How Git Works

Our focus on this module will be:

- Internals of Git
- How Git saves content
- Low level Git commands

"The secret to learning git is about knowing the conceptual models behind Git."

Git is like an onion

- Imagine git is an onion
- We will understand it one layer at a time

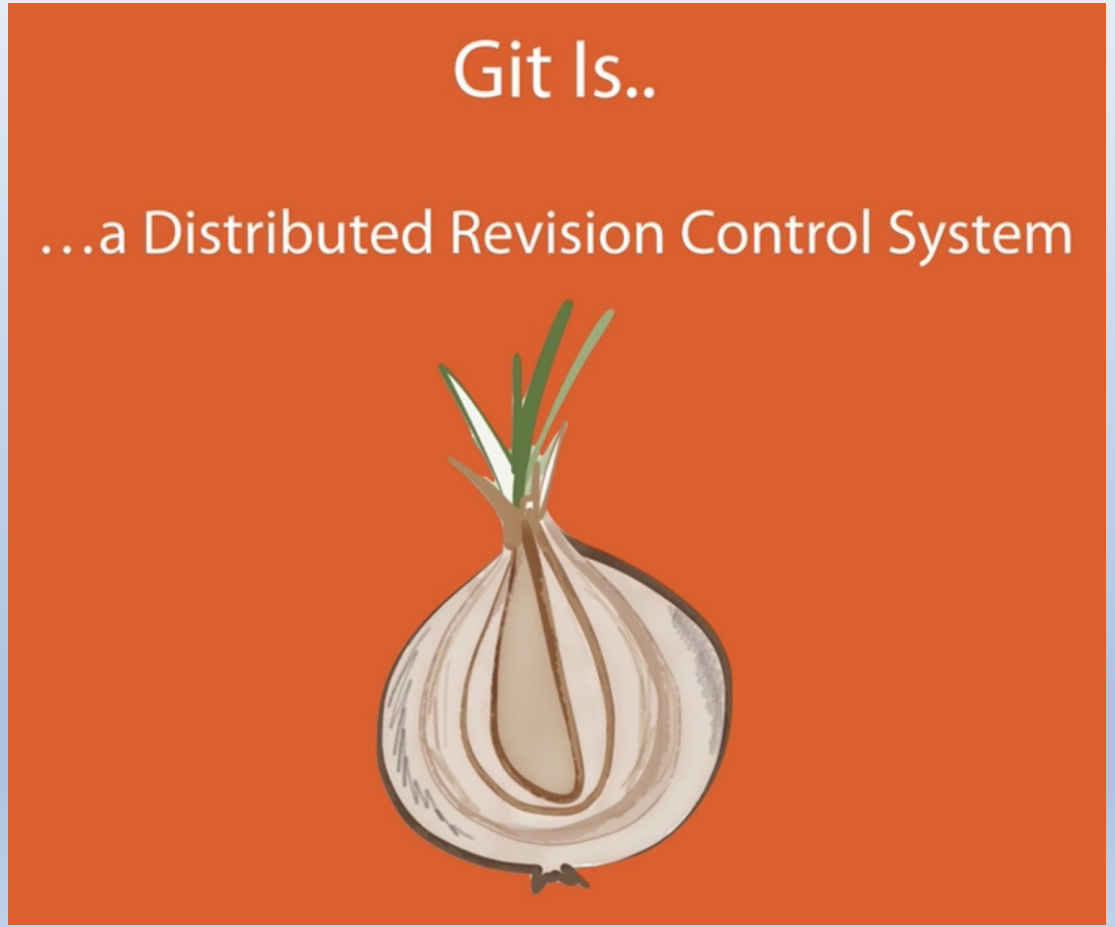
"Porcelain" Commands

- git add
- git commit
- git push
- git pull
- git branch
- git checkout
- git merge
- git rebase
- …

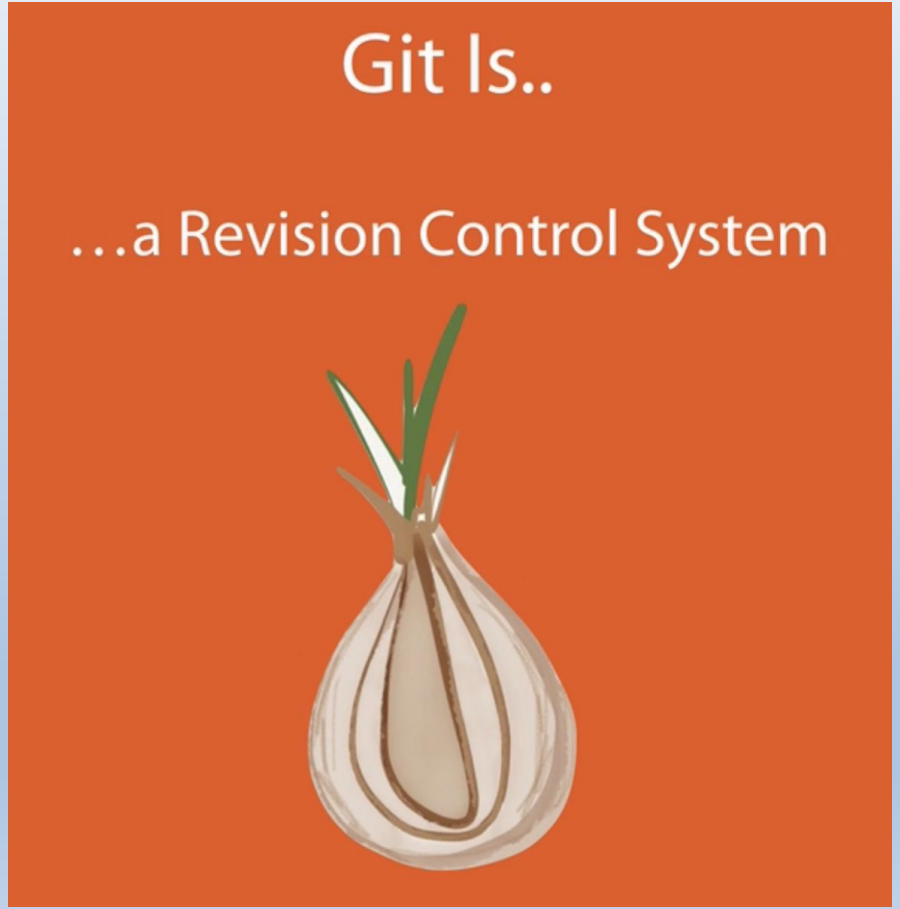
“Plumbing” Commands

- git cat-file
- git hash-object
- git count-objects
- …

Git is like an onion

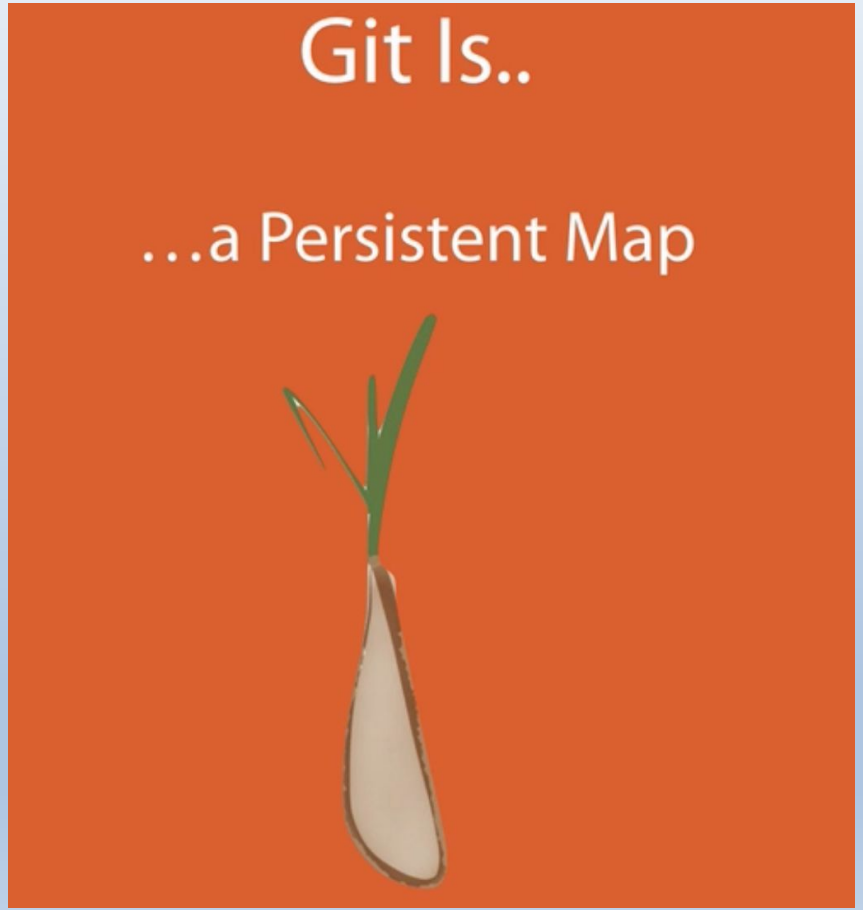


Git is like an onion



Git is like an onion

Git is like an onion



A map of key value pairs

- At its core, Git is just a persistent map of key value pairs
- Where, value can be the content of text/binary file
- and key is the hash of that content

SHA1

- SHA1 is a cryptographic hash function.
- It's result is usually expressed as a 160 bit hex number.
- SHA1 was developed by the NSA.
- SHA1 is widely considered the successor to MD5.

Git and SHA1

- Give a value (a text or a file.. any content) to git and it will calculate a key for it using SHA1 hash
- This hash key is persistent
- Key changes when the content changes

Git and SHA1 (Example)

```
echo "Hello, World" | git hash-object --stdin

git hash-object --stdin <<< "Hello, World"
```

3fa0d4b98289a95a7cd3a45c9545e622718f8d2b

Store content using git

```
echo "Hello, World" | git hash-object --stdin -w
```

- Create a sample git repo directory.
- Initialize repository:

```
cd gitrepo

git init
```

Store content using git (before)

```
[vagrant@localhost gitrepo]$ tree .git/
.git/
├── branches
├── config
├── description
├── HEAD
├── hooks
├── info
│   └── exclude
├── objects
│   ├── info
│   └── pack
└── refs
    ├── heads
    └── tags

9 directories, 13 files
```

Store content using git (after)

```
[vagrant@localhost gitrepo]$ tree .git/
.git/
├── branches
├── config
├── description
├── HEAD
├── hooks
├── info
│   └── exclude
├── objects
│   ├── 3f
│   │   └── a0d4b98289a95a7cd3a45c9545e622718f8d2b
│   ├── info
│   └── pack
└── refs
    ├── heads
    └── tags
```

Store content using git (after)

- First characters of the hash were used in creating a directory under objects.
- It's a trick to avoid piling up all the content into a single huge clutter.

```
├── objects
|   ├── 3f
|   |   └── a0d4b98289a95a7cd3a45c9545e622718f8d2b
```

Store content using git (after)

- Our content is inside the file with hash name.
- Git calls this content file, a blob.
- Git compresses the file so we can't directly open it
- Low level command to cat file content:

  git cat-file <hash> <command-options>

Print stored content

- git cat-file -t : to print content type

  git cat-file 3fa0d4b98289a95a7cd3a45c9545e622718f8d2b -t

- git cat-file -p : to print content of file

  git cat-file 3fa0d4b98289a95a7cd3a45c9545e622718f8d2b -p

First Commit

```
$ tree cookbook/

cookbook/

├── menu.txt

└── recipes

    ├── apple_pie.txt

    └── README.txt
```

cd cookbook; git init

git status

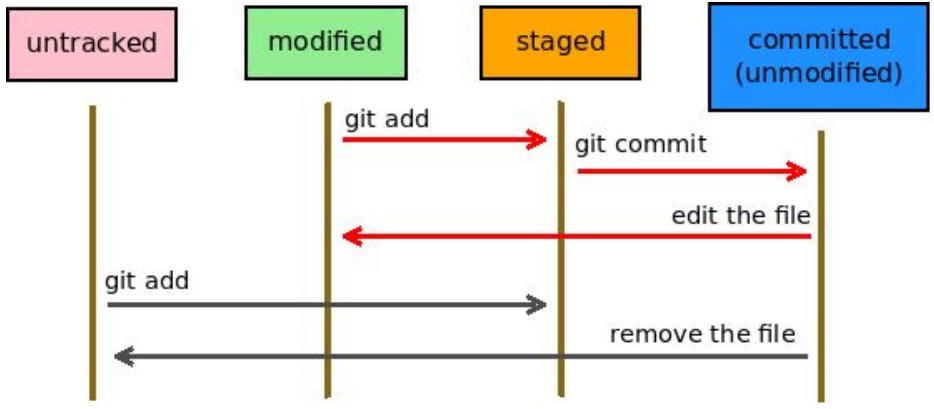
The ~~four~~ three states

Untracked – you have created a new file

Modified – you have changed the file, but not marked to be committed

Staged – you marked a modified file to be committed

Committed – data is safely stored in your local database



- Set your **username**: **git config --global user**.**name** "FIRST_NAME LAST_NAME"
- Set your email address: **git config --global user**.email "MY_NAME@example.com"

Stage files for commit

- Git will only commit files and directories that are staged.
- Stage files using command:

```
git add <file/directory>

git add menu.txt

git add recipes/
```

First Commit

- Commit files using command:

  git commit -m “My first commit”

- Check logs of the commits using command:

  git log

First Commit

```
$ tree .git
├── objects
│   ├── 23
│   │   └── 991897e13e47ed0adb91a0082c31c82feocbe5
│   ├── 3d
│   │   └── 5c36edff14e8091cd7cd89432655d074aa57a5
│   ├── 7a
│   │   └── 1193f27083a97c8b22cad0a658dd35efe09c39
│   ├── 7c
│   │   └── 58885ba801d17fc67e76a89afa4fb9faeaf01c
│   ├── e6
│   │   └── 9de29bb2d1d6434b8b29ae775ad8c2e48c5391
```

First Commit

- Use git cat-file to print commit file:

```
$ git cat-file 7c58885ba801d17fc67e76a89afa4fb9faeaf01c -p

tree 7a1193f27083a97c8b22cad0a658dd35efe09c39
author vagrant <vagrant@localhost.localdomain> 1523669977 +0000
committer vagrant <vagrant@localhost.localdomain> 1523669977 +0000

My first commit
```

First Commit

- tree is pointing to the root directory of the project.
- Print tree hash to see what's inside its file:

```
$ git cat-file 7a1193f27083a97c8b22cad0a658dd35efe09c39 -p
100644 blob 23991897e13e47ed0adb91a0082c31c82feocbe5 menu.txt
040000 tree 3d5c36edff14e8091cd7cd89432655d074aa57a5 recipes
```

First Commit

commit file (381a48e)

→ tree hash file (84fe5)

→ menu.txt (81c1f)

→ recipes/ (b7927)

→ README.txt (5579)

→ apple_pie.txt (81c1f)

Second Commit

- Write Apple Pie in both menu.txt and apple_pie.txt file.
- Then commit the files again.
- Now git log and
- git cat-file new commit to see the tree file

Second Commit

- git cal-file new tree

```
$ git cat-file -p d3994c91521036b6f0e359821f5705db0d5c44e7
100644 blob 23991897e13e47e0adb91a0082c31c82fe0cbe5    menu.txt
040000 tree a5b109541561e04cc4504d6d6e17d7bc6306f690    recipes
```

- git cal-file recipes directory

```
$ git cat-file -p a5b109541561e04cc4504d6d6e17d7bc6306f690
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391    README.txt
100644 blob 23991897e13e47e0adb91a0082c31c82fe0cbe5    apple_pie.txt
```

Notice the same SHA1 hash of menu.txt and apple_pie.txt

Second Commit

commit file (f01c)

→ tree hash file (9c39)

→ menu.txt (cbe5)

→ recipes/ (57a5)

→ README.txt (5391)

→ apple_pie.txt (cbe5)

The Whole Object Database.

Versioning made easy

- Now append another recipe to menu.txt and stage the file for commit.
- Then commit the file.
- Let's see the commit logs again:

  git log

Versioning made easy: Third commit

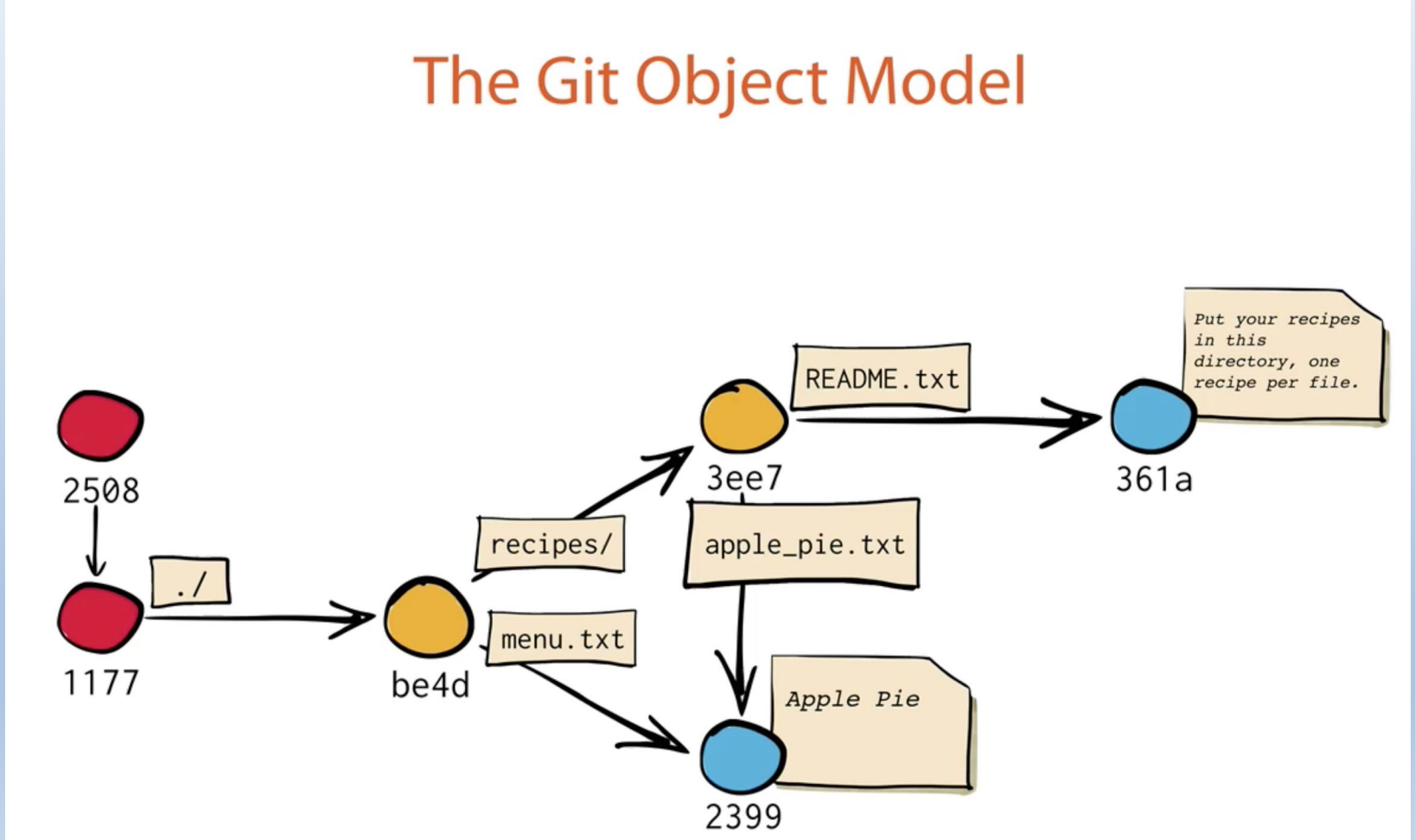
```
$ git log
commit cce0cbdd3b55b8e5a3a17ce5bbd2737987892b10
Author: Anil Pemmaraju <apemmaraju@nisum.com>
Date:   Sat Apr 14 04:51:05 2018 +0000

    Adding cheesecake to recipes list

commit 770137950045d2d577000e06d3edcce66921223c
Author: Anil Pemmaraju <apemmaraju@nisum.com>
Date:   Sat Apr 14 02:01:32 2018 +0000

    Second commit

commit 7c58885ba801d17fc67e76a89afa4fb9faeaf01c
Author: vagrant <vagrant@localhost.localdomain>
Date:   Sat Apr 14 01:39:37 2018 +0000

    My first commit
```

Versioning made easy: Third commit

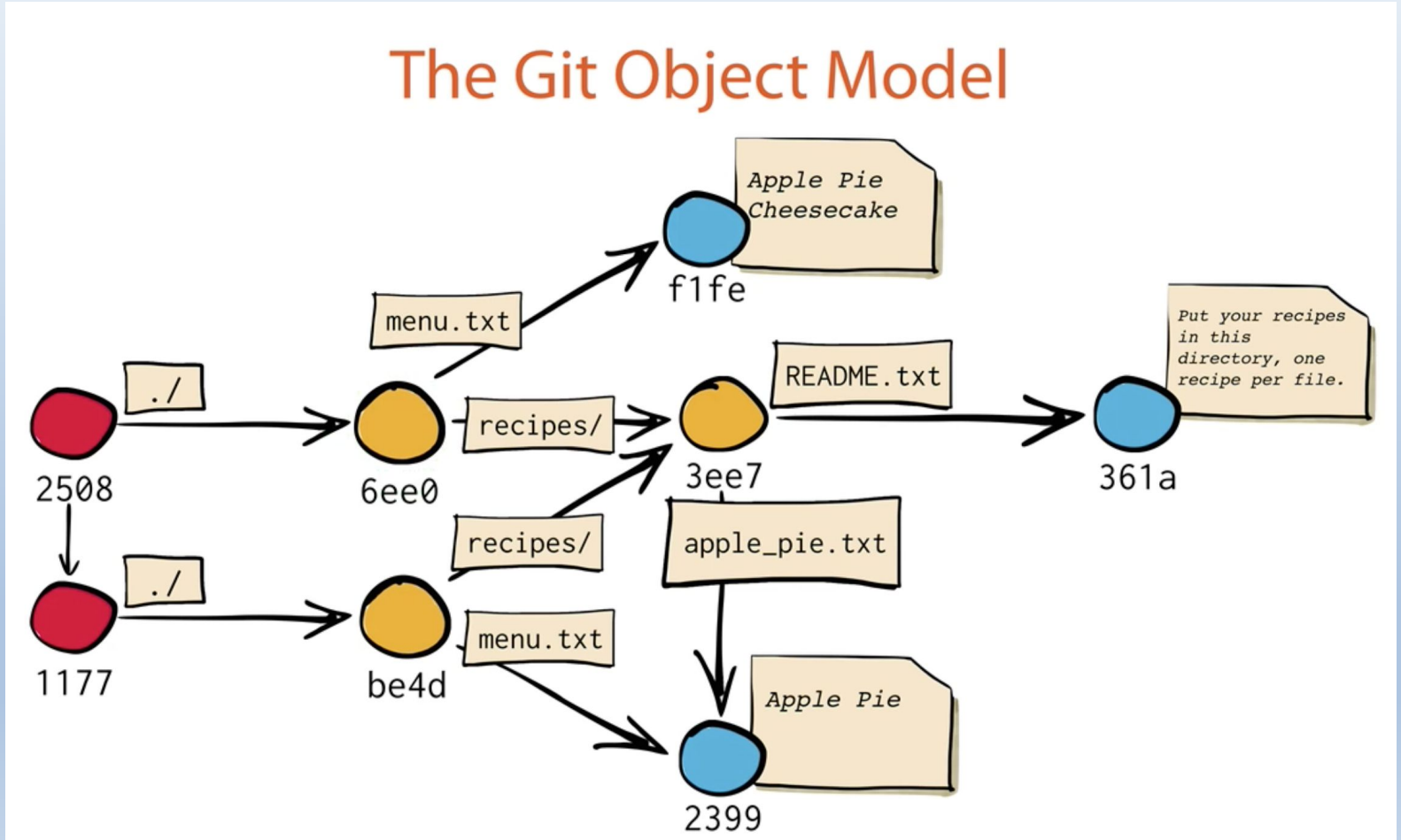
```
$ git cat-file -p cce0cbdd3b55b8e5a3a17ce5bbd2737987892b10

tree af19bfadbc7509d38e9ad3d175c2b496d2cecb9d
parent 770137950045d2d577000e06d3edcce66921223c
author Anil Pemmaraju <apemmaraju@nisum.com> 1523681465 +0000
committer Anil Pemmaraju <apemmaraju@nisum.com> 1523681465 +0000

Adding cheesecake to recipes list
```

- Tree is the directory tree of project.
- Parent is the commit before this one.
- First commit doesn’t have a parent hash.

Versioning made easy: Third commit



Versioning made easy: New tree



Versioning made easy: New tree

- Because 3rd commit has modified version of menu.txt
- This is why 3rd commit has a different tree than the previous commit
- The new tree is pointing to new menu.txt while using old recipes/ directory tree hash.

Total objects in the Object Database

- Use command:

  $ git count-objects

Efficiencies of Object Database

- It compresses files.
- Deduplicate content (keep only 1 copy of the file).
- It may store only difference of the file when modifying it.

  (optimization)
- It uses info and pack directories for this kind of stuff.

Tags

- An object type.
- A tag is like a label for the current state of the repository.
- There are two type of tags:
  - Regular tags
  - Annotated tags

Annotated Tags

- It's a tag with a comment or a message.
- Example:

  git tag -a mytag -m "I love cheesecake"

  git tag : to list tags

Annotated Tags

- You can use tag's name when printing its file using git cat-file command. Contains Meta-data

```
$ git cat-file -p mytag
object cce0cbdd3b55b8e5a3a17ce5bbd2737987892b10
type commit
tag mytag
tagger Anil Pemmaraju <apemmaraju@nisum.com> 1523683099 +0000


I love cheesecake
```

Annotated Tags

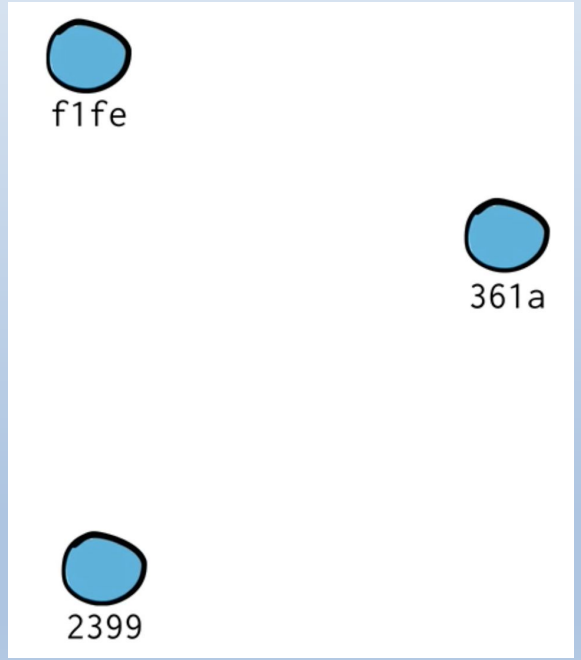
```
$ git cat-file -p mytag
object cce0cbdd3b55b8e5a3a17ce5bbd2737987892b10
type commit
tag mytag
tagger Anil Pemmaraju <apemmaraju@nisum.com> 1523683099 +0000

I love cheesecake
$ git log -1
commit cce0cbdd3b55b8e5a3a17ce5bbd2737987892b10
Author: Anil Pemmaraju <apemmaraju@nisum.com>
Date:   Sat Apr 14 04:51:05 2018 +0000

    Adding cheesecake to recipes list
```

Git Object types

- Tags

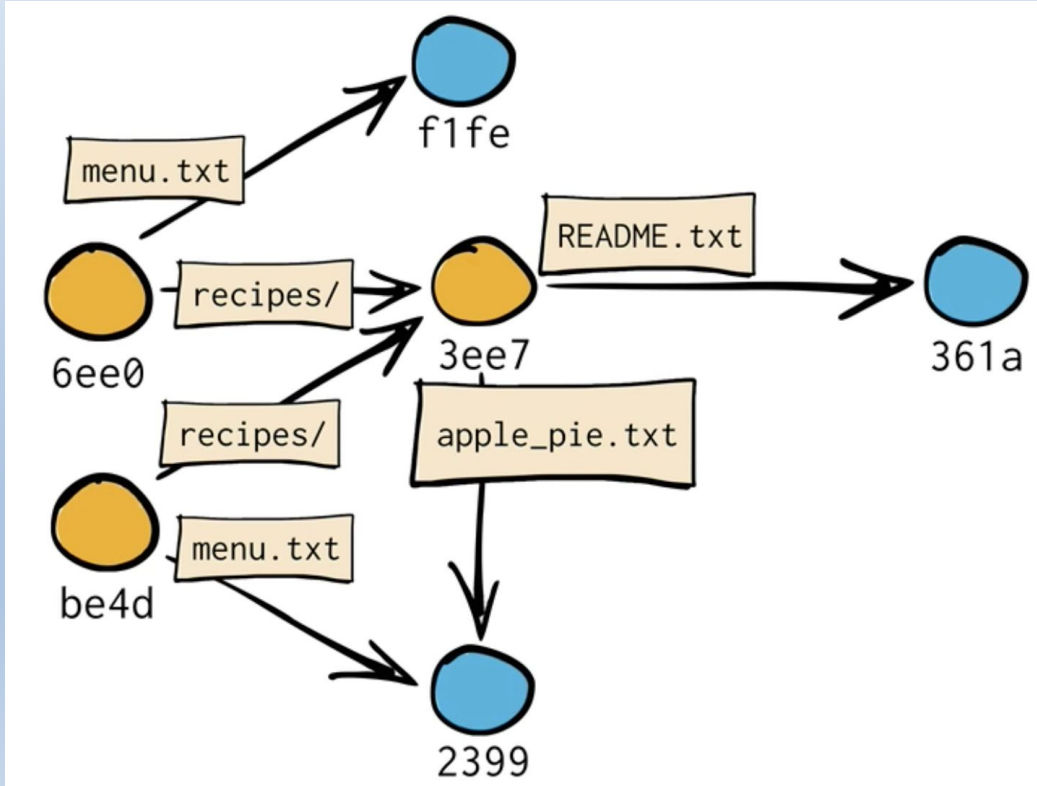
What Git Really Is

- It contain Blobs (files or content)



What Git Really Is

- Names of the objects are not stored in blobs or trees.
- They are stored in the trees that contains them.



What Git Really Is: Git vs Filesystem

- This structure is similar to filesystem.
- Files doesn't contain their names.
- Directories does.

What Git Really Is: Git vs Filesystem

- Git has object database while filesystem has inode table.
- Git has SHA1 hashes while filesystem has inodes.
- Git has tags while filesystem has symlinks
- **Git is a versioned filesystem because it has commits**

What Git Really Is