GitHub and command line

www.romanzolotarev.com/github/

Why command line? Okay, GitHub has a clean and straightforward website, but while I stay in Terminal all day long, it is much easier to write a function and place it into my `.bashrc` than switching to a web browser and clicking buttons.

Basic setup

First of all, make sure you have enabled two-factor authentication for your GitHub account.

Then generate SSH keys and add your public key to your profile on GitHub to be able to connect to GitHub via SSH. Test it right away.

```
ssh -T
[email protected]
```

By the way, did you know GitHub makes your public key available via HTTPS? Having your keys accessible is very convenient, for example, when you are on one of your remote servers.

```
touch ~/.ssh/authorized_keys
curl https://github.com/USERNAME.keys >>
~/.ssh/authorized_keys
```

hub

If you work in Terminal most of the time, install hub. It is a CLI tool made by GitHub. It makes it easier to use GitHub from the command line: create repositories, create pull requests, compare branches, etc.

To install it on macOS with Homebrew run:

```
brew install
hub
```

As soon as you have got `hub` you can do more things from the command line.

```
# Short repository names
hub clone dotfiles
# git clone git://github.com/YOUR_USERNAME/dotfiles.git

hub clone romanzolotarev/dotfiles
# git clone
git://github.com/romanzolotarev/dotfiles.git

# Open the current repository in web browser
hub browse

# Open the current repository's issues page
hub browse -- issues

# List the current repository's issues
hub issue

# Open a text editor for your pull request message
hub pull-request

# Open compare view between two releases
hub compare v0.9..v1.0

# Create a repository with the name of current
directory
hub create
```

API

Explore the documentation.

To parse JSON from API responses, I suggest you install jq.

```
brew install
jq
```

Let's play with `/user/keys` endpoint to see what is possible with GitHub API. For this endpoint you need your personal access token. Get one on your profile page or via command line. To make further examples work your token should be in `admin:public_key` scope.

Get access token

Replace `USERNAME`, `OTP_CODE`, `NOTE` with actual values:

```
curl -s https://api.github.com/authorizations \
  -H 'X-GitHub-OTP: OTP_CODE' \
  -u USERNAME \
  -d '{"scopes": ["admin:public_key"], "note": "NOTE"}'
\
  | jq -r .token
```

In response you will your personal access token. Treat it carefully as a password.

List your public keys

For example, you can check your public keys via API now. Replace `ACCESS_TOKEN` with your personal access token.

```
curl -s https://api.github.com/user/keys
\
  -H 'Authorization: token ACCESS_TOKEN'
\
  | jq
```

Add your public key

You can even add new public keys. Replace `ACCESS_TOKEN`, `TITLE`, `PUBLIC_KEY` with actual values, where `PUBLIC_KEY` is a path to your public SSH key.

```
curl -s https://api.github.com/user/keys \
  -H 'Authorization: token ACCESS_TOKEN' \
  -d "{\"title\": \"TITLE\", \"key\": \"$(cat PUBLIC_KEY)\"}"
\
  | jq
```

Delete your public key

To delete a key you need to get its `ID` at GitHub. Do not forget to replace `ACCESS_TOKEN` and `ID` with actual values.

```
curl -X 'DELETE' https://api.github.com/user/keys/ID
\
  -H 'Authorization: token ACCESS_TOKEN'
```

Happy curling.