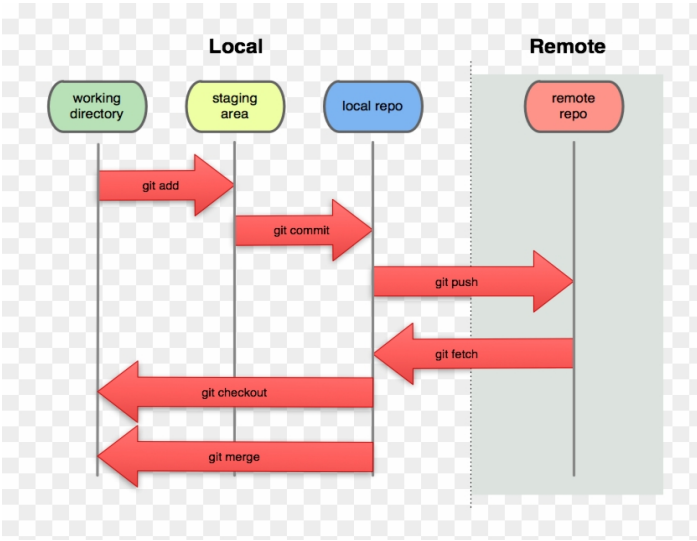
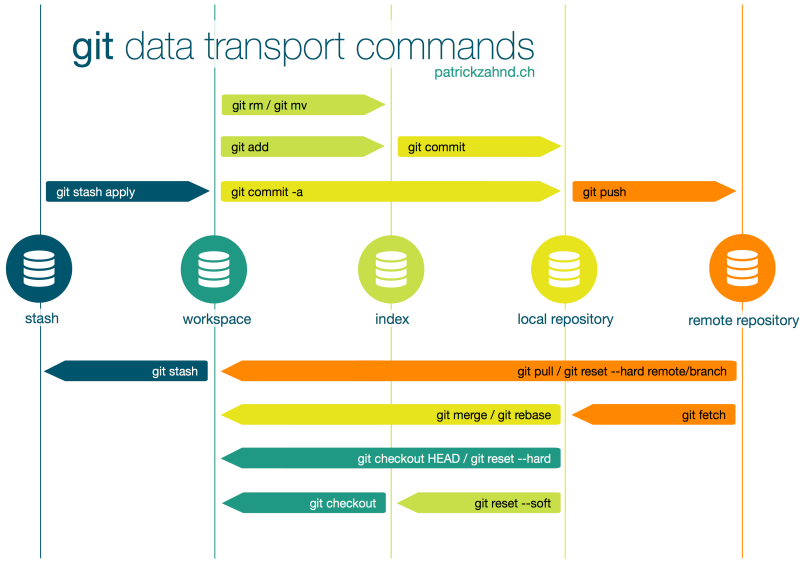
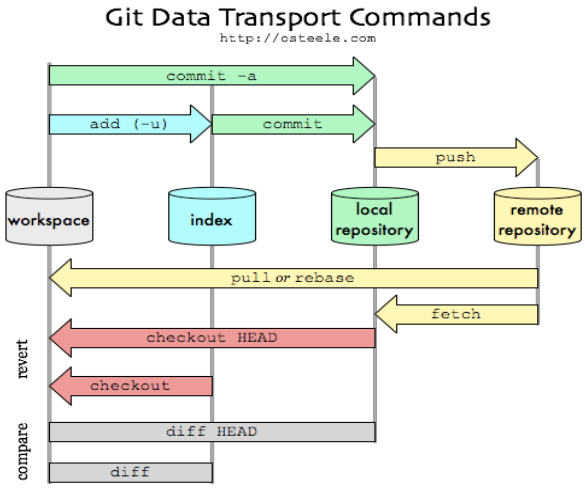
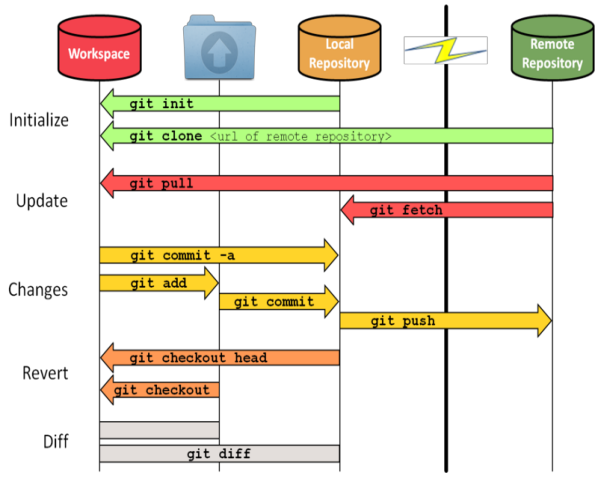








www.attakatara.wordpress.com

Git Cheat Sheet

http://git.or.cz/

Remember: git command --help

Global Git configuration is stored in $HOME/.gitconfig (git config --help)

Create

From existing data
```
cd ~/projects/myproject
git init
git add .
```

From existing repo
```
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://you@host.org/proj.git
```

Show

Files changed in working directory
```
git status
```

Changes to tracked files
```
git diff
```

What changed between $ID1 and $ID2
```
git diff $id1  $id2
```

History of changes
```
git log
```

History of changes for file with diffs
```
git log -p $file $dir/ec/tory/
```

Who changed what and when in a file
```
git blame $file
```

A commit identified by $ID
```
git show $id
```

A specific file from a specific $ID
```
git show $id:$file
```

All local branches
```
git branch
```
(star '*' marks the current branch)

Cheat Sheet Notation

$id : notation used in this sheet to represent either a commit id, branch or a tag name
$file : arbitrary file name
$branch : arbitrary branch name

Concepts

Git Basics

```
master    : default development branch
origin    : default upstream repository
HEAD      : current branch
HEAD^     : parent of HEAD
HEAD~4    : the great-great grandparent of HEAD
```

Revert

Return to the last committed state
```
git reset --hard
```
⚠ you cannot undo a hard reset

Revert the last commit
```
git revert HEAD
```
Creates a new commit

Revert specific commit
```
git revert $id
```
Creates a new commit

Fix the last commit
```
git commit -a --amend
```
(after editing the broken files)

Checkout the $id version of a file
```
git checkout $id $file
```

Branch

Switch to the $id branch
```
git checkout $id
```

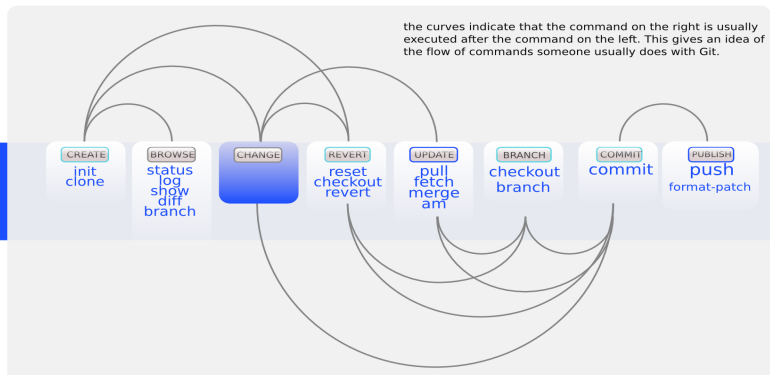
Merge branch1 into branch2
```
git checkout $branch2
git merge branch1
```

Create branch named $branch based on the HEAD
```
git branch $branch
```

Create branch $new_branch based on branch $other and switch to it
```
git checkout -b $new_branch $other
```

Delete branch $branch
```
git branch -d $branch
```

Commands Sequence



Update

Fetch latest changes from origin
```
git fetch
```
(but this does not merge them).

Pull latest changes from origin
```
git pull
```
(does a fetch followed by a merge)

Apply a patch that some sent you
```
git am -3 patch.mbox
```
(in case of a conflict, resolve and use git am --resolved)

Publish

Commit all your local changes
```
git commit -a
```

Prepare a patch for other developers
```
git format-patch origin
```

Push changes to origin
```
git push
```

Mark a version / milestone
```
git tag v1.0
```

Useful Commands

Finding regressions
```
git bisect start          (to start)
git bisect good $id       ($id is the last working version)
git bisect bad $id        ($id is a broken version)

git bisect bad/good       (to mark it as bad or good)
git bisect visualize      (to launch gitk and mark it)
git bisect reset          (once you're done)
```

Check for errors and cleanup repository
```
git fsck
git gc --prune
```

Search working directory for foo()
```
git grep "foo()"
```

Resolve Merge Conflicts

To view the merge conclicts
```
git diff          (complete conflict diff)
git diff --base  $file  (against base file)
git diff --ours   $file  (against your changes)
git diff --theirs $file  (against other changes)
```

To discard conflicting patch
```
git reset --hard
git rebase --skip
```

After resolving conflicts, merge with
```
git add $conflicting_file  (do for all resolved files)
git rebase --continue
```

Zack Rusin
Based on the work of Sébastien Pierre
Xprima Corp.

GIT CHEAT SHEET

GIT

- Git is a distributed revision control and source code management system with an emphasis on speed.
- it is repository which is used to manage projects, set of files as they changes over the time.
- Using git every code change or commit you get latest development code for the project.

GIT OPERATIONS & COMMANDS

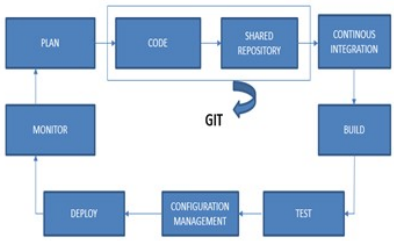
Git Configurations
- Initial config of username, email and code highlighting (optional) is to be performed.
- $git config – global user.name"firstname lastname"
- $git config – global user.email" abc123@abc.com"
- $git config – global color.ui true (enables code highlights)
- $git config --list

Initialize
- You have to initialize by using 'init'
- To know the status run the 'status' command
- $git init
- $git status

Create/Add files
- To add a file: $git add<filename>
- To add multiple files: $git add<filename> <2nd filename>
- To add all updated files: $git add –all (use -A instead of -all too)
- To remove files: $git rm -r <filename>

Commit changes
- To pass a message, use 'commit' and '-m': $git commit -m " body_of_message"
- Amend lets you amend the last commit or the last message: $git commit --amend -m " new_message"

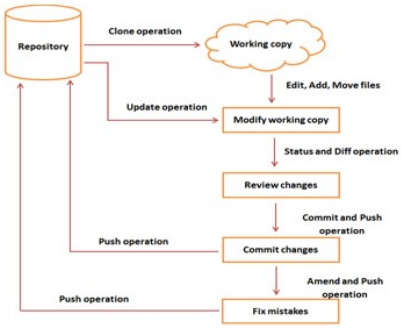
Push and Pull
- A remote repository typically represents a remote server or a git server: Create a remote repository via github "https://github.com/YourUsername/appname.git"
- To add a link: $git remote add origin<link>
- Pushing files: $git push -u origin master
- To clone file: $git clone <clone>



Version Control

- It is the management of changes to the code, documents, programs, large sites and other info.
- The changes are termed as versions.
- Version control system is used(VCS)
- The functions are :
  - Allows developers to work simultaneously.
  - Does not allow overwriting each other's changes.
  - Maintains a history of every version.

2 types of VCS - centralized and distributed. Git is distributed



GIT & GITHUB

It is a VCS that supports distributed nonlinear workflows by providing data assurance for developing quality software.

Features:
- Distributed- distributed development of code
- compatible- with existing systems and protocols
- Non- linear- non linear development of code
- Branching- easy to create and merge branches
- Lightweight- lossless compression
- Reliable- not viable to loss of data upon crashes
- Secure- SHA1 and checksum are used
- Economical- free

Branching and Merging

Command	Description
git branch	List branches
git branch -a	List all branches
git branch [branch name]	Create a new branch
git branch -d [branch name]	Delete branch
git push origin --delete [branchName]	Delete a remote branch
git checkout -b [branch name]	Create a new branch and switch to it
git checkout -b [branch name] origin/[branch name]	Clone a remote branch and switch to it
git checkout [branch name]	Switch to a branch
git checkout -	Switch to the branch last checked out
git checkout -- [file-name.txt]	Discard changes to a file
git merge [branch name]	Merge a branch into the active branch
git stash	Stash changes in a dirty working directory
git stash clear	Remove all stashed entries

Updating Projects

Commands	Description
git push origin [branch name]	Push a branch to your remote repository
git push -u origin [branch name]	Push changes to remote repository(-u remembers the branch for next use)
git push origin --delete [branch name]	Delete a remote branch
git pull	Update local repository to the newest commit
git pull origin [branch name]	Pull changes from remote repository
git remote add origin ssh://git@github.com/[username]/[repository-name].git	Add a remote repository
git remote set-url origin ssh://git@github.com/[username]/[repository-name].git	Set a repository's origin branch to SSH

Inspection and Comparison

Command	Comparison
git log	View changes
View changes	View changes (detailed)
git diff [source branch] [target branch]	Preview changes before merging