***Конфигурация gitconfig***

* **Git config –-global user.name “Maxim”**
* **Git config –-global user.email** [**graal@inbox.ru**](mailto:graal@inbox.ru)
* **Git config -l просмотр файла**

**Linux config file : /home/myuser/.gitconfig**

**Windows config file : C:\User\myuser\.gitconfig**

***Работа с Local Repository***

mkdir myproject

git init . ( если находишься в этой директории) создается папка “.git”

git status – отслеживание что делаете

git add \* - добавление всех файлов

git commit -m "my initial commit is version one"

***История изменений | gitignore | восстановление файлов***

Git log – история всех комитов

git log -1 – изменения в последнем комите

git log -p -1 – что измененно в последнем комите

git restore file4.txt – возвращение файла в пред состояние

git checkout – еще команда для веток

git diff –staged - разница между последним комитом и stage

.gitignore – файл где прописываются что игнорируется и не будет видно в git status

**Загрузка проекта на GitHub - git push**

**git clone** [**https://github.com/LeonidUspeshny/mysuperproject1.git**](https://github.com/LeonidUspeshny/mysuperproject1.git) **- клонирует директорий который выбран на github**

**gti push origin – кидает в репозиторий содержимое папки(при первом разе спросит логин и пароль)**

***Hастройка логина в GitHub через SSH Key на Linux***

Git remote -v

Git remote set-url origin [git@github.com:LeonidUspeshny/mysuperproject1.git](mailto:git@github.com:LeonidUspeshny/mysuperproject1.git)

Git clone [git@github.com:LeonidUspeshny/mysuperproject1.git](mailto:git@github.com:LeonidUspeshny/mysuperproject1.git)

***Hастройка логина в GitHub через SSH Key на Windows***

**Копируем паблик кей заходим в настройки github в раздел ssh and gpg key создаем новый ключ называем его как нибудь вставляем паблик кей сохраняем копируем репозиторий работаем там что то делаем**

**Git add . git commit – “” git push origin**

***Создание и работа с Ветвлениями – Branch***

Git branch – просмотр веток

Git branch test - создать в ветку test

Git checkout test – переход в ветку test

Git checkout -b test - создает ветку test и сразу туда переходит

Git checkout master – переход в основную ветку master

Git merge test - объединение двух веток с изменениями которые были сделаны в ветке test

Git branch -d test – удаляет ветку test предупреждая что там были сделаны изменения

Git branch -D test – удаляет ветку test и переходит в ветку master

***Возврат на предидущие версии***

Git log –

Git checkout “номер комита” – возврат к предыдущей версии файла

Git checkout master – возврат обратно на конечный результат

Git reset –hard HEAD~ - возврат на предыдущую версию

Git reset –hard HEAD~2 – возврат на 2 версии назад

Git reset –soft HEAD~4 – возврат на 4 версии назад

Git reset –-soft HEAD~3 – удаляет 3 последних комита с сохранением последней версии файла.

Git commit –amend – открывается юникс редактор (insert,esc,: enter) редактируем коммит

***Полный рабочий цикл действий Git и GitHub***

Git clone “ссылка на репозиторий”

Git checkout -b leonid\_task009

Git branch проверяем в какой ветке находимся

Копируем, правим файл, что то делаем с файлами и т.д.

Git status

Git add .

Git status

Git commit -m “update file leonid”

Git branch

Git push --set-upstream origin leonid\_task009 – создаем в гит хабе бренч leonid\_task009

Git checkout master – переходим на основной бренч

Git branch -d leonid – удаляем свой бренч локально

git push origin --delete leonid – удаленно удаляется бренч leonid

***Как работать с тегами в репозитории Git на примере GitHub - Git Tags***

Git tag –

Git tag v1.0.0 -

Git tag -a v1.2.0 -

Git push origin v1.0.0 -

Git checkout v2.0.0 -

Git tag -d v1.1.1 -

Git push origin –delete v1.1.1 -

Git log -pretty=oneline –

***Как удалить из истории секретную информацию локально и на удалённом репозитории***

Git reset –hard HEAD~1- вернуться на 1 коммит

Git push origin main –-force – перезаписывает удаленно коммит

***GitHub Actions - Основы Автоматизации - DevOps - GitOps***