git ATLAS Cheat Sheet

Getting started

Cloning your fork of the **entire** repository to a local disk (not on AFS, please!):
```
git clone https://:@gitlab.cern.ch:8443/YOUR_USER_NAME/athena.git
```
Add the main repository as upstream:
```
git remote add upstream https://:@gitlab.cern.ch:8443/atlas/athena.git
```

Alternative: Set up a **sparse**-checkout workdir:
```
git atlas init-workdir https://@gitlab.cern.ch:8443/YOUR_USER_NAME/athena.git
```
Add / List / Remove packages in your dir:
```
git atlas addpkg <package> / git atlas lspkg / git atlas rmpkg <package>
```

Start a new mini-project

Bring you local clone up-to-date with the main repository
```
git fetch upstream
```
Create a branch for your changes based on another branch (like 21.0 or master)
```
git checkout –b MY_BRANCH_NAME upstream/parent_branch --no-track
```
Or switch to an existing branch:
```
git checkout MY_BRANCH_NAME
```

Inspection:

List remotes:
```
git remote –v show
```
List changed files in the work-dir:
```
git status
```
Changes of tracked files not yet staged
```
git diff
```
Show (last) commit
```
git show [commit-id]
```
Show branches (verbose, all)
```
git branch [-v] [-av]
```
List tags:
```
git tag -l
```
Get previous commit messages:
```
git log
```

Local changes:

Add files for later commits
```
git add [file]
```
Undo the above:
```
git reset[file]
```
Store a snapshot added files:
```
git commit
```
Edit previous commit (before push):
```
git commit --ammend
```
Reverting a commit:
```
git revert [commit-id]
```
Discard local (uncommitted) changes
```
git checkout [path]
```

Merging:

Merge changes from *Branch_Name* into current branch
```
git merge BRANCH_NAME
```

Uploading changes:

Pushing your local work to your fork
```
git push origin –u MY_BRANCH_NAME
```

Notation

upstream: The official athena repository on gitlab the used to build nightlies and releases

origin: Your private fork of the official athena repository on gitlab **branch**: Either a release-series (like 21.0.X) or your feature branch

master: The development branch

commit id: a hash uniquely identifying the state of a repo

HEAD: The most recent commit id of the current branch

tag: A human-readable alias of a commit id