INF333 2023-2024 Spring Semester

Burak Arslan

Lab VII

Mar 29th, 2024
Due: Mar 29th, 10:59:59

1 A Not-so-gentle introduction to `git` (100pts)

Create a local copy of the course pintos repository:

```
$ git clone https://burakarslan.com/git/pintos.git
```

Note that you have only one remote defined.

```
$ git remote -v
```

Clone operation is a shortcut for the following operations:

```
$ mkdir pintos
$ cd pintos
$ git init
$ git remote add origin https://burakarslan.com/git/pintos.git
$ git fetch origin
$ git checkout -b master -t origin/master
```

You don't have push privileges to this repository. So you need to create a new repository that comes with write permissions for your team. Let's say you are using github and your private repository address is:

```
https://github.com/sirinler/pintos
```

Assuming you are using SSH authentication, the ssh equivalent of this address is:

```
git@github.com:sirinler/pintos
```

To push your local clone to the remote repository, you must first add it as a remote:

```
$ git remote add github git@github.com:sirinler/pintos
```

Then push to your new remote:

```
$ git push github master
```

Q1: Explain what this command does in humanese.

Q2: How do you ensure that "git push" pushes to the remote named github by default?

When there are modifications in the origin remote, you first need to fetch the changes:

```
$ git fetch origin
```

Now, origin/master contains the new changes. Let's integrate them to your local checkout.

There are 3 ways to do it: rebase, merge, cherry pick.

1. Merge creates a new commit with multiple parents.

   ```
   $ git merge origin/master
   $ git push
   ```

2. Rebase puts the non-common patches in the current branch on top of the given branch.

   ```
   $ git rebase origin/master
   $ git push -f
   ```

3. Cherry-pick applies only the given patch. If there are multiple patches, you need to cherry-pick them one by one. Rebase automates this process.

   So let's imagine there are 4 new patches downloaded to the remote repository:

```
$ git fetch origin
$ git cherry-pick origin/master~3
$ git cherry-pick origin/master~2
$ git cherry-pick origin/master~1
$ git cherry-pick origin/master
```

and

```
$ git fetch origin
$ git rebase origin/master
```

do the same thing.

Q3: What does the “branch n“ notation mean?

Q4: What is the difference between git push and git push -f? Why does git merge works with git push, but git rebase requires git push -f?

Read more about rebase vs merge vs cherry-pick:
https://blog.devgenius.io/git-merge-vs-rebase-vs-cherry-pick-a6e483d886b9