How to contribute code to MCCE

MCCE Development Through Github

Part 1. One-time set up

On your workstation / coding computers, set up the git environment.

Identity:

```
$ git config --global user.name "<Your Name>"
$ git config --global user.email <your email>
```

Save credential for 6 hours:

```
$ git config --global credential.helper 'cache --timeout=21600'
```

Save credential permanently:

```
$ git config --global credential.helper store
```

Alias of showing git history:

Add this to ~/.gitconfig

```
[alias]
lg = log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) -
%C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold
yellow)%d%C(reset)' --all
```

Github access token

Github adopts personal access token to replace password authentication. In short, you need to use a personal access token as password to push commits to github. See documentation

https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

Be aware, once the token is created, you will not be able to see the token content any more. So save the token in a safe place. If you have successully pushed a commit to github and used above method to save credential permenently, the token is stored in ~/.git-credentials.

Part 2. Work with your own fork of MCCE project

MCCE project at https://github.com/GunnerLab/Stable-MCCE hosts the current stable code of MCCE. If you need to edit or add code, the recommended work flow is like this:

1. Log in github and go to https://github.com/GunnerLab/Stable-MCCE, create a fork the repository so you have your own Stable-MCCE repository. This is a one-time set up.
2. All the local code developing should be performed in a develope branch and pushed to the forked Stable-MCCE repo on github.
3. Once the code is done and tested, create a pull request from the forked Stable-MCCE repo on github.
4. After receiving confirmation that your pull request had been merged to the master repo, you can pull from the master repo to update your code and safely delete the develope branches.

Part 3. Examples of Git commands

Clone a remote repo:

Clone a remote repo to local computer:

```
git clone <url>
```

Convert a local directory:

Convert an existing local directory to local git repository:

```
git init
```

Add all files to track:

```
git add .
```

Add a single file to track:

```
git add <file name>
```

Check status:

```
git status
```

Commit:

```
git commit -a -m "<commit message>"
```

Show commit history:

```
git lg
```

It requires you set up the "lg" alias in ~/.gitconfig

Part 4. More on git branches

Create a new branch and revert to a past commit:

```
git checkout -b <new branch> <commit hash>
```

<new branch> is a branch name you make.

<commit hash> is the string as reported by `git log`

If you want to make this branch as the new master branch, do a swap as following:

1. make sure your are in the new branch

   ```
   $ git checkout <new branch>
   ```

2. force master to merge with current branch and use current branch as favored:

   ```
   $ git merge -s ours master
   ```

3. go to the master branch and reconcile again:

```
$ git checkout master
$ git merge <new branch>
```

4. after merge, delete the branch.

```
$ git branch -d <new branch>
```

Show branches:

`git branch`

Show remote branches:

`git branch -r`

Switch between branches:

`git checkout <branch name>`

Create and switch branch:

`git checkout -b <branch name>`

Merge branches:

`git merge <another branch>`

This will merge a branch to current branch:

Merge with current favored:

`git merge -s ours <another branch>`

Delete branches:

`git branch -d <branch name>`

Force delete:

`git branch -D <branch name>`

Part 5. Sync remote and local repository

Check remote repo:

`git remote -v`

"origin" is the default name of your first remote.

Add more remotes:

`git remote add <remote> <url>`

Pull from remote repo:

`git pull`

Or

`git pull <remote> <branch>`

Push to remote:

`git push`

Push new local branch to remote:

`git push -u <remote> <branch>`

Delete remote branch:

`git push <remote> --delete <branch>`

Delete remote tracking branch:

`git remote prune <remote>`

Often used on remote repo origin:

`git remote prune origin`

This command is useful to clean up the remote repository.

Part 6. Merge branches

Common scenario of merge:

- Start a new feature:

```
$ git checkout -b new-feature
```

- Edit some files, then commit the change:

```
$ git commit -a -m "Start a feature"
```

- Edit some files, then commit more changes:

```
$ git commit -a -m "Finish a feature"
```

- Merge in the new-feature branch:

```
$ git checkout master
$ git merge new-feature
$ git branch -d new-feature
```

Conflict in merge:

When conflicts occur, the conflicting files will have visual marks like:

```
<<<<<<< master
conflicting text in receiving branch
=======
conflicting text in merging branch
>>>>>>> branch
```

You need to edit text and remove <<<<<<, ======, >>>>>> lines.

Then run a commit:

```
$ git commit -a -m "<commit message>"
```

Revision #2
Created 8 March 2023 19:25:01 by Junjun Mao
Updated 24 March 2023 21:09:20 by Junjun Mao