GitLab GIT CHEAT SHEET

5. REVIEW YOUR WORK

```
$ git log [-n count]
```

List commit history of current branch. **-n count** limits list to last **n** commits.

```
$ git log --oneline --graph --decorate
```

An overview with references labels and history graph. One commit per line.

```
$ git log ref..
```

List commits that are present on current branch and not merged into **ref**. A **ref** can be e.g. a branch name or a tag name.

```
$ git log ..ref
```

List commit, that are present on **ref** and not merged into current branch.

```
$ git reflog
```

List operations (like checkouts, commits etc.) made on local repository.

8. SYNCHRONIZING REPOSITORIES

```
$ git fetch [remote]
```

Fetch changes from the **remote**, but not update tracking branches.

```
$ git fetch --prune [remote]
```

Remove remote refs, that were removed from the **remote** repository.

```
$ git pull [remote]
```

Fetch changes from the **remote** and merge current branch with its upstream.

```
$ git push [--tags] [remote]
```

Push local changes to the **remote**. Use **--tags** to push tags.

```
$ git push -u [remote] [branch]
```

Push local branch to **remote** repository. Set its copy as an upstream.

6. TAGGING KNOWN COMMITS

```
$ git tag
```

List all tags.

```
$ git tag [name] [commit sha]
```

Create a tag reference named **name** for current commit. Add **commit sha** to tag a specific commit instead of current one.

```
$ git tag -a [name] [commit sha]
```

Create a tag object named **name** for current commit.

```
$ git tag -d [name]
```

Remove a tag from a local repository.

7. REVERTING CHANGES

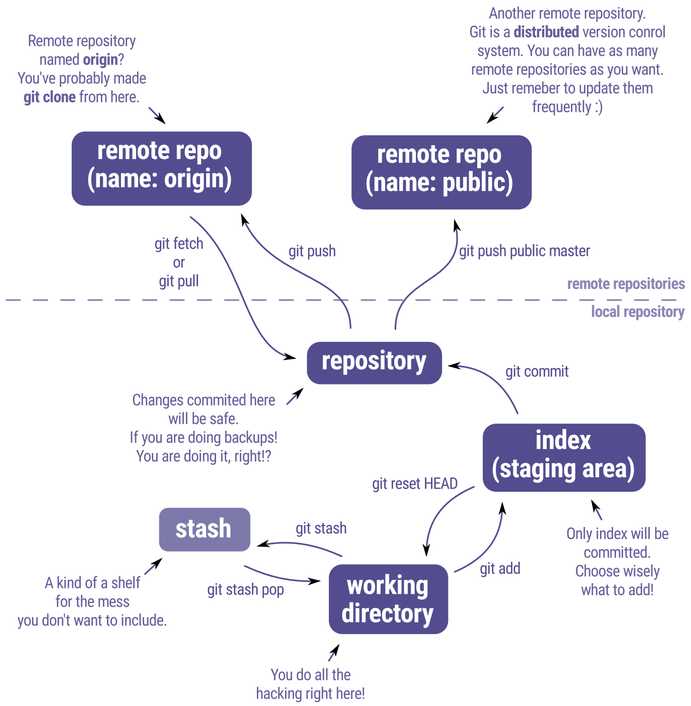
```
$ git reset [--hard] [target reference]
```

Switch current branch to the **target reference**, and leaves a difference as an uncommited changes. When **--hard** is used, all changes are discarded.

```
$ git revert [commit sha]
```

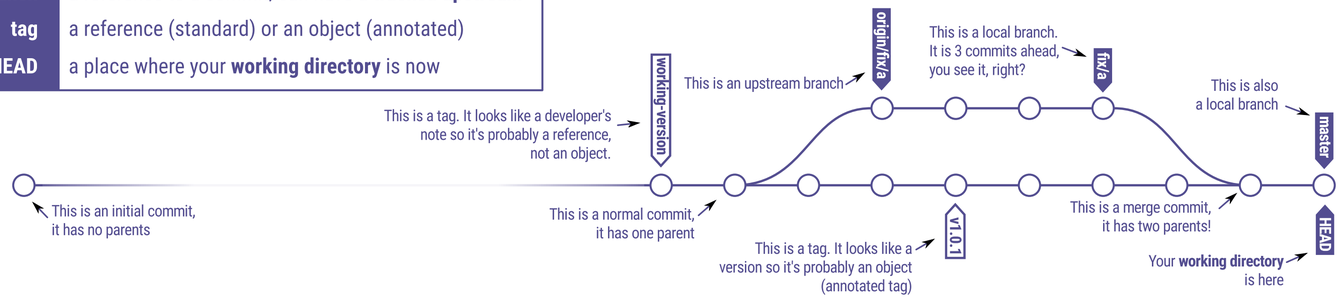
Create a new commit, reverting changes from the specified commit. It generates an **inversion** of changes.

C. THE ZOO OF WORKING AREAS



commit	an object
branch	a reference to a commit; can have a **tracked upstream**
tag	a reference (standard) or an object (annotated)
HEAD	a place where your **working directory** is now

D. COMMITS, BRANCHES AND TAGS



*And this is the past. Here was chaos, where no **version control** was used. Don't live in chaos!*
Use Git!

Collaborating on GitLab