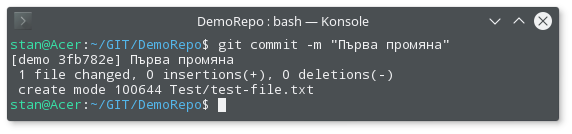
**Ръководство за работа с git**

Git е система за контрол на версиите на софтуерен проект. Характерното при нея е че всеки един от разработчиците пази копие на хранилището на своя локален компютър. Което дава възможността на много хора да работят по един и същ файл по едно и също време, без да си пречат.  
  
**Клон** на едно хранилище представлява инстанция (копие на друг клон) на това хранилище в определен момент от времето на разработка.  
  
**Къмит** е сбор от променени файлове, той се прави с цел по-лесно качване на код в хранилището и по-лесно следене на промените.За удобство е прието да се добавя съобщение, което да дава някаква кратка информация за това какво е променено.  
  
**Конфликт** е когато двама човека са променяли един и същ номер ред в един и същ файл (празните редове включително, git приема един празен ред в повече или по-малко като конфликт).  
  
**Това са основните команди във git:**  
  
git clone “линк към хранилището” - Клонира се цялото хранилище с всичките му клонове към момента на клониране, на локалния компютър. Създава се папка с името на проекта, в която вече работят git командите.  
  
git checkout -b “име на клон” - създаване на нов клон, взема се копие от файловете на текущия, ако се изпълни в клон “master” ще вземе кода от master  
  
git checkout - Прехвърляне в нов клон на хранилището, дори и да го нямате на локалния компютър, гит автоматично го тегли.  
  
git branch - списък със всички клонове, които ги има на локалния компютър, в зелено е текущия

git pull - изтегля всички промени направени по клоновете, които имате на локалния компютър от хранилището  
 например: имате 2 клона на вашия компютър - “master” и “GH-1”, преди вие да качите промени по някой от тези бранчове, някой друг е качил промяна. При това положение се налага да се изтегли новата версия на клона. След това ако няма конфликти git ще ви разреши да качите вашите промени. Ако ли не текстовия редактор ще ви предостави възможност да сравниет разликите и вие да изберете какво да се направи по-напред(запазване на вашите промени, запазване на чуждите промени, запазване и на двете промени).

git status - Изкарва ни информация с това, на кой клон се намираме и всички промени, които сме направили по клона (добавени, променени или изтрити файлове). С червено са тези които няма да се добавят при следващия къмит(commit), със зелено са тези които ще се добавят.  
  
git add “път на файла” - добавя файла към следващия commit  
  
git add . - добавя всички червени файлове от git status (не е препоръчително, защото не винаги искаме всички файлове да се добавят)

git commit -m “[GH-#номер\_на\_клон]Какво е добавено с този къмит” - Команда за създаване на къмит със съобщение, зелените файлове от git status влизат в него, и се генерира уникален ключ на къмита, който може да бъде достъпен ако се наложи. На снимката отдолу, ключът на къмита е 3fb782e.



*Професионалният начин за организиране на къмит съобщенията е с в командата по-горе, но гит позволява пълна свобода при къмит съобщенията, както се вижда от снимката.*

git push origin “име на клона” - Създадения къмит се качва в хранилището.

**Специфични команди и операции в git:**  
  
git checkout master  
git merge GH-1   
Това е пример за сливане на кодът от GH-1 в master, ако има конфликти git не позволява сливането и се извършва процесът по избиране на промени.  
*При групови проекти е добра практика да се прави pull request през сайта, където всеки разработчик може да види сливането под формата на предложение, това се прави с цел по-добър контрол на работния процес.*

git stash - Ако имате файлове които не искате да къмитвате, а искате да смените бранч. Променените файлове се добавят към една опашка от чакащи промени.  
  
git stash pop - Изкарва всички чакащи променени файлове от опашката.

git cherry-pick - Начин за вземане на цял клон по зададен ключ. Това е решение, ако къмитнете към грешен бранч и искате да върнете предишното състояние на бранча, а вашите промени да си станат пак чакащи.