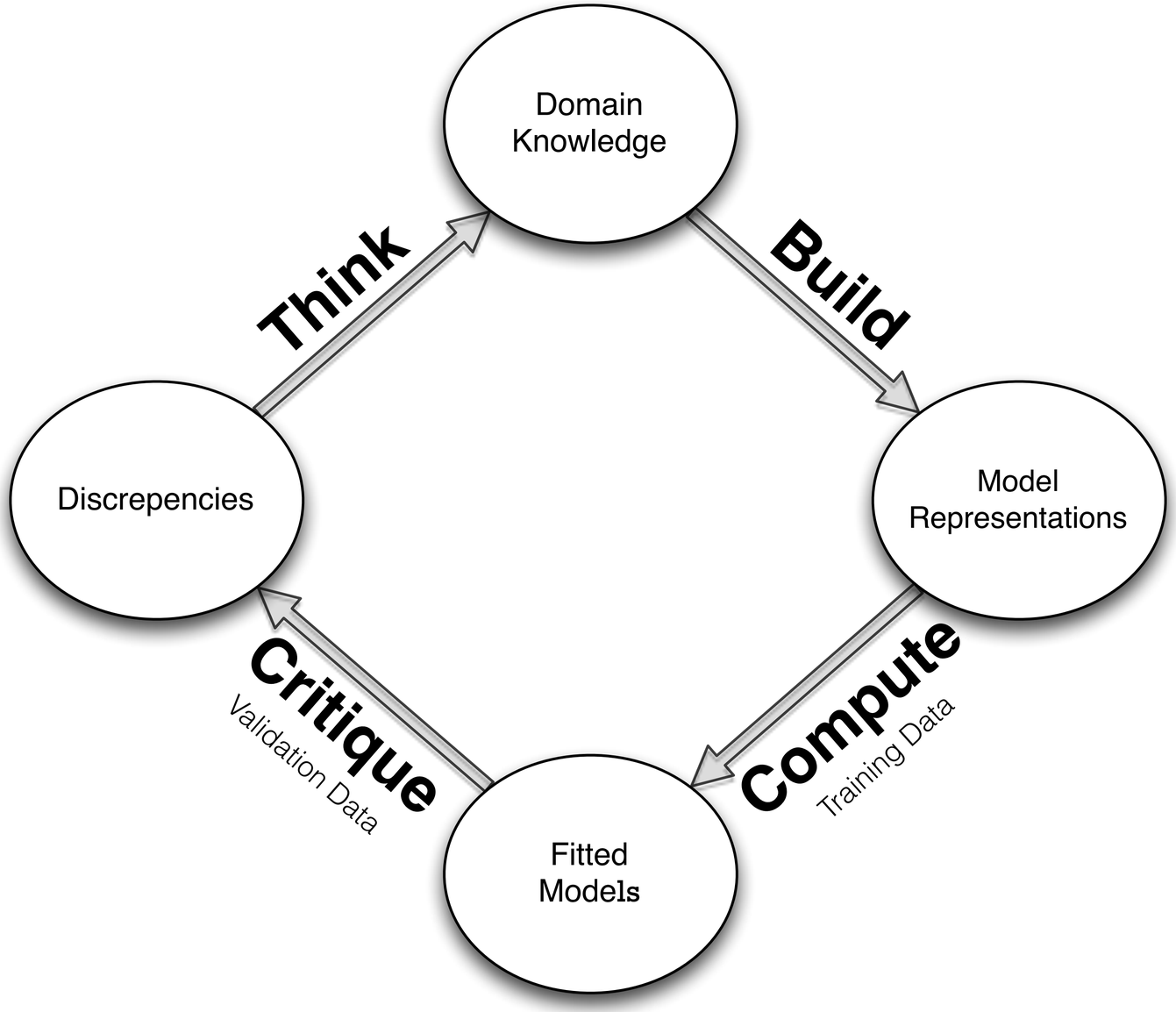
Git -eration

Michael Colaresi

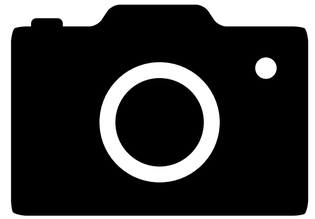
Science proceeds through iterative learning

Trial, error, thinking, correction, repeat



Static Files and Directories

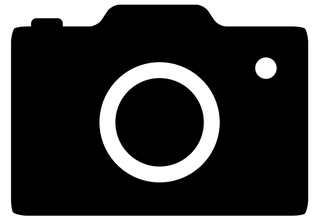
The snapshot metaphor



- Files —> a bundle of bits
- Directories —> tree structure of references/“locations”
- yaml objects
  - strings
  - maps/dictionary
  - sequences/lists

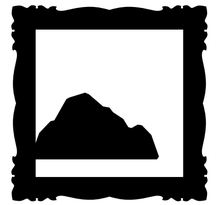
Static Files and Directories

The snapshot metaphor



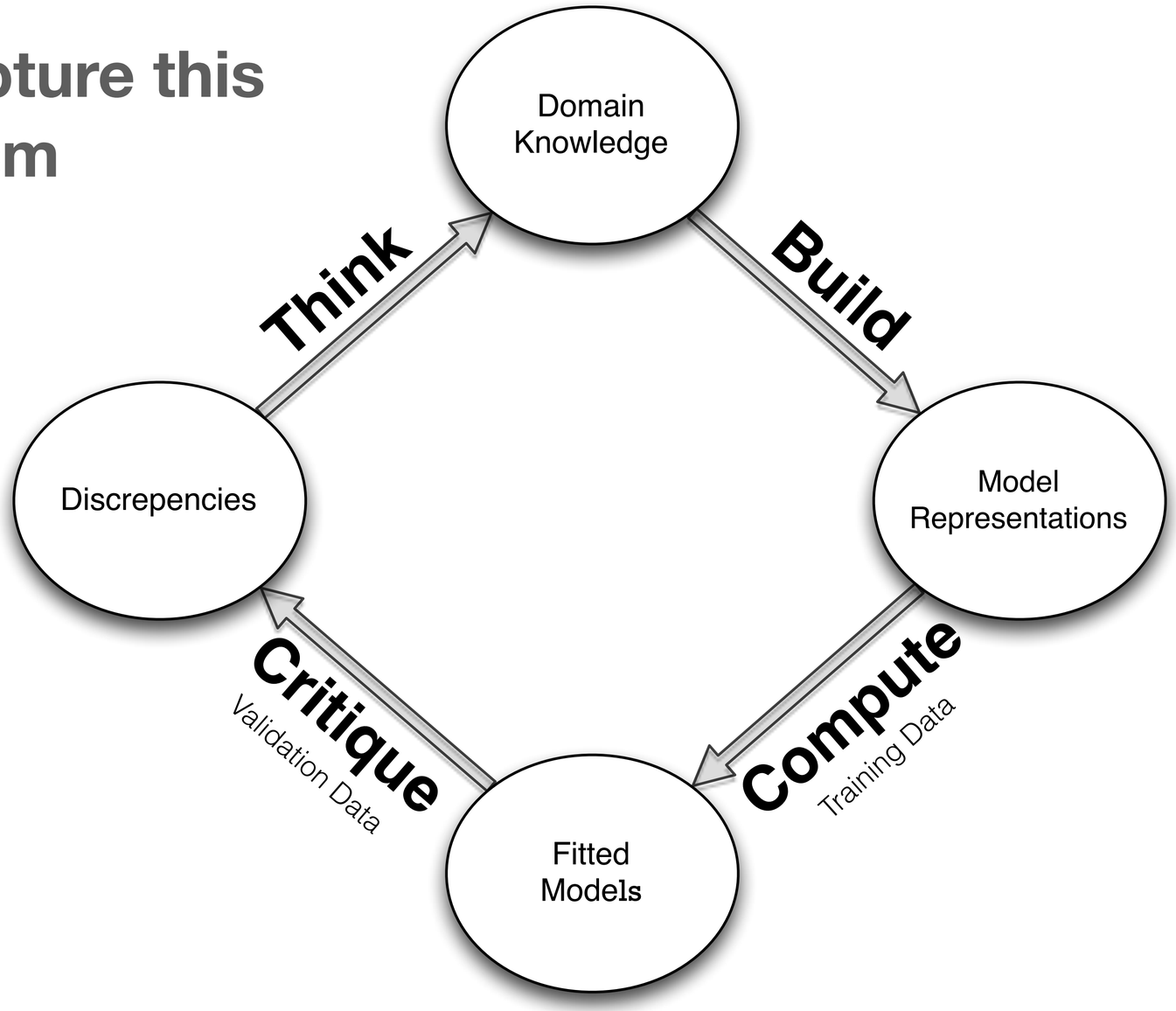
- Files —> a bundle of bits
- Directories —> tree structure of references/“locations”
- yaml objects
  - strings
  - maps/dictionary
  - sequences/lists

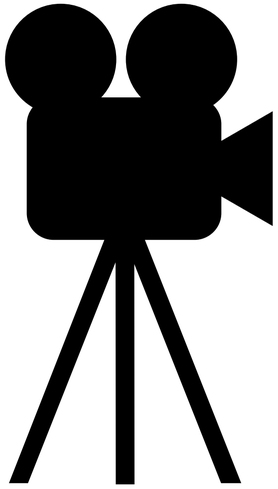
These are STATIC

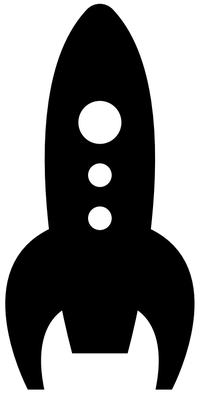


Science proceeds through iterative learning

We need to capture this dynamism

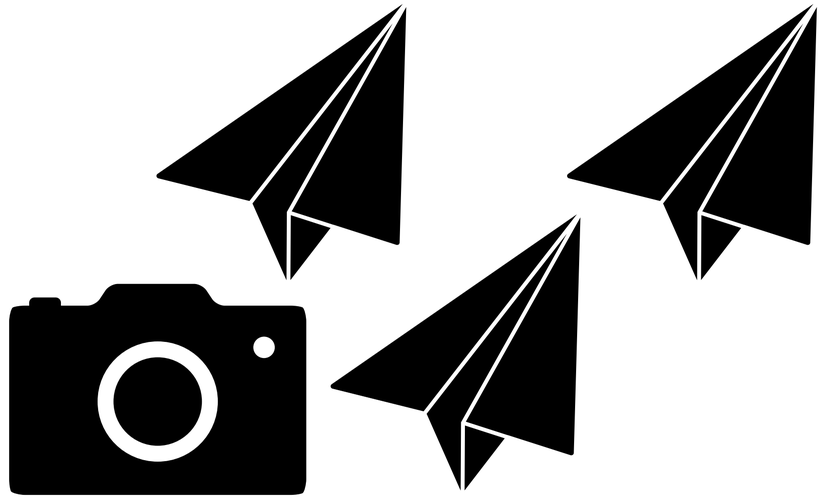






Version control

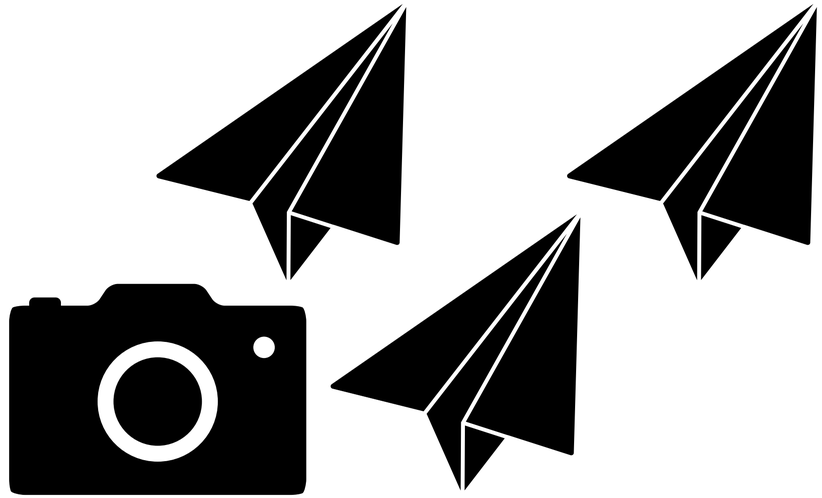
The really bad but most common way



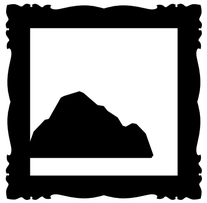
- Name files by versions
  - Code1_v2.py, Code1_v3_final.py, Code2_redone_v3_finalfinal.py
  - Datav2.dat, Datav3.dat, Data2v4.xls, Data3.csv
  - **what does this even mean? Is Code2* related to Code1*? What is the relationship between these files? What about Datav2.dat and Datav3.dat?**
- Overwrite changes
  - Code.py is overwritten every time you make a change
    - **What if you want to roll back and see what you did previously**
    - **What if you do not make a big change to your code because you are afraid of messing it up**
- Email back and forth between co-authors
  - **When did you send that again?**

Version control

The really bad but most common way



- Name files by versions
  - Code1_v2.py, Code1_v3_final.py, Code2_redone_v3_finalfinal.py
  - Datav2.dat, Datav3.dat, Data2v4.xls, Data3.csv
  - **what does this even mean? Is Code2* related to Code1*? What is the relationship between these files? What about Datav2.dat and Datav3.dat?**
- Overwrite changes
  - Code.py is overwritten every time you make a change
    - **What if you want to roll back and see what you did previously**
    - **What if you do not make a big change to your code because you are afraid of messing it up**
- Email back and forth between co-authors
  - **When did you send that again?**

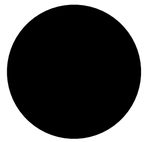


Files are pictures cluttering the attic

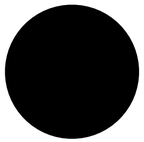
What do we need to do better?

A library and system of organization for our INFORMATION

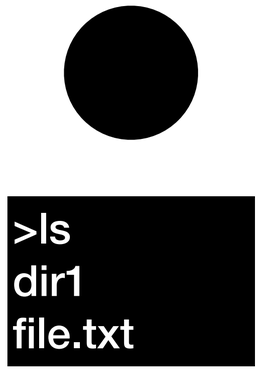
Current state



Still need “state”

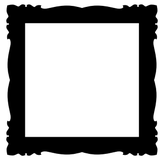
Current state 

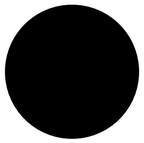
State has files and directories!



Look at the “state” of your directory as a tree

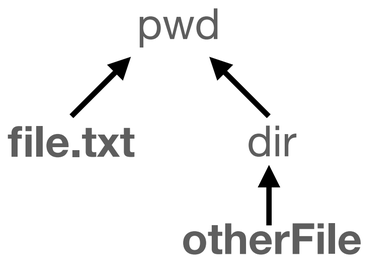
Snapshot





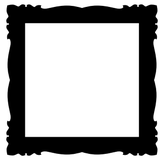
```
>ls
dir1
file.txt
```

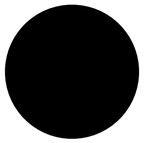
tree



Start a ledger, giving the snapshot a unique id

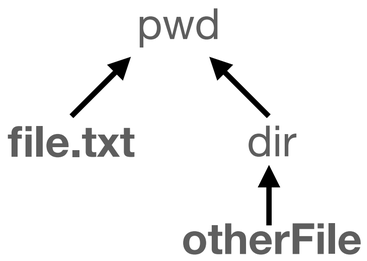
Snapshot





```
>ls
dir1
file.txt
```

tree

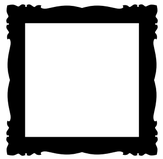


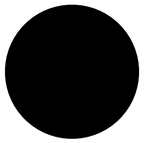
id fdafd8932adfagdgdd

Snapshot id: fdafd8932adfagdgdd

Add the tree to the ledger

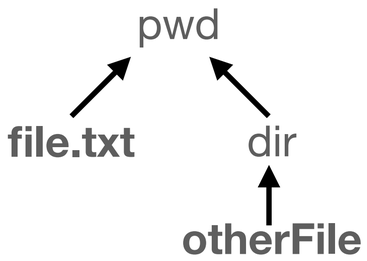
Snapshot





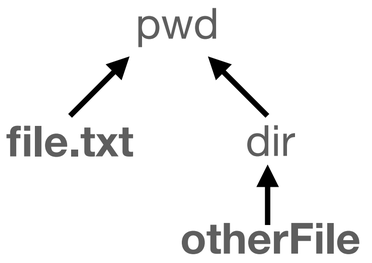
```
>ls
dir1
file.txt
```

tree



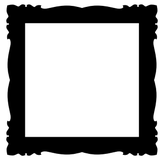
id fdafd8932adfagdgdd

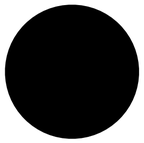
Snapshot id: fdafd8932adfagdgdd



Assign unique ids to all the files and directories

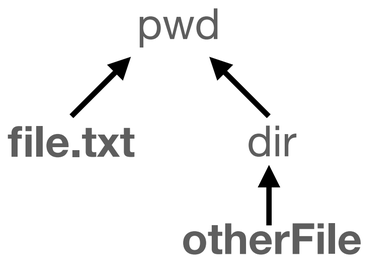
Snapshot





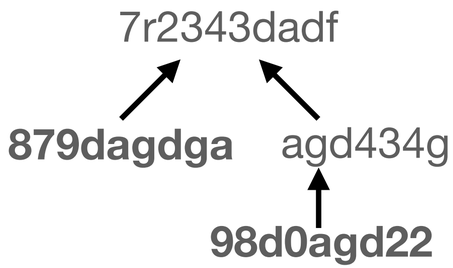
```
>ls
dir1
file.txt
```

tree



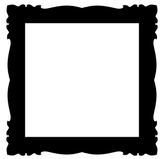
id fdafd8932adfagdgdd

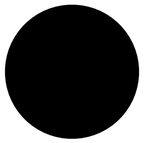
Snapshot id: fdafd8932adfagdgdd



Record the references from ids to names

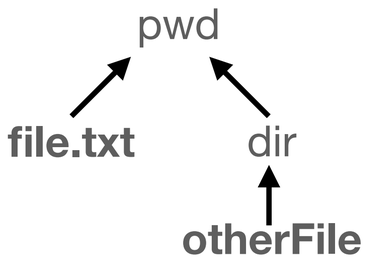
Snapshot





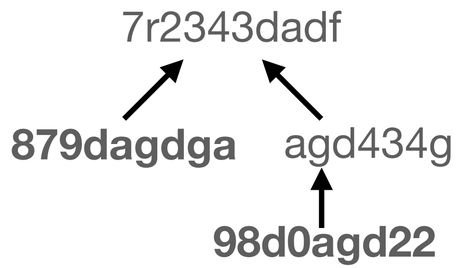
```
>ls
dir1
file.txt
```

tree



id fdafd8932adfagdgdd

Snapshot id: fdafd8932adfagdgdd



repo: 7r2343dadf

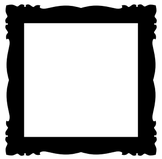
file.txt: 879dagdga

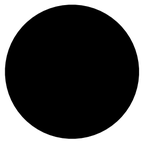
dir: agd434g

otherFile: 98d0agd22

This of the working directory as a repository

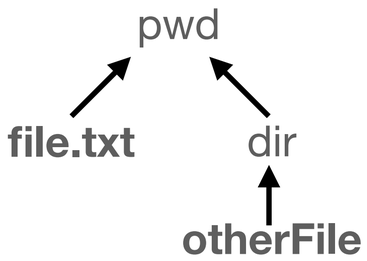
Snapshot





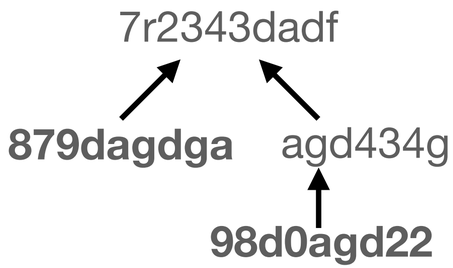
```
>ls
dir1
file.txt
```

tree



id fdafd8932adfagdgdd

Snapshot id: fdafd8932adfagdgdd



repo: 7r2343dadf

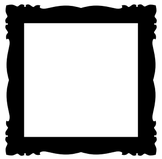
file.txt: 879dagdga

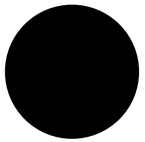
dir: agd434g

otherFile: 98d0agd22

References are maps!

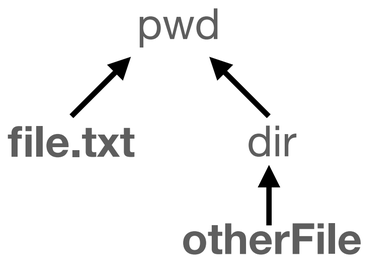
Snapshot





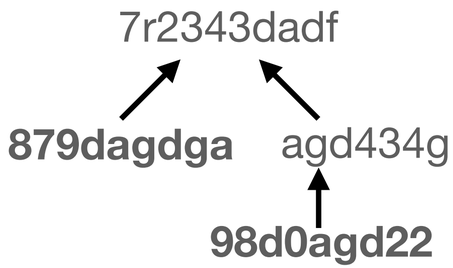
```
>ls
dir1
file.txt
```

tree



id fdafd8932adfagdgdd

Snapshot id: fdafd8932adfagdgdd



repo: 7r2343dadf

file.txt: 879dagdga

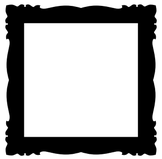
dir: agd434g

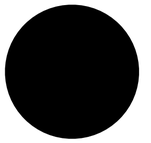
otherFile: 98d0agd22

References

There are 2 types of objects here: blobs and trees

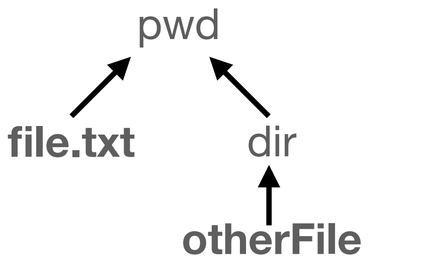
Snapshot





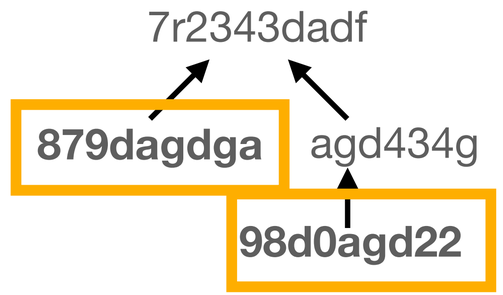
```
>ls
dir1
file.txt
```

tree



id fdafd8932adfagdgdd

Snapshot id: fdafd8932adfagdgdd



repo: 7r2343dadf

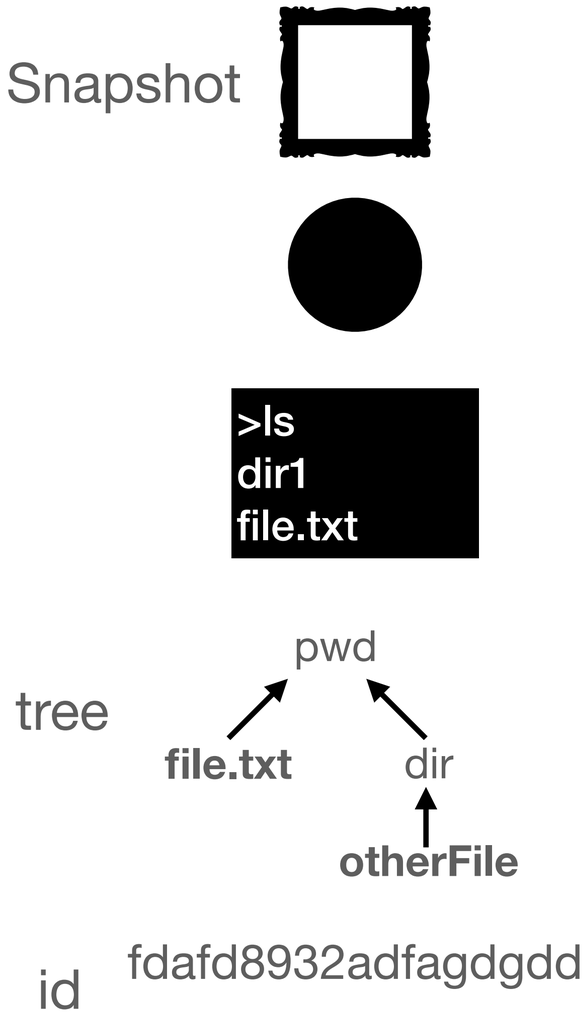
file.txt: **879dagdga**

dir: agd434g

otherFile: **98d0agd22**

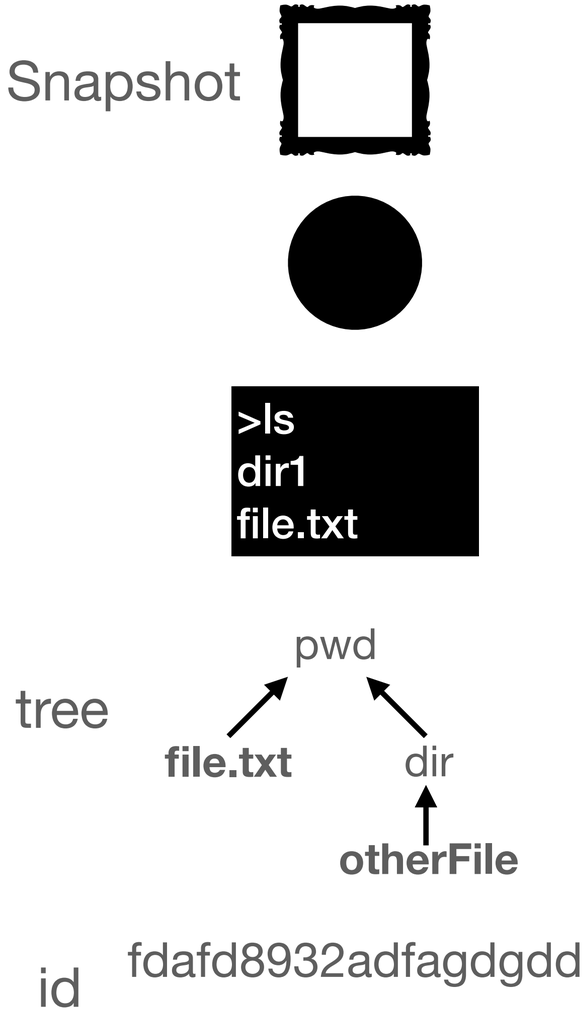
Blobs

There are 2 types of objects here: blobs and trees



A snapshot is a collection of blobs and trees

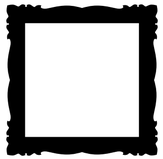
Call this a commit: since it has meta-data not just snapshot

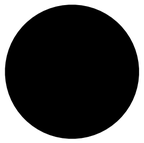


A snapshot is a collection of blobs and trees

Now that we recorded that “commit”; do science

Snapshot

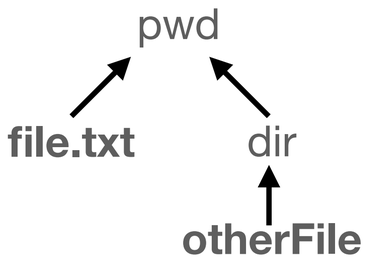




```
>ls
dir1
file.txt
```

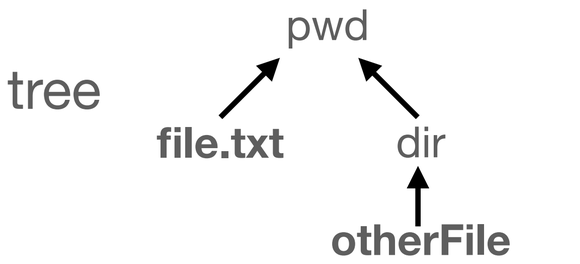
```
>echo “more” >> file.xt
```

tree



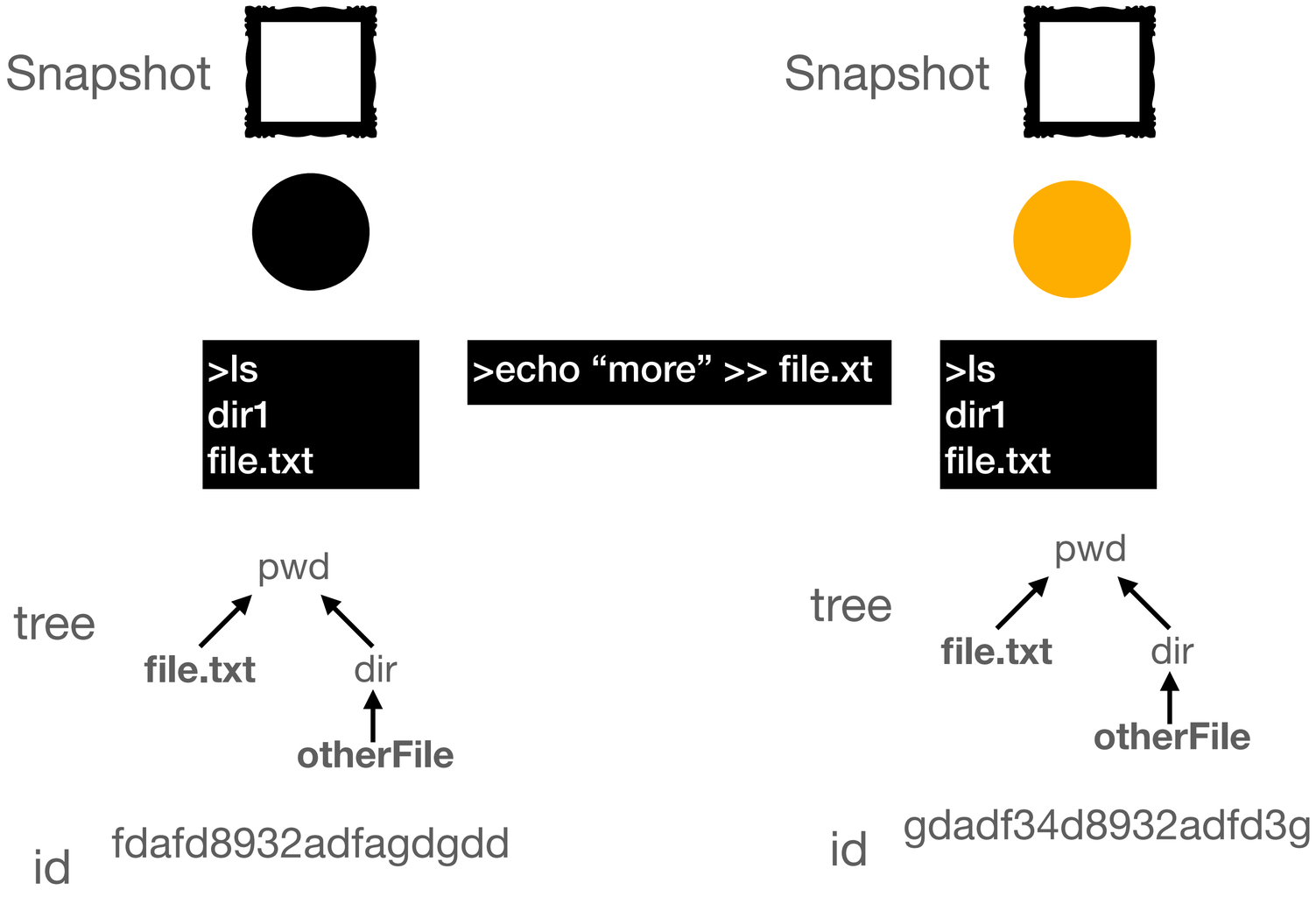
id fdafd8932adfagdgdd

This leads to a CHANGE in state

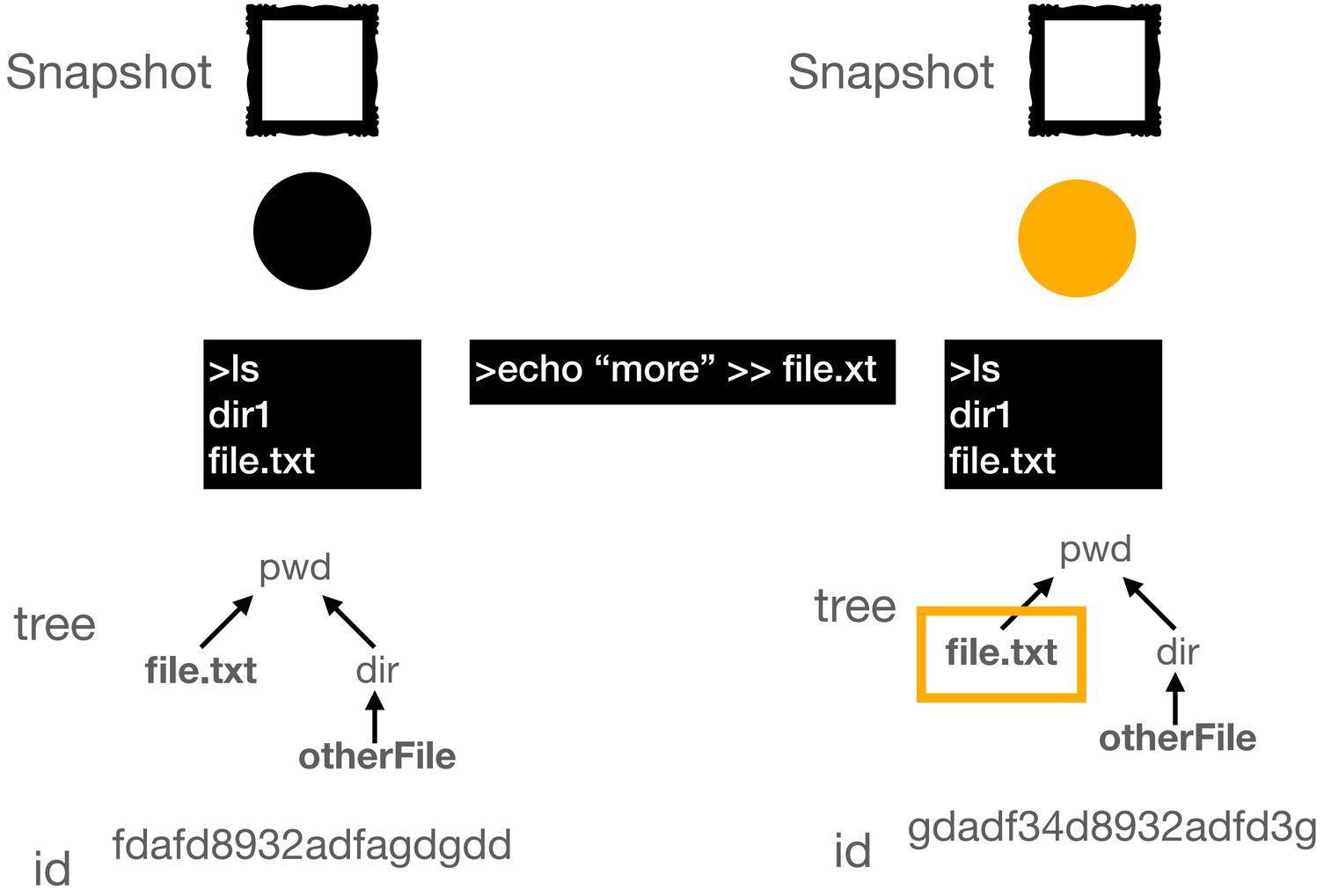


id fdafd8932adfagdgdd

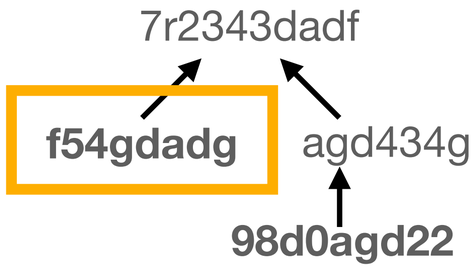
Still have same “names” of files



But when we make a new ledger entry:



commit id: gdadf34d8932adfd3g



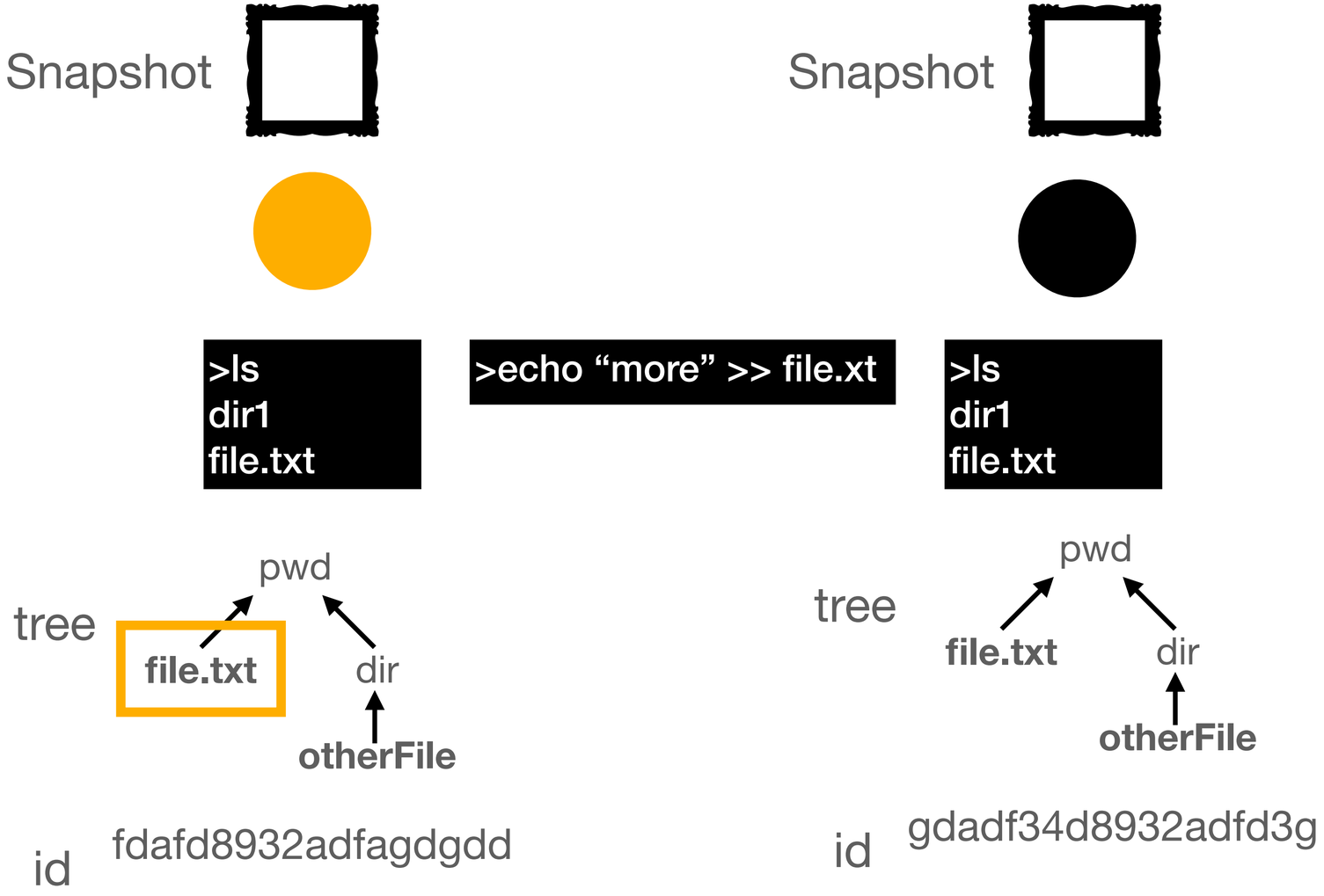
repo: 7r2343dadf

file.txt: f54gdadg

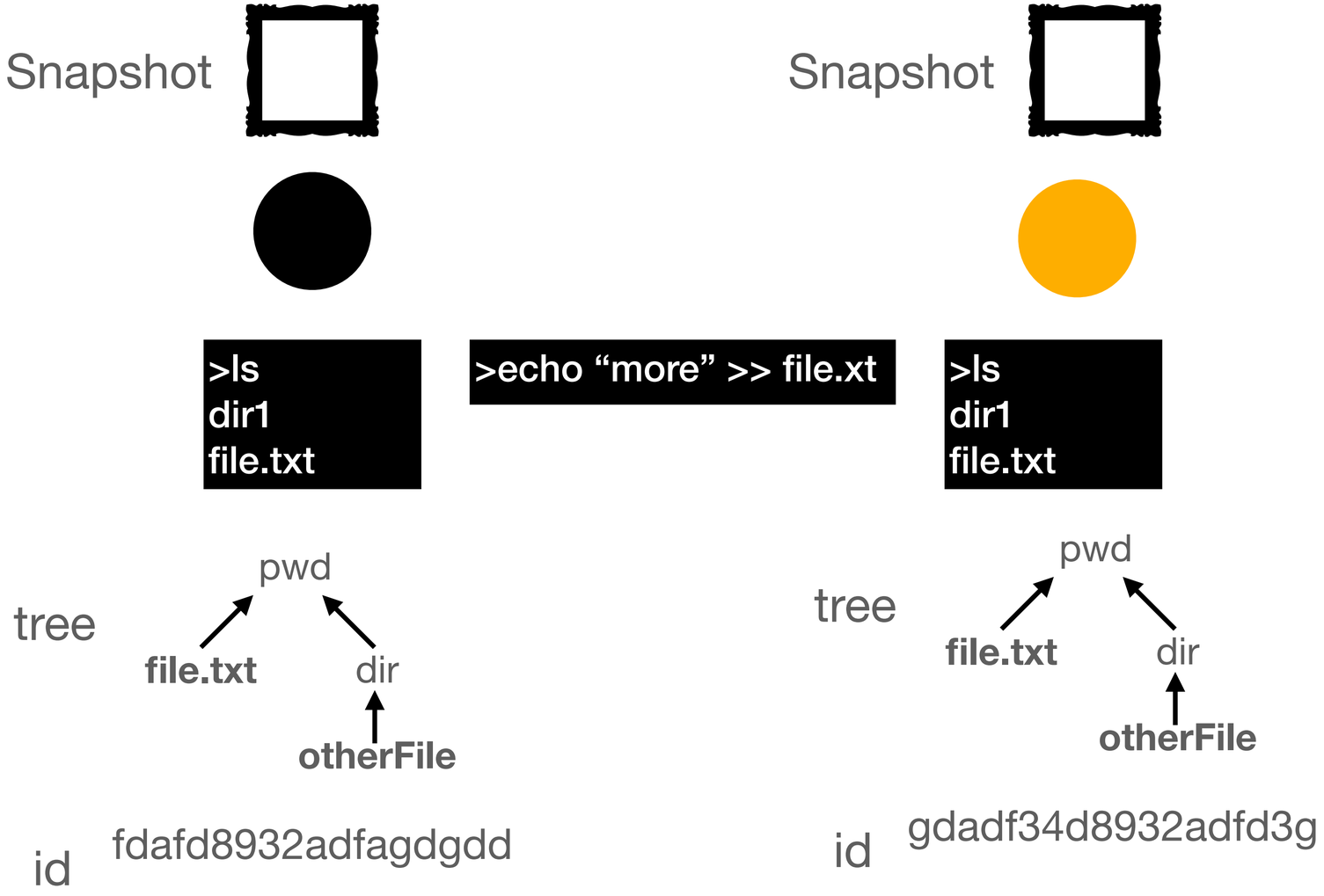
dir: agd434g

otherFile: 98d0agd22

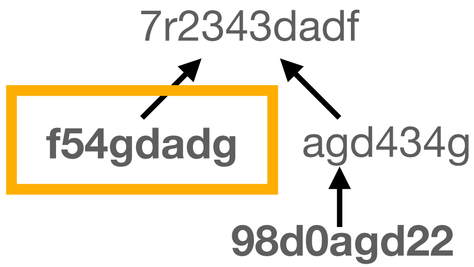
file.txt has different ids across commits



file.txt has different ids across commits



commit id: gdadf34d8932adfd3g



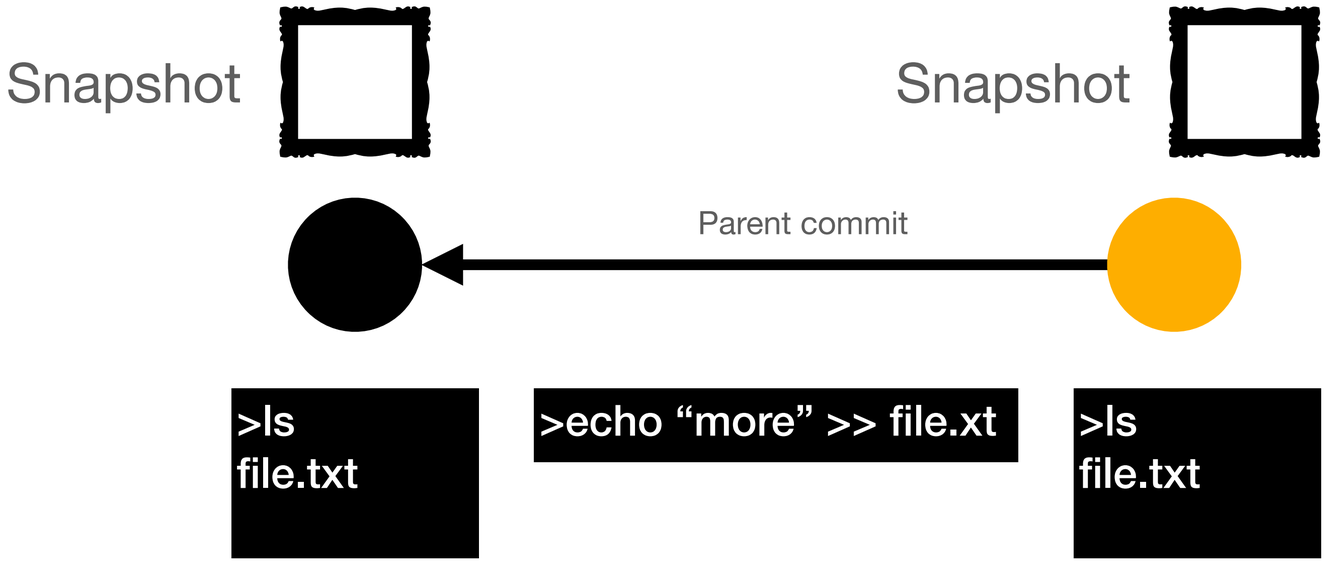
repo: 7r2343dadf

file.txt: f54gdadg

dir: agd434g

otherFile: 98d0agd22

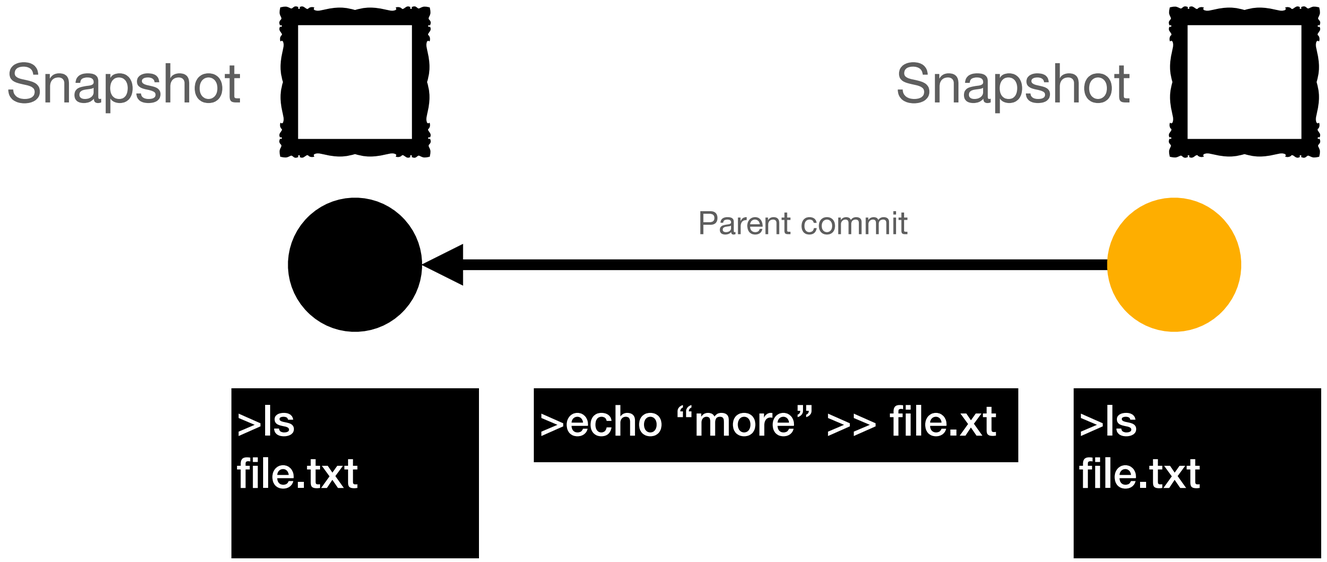
We CONNECT the commits in a history



id fdafd8932adfagdgdd

id gdadf34d8932adfd3g

Record the “parent” commit



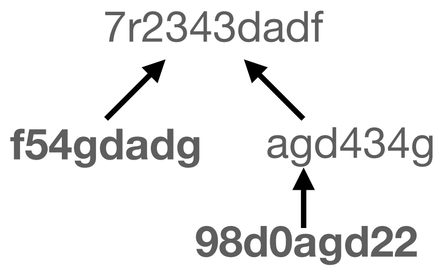
Now we have dynamism, movement,
from one commit to the next

id fdafd8932adfagdgdd

id gdadf34d8932adfd3g

commit id: gdadf34d8932adfd3g

Parent id: fdafd8932adfagdgdd



repo: 7r2343dadf

file.txt: f54gdadg

dir: agd434g

otherFile: 98d0agd22

History of commits is just a path from a commit to parent
That parent is a commit, so it has a parent (or is the first),
And so on!

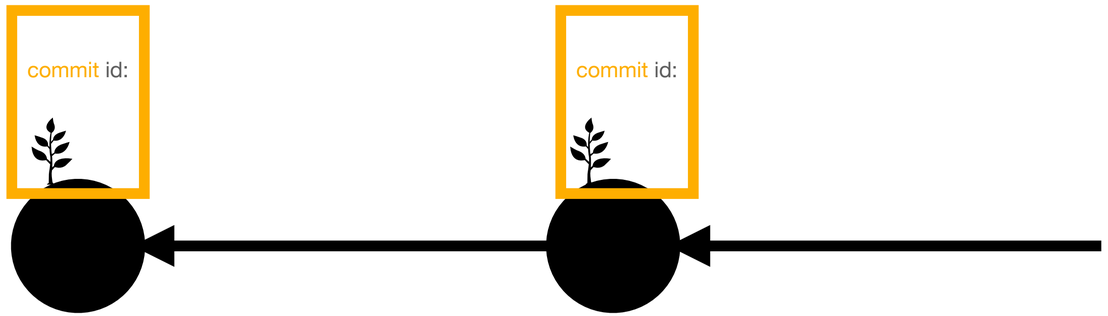
Each commit holds trees that can hold trees and blobs

History of commits is just a path from a commit to parent
That parent is a commit, so it has a parent (or is the first),
And so on!

Each commit holds trees that can hold trees and blobs

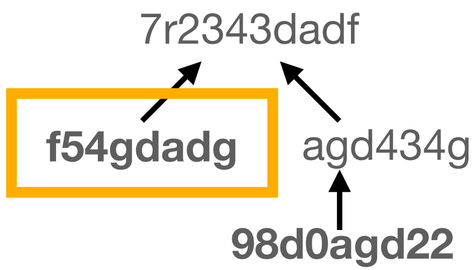
History of commits is just a path from a commit to parent
That parent is a commit, so it has a parent (or is the first),
And so on!

Each commit holds trees that can hold trees and blobs



Each commit holds trees that can hold trees and blobs

commit id: gdadf34d8932adfd3g



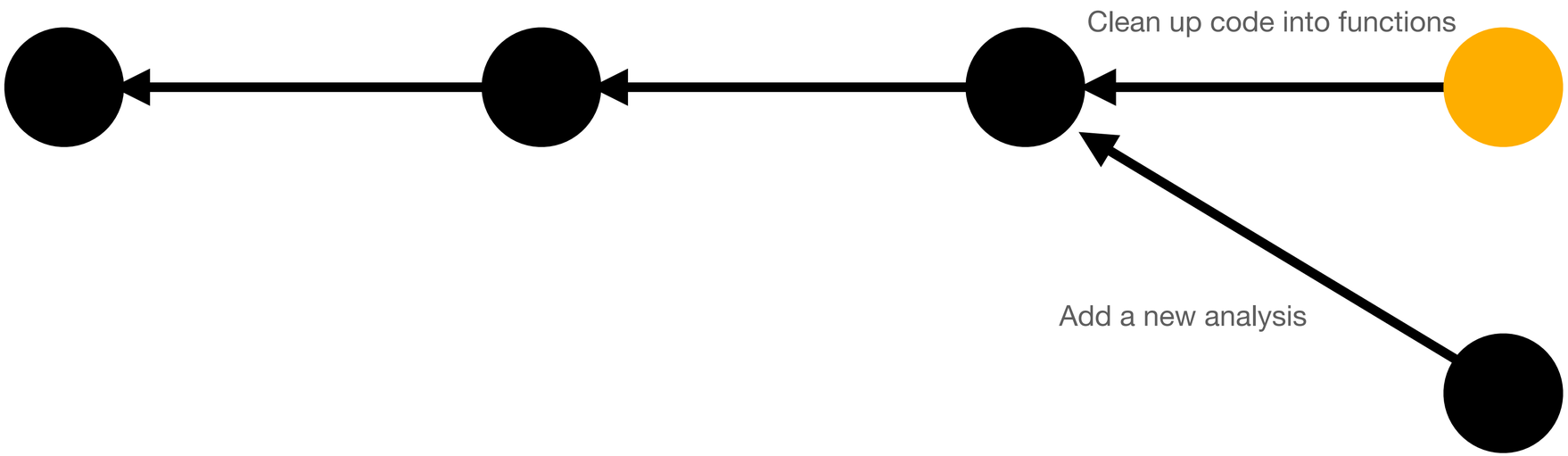
repo: 7r2343dadf

file.txt: f54gdadg

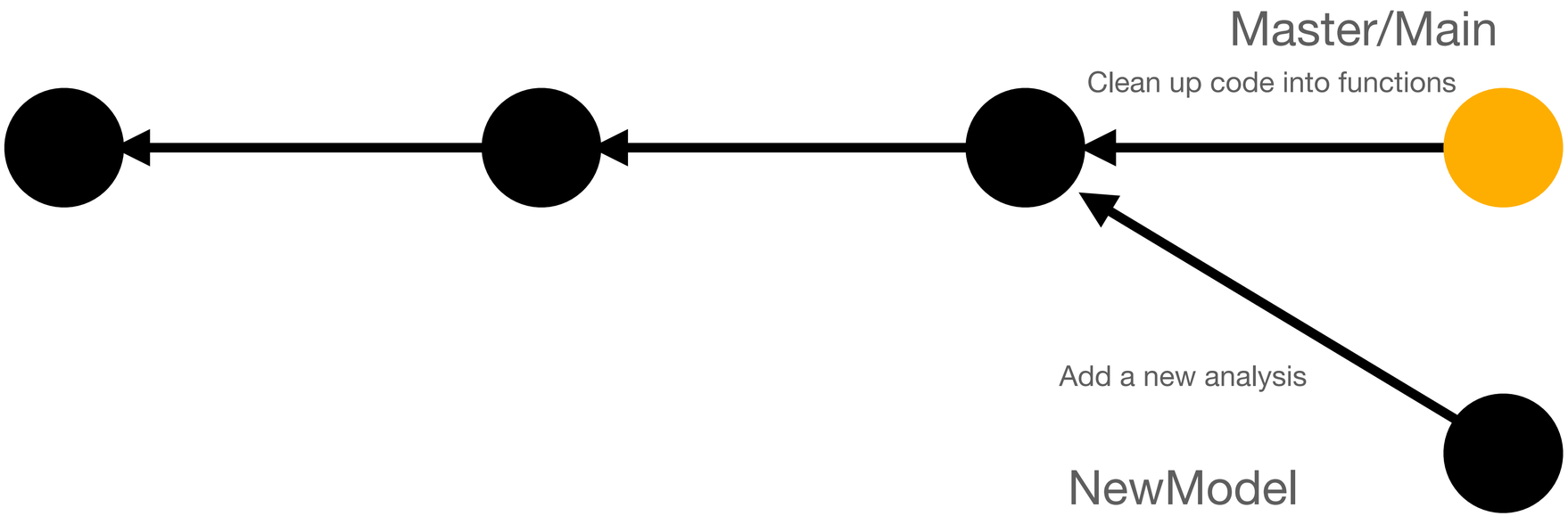
dir: agd434g

otherFile: 98d0agd22

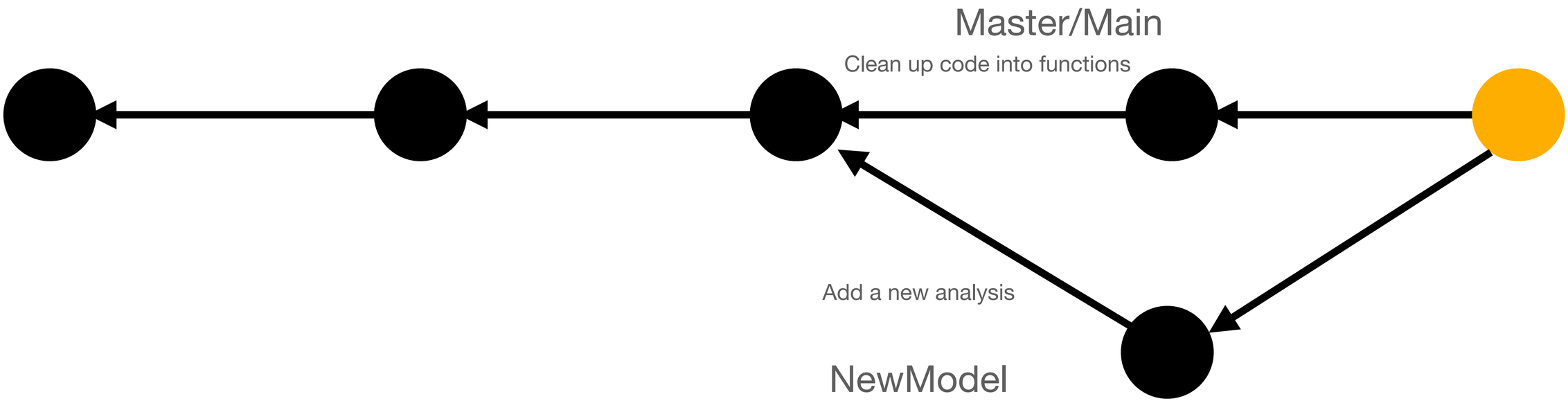
With history secure…



Can “branch” off to try new things



Can “branch” off to try new things



Then merge back
Each dot is a commit
Each arrow points to the parent to the commit

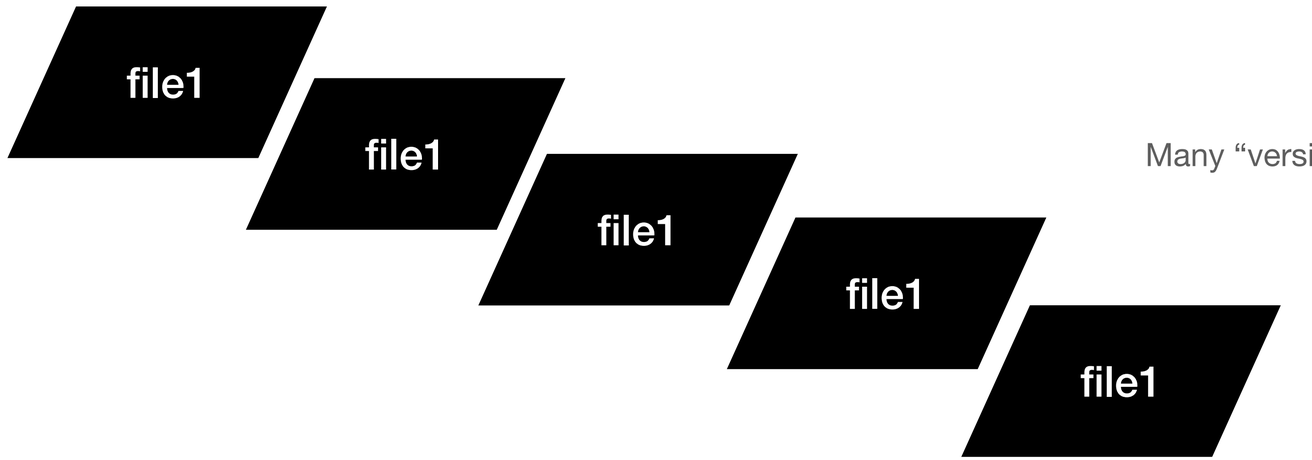
Git the hang of it

This is the structure of git

It is a set of tools that coherently account for blobs/files, trees/directories AND changes to these, as paths through commits

It is a model of history as a series of snapshots (trees and blobs)

Where the paths are tied together through child-parent relations



Many “versions” of a file/directory

Git the hang of it

This is the structure of git

It is a set of tools that coherently account for blobs/files, trees/ directories AND changes to these, as paths through commits

It is a model of history as a series of snapshots (trees and blobs)

Where the paths are tied together through child-parent relations

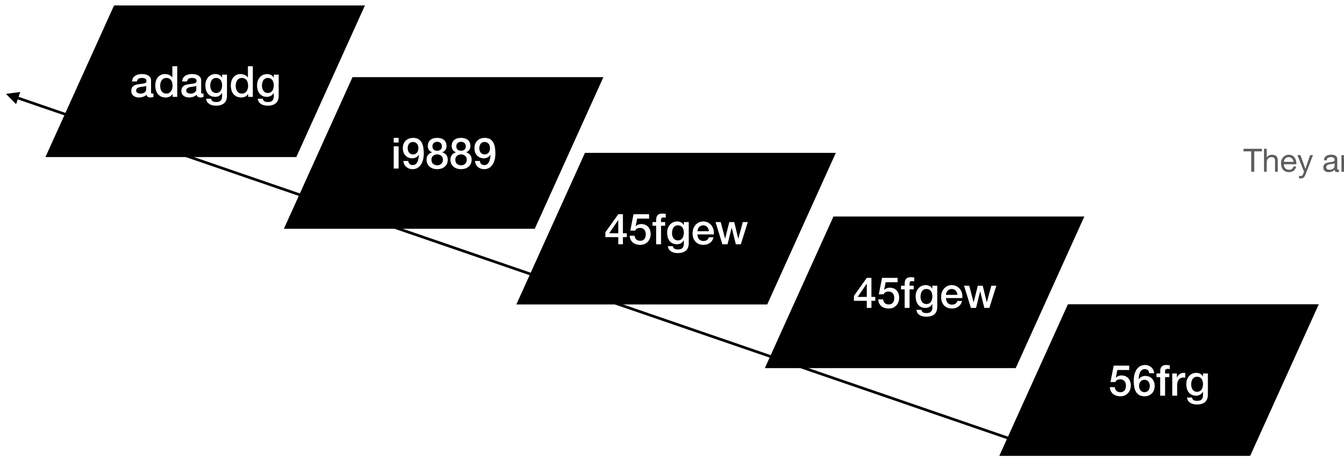
Git the hang of it

This is the structure of git

It is a set of tools that coherently account for blobs/files, trees/ directories AND changes to these, as paths through commits

It is a model of history as a series of snapshots (trees and blobs)

Where the paths are tied together through child-parent relations



They are tied together through the commit parents in their history

Basic git commands

- git init
  - Set up a rep
- git add
  - “stage” files and directories for the next commit (“say cheese”), who is in the picture?
- git commit
  - take the snapshot, assign a new commit id to the what was staged
- git status
  - Compare working directory with what is staged, tracked, etc
    - Tracked means that a current file in the repository has a reference in a previous commit
- git log
  - Look at time line of history
- git checkout
  - for branching