Understanding Git with Alloy

Milestone 3

Cláudio Lourenço Renato Neves

University of Minho
Formal Methods in Software Engineering

July 10, 2012

Table of contents

Git as VCS

Git is one of many Version Control Systems

- Fast
- Efficient
- Oriented to snapshots, not differences
- Widely used

Motivation for this project

Gap in the understanding of Git

- Lack of precise descriptions
- Contradictions in some manuals
- Developers could benefit from a manual that is precise and rigorous

- "if there are any uncommitted changes when you run git checkout, Git will behave very strangely." [1]
- "When you create a branch, it will contain everything committed on the branch you created it from at that given point. So if you commit more things on the master branch like you have done (after creating b), then switch to branch b, they won't appear. This is the correct behavior. Does that answer your question?" [2]

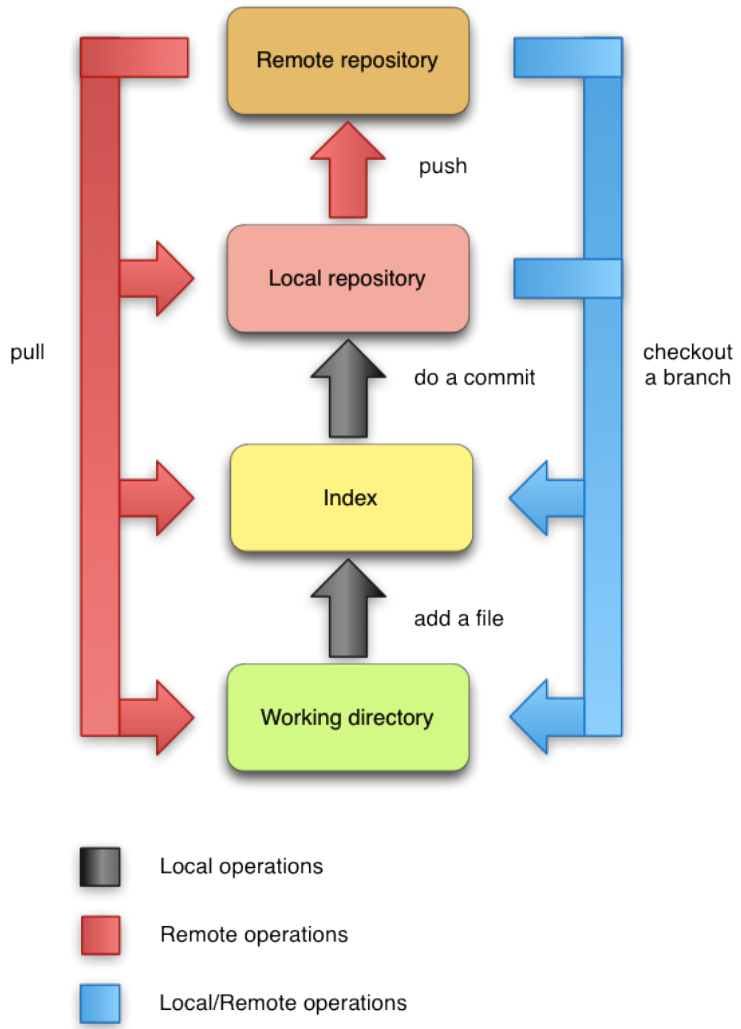
[1] Understanding Git
[2] An average user of Git

Objectives of this project

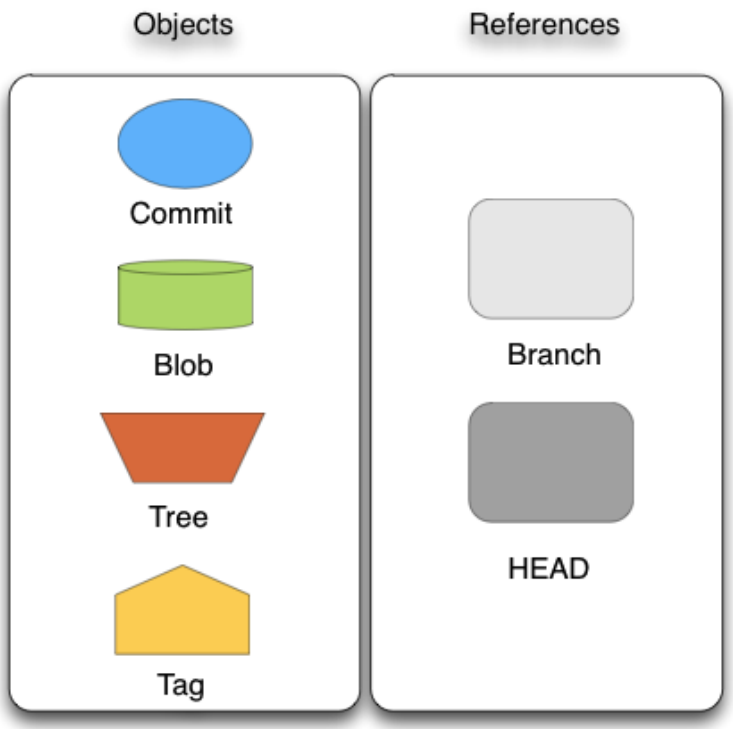
Shine some light in the dark world of Git

- Build a precise model of how Git works
- Analyze the model
- Build a user manual based on specification a analysis

The Git Structure



Repository



Blob and Tree

Blob

- Represents the content of a file;
- The name is calculated from its content;

```
sig Blob extends Object {}
```

Tree

- Relation from names to Blobs or/and Trees;
- Used to represent the file system structure;

```
sig Tree extends Object {
    contains: Name -> lone(Tree+Blob)
}
```

Commit

- It is like a snapshot of the project on a certain moment in time;
- Author, Committer, Comment - Not important for us;
- Parent - The Commit which originated the current;
- Tree - Pointer to a Tree Object;

```
sig Commit extends Object {
    points : Tree,
    parent : set Commit,
    abs: Path -> Object,
    merge : set State
}

sig RootCommit extends Commit {}
```

Branch and HEAD

Branch

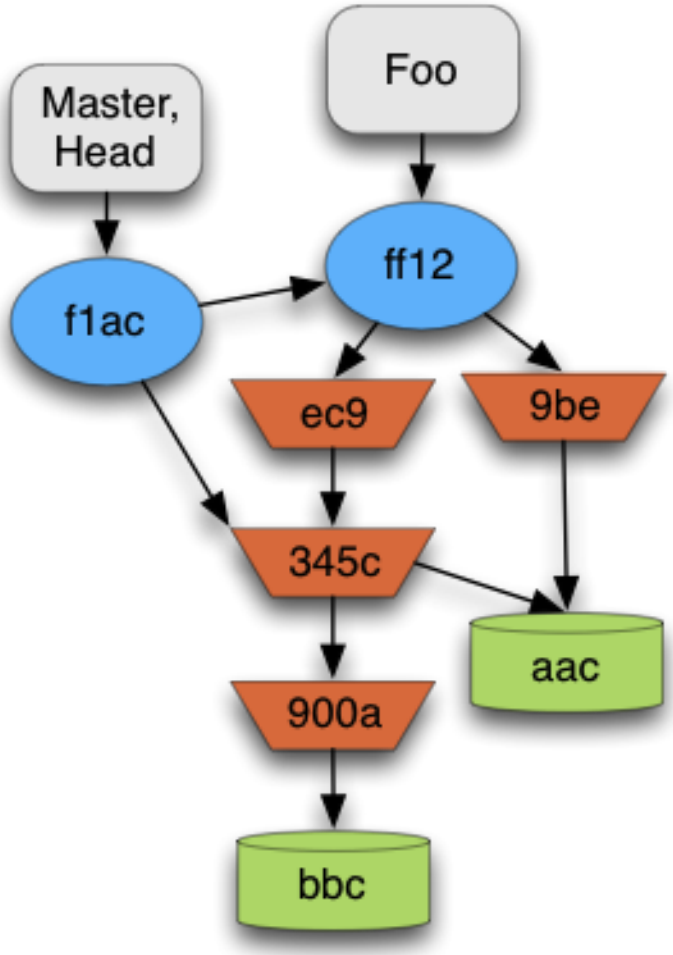
- It is just a pointer to a commit;

HEAD

- Special reference that identifies the current Branch;

```
sig Branch{
    marks: Commit lone -> State,
    branches: set State,
    head: set State
}

lone sig Master extends Branch{}
```

Repository



Working Directory

- Subset of a file system with the content of a project;
- These files can be the current files or files retrieved from the repository.

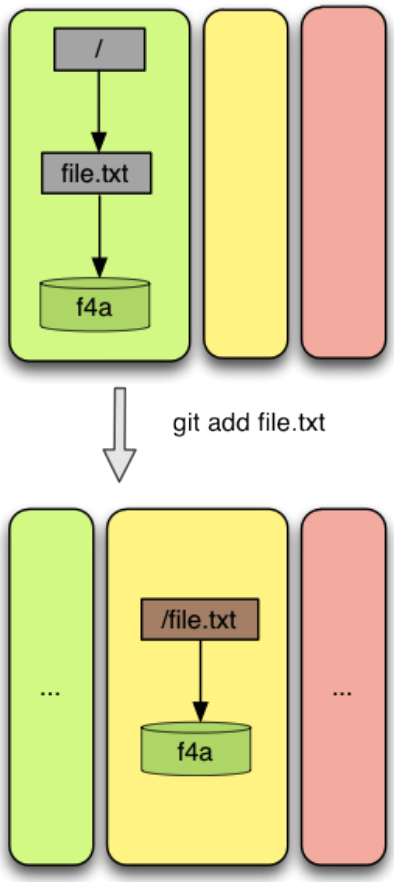
```
sig Path {
    pathparent: lone Path,
    name: Name,
    unmerge: set State
}

one sig Root extends Path{}
```

Index

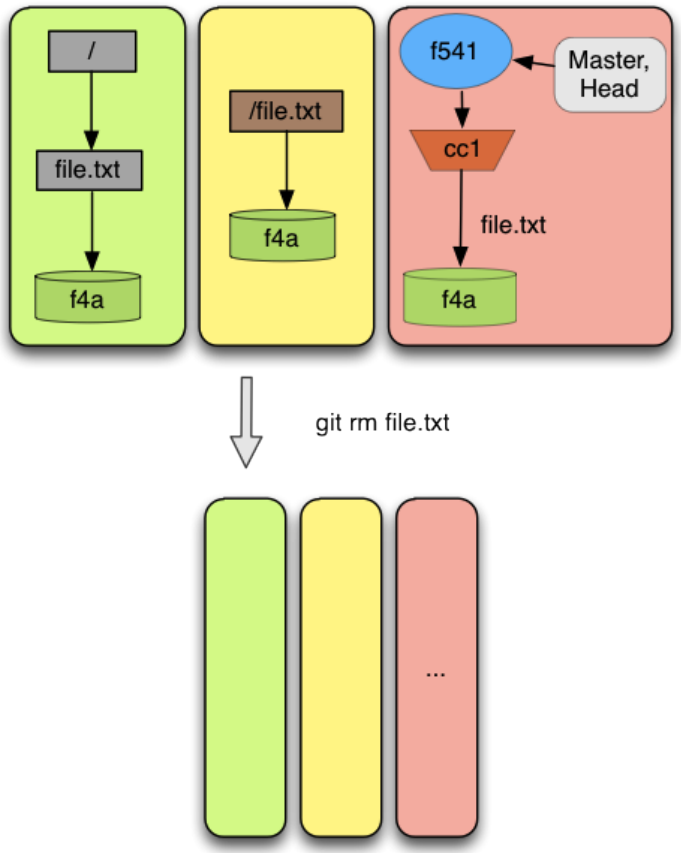
- Something in between the working directory and repository;
- It keeps a relation from file to content;
- The files in index will be in the next commit;

```
sig File{
    path: Path,
    blob: Blob,
    index: set State
}
```

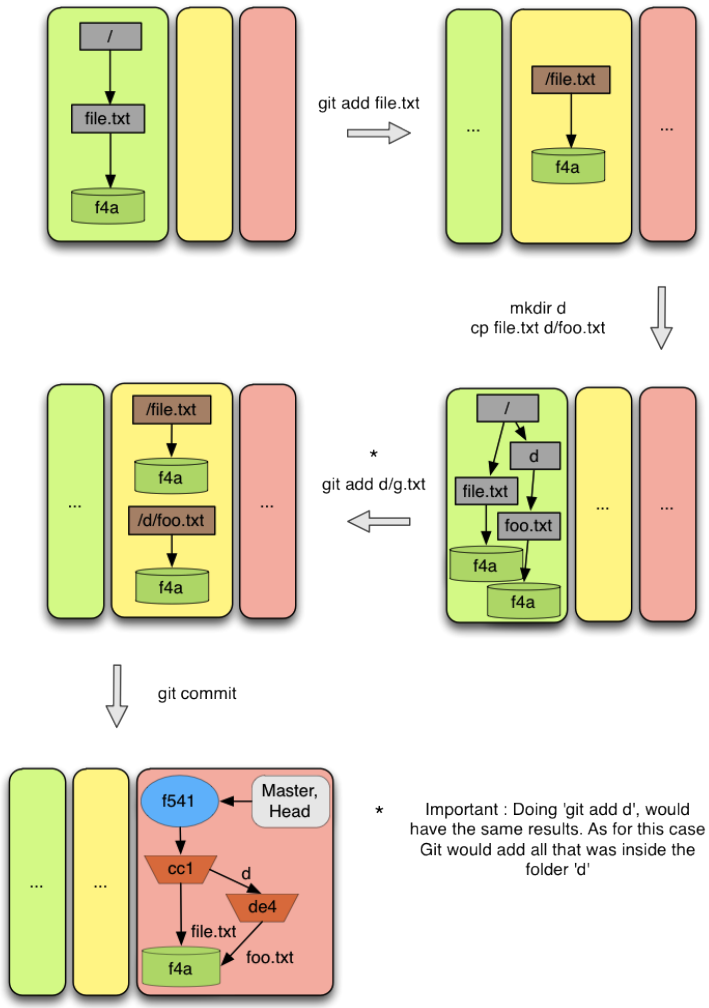
Add



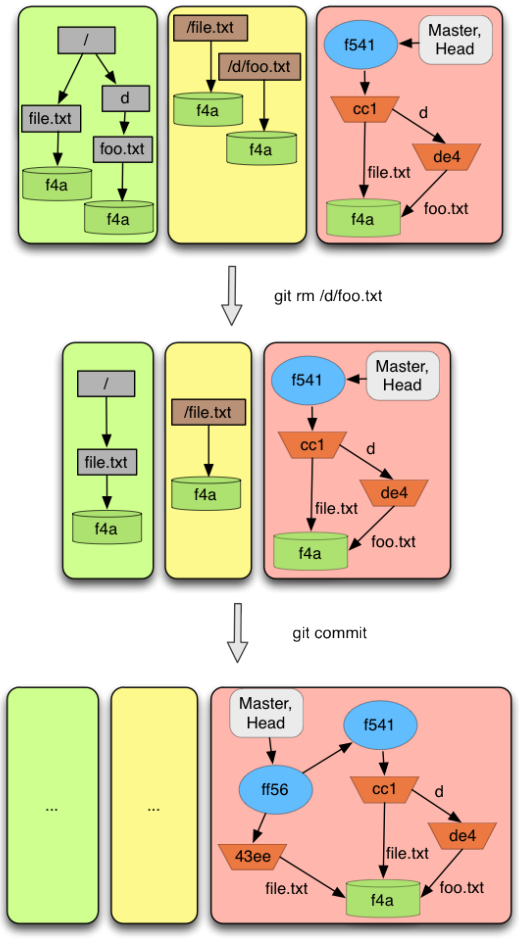
Remove


/
file.txt
f4a
/file.txt
f4a
f541
Master,
Head
cc1
file.txt
f4a
git rm file.txt
...

Commit

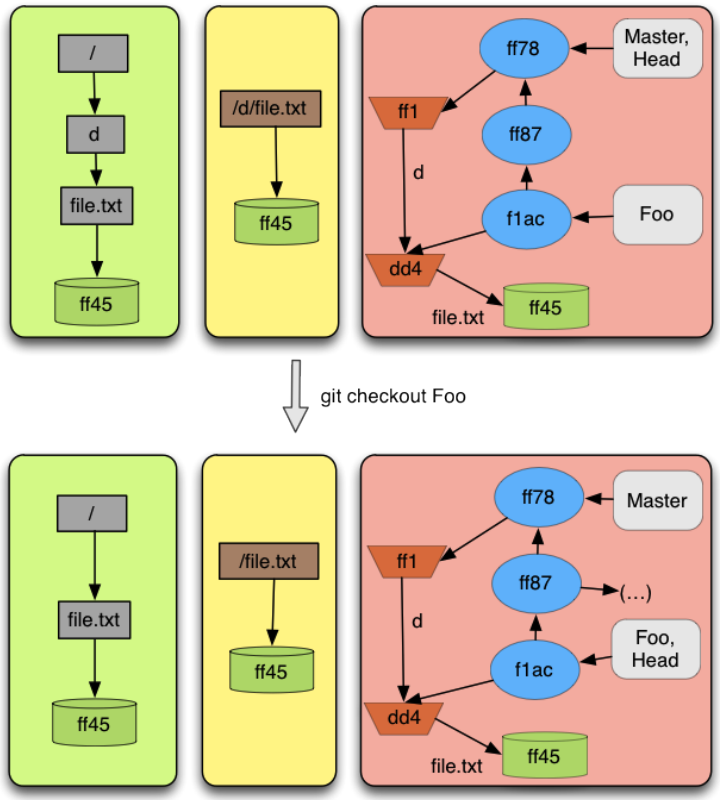

/
file.txt
f4a
git add file.txt
/file.txt
f4a
mkdir d
cp file.txt d/foo.txt
/file.txt
f4a
/d/foo.txt
f4a
* git add d/g.txt
/
d
file.txt
foo.txt
f4a
f4a
git commit
f541
Master, Head
cc1
d
de4
file.txt
foo.txt
f4a
* Important : Doing 'git add d', would have the same results. As for this case Git would add all that was inside the folder 'd'

Commit

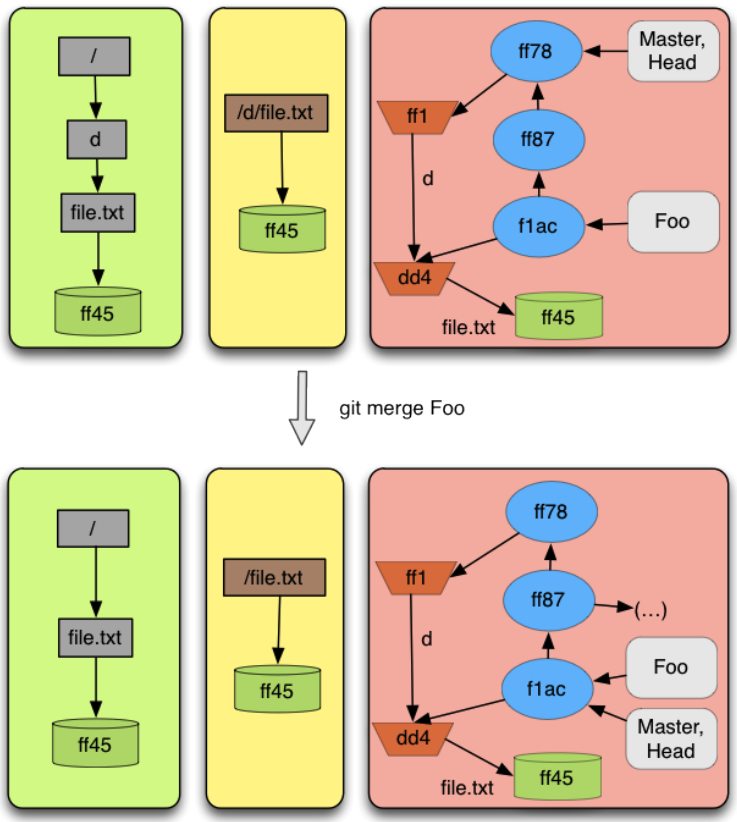


CSAIL

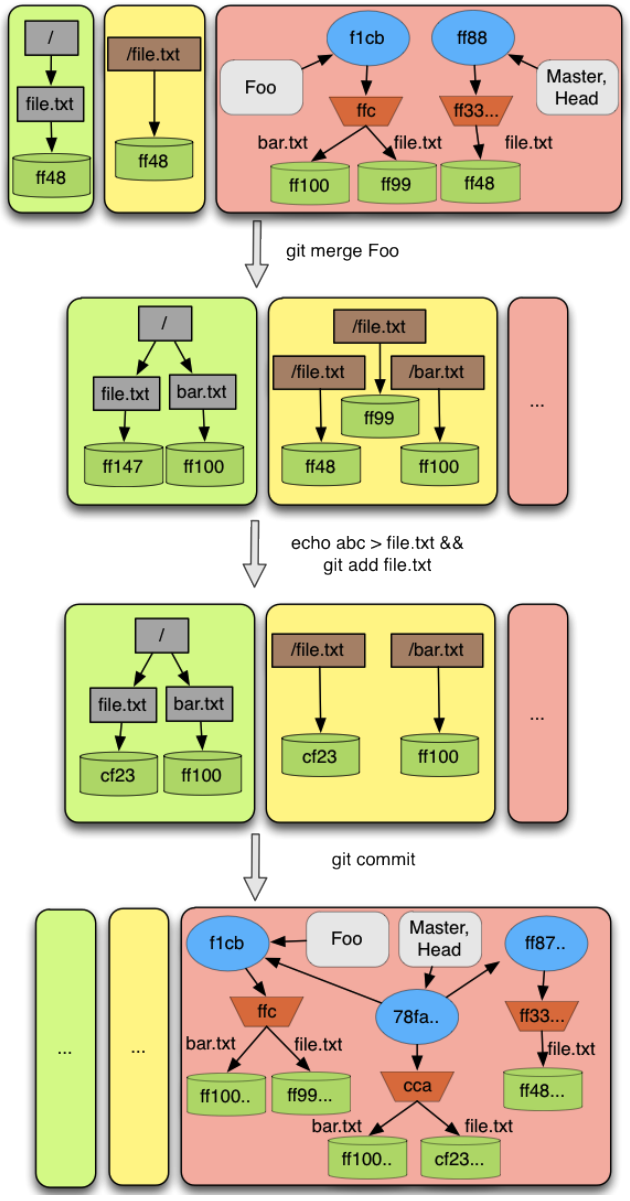
Checkout


/
d
file.txt
ff45
/d/file.txt
ff45
ff78
Master, Head
ff1
ff87
d
f1ac
Foo
dd4
file.txt
ff45
git checkout Foo
/
file.txt
ff45
/file.txt
ff45
ff78
Master
ff1
ff87
(...)
d
Foo, Head
f1ac
dd4
file.txt
ff45

A fast-forward Merge



A 2-way Merge



CSAIL

Modeled Operations

- Add and Remove
- Commit
- Branch and Branch Remove
- Checkout
- Merge (2-way and fast-forward)

Manual

Built a manual that describes

- Git internals
- Git operations

Website

- Website created based on the manual
- http://nevrenato.github.com/CSAIL_Git

Future Work

- Model more operations (rebase, fetch, 3-way merge...)
- Specify more properties that the model does (not) guarantee
- Build interactive diagrams of concrete examples of operations

Conclusions