Git Cheat Sheet

Create
From existing data
cd~/my_project_dir \$ git init \$ git add .
From existing repo \$ git clone \(^{\preceq}\) existing/repo \(^{\preceq}\) new/repo \$ git clone \(^{\preceq}\) you\(^{\preceq}\) host.\(^{\preceq}\) deforted \$ git clone \(^{\preceq}\) host.\(^{\preceq}\) project.\(^{\preceq}\) git
Browse
Files changed in working directory
\$ git status
Changes of tracked files \$ git diff
Changes between ID1 and ID2 \$ git diff <id1> <id2></id2></id1>
History of changes \$ git log
Useful tips
Documentation/help \$ git help [command] \$ man git-[command]
Push branch to remote \$ git push <origin> <branch></branch></origin>
Delete branch (locally and remote) \$ git branch -d <branch> \$ git push <origin> :<branch></branch></origin></branch>
Update
Fetch latest changes from origin
\$ git fetch
see what changed \$ git remote update

Revert
Revert the last commit
\$ git revert HEAD
Return to last committed state
\$ git checkout -f git reset -hard
Revert specific commit \$ git revert <id></id>
git revert (ID)
fix/change last commit
\$ git commit ——amend
branch
List all branches (locally and remote)
\$ git branch -a
Switch to BRANCH
\$ git checkout BRANCH
Merge Branch B1 into B2 \$ git checkout <b2></b2>
\$ git merge <b1></b1>
Create Branch based on HEAD \$ git branch <branch></branch>
git branch (Dranch)
Create Branch based on HEAD and checkout
\$ git checkout -b <branch></branch>
Create Branch based on another
\$ git branch <new> <branch></branch></new>
Delete a branch
\$ git branch -d <branch></branch>
G "
Commit

View r
\$ git
View r
\$ git
View r
\$ git
3.7 •
View r
φ git (
After
\$ git
\$ git :
Push o
\$ git
Make a
φgit
Prepar
\$ git
J
\$ git
user
user.n
user.e
color
color.
ontire!
optimi packs

View merge conflicts \$ git diff View merge conflicts against base file \$ git diff -base <file> View merge conflicts against other changes \$ git difftheirs <file> View merge conflicts against your changes \$ git diffours <file> View merge conflicts against your changes \$ git diffours <file> After resolving conflicts, merge with \$ git add <conflicting_file> \$ git rebasecontinue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag <version_name> Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in 7.gitconfig user user.name NAME user.email EMAIL</version_name></conflicting_file></file></file></file></file>	
View merge conflicts against base file \$ git diff -base <file> View merge conflicts against other changes \$ git difftheirs <file> View merge conflicts against your changes \$ git diffours <file> After resolving conflicts, merge with \$ git add <conflicting_file> \$ git rebasecontinue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag <version_name> Prepare a patch \$ git format-patch origin \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in //.gitconfig user user.name NAME</version_name></conflicting_file></file></file></file>	Resolve merge conflicts
View merge conflicts against base file \$ git diff -base <file> View merge conflicts against other changes \$ git difftheirs <file> View merge conflicts against your changes \$ git diffours <file> After resolving conflicts, merge with \$ git add <conflicting_file> \$ git rebasecontinue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag <version_name> Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in //.gitconfig user user.name NAME</version_name></conflicting_file></file></file></file>	View merge conflicts
\$ git diff -base <file> View merge conflicts against other changes \$ git difftheirs <file> View merge conflicts against your changes \$ git diffours <file> After resolving conflicts, merge with \$ git add <conflicting_file> \$ git rebasecontinue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag <version_name> Prepare a patch \$ git format-patch origin \$ git format-patch origin \$ git config [global] [option] global is stored in \(\). gitconfig user user.name NAME</version_name></conflicting_file></file></file></file>	\$ git diff
\$ git diff -base <file> View merge conflicts against other changes \$ git difftheirs <file> View merge conflicts against your changes \$ git diffours <file> After resolving conflicts, merge with \$ git add <conflicting_file> \$ git rebasecontinue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag <version_name> Prepare a patch \$ git format-patch origin \$ git format-patch origin \$ git config [global] [option] global is stored in \(\). gitconfig user user.name NAME</version_name></conflicting_file></file></file></file>	
View merge conflicts against other changes \$ git diff — theirs < FILE> View merge conflicts against your changes \$ git diff — ours < FILE> After resolving conflicts, merge with \$ git add < CONFLICTING_FILE> \$ git rebase — continue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag < version_name> Prepare a patch \$ git format-patch origin \$ git format-patch origin \$ git config [— global] [option] global is stored in 7/.gitconfig user user.name NAME	View merge conflicts against base file
\$ git diff — theirs < FILE> View merge conflicts against your changes \$ git diff — ours < FILE> After resolving conflicts, merge with \$ git add < CONFLICTING_FILE> \$ git rebase — continue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag < version_name> Prepare a patch \$ git format-patch origin \$ git format-patch origin Configuration \$ git config [— global] [option] global is stored in 7/.gitconfig user user.name NAME	\$ git diff -base <file></file>
\$ git diff — theirs < FILE> View merge conflicts against your changes \$ git diff — ours < FILE> After resolving conflicts, merge with \$ git add < CONFLICTING_FILE> \$ git rebase — continue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag < version_name> Prepare a patch \$ git format-patch origin \$ git format-patch origin Configuration \$ git config [— global] [option] global is stored in 7/.gitconfig user user.name NAME	
View merge conflicts against your changes \$ git diff —ours <file> After resolving conflicts, merge with \$ git add <conflicting_file> \$ git rebase —ocontinue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag <version_name> Prepare a patch \$ git format-patch origin \$ git format-patch origin Configuration \$ git config [—global] [option] global is stored in \(\)-gitconfig user user.name NAME</version_name></conflicting_file></file>	
\$ git diffours <file> After resolving conflicts, merge with \$ git add <conflicting_file> \$ git rebasecontinue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag <version_name> Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in //.gitconfig user user.name NAME</version_name></conflicting_file></file>	\$ git diff ——theirs < FILE>
\$ git diff —-ours <file> After resolving conflicts, merge with \$ git add <conflicting_file> \$ git rebase —-continue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag <version_name> Prepare a patch \$ git format-patch origin Configuration \$ git config [—-global] [option] global is stored in \(\)-gitconfig user user.name NAME</version_name></conflicting_file></file>	View mongo conflicts against your shanges
After resolving conflicts, merge with \$ git add <conflicting_file> \$ git rebase ——continue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag <version_name> Prepare a patch \$ git format-patch origin Configuration \$ git config [——global] [option] global is stored in \(\)_gitconfig user user_name NAME</version_name></conflicting_file>	
\$ git add <conflicting_file> \$ git rebasecontinue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag <version_name> Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in \(\cdot \). gitconfig user user.name NAME</version_name></conflicting_file>	V S. C.
\$ git add <conflicting_file> \$ git rebasecontinue Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag <version_name> Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in \(\cdot \). gitconfig user user.name NAME</version_name></conflicting_file>	After resolving conflicts, merge with
Publish Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag < version_name > Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in \(\cdot \).gitconfig user user.name NAME	, , ,
Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag < version_name > Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in ~/.gitconfig user user.name NAME	\$ git rebase ——continue
Push changes to origin \$ git push [origin] [branch] Make a Version \$ git tag < version_name > Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in \(^{\text{-}}\).gitconfig user user.name NAME	
\$ git push [origin] [branch] Make a Version \$ git tag < version_name > Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in \(\cdot \). git config user user.name NAME	Publish
\$ git push [origin] [branch] Make a Version \$ git tag < version_name > Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in \(\cdot \). git config user user.name NAME	Push changes to origin
\$ git tag <version_name> Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in \(\cdot \).gitconfig user user.name NAME</version_name>	
\$ git tag <version_name> Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in ~/.gitconfig user user.name NAME</version_name>	
Prepare a patch \$ git format-patch origin Configuration \$ git config [global] [option] global is stored in \(\cdot \).gitconfig user user.name NAME	Make a Version
\$ git format-patch origin Configuration \$ git config [global] [option] global is stored in ~/.gitconfig user user.name NAME	\$ git tag <version_name></version_name>
\$ git format-patch origin Configuration \$ git config [global] [option] global is stored in ~/.gitconfig user user.name NAME	
Configuration \$ git config [global] [option]	•
\$ git config [global] [option] global is stored in ~/.gitconfig user user.name NAME	\$ git format-patch origin
\$ git config [global] [option] global is stored in ~/.gitconfig user user.name NAME	
global is stored in \(^{/}\).gitconfig user user.name NAME	Configuration
global is stored in \(^{/}\).gitconfig user user.name NAME	\$ git config [global] [option]
user.name NAME	
user.name NAME	

\$ git config [global] [option]
global is stored in ~/.gitconfig
user
user.name NAME
user.email EMAIL
color
color.ui auto
optimisation
packs.threads 0
diff.renamelimit 0

Commit all local changes \$ git commit -a