Your needs

You have a planned software project?
Don't stay alone; Tool Team has solutions for you.
Follow the services lines we offer you below.
You have question? We have answers. Call us!

Tool Team purpose

- Project Management
  - Issue tracking
  - Wiki, Forum, Roadmap
  - SCM Integration
  - Gantt Chart, Calendar
  - IDE Linkable
- Version Control System
  - Team Management
  - Create, Review
  - Activity Feeds
- Continuous Integration
  - Productivity Increase
  - Build, Test software's
  - Continuous Delivery
- Code Quality Analysis
  - Coding Rules
  - Duplications
  - Potential Bugs, Complexity
- Applicative Security Audits

Contact list

Management Team		
Role	P. NOM	+33500000000
Role	P. NOM	+33500000000
Role	P. NOM	+33500000000

Pilot		
Role	P. NOM	+33500000000
Role	P. NOM	+33500000000
Role	P. NOM	+33500000000

Tool's Ecosystem

TITRE

1. Installation

Git provides desktop and command line tools. The desktop tool includes graphical user interface to help users into most common usage

Git for Windows

https://tortoisegit.org/download

Git for Mac

https://www.sourcetreeapp.com

Git for all platforms

http://git-scm.com

2. Configuration

Configure user information for all local repositories

$ git config --global user.name "[name]"
Sets the name you want attached to your commits transactions
$ git config --global user.emal "[email address]"
Sets the email you want attached to your commits transactions

3. Create repositories

Start a new repository or obtain one from an existing URL

$ git init [project-name]
Creates a new local repository with the specified name
$ git clone [url]
Downloads a project and its entire version history

4. Make changes

Review, edits and craft a commit transaction

$ git status
Lists all new or modified files to be commited
$ git diff
Shows file differences not yet staged
$ git add [file]
Snapshots the file in preparation for versionning
$ git diff --staged
Shows file differences between staging and the last file version
$ git reset [file]
Unstages the file, but preserve its contents
$ git commit -m "[descriptive message]"
Record file snapshots permanently in version history

5. Group changes

Name a series of commits and combine completed efforts

$ git branch
List all local branches in the current repository
$ git branch [branch-name]
Creates a new branch
$ git checkout [branch-name]
Switches to the specified branch and updates the directory
$ git merge [branch]
Combines the specified branch's history into the current branch
$ git branch -d [branch-name]
Deletes the specified branch

6. Refactor filenames

Relocate and remove versioned files

$ git rm [file]
Deletes the file from the working directory and stages the deletion
$ git rm --cached [file]
Removes the file from git VCS but preserves the file locally
$ git mv [file-original] [file-renamed]
Changes the file name and prepares it for commit

7. Suppress tracking

Exclude temporary files and paths

.log** **build/** **temp-
A text file names .gitignore suppresses accidental versionning of files and path matching the specified patterns

8. Save fragments

Shelve and restore incomplete changes

$ git stash
Temporarily stores all modified tracked files
$ git stash pop
Restores the most recently stashed files
$ git stash list
Lists all stashed changesets
$ git statsh drop
Discard the most recently stashed changesets

9. Review history

Browse and inspect the evolution of project files

$ git log
Lists version history for the current branch
$ git log --follow [file]
List version history for a file, including renames
$ git diff [first-branch]...[second-branch]
Shows content differences between two branches
$ git show [commit]
Outputs metadata and content changes of the specified commit

10. Redo commits

Erase mistakes and craft replacement history

$ git reset [commit]
Undoes all commits after [commit], preserving changes locally
$ git reset --hard [commit]
Discards all history and changes back to the specified commit

11. Synchronize changes

Register a repository bookmark and exchange version history

$ git fetch [bookmark]
Downloads all history from the repository bookmark
$ git merge [bookmark]/[branch]
Combines bookmark's branch into current local branch
$ git push [alias] [branch]
Uploads all local branch commit to git remote server
$ git pull
Downloads bookmark history and incorporates changes