**Git есть в linux и под WSL**

Для того что бы можно было клонировать репозиторий, а так же пушить в него изменения, то нужно установить в настройках SSH ключ

**rsa** – приватный ключ

**rsa\_pub** – публичный ключ

**rsa.ppk** – приватный ключ для PuTTY

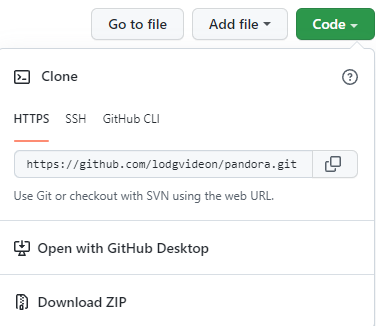
На хосте **ОБЯЗАТЕЛЬНО** должны находиться и приватный ключ и публичный. А так же публичный ключ должен находиться на сервере, к которому мы подключаемся. Например, у меня не получилось при добавлении публичного ключа, который мне сгенерил Камышов, на BitBucket вносить изменения, так как у меня на хосте не было приватного ключа (был только ключ формата PuTTY). По этому я создал свой собственный ключ

**ssh-keygen -t rsa** – генерирует SSH ключ в папку ~/.ssh

**cat ~/.ssh/id\_rsa.pub** – наш публичный ключ. Копируем его содержимое и вставляем туда где он нужен. Например в github

**sudo chmod 600 id\_rsa** – даем приватному ключу право на исполнение

Копируем ссылку на гит - https://github.com/lodgvideon/pandora.git



git clone <https://github.com/lodgvideon/pandora.git> - клонируем себе репозиторий

git clone -b [название\_ветки] [репозиторий] – клонируем себе определенную ветку из репозитория

git pull – обновить локальный код (спулить из репозитория). Нужно находиться в этом репозитории

---Далее требуется быть в папке с проектом---

git checkout -b [название\_ветки] – создать ветку и перейти в нее

git checkout [название\_ветки] – поменять ветку

git branch [название\_ветки] – создать ветку

git switch [название\_ветки] – поменять ветку

git branch – проверить на какой ветке сейчас находишься

git add [файл.расширение] – selects that file, and moves it to the staging area, marking it for inclusion in the next commit

git commit -m 'Initial commit' – запушить изменения в локальный репозиторий

git push --set-upstream origin [название\_ветки] – запушить изменения в репозиторий проекта

git status – получить статус того какие файлы мы пушим

git merge [название\_ветки] – мержит изменения ветки на которой находимся с указанной веткой

git rebase [название\_ветки] - мержит изменения ветки на которой находимся с указанной веткой. В отличии от merge он не смерживает ветки, а перемещает ветку на которой находимся с веткой которую указали, и “ветвления” пропадают

**HEAD** - это символическое имя текущего выбранного коммита — это, по сути, тот коммит, над которым мы в данным момент работаем.

git checkout [хэш\_комита] – перейти на HEAD

git checkout [название\_ветки]^ - переключиться на прошлый комит. ^^ - переключиться на позапрошлый комит.

git checkout HEAD~[N] – переключиться на N кол-во комитов назад

git branch -f [название\_ветки] HEAD~[N] – переместит ветку на N комитов назад

git branch -f [название\_ветки] [название\_ветки]~[N] - переместит ветку на N комитов назад

git branch -f [название\_ветки] [хэш\_комита] – перенести комит в ветку

git reset HEAD~[N] – переносит ветку на N комитов назад (по сути удаляет последующие комиты). Нужно находиться на нужной ветке. Работает только локально

git revert HEAD – создает новй комит, который не содержит изменения прошлого комита. Работает для удаленных веток

в .gitignore прописывается что можно залить, а что нет