clearvision

Module 2

Git Concepts

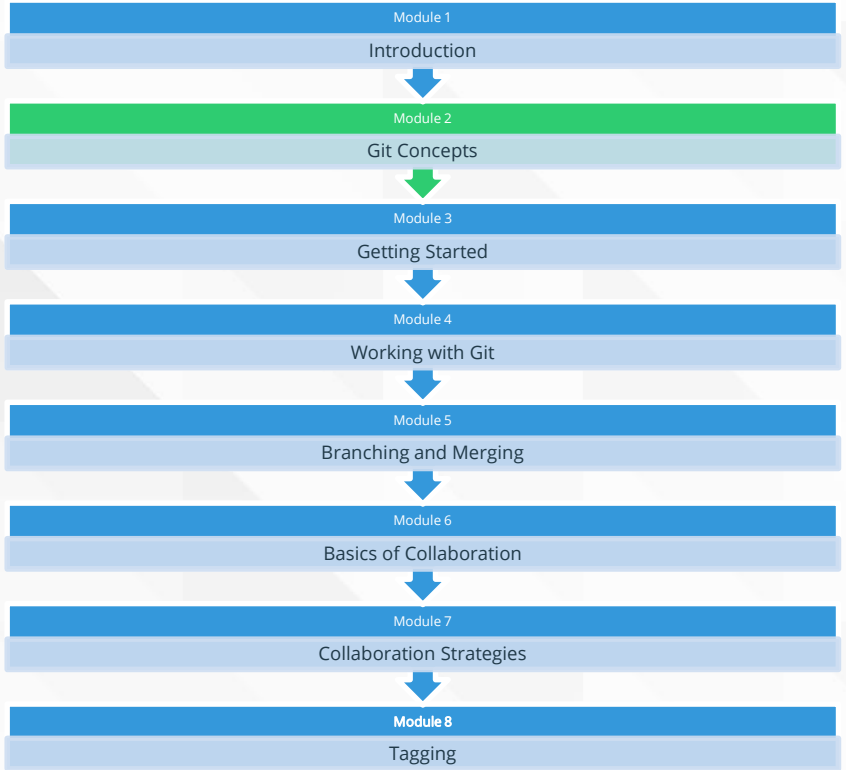
git

Course Modules

clearvision

Module Contents

- The Git Data Model
- Hash Values (SHA-1)
- Git Object Types
- Basic Configuration



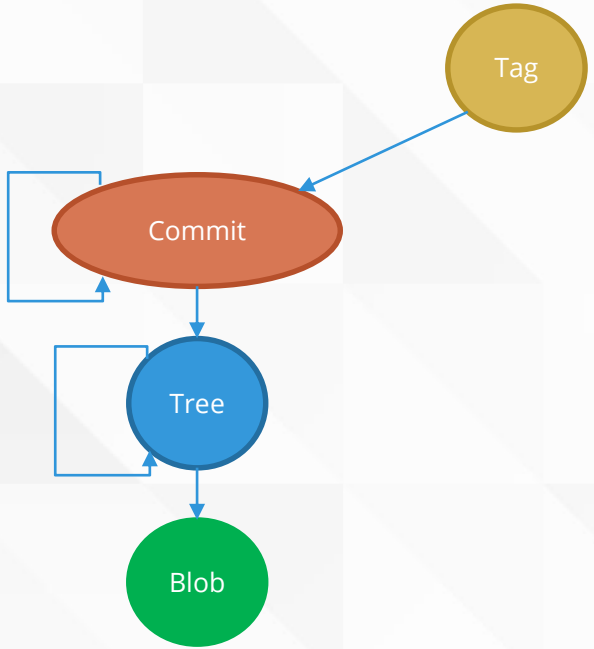
git

Git Object Types

clearvision

The Git object database stores four different object types :

- Blob
- Tree
- Commit
- Tag



The next few slides will Cover these objects in more Detail.

git

Git Object Types - Blob

clearvision

A Blob object contains the contents of a file in the repository stored in compressed form.

On its own, a Blob is meaningless as there is no reference to the file from which it originated - or how it relates to any other object in the system.

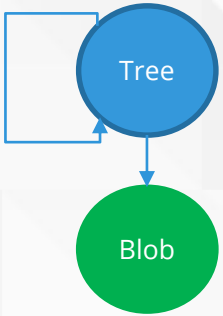
git

Git Object Types - Tree

clearvision

A Tree object is equivalent to a 'directory' and contains a list of Blobs and other Tree objects

These are recorded as pointers along with the identity of the Blob(s) or Tree(s) to which they refer.



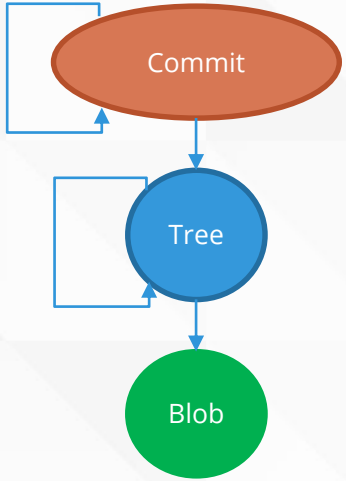
git

Git Object Types - Commit

clearvision

A Commit object provides the snapshot image of a particular directory tree along with additional metadata including :

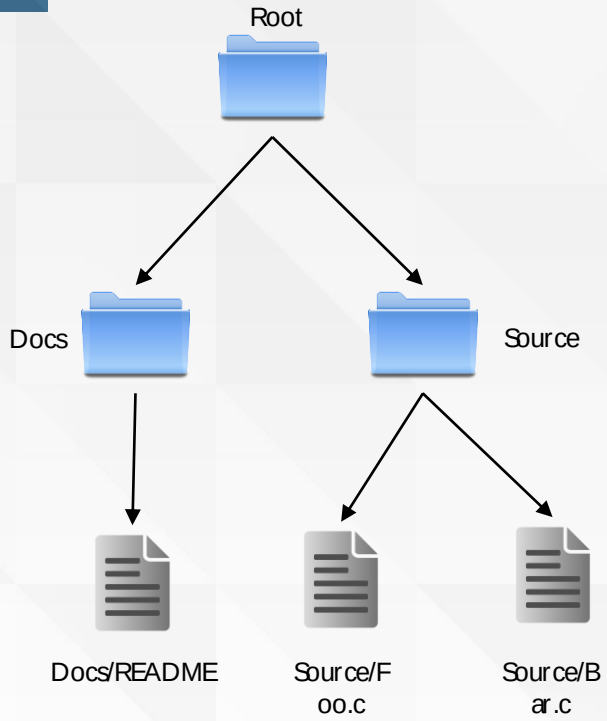
- A time stamp
- The Author
- The Committer
- A log message
- The identities of any parent commit objects.

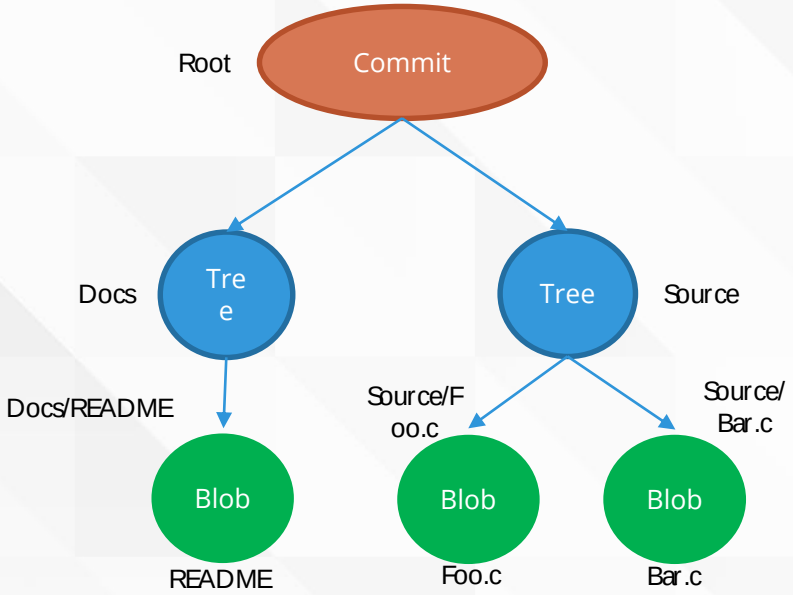


git

Git Object Types

clearvision





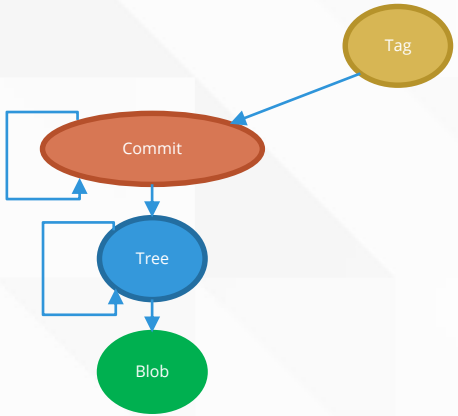
git

Git Object Types - Tag

clearvision

A Tag object is a container to hold a reference to a Commit along with additional meta data including

- The Tag name
- The tagger
- A Tag description.



Most commonly a Tag object is used to store release data pertaining to particular commit objects. Usually including a more meaningful name than it's unique identity – e.g. “v2.0”

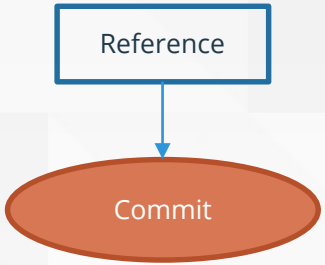
git

References

clearvision

In addition to the core data structures of and the object database and the index there is an mechanism called a reference.

A reference points to a SHA-1 hash identity of a particular commit – for example, the most recent version of our cv.

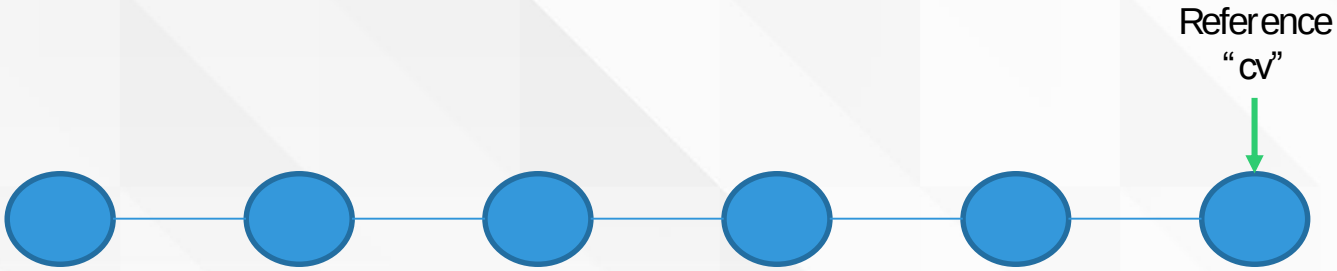


git

References

clearvision

- Thinking back to our earlier example, references are used to track the latest changes that occur over time



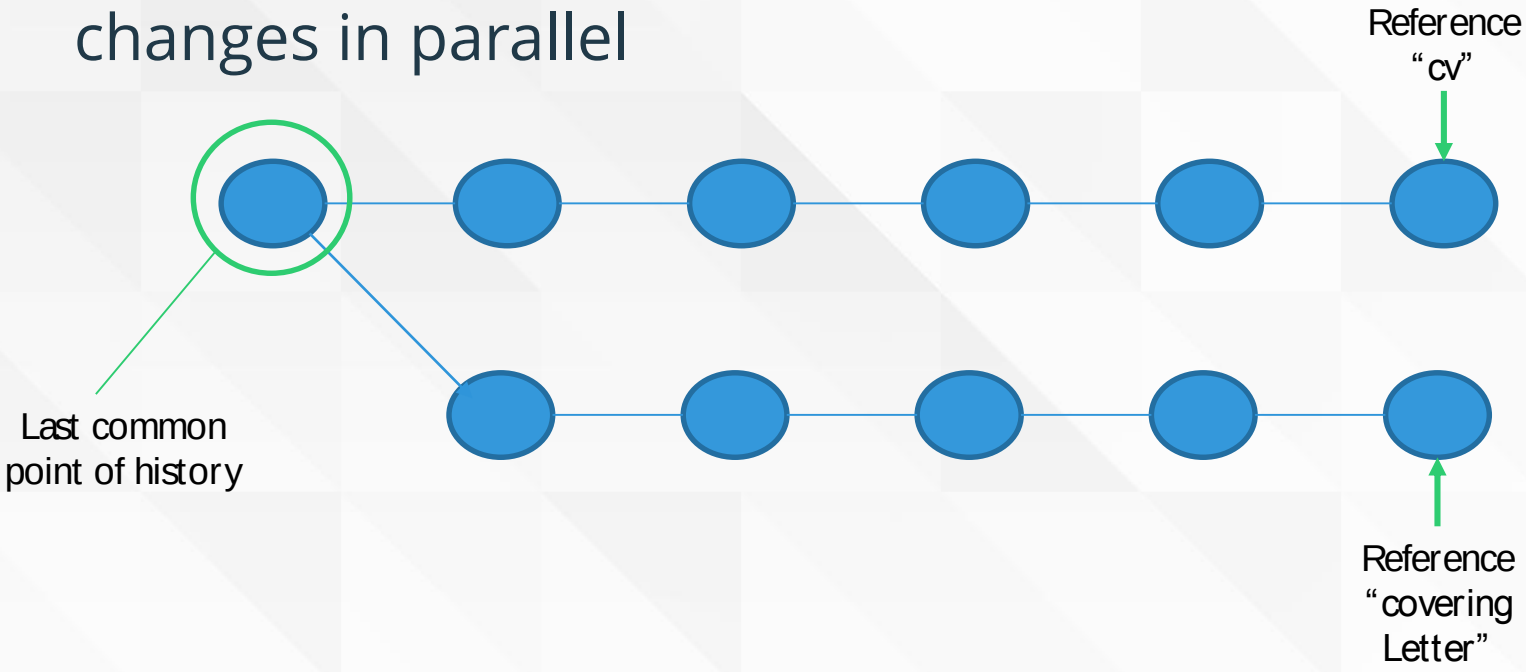
- Once we know the latest, we can use the history to trace back earlier versions of our data.

git

References

clearvision

- Multiple references can be used to track different changes in parallel

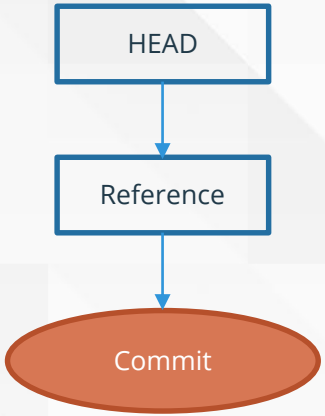


git

References – HEAD revision

clearvision

There is also one final standard reference - the HEAD which contains a pointer to the location you are currently working (e.g. a branch)



git

The Git Data Model - Hashing

clearvision

Every object versioned through git is generated a unique 40 character hash reference known as the SHA (pronounced shar).

- These hash references are used by git to store data as binary objects.

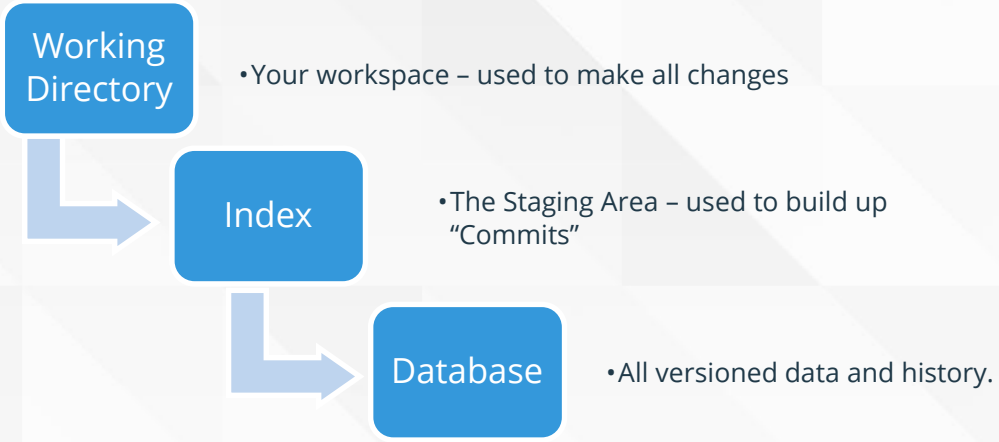
git

The Git Data Model

clearvision

Git essentially stores data in two structures.

- A changeable index. (deltas)
- An immutable database. (objects)



git

Git - Basic Setup

clearvision

- There are a number of basics you should set up before using Git
- Remember the metadata required by a commit?
  - Author
  - Timestamp
  - Committer
  - Commit message
  - Parent
- Author details include…
  - Name
  - Email address

git

Git Configuration

clearvision

To tell git who you are we use the *git config* command.

The three most commonly used actions are :

- git config --global user.name "your name“
- git config --global user.email your@email.com
- git config --global core.editor “path/to/editor”

Note : “--global” denotes a user setting, leaving it out denotes a repository setting

git

clearvision

Lab Exercise

- "Configurations" activity
- "Git objects" activity
- Complete Lab exercise "Git concepts"

git