Git from the inside out

Talk structure

Git is a graph

Git is a graph

This graph dictates Git's behaviour

Git is a graph

If you understand this graph,
you understand Git

Run Git commands on a repository

Observe how those commands change the graph

Create a project

Create a project

```
~ $ mkdir alpha
```

Create a project

```
      ~ $ mkdir alpha
      ~ $ cd alpha
~/alpha $
```

Create `data` directory

```
~/alpha $ mkdir data
```

Create `data/letter.txt`

```
~/alpha $ mkdir data
~/alpha $ printf 'a' > data/letter.txt
```

Project layout

```
~/alpha $ tree
          alpha
          └── data
               └── letter.txt
```

Initialize the repository

Initialize the repository

```
~/alpha $ git init
          Initialized repository
```

File layout

```
~/alpha $ git init
          Initialized repository
~/alpha $ tree -a
          alpha
          └── data
               └── letter.txt
```

File layout

```
~/alpha $ git init
          Initialized repository
~/alpha $ tree -a
          alpha
          ├── data
          │   └── letter.txt
          └── .git
              └── objects
                  etc...
```

Add a new file to Git

Add `data/letter.txt` to Git

```
~/alpha $ git add data/letter.txt
```

1. Create a blob object

```
~/alpha $ git add data/letter.txt
~/alpha $ tree -a .git
          .git
          └── objects
              └── 2e
                  └── 65
```

Hashes

Hash of `data/letter.txt` content

```
~/alpha $ git hash-object data/letter.txt
          2e65
```

1. Create a blob object

```
~/alpha $ git add data/letter.txt
~/alpha $ tree -a .git
          .git
          └── objects
              └── 2e
                  └── 65
```

```
~/alpha $ git hash-object data/letter.txt
          2e65
```

1. Create a blob object

```
~/alpha $ git add data/letter.txt
~/alpha $ tree -a .git
          .git
          └── objects
              └── 2e
                  └── 65
~/alpha $ cat .git/objects/2e/65
          xK??OR0dH
```

1. Create a blob object

```
~/alpha $ git add data/letter.txt
~/alpha $ tree -a .git
          .git
          └── objects
              └── 2e
                  └── 65
~/alpha $ cat .git/objects/2e/65
          xK??OR0dH
~/alpha $ git cat-file -p 2e65
          a
```

2. Make an entry in the index

```
~/alpha $ git add data/letter.txt
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ cat .git/index
          ?H?u.data/letter.txtd
```

2. Make an entry in the index

```
~/alpha $ git add data/letter.txt
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ cat .git/index
          ?H?u.data/letter.txtd
~/alpha $ git ls-files -s
          data/letter.txt 2e65
```

2. Make an entry in the index

```
~/alpha $ git add data/letter.txt
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ cat .git/index
          ?H?u.data/letter.txtd
~/alpha $ git ls-files -s
          data/letter.txt 2e65
```

Re-add a file to a repository

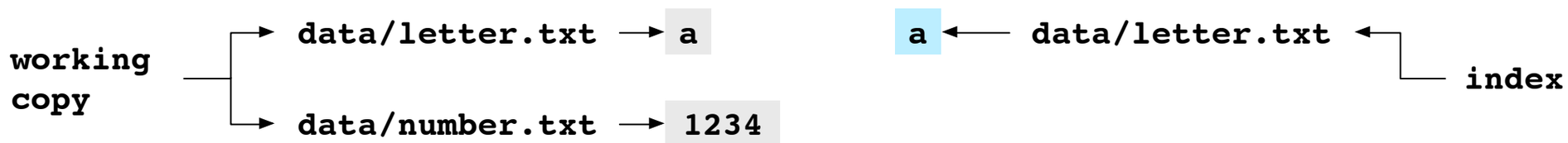
After `data/letter.txt` added

Create `data/number.txt`

```
~/alpha $ printf '1234' > data/number.txt
```

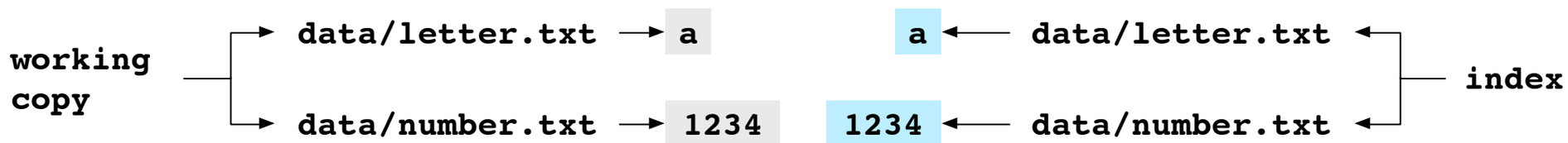
Create `data/number.txt`

```
~/alpha $ printf '1234' > data/number.txt
```



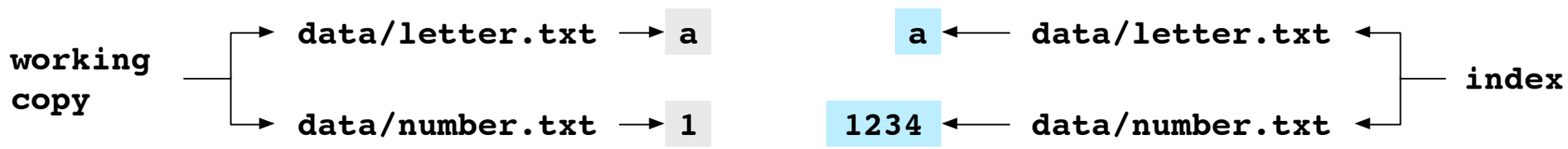
Add `data/number.txt`

```
~/alpha $ git add data/number.txt
```



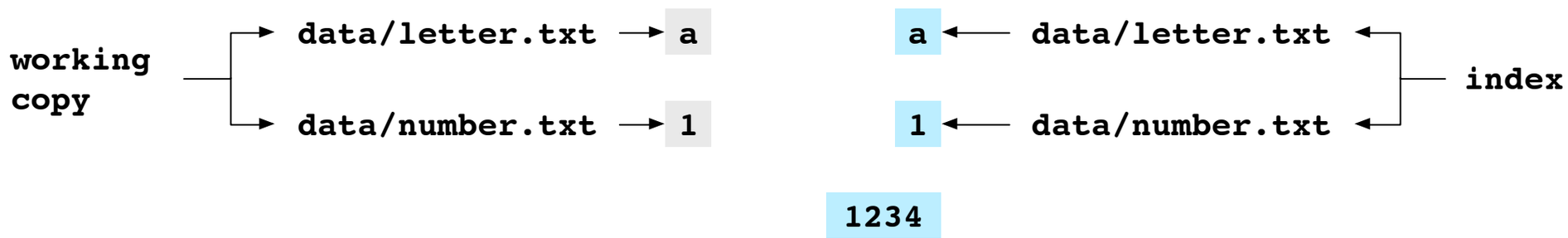
Edit `data/number.txt`

```
~/alpha $ printf '1' > data/number.txt
```



Add `data/number.txt`

```
~/alpha $ printf '1' > data/number.txt
~/alpha $ git add data/number.txt
```



Make a commit

Make a commit

```
~/alpha $ git commit -m 'a1'
          master 774b
```

Before the commit

a	1

1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 0eed
          blob 2e65 letter.txt
          blob 56a6 number.txt
```

a **1**

1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 0eed
          blob 2e65 letter.txt
          blob 56a6 number.txt
```

a 1

1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 0eed
          blob 2e65 letter.txt
          blob 56a6 number.txt
```

a **1**

1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 0eed
          blob 2e65 letter.txt
          blob 56a6 number.txt
```

a 1

1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 0eed
          blob 2e65 letter.txt
          blob 56a6 number.txt
```

a **1**

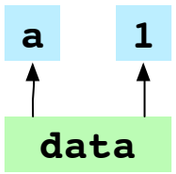
1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 0eed
          blob 2e65 letter.txt
          blob 56a6 number.txt
```

a 1

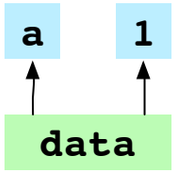
1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 0eed
          blob 2e65 letter.txt
          blob 56a6 number.txt
```



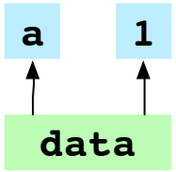
1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p ffe2
          tree 0eed  data
```



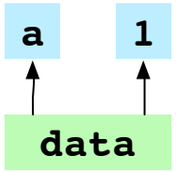
1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p ffe2
          tree 0eed  data
```



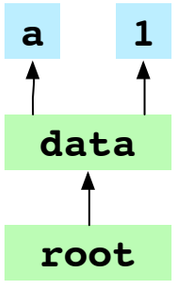
1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p ffe2
          tree 0eed  data
```



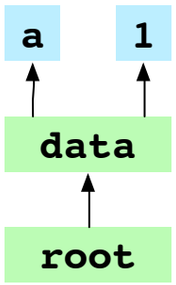
1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p ffe2
          tree 0eed  data
```



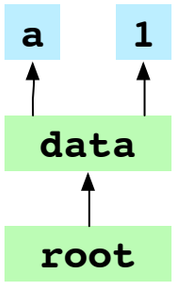
2. Make the commit object

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 774b
          tree ffe2
          author mr@c.com 1424798436
          a1
```



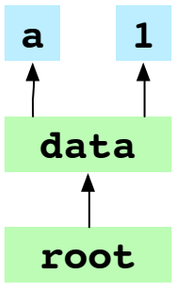
2. Make the commit object

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 774b
          tree ffe2
          author mr@c.com 1424798436
          a1
```



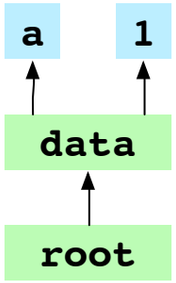
2. Make the commit object

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 774b
          tree ffe2
          author mr@c.com 1424798436
          a1
```



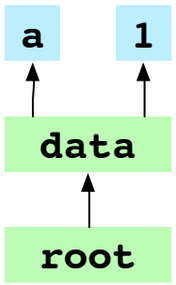
2. Make the commit object

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 774b
          tree ffe2
          author mr@c.com 1424798436
          a1
```



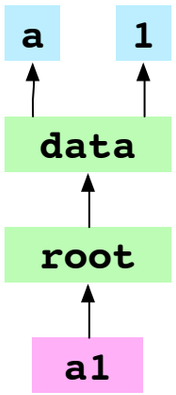
2. Make the commit object

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 774b
          tree ffe2
          author mr@c.com 1424798436
          a1
```



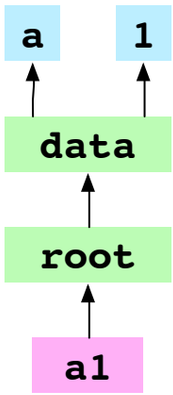
2. Make the commit object

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ git cat-file -p 774b
          tree ffe2
          author mr@c.com 1424798436
          a1
```



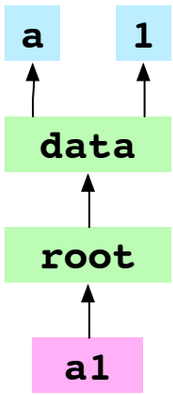
3. Point HEAD at the new commit

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ cat .git/HEAD
          ref:     refs/heads/master
```



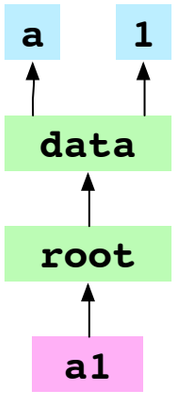
3. Point HEAD at the new commit

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ cat .git/HEAD
          ref:     refs/heads/master
~/alpha $ cat .git/refs/heads/master
          774b
```



3. Point HEAD at the new commit

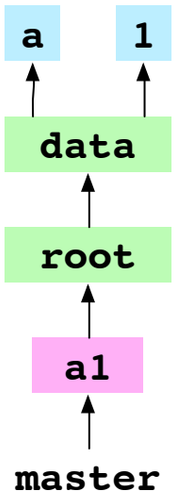
```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ cat .git/HEAD
          ref:     refs/heads/master
~/alpha $ cat .git/refs/heads/master
          774b
```




```
~/alpha $ git commit -m 'a1'
          master 774b
```

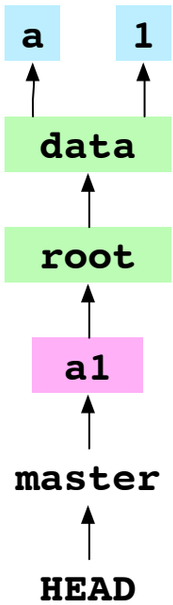
3. Point HEAD at the new commit

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ cat .git/HEAD
          ref:     refs/heads/master
~/alpha $ cat .git/refs/heads/master
          774b
```



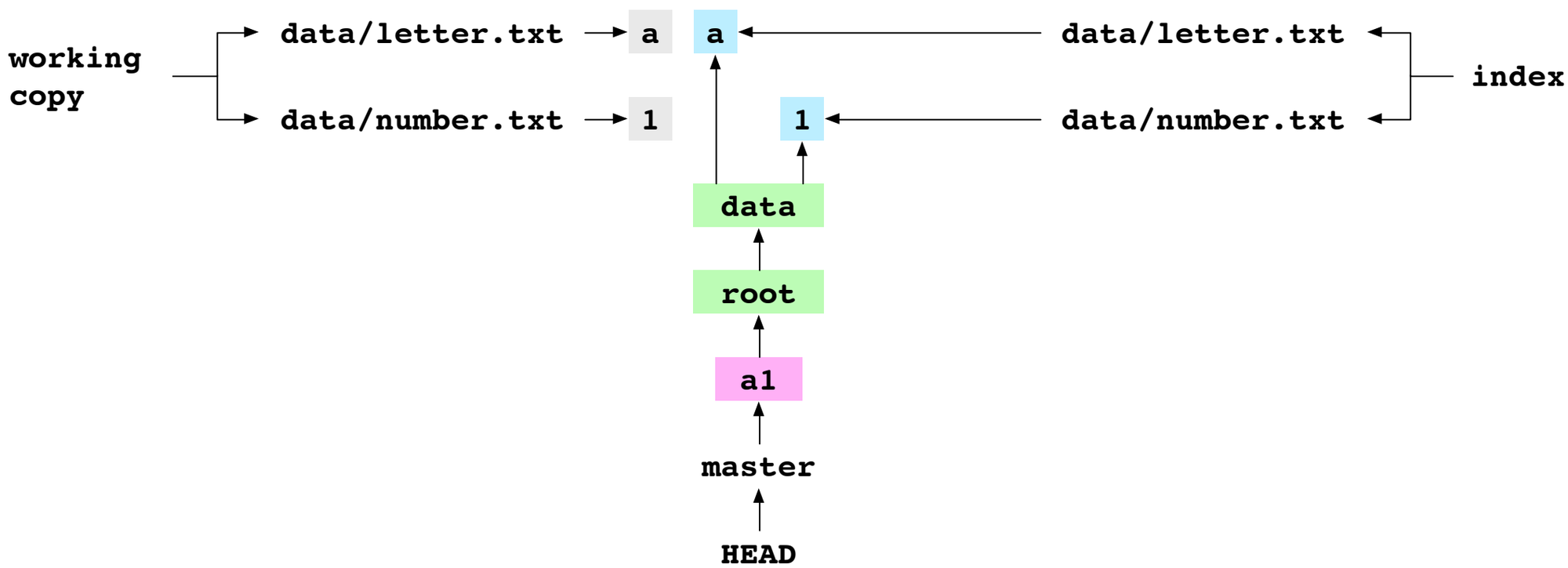
3. Point HEAD at the new commit

```
~/alpha $ git commit -m 'a1'
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
~/alpha $ cat .git/HEAD
          ref:     refs/heads/master
~/alpha $ cat .git/refs/heads/master
          774b
```



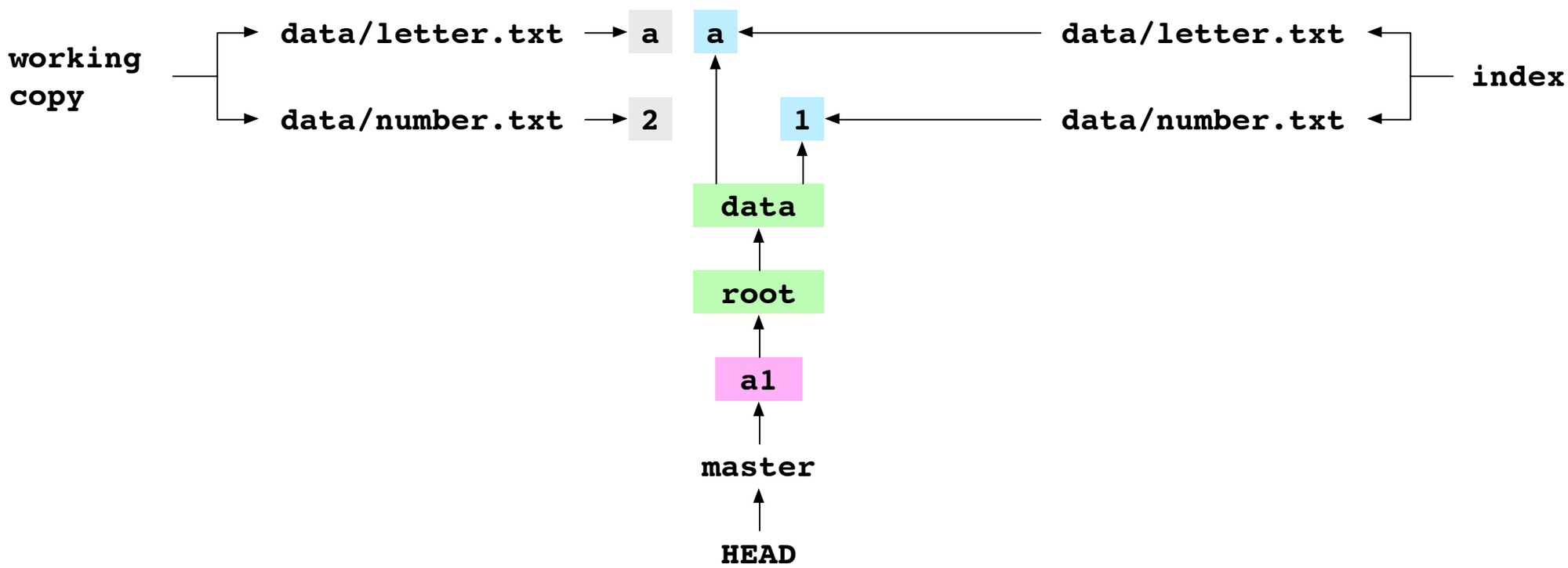
Make a commit that is not
the first commit

After the first commit



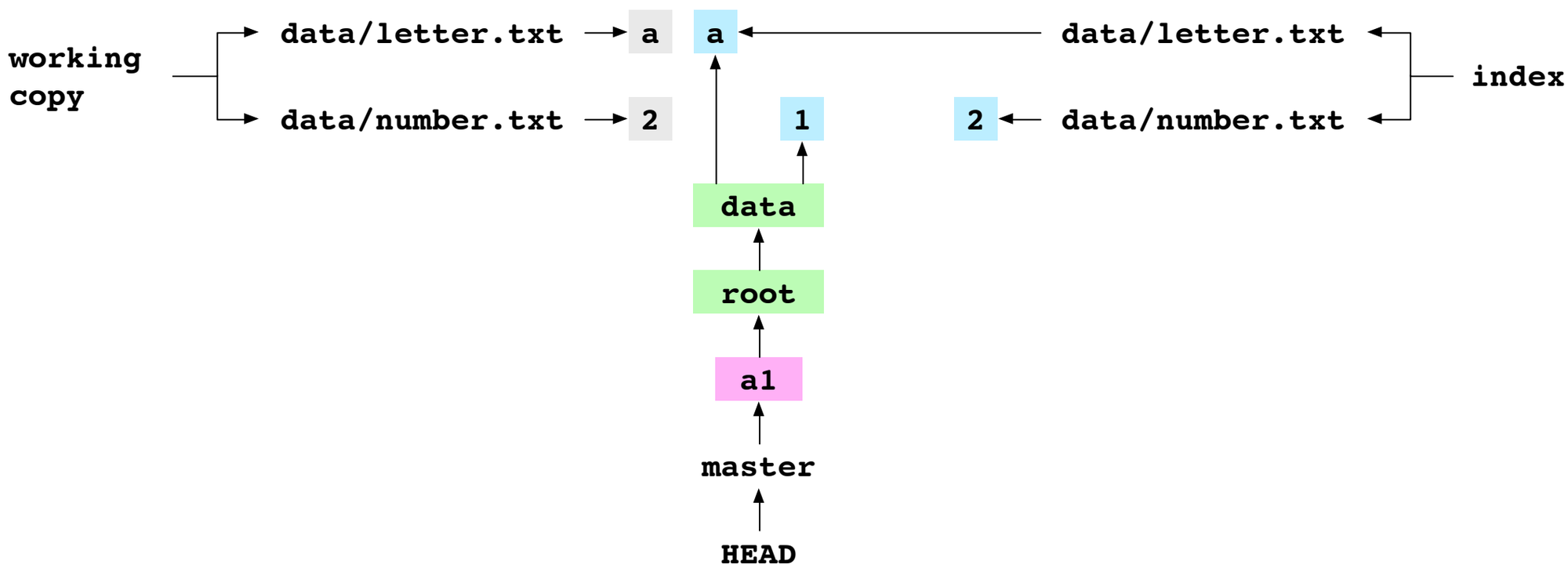
Edit `number.txt`

```
~/alpha $ printf '2' > data/number.txt
```



Add number.txt

```
~/alpha $ git add data/number.txt
```

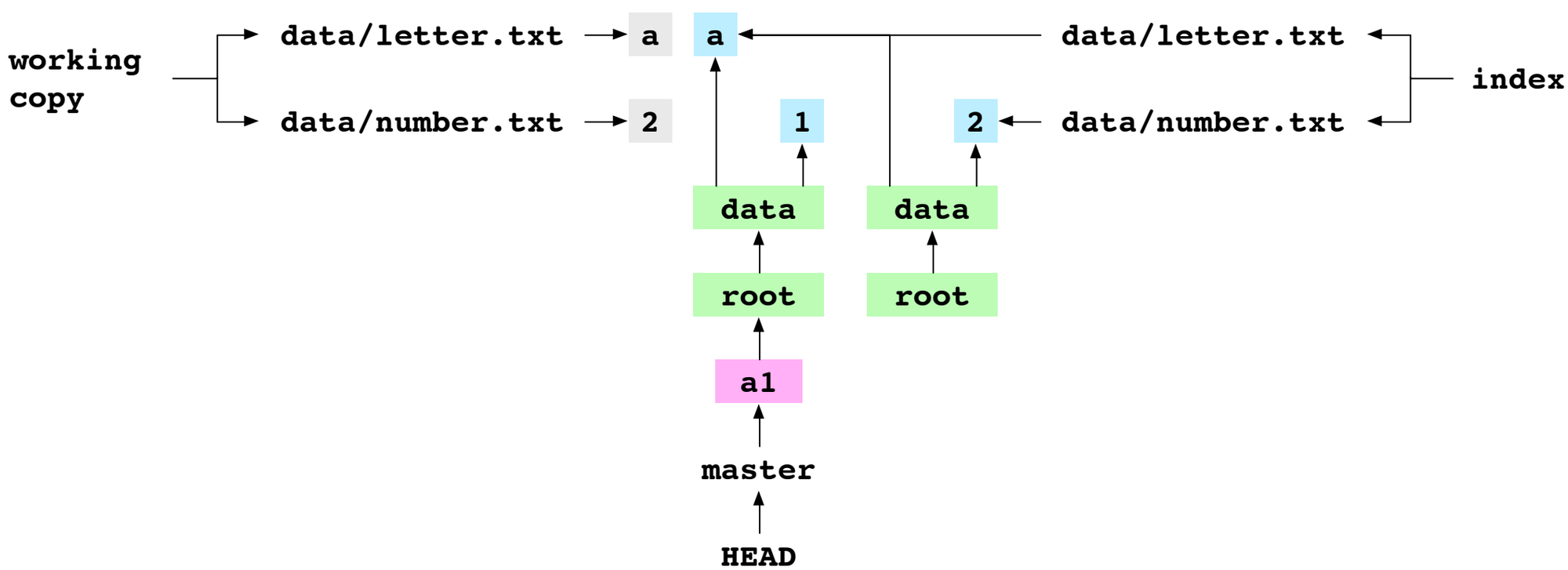


Make a2 commit

```
~/alpha $ git commit -m 'a2'
          master f0af
```

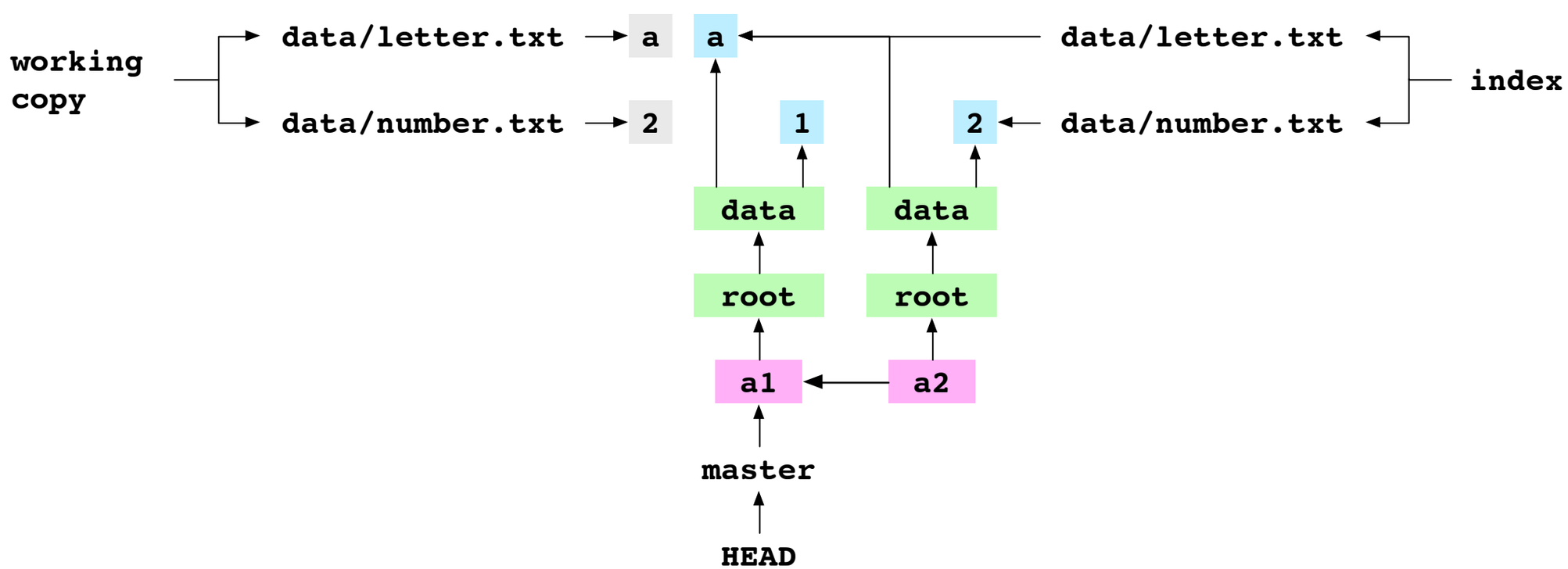
1. Make a tree graph of the contents of the index

```
~/alpha $ git commit -m 'a2'
          master f0af
```



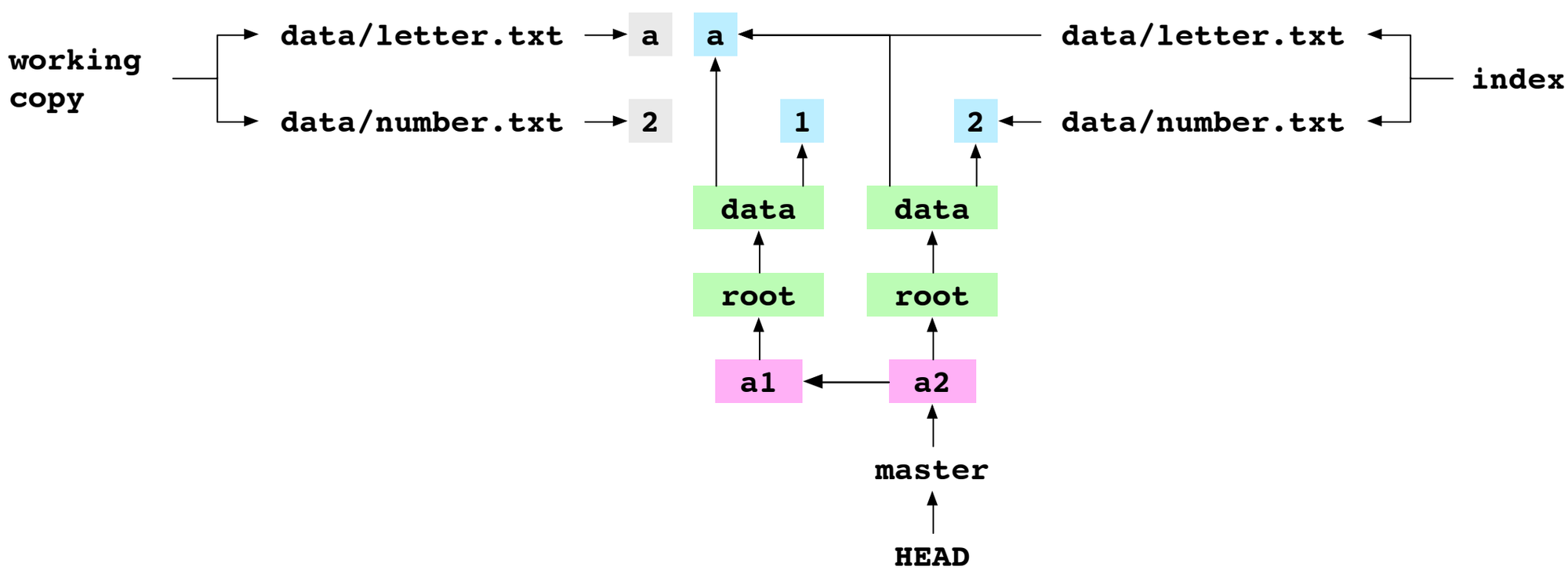
2. Create the commit object

```
~/alpha $ git commit -m 'a2'
          master f0af
~/alpha $ git cat-file -p f0af
          tree ce72
          parent 774b
          author mr@c.com 1424798436
          a2
```

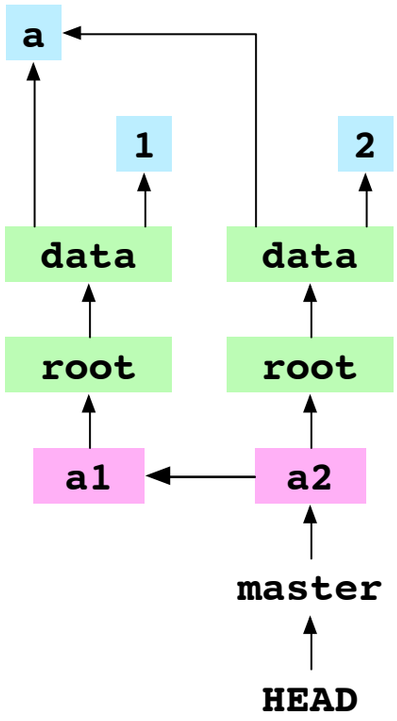


3. Point HEAD at the new commit

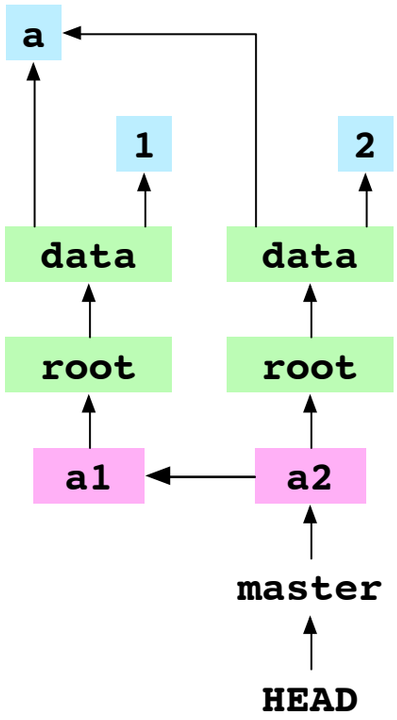
```
~/alpha $ git commit -m 'a2'
          master f0af
```



Content is stored as trees

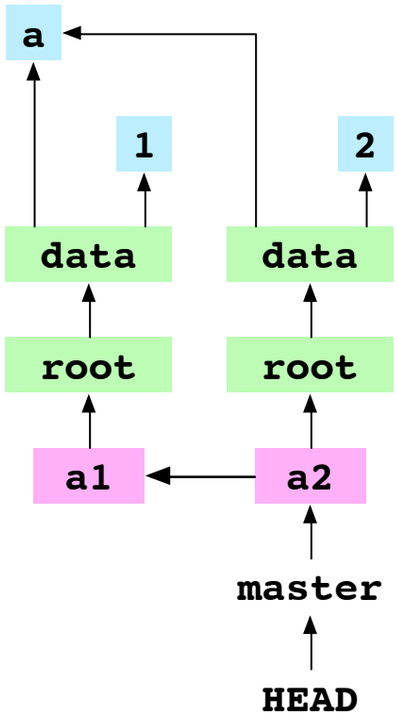


Content is stored as trees

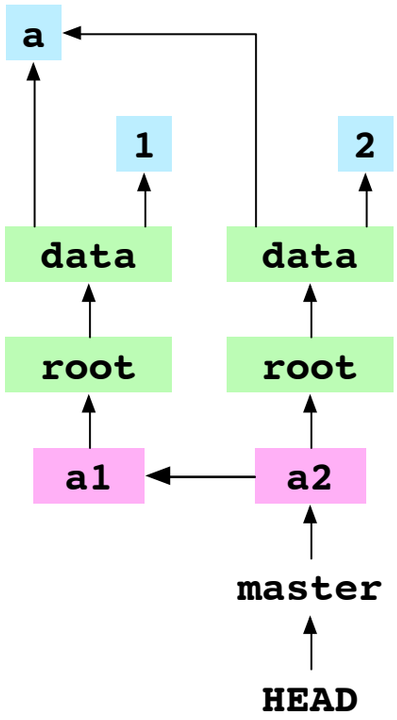


The objects database stores diffs

Each commit has a parent

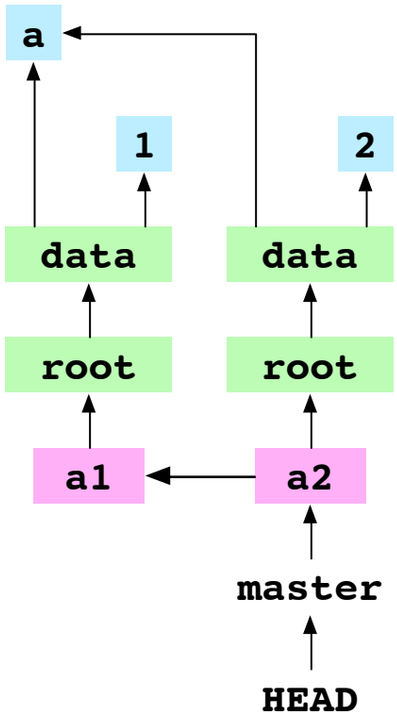


Each commit has a parent

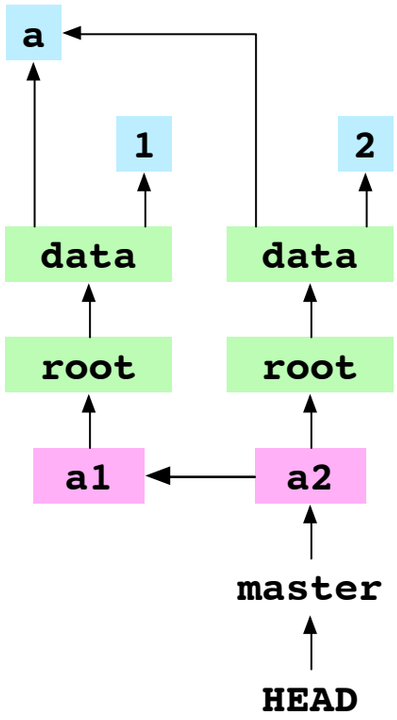


A repository stores the history of a project

Refs are entry points to the commit history



Refs are entry points to the commit history



Commits can be given meaningful names

Objects are immutable

Objects are immutable

Content is edited, not deleted

Refs are mutable

Refs are mutable

The meaning of a ref can change

Check out a commit

Check out a commit

```
~/alpha $ git checkout f0af
          HEAD is detached
```

Check out a commit

```
~/alpha $ git checkout f0af
          HEAD is detached
```

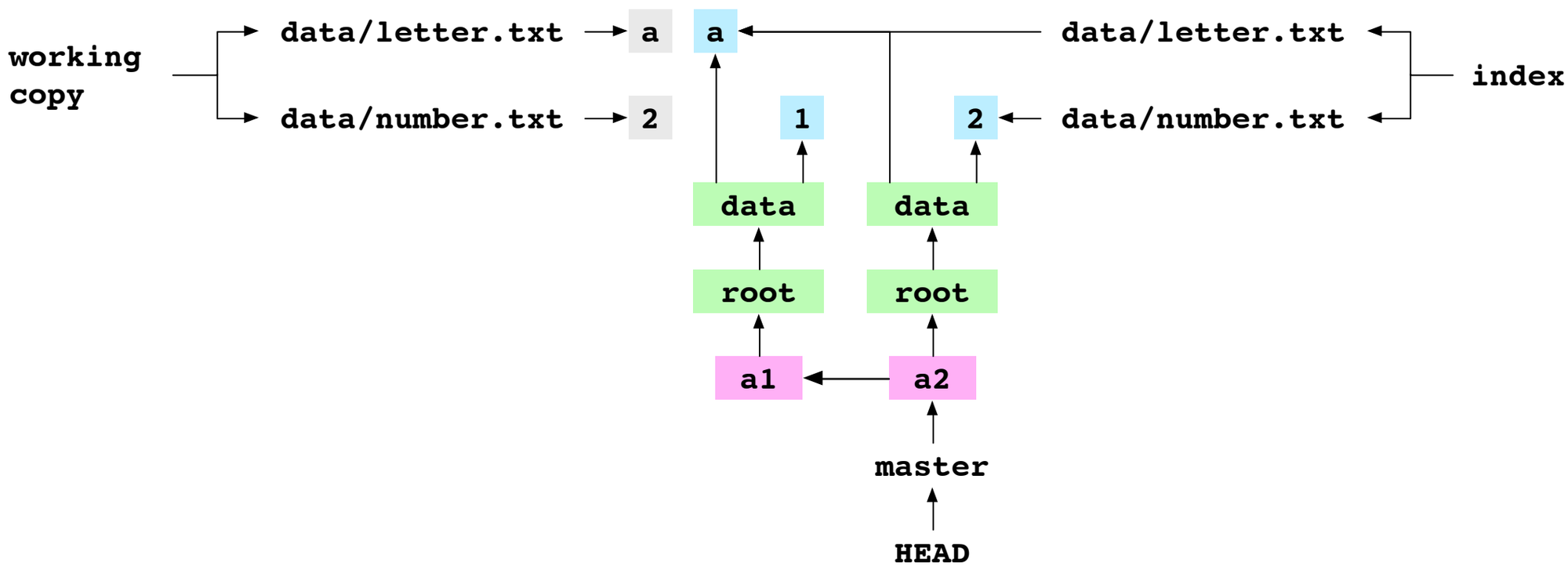
Check out a commit

```
~/alpha $ git checkout f0af
          HEAD is detached
```

```
~/alpha $ git commit -m 'a2'
          master f0af
```

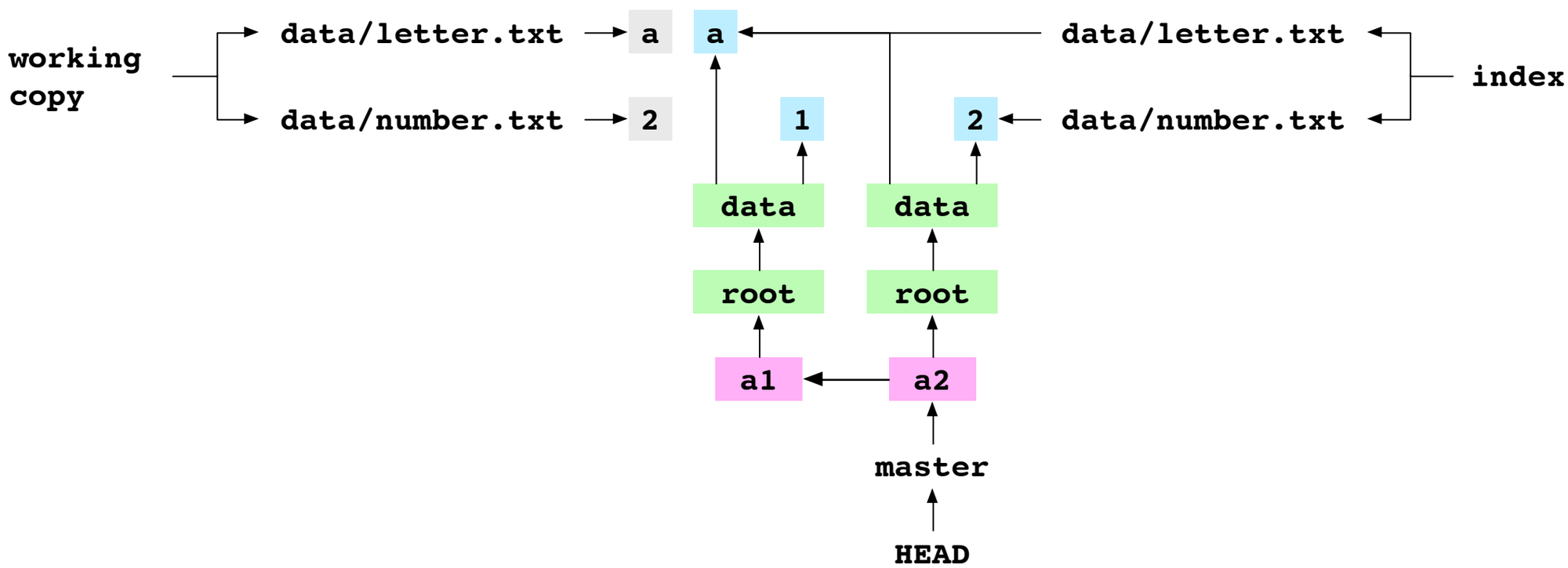
1. Write the commit tree to the working copy

```
~/alpha $ git checkout f0af
          HEAD is detached
```



2. Write the commit tree to the index

```
~/alpha $ git checkout f0af
          HEAD is detached
```



3. Point HEAD at the thing that was checked out

```
~/alpha $ git checkout f0af
          HEAD is detached
~/alpha $ cat .git/HEAD
          f0af
```

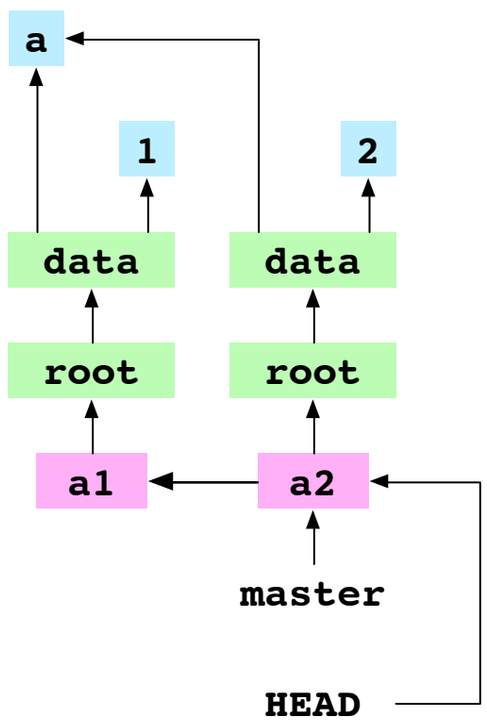
3. Point HEAD at the thing that was checked out

```
~/alpha $ git checkout f0af
          HEAD is detached
~/alpha $ cat .git/HEAD
          f0af
```

```
~/alpha $ cat .git/HEAD
          ref:      refs/heads/master
```

3. Point HEAD at the thing that was checked out

```
~/alpha $ git checkout f0af
          HEAD is detached
~/alpha $ cat .git/HEAD
          f0af
```



Make commit a3

```
~/alpha $ printf '3' > data/number.txt
```

Make commit a3

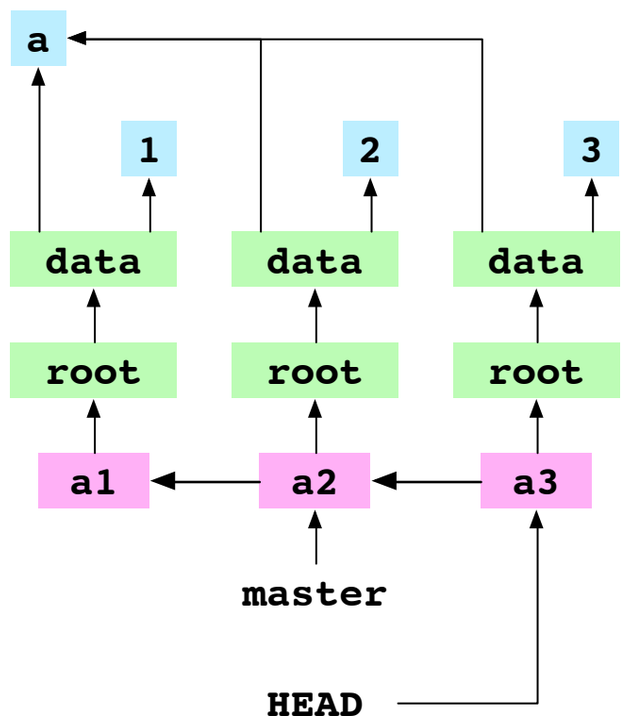
```
~/alpha $ printf '3' > data/number.txt
~/alpha $ git add data/number.txt
~/alpha $ git commit -m 'a3'
          detached HEAD 3645
```

Make commit a3

```
~/alpha $ printf '3' > data/number.txt
~/alpha $ git add data/number.txt
~/alpha $ git commit -m 'a3'
          detached HEAD 3645
```

Make commit a3

```
~/alpha $ printf '3' > data/number.txt
~/alpha $ git add data/number.txt
~/alpha $ git commit -m 'a3'
          detached HEAD 3645
```



Create a branch

Create a branch

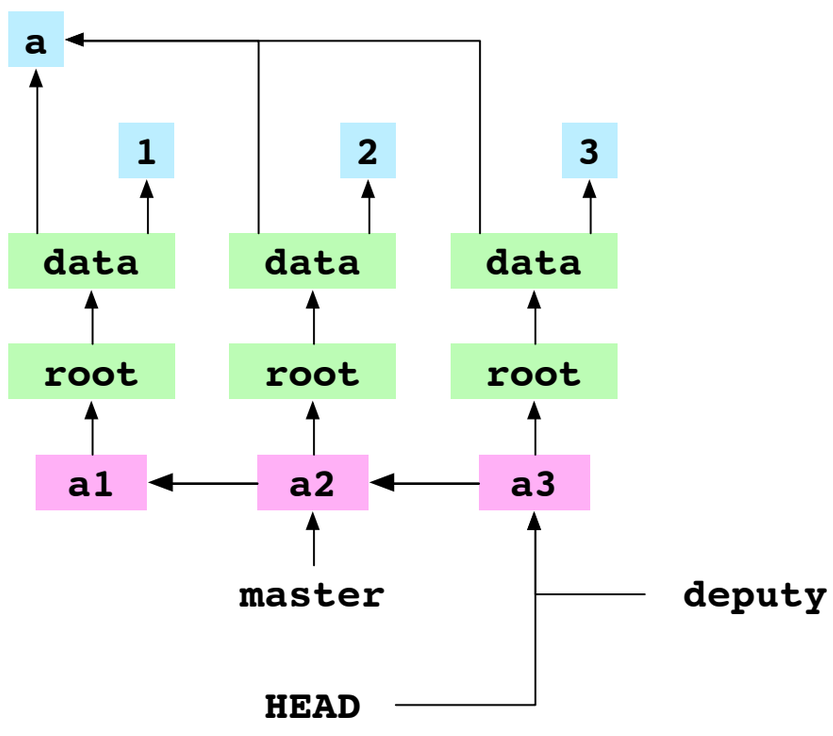
```
~/alpha $ git branch deputy
```

Create a branch

```
~/alpha $ git branch deputy
~/alpha $ cat .git/refs/heads/deputy
          3645
```

Create a branch

```
~/alpha $ git branch deputy
~/alpha $ cat .git/refs/heads/deputy
          3645
```



Branches are just refs, refs are just files

Branches are just refs, refs are just files

Branches are lightweight

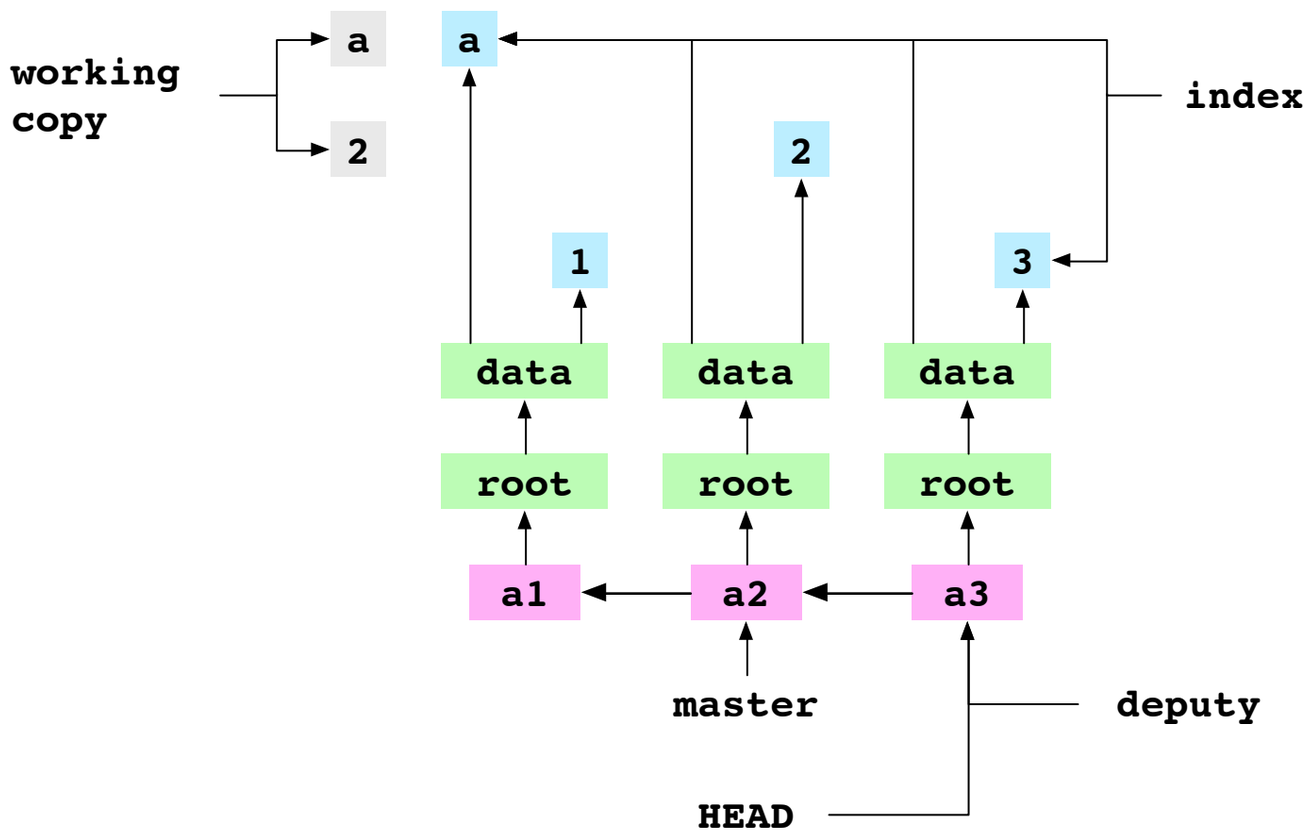
Check out a branch

Check out `master`

```
~/alpha $ git checkout master
          Switched to branch master
```

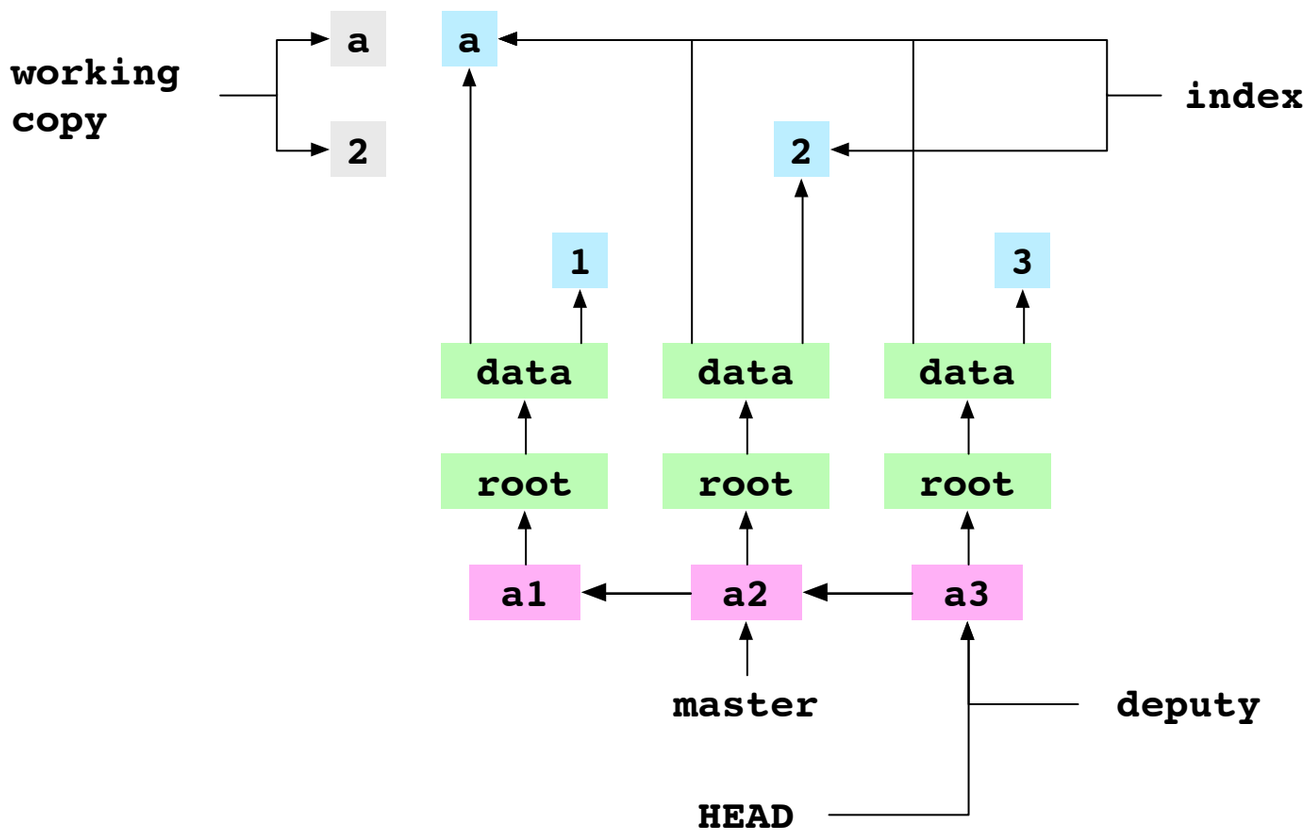
1. Write the commit tree to the working copy

```
~/alpha $ git checkout master
          Switched to branch master
```



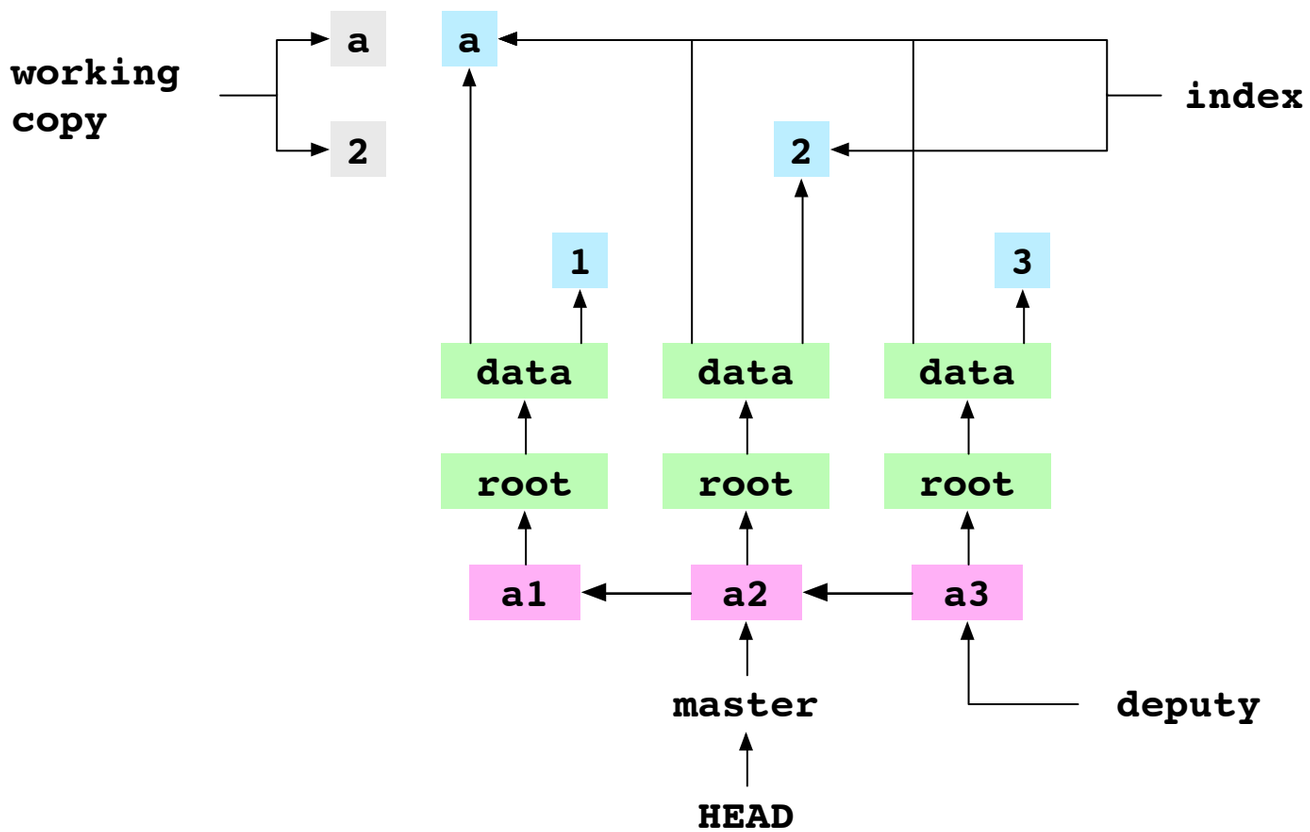
2. Write the commit tree to the index

```
~/alpha $ git checkout master
          Switched to branch master
```



3. Point HEAD at the thing that was checked out

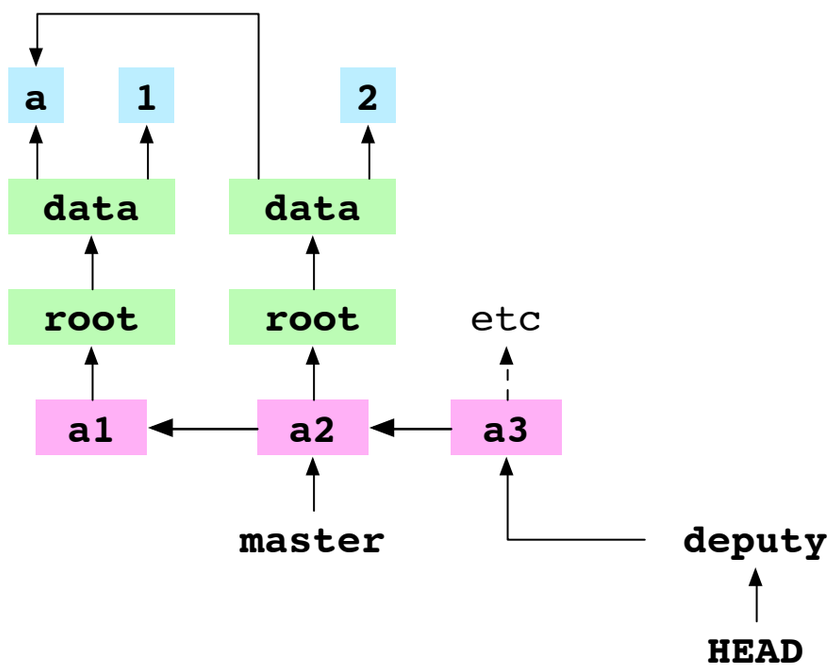
```
~/alpha $ git checkout master
          Switched to branch master
```



Merge an ancestor

Check out deputy

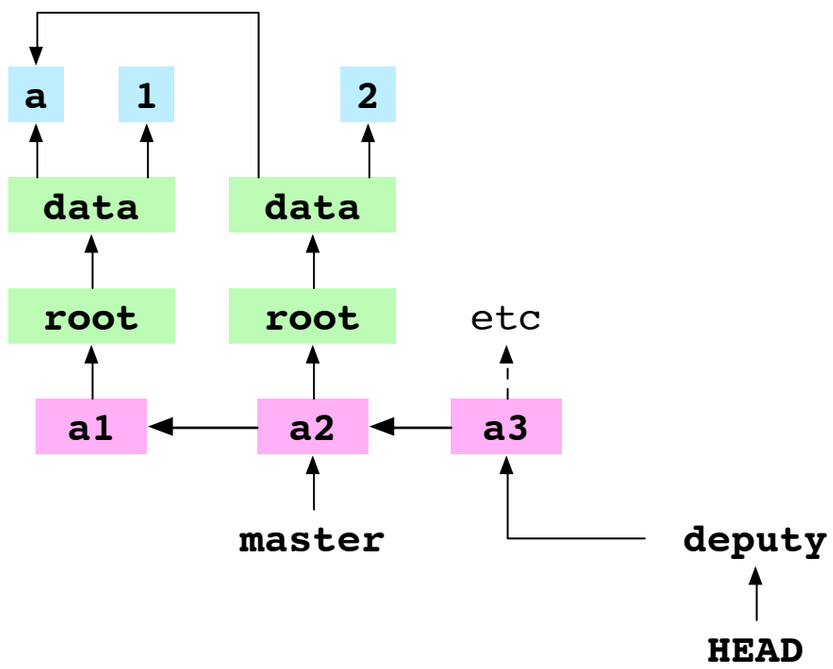
```
~/alpha $ git checkout deputy
          Switched to branch deputy
```



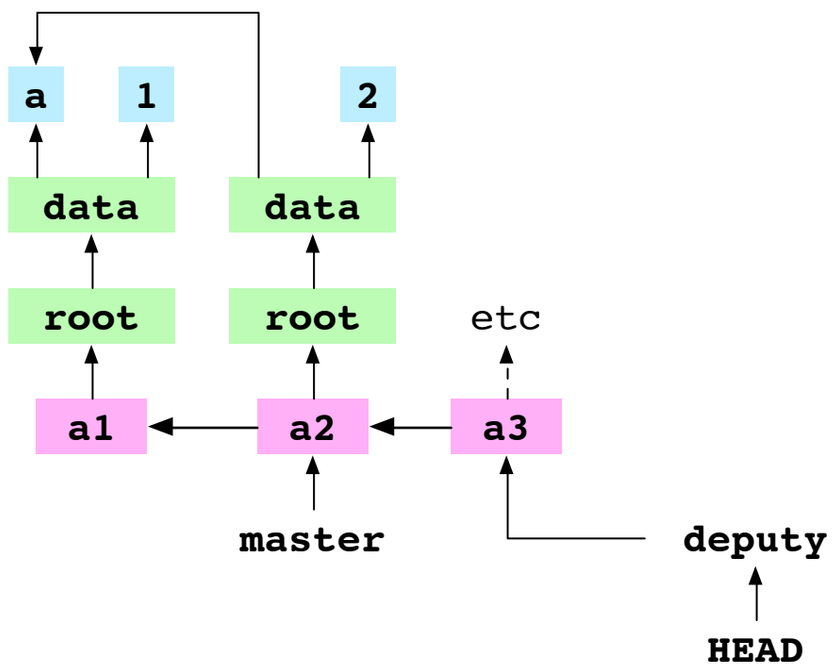
Merge master (a2) into deputy (a3)

```
~/alpha $ git merge master
          Already up-to-date
```

A commit is a set of changes



A commit is a set of changes

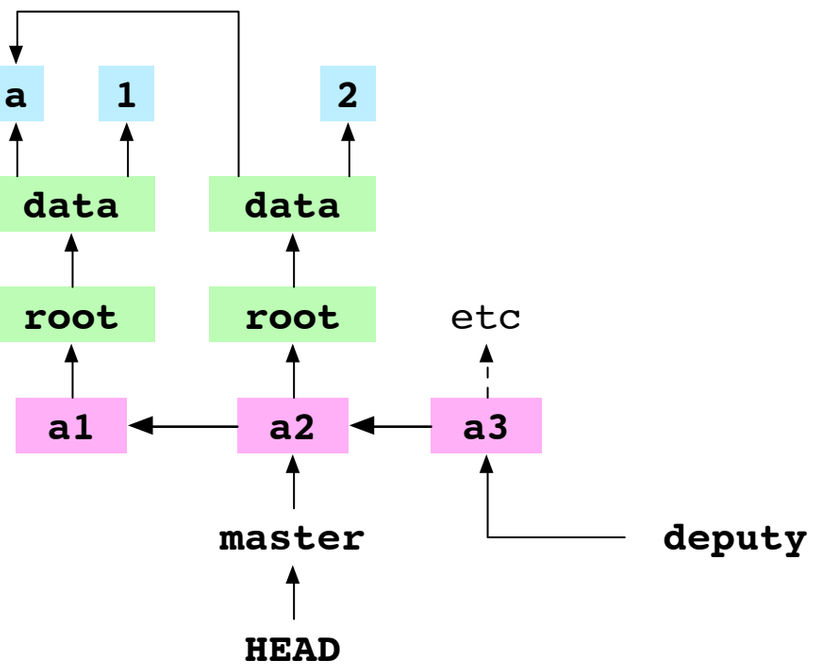


If an ancestor is merged into a descendent, Git does nothing

Merge a descendent

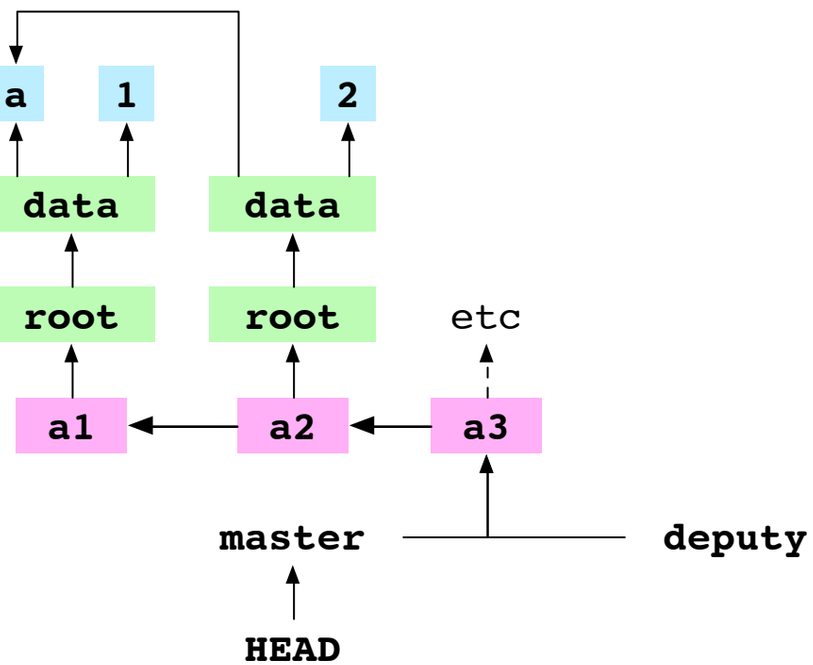
Check out `master`

```
~/alpha $ git checkout master
          Switched to branch master
```

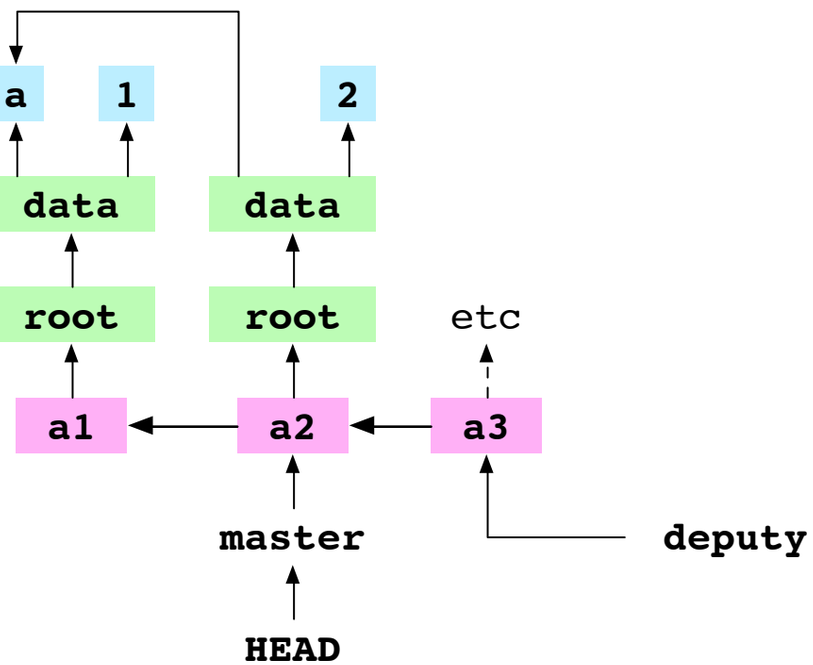


Merge deputy (a3) into master (a2)

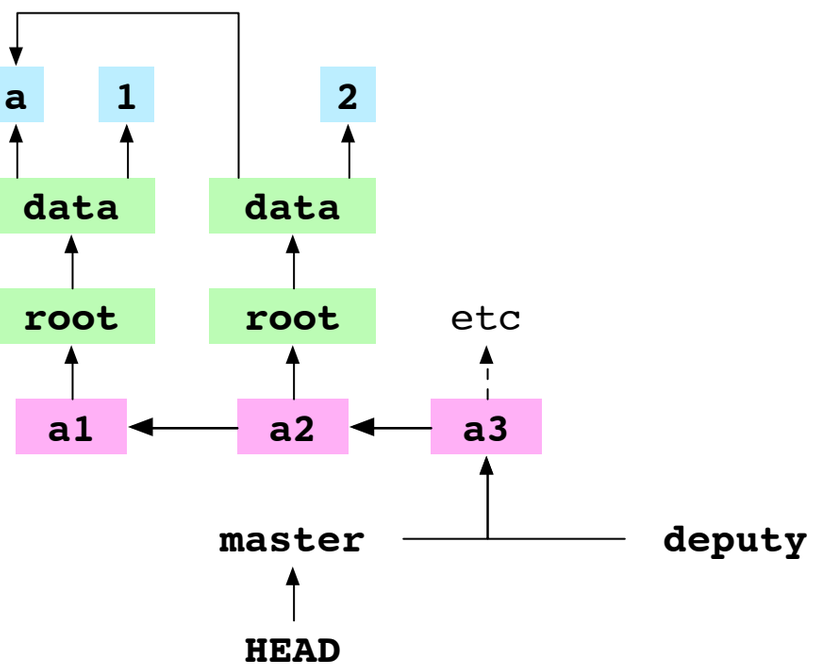
```
~/alpha $ git merge deputy
          Fast-forward
```



A commit is a set of changes



A commit is a set of changes



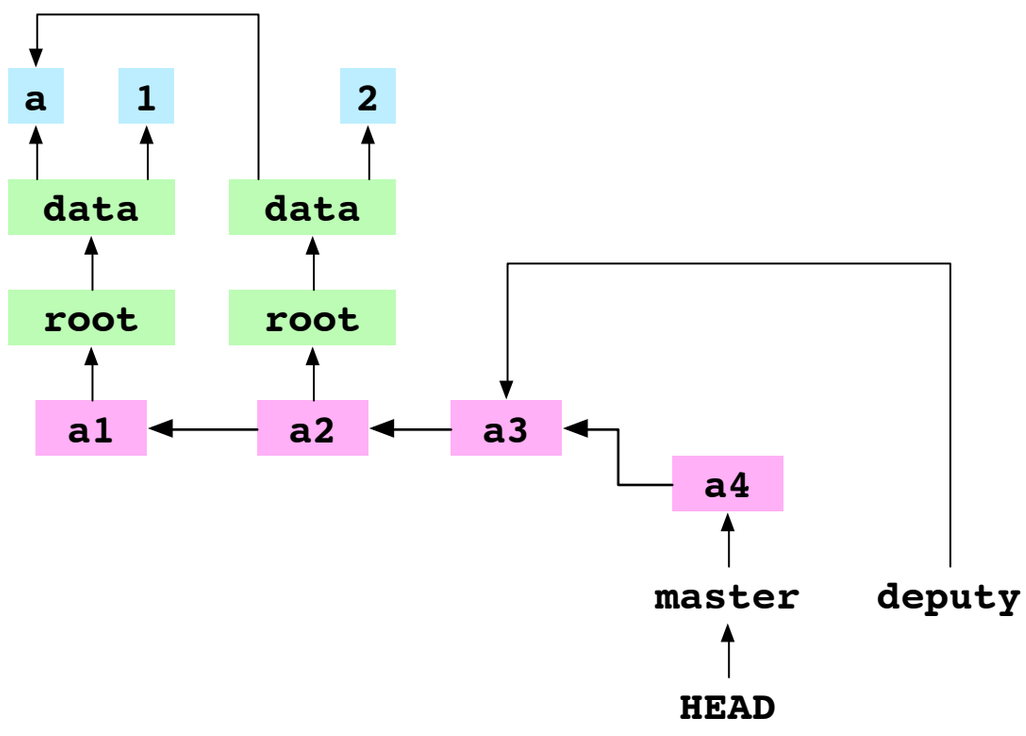
If a descendent is merged into an ancestor, history is not changed but `HEAD` *is* changed

Make commit a4 on master

```
~/alpha $ printf '4' > data/number.txt
~/alpha $ git add data/number.txt
~/alpha $ git commit -m 'a4'
          master 7b7b
```

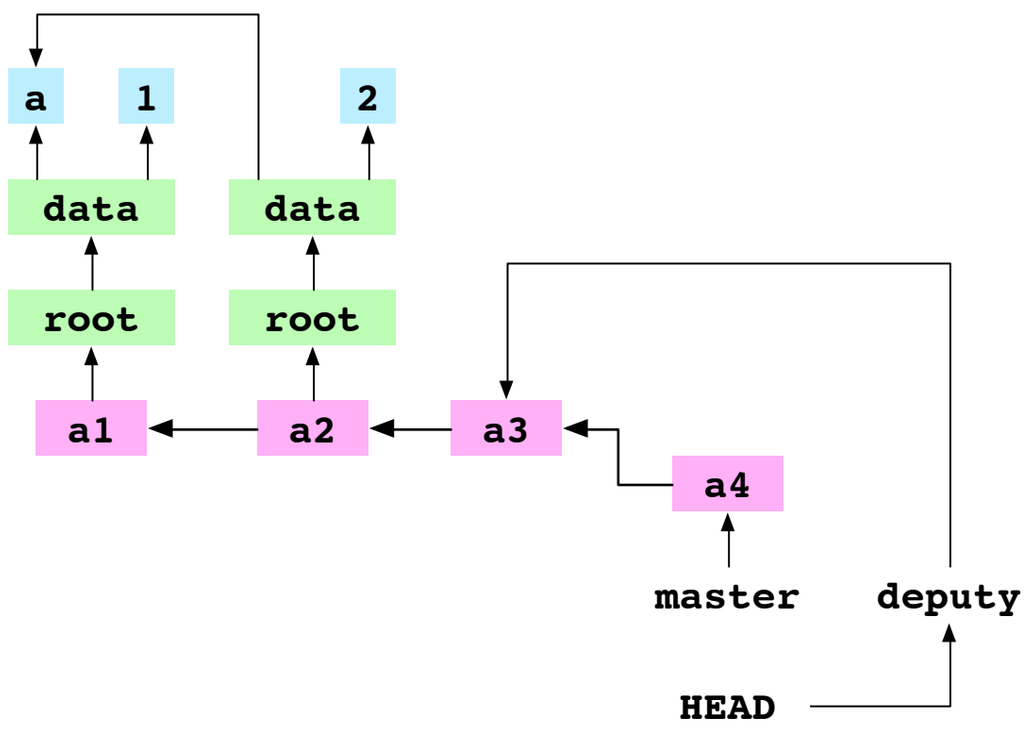
Make commit a4 on master

```
~/alpha $ printf '4' > data/number.txt
~/alpha $ git add data/number.txt
~/alpha $ git commit -m 'a4'
          master 7b7b
```



Check out deputy

```
~/alpha $ git checkout deputy
          Switched to branch deputy
```

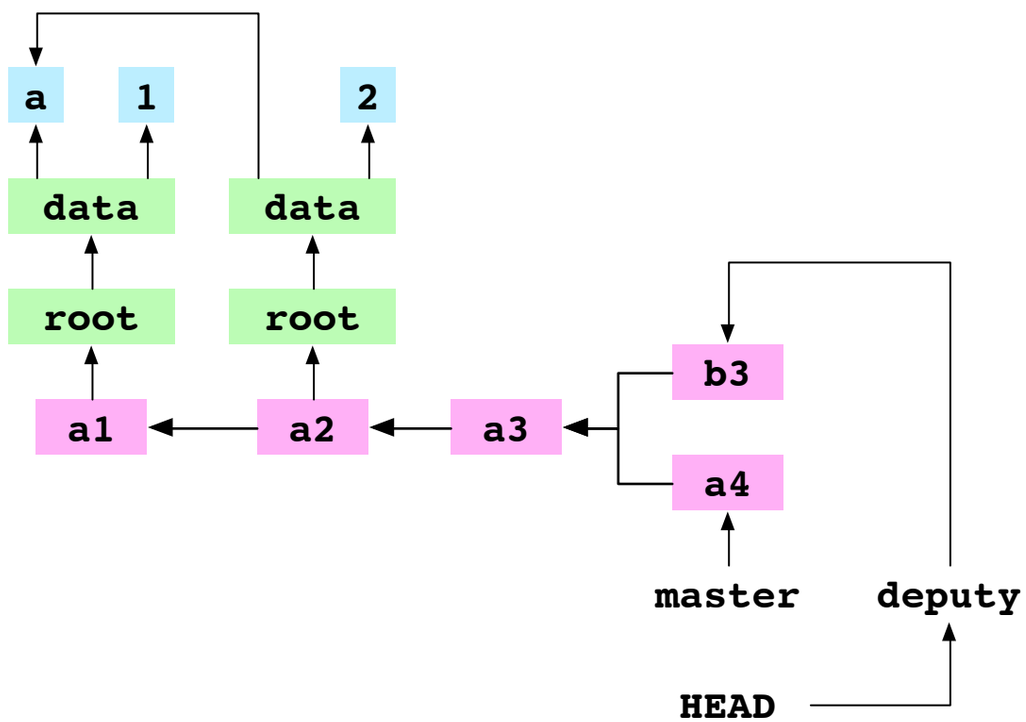


Make commit b3 to deputy

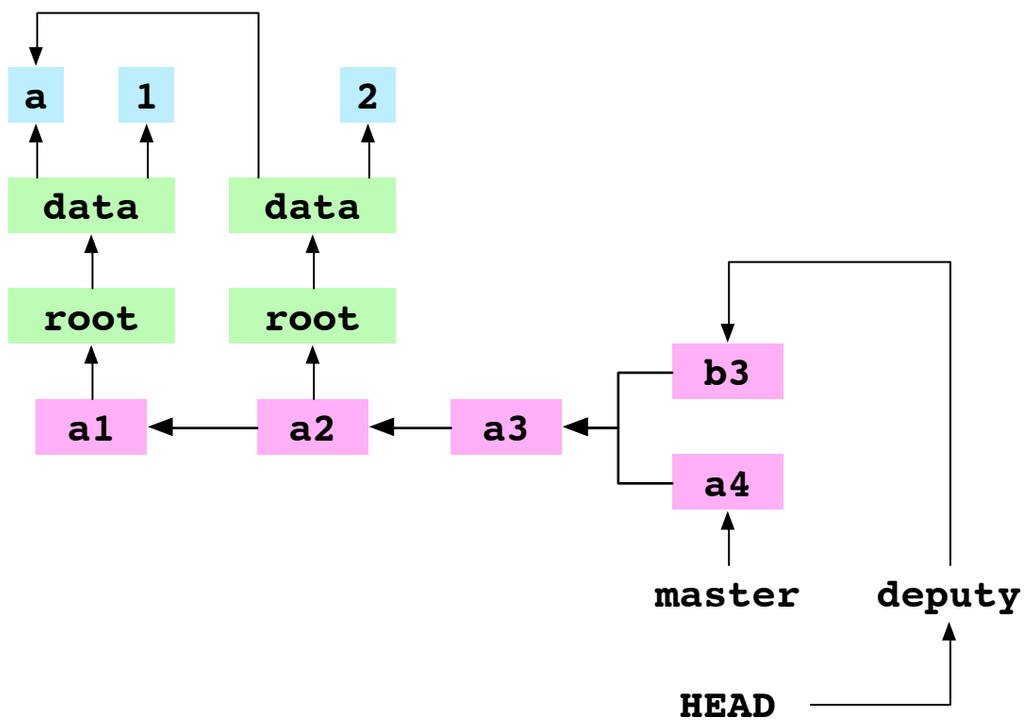
```
~/alpha $ printf 'b' > data/letter.txt
~/alpha $ git add data/letter.txt
~/alpha $ git commit -m 'b3'
          deputy 982d
```

Make commit b3 to deputy

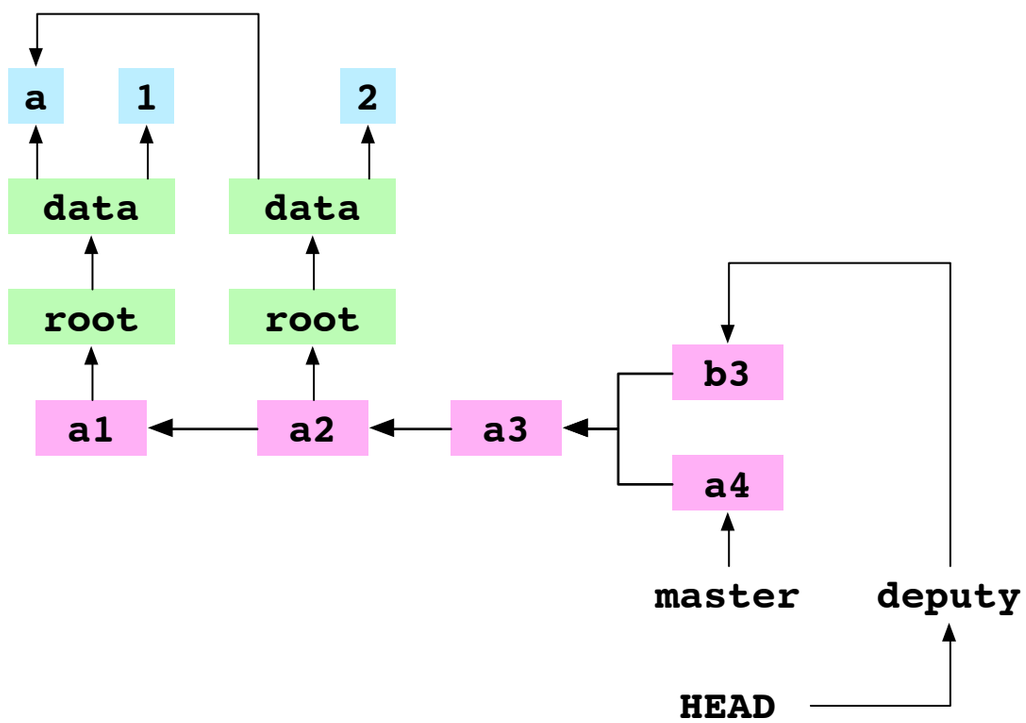
```
~/alpha $ printf 'b' > data/letter.txt
~/alpha $ git add data/letter.txt
~/alpha $ git commit -m 'b3'
          deputy 982d
```



Commits can share parents



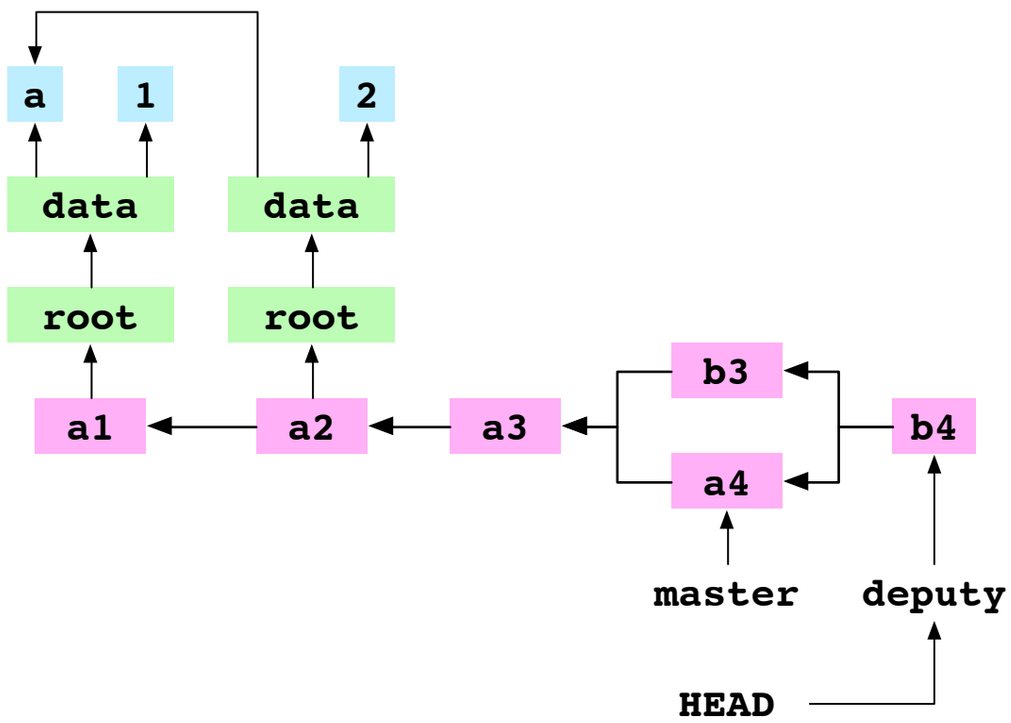
Commits can share parents



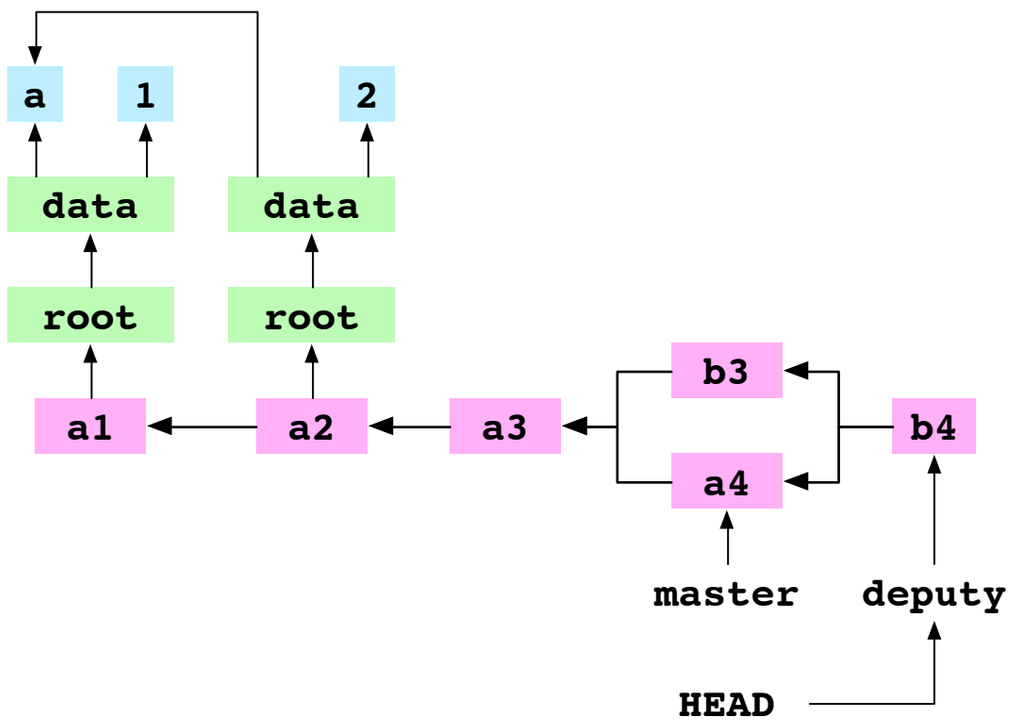
New lineages can be created

Merge two commits from different lineages

Commits can have multiple parents



Commits can have multiple parents



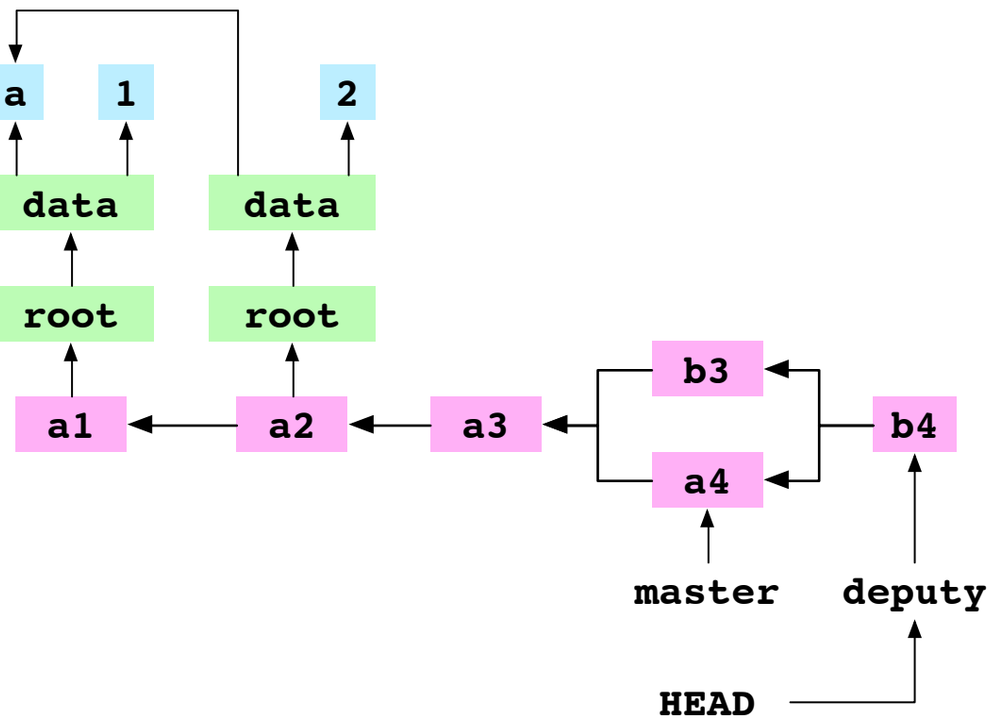
Lineages can be joined with a merge commit

Merge `master` (a4) into `deputy` (b3)

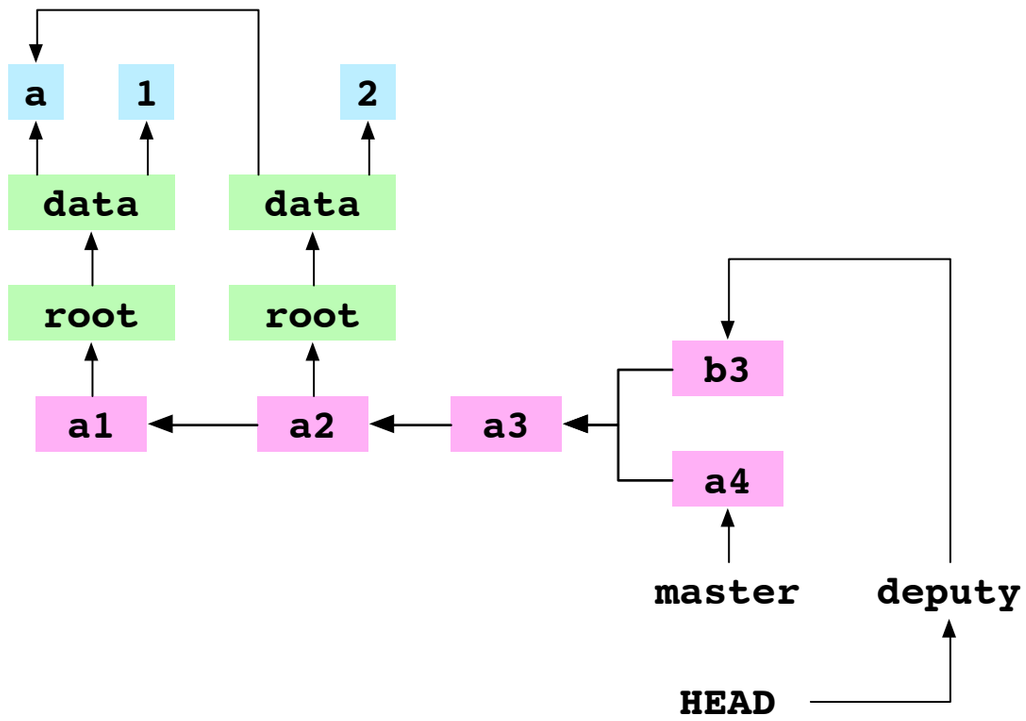
```
~/alpha $ git merge master -m 'b4'
          Merged
```

Merge `master` (a4) into `deputy` (b3)

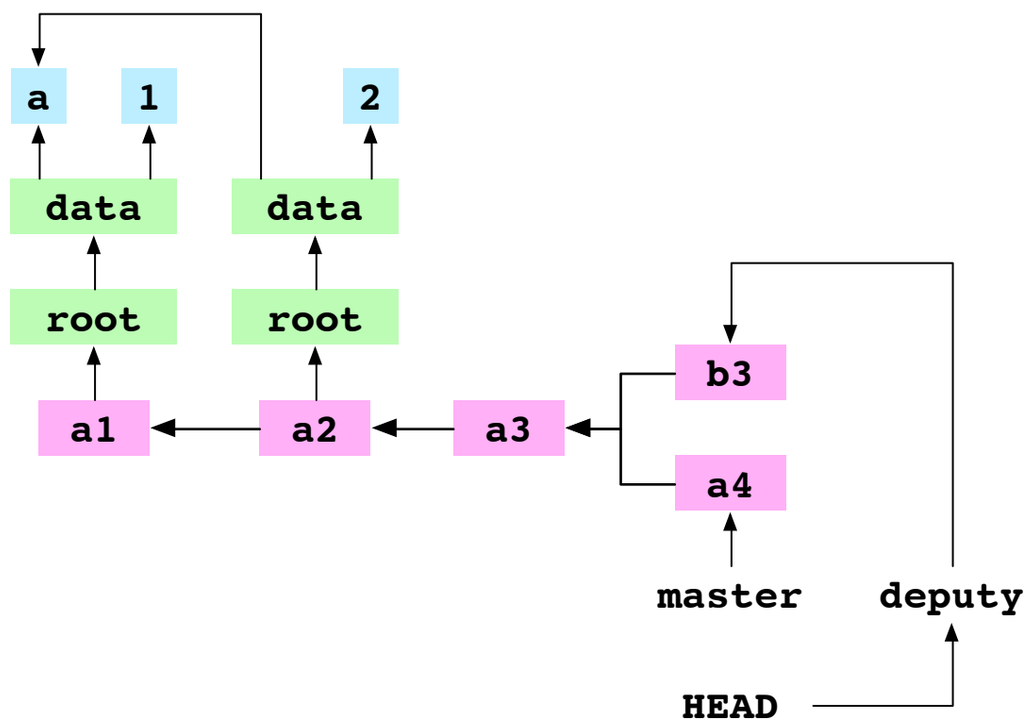
```
~/alpha $ git merge master -m 'b4'
          Merged
```



Commits have parents

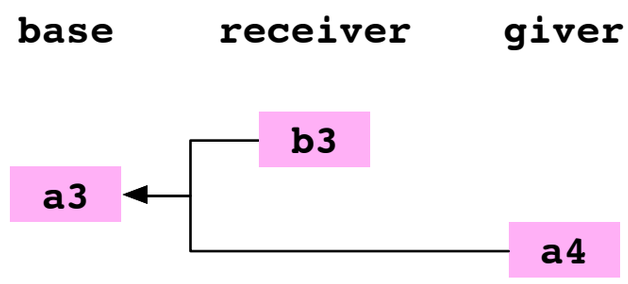


Commits have parents

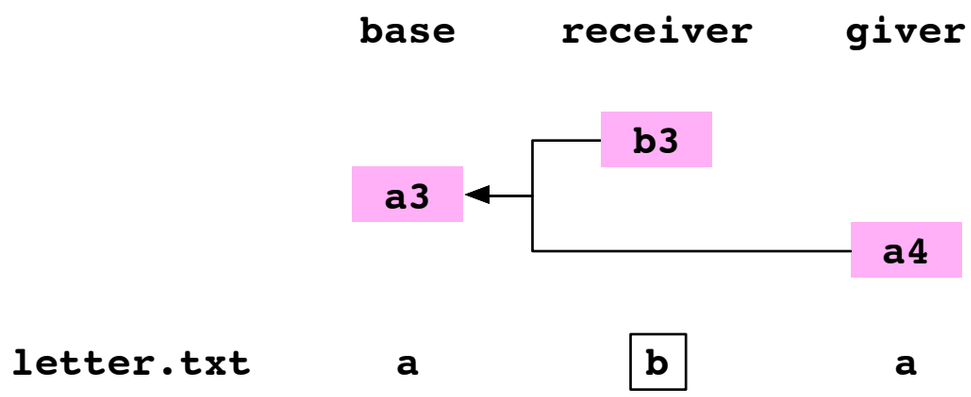


It is possible to find the point at which two lineages diverged

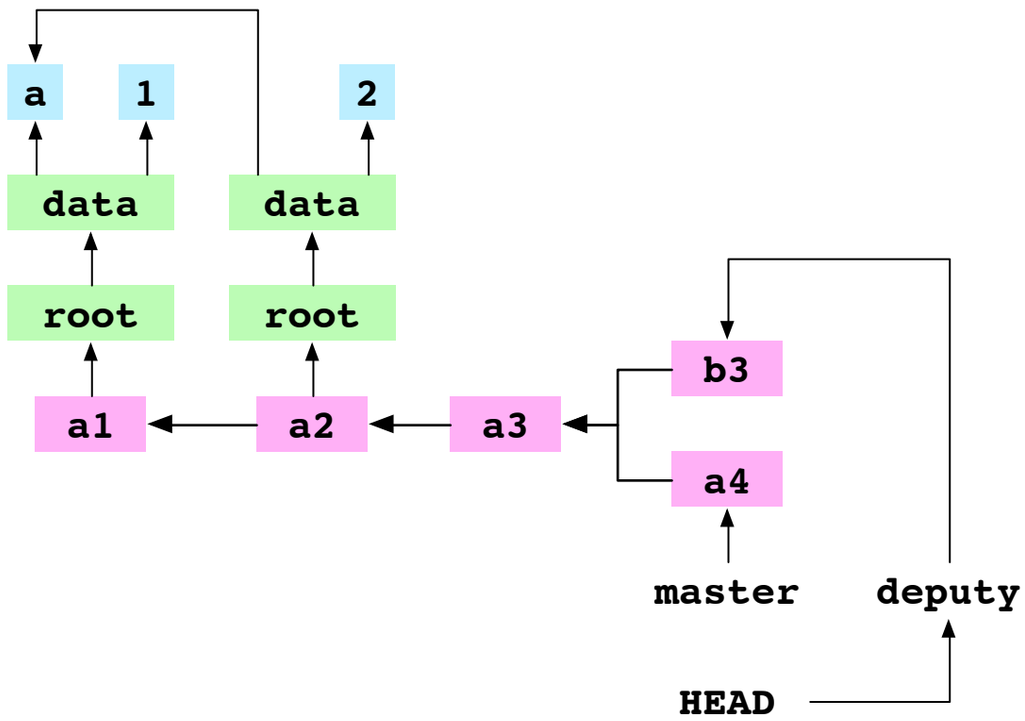
1. Generate the diff that combines the changes made by the receiver and giver



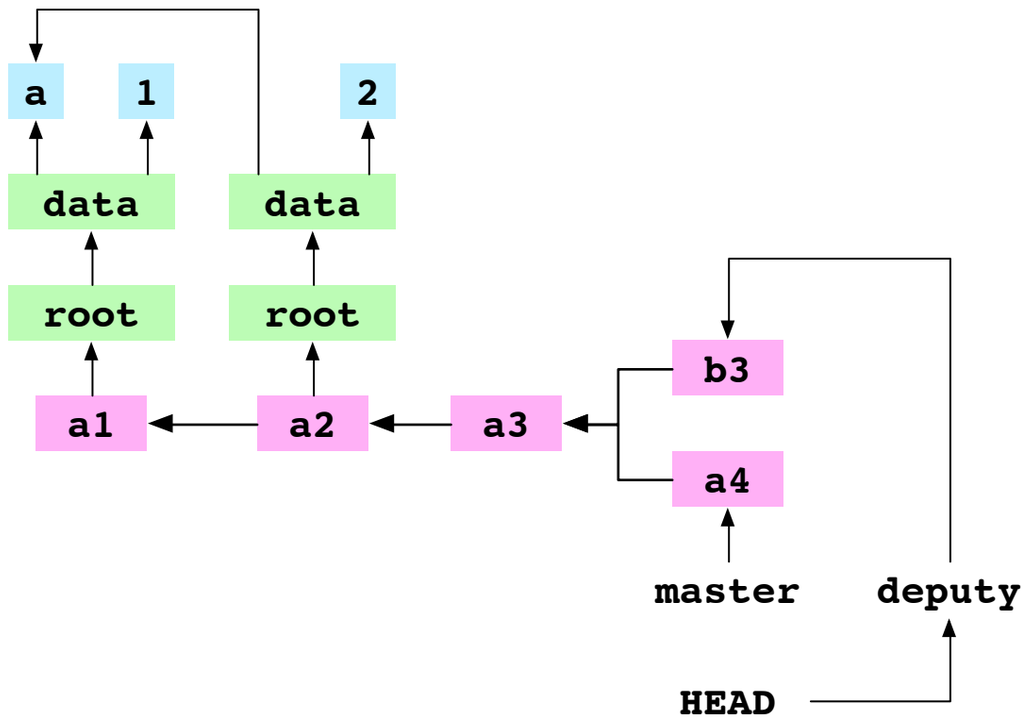
1. Generate the diff that combines the changes made by the receiver and giver



A merge has a base commit

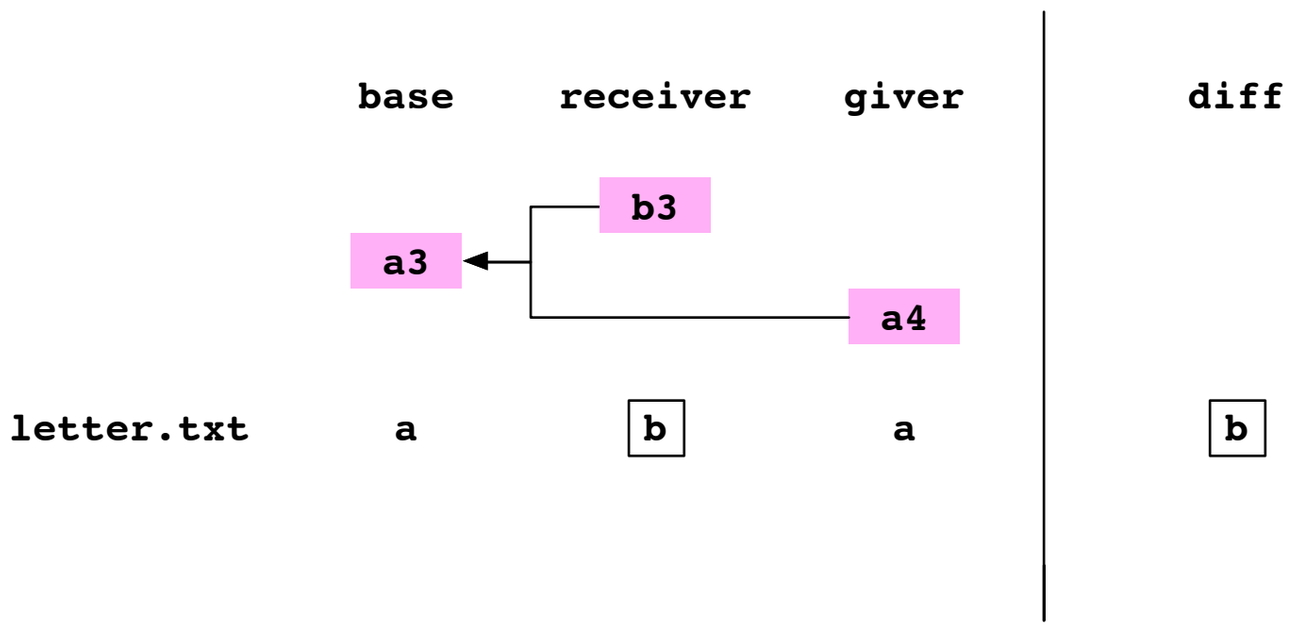


A merge has a base commit

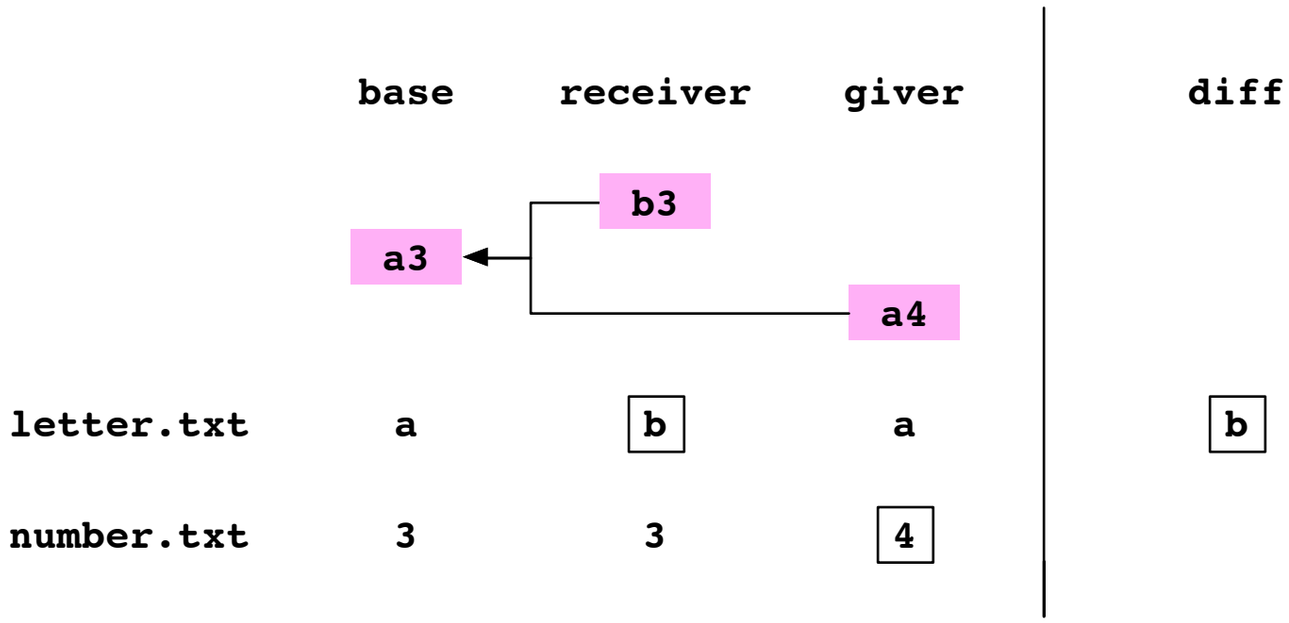


Git can automatically resolve the merge of a file that has changed from the base in only the receiver or giver

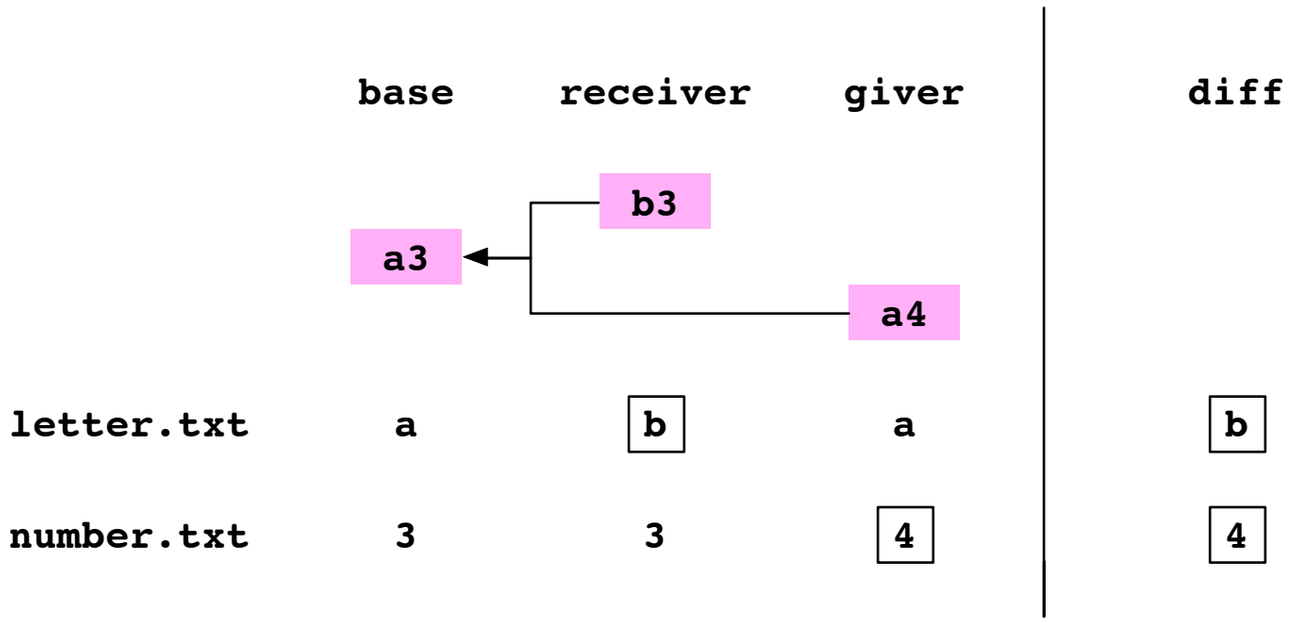
1. Generate the diff that combines the changes made by the receiver and giver



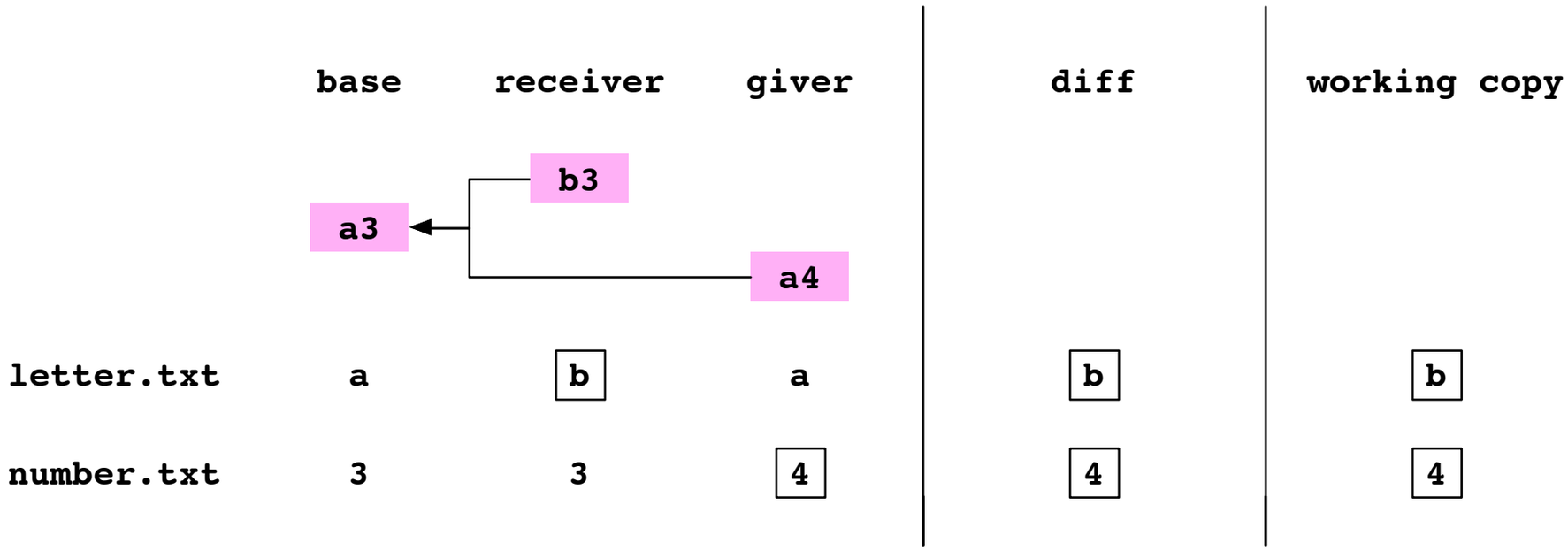
1. Generate the diff that combines the changes made by the receiver and giver



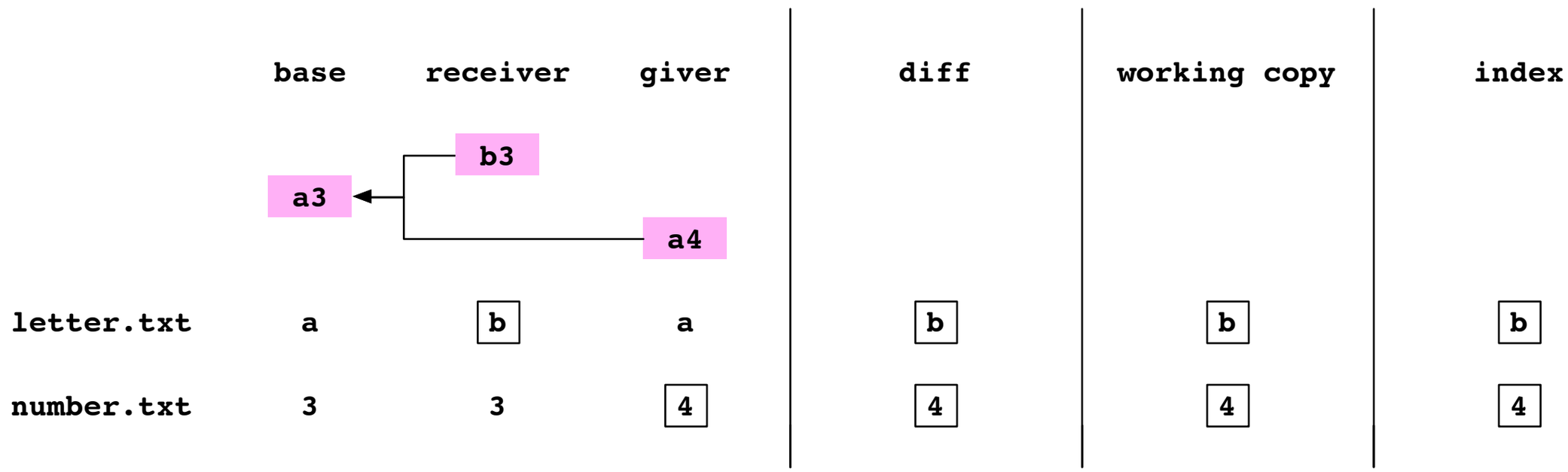
1. Generate the diff that combines the changes made by the receiver and giver



2. Apply the diff to the working copy

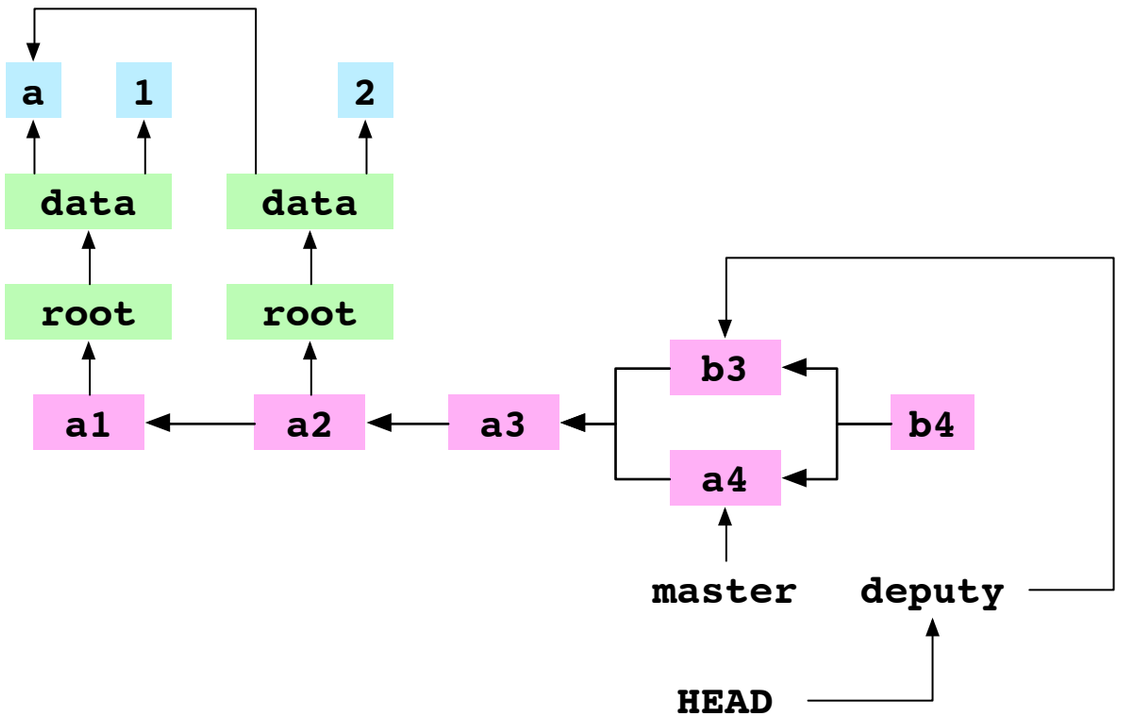


3. Apply the diff to the index



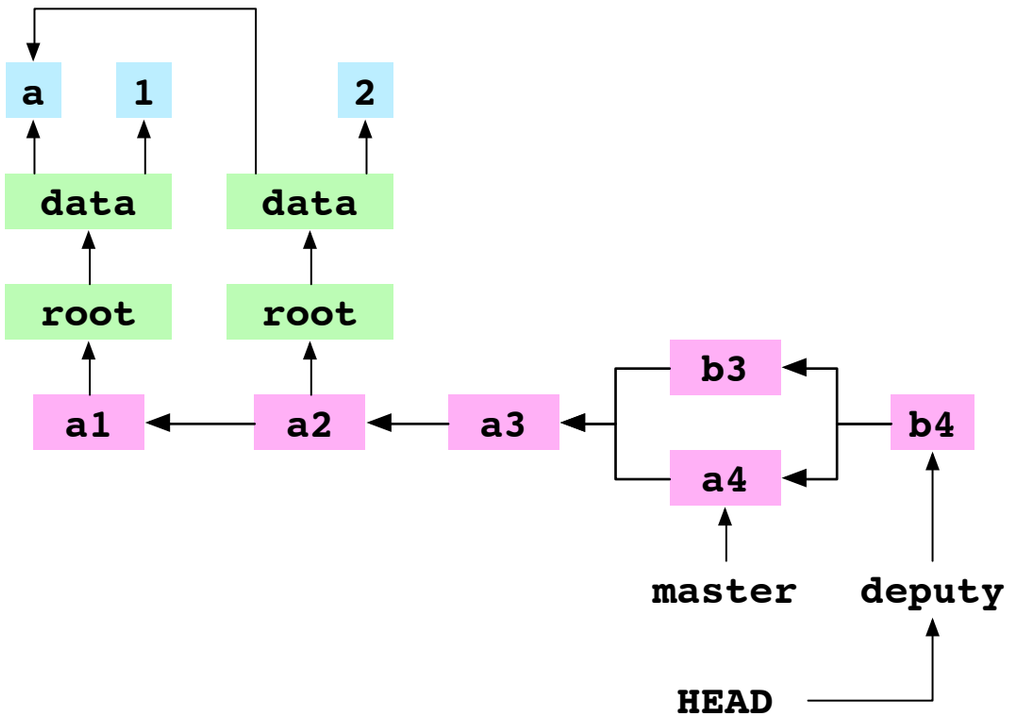
4. Commit the updated index

```
~/alpha $ git cat-file -p a2ec
          tree 2029
          parent 982d
          parent 7b7b
          author mr@c.com 1424798436
          b4
```



5. Point HEAD at the new commit

```
~/alpha $ cat .git/refs/heads/deputy
          a2ec
```



Merge commits from different lineages, where the commits both modify the same file

Check out `master`

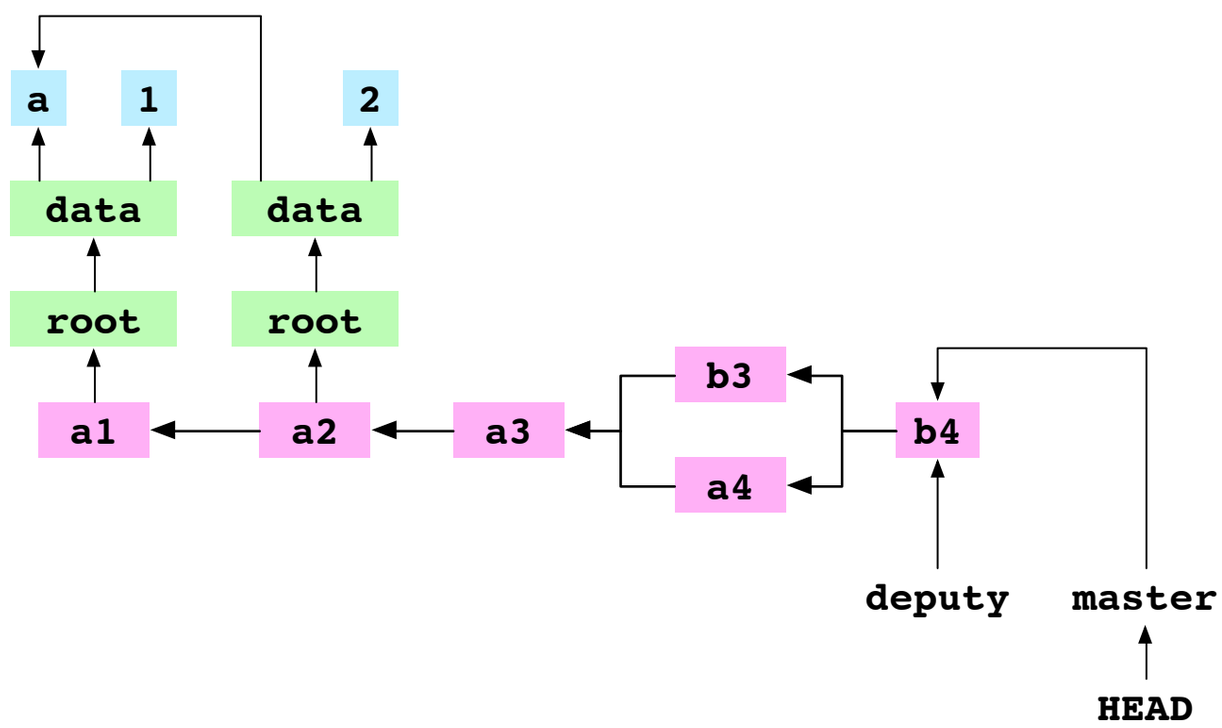
```
~/alpha $ git checkout master
          Switched to branch master
```

Merge `deputy` into `master` to bring `master` up to date

```
~/alpha $ git checkout master
          Switched to branch master
~/alpha $ git merge deputy
          Fast-forward
```

Merge `deputy` into `master` to bring `master` up to date

```
~/alpha $ git checkout master
          Switched to branch master
~/alpha $ git merge deputy
          Fast-forward
```



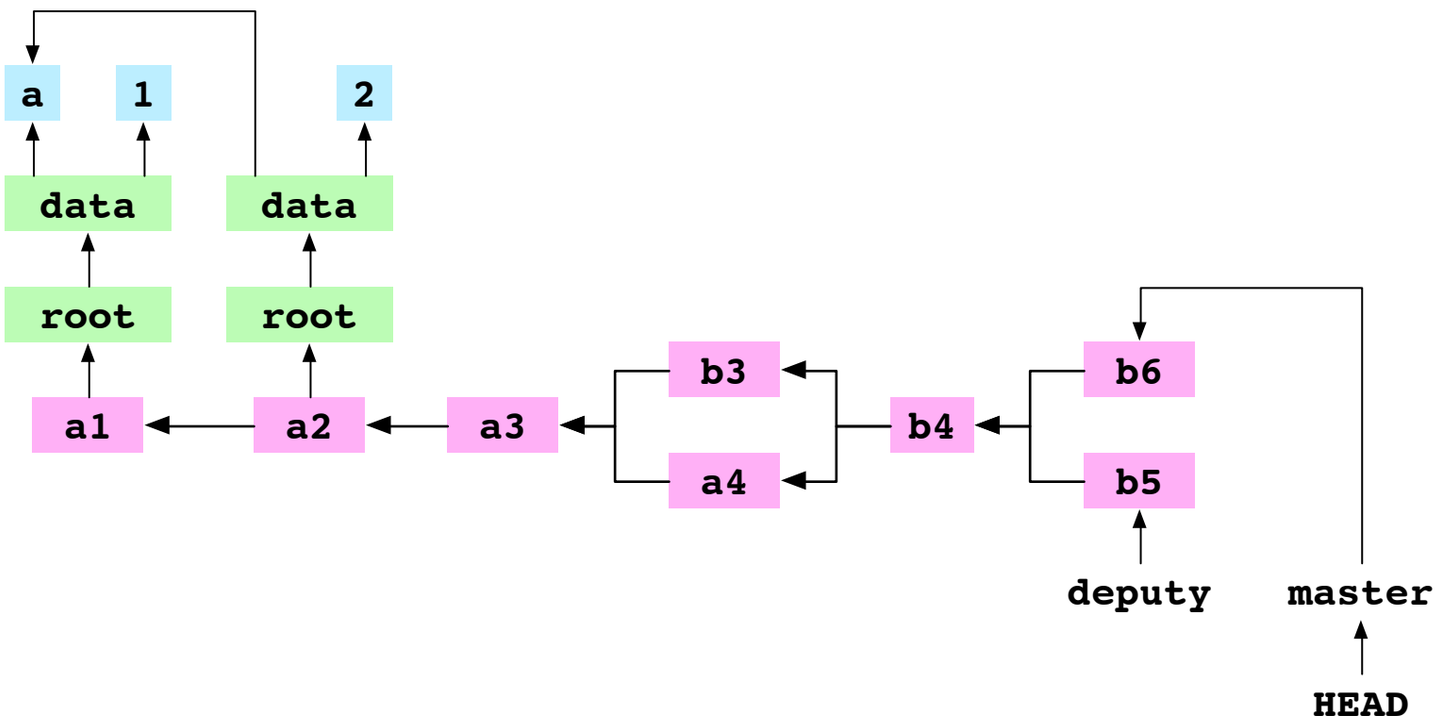
Make commit b5 to deputy

```
~/alpha $ git checkout deputy
          Switched to branch deputy
~/alpha $ printf '5' > data/number.txt
~/alpha $ git add data/number.txt
~/alpha $ git commit -m 'b5'
          deputy bd79
```

Make commit b6 to master

```
~/alpha $ git checkout master
          Switched to branch master
~/alpha $ printf '6' > data/number.txt
~/alpha $ git add data/number.txt
~/alpha $ git commit -m 'b6'
          master 4c3c
```

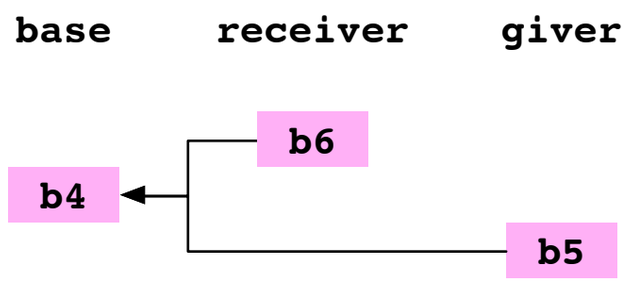
After making commits b5 and b6



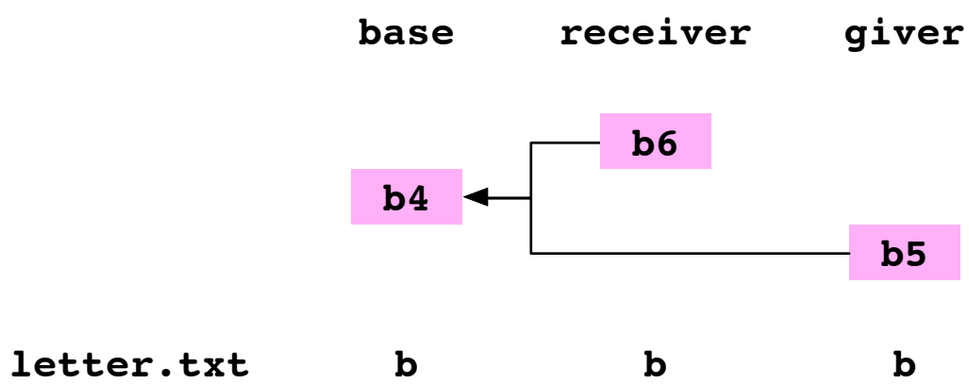
Merge deputy (b5) into master (b6)

```
~/alpha $ git merge deputy
          Conflict in data/number.txt
```

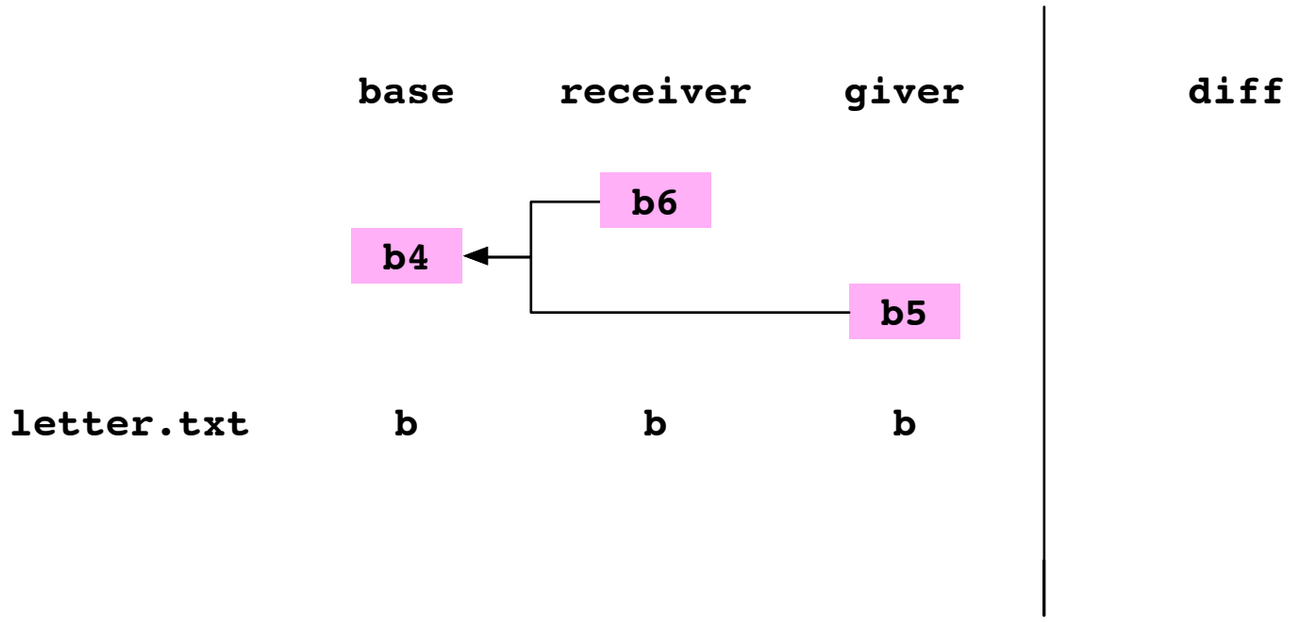
1. Generate the diff that combines the changes made by the receiver and giver



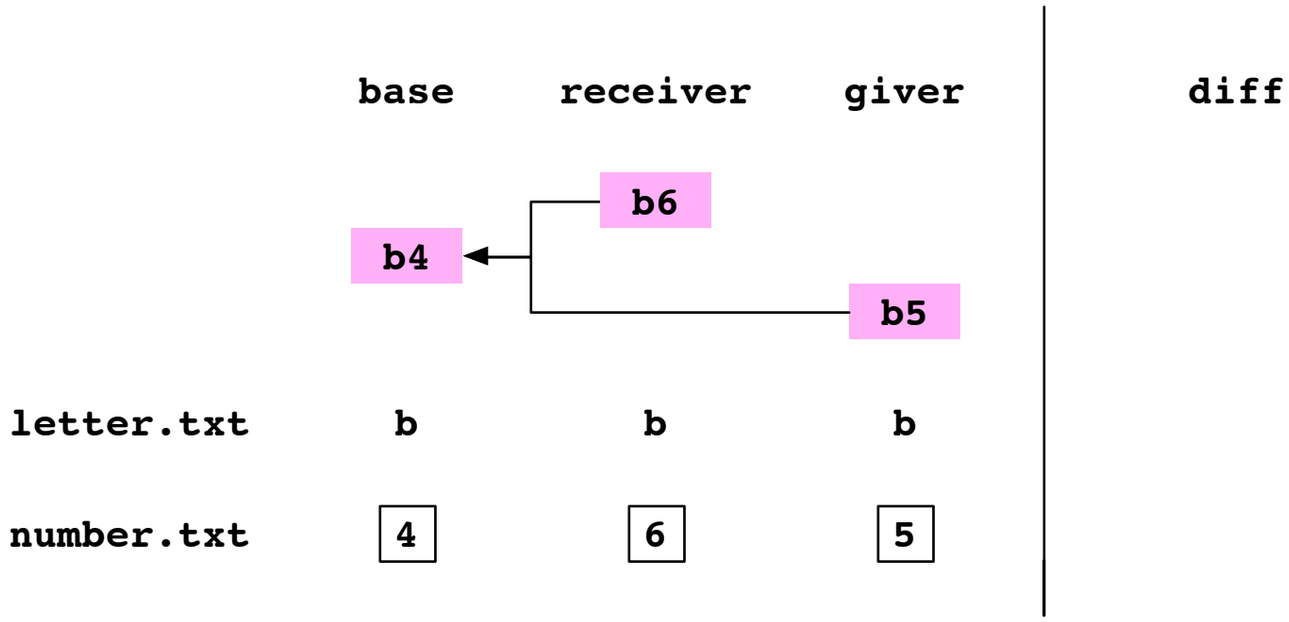
1. Generate the diff that combines the changes made by the receiver and giver



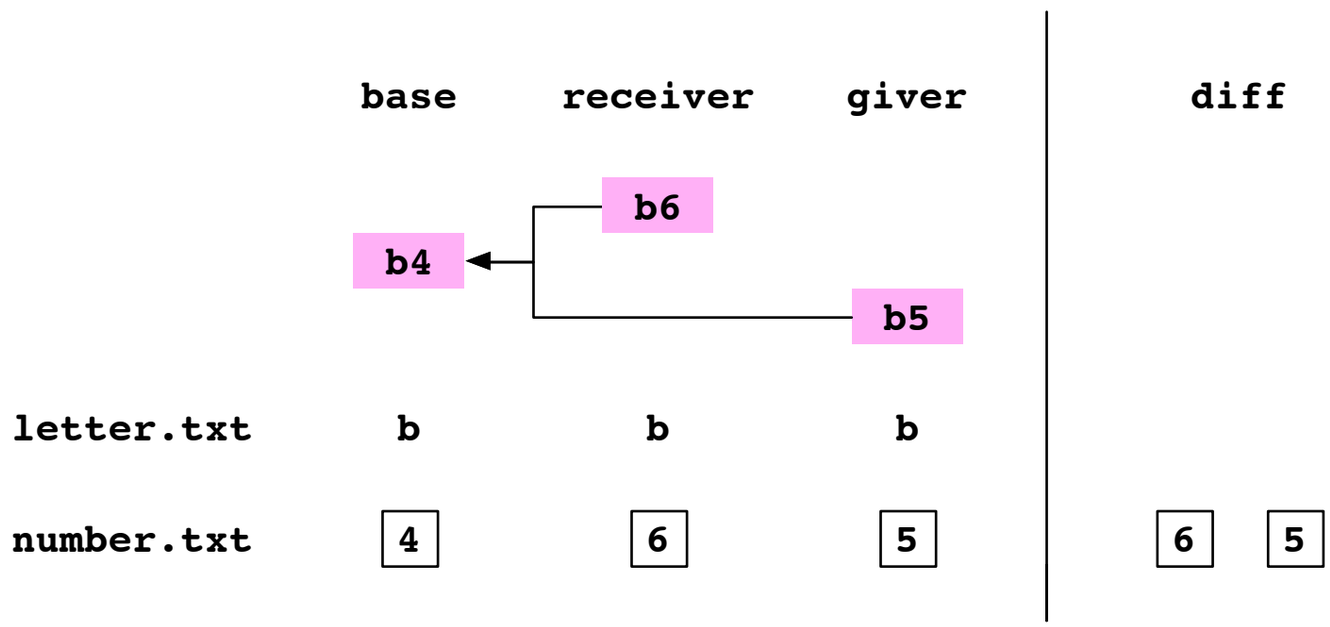
1. Generate the diff that combines the changes made by the receiver and giver



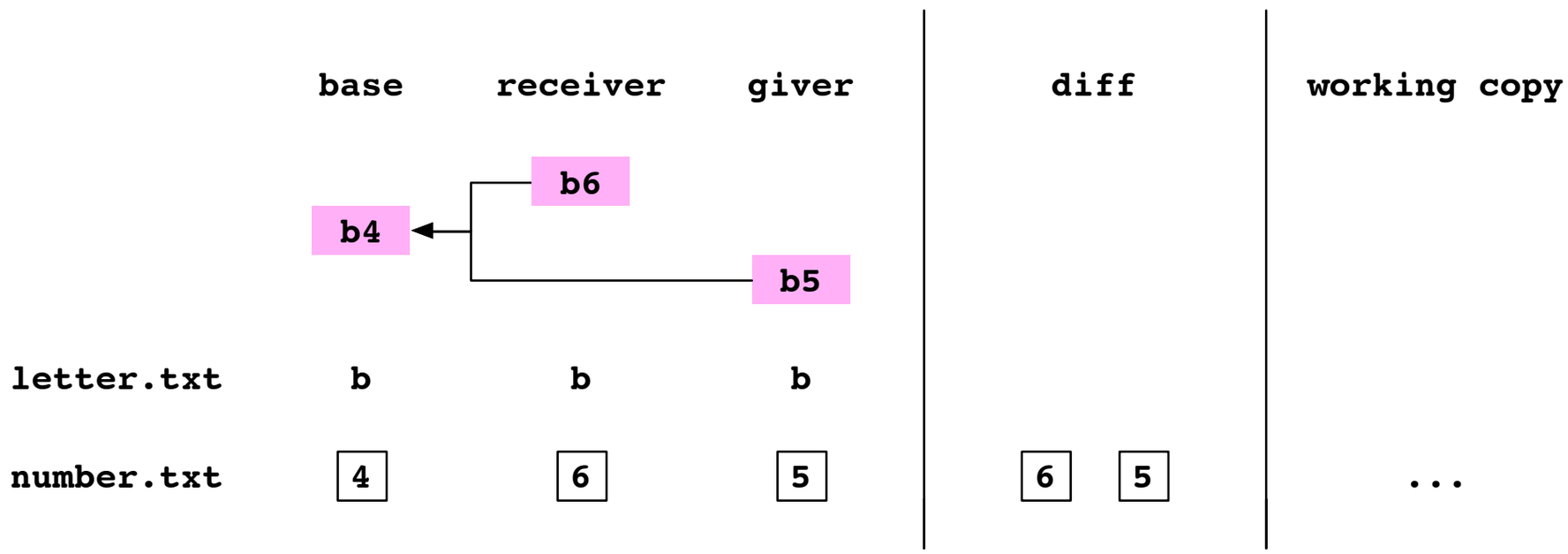
1. Generate the diff that combines the changes made by the receiver and giver



1. Generate the diff that combines the changes made by the receiver and giver



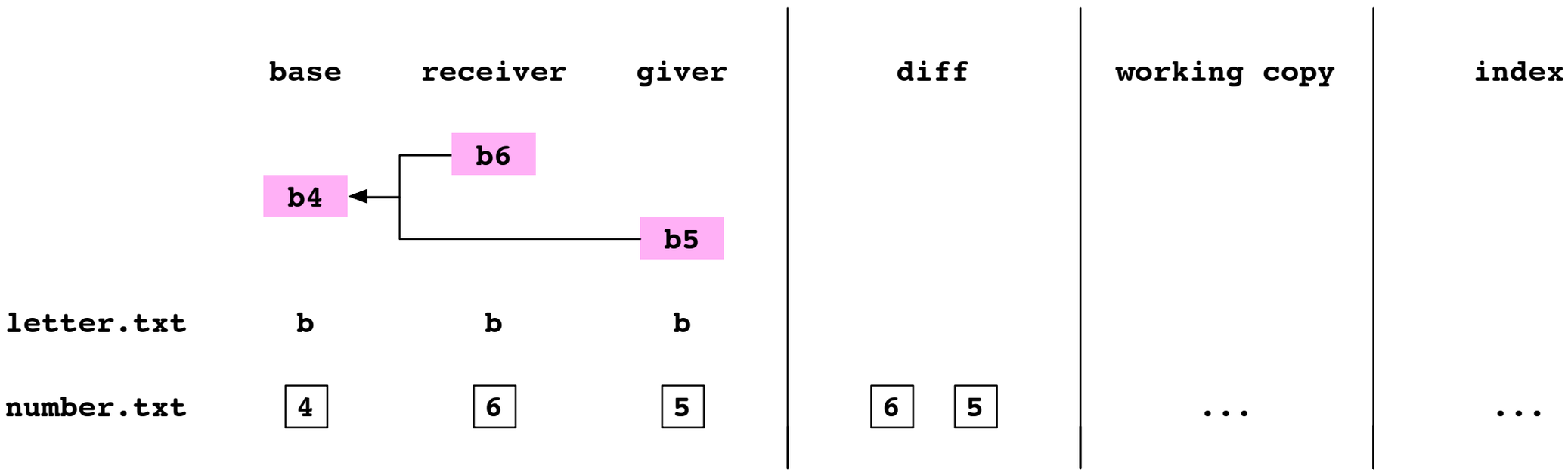
2. Apply the diff to the working copy



The `number.txt` conflict in the working copy

```
<<<<<<< HEAD
6
=======
5
>>>>>>> deputy
```

3. Apply the diff to the index



The index before the merge

```
0 data/letter.txt 63d8
0 data/number.txt 62f9
```

The index before the merge

```
0 data/letter.txt 63d8
0 data/number.txt 62f9
```

The index after the merge

```
0 data/letter.txt 63d8
1 data/number.txt bf0d
2 data/number.txt 62f9
3 data/number.txt 7813
```

The index after the merge

```
0 data/letter.txt 63d8
1 data/number.txt bf0d
2 data/number.txt 62f9
3 data/number.txt 7813
```

The index after the merge

```
0 data/letter.txt 63d8
1 data/number.txt bf0d
2 data/number.txt 62f9
3 data/number.txt 7813
```

The index after the merge

```
0 data/letter.txt 63d8
1 data/number.txt bf0d
2 data/number.txt 62f9
3 data/number.txt 7813
```

4. The user resolves the conflicts in the working copy

```
~/alpha $ printf '11' > data/number.txt
```

4. The user resolves the conflicts in the index

```
~/alpha $ printf '11' > data/number.txt
~/alpha $ git add data/number.txt
```

The index after the conflict in `number.txt` was resolved

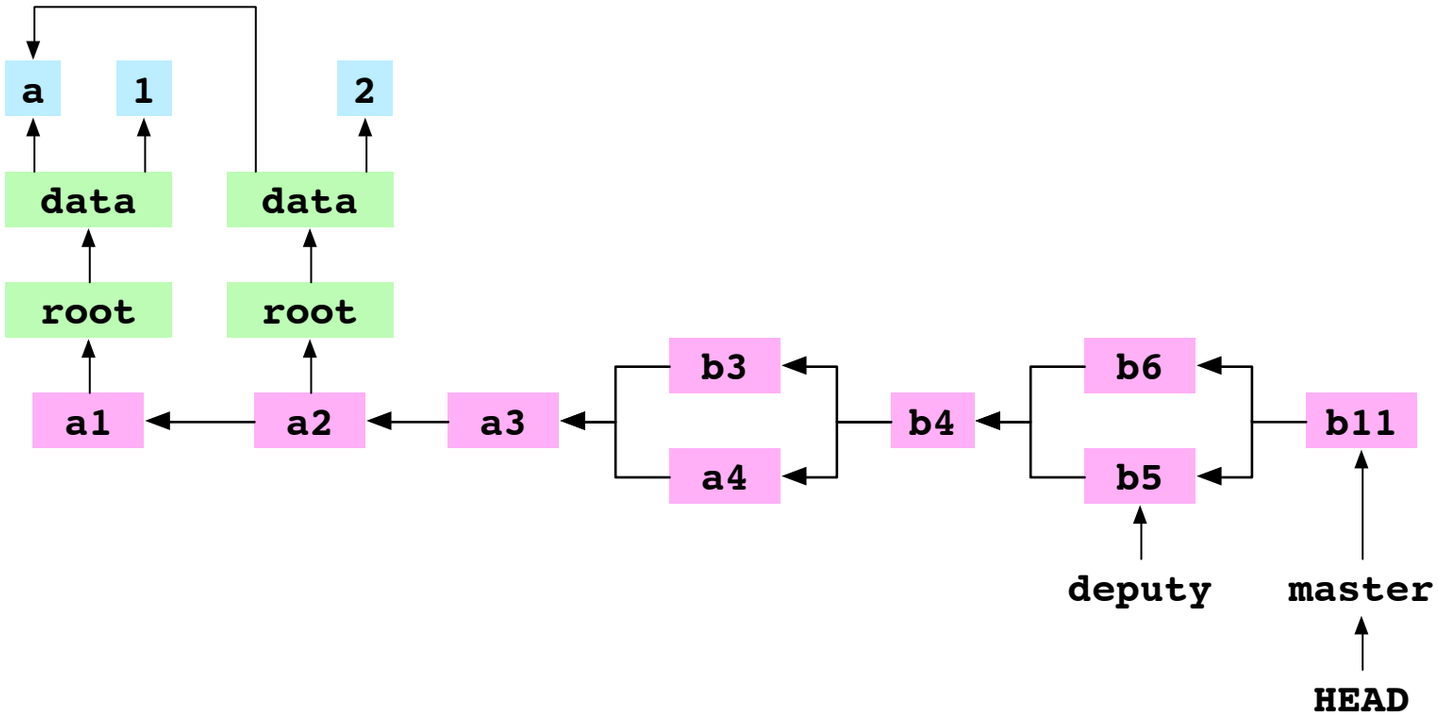
```
0 data/letter.txt 63d8
0 data/number.txt 9d60
```

6. The user commits the merge

```
~/alpha $ git commit -m 'b11'
          master 251a
```

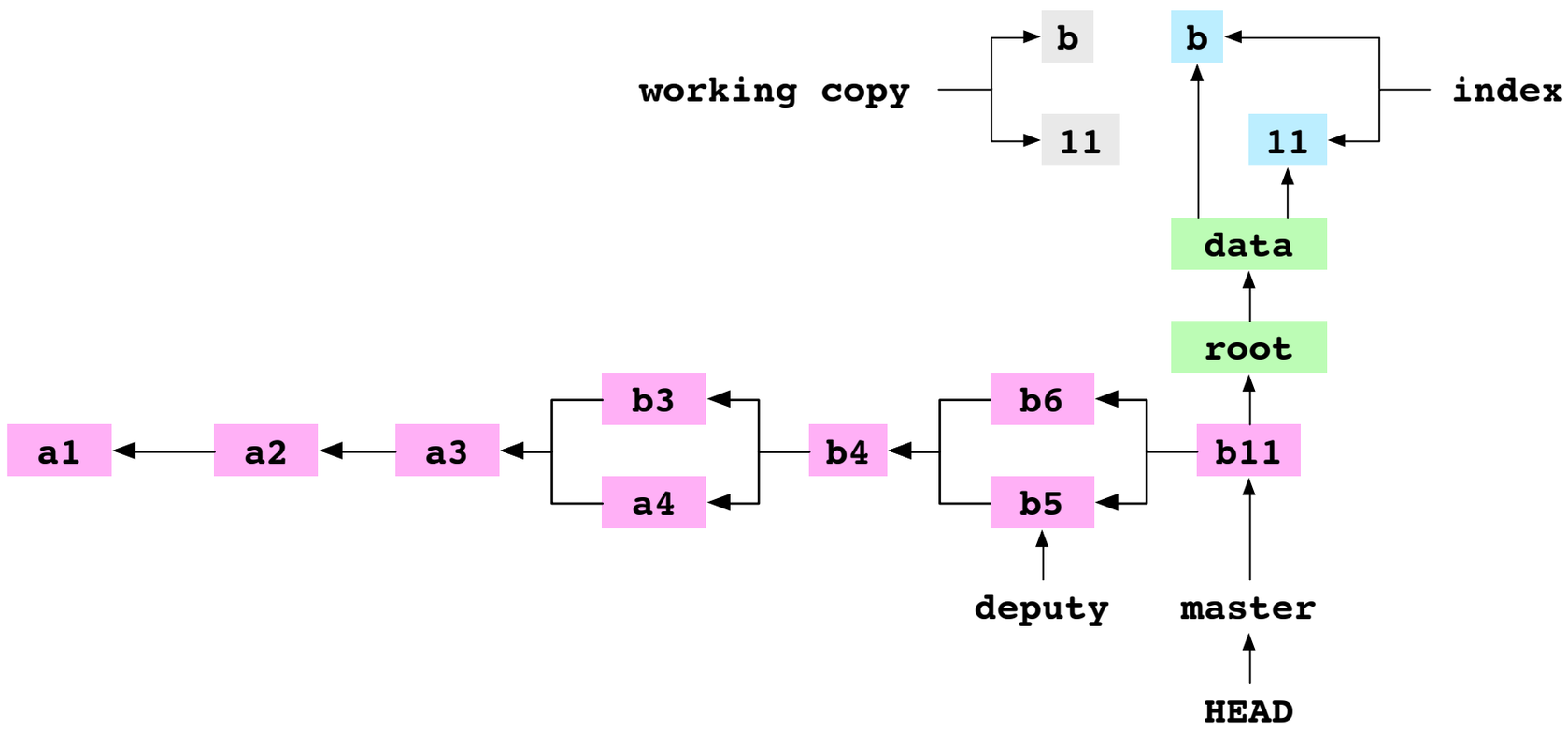
6. The user commits the merge

```
~/alpha $ git commit -m 'b11'
          master 251a
```



Remove a file

After the b11 commit

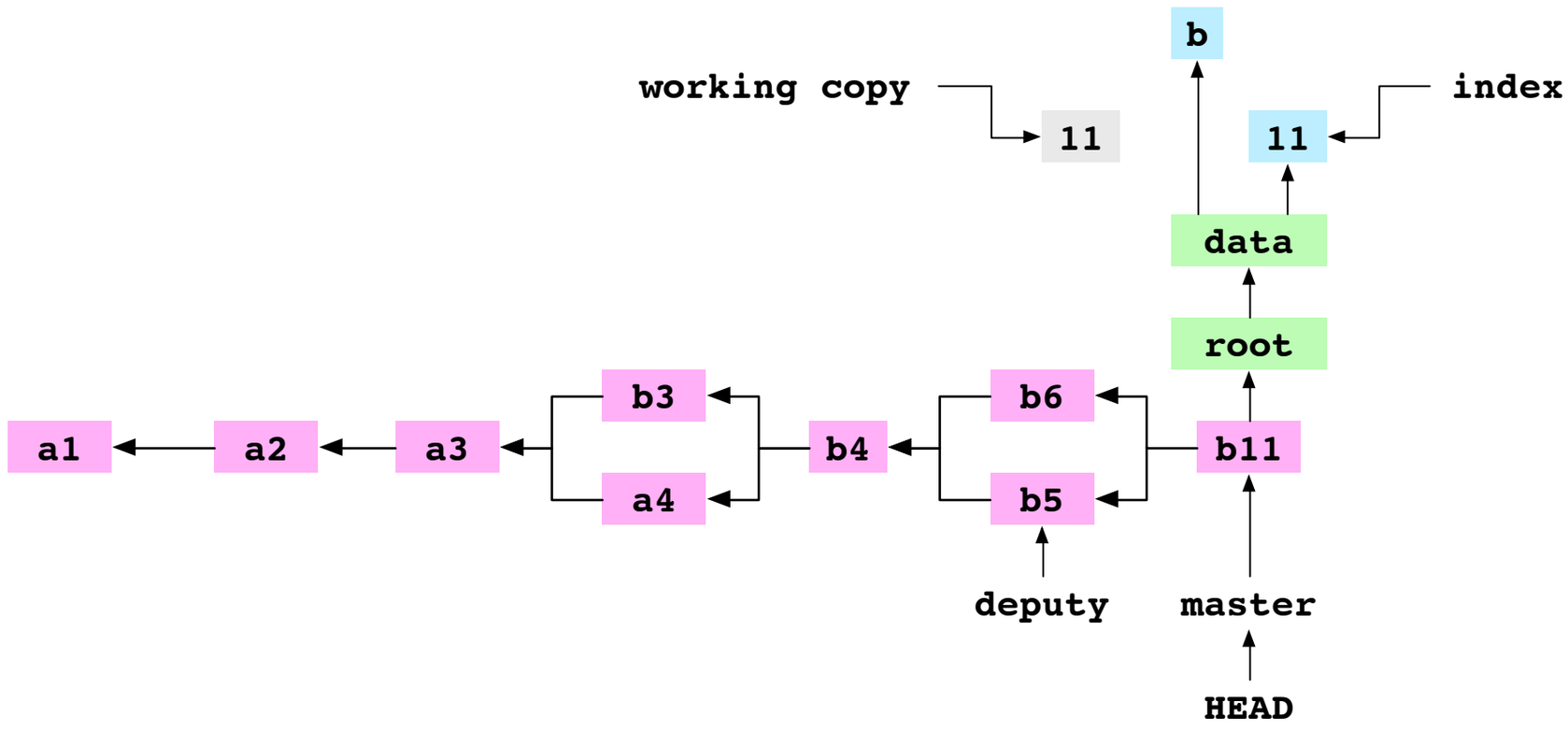


Remove `letter.txt`

```
~/alpha $ git rm data/letter.txt
          Removed data/letter.txt
```

Remove `letter.txt`

```
~/alpha $ git rm data/letter.txt
          Removed data/letter.txt
```

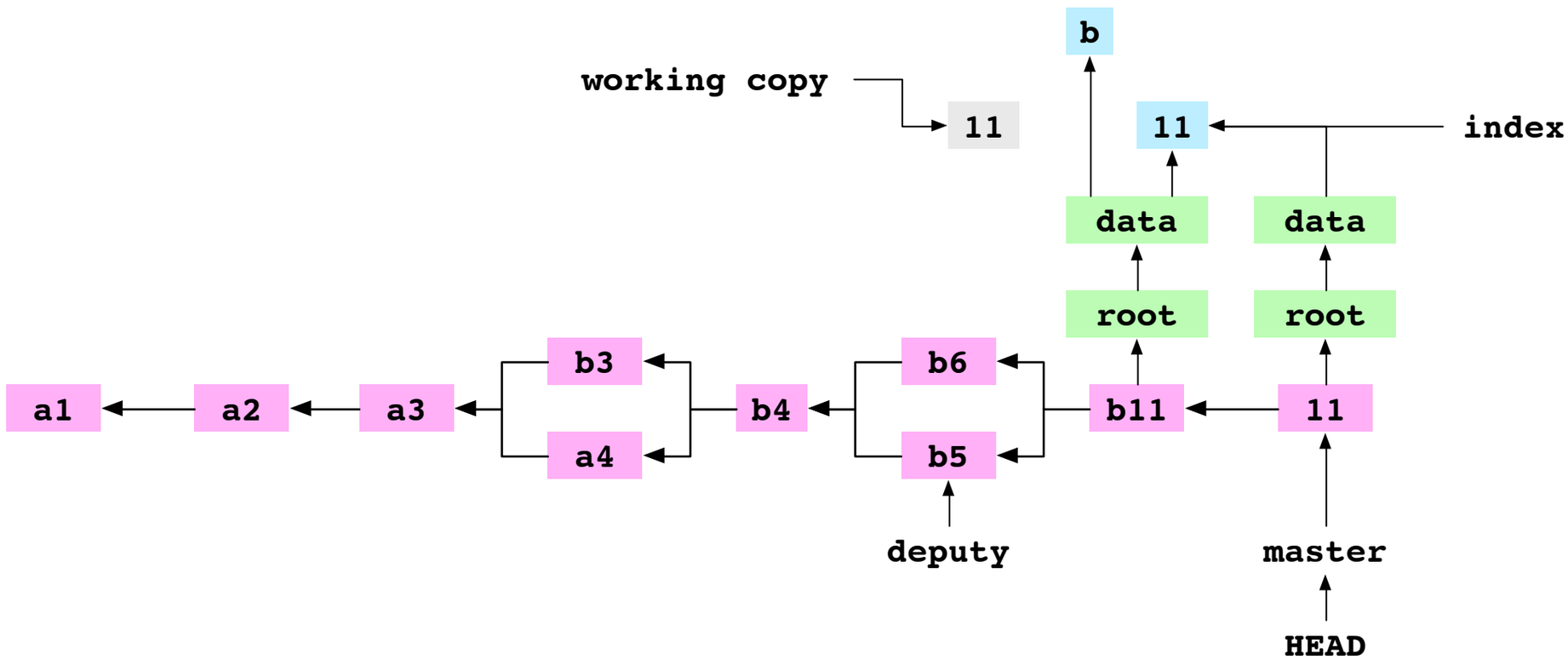


Commit the removal of `letter.txt`

```
~/alpha $ git commit -m '11'
          master d14c
```

Commit the removal of `data/letter.txt`

```
~/alpha $ git commit -m '11'
          master d14c
```



Copy a repository

Copy the `alpha` repository to the `bravo` directory

```
~/alpha $ cd ..
      ~ $ cp -R alpha bravo
```

Copy the `alpha` repository to the `bravo` directory

```
~/alpha $ cd ..
      ~ $ cp -R alpha bravo
      ~ $ tree -a
          ~
          ├── alpha
          │   ├── data
          │   │   └── number.txt
          │   └── .git
          │           etc...
          └── bravo
              ├── data
              │   └── number.txt
              └── .git
                      etc...
```

Copy the `alpha` repository to the `bravo` directory

```
~/alpha $ cd ..
      ~ $ cp -R alpha bravo
      ~ $ tree -a
          ~
          ├── alpha
          │   ├── data
          │   │   └── number.txt
          │   └── .git
          │           etc...
          └── bravo
              ├── data
              │   └── number.txt
              └── .git
                      etc...
```

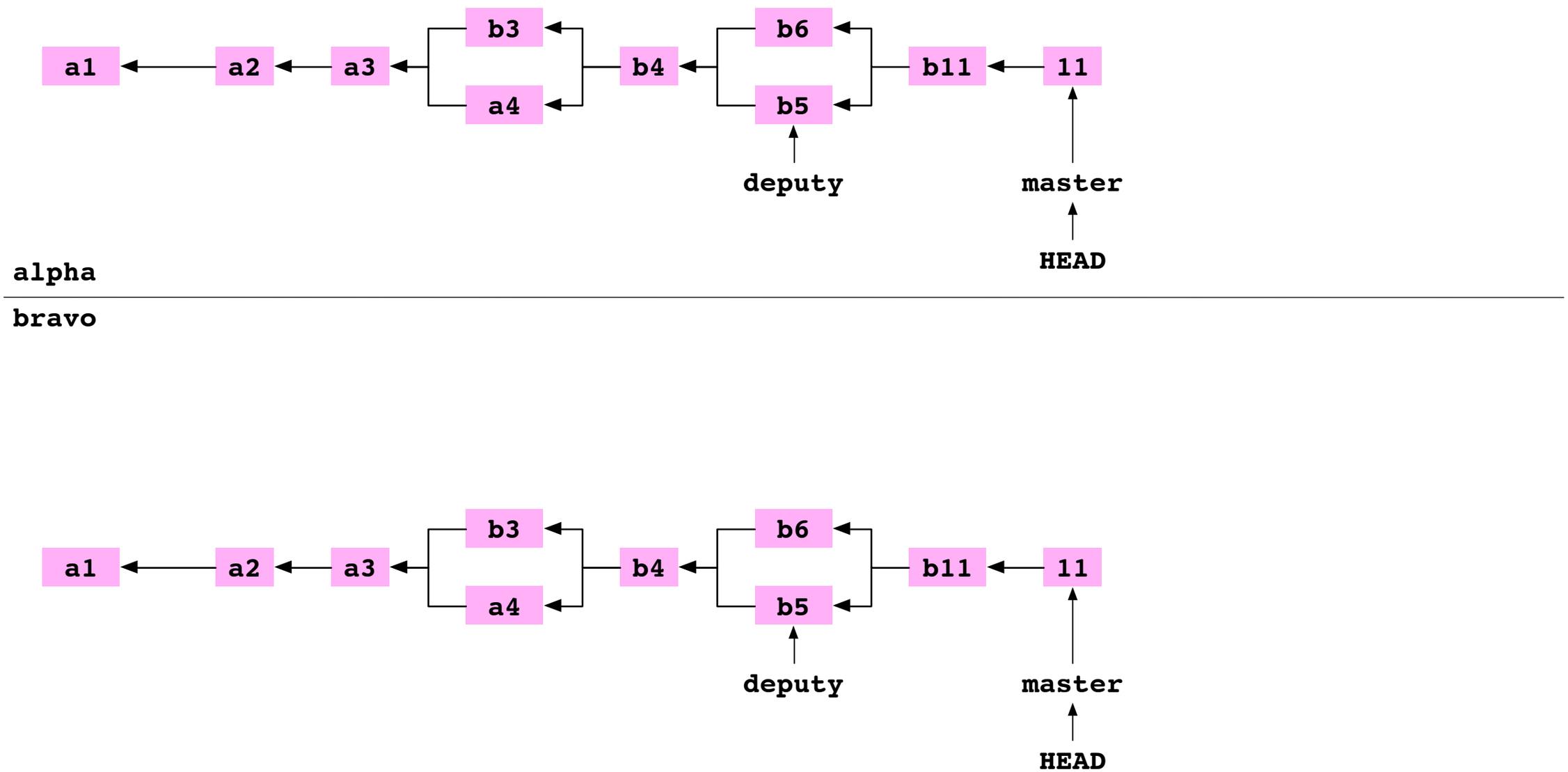
Copy the `alpha` repository to the `bravo` directory

```
~/alpha $ cd ..
      ~ $ cp -R alpha bravo
      ~ $ tree -a
          ~
          ├── alpha
          │   ├── data
          │   │   └── number.txt
          │   └── .git
          │           etc...
          └── bravo
              ├── data
              │   └── number.txt
              └── .git
                      etc...
```

Copy the `alpha` repository to the `bravo` directory

```
~/alpha $ cd ..
      ~ $ cp -R alpha bravo
      ~ $ tree -a
          ~
          ├── alpha
          │   ├── data
          │   │   └── number.txt
          │   └── .git
          │           etc...
          └── bravo
              ├── data
              │   └── number.txt
              └── .git
                      etc...
```

The `alpha` and `bravo` repositories



Connect a repository to another repository

Move to the `alpha` repository

```
      ~ $ cd alpha
~/alpha $
```

Set `bravo` as a remote repository on `alpha`

```
      ~ $ cd alpha
~/alpha $ git remote add bravo ../bravo
```

Set `bravo` as a remote repository on `alpha`

```
      ~ $ cd alpha
~/alpha $ git remote add bravo ../bravo
```

Set `bravo` as a remote repository on `alpha`

```
      ~ $ cd alpha
~/alpha $ git remote add bravo ../bravo
```

Set `bravo` as a remote repository on `alpha`

```
      ~ $ cd alpha
~/alpha $ git remote add bravo ../bravo
~/alpha $ cat .git/config
          remote bravo
            url = ../bravo
```

Fetch a branch from a remote repository

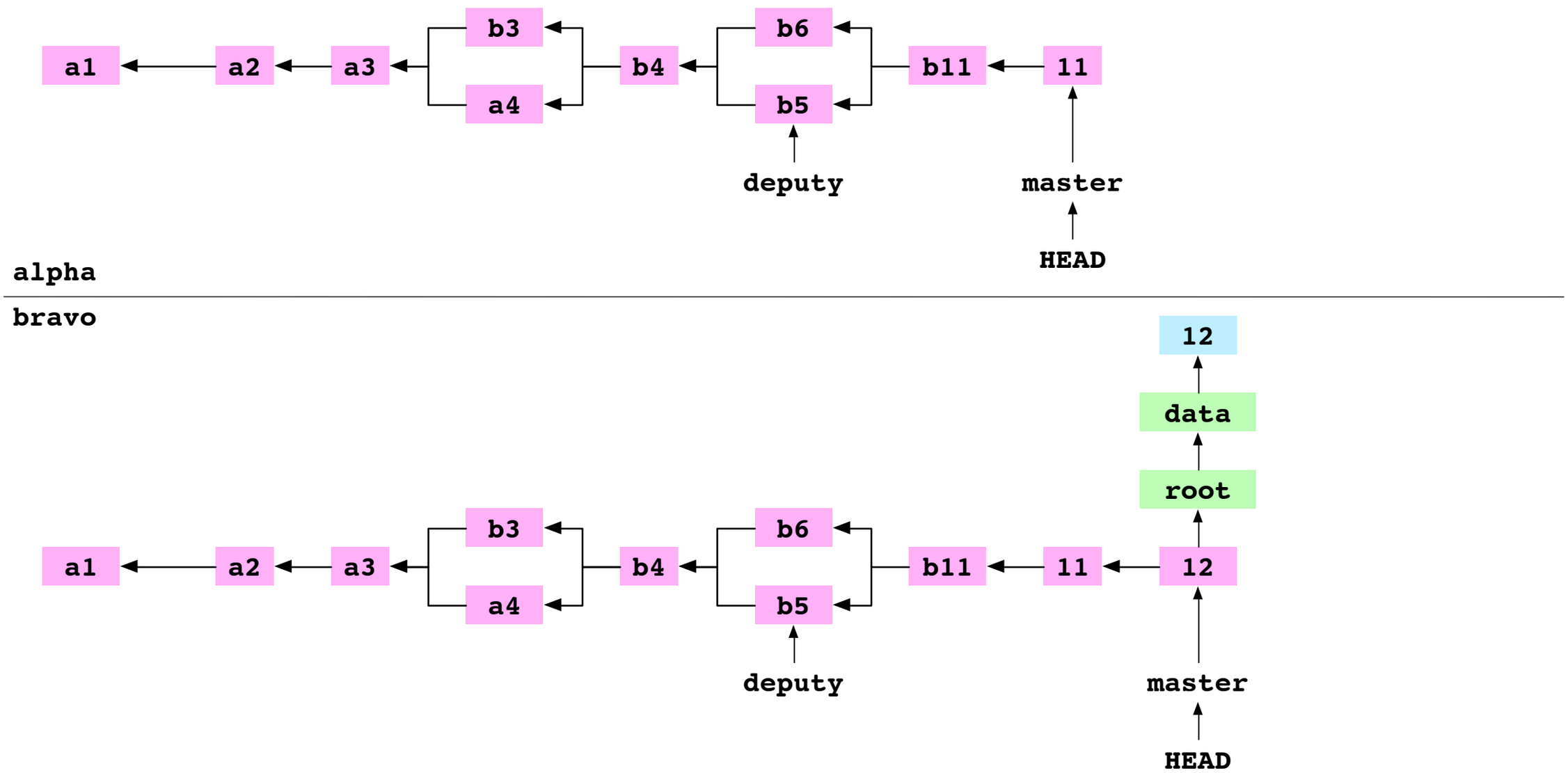
Move to the `bravo` repository

```
~/alpha $ cd ../bravo
~/bravo $
```

Set `number.txt` to '12' and commit

```
~/alpha $ cd ../bravo
~/bravo $ printf '12' > data/number.txt
~/bravo $ git add data/number.txt
~/bravo $ git commit -m '12'
          master 94cd
```

After the 12 commit made to `bravo`



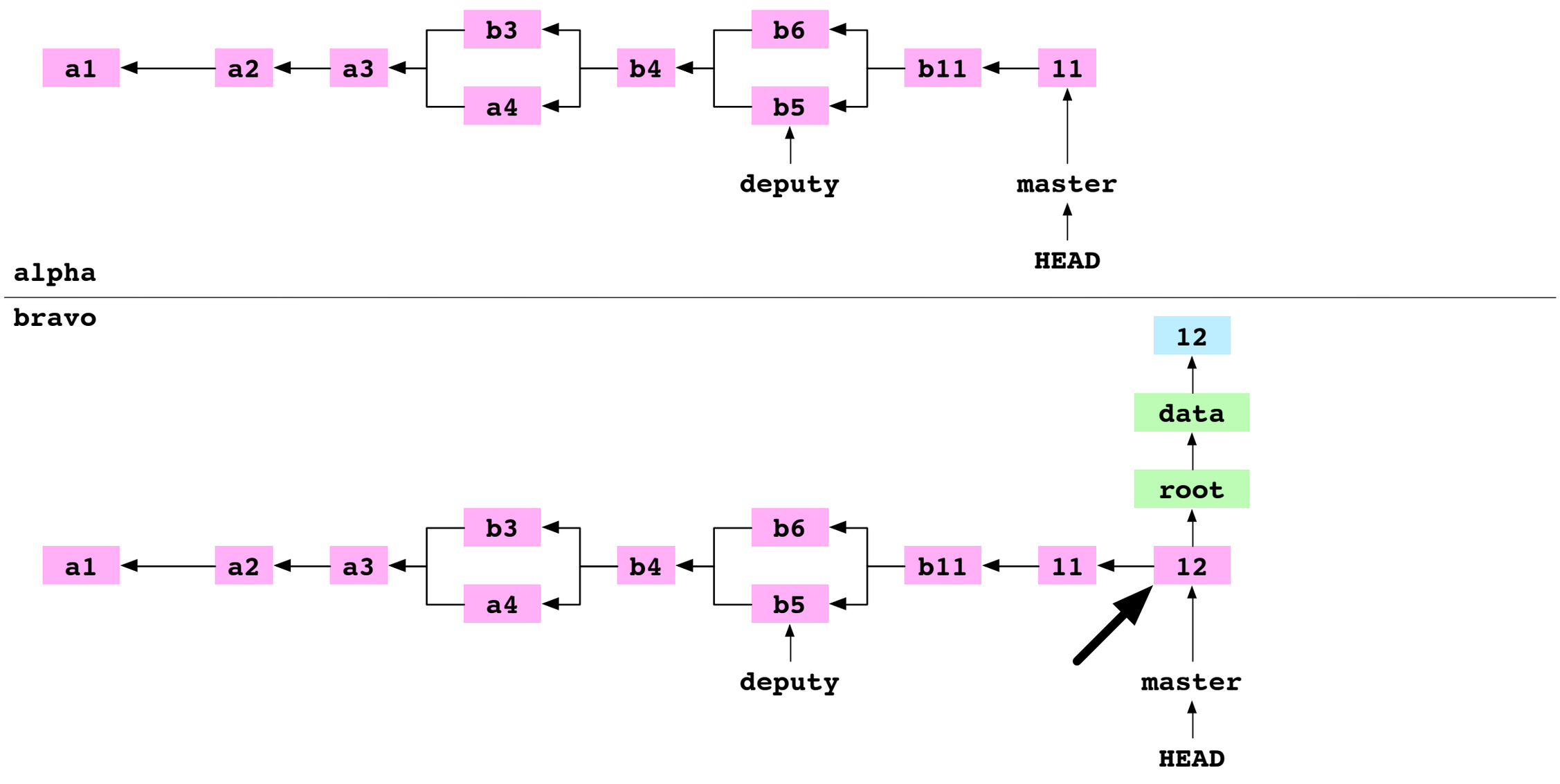
Move to the `alpha` repository

```
~/bravo $ cd ../alpha
~/alpha $
```

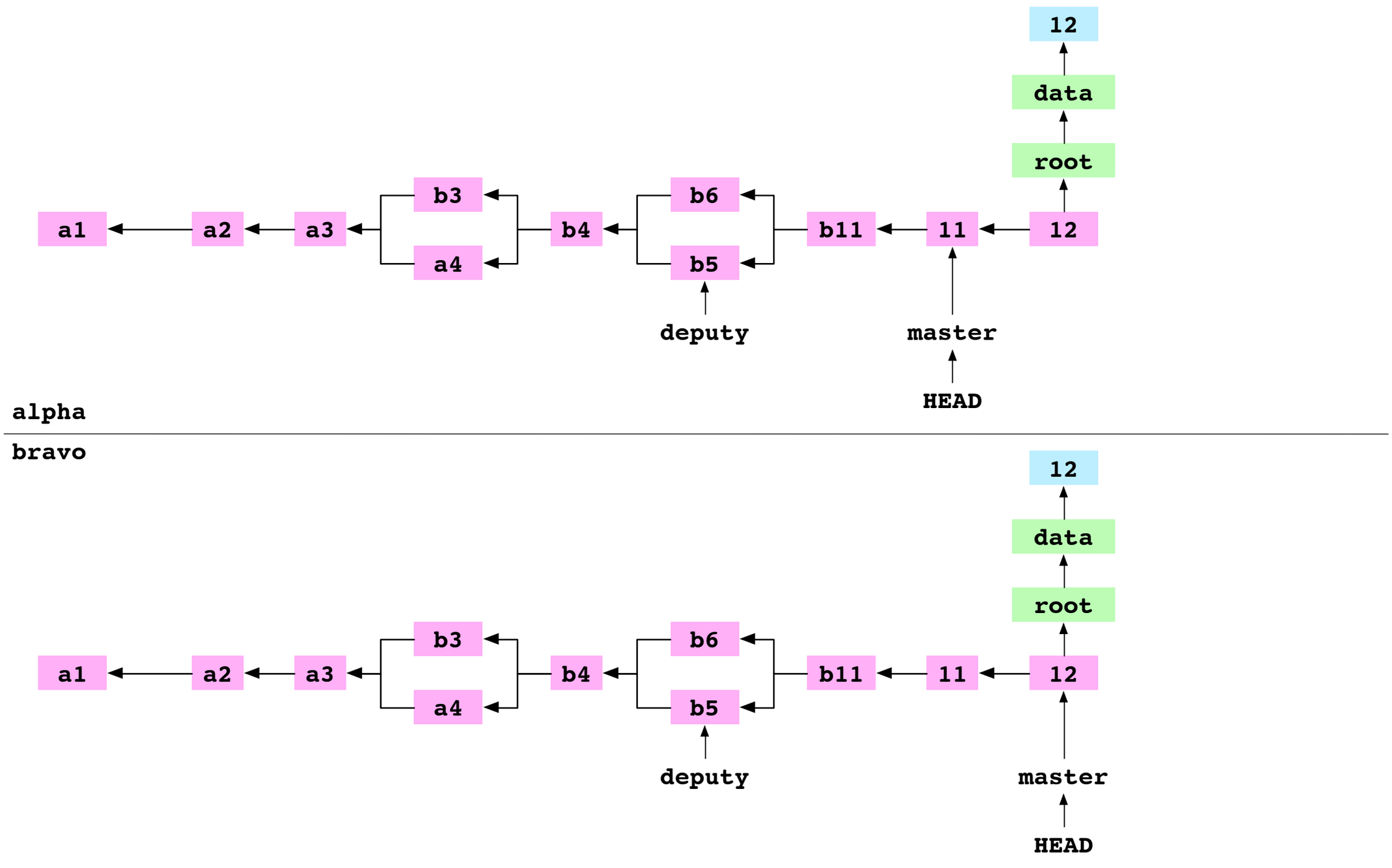
Fetch `master` from `bravo` into `alpha`

```
~/bravo $ cd ../alpha
~/alpha $ git fetch bravo master
          Fetching objects
          master -> FETCH_HEAD
```

1. Find the HEAD commit on the repository being fetched



2. Copy to the fetching repository the HEAD commit and its dependent objects



3. Point the ref for the remote branch at the fetched commit

```
~/alpha $ cat .git/refs/remotes/bravo/master
          94cd
```

3. Point the ref for the remote branch at the fetched commit

```
~/alpha $ cat .git/refs/remotes/bravo/master
          94cd
```

3. Point the ref for the remote branch at the fetched commit

```
~/alpha $ cat .git/refs/remotes/bravo/master
          94cd
```

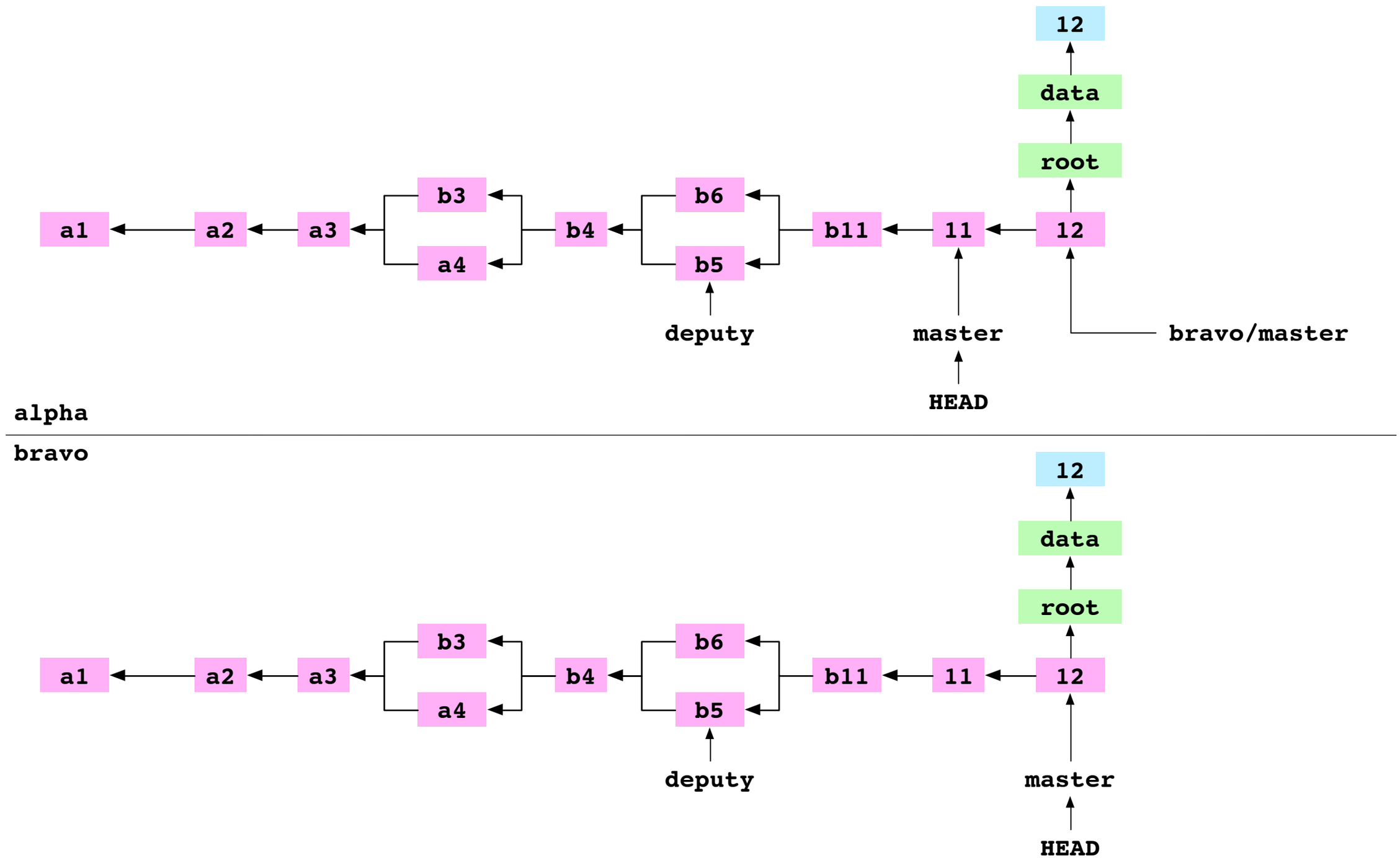
3. Point the ref for the remote branch at the fetched commit

```
~/alpha $ cat .git/refs/remotes/bravo/master
          94cd
```

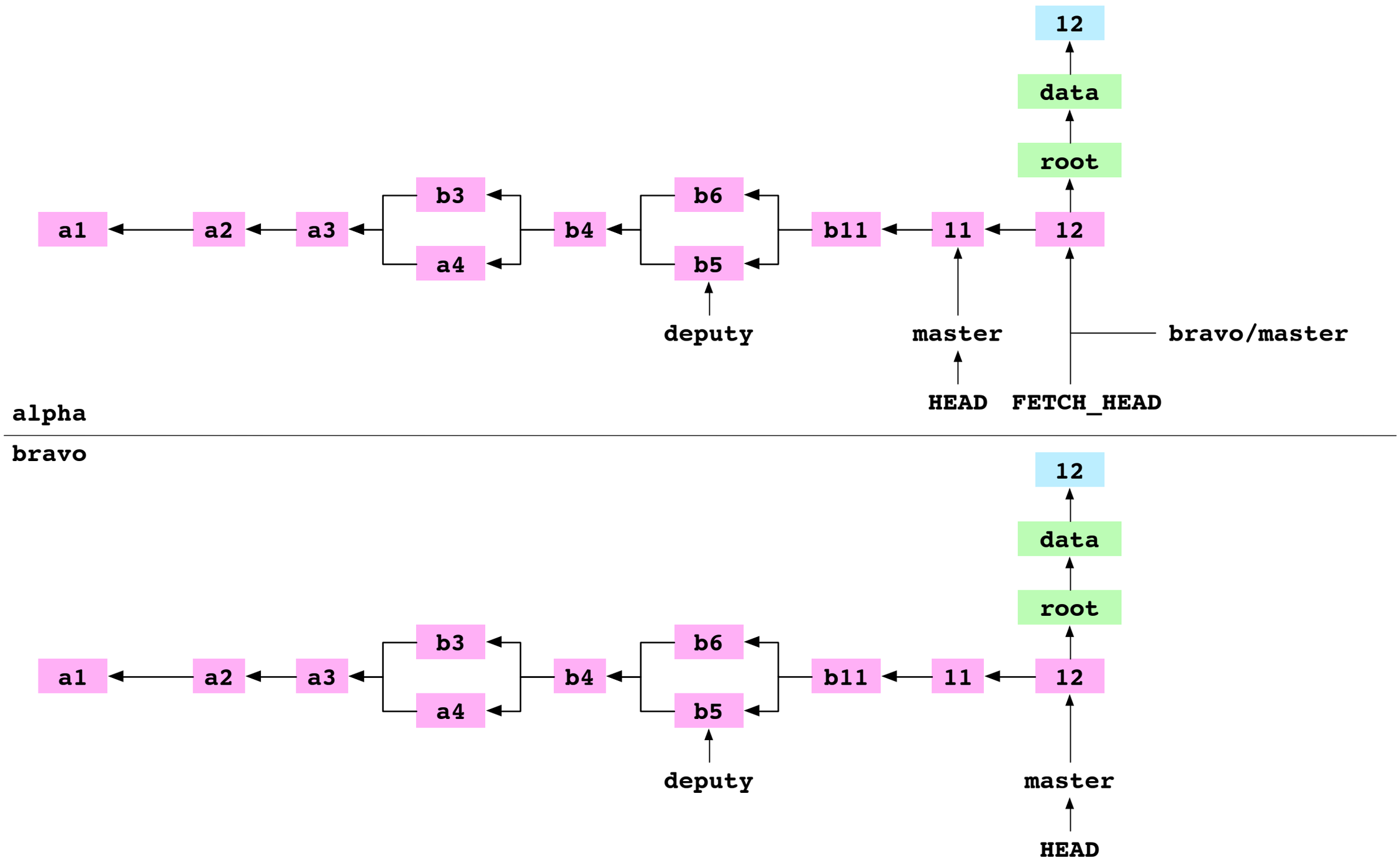
3. Point the ref for the remote branch at the fetched commit

```
~/alpha $ cat .git/refs/remotes/bravo/master
          94cd
```

3. Point the ref for the remote branch at the fetched commit

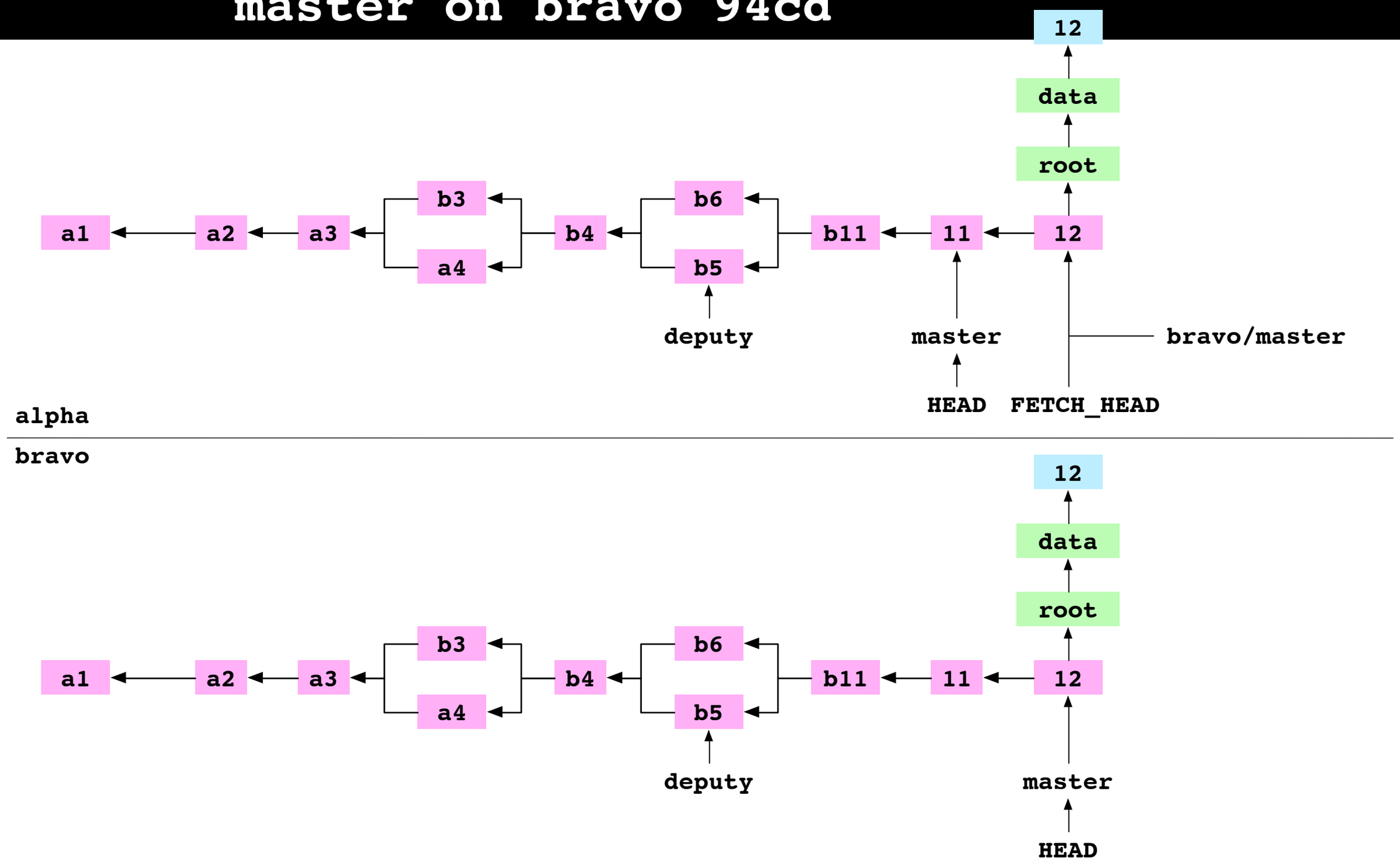


4. Point `FETCH_HEAD` at the fetched commit



4. Point FETCH_HEAD at the fetched commit

```
~/alpha $ cat .git/FETCH_HEAD
          master on bravo 94cd
```



Objects can be copied

Objects can be copied

History can be shared between repositories

Repositories store remote refs

Repositories store remote refs

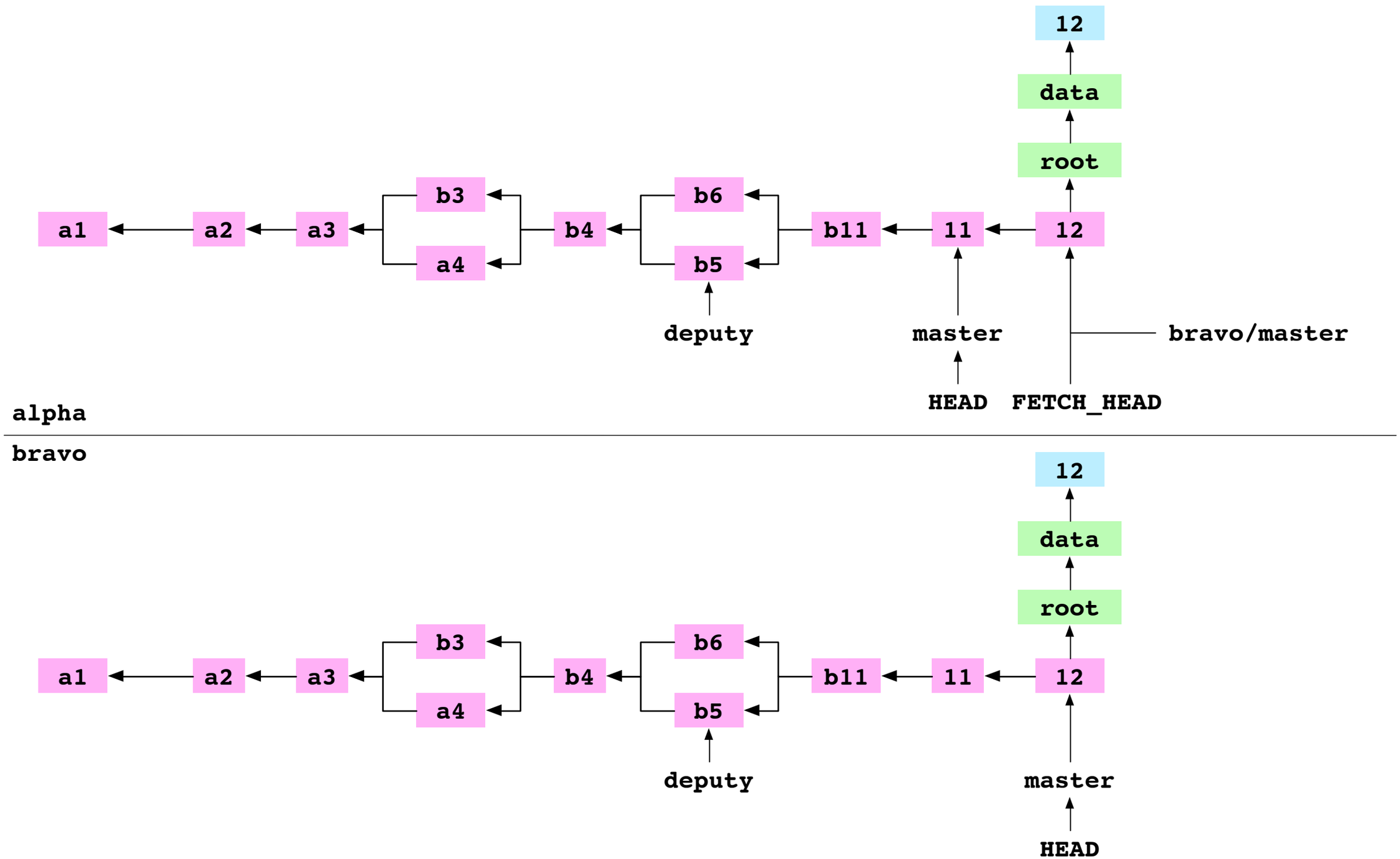
A repository can record locally the state of a branch on a remote repository

Merge `FETCH_HEAD`

Merge FETCH_HEAD

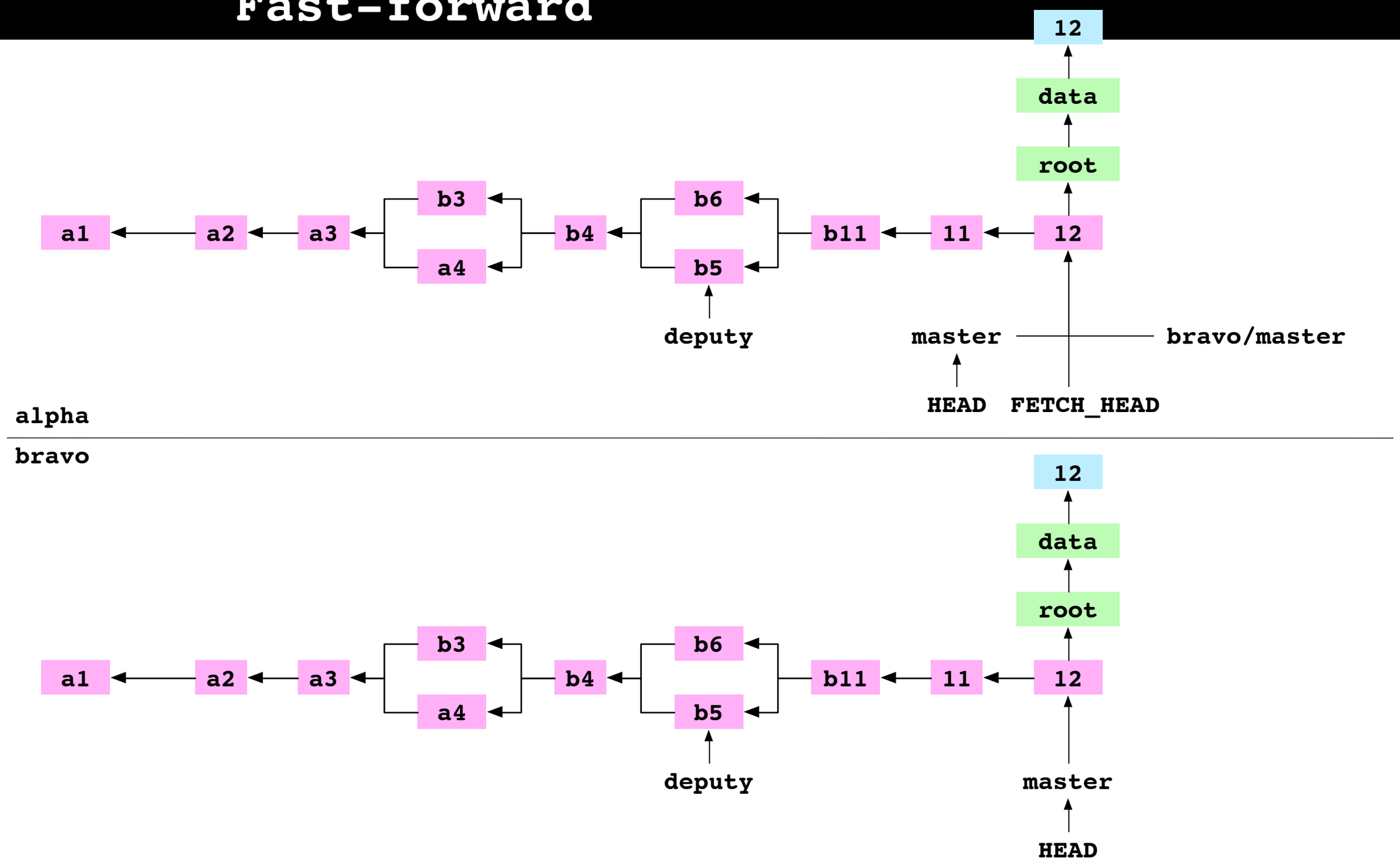
```
~/alpha $ git merge FETCH_HEAD
          Fast-forward
```

Before merging FETCH_HEAD



After merging FETCH_HEAD

```
~/alpha $ git merge FETCH_HEAD
          Fast-forward
```



Pull a branch from a remote

Pull `master` from `bravo` into `alpha`

```
~/alpha $ git pull bravo master
          Already up-to-date
```

Clone a repository

Clone `alpha` to `charlie`

```
~/alpha $ cd ..
      ~ $ git clone alpha charlie
          Cloned into charlie
```

1. Create the directory for the new repository

```
~/alpha $ cd ..
      ~ $ git clone alpha charlie
          Cloned into charlie
      ~ $ ls
          alpha
          bravo
          charlie
```

2. Move into the clone's directory

```
~/alpha $ cd ..
      ~ $ git clone alpha charlie
          Cloned into charlie
      ~ $ ls
          alpha
          bravo
          charlie
      ~ $ cd charlie
~/charlie $
```

3. Initialize the clone's directory as a Git repository

```
~/alpha $ cd ..
      ~ $ git clone alpha charlie
          Cloned into charlie
      ~ $ ls
          alpha
          bravo
          charlie
      ~ $ cd charlie
~/charlie $ tree .git
          └── objects
              etc...
```

4. Check out the branch that was checked out on the repository being cloned

```
  ~/alpha $ cd ..
        ~ $ git clone alpha charlie
            Cloned into charlie
- - - - - - - - - - - - - - - - - - - - - - - - - - -
~/charlie $ cat .git/HEAD
            ref:     refs/heads/master
```

5. Pull the branch that was checked out on the repository being cloned

```
  ~/alpha $ cd ..
        ~ $ git clone alpha charlie
            Cloned into charlie
- - - - - - - - - - - - - - - - - - - - - - - - - - -
~/charlie $ cat .git/HEAD
            ref:     refs/heads/master
~/charlie $ cat .git/refs/heads/master
            94cd
```

Move into `alpha`

```
~/charlie $ cd ../alpha
  ~/alpha $
```

Set `number.txt` to '13' and commit to `master`

```
~/charlie $ cd ../alpha
  ~/alpha $ printf '13' > data/number.txt
  ~/alpha $ git add data/number.txt
  ~/alpha $ git commit -m '13'
            master 3238
```

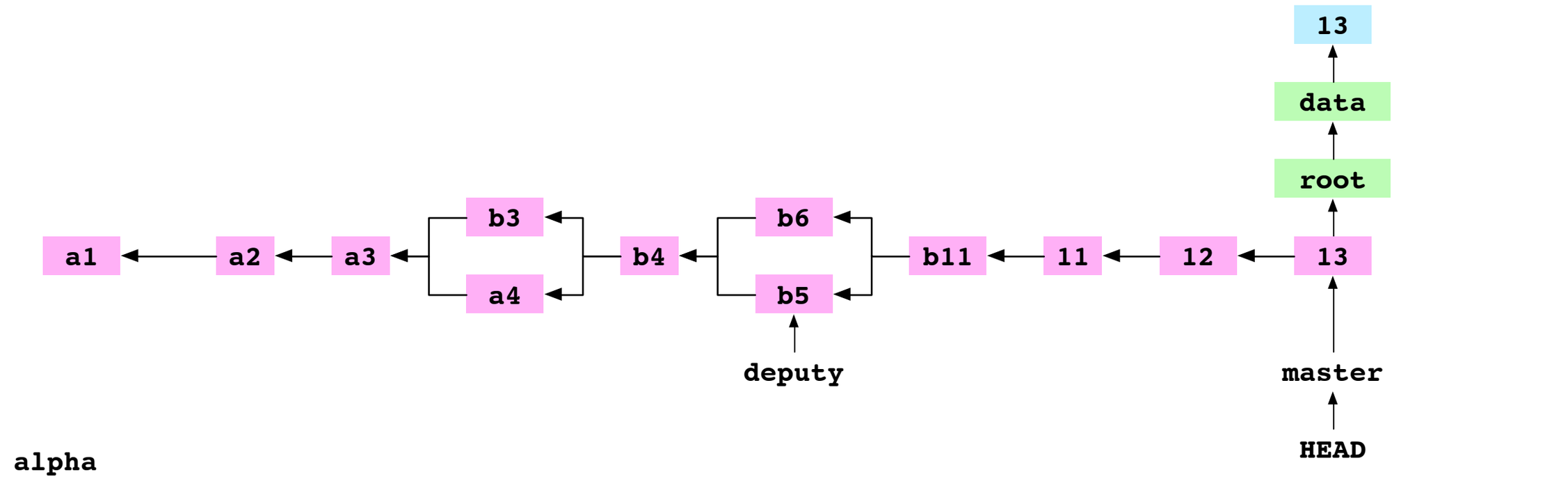
Set `charlie` as a remote repository on `alpha`

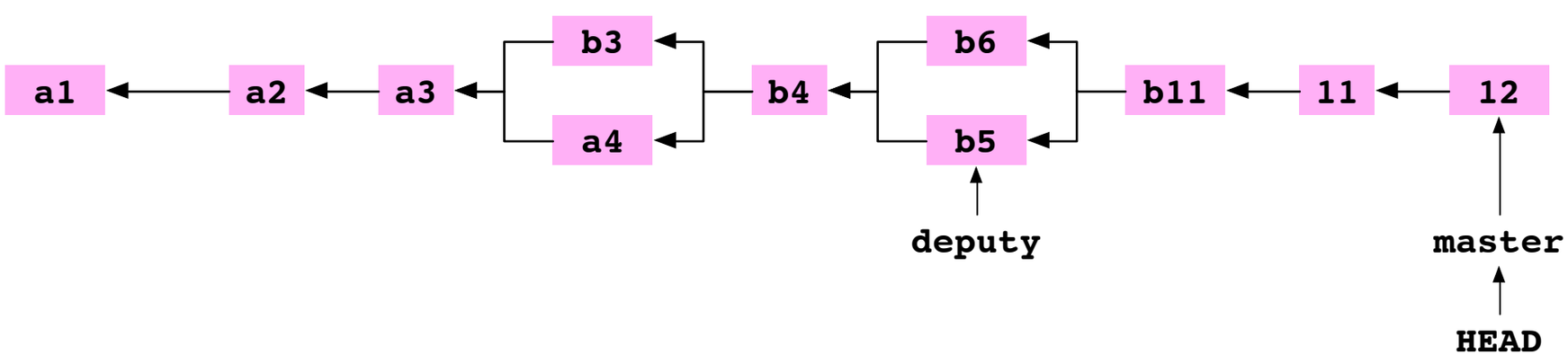
```
~/charlie $ cd ../alpha
  ~/alpha $ printf '13' > data/number.txt
  ~/alpha $ git add data/number.txt
  ~/alpha $ git commit -m '13'
            master 3238
  ~/alpha $ git remote add charlie ../charlie
```

Push `master` to `charlie`

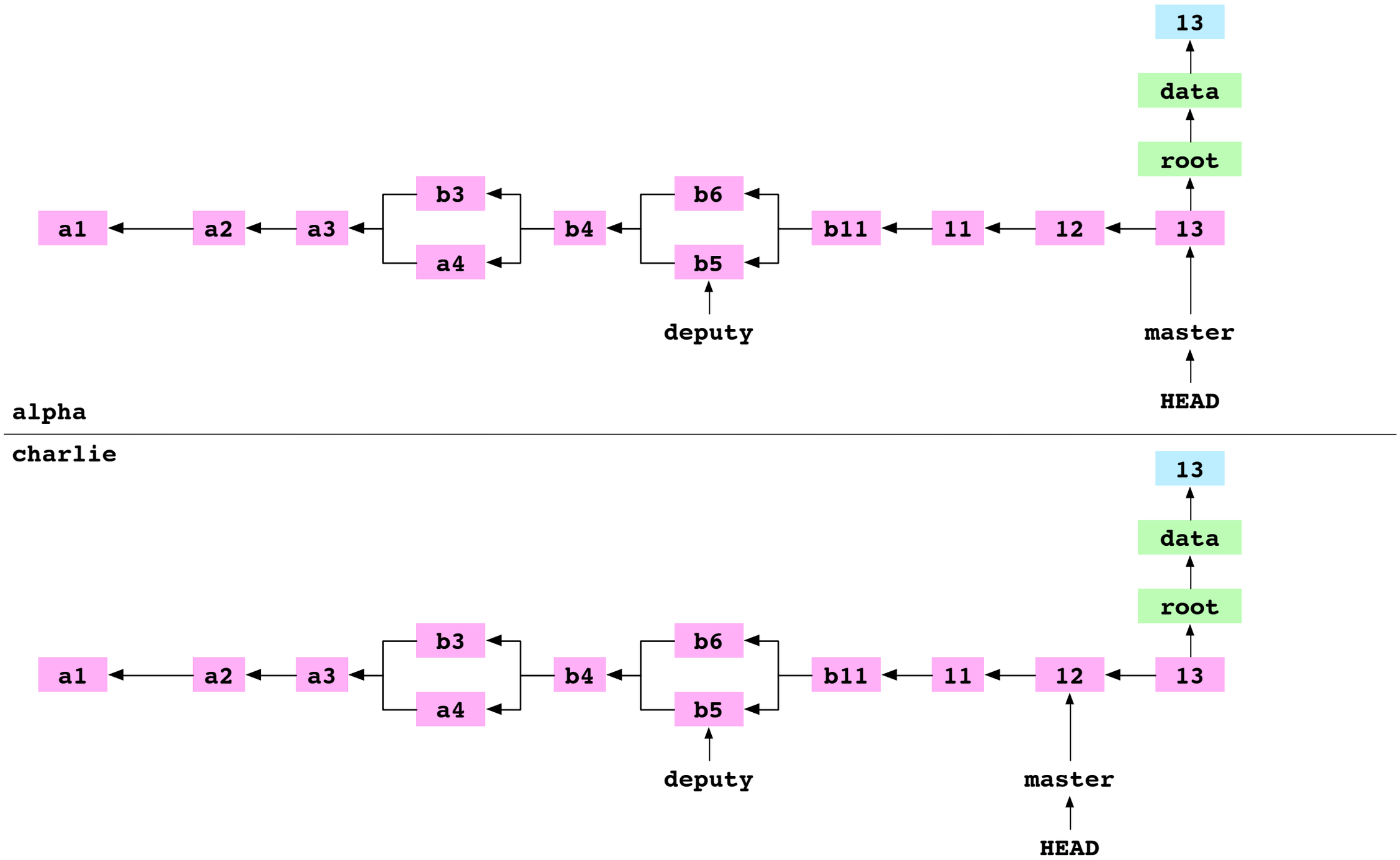
```
~/charlie $ cd ../alpha
  ~/alpha $ printf '13' > data/number.txt
  ~/alpha $ git add data/number.txt
  ~/alpha $ git commit -m '13'
            master 3238
  ~/alpha $ git remote add charlie ../charlie
  ~/alpha $ git push charlie master
            Writing objects.
            Refusing to update the checked out
            branch because it will make the index
            and working copy inconsistent
```

Before the push





After the push



Push `master` to `charlie`

```
~/charlie $ cd ../alpha
  ~/alpha $ printf '13' > data/number.txt
  ~/alpha $ git add data/number.txt
  ~/alpha $ git commit -m '13'
            master 3238
  ~/alpha $ git remote add charlie ../charlie
  ~/alpha $ git push charlie master
            Writing objects.
            Refusing to update the checked out
            branch because it will make the index
            and working copy inconsistent
```

Push `master` to `charlie`

```
~/charlie $ cd ../alpha
  ~/alpha $ printf '13' > data/number.txt
  ~/alpha $ git add data/number.txt
  ~/alpha $ git commit -m '13'
            master 3238
  ~/alpha $ git remote add charlie ../charlie
  ~/alpha $ git push charlie master
            Writing objects.
            Refusing to update the checked out
            branch because it will make the index
            and working copy inconsistent
```

Clone a bare repository

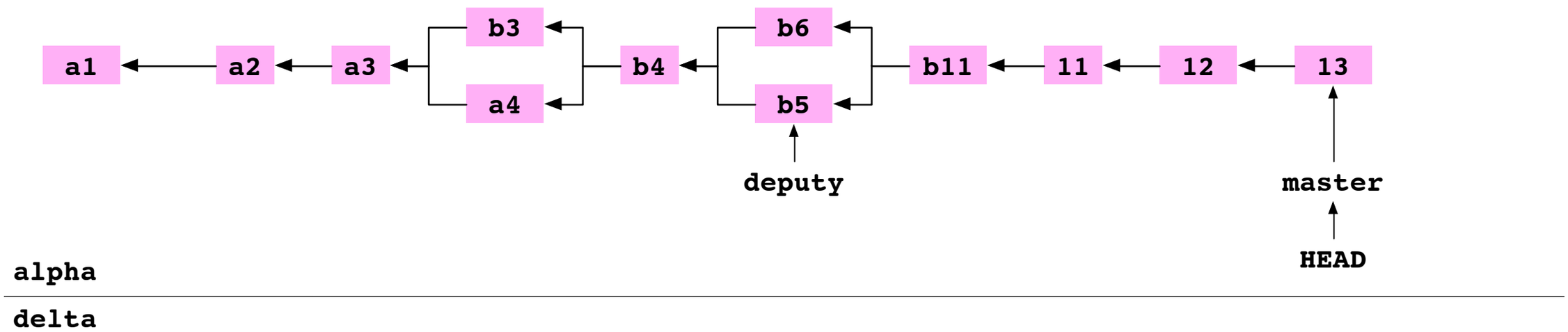
Clone `alpha` to bare repository `delta`

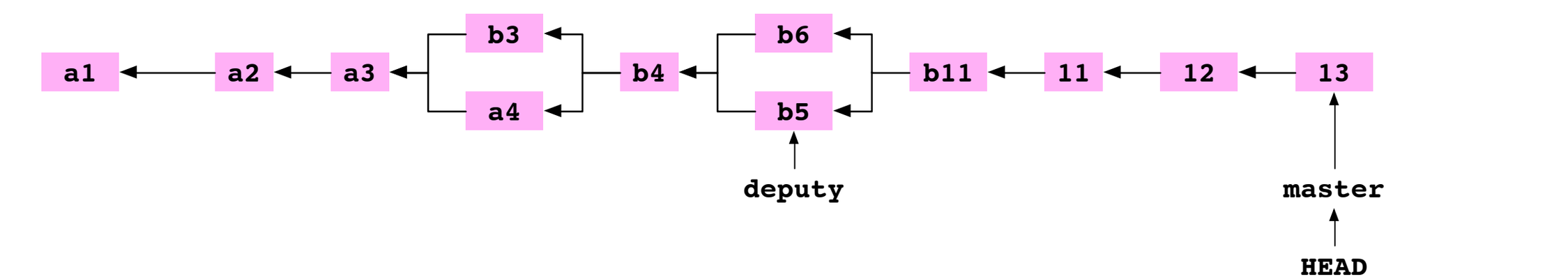
```
~/alpha $ cd ..
      ~ $ git clone alpha delta --bare
          Cloning into bare repository delta
```

File layout of `delta`

```
~/alpha $ cd ..
      ~ $ git clone alpha delta --bare
          Cloning into bare repository delta
      ~ $ tree delta
          delta
          ├── config
          └── objects
          etc...
```

The `alpha` and `delta` repositories





Move into `alpha`

```
      ~ $ cd alpha
~/alpha $
```

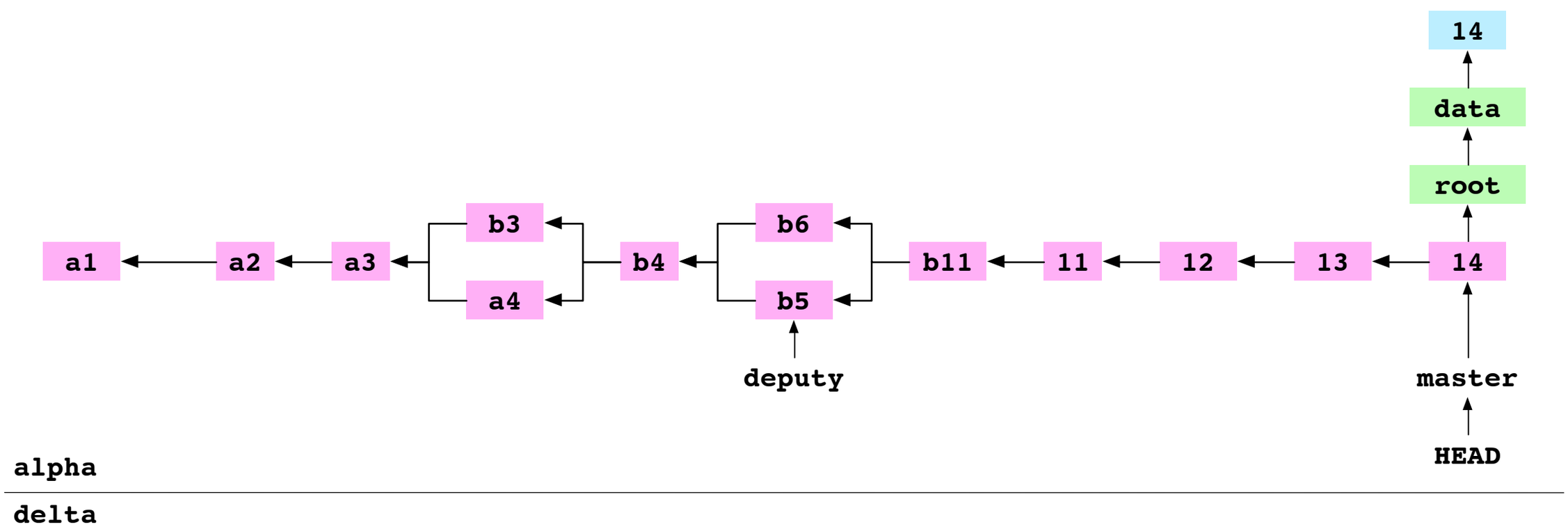
Set `delta` as a remote repository on `alpha`

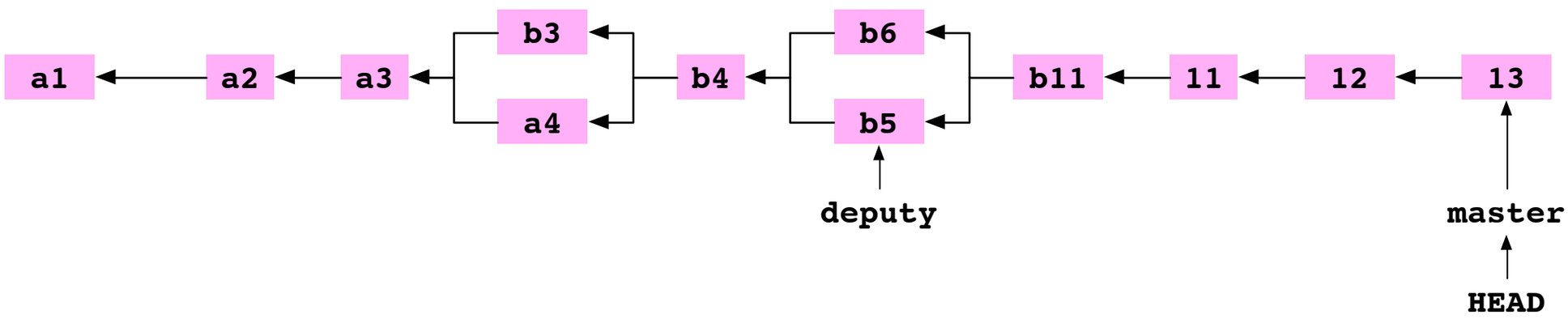
```
      ~ $ cd alpha
~/alpha $ git remote add delta ../delta
```

Set `number.txt` to '14' and commit

```
      ~ $ cd alpha
~/alpha $ git remote add delta ../delta
~/alpha $ printf '14' > data/number.txt
~/alpha $ git add data/number.txt
~/alpha $ git commit -m '14'
          master cb51
```

After the `14` commit made to `alpha`

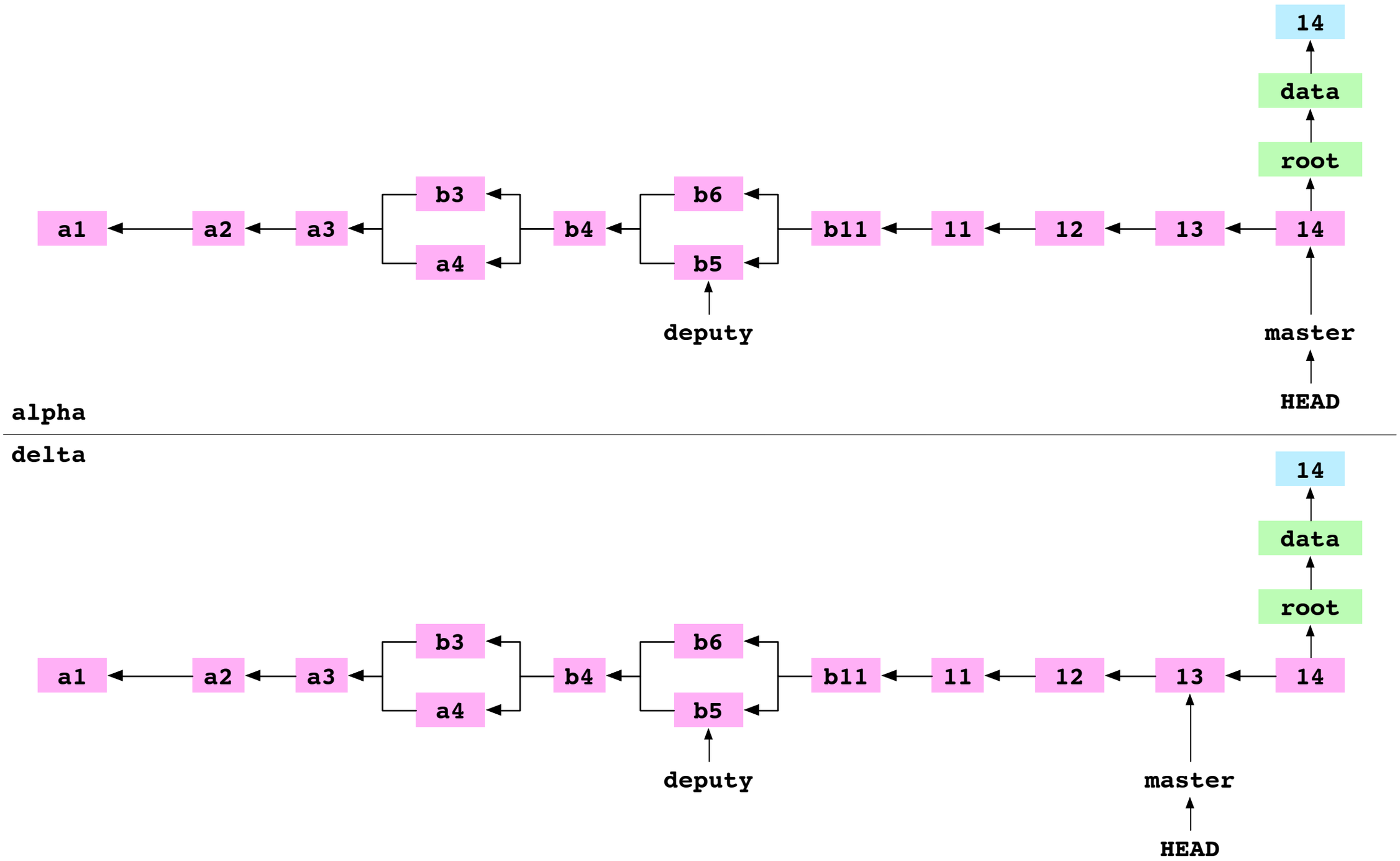




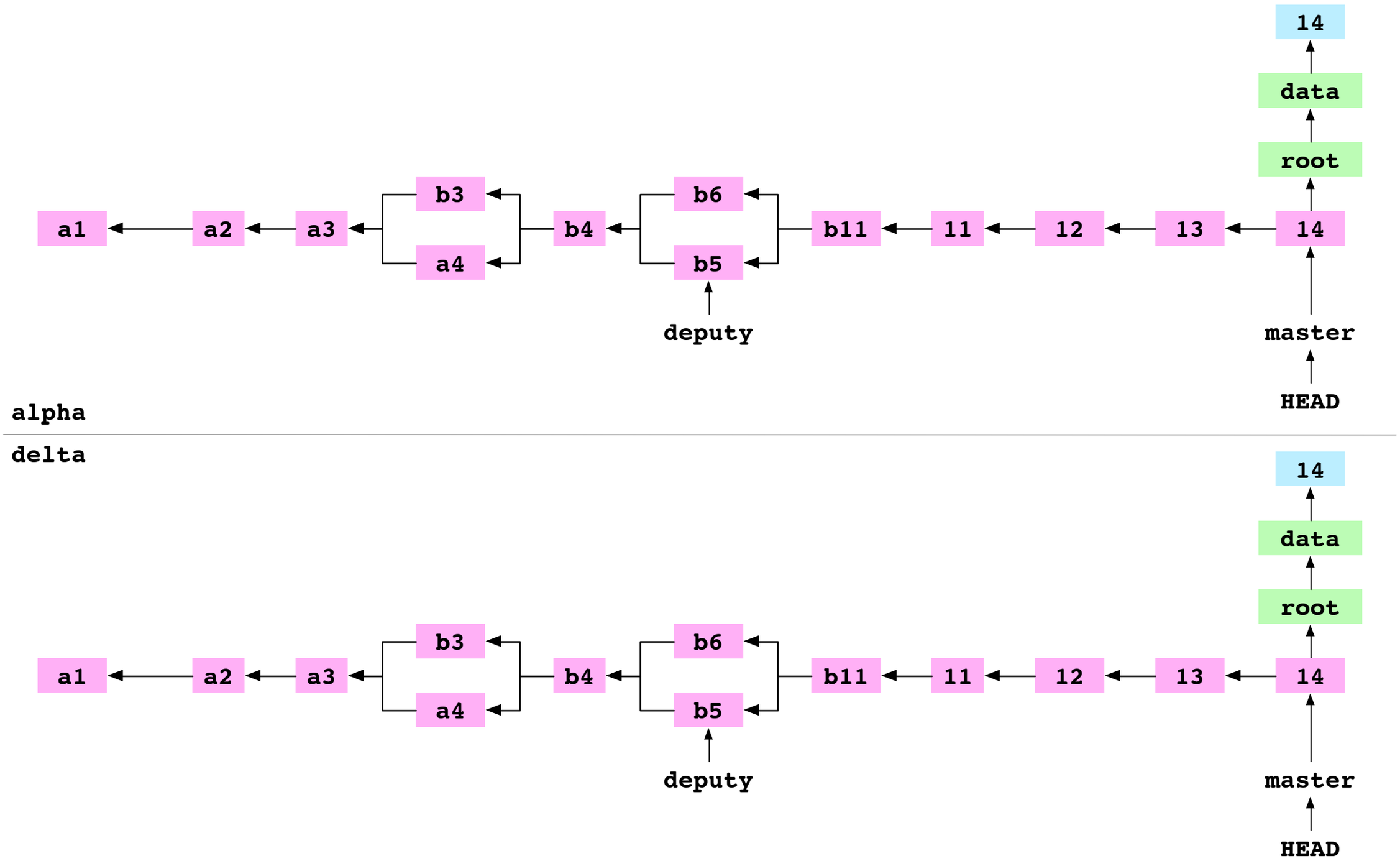
Push `master` to `delta`

```
~/alpha $ git push delta master
          Writing objects
          To ../delta
            3238..cb51 master
```

1. Copy the commit at HEAD and its dependent objects to the remote repository



2. Point HEAD on the remote to the pushed HEAD commit



Phew

Git is a graph

Git is a graph

This graph dictates Git's behaviour

Git is a graph

If you understand this graph,
you understand Git