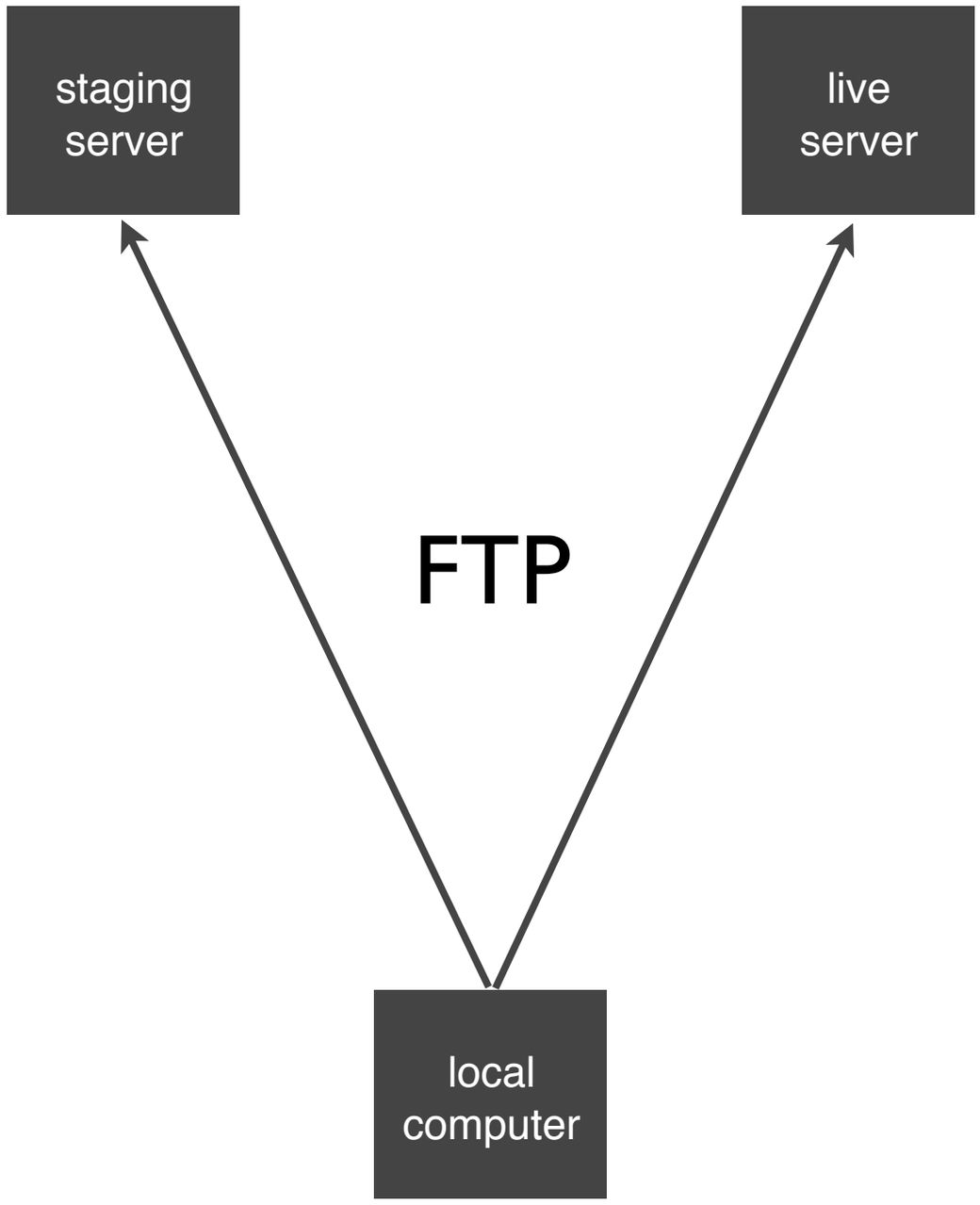
git milk?

a practical guide for git newbies

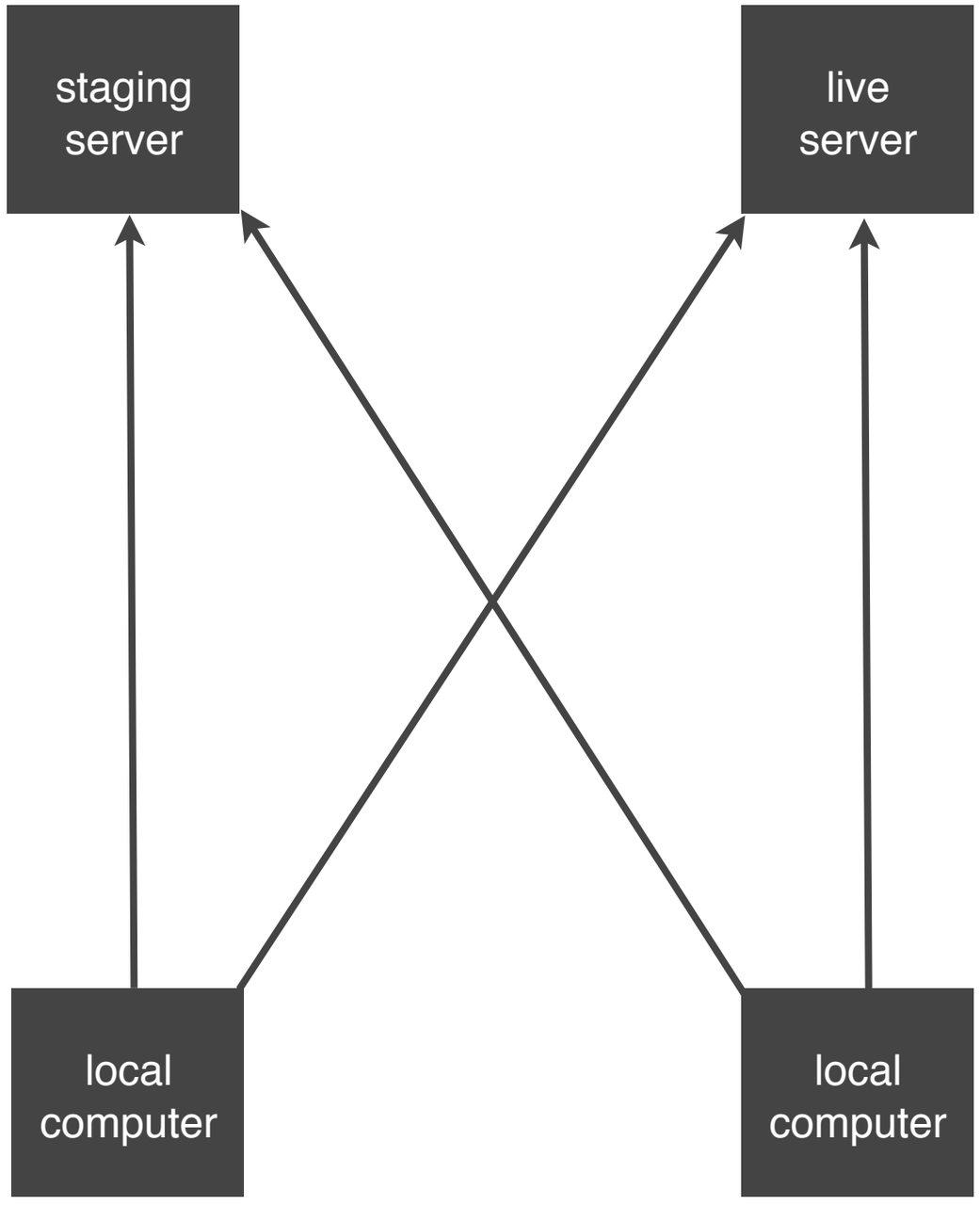
Rick Hood

drupal.org/user/54879

common media inc.
greenfield, ma



one developer... ...this works



this does not work

need version control of some sort

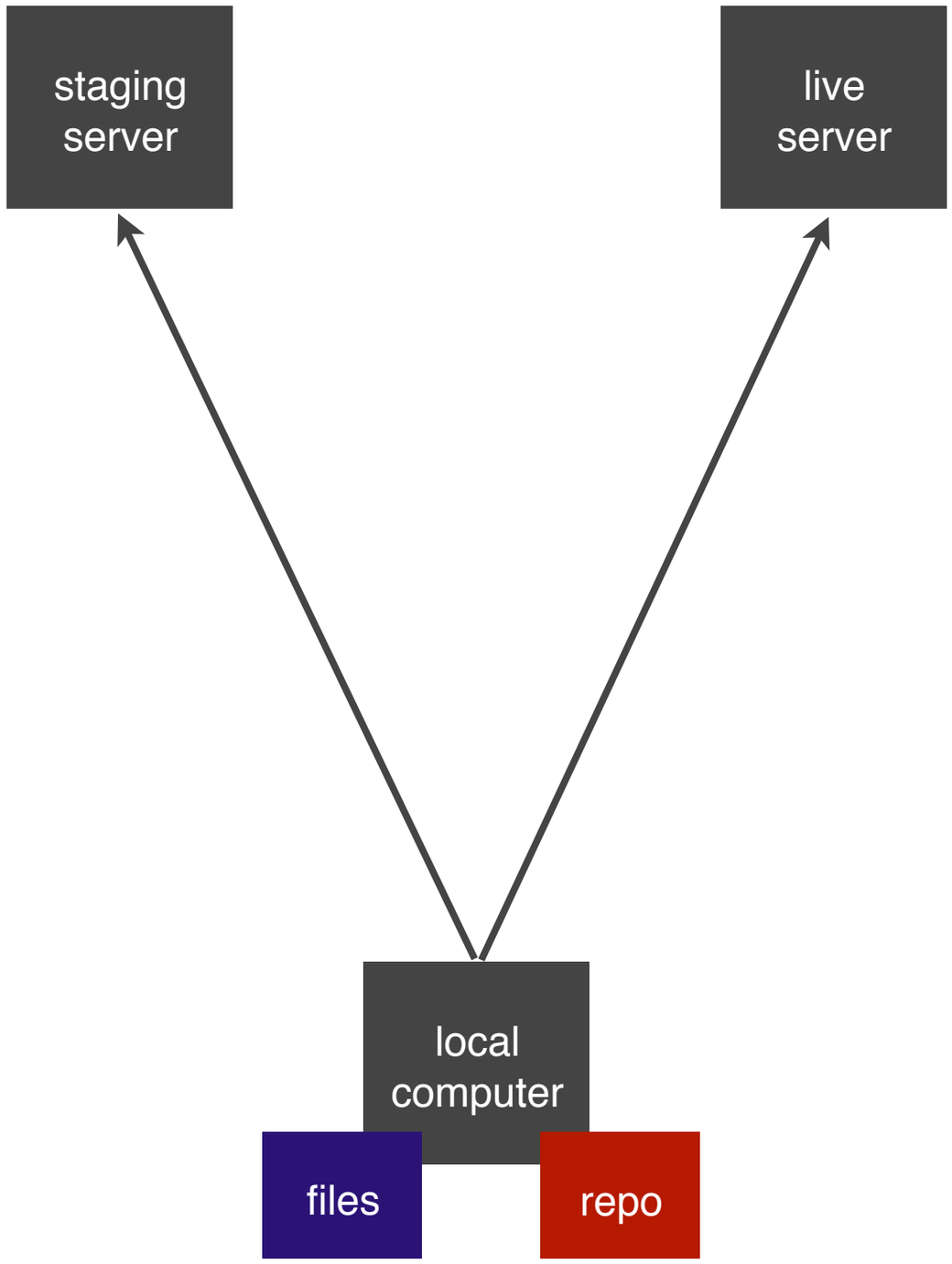
need version control of some sort

git = one way to do version control

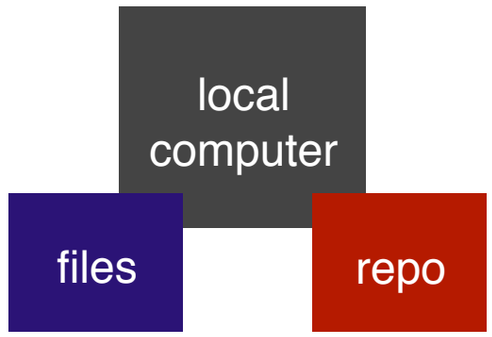
repository

repository

contains data on what files have changed and how

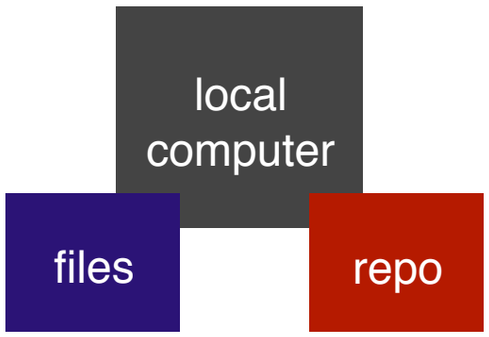


now with version control I have files and a repo



now with version control I have files and a repo

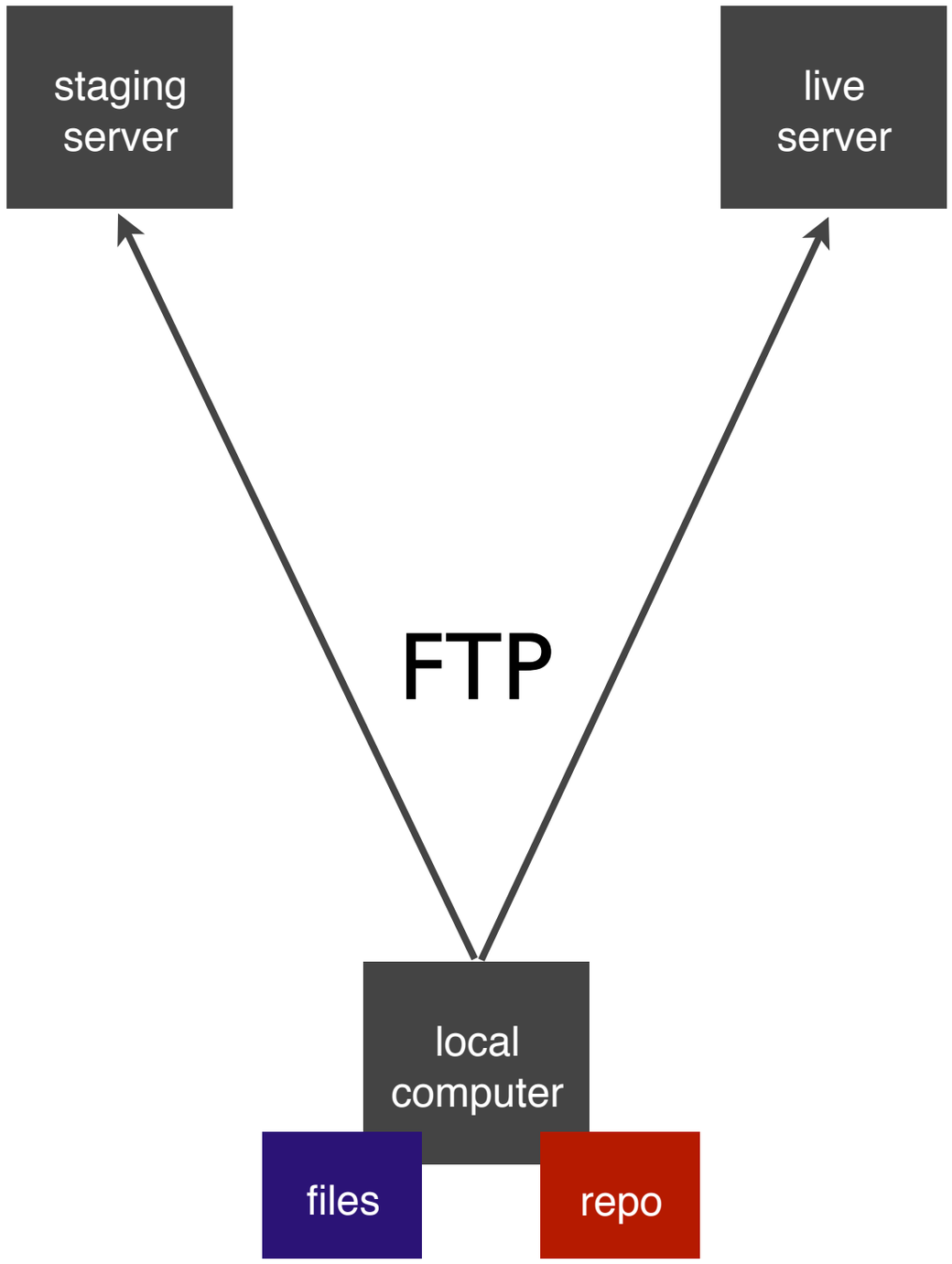
the files I had before... ... not new



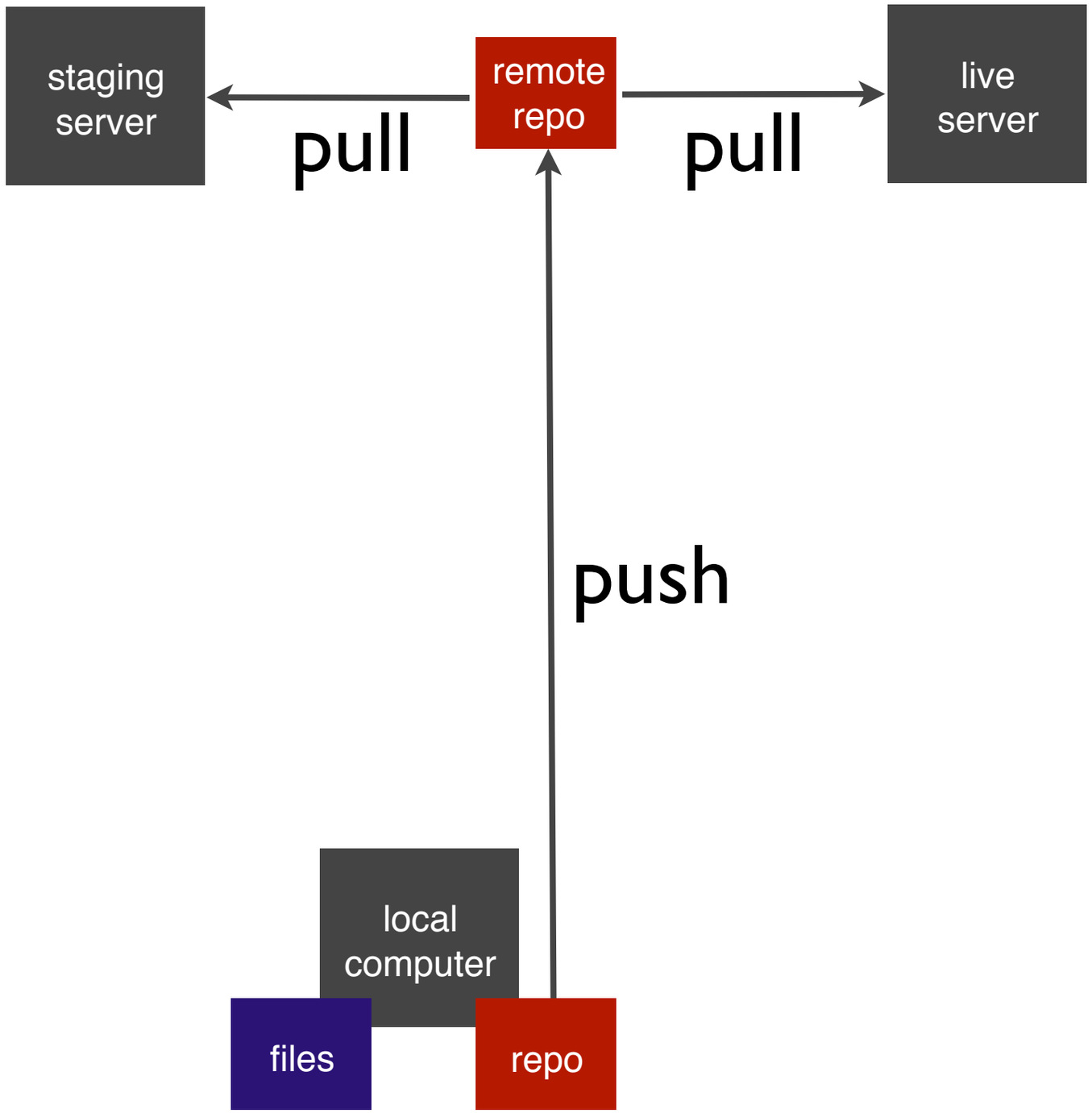
now with version control I have files and a repo

the files I had before... ... not new

the repository is new; it keeps track of changes to files



back to this



no more FTP... ...push to repo, pull from repo

branch

branch

repos can have more than one version

branch

repos can have more than one version

a version of a repo is called a branch

branches

master

the one you don't touch until you are sure

devel

the one you work with until you are sure

branches

master

the one you don't touch until you are sure

devel

the one you work with until you are sure

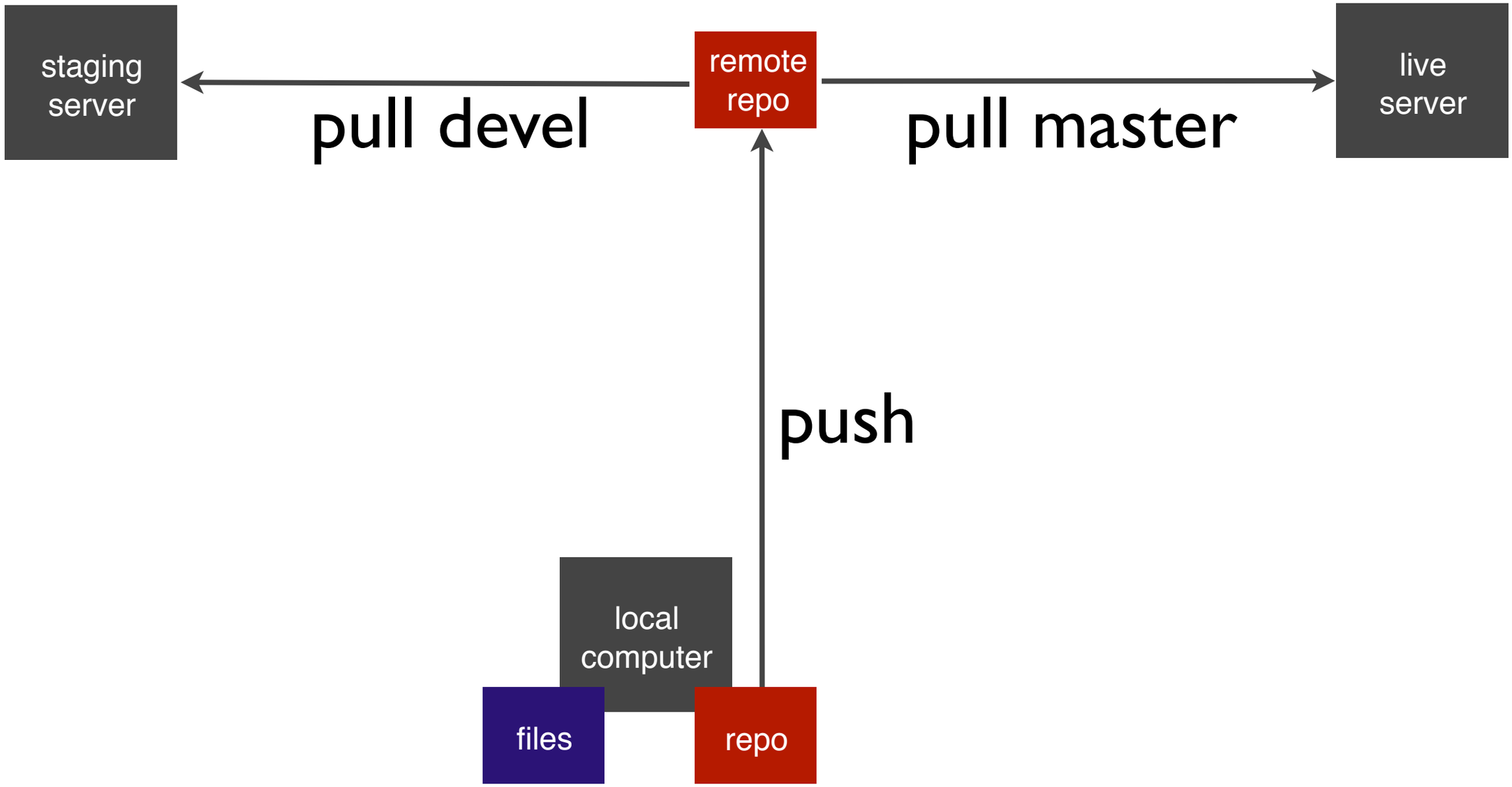
my_cool_feature

a branch for working on something completely different; until you are sure it works

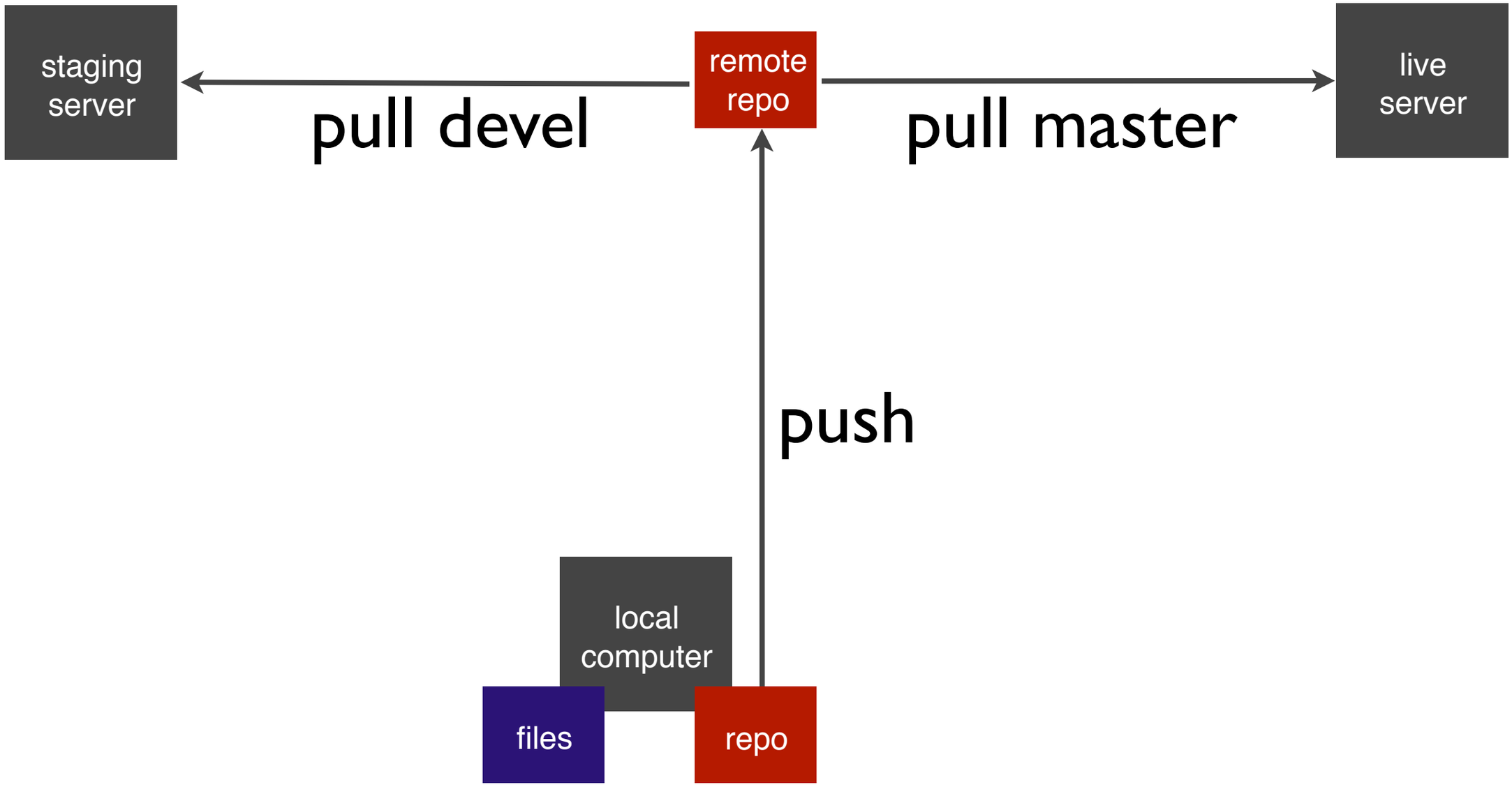
when you are sure... you merge

my_cool_feature  devel

devel  master

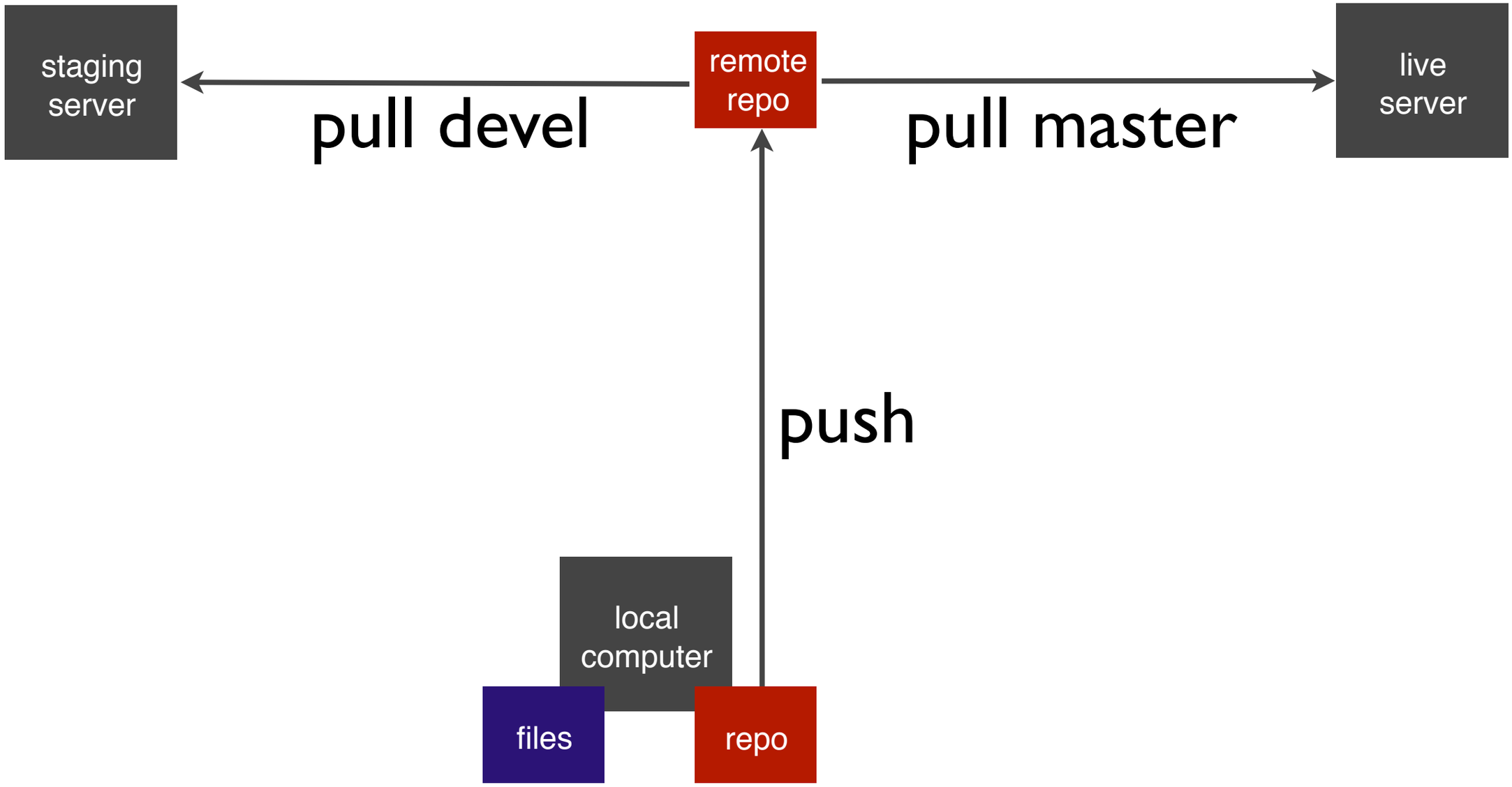


pull devel to staging and master to live



pull devel to staging and master to live

once devel is merged into master, files are the same



pull devel to staging and master to live

once devel is merged into master, files are the same

at which point staging and live are the same if you have pulled to both

ok that's it on the general idea

questions so far?

ok that's it on the general idea

so how do you actually do this?

install git

install git

http://git-scm.com/

install git

http://git-scm.com/

http://thinkvitamin.com/code/how-to-install-git/
(http://bit.ly/gitinstall)

git commands

git commands

http://gitref.org

```
Richard-Hoods-MacBook-Pro:~ rick$ git --version
git version 1.7.5.4
Richard-Hoods-MacBook-Pro:~ rick$
```

Start up terminal. Do git --version (or any git command) to see if git is working.

```
Richard-Hoods-MacBook-Pro:~ rick$ git --version
git version 1.7.5.4
Richard-Hoods-MacBook-Pro:~ rick$
Richard-Hoods-MacBook-Pro:~ rick$ cd sites
Richard-Hoods-MacBook-Pro:sites rick$ cd git
Richard-Hoods-MacBook-Pro:git rick$ ls
hello.txt
Richard-Hoods-MacBook-Pro:git rick$ git init
Initialized empty Git repository in /Users/rick/Sites/Git/.git/
Richard-Hoods-MacBook-Pro:git rick$
```

I cd to my sites folder, then to a folder I called “git”.
In that folder I do git init to start a new git repo there.

Initialized empty Git repository in /Users/rick/Sites/Git/.git/
Richard-Hoods-MacBook-Pro:git rick$ git status
On branch master
#
Initial commit
#
Untracked files:
(use "git add <file>..." to include in what will be committed)
#
hello.txt
nothing added to commit but untracked files present (use "git add" to track)
Richard-Hoods-MacBook-Pro:git rick$

This is telling me I have an untracked file (hello.txt) and
I should add it if I want to eventually commit it to the repo.

```
#   hello.txt
nothing added to commit but untracked files present (use
"git add" to track)
Richard-Hoods-MacBook-Pro:git rick$ git add .
Richard-Hoods-MacBook-Pro:git rick$ git commit -m
"added hello.txt"
[master (root-commit) f13fbfe] added hello.txt
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 hello.txt
Richard-Hoods-MacBook-Pro:git rick$
```

Did add . to add all changes to be ready to commit.
Then did a commit to the repo.
Shortcut command to add and commit at the same time:
git commit -am "added hello.txt" (note the -a) but this does not work on new files, only on files already added.

```
 git add" to track)
Richard-Hoods-MacBook-Pro:git rick$ git add .
Richard-Hoods-MacBook-Pro:git rick$ git commit -m
"added hello.txt"
[master (root-commit) f13fbfe] added hello.txt
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 hello.txt
Richard-Hoods-MacBook-Pro:git rick$ git status
# On branch master
nothing to commit (working directory clean)
Richard-Hoods-MacBook-Pro:git rick$
```

On branch master, nothing to commit.
Now I go and do some work on hello.txt

```
Richard-Hoods-MacBook-Pro:git rick$
Richard-Hoods-MacBook-Pro:git rick$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be
committed)
#   (use "git checkout -- <file>..." to discard changes in
working directory)
#
#   modified:   hello.txt
#
no changes added to commit (use "git add" and/or "git
commit -a")
Richard-Hoods-MacBook-Pro:git rick$
```

This is telling me I am on branch master and I have modified hello.txt and I should do an add.

```
#   modified:   hello.txt
#
no changes added to commit (use "git add" and/or "git
commit -a")
Richard-Hoods-MacBook-Pro:git rick$
Richard-Hoods-MacBook-Pro:git rick$ git add hello.txt
Richard-Hoods-MacBook-Pro:git rick$ git commit -m
"changed hello.txt"
[master 51cafc2] changed hello.txt
 1 files changed, 1 insertions(+), 1 deletions(-)
Richard-Hoods-MacBook-Pro:git rick$
```

I did my add, this time using the filename, instead of `add .` (add all).
Then did my commit.

The workflow you will do over and over again is this:

A. Do some work on your files.
B. Do git status to see what git says has changed (optional)
C. Do git add . (or git add filename)
D. Do git commit -m “my commit message”

What if my commit was a mistake?

What if my commit was a mistake?

You can undo your last commit with:

git reset HEAD~1 (that is a one at the end)
(the above gets rid of the last commit)

What if my commit was a mistake?

You can undo your last commit with:

git reset HEAD~1 (that is a one at the end)
(the above gets rid of the last commit)

git reset --hard HEAD~1
(the above gets rid of the commit and the file changes)

http://bit.ly/git-undo-last-commit
(see comment 198)

questions?

branches

branches

a branch is version of the repository

branches

a branch is version of the repository

I can work on one branch without affecting other branches

branches

a branch is version of the repository

I can work on one branch without affecting other branches

Then later I can merge the branch I did my work on into the main branch

```
Richard-Hoods-MacBook-Pro:git rick$ git add hello.txt
Richard-Hoods-MacBook-Pro:git rick$ git commit -m
"changed hello.txt"
[master 51cafc2] changed hello.txt
 1 files changed, 1 insertions(+), 1 deletions(-)
Richard-Hoods-MacBook-Pro:git rick$
Richard-Hoods-MacBook-Pro:git rick$ git branch
* master
Richard-Hoods-MacBook-Pro:git rick$
```

This shows that there is one branch, called master, and I am on that branch.

```
Richard-Hoods-MacBook-Pro:git rick$
Richard-Hoods-MacBook-Pro:git rick$ git branch
* master
Richard-Hoods-MacBook-Pro:git rick$
Richard-Hoods-MacBook-Pro:git rick$ git branch devel
Richard-Hoods-MacBook-Pro:git rick$ git branch
  devel
* master
Richard-Hoods-MacBook-Pro:git rick$
```

Now I have a new branch called devel, but I am still on the master branch.

```
Richard-Hoods-MacBook-Pro:git rick$ git branch
  devel
* master
Richard-Hoods-MacBook-Pro:git rick$
Richard-Hoods-MacBook-Pro:git rick$ git checkout devel
Switched to branch 'devel'
Richard-Hoods-MacBook-Pro:git rick$ git branch
* devel
  master
Richard-Hoods-MacBook-Pro:git rick$
```

Now I have switched to my devel branch so I can work with hello.txt without worrying about messing up the version of hello.txt that is in the master branch.

```
Richard-Hoods-MacBook-Pro:git rick$ git branch
* devel
  master
Richard-Hoods-MacBook-Pro:git rick$ git checkout master
Switched to branch 'master'
Richard-Hoods-MacBook-Pro:git rick$ git merge devel
Updating 3bbadf3..6fddb31
Fast-forward
hello.txt |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)
Richard-Hoods-MacBook-Pro:local rick$
```

Above I switch back to the master branch, then merge in my devel branch so that now master will have the changes that I did to hello.txt when I was working on the devel branch.

I will walk you through this on the actual command line.

questions?

The workflow you will do over and over again is this:

A. Do some work on your files.
B. Do git status to see what git says has changed (optional)
C. Do git add .
D. Do git commit -m "my commit message"

But so far this is all dealing only with a local repo. Once you are dealing with a remote repo (more in a bit), you will also do this:

E. Push to the remote repo: git push origin devel

That pushes your local devel branch to the remote devel branch. Then others can get to your changes of devel by doing git pull origin devel

But we don't have a remote repo....

.... how do we get one?

But we don't have a remote repo....

.... how do we get one?

A. Somebody tells you where one is set up and you go get it with the clone command like this:

git clone git://git.kernel.org/pub/scm/git/git.git

But we don't have a remote repo....

.... how do we get one?

A. Somebody tells you where one is set up and you go get it with the clone command like this:
git clone git://git.kernel.org/pub/scm/git/git.git

B. You set up a repo on a git server someplace, like github.com. You associate your local repo with a remote repo with a remote add command something like this:
$ git remote add origin git@github.com:username/myrepo.git

But we don't have a remote repo....

.... how do we get one?

A. Somebody tells you where one is set up and you go get it with the clone command like this:
git clone git://git.kernel.org/pub/scm/git/git.git

B. You set up a repo on a git server someplace, like github.com. You associate your local repo with a remote repo with a remote add command something like this:
$ git remote add origin git@github.com:username/myrepo.git

Very good instructions for all of this is here:

http://help.github.com/create-a-repo/

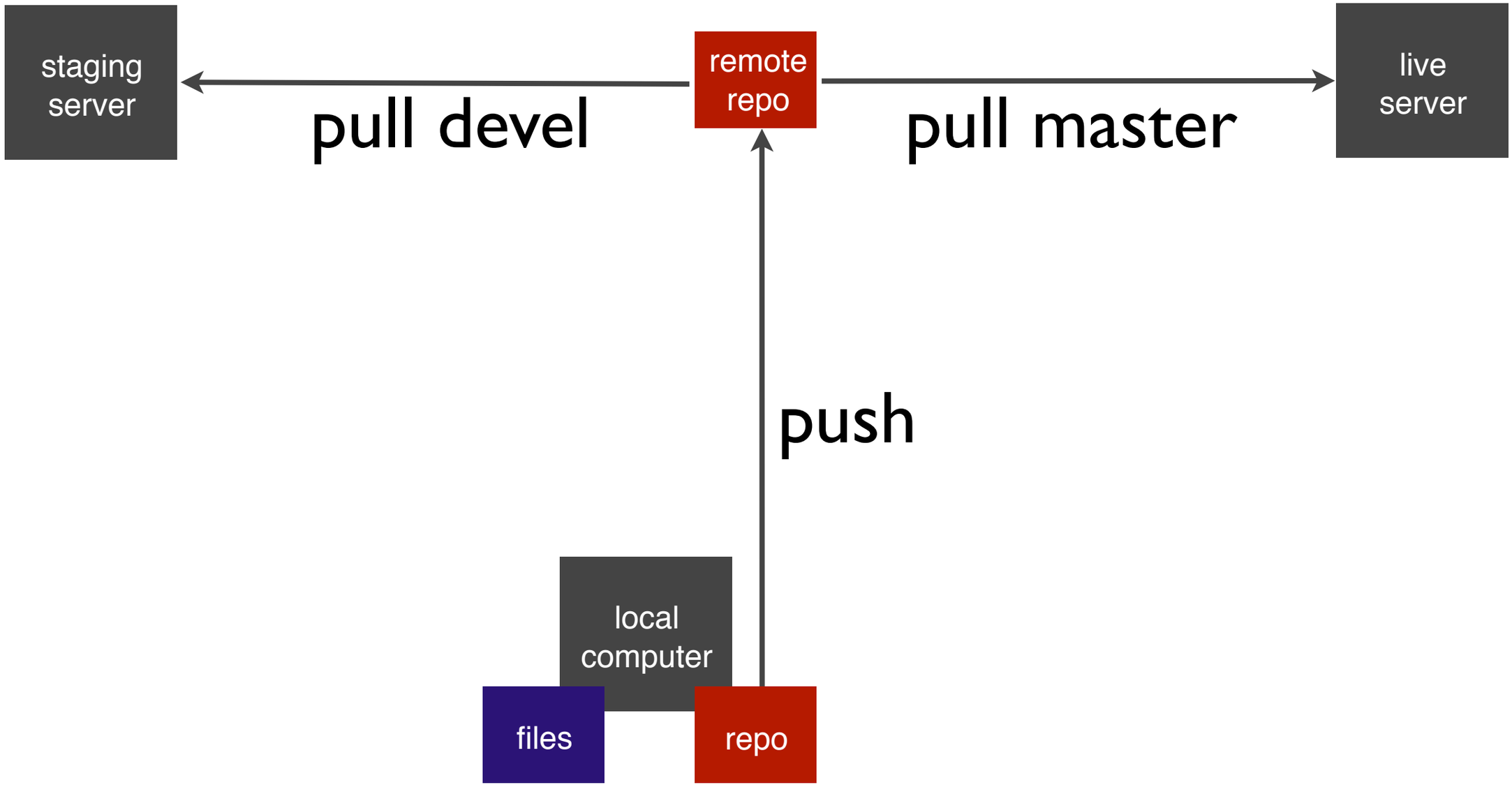
At github.com when you add a new repo, it comes back with instructions:

Next steps:

```
mkdir drupalcamp
cd drupalcamp
git init
touch README
git add README
git commit -m 'first commit'
git remote add origin git@github.com:rick02840/drupalcamp.git
git push -u origin master
```

Existing Git Repo?

```
cd existing_git_repo
git remote add origin git@github.com:rick02840/drupalcamp.git
git push -u origin master
```



pull devel to staging and master to live

once devel is merged into master, files are the same

at which point staging and live are the same if you have pulled to both

questions?

Now we will move on to a demo

But here again are key links for more info:

http://git-scm.com/

http://thinkvitamin.com/code/how-to-install-git/
(http://bit.ly/gitinstall)

http://gitref.org

http://help.github.com/create-a-repo/