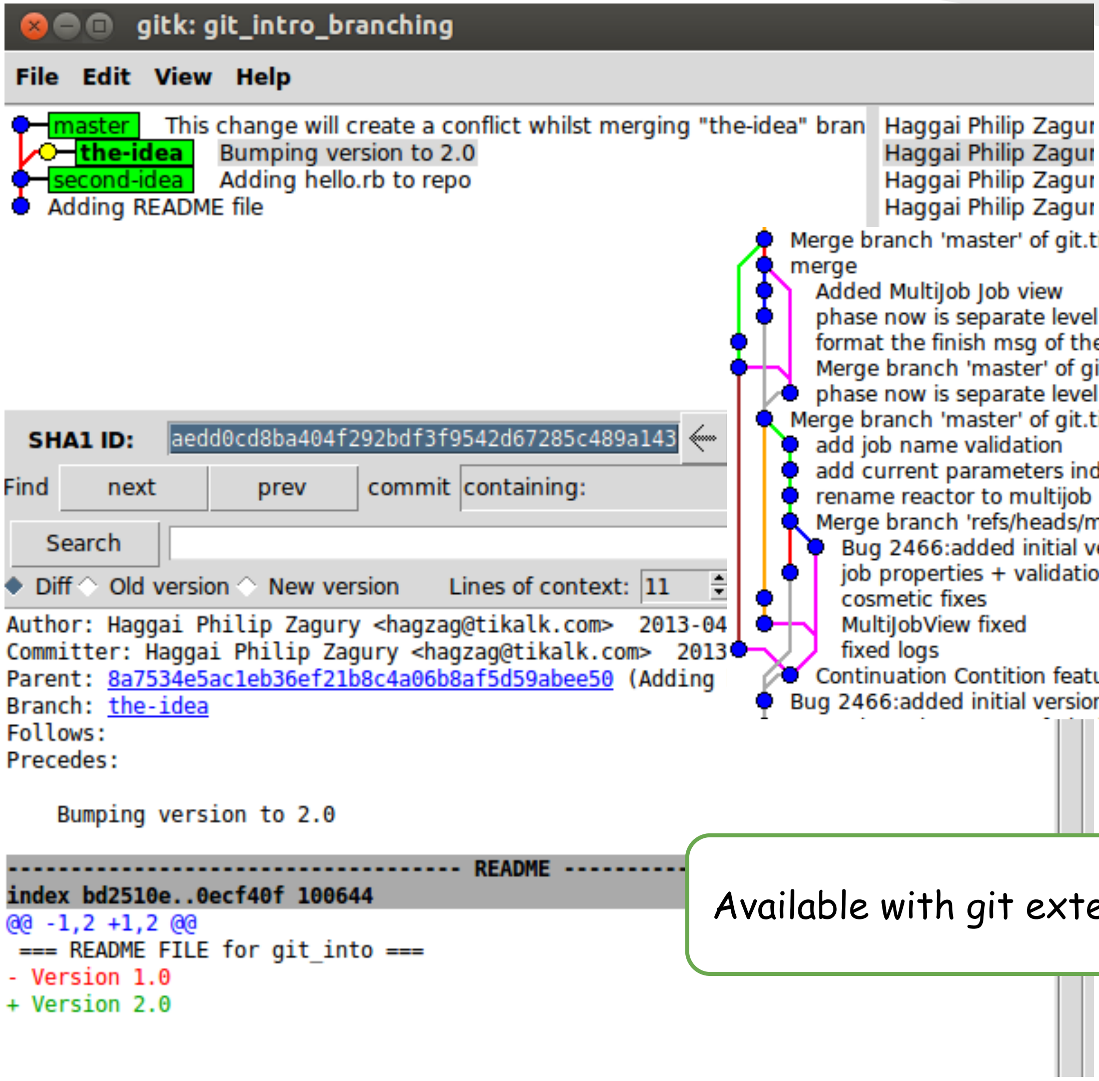
```
$> sed -i s/1\.0/1\.1/g README
$> git commit -a -m "This change will create a conflict whilst merging \"the-idea\" branch"
[master 1706dbd] This change will create a conflict whilst merging "the-idea" branch
 1 file changed, 1 insertion(+), 1 deletion(-)
```



"Visualize it"

```
$> git log --oneline --graph --decorate --all
* 1706dbd (master) This change will create a conflict whilst merging "the-idea" branch
|  * aedd0cd (HEAD, the-idea) Bumping version to 2.0
|/
* 8a7534e (second-idea) Adding hello.rb to repo
* 38a5307 Adding README file
```

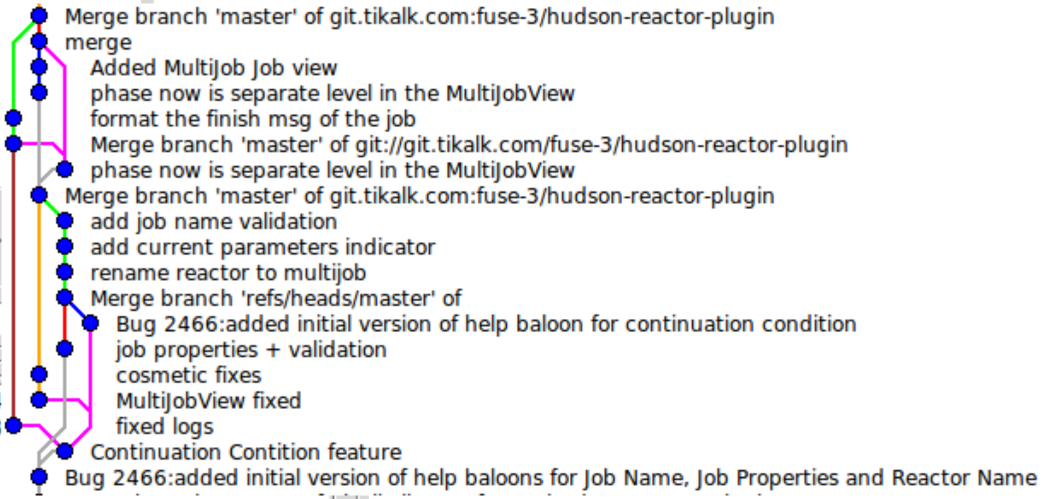
gitk - viewing changes …


gitk: git_intro_branching
File Edit View Help
master
the-idea
second-idea
SHA1 ID:
Find next prev commit containing:
Search
Diff Old version New version Lines of context: 11

```
This change will create a conflict whilst merging "the-idea" bran   Haggai Philip Zagur
Bumping version to 2.0                                               Haggai Philip Zagur
Adding hello.rb to repo                                              Haggai Philip Zagur
Adding README file                                                   Haggai Philip Zagur
```

```
SHA1 ID: aedd0cd8ba404f292bdf3f9542d67285c489a143

Author: Haggai Philip Zagury <hagzag@tikalk.com>  2013-04
Committer: Haggai Philip Zagury <hagzag@tikalk.com>  2013
Parent: 8a7534e5ac1eb36ef21b8c4a06b8af5d59abee50 (Adding
Branch: the-idea
Follows:
Precedes:

    Bumping version to 2.0

------------------------------------ README ---------
index bd2510e..0ecf40f 100644
@@ -1,2 +1,2 @@
 === README FILE for git_into ===
- Version 1.0
+ Version 2.0
```

```
$> gitk --all
```



```
Merge branch 'master' of git.tikalk.com:fuse-3/hudson-reactor-plugin
merge
  Added MultiJob Job view
  phase now is separate level in the MultiJobView
  format the finish msg of the job
  Merge branch 'master' of git://git.tikalk.com/fuse-3/hudson-reactor-plugin
  phase now is separate level in the MultiJobView
Merge branch 'master' of git.tikalk.com:fuse-3/hudson-reactor-plugin
  add job name validation
  add current parameters indicator
  rename reactor to multijob
  Merge branch 'refs/heads/master' of
    Bug 2466:added initial version of help baloon for continuation condition
    job properties + validation
    cosmetic fixes
    MultiJobView fixed
    fixed logs
  Continuation Contition feature
Bug 2466:added initial version of help baloons for Job Name, Job Properties and Reactor Name
```

TIP

Available with git extensions & others | equivalent

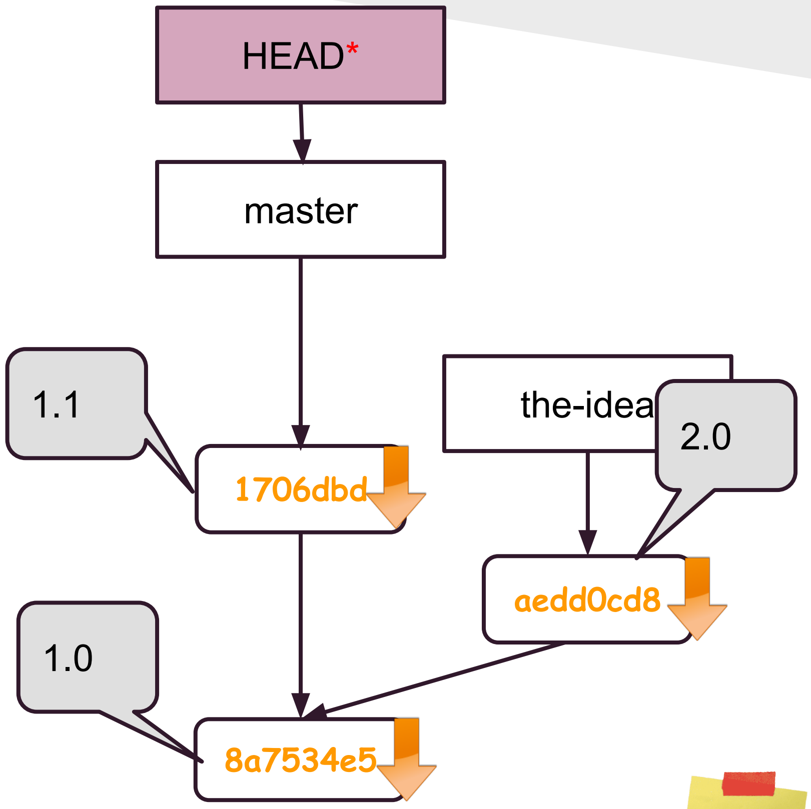
Tagging

Wait, I need to tag the version 1.0 ...

And no, a **TAG isn't a BRANCH** !

Tag is an object in the DAG + commit message & optional gpg signature

```
$> git tag -a v1.0 -m 'version 1.0' 8a7534e5
$> git show v1.0
tag v1.0
Tagger: Haggai Philip Zagury <hagzag@tikalk.com>
Date:   Wed Apr 24 01:24:03 2013 +0300
verion 1.0
commit 38a5307967fe2c9f92eb3c5a46ccdcc18410b4f3
```



git tag http://git-scm.com/docs/git-tag

TIP

git tag foo - will create a tag named foo to the current HEAD reference

Commands

- git branch
- git tag
- git checkout

Browsing

- git log [--online]
- gitk

Merging with Git

Diff because you can't merge without a diff :)

```
$> git diff master
diff --git a/README b/README
index 10f515a..0ecf40f 100644
--- a/README
+++ b/README
@@ -1,2 +1,2 @@
 === README FILE for git_into ===
- Version 1.1
+ Version 2.0
```

TIP

git diff (with no args) diff working tree to index
git diff arg1 arg2 -- <path> (git diff the-idea -- ./)

git config --global diff.tool <path_to_diff_tool>

Merge

git merge master the-idea

```
$> git merge master the-idea
Auto-merging README
CONFLICT (content): Merge conflict in README
Automatic merge failed; fix conflicts and then commit the result.
```

```
=== README FILE for
git_into ===
<<<<<<< HEAD
 Version 1.1
=======
 Version 2.0
>>>>>>> the-idea
```

```
$> git merge master the-idea
Auto-merging README
CONFLICT (content): Merge conflict in README
Automatic merge failed; fix conflicts and then commit the result.
```

Merge

Merge 2 or more commits

git merge <branch>

git merge <commit>

Git merge <branch1> <branch2> (Octopus)

```
$> git log --pretty=oneline --graph --decorate --all
*   3bd95903b3e2a3934b1d3bc1495f7c5c9ced5df2 (HEAD, master) Merge branch 'the-idea'
|\
| * aedd0cd8ba404f292bdf3f9542d67285c489a143 (the-idea) Bumping version to 2.0
* | 1706dbd411de152c462172386eafa238fc50f50b This change ... conflict ... merging "the-idea" branch
|/
* 8a7534e5ac1eb36ef21b8c4a06b8af5d59abee50 (second-idea) Adding hello.rb to repo
* 38a5307967fe2c9f92eb3c5a46ccdcc18410b4f3 Adding README file

$> git log
commit 3bd95903b3e2a3934b1d3bc1495f7c5c9ced5df2
Merge: 1706dbd aedd0cd
Author: Haggai Philip Zagury <hagzag@tikalk.com>
Date:   Tue Apr 23 22:34:54 2013 +0300
    Merge branch 'the-idea'
    Conflicts:
        README
```

Git Merge Abort

In a non conflicting merge => the repo is in a idle state.

In a conflict unless using ***git merge abort*** the current state is that there is a "ready-made" commit message for the next ***git commit*** + conflicted files are ***marked*** in the working directory

Fast Forward

When the target (**HEAD**) is ancestor of the merged commit we can simply move the label.

```
$> git log --pretty=oneline --graph --decorate --all
* 6c3ad0acdec4d777280a982fc455bda3d6207961 (HEAD, the-idea) Adding two more files to show fast
forward
* 160f9d7b0ed4196f7ded236a7e88f1d51b78b3eb Adding more files ...
* 806046ca43af65fdbda9dc36c156cea04d8ff1ae Adding a file
* 154756115a95beba276055e0e4d01d546b11d8c0 (master) Adding readme file


$> git merge the-idea
Updating 1547561..6c3ad0a
Fast-forward
 0 files changed
 create mode 100644 123/123.txt
 create mode 100644 234/234.txt
 create mode 100644 file.txt
 create mode 100644 file2.txt
 create mode 100644 file3.txt
```

TIP

Do some "housekeeping" and delete Redundant branches ...

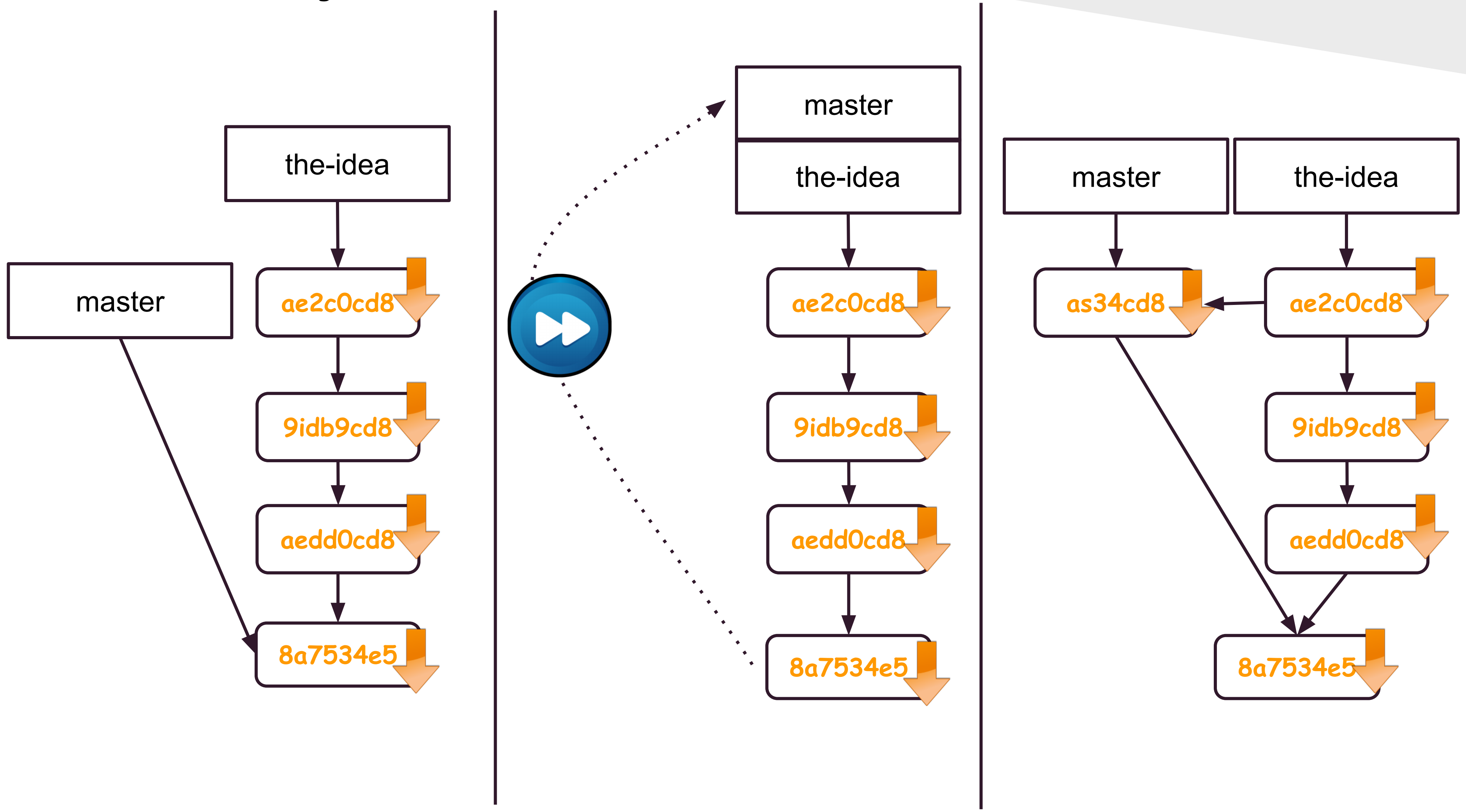
git branch -d the-idea

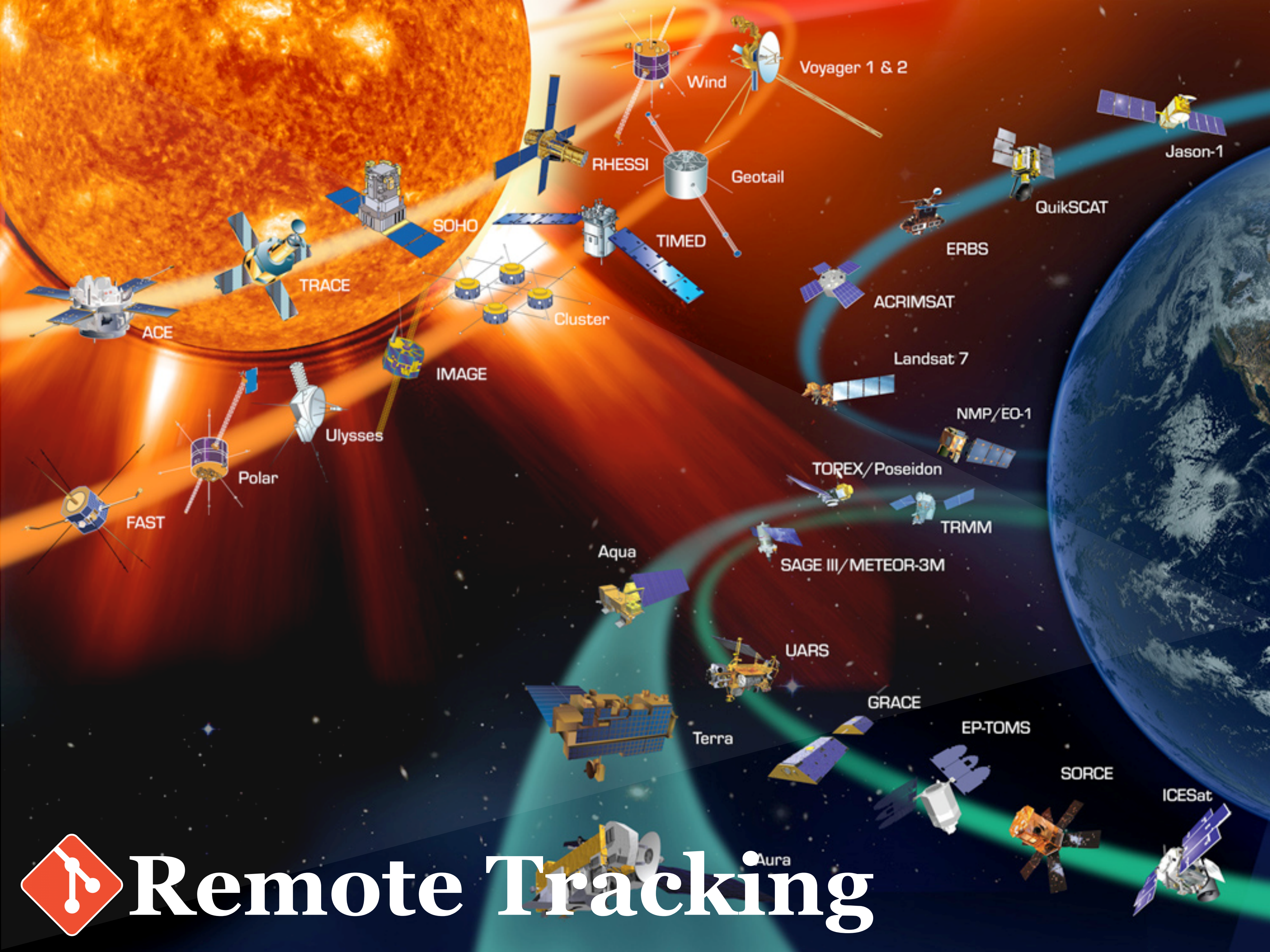
Non/Fast Forward

Before merge

Fast Forward

Non Fast Forward

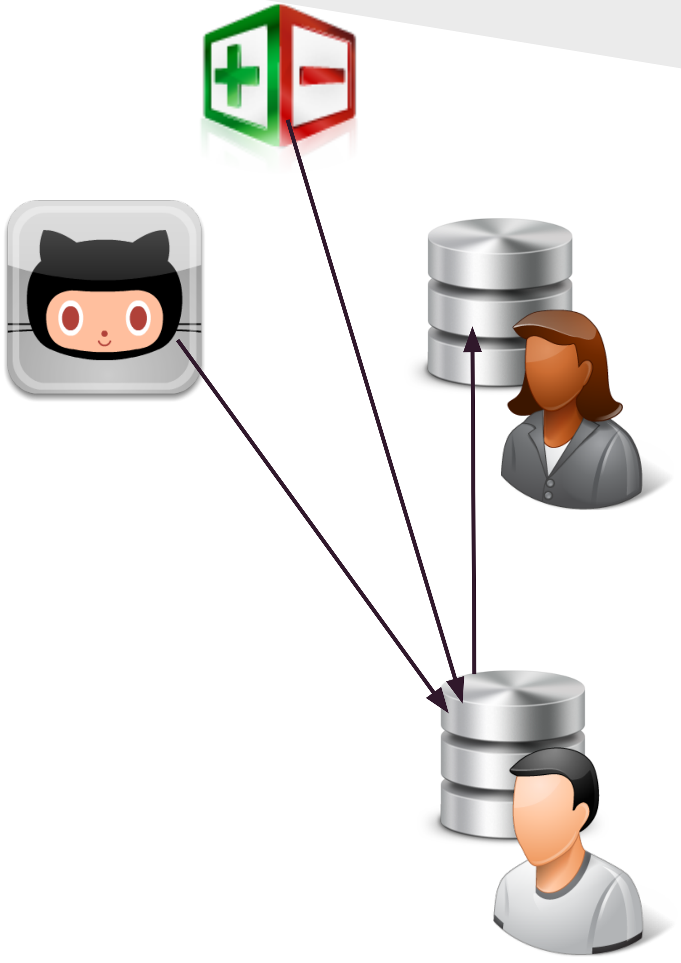




Remote Tracking

Remotes ?

```
$> git clone git@github.com:jenkinsci/tikal-multijob-plugin.git
Cloning into 'tikal-multijob-plugin'...
remote: Counting objects: 1872, done.
remote: Compressing objects: 100% (661/661), done.
remote: Total 1872 (delta 601), reused 1745 (delta 476)
Receiving objects: 100% (1872/1872), 199.15 KiB | 195 KiB/s, done.
Resolving deltas: 100% (601/601), done.
```



git clone http://git-scm.com/docs/git-clone

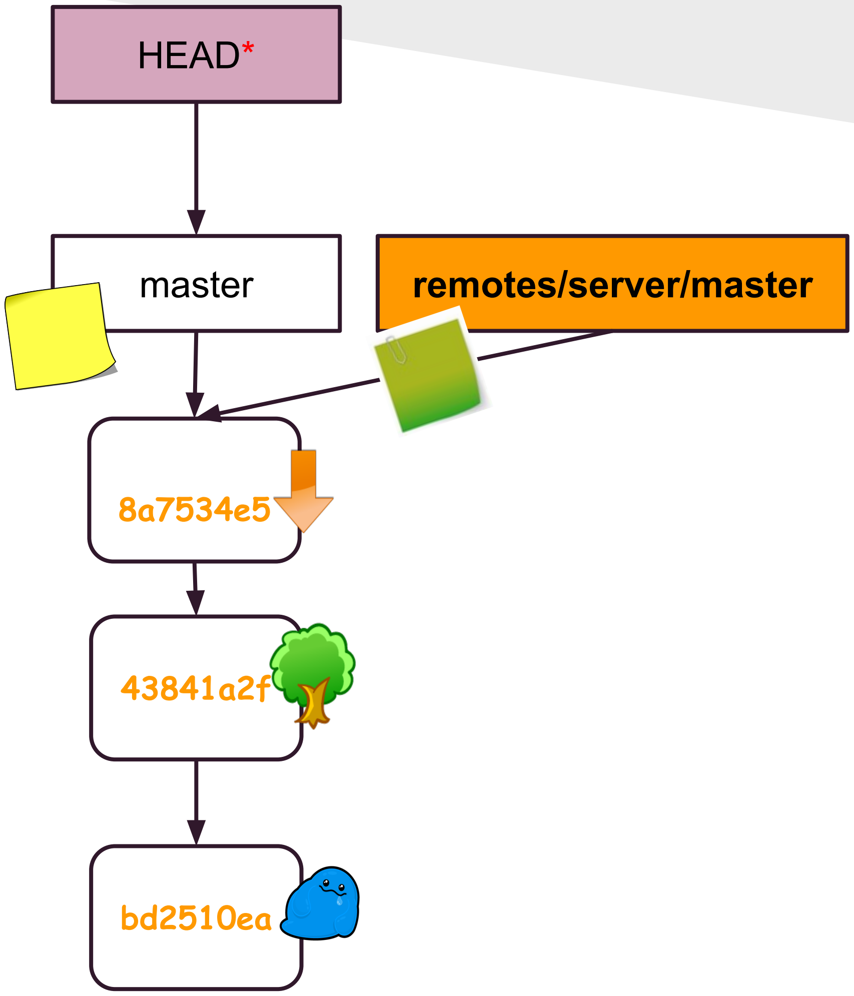
Remote tracking Reference(s)

The remote/master is the same type of reference like the "local" master but from a different **<u>namespace</u>**.

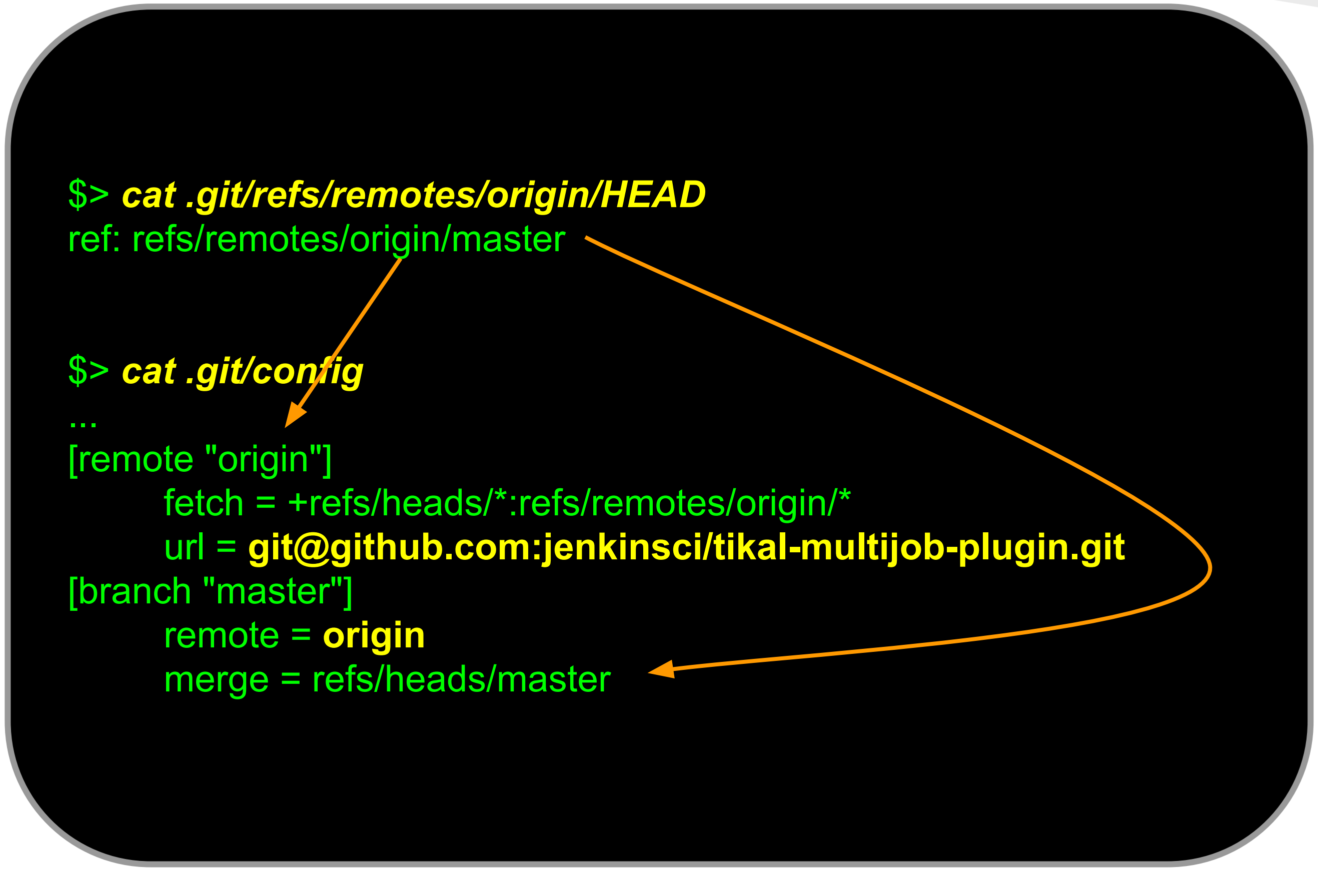
.git/refs/remotes/...

This namespace is **mapped to** the **remote** server !

- represented by a ***url***



Cloned repository



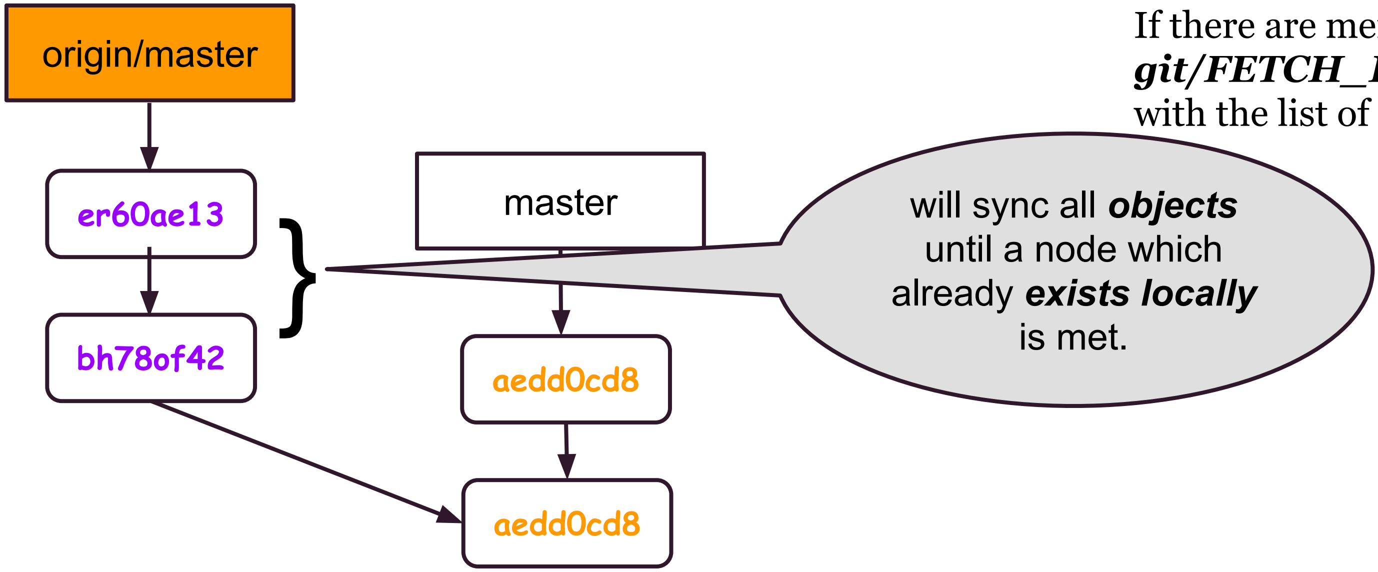
```
$> cat .git/refs/remotes/origin/HEAD
ref: refs/remotes/origin/master


$> cat .git/config
...
[remote "origin"]
        fetch = +refs/heads/*:refs/remotes/origin/*
        url = git@github.com:jenkinsci/tikal-multijob-plugin.git
[branch "master"]
        remote = origin
        merge = refs/heads/master
```

Remotes - git fetch

Update **all refs** of origin (Branches, tags, blobs, trees etc).

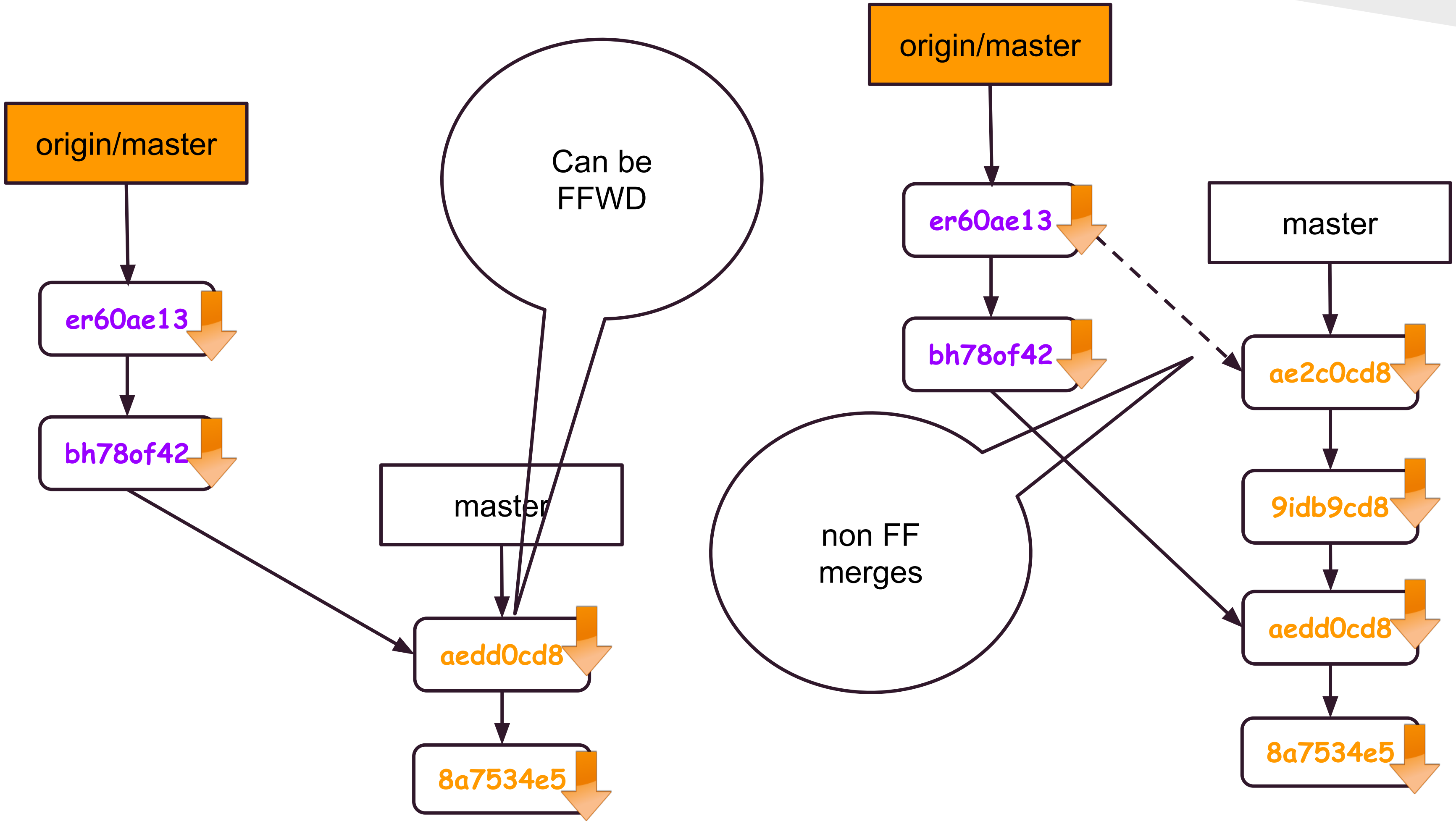
Nothing except origin refs have change locally in our repository.

If there are merges to be made a ***.git/FETCH_HEAD*** file will be created with the list of commits who need merge.


origin/master
er60ae13
bh78of42
master
aedd0cd8
aedd0cd8
will sync all ***objects*** until a node which already ***exists locally*** is met.

Remotes - git pull

Attempts to fetch & merge at the same time.



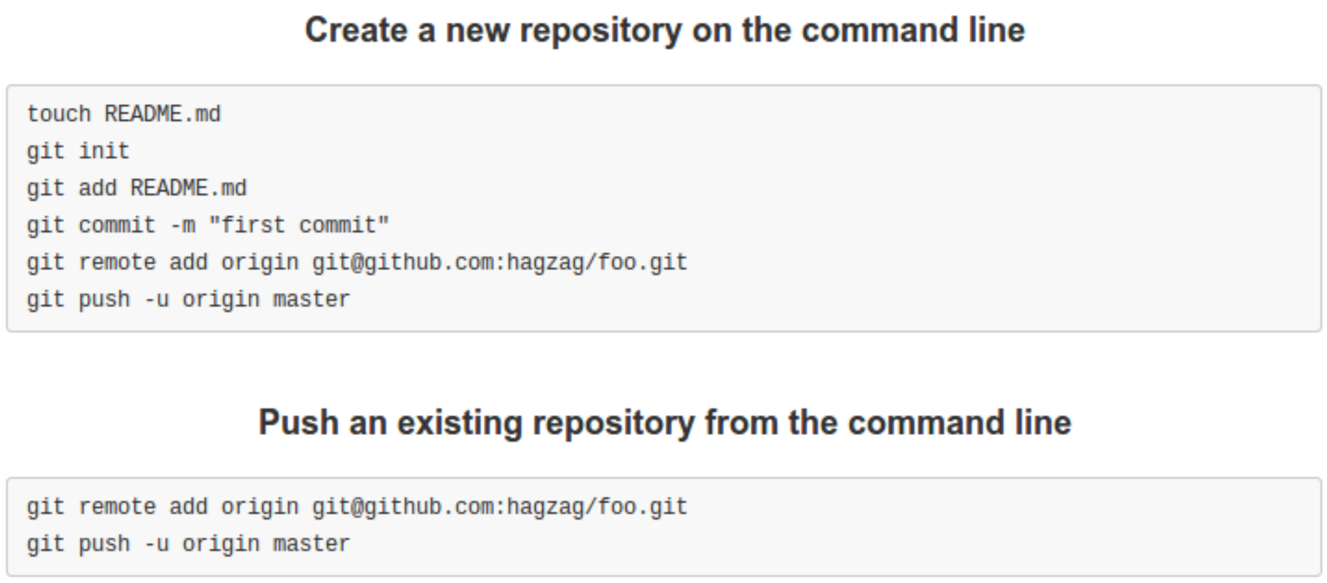
Remotes - git push [share]

If you have one remote [origin], ***git push*** will suffice.

If you have more than one …

git push <foo> master

Adding a remote



Create a new repository on the command line

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:hagzag/foo.git
git push -u origin master
```

Push an existing repository from the command line

```
git remote add origin git@github.com:hagzag/foo.git
git push -u origin master
```

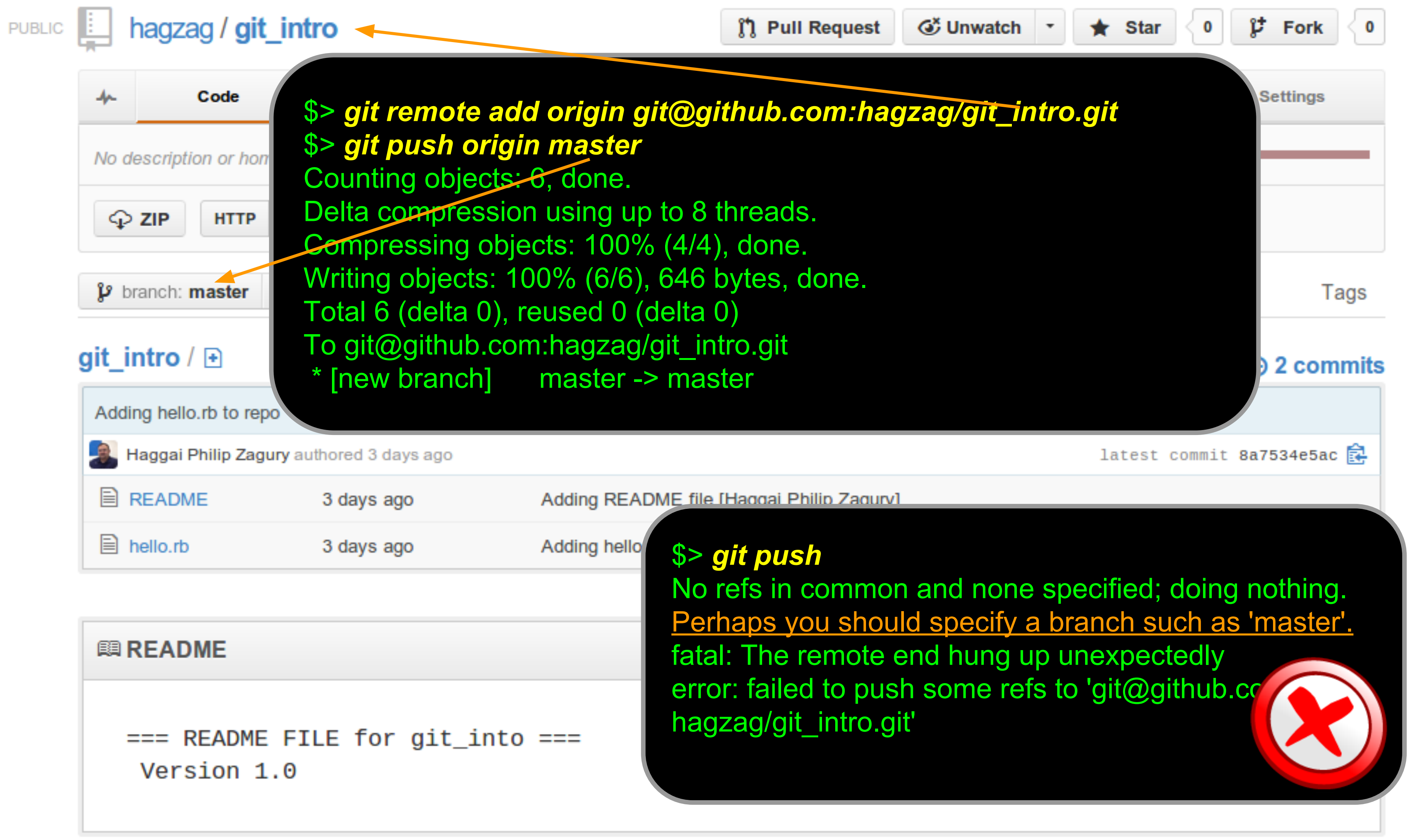
git remote add origin https://server/repo_name.git

git remote add origin git@server:user/repo_name.git

git push (to:)***origin*** (branch:)***master***

git remote http://git-scm.com/docs/git-remote

Pushing [Sharing]



PUBLIC hagzag / git_intro

Pull Request | Unwatch | Star 0 | Fork 0

Code | Settings

No description or hom

ZIP | HTTP

branch: master | Tags

git_intro / | 2 commits

Adding hello.rb to repo

Haggai Philip Zagury authored 3 days ago | latest commit 8a7534e5ac

README	3 days ago	Adding README file [Haggai Philip Zagury]
hello.rb	3 days ago	Adding hello

README

```
=== README FILE for git_into ===
 Version 1.0
```

```
$> git remote add origin git@github.com:hagzag/git_intro.git
$> git push origin master
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 646 bytes, done.
Total 6 (delta 0), reused 0 (delta 0)
To git@github.com:hagzag/git_intro.git
 * [new branch]      master -> master
```

```
$> git push
No refs in common and none specified; doing nothing.
Perhaps you should specify a branch such as 'master'.
fatal: The remote end hung up unexpectedly
error: failed to push some refs to 'git@github.co
hagzag/git_intro.git'
```

Commands

- git clone
- git fetch
- git pull
- git push
- git remote [add]

Collaboration workflow

1. Clone a remote repo
2. Perform changes [master/private branch]
3. Pull (if your lucky ...) / Fetch - Merge to sync
4. (more changes?) -> Push to remote

clone
branch
checkout
add
commit
fetch
merge
pull
push

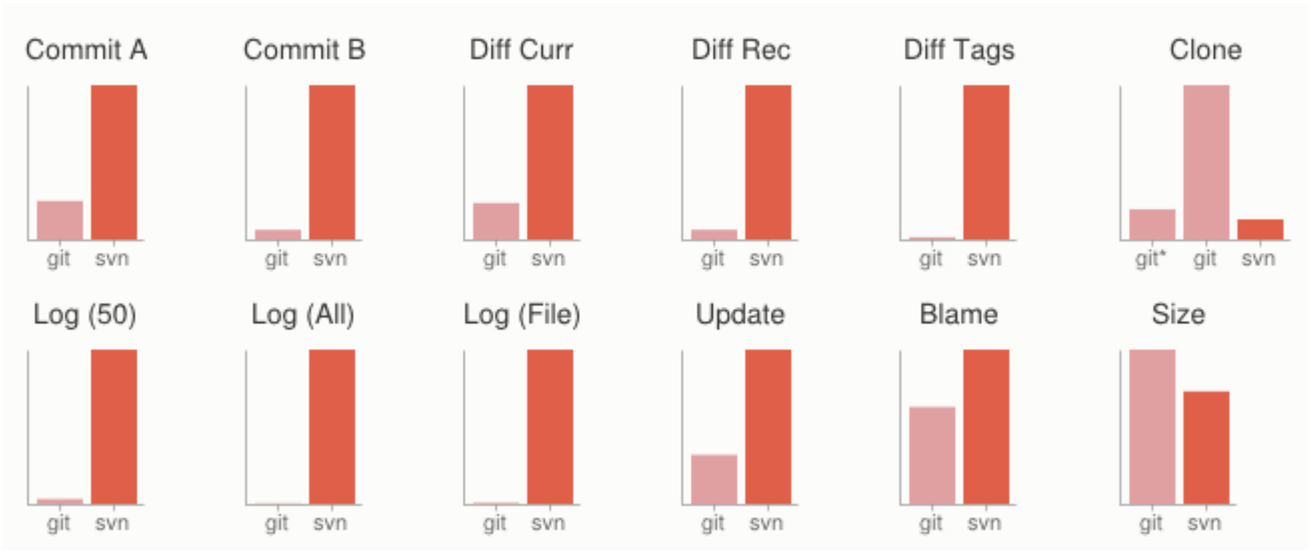
More about

A few useful facts

Git Speed

The only metrics "slower" than svn are

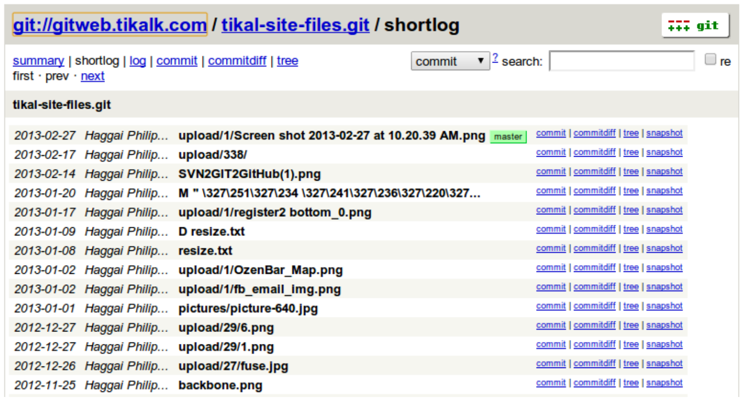
Clone and Size on disk due to the nature of Git which has all the History since the beginning of time ...



Do more with

Backup with Git (or: git not just an SCM)

- I had a based website, with digital assets (png, jpeg, zip files etc) which I needed to backup.
- Website source was in
- The result =>
- Someone deleted a file and needed recovery => It's all in Git's history.



git://gitweb.tikalk.com / tikal-site-files.git / shortlog

summary | shortlog | log | commit | commitdiff | tree
first · prev · next

tikal-site-files.git

2013-02-27	Haggai Philip...	upload/1/Screen shot 2013-02-27 at 10.20.39 AM.png master	commit | commitdiff | tree | snapshot
2013-02-17	Haggai Philip...	upload/338/	commit | commitdiff | tree | snapshot
2013-02-14	Haggai Philip...	SVN2GIT2GitHub(1).png	commit | commitdiff | tree | snapshot
2013-01-20	Haggai Philip...	M " \327\251\327\234 \327\241\327\236\327\220\327...	commit | commitdiff | tree | snapshot
2013-01-17	Haggai Philip...	upload/1/register2 bottom_0.png	commit | commitdiff | tree | snapshot
2013-01-09	Haggai Philip...	D resize.txt	commit | commitdiff | tree | snapshot
2013-01-08	Haggai Philip...	resize.txt	commit | commitdiff | tree | snapshot
2013-01-02	Haggai Philip...	upload/1/OzenBar_Map.png	commit | commitdiff | tree | snapshot
2013-01-02	Haggai Philip...	upload/1/fb_email_img.png	commit | commitdiff | tree | snapshot
2013-01-01	Haggai Philip...	pictures/picture-640.jpg	commit | commitdiff | tree | snapshot
2012-12-27	Haggai Philip...	upload/29/6.png	commit | commitdiff | tree | snapshot
2012-12-27	Haggai Philip...	upload/29/1.png	commit | commitdiff | tree | snapshot
2012-12-26	Haggai Philip...	upload/27/fuse.jpg	commit | commitdiff | tree | snapshot
2012-11-25	Haggai Philip...	backbone.png	commit | commitdiff | tree | snapshot

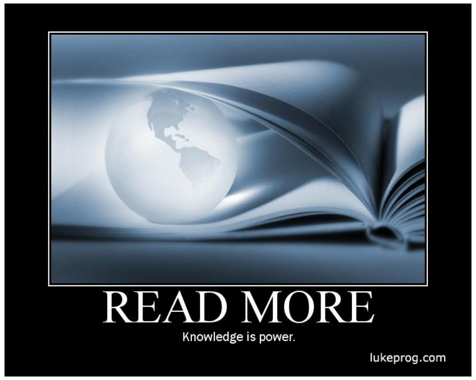
- Rsync my previous method would use --delete which clearly removes older files => lose history of my digital assets ! [**save space**, **gain control over history**, **fast disaster recovery**]
- See Gist: https://gist.github.com/hagzag/5396510

etckeeper

In a nutshell a set of tools which enables one to store /etc/* content into version control.

etckeeper works with git, mercurial, darcs, or bzr [common DVCS systems].

On a change in one of the files the change will be submitted to VCS.

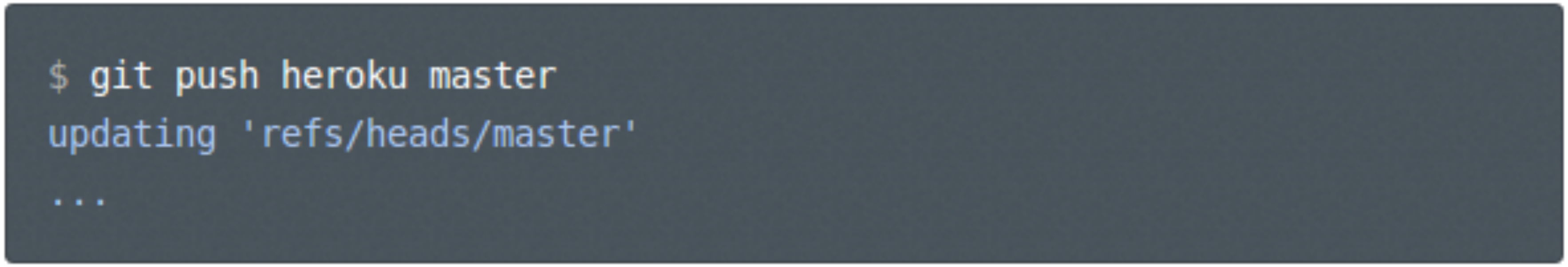


http://joeyh.name/code/etckeeper/

https://help.ubuntu.com/10.04/serverguide/etckeeper.html

Deploying with Git

- A few tracks are available per language
- Remote master = the production which heroku will deploy for you based on Git
- https://devcenter.heroku.com/articles/git



```
$ git push heroku master
updating 'refs/heads/master'
...
```

References

- ProGit: http://git-scm.com/book
- Git Internals: https://peepcode.com/products/git-internals-pdf - well spent 12$ [before git pro existed] ...
- Git-scm.org: http://git-scm.com/documentation
- "Git for Computer Scientists": http://eagain.net/articles/git-for-computer-scientists/

- Icons in this presentation taken from: http://www.icons-land.com/

Haggai Philip Zagury
haggai@tikalk.com

http://www.flickr.com/photos/robnas/3400482826

What's next, you ask

- Git workflows / implementations
- Branching schemes
- Advanced Git topics:
  - rebase
  - cherry picking

http://www.flickr.com/photos/drachmann/327122302/