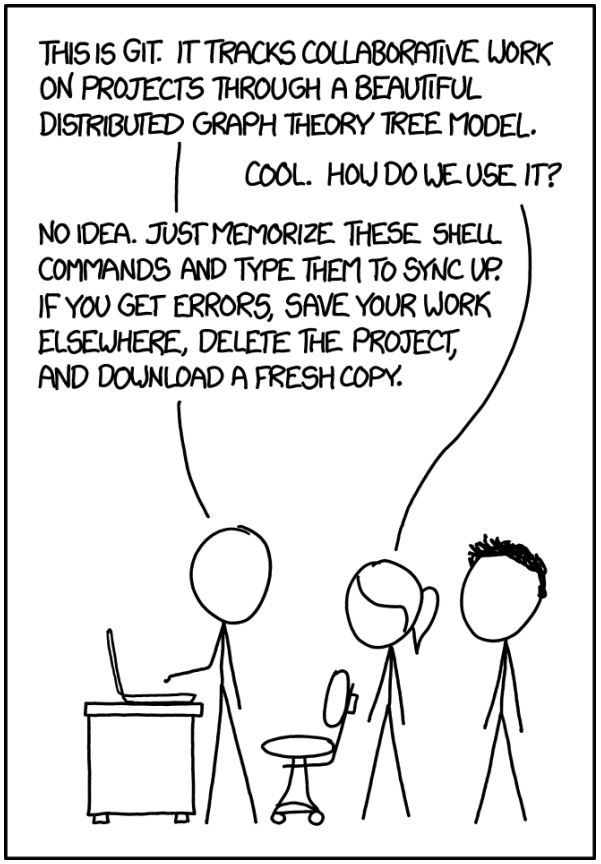
`https://xkcd.com/1597/` on Git



Start Work

```
$ mkdir r1 && cd r1
$ echo hi > greeting.txt
$ echo bye > parting.txt
$ tree -a
.
├── greeting.txt
└── parting.txt
```

Start Work

```
$ mkdir r1 && cd r1
$ echo hi > greeting.txt
$ echo bye > parting.txt
$ tree -a
.
├── greeting.txt
└── parting.txt
```

Like `ls -a`,
but prettier and
with subdirectories

Git Repository

A Git repository is "just" a directory

```
$ git init && git checkout -b main
Initialized empty Git repository ...
$ tree -a
.
├── .git
│   ├── ...
│   └── ...
├── greeting.txt
└── parting.txt
```

Git Repository

A Git repository is "just" a directory

```
$ git init && git checkout -b main
Initialized empty Git repository ...
$ tree -a
.
├── .git
│   ├── ...
│   └── ...
├── greeting.txt
└── parting.txt
```

repository

Git Repository

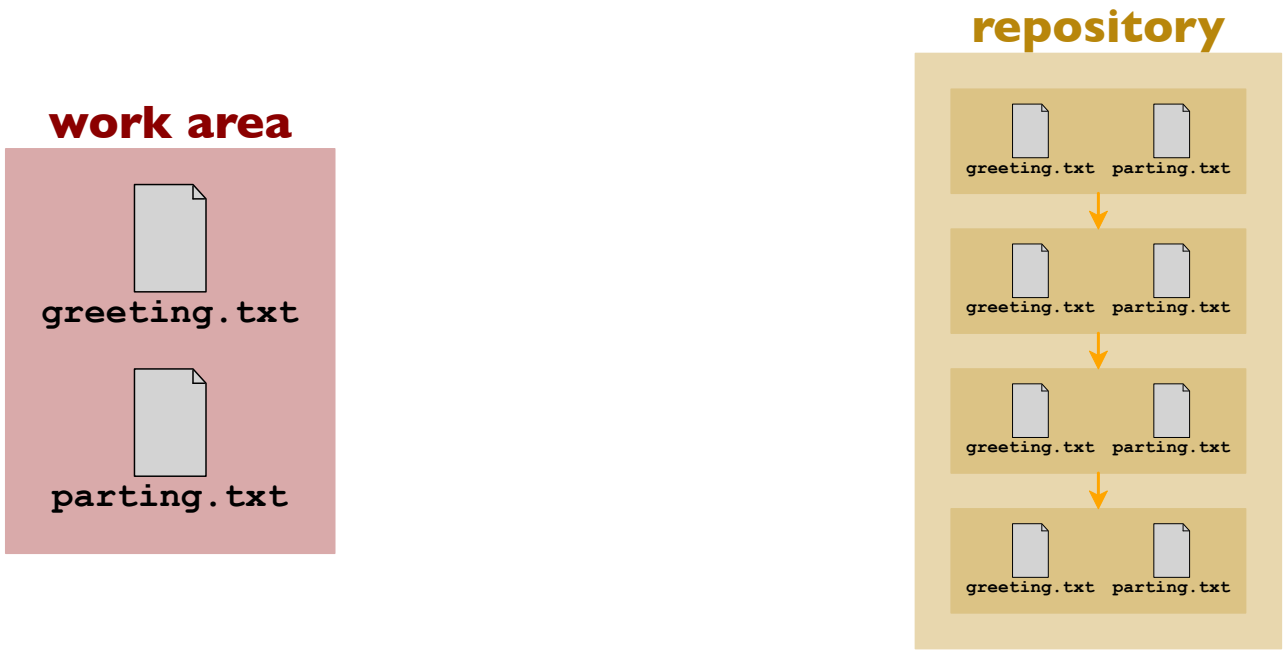
A Git repository is "just" a directory

```
$ git init && git checkout -b main
Initialized empty Git repository ...
$ tree -a
.
├── .git
│   ├── ...
│   └── ...
├── greeting.txt
└── parting.txt
```

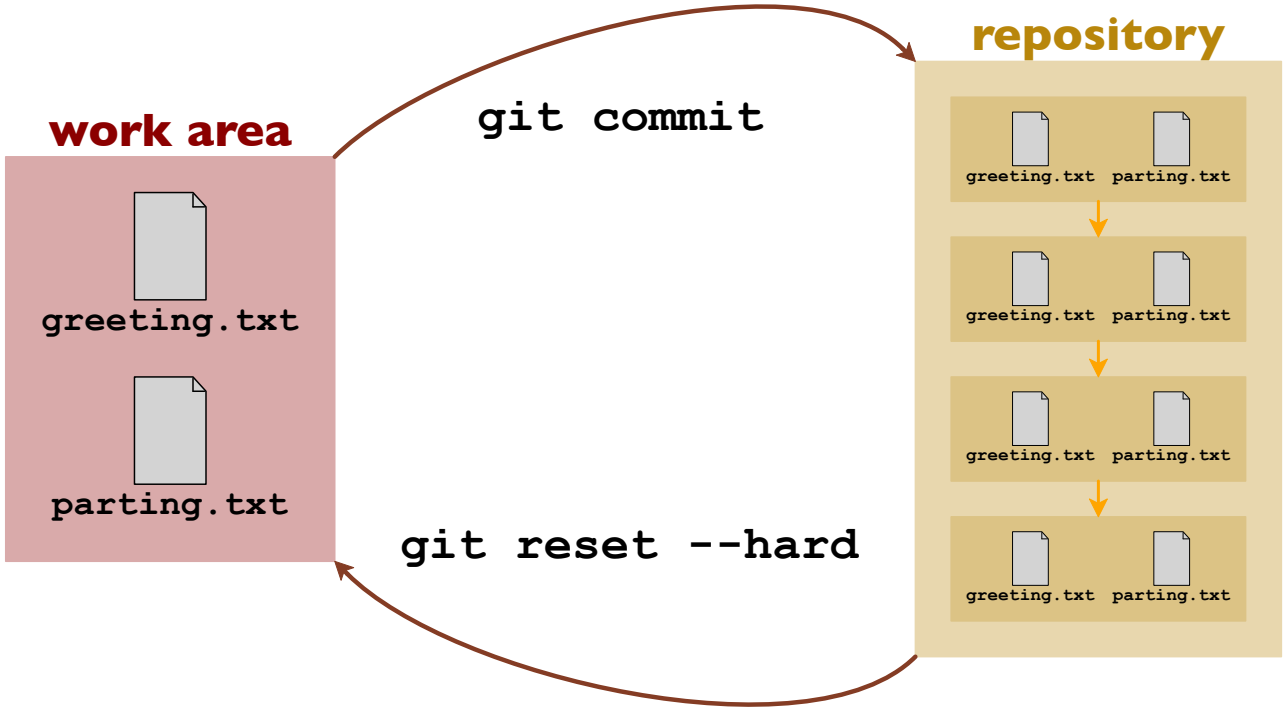
repository

work area

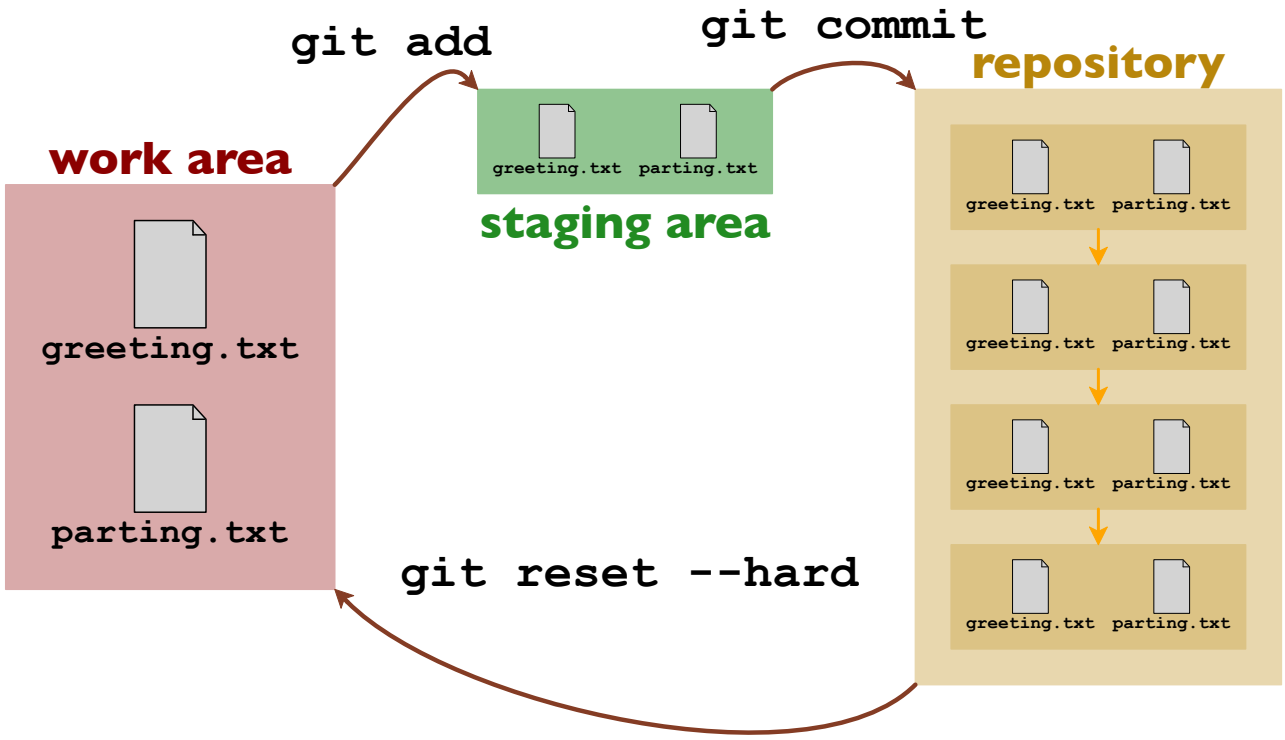
Git Workflow



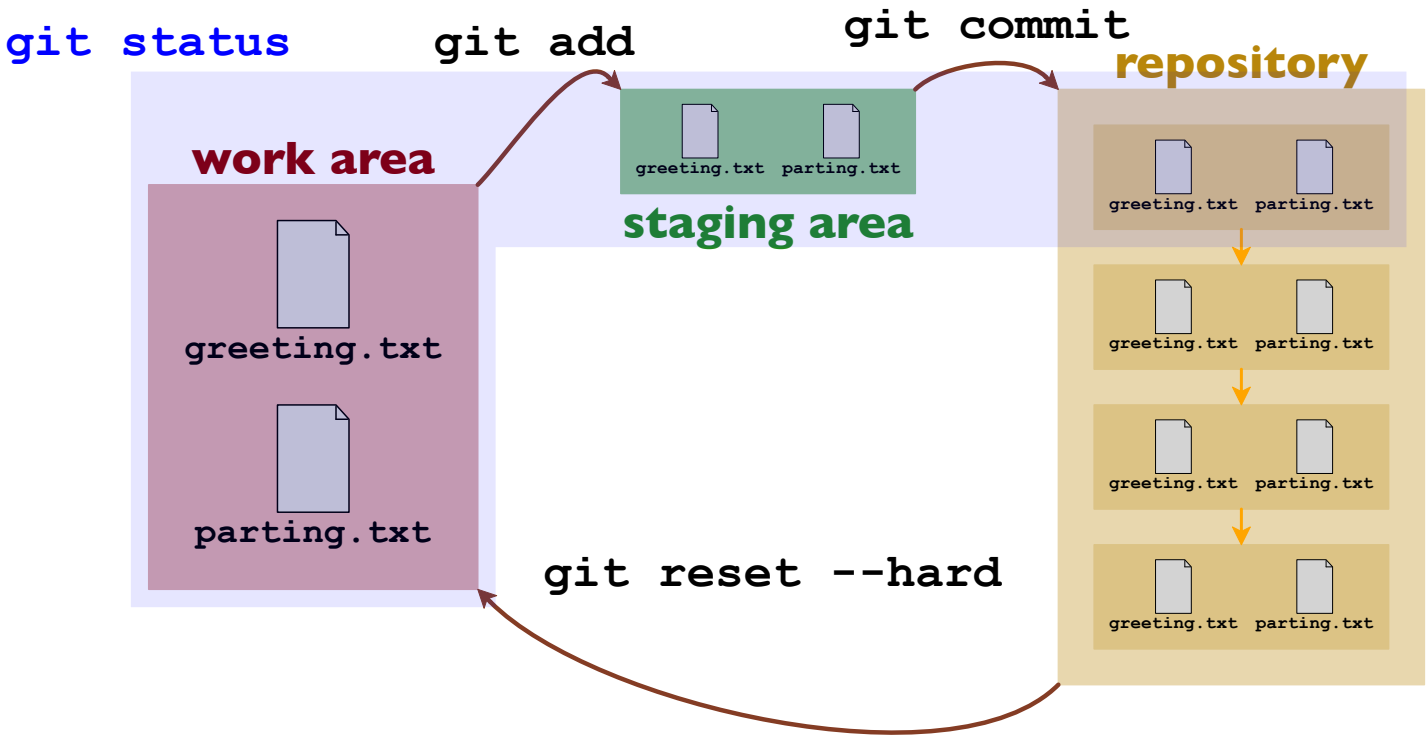
Git Workflow



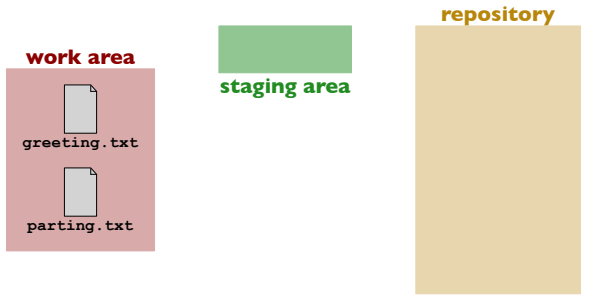
Git Workflow



Git Workflow

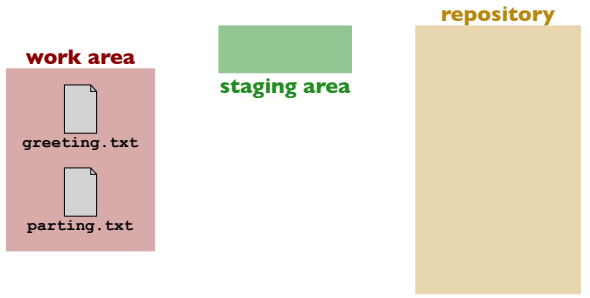


Checking the Status




```
$ git status
On branch main

No commits yet

Untracked files:
  greeting.txt
  parting.txt
```

Checking the Status



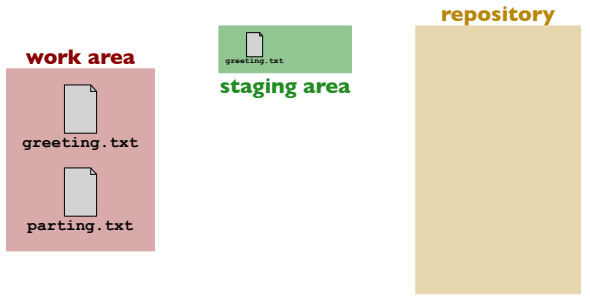

```
$ git status
On branch main

No commits yet

Untracked files:
  greeting.txt
  parting.txt
```

commit in repository

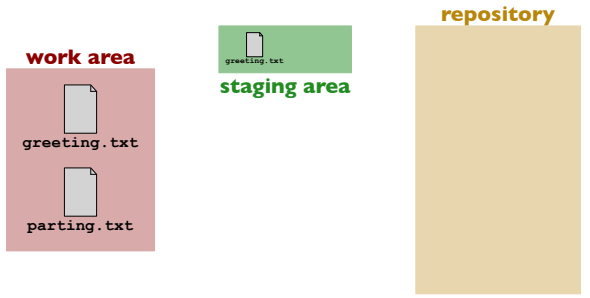
workarea difference from **staging area**

Checking the Status

```
$ git add greeting.txt

$ git status
On branch main

No commits yet

Changes to be committed:
  new file:   greeting.txt

Untracked files:
  parting.txt
```



Checking the Status



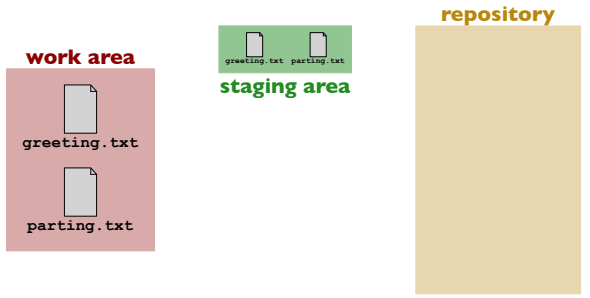

```
$ git add greeting.txt

$ git status
On branch main
No commits yet

Changes to be committed:
  new file:   greeting.txt

Untracked files:
  parting.txt
```

commit in repository

staging area difference from **commit**

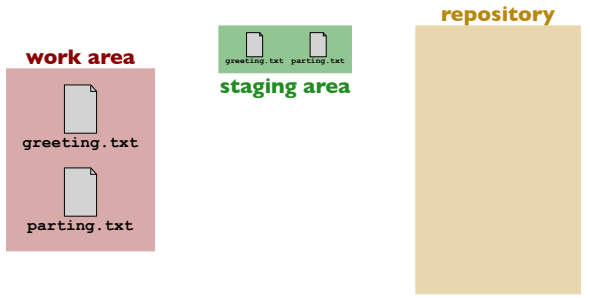
workarea difference from **staging area**

Checking the Status




```
$ git add parting.txt

$ git status
On branch main

No commits yet

Changes to be committed:
  new file:   greeting.txt
  new file:   parting.txt
```

Checking the Status

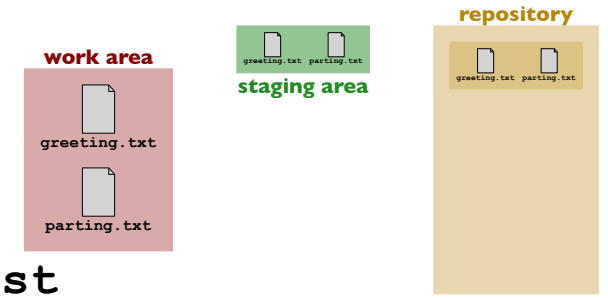



```
$ git add parting.txt

$ git status
On branch main

No commits yet

Changes to be committed:
  new file:   greeting.txt
  new file:   parting.txt
```

commit in repository

staging area difference from **commit**

Checking the Status



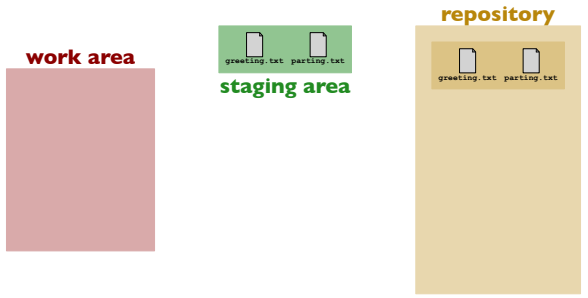

```
$ git commit -m first
[main (root-commit) 04be3f4] first
2 files changed, 2 insertions(+)
create mode 100644 greeting.txt
create mode 100644 parting.txt

$ git status
On branch main
nothing to commit, working tree clean
```

Checking the Status


```
$ git commit -m first
[main (root-commit) 04be3f4] first
2 files changed, 2 insertions(+)
create mode 100644 greeting.txt
create mode 100644 parting.txt

$ git status
On branch main
nothing to commit, working tree clean
```

commit in repository

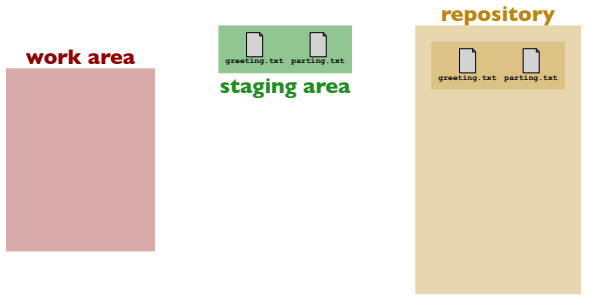
Checking the Status



A Git repository is "just" a directory

```
$ mkdir ../r2 && cd ../r2
$ cp -r ../r1/.git .git

$ git status
On branch main
Changes not staged for commit:
 deleted:    greeting.txt
 deleted:    parting.txt
```

Checking the Status



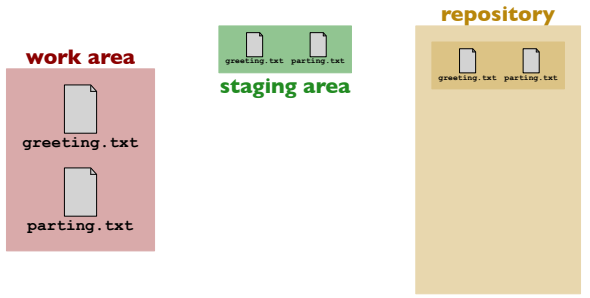
A Git repository is "just" a directory

```
$ mkdir ../r2 && cd ../r2
$ cp -r ../r1/.git .git

$ git status
On branch main
Changes not staged for commit:
  deleted:    greeting.txt
  deleted:    parting.txt
```

commit in repository

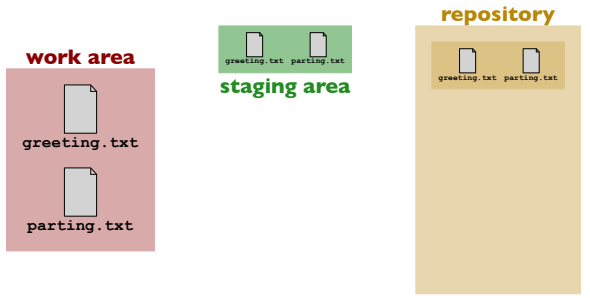
workarea difference from **staging area**

Checking the Status




```
$ git reset --hard
HEAD is now at 04be3f4 first

$ git status
On branch main
nothing to commit, working tree clean
```

Checking the Status




```
$ git reset --hard
HEAD is now at 04be3f4 first

$ git status
On branch main
nothing to commit, working tree clean
```

commit in repository

Cloning

Using **`git clone`** is usually better than **`cp -r`**:

```
$ cd ..
$ git clone r1 r3
Cloning into 'r3'...
done.
$ cd r3
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Cloning

Using **`git clone`** is usually better than **`cp -r`**:

```
$ cd ..
$ git clone r1 r3
Cloning into 'r3'...
done.
$ cd r3
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Almost the same as
`cp -r r2/.git r3/.git`
plus **`git reset --hard`** …

Cloning

Using **`git clone`** is usually better than **`cp -r`**:

```
$ cd ..
$ git clone r1 r3
Cloning into 'r3'...
done.
$ cd r3
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

... but remembers **`r2`** as **`origin`**
for future **`git pull`** and **`git push`**

Cloning

```
$ git clone https://github.com/UtahMSD/r1
```

vs.

```
$ git clone r1
```

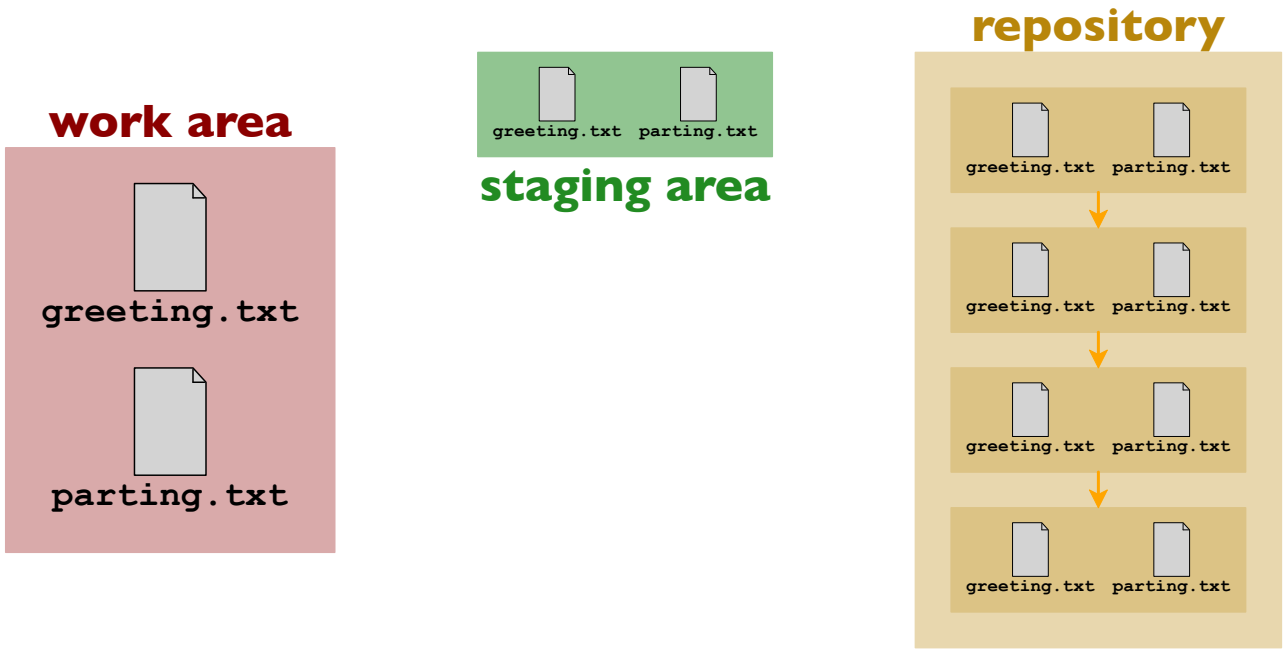
is similar to viewing

https://msd.utah.edu/index.html

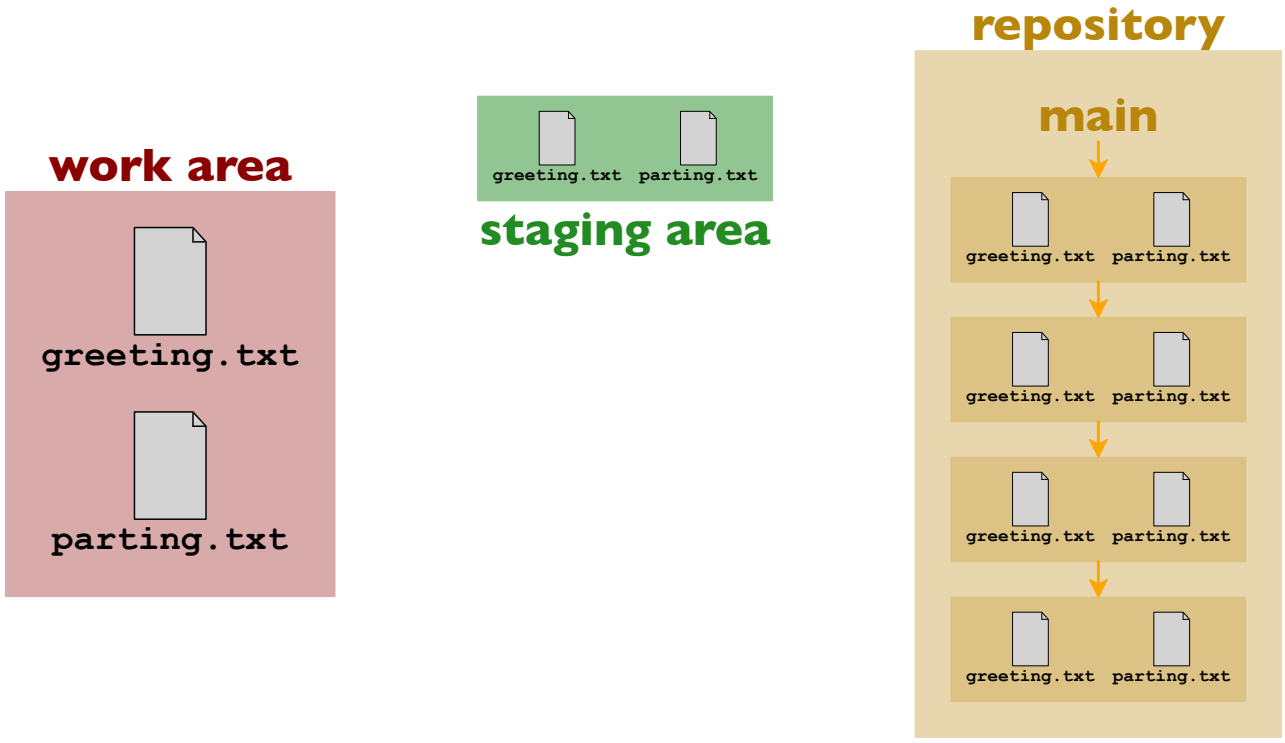
vs.

index.html

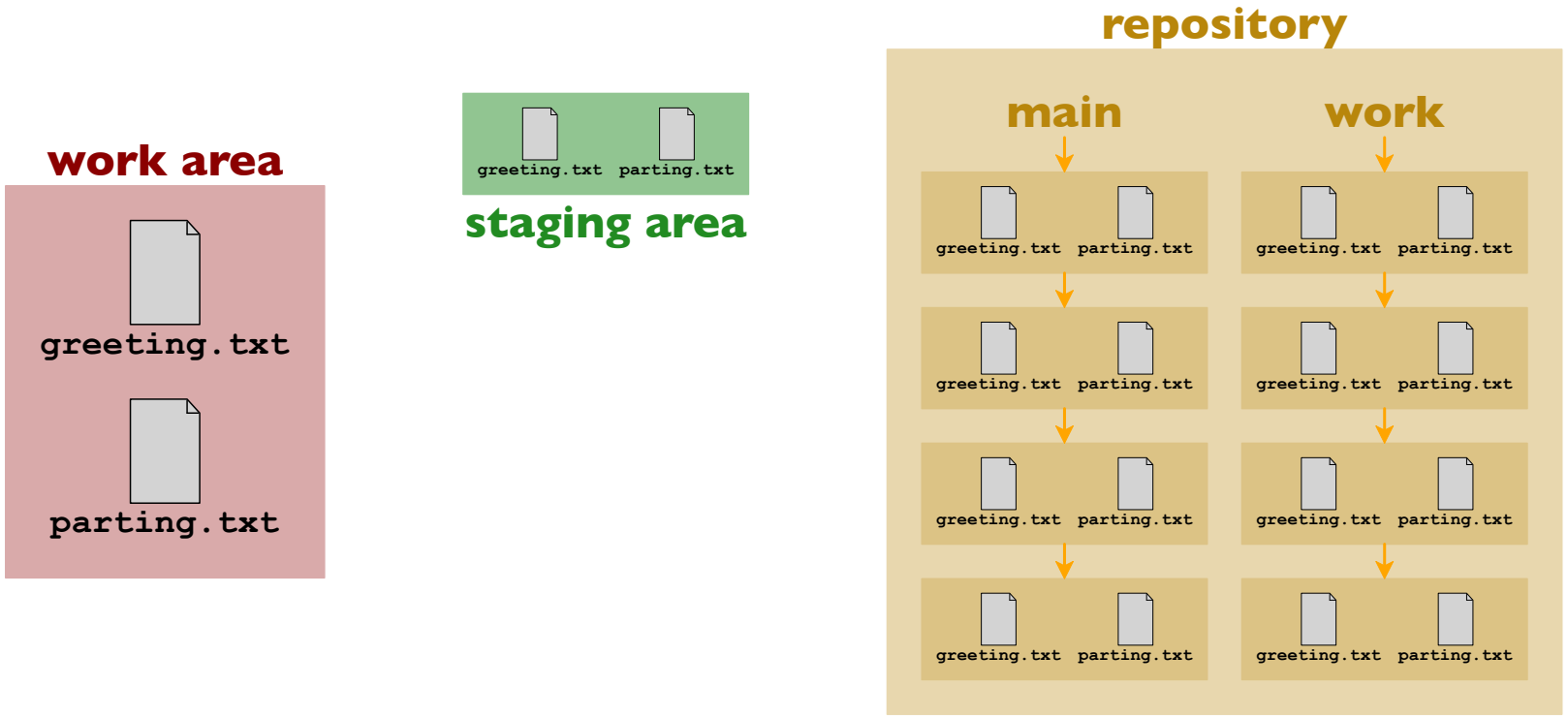
Branches


work area
greeting.txt
parting.txt
greeting.txt
parting.txt
staging area
repository
greeting.txt
parting.txt
greeting.txt
parting.txt
greeting.txt
parting.txt
greeting.txt
parting.txt

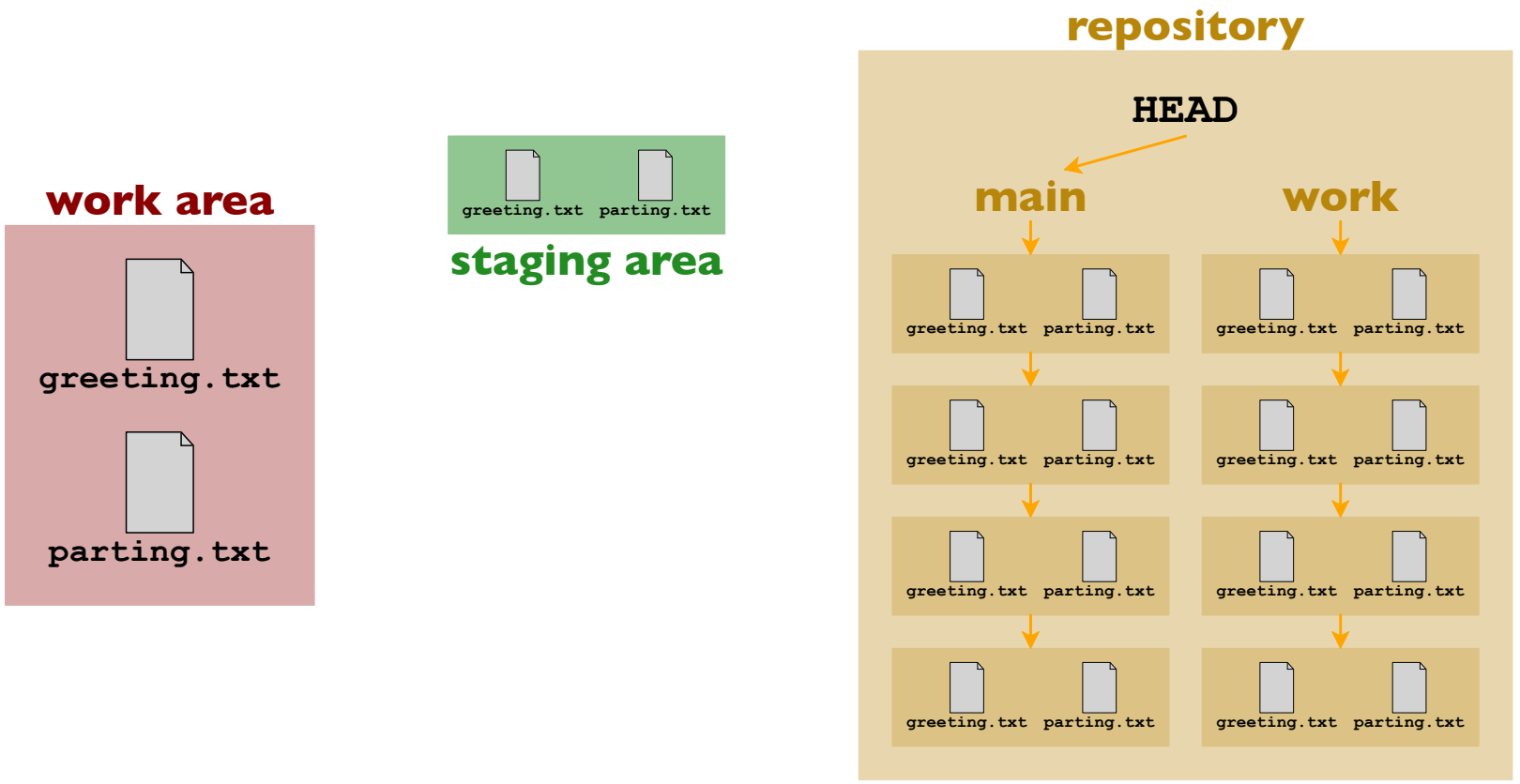
Branches


work area
greeting.txt
parting.txt
greeting.txt
parting.txt
staging area
repository
main
greeting.txt
parting.txt
greeting.txt
parting.txt
greeting.txt
parting.txt
greeting.txt
parting.txt

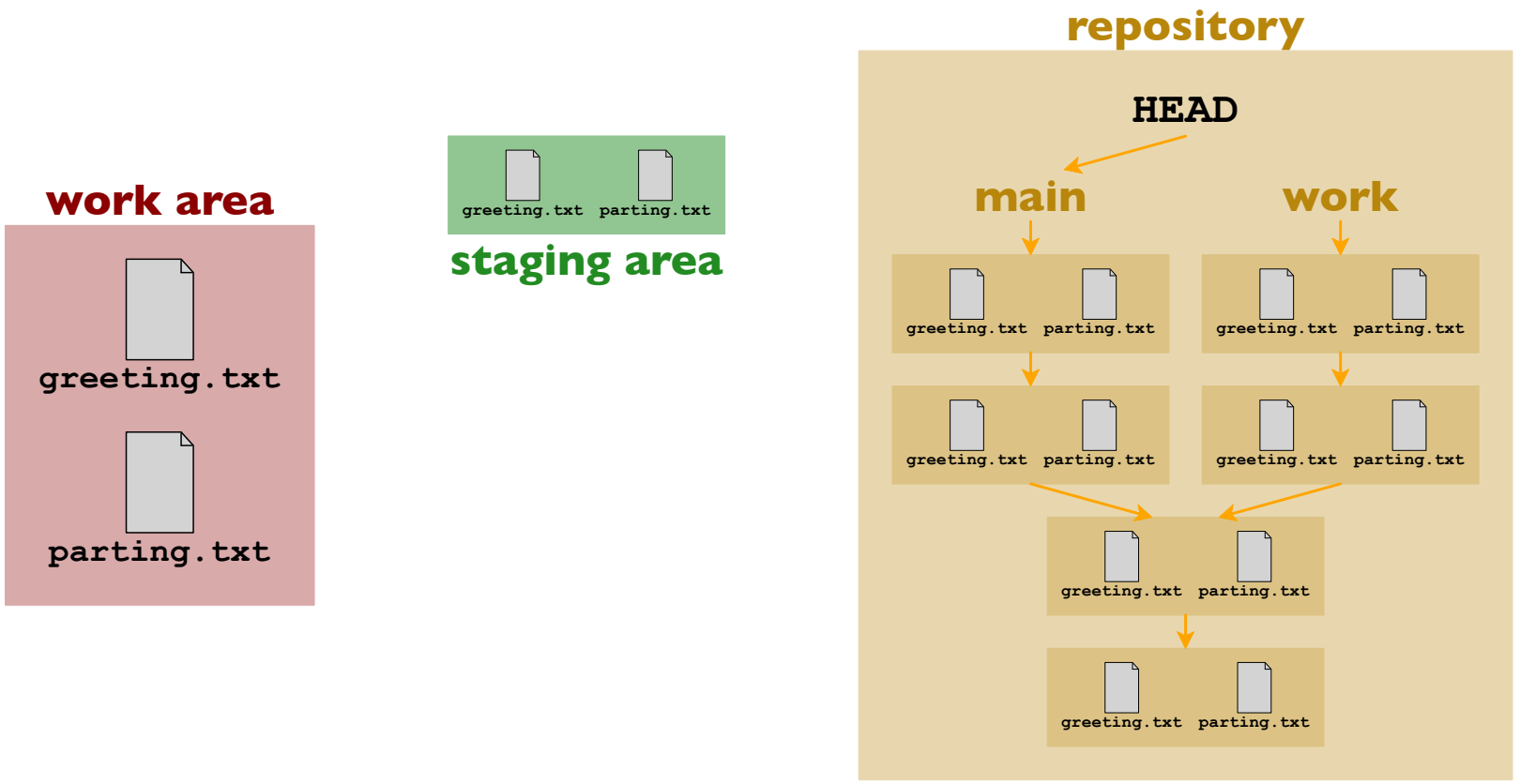
Branches


work area
greeting.txt
parting.txt
greeting.txt
parting.txt
staging area
repository
main
work
greeting.txt
parting.txt

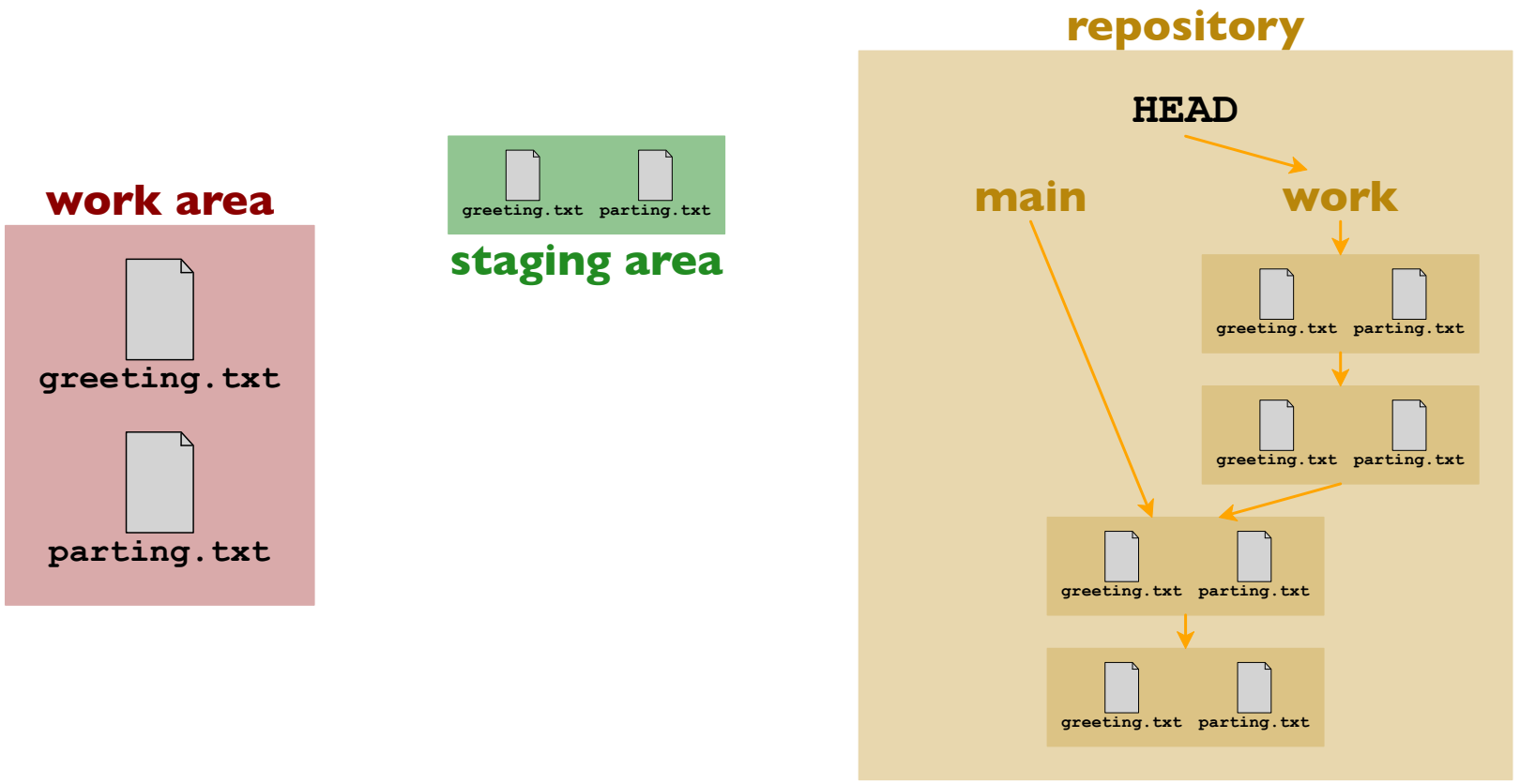
Branches


work area
greeting.txt
parting.txt
greeting.txt parting.txt
staging area
repository
HEAD
main
work
greeting.txt parting.txt

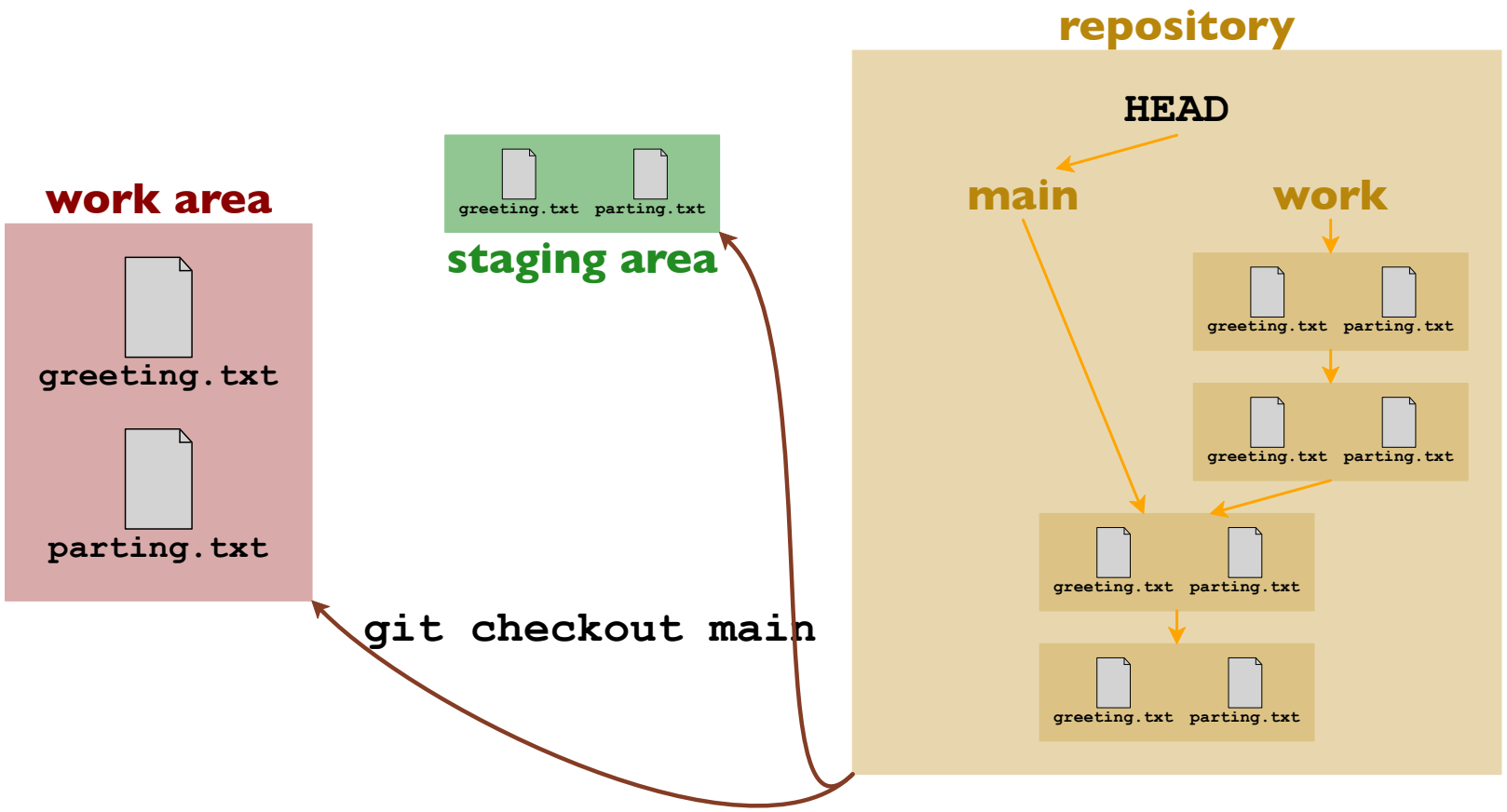
Branches


work area
greeting.txt
parting.txt
greeting.txt parting.txt
staging area
repository
HEAD
main
work
greeting.txt parting.txt
greeting.txt parting.txt
greeting.txt parting.txt
greeting.txt parting.txt
greeting.txt parting.txt
greeting.txt parting.txt
greeting.txt parting.txt

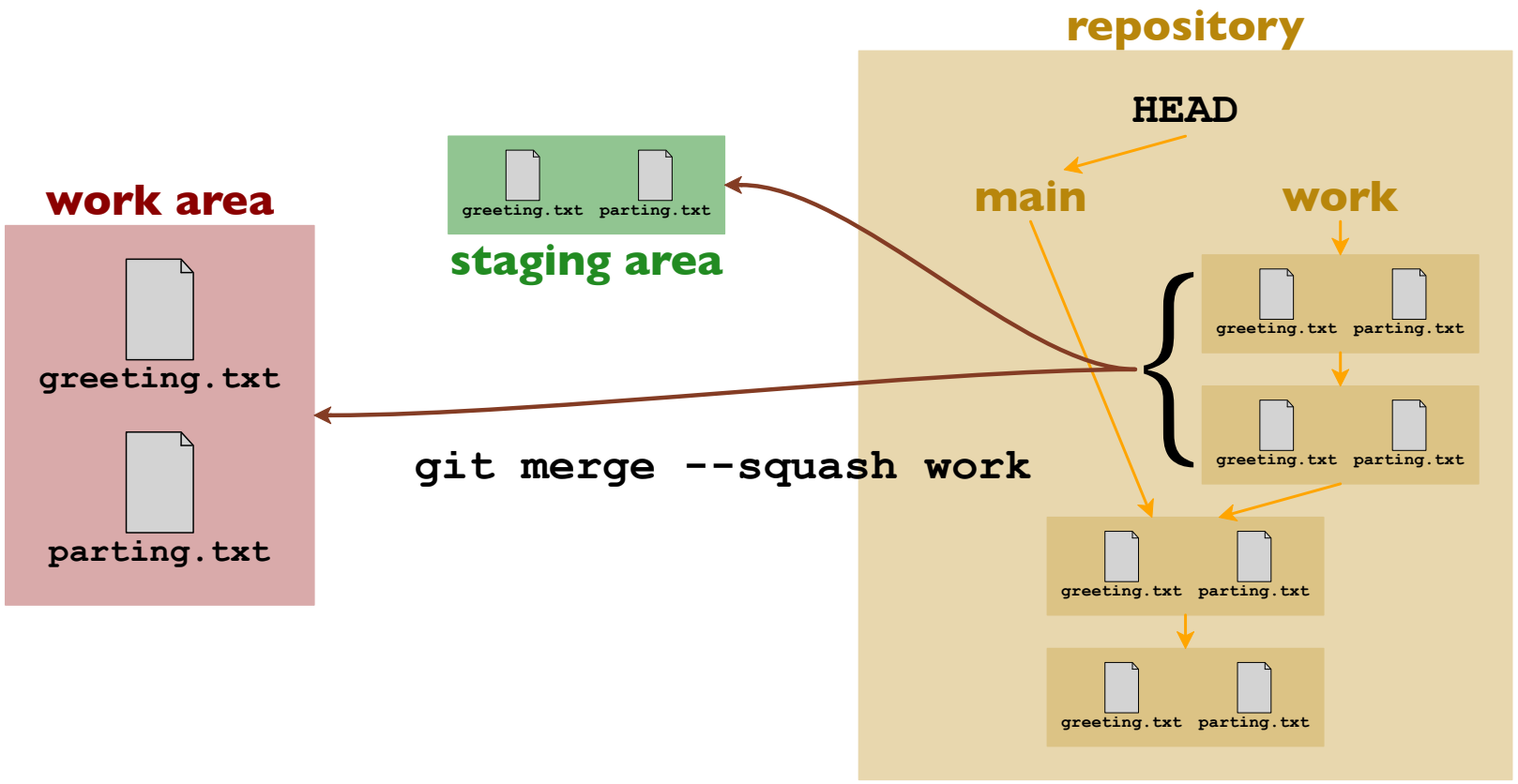
Branches


work area
greeting.txt
parting.txt
greeting.txt
parting.txt
staging area
repository
HEAD
main
work
greeting.txt
parting.txt
greeting.txt
parting.txt
greeting.txt
parting.txt
greeting.txt
parting.txt

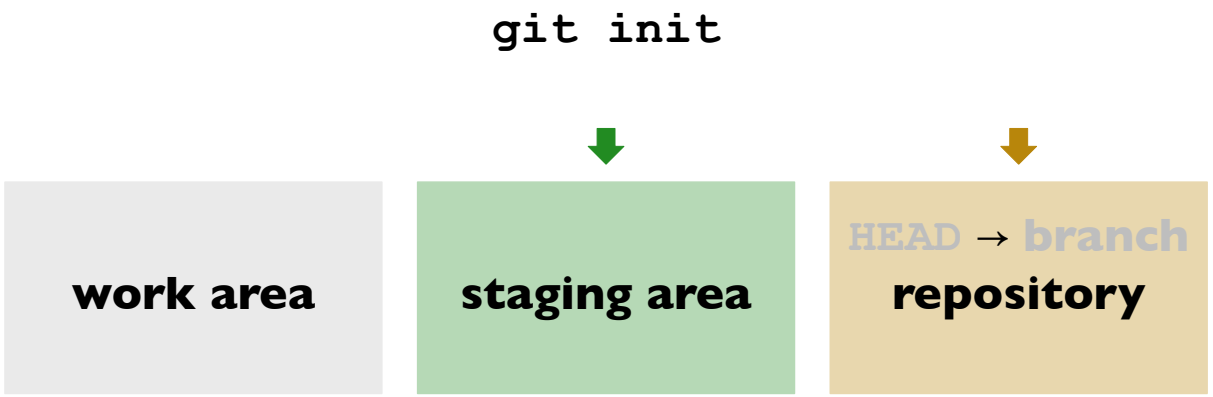
Branches


repository
HEAD
main
work
greeting.txt
parting.txt
staging area
greeting.txt
parting.txt
work area
greeting.txt
parting.txt
git checkout main

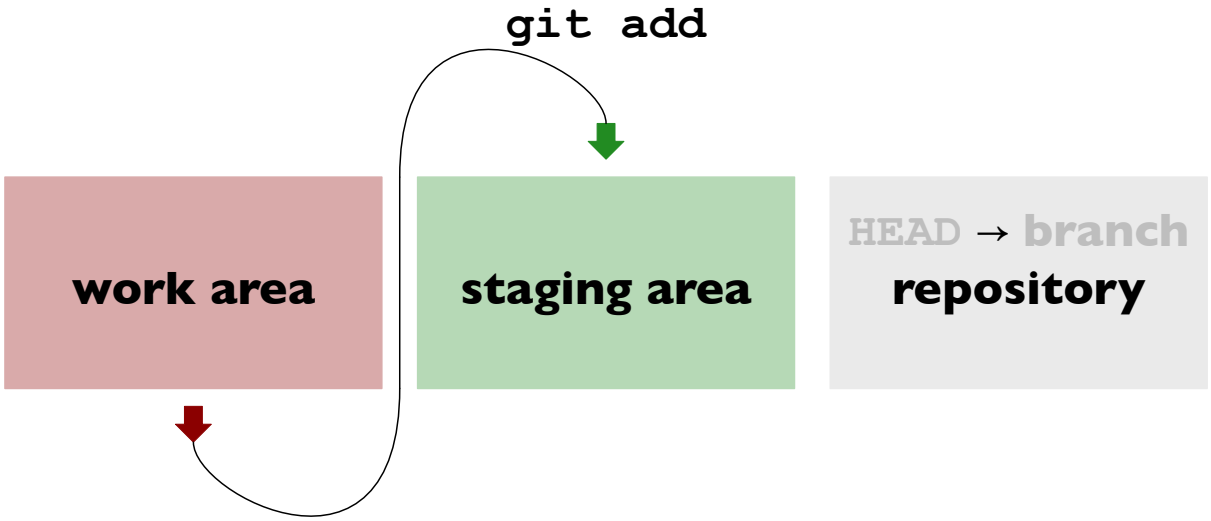
Branches


repository
HEAD
main
work
work area
staging area
greeting.txt
parting.txt
git merge --squash work

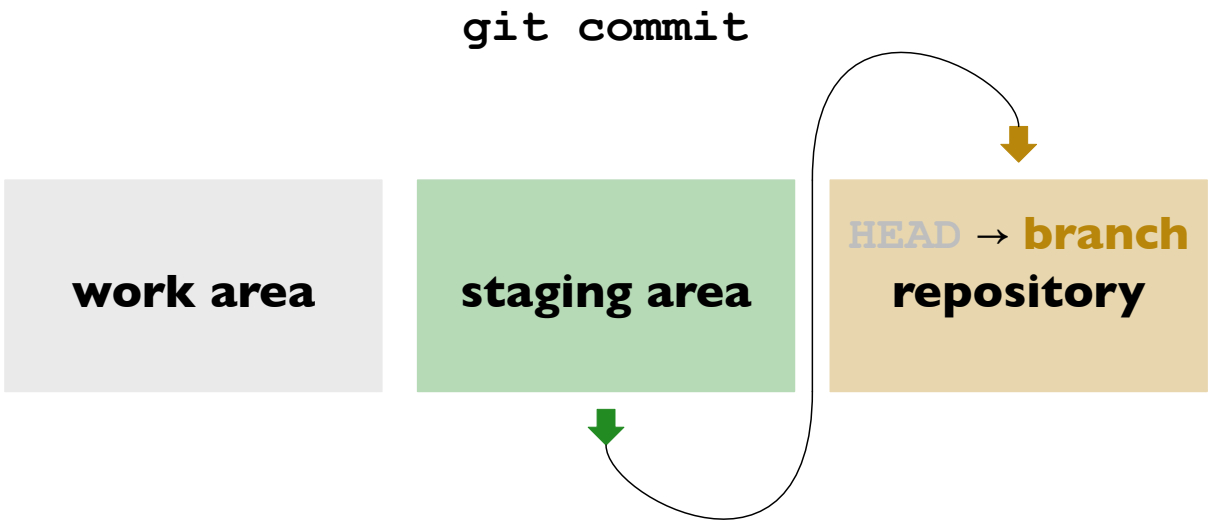
Git Command Survey



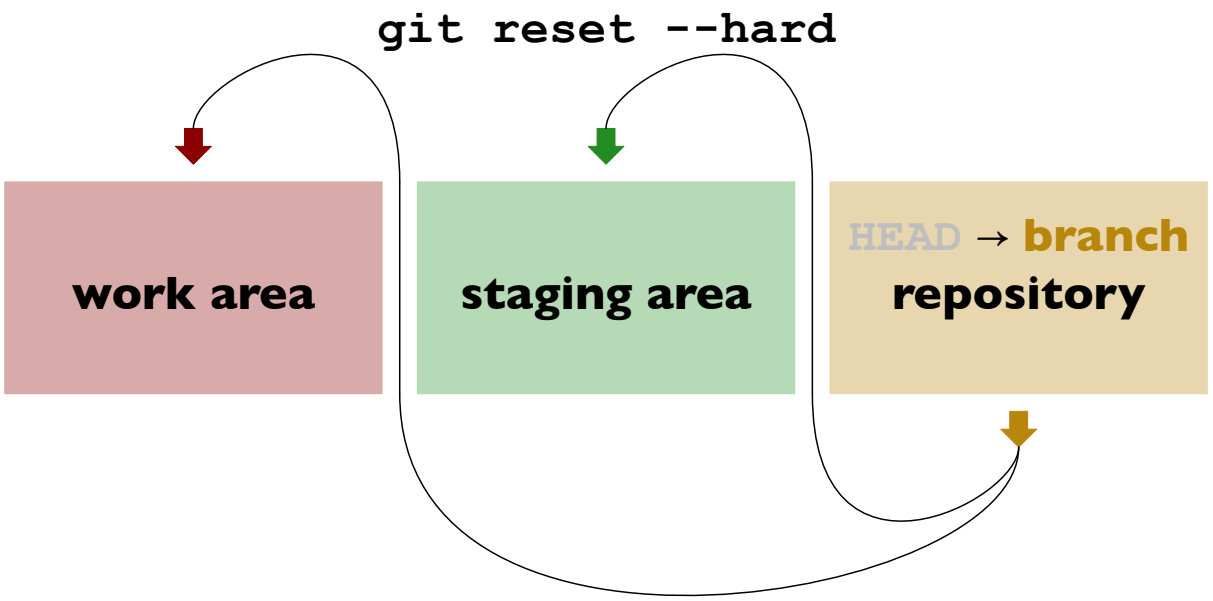
Git Command Survey



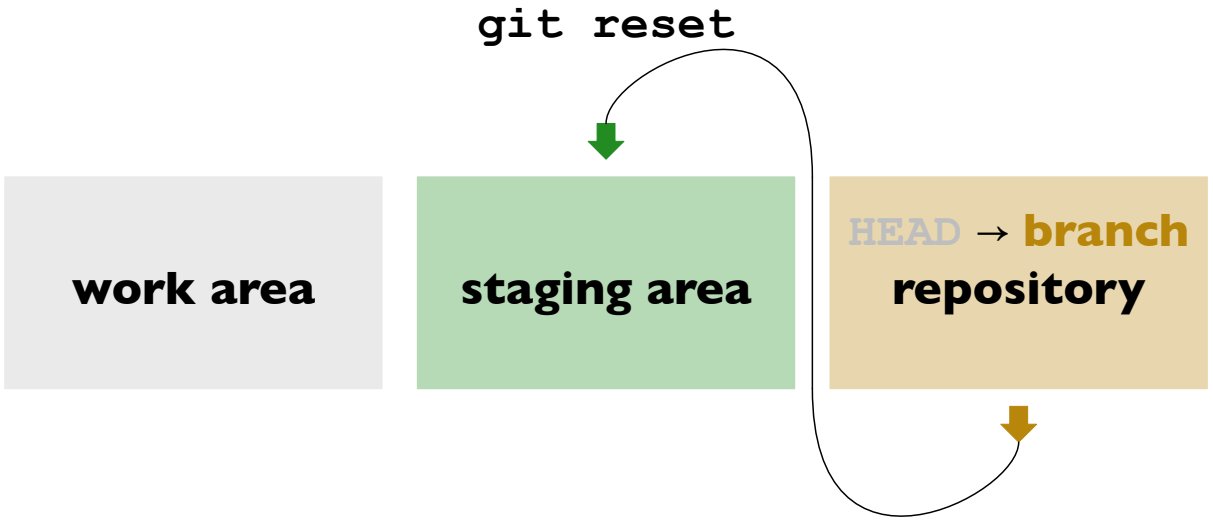
Git Command Survey



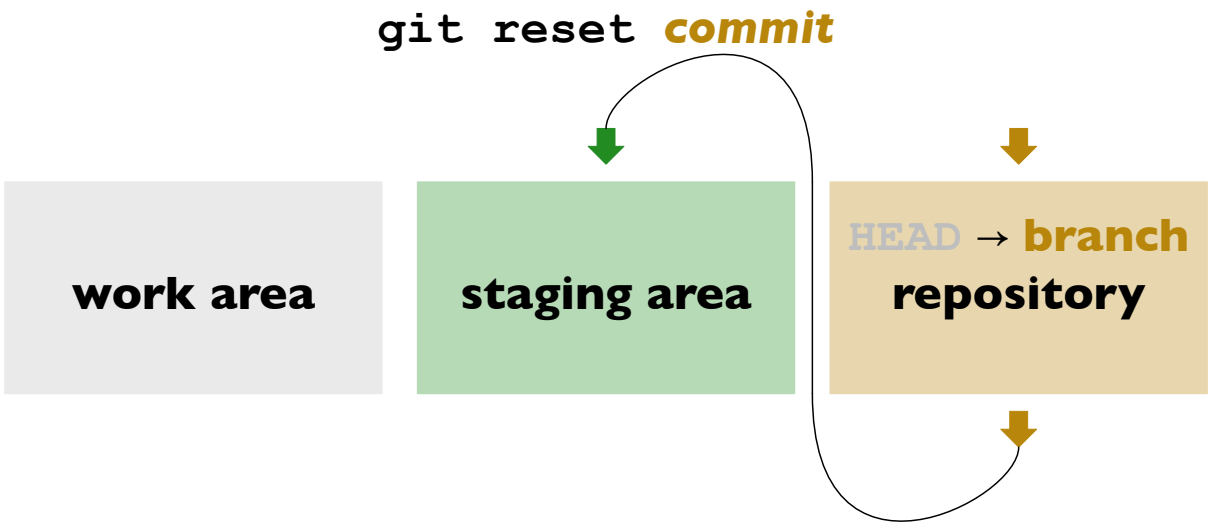
Git Command Survey



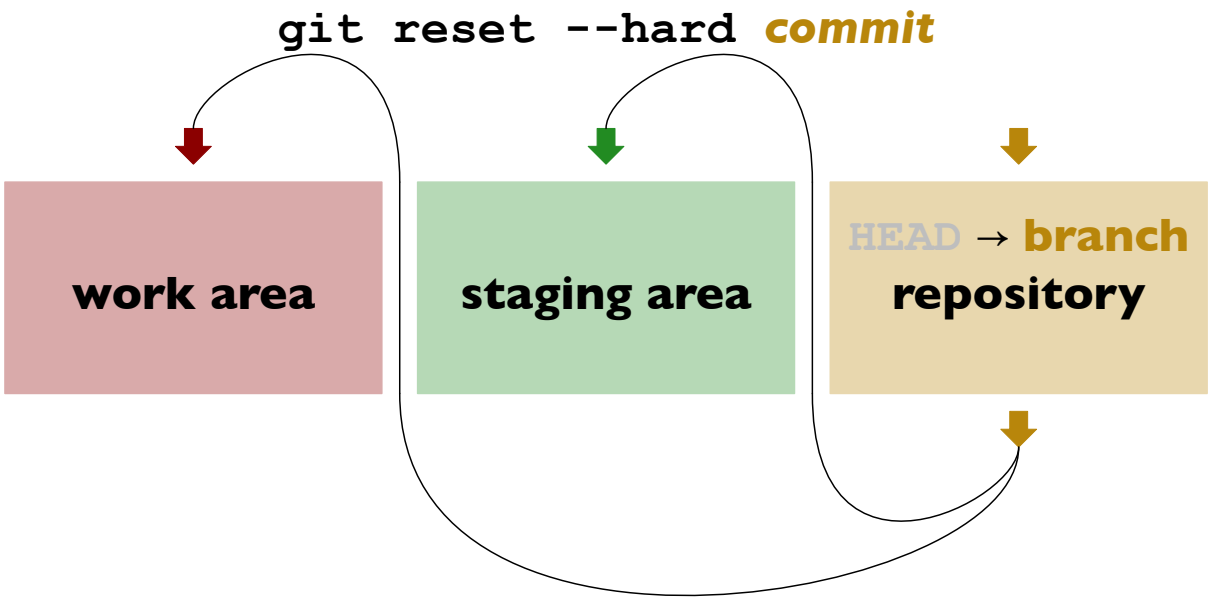
Git Command Survey



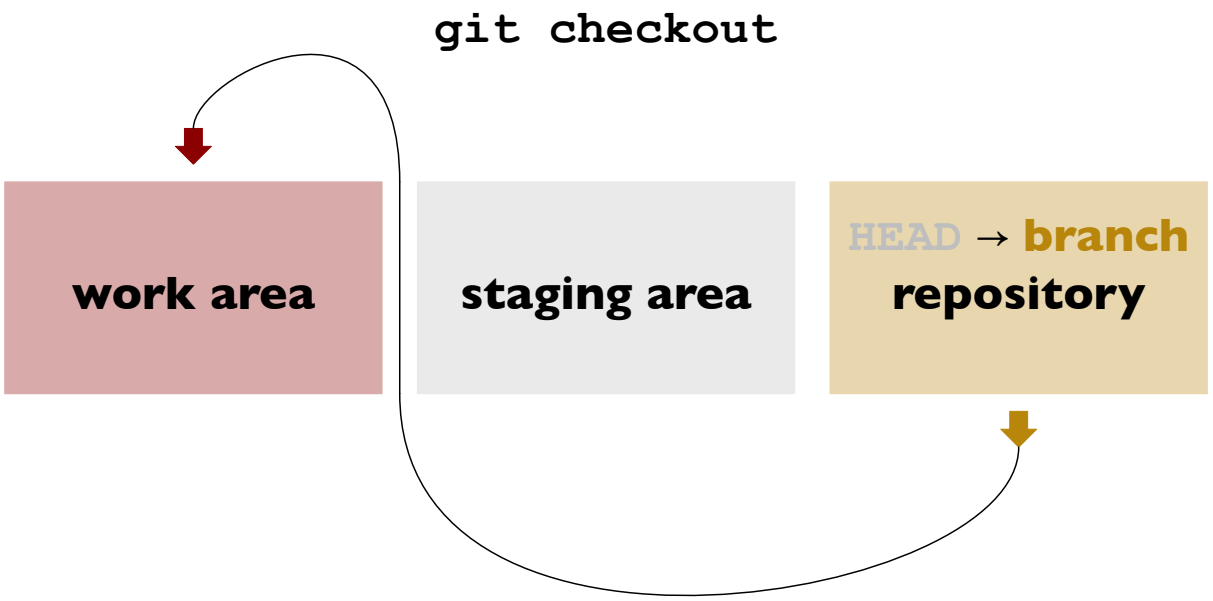
Git Command Survey



Git Command Survey

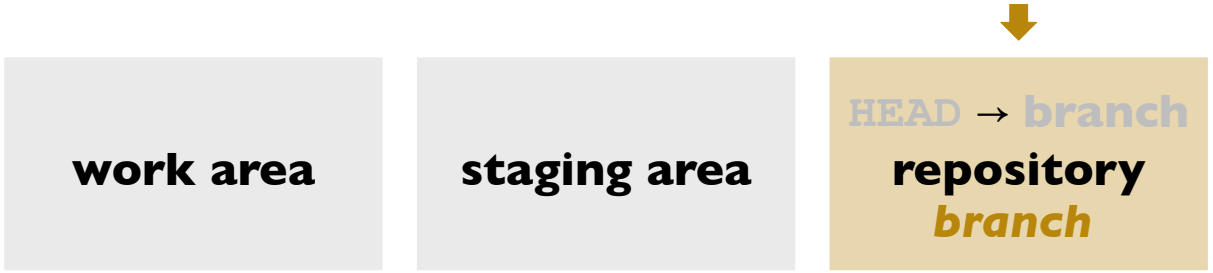


Git Command Survey

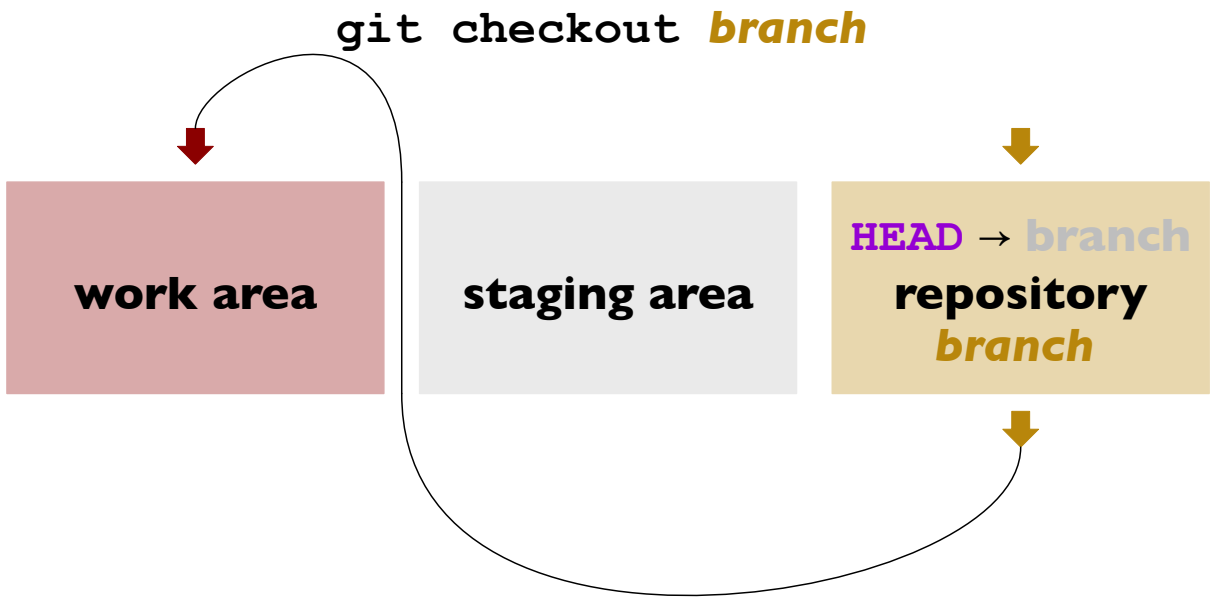


Git Command Survey

`git branch` ***branch***

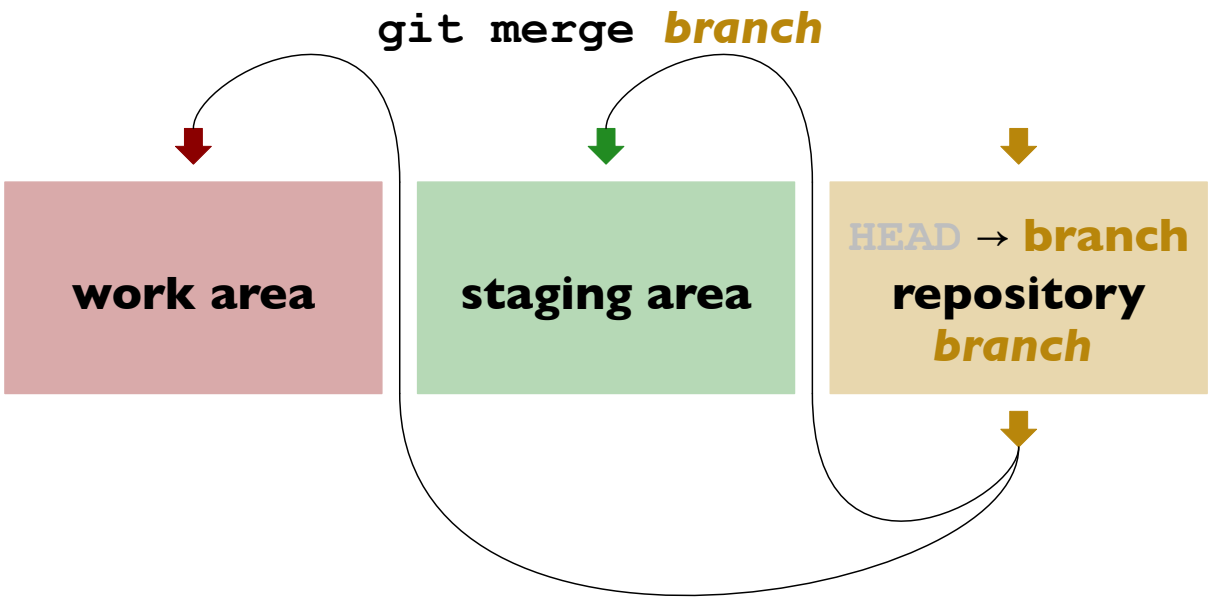


Git Command Survey



If possible while keeping **work area** and **staging area** changes!

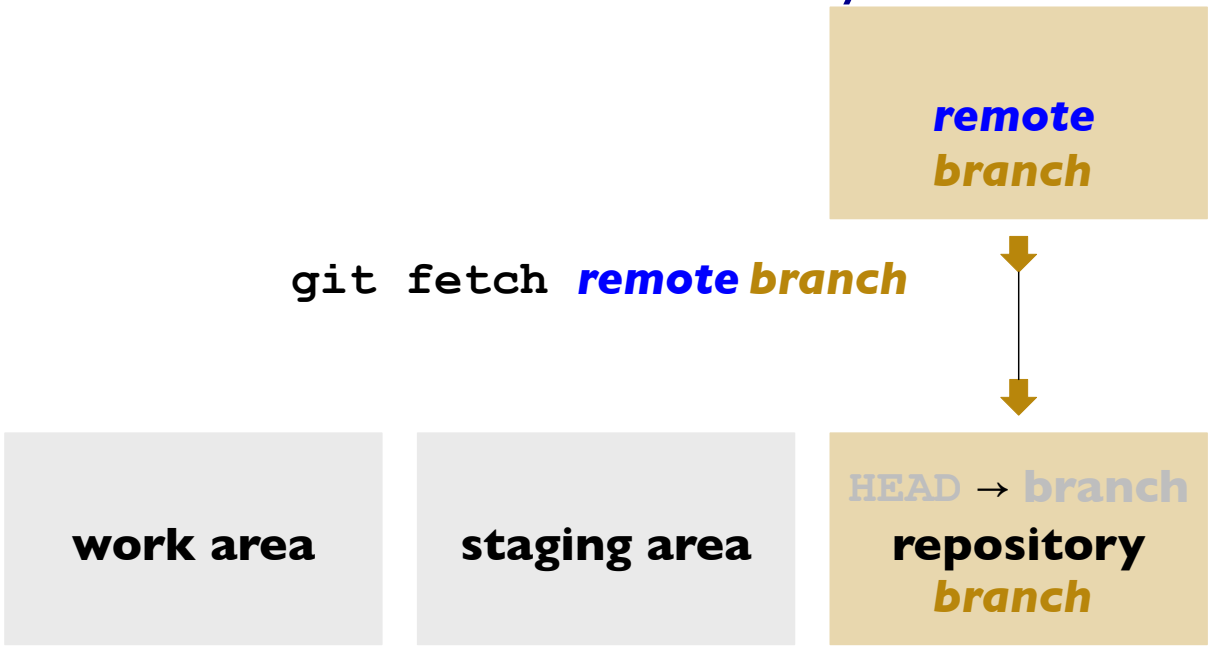
Git Command Survey



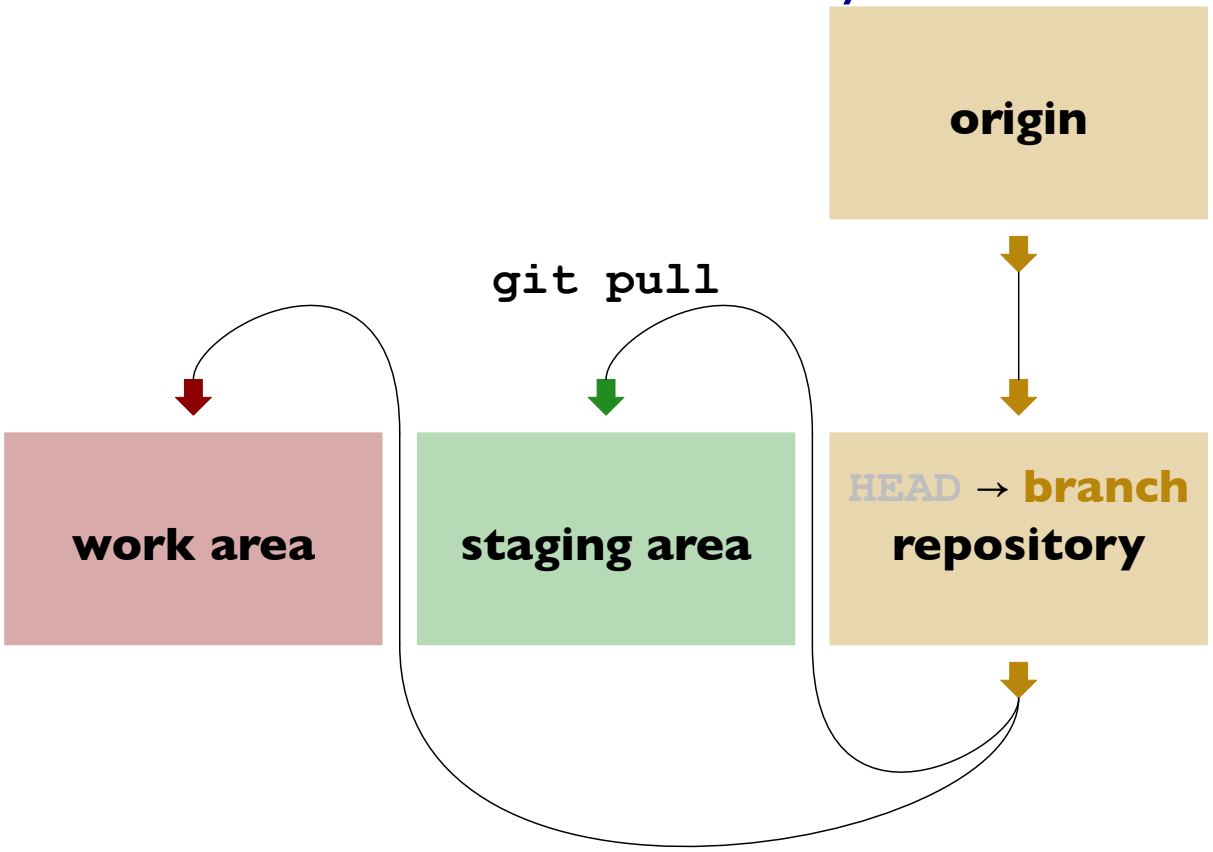
Git Command Survey

Git Command Survey

Git Command Survey

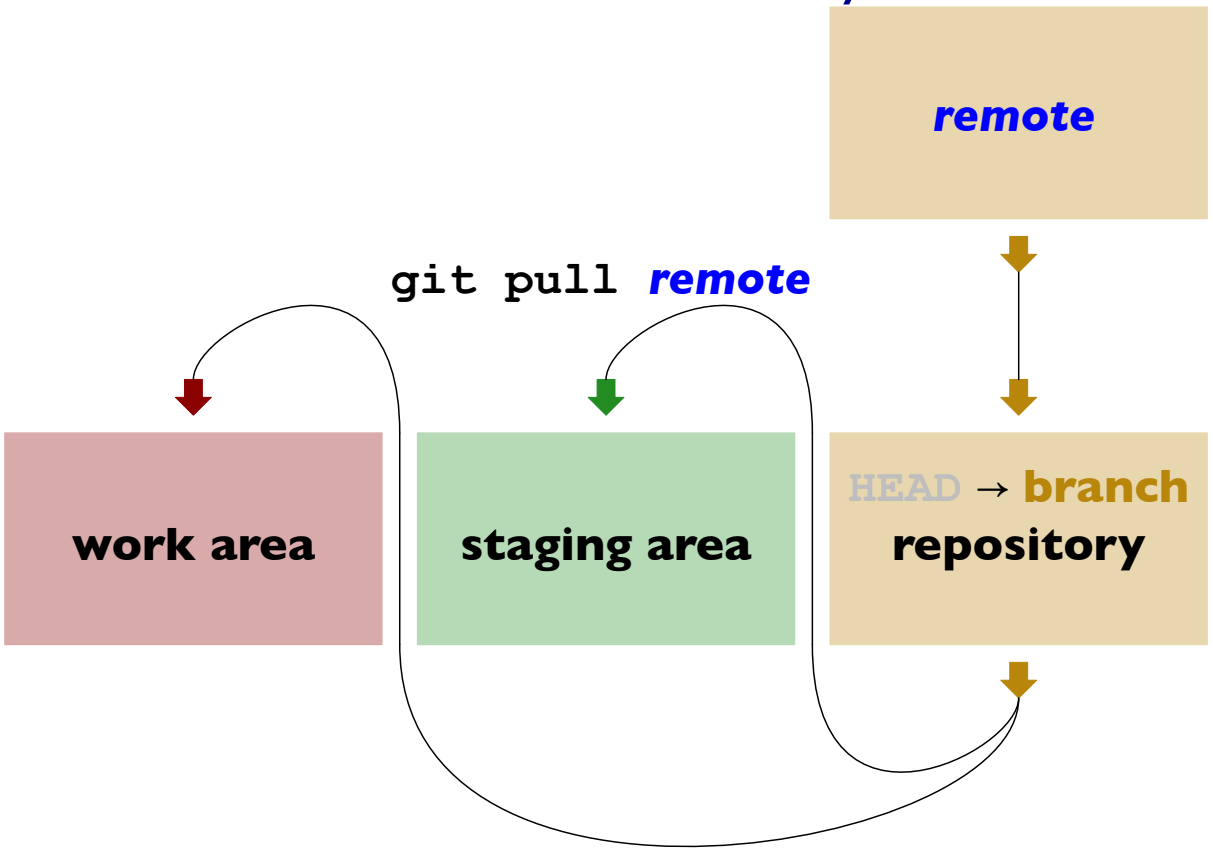


Git Command Survey



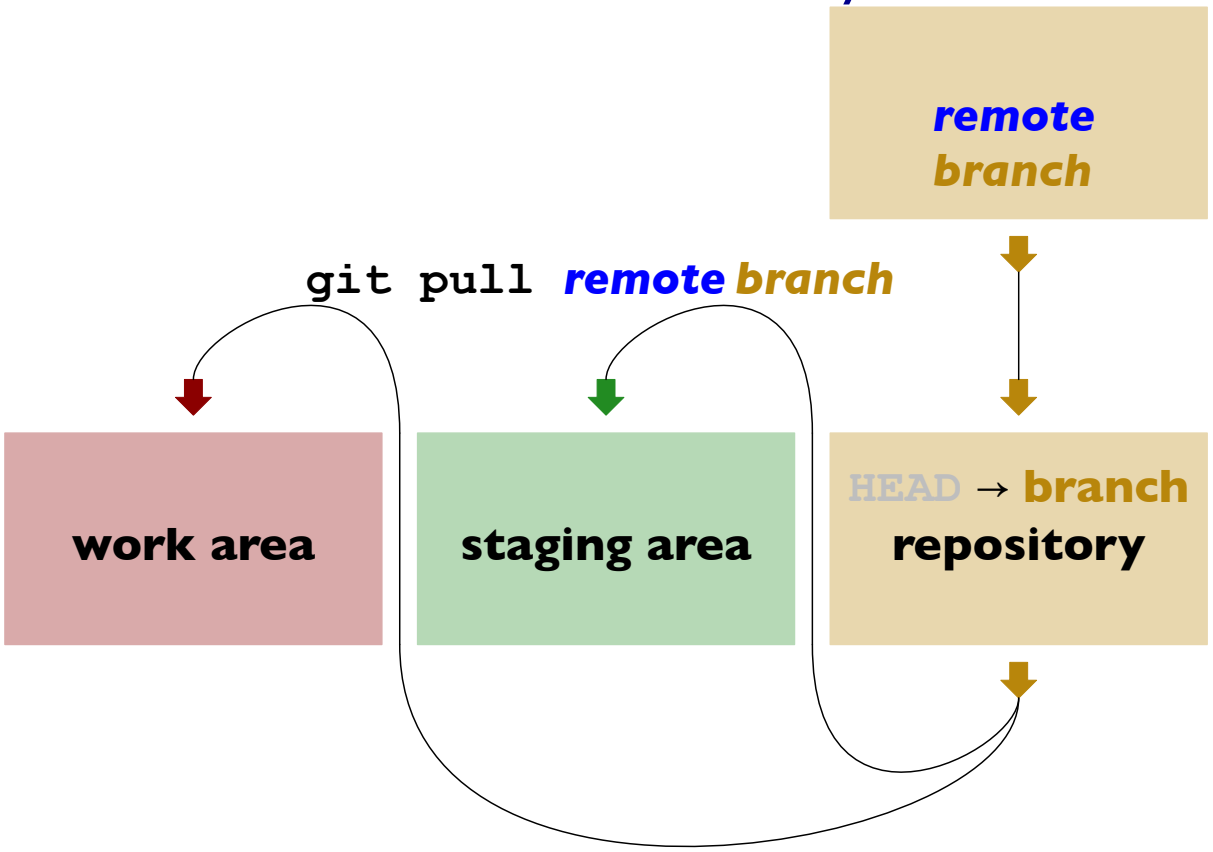
`git pull` is essentially `git fetch && git merge`

Git Command Survey



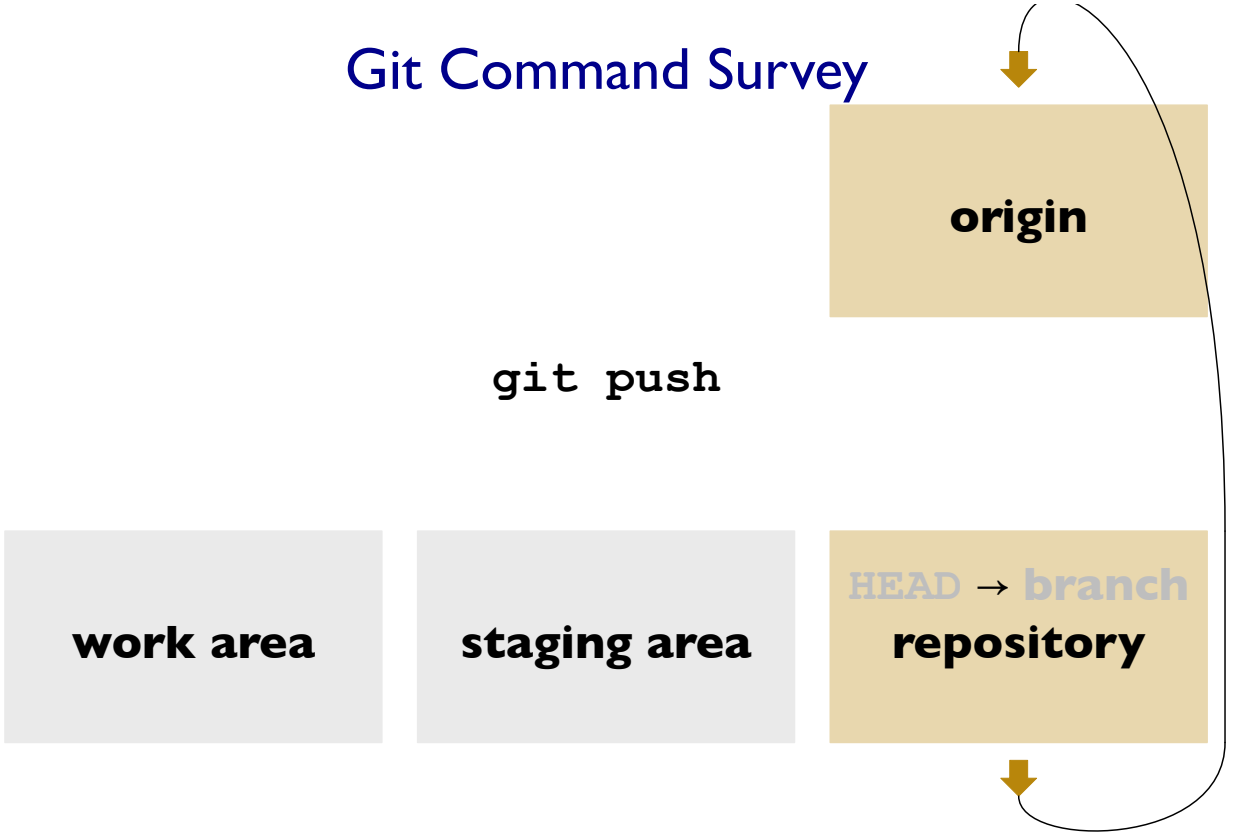
`git pull` is essentially `git fetch && git merge`

Git Command Survey

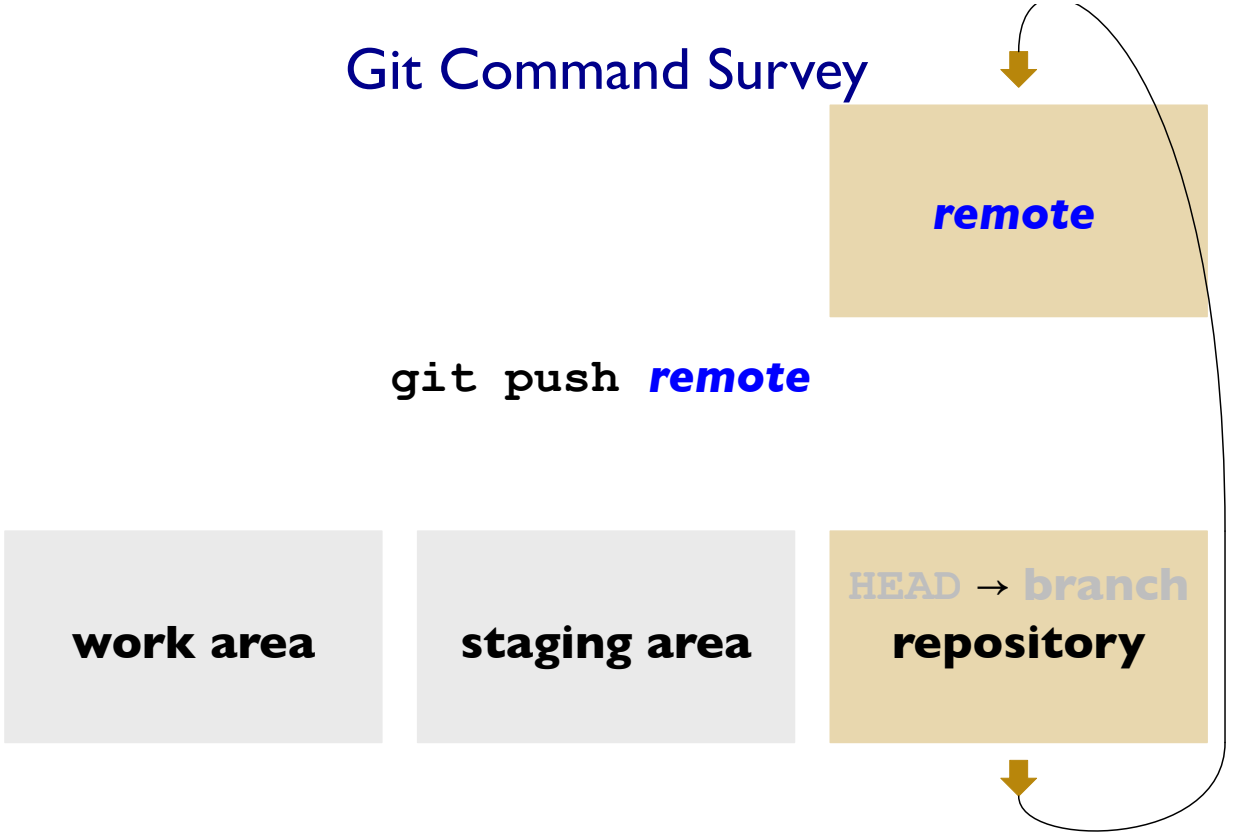


`git pull` is essentially `git fetch && git merge`

Git Command Survey



Git Command Survey



Git Command Survey