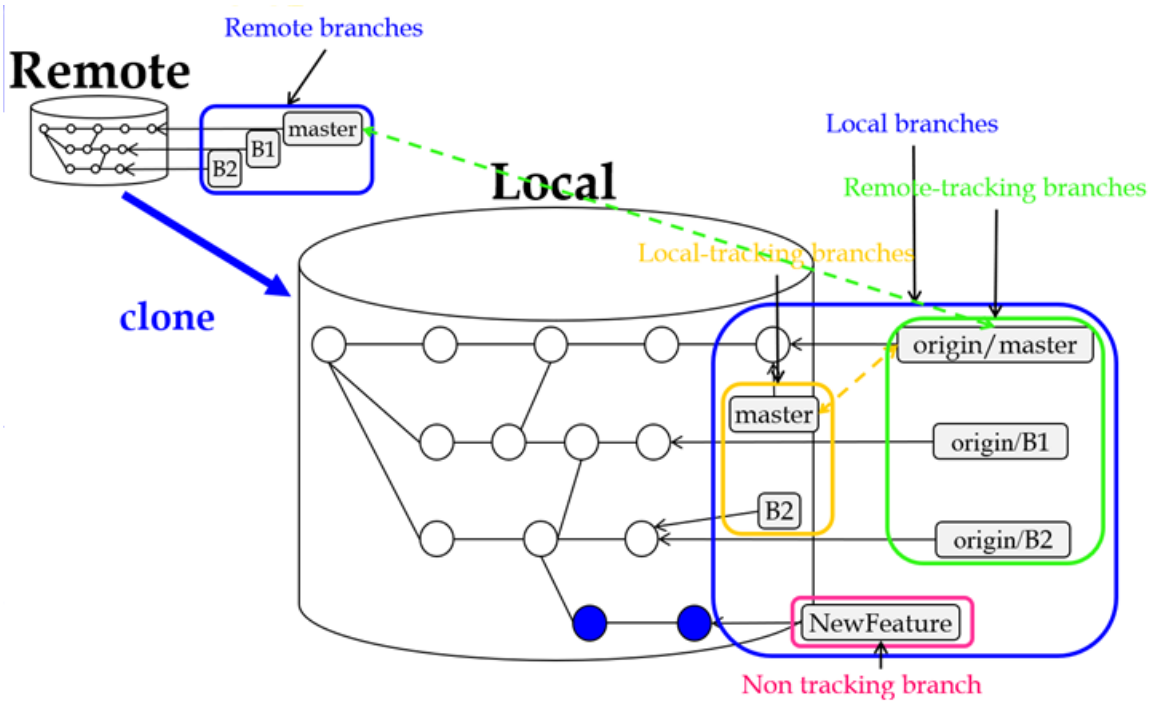
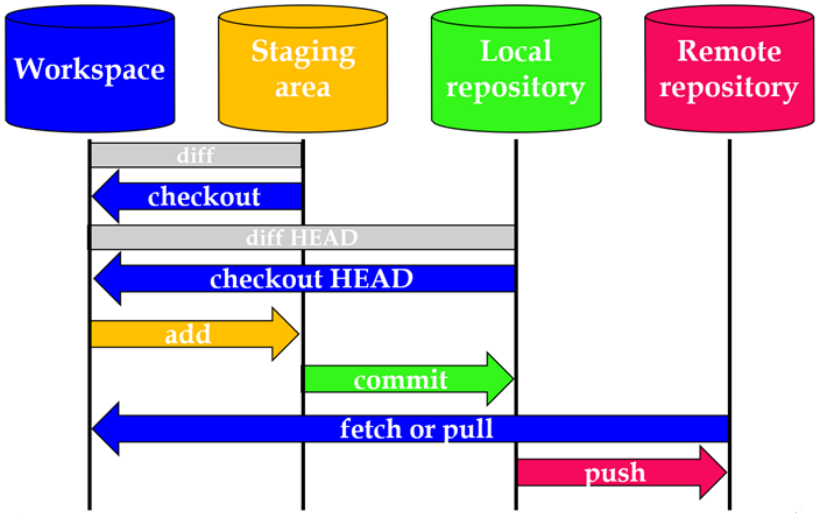
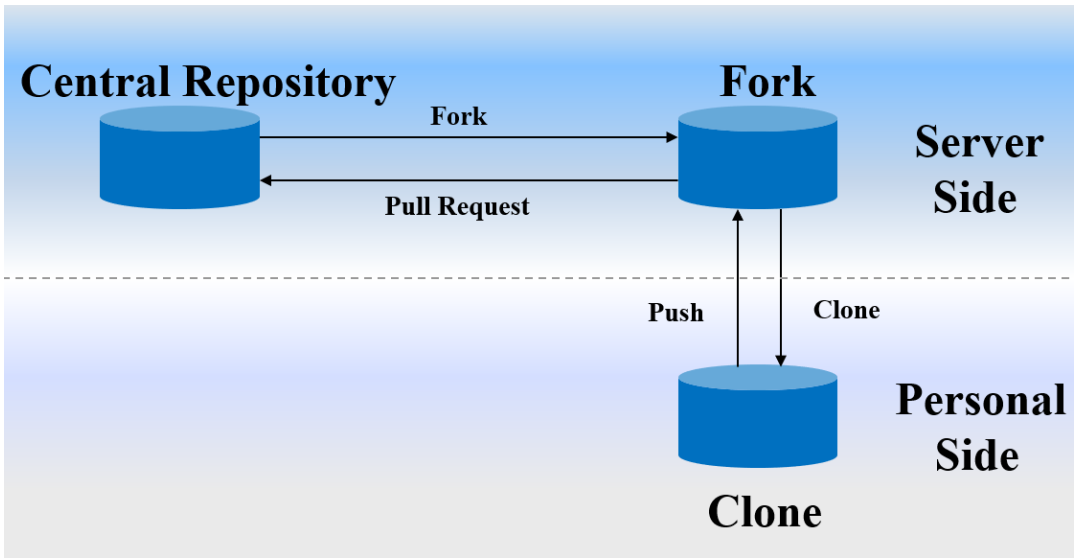

Remote branches
Remote
master
B1
B2
Local branches
Remote-tracking branches
Local
Local-tracking branches
clone
origin/master
master
origin/B1
B2
origin/B2
NewFeature
Non tracking branch


Workspace
Staging area
Local repository
Remote repository
diff
checkout
diff HEAD
checkout HEAD
add
commit
fetch or pull
push


Central Repository
Fork
Fork
Pull Request
Server Side
Push
Clone
Personal Side
Clone

Commands for Setting Up Git Repositories

Create an empty repository in the project folder: **cd to project folder**

git init

Clone a repository from GitHub and add it to the project folder:

git clone (repo URL)

Clone a repository to a specific folder:

git clone (repo URL) (folder)

Display a list of remote repositories with URLs:

git remote -v

Remove a remote repository:

git remote rm (remote repo name)

Retrieve the most recent changes from origin but don't merge:

git fetch

Retrieve the most recent changes from origin and merge:

git pull

Commands for Managing File Changes

Add file changes to staging:

git add (file name)

Add all directory changes to staging:

git add .

Add new and modified files to staging:

git add -A

Remove a file and stop tracking it:

git rm (file_name)

Untrack the current file:

git rm –cached (file_name)

Recover a deleted file and prepare it for commit:

git checkout <deleted file name>

Display the status of modified files:

git status

Display a list of ignored files:

git ls-files –other –ignored –exclude-standard

Display all unstaged changes in the index and the current directory:

git diff

Display differences between files in staging and the most recent versions:

git diff –staged

Display changes in a file compared to the most recent commit:

git diff (file_name)

Commands for Declaring Git Commits

Commit changes along with a custom message:

git commit -m "(message)"

Commit and add all changes to staging:

git commit -am "(message)"

Switch to a commit in the current branch:

git checkout <commit>

Show metadata and content changes of a commit:

git show <commit>

Discard all changes to a commit:

git reset –hard <commit>

Discard all local changes in the directory:

git reset –hard Head

Show the history of changes:

git log

Stash all modified files:

git stash

Retrieve stashed files:

git stash pop

Empty stash:

git stash drop

Define a tag:

git tag (tag_name)

Push changes to origin:

git push

Commands for Git Branching

Display a list of all branches:

git branch

Make a new branch and switch to it:

git checkout -b <branchname>

Switch to a branch:

git checkout <branchname>

Delete a branch:

git branch -d <branchname>

Merge a different branch with your active branch:

git merge <branchname>

Fetch a branch from the repository:

git fetch remote <branchname>

View merge conflicts between branches:

git diff <sourcebranch> <targetbranch>

Preview changes before merging branches:

git diff <sourcebranch> <targetbranch>

Push all local branches to a designated remote repository:

git push –all

Git Tips

Knowing all the Git commands won't get you far if you don't know how to make the most of them. Here are some version control best practices to follow:

1. **Commit Often**

Keep your commits small by committing changes as often as possible. This makes it easier for team members to integrate their work without encountering merge conflicts.

2. **Test, Then Commit**

Never commit incomplete work. Always test your changes before sharing your code with others.

3. **Use Commit Messages**

Write commit messages to let other team members know what kind of changes you made. Be as descriptive as possible so your teammates know exactly what to look for.

4. **Branch Out**

Take full advantage of branches to help you keep track of different lines of development. Don't be afraid to go out on a limb and create a new branch to experiment with new features and ideas.

5. **Settle on a Common Workflow**

There are several different ways to set up your Git workflow. Whichever one you choose, make sure you and your teammates are on the same page from the very beginning.

Automation with Python

- navigate to MyProjects
- create folder with project name
- navigate into the folder

```
> mkdir vdjango

> cd vdjango

vdjango> virtualenv -p C:\Users\kyung.lee\AppData\Local\Programs\Python\Python36\python.exe .

vdjango> .\Scripts\activate

vdjango> pip install -r requirements.txt
```

- git init
- go to GitHub and create new repository
- copy the remote
- add remote to my local folder
- create a readme file
- git add
- git commit
- git push
- code .