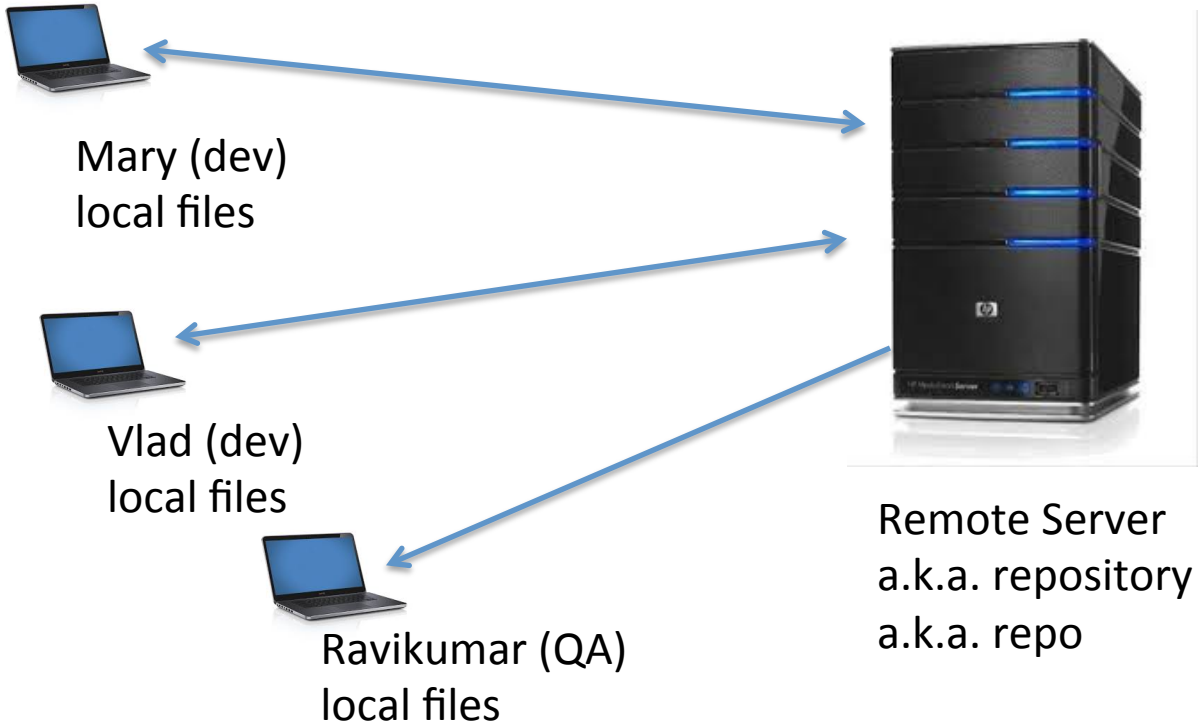
Java Programming
Unit 19

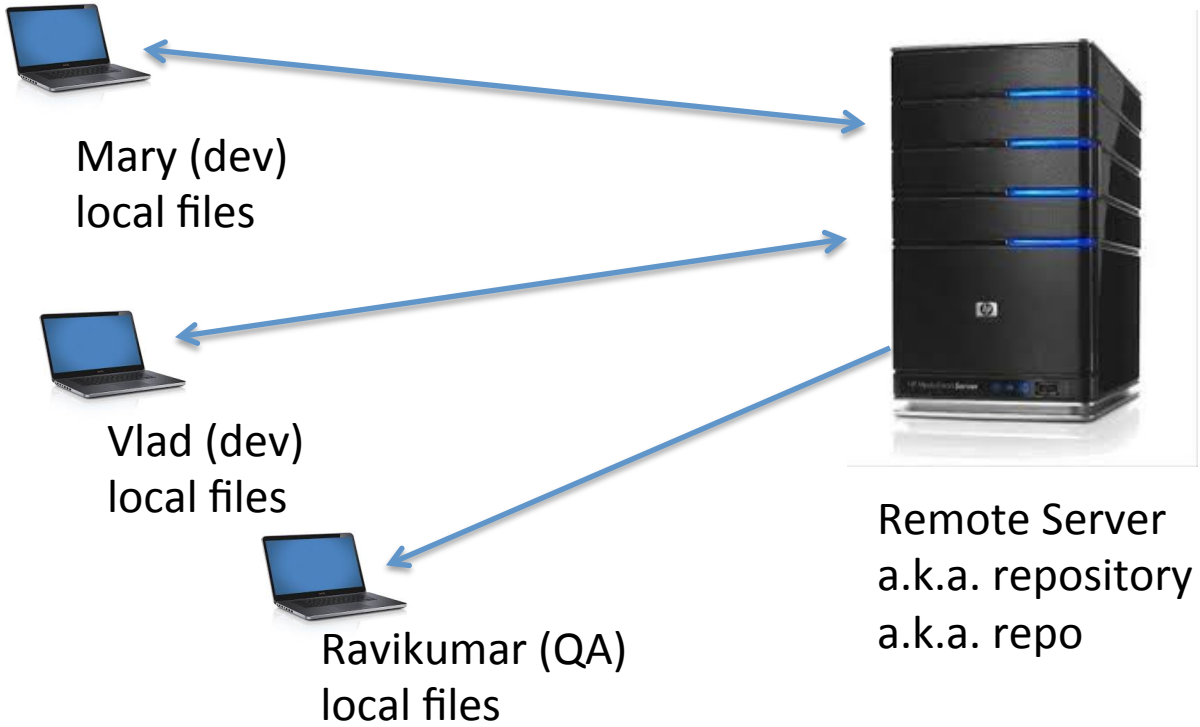
Intro to the version control system GIT

Centralized file repositories



- SVN
- CVS
- TFS
- Perforce
- …

Centralized file repositories



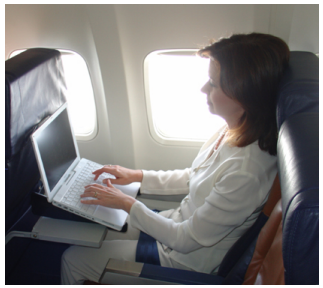
- SVN
- CVS
- TFS
- Perforce
- …

Mary commits her code changes to the remote server and checks out Vlad's changes

Vlad commits his code changes to the remote server and checks out Mary's changes

Ravikumar only checks out the code for the QA testing.

Distributed file repositories



Mary (dev), local files, local repo

Vlad (dev), local files, local repo

Remote Repo

Ravikumar (QA), local repo

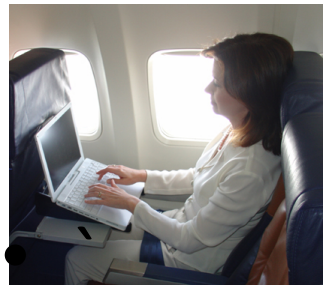
- GIT
- Mercurial
- Bazaar

Each developer has a full local copy of the repo.

It's hard to lose the repository – it's on every user's computer.

Each user always works with the local repo until he needs to synchronize.

Working with GIT



Mary (dev), local files, local repo

Vlad (dev), local files, local repo

pull/push

Remote Git Repo,
like GitHub

pull

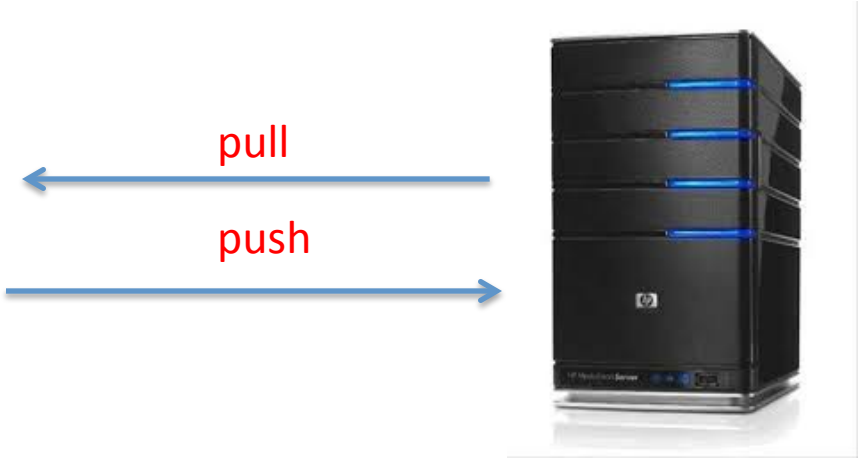
Ravikumar (QA), local repo

Mary works on the plane and **commits** her code changes to the local repo

Vlad **commits** his code changes to the local repo and **pushes** latest changes to the remote repo)

Ravikumar works for QA. He **pulls** out the code for testing.

Sync up with the remote server



Remote Git repository

Mary landed. She goes to Starbucks to get online to pull Vlad's changes to her local repo and push hers changes to the remote one.

git add

git reset

git commit

Commits are local

Paying for groceries
Is a commitment. But still,
you have a chance to return them

git reset –hard HEAD~1

git push

Pushes are usually remote

git pull

Pulls are usually done from a remote repository, which could be a central server or another user's ~~refrigerator~~ computer.

Downloads and documentation

- http://git-scm.com Git
- https://help.github.com GitHub repo
- http://git-scm.com/book Pro Git book in English
- http://git-scm.com/book/ru Pro Git book in Russian
- http://gitref.org Reference manual

MAC users, if you already had an older version of git installed, after installing the new version add this to ~/.bash_profile:

```
export PATH=/usr/local/git/bin:$PATH
```

Setting up name and email

- git config --global user.name ”Yakov Fain"
- git config --global user.email "yakovfain@gmail.com"

Two Ways of Creating a New Git Repo

1. **From scratch**

   a) cd to a directory where your project files are

   b) run git init (this will create .git subdirectory)

2. **Clone** the existing repo from somewhere, e.g. from GitHub, for example:

   git clone git://github.com/yfain/javatraining.git

Walkthrough 1: **git init**

1. Go to a Terminal window

2. Create an empty dir temp1:
   mkdir temp1

3. cd temp1

4. Directory is empty
   ls –la

5. Create a local git repo
   git init

6. Note a new subdir .git
   ls –la

7. See the structure of the new repo
   tree –a

There is no *logs* directory yet.
The *objects* directrory is empty.
This is your entire new Git repo.

temp1 — bash — 84×37

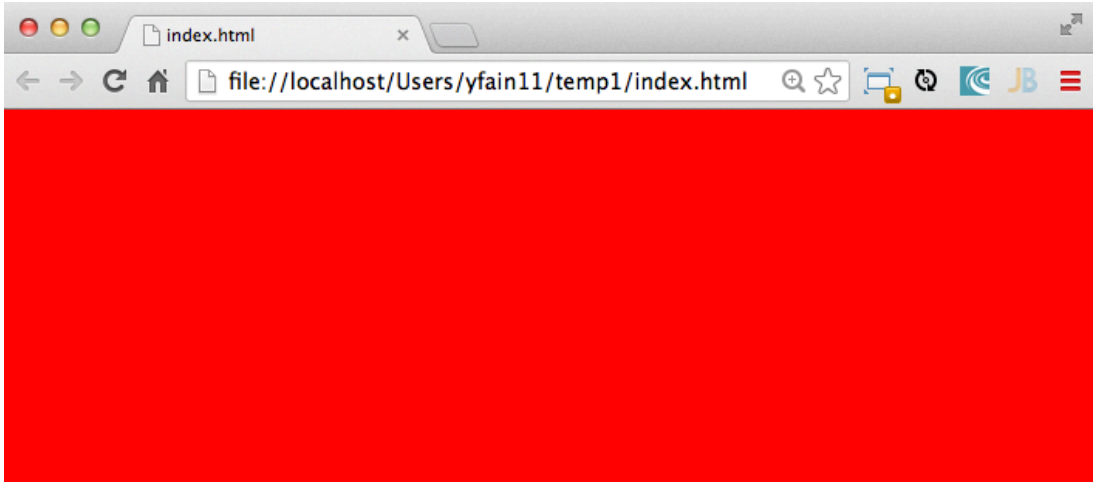
```
Yakov:temp1 yfain11$ git --version
git version 1.8.3.2
Yakov:temp1 yfain11$ git init
Initialized empty Git repository in /Users/yfain11/temp1/.git/
Yakov:temp1 yfain11$ ls -la
total 0
drwxr-xr-x    3 yfain11  staff   102 Sep 28 09:17 .
drwxr-xr-x  107 yfain11  staff  3638 Sep 28 09:17 ..
drwxr-xr-x   10 yfain11  staff   340 Sep 28 09:17 .git
Yakov:temp1 yfain11$ tree -a
.
└── .git
    ├── HEAD
    ├── branches
    ├── config
    ├── description
    ├── hooks
    │   ├── applypatch-msg.sample
    │   ├── commit-msg.sample
    │   ├── post-update.sample
    │   ├── pre-applypatch.sample
    │   ├── pre-commit.sample
    │   ├── pre-push.sample
    │   ├── pre-rebase.sample
    │   ├── prepare-commit-msg.sample
    │   └── update.sample
    ├── info
    │   └── exclude
    ├── objects
    │   ├── info
    │   └── pack
    └── refs
        ├── heads
        └── tags

10 directories, 13 files
Yakov:temp1 yfain11$
```

Creating a file and committing to a repo

1. Create index.html:

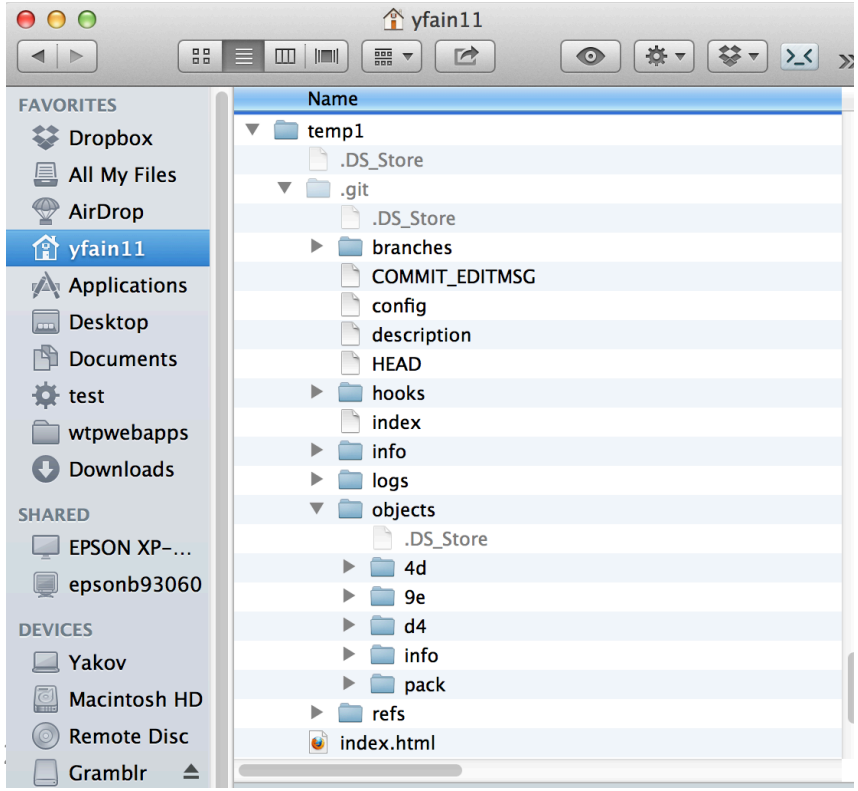
```
<html>
  <body bgcolor="red">
  </body>
</html>
```

2. Open it in your browser – it's red



3. git status
4. git add .
5. git commit –m "created red page"

6. Go to your .git directory.
   a) You'll see new log subdir
   b) Look inside your objects dir



Branching

- The default branch is master
  git branch
- To create a branch named blue
  git branch blue
- To create and switch to branch blue:
  git checkout –b blue

```
Yakov:temp1 yfain11$ git branch
* master
Yakov:temp1 yfain11$ git checkout -b blue
Switched to a new branch 'blue'
Yakov:temp1 yfain11$
```

- To switch back to master:
  git checkout master

- To merge blue into master:
  git merge blue

```
Yakov:heads yfain11$ pwd
/Users/yfain11/temp1/.git/refs/heads
Yakov:heads yfain11$ ls
master
Yakov:heads yfain11$
```

```
Yakov:heads yfain11$ pwd
/Users/yfain11/temp1/.git/refs/heads
Yakov:heads yfain11$ ls
blue    master
Yakov:heads yfain11$
```

Walkthrough 2: Adding Another Branch

1. Check which branch are you in (should be in master)
   git branch

2. Create a new branch named blue and switch to it:
   git checkout –b blue

3. Change the background in index.html to be blue:
   `<body bgcolor="blue">`
   Open index.html in the browser – background should be blue

4. Commit the change
   git add .
   git commit –m "changing background from red to blue"

5. Switch back to the branch master
   git checkout master

6. Open index.html in your browser – background should be red.
   Check your working directory – there is only one version of index.html.

Walkthrough 3: One more branch + merging

1. Create and switch one more branch called branch2
   git checkout –b branch2

2. Add some text inside the <body> tag in index.html:
   <h1>Hello from branch2</h1>

   Open index.html in the Web browser

3. Commit the change:
   git add .
   git commit –m “added some text to the page”

4. Switch back to master branch
   git checkout master

   Open index.html – it’s red and no text

5. Merge with branch2
   git merge branch2
   Open index.html – it’s red with text

Walkthrough 3: Merge conflict

Merging master with blue will cause the conflict.
Master branch has
<body bgcolor="red">
Blue branch has
<body bgcolor="red”>

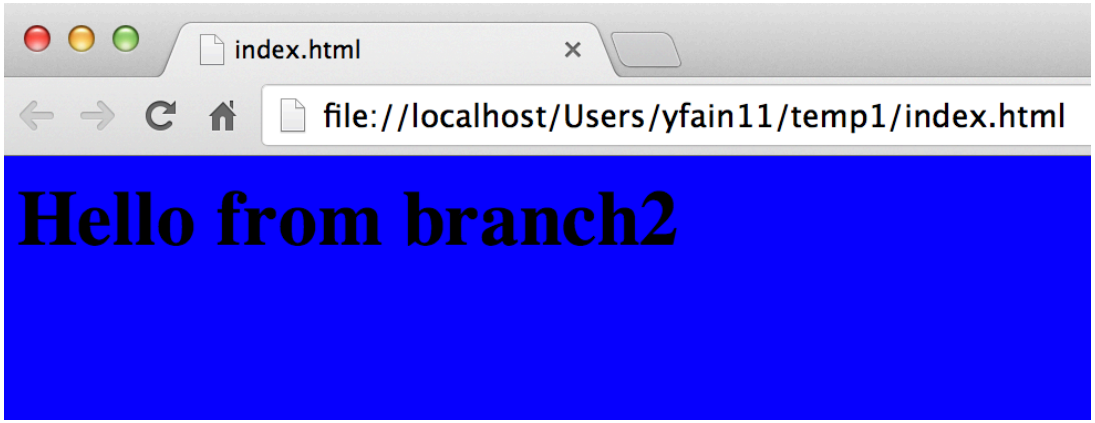
Open index.html in the editor and manually change it to keep the line with blue color.

The final version of index.html:

```
<html>
 <body bgcolor="blue">
   <h1>Hello from branch2</h1>
 </body>
</html>
```

Do git add . and git commit

```
Yakov:temp1 yfain11$ git branch
  blue
  branch2
* master
Yakov:temp1 yfain11$ git merge blue
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
Yakov:temp1 yfain11$ cat index.html
<html>
<<<<<<< HEAD
  <body bgcolor="red">
      <h1>Hello from branch2</h1>

=======
  <body bgcolor="blue">

>>>>>>> blue
  </body>
</html>Yakov:temp1 yfain11$
```



Git config files

Global Git Config:

~/.gitconfig

Local Git Config:

your_project_dir/.git/config

GitHub – Online project hosting using Git

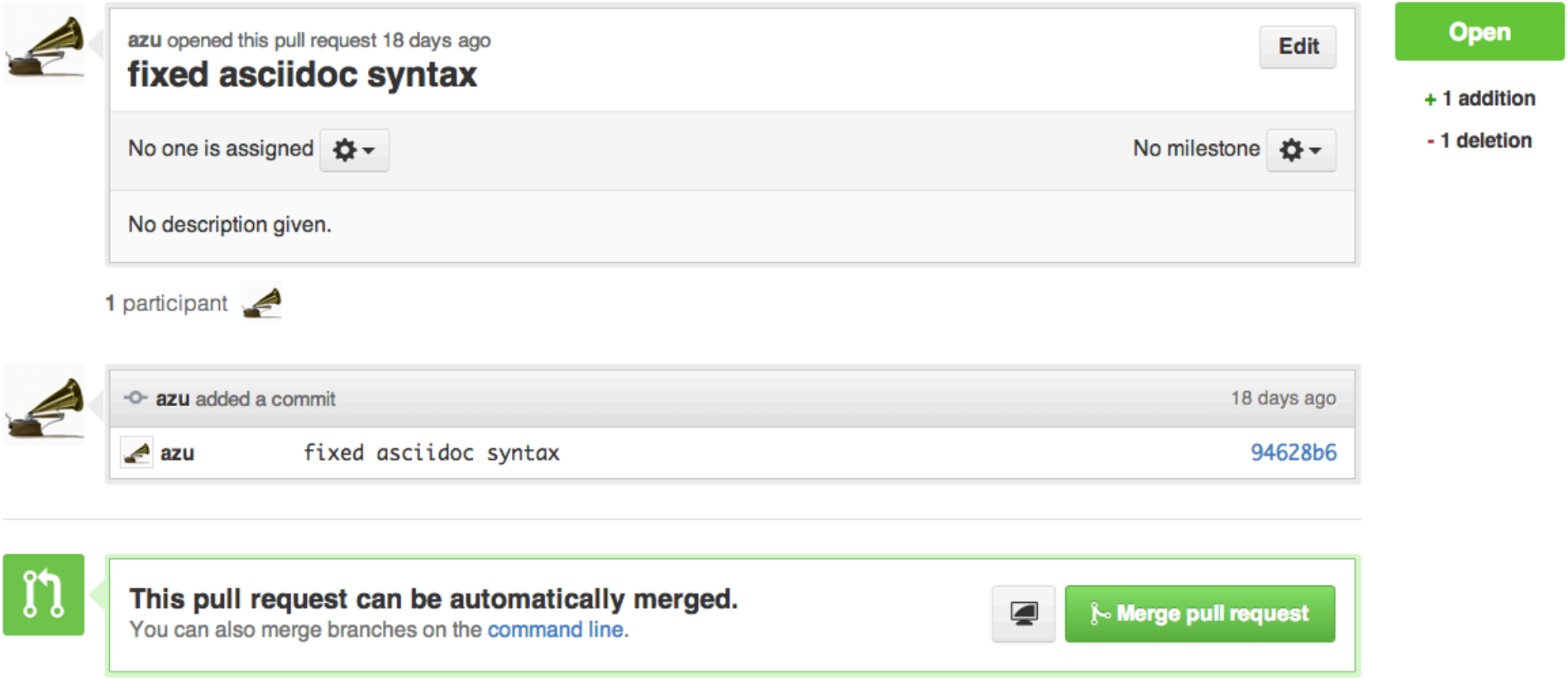
To start using GitHub:

- Create an organization
- Add users to the organization
- Push your local repository to GitHub

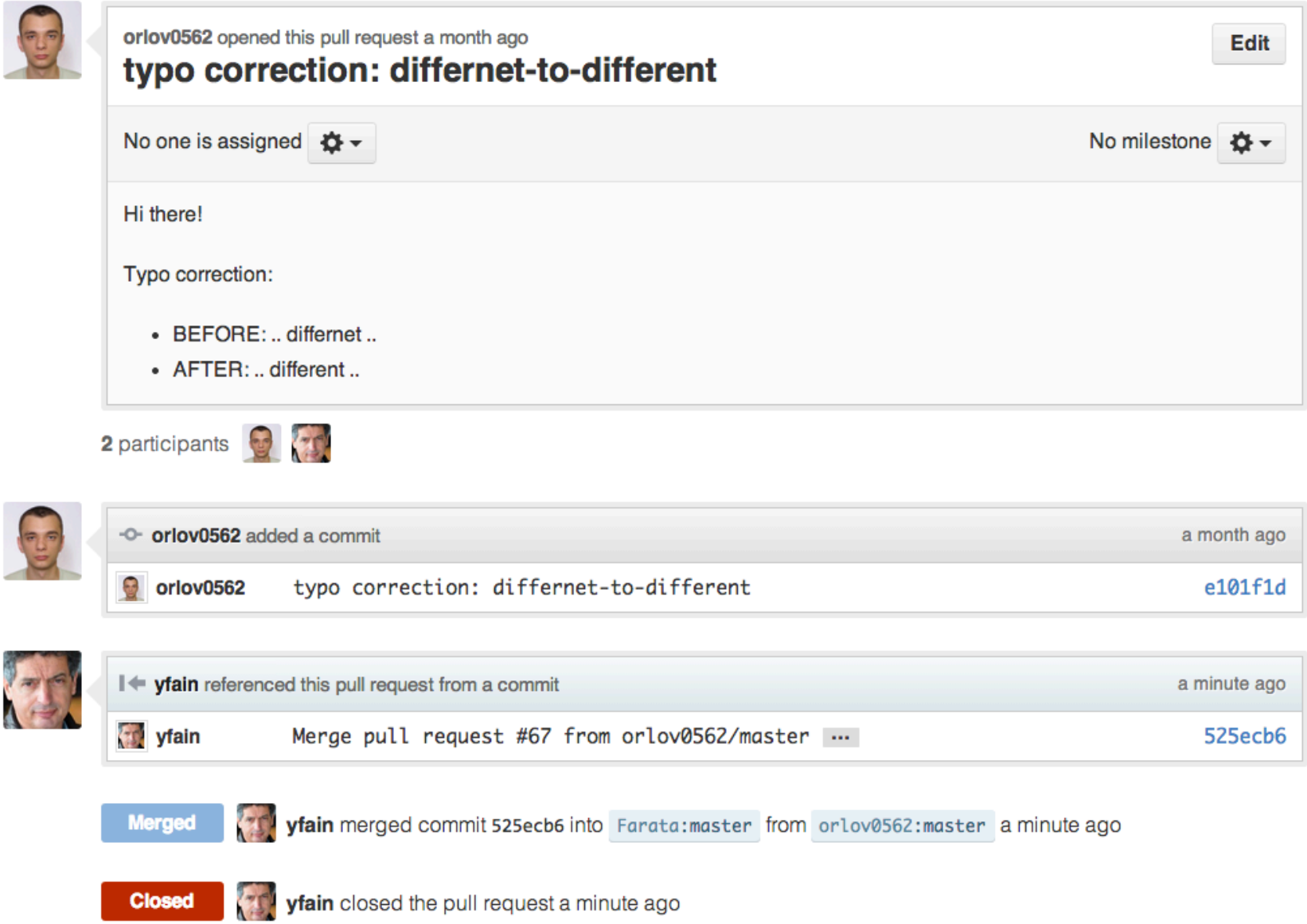
GitHub: Pull Request

- The user VasyaF, who doesn't belong to our project can fork the proj, make the changes and send a Pull Request.
- Admin of our project sees the pull request and agrees to pull the change from VasyaF's repo to ours – he presses Merge Pull Request.

Merging Pull Request Sample 1



Merging Pull Request Sample 2



Homework

- Create a new local repository called JavaTraining.

- Move any of your homeworks to be subdirectories of JavaTraining, for example JavaTraining/Lesson2

- Create a new repository on GitHub.com
  (watch this video for help: http://www.youtube.com/watch?v=TPY8UwlTIc0)

- You'll need to add new remote server to your git configurarion, for example:

  git remote add origin https://github.com/your_user_i/repo.git

  For help read this: https://help.github.com/articles/adding-a-remote

- **Push** your JavaTraining repository there.
  git push origin master
- Send the URL of your repository to the instructor.

Important! The email you use at GitHub should be the same as you used while configuring local git.