# One-time setup

**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:**

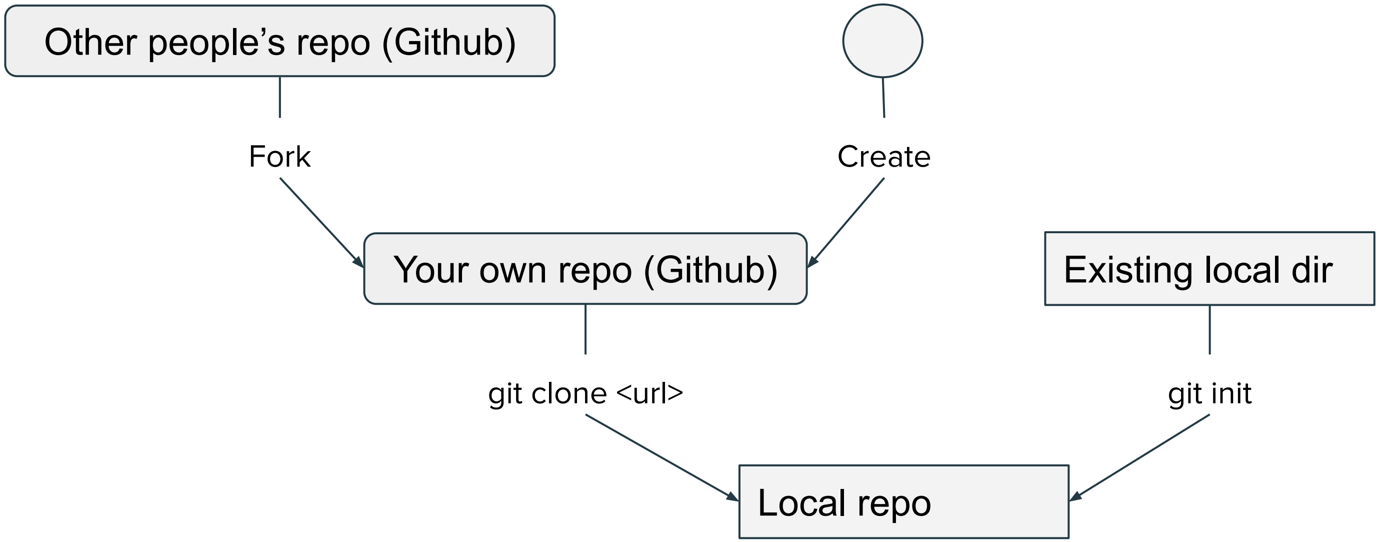
```
git config --global credential.helper store
```

**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
```

# Start a new repository



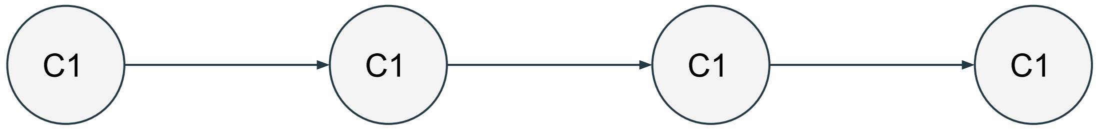
**Clone a remote repo to local computer:**

```
git clone <url>
```

**Convert a existing local directory to local git repository:**

```
git init
```

## Single line development



**Add all files to track:**

```
git add .
```

**Add a file to track:**

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

**Check status:**

```
git status
```

**Show history:**

```
git lg
```

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

**Commit:**

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

## Go to a past commit

**Create a new branch and revert to a past commit:**

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

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>
   ```

# Multi-line development

**Show branches:**

```
git branch
```

**Show remote branches:**

```
git branch -r
```

**Switch between branches:**

```
git checkout <branch name>
```

**Create and switch to a new branch:**

```
git checkout -b <branch name>
```

**Merge branches:**

```
git merge <another branch>
```

This will merge a branch to current branch:

**Merge automatically with current branch favored:**

```
git merge -s ours <another branch>
```

**Delete branches**

```
git branch -d <branch name>
```

**Force delete:**

```
git branch -D <branch name>
```

# Synchronize remote and local repositories

**Check remote:**

```
git remote -v
```

“origin” is the default name of your first remote. You can add more remotes:

```
git remote add <remote> <url>
```

**Pull from remote repo:**

```
git pull
```

Or

```
git pull <remote> <branch>
```

**Push to remote repo:**

```
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>
```

# Merge branches

**Common scenario of merge:**

Start a new feature:

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

Edit some files:

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

Edit some files:

```
    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>"
```