|  |  |
| --- | --- |
| git **add** . | добавить в список назначенных к выгрузке ВСЕ файлы и папки |
| git rm --cached blog/blog.php | исключить из назначенных к выгрузке конкретный файл |
| git **commit** -m "README file was changed" | выгрузить в ЛОКАЛЬНЫЙ репозиторий файлы и дать комментарий |
| git **push** -u origin master | из локального выгрузить ветку главную в репозиторий, подключенный как origin, в ветку master |
| git push | краткая команда выгрузить из локалки в репозиторий |
|  |  |
| git **reset** --soft origin | **Главная команда!**  Отменяет сделанный коммит, если он ещё не запушен. Вместо origin ставьте свой репозиторий и ветку можно ещё добавить. |
| git **checkout** <hello.html> | отбэкапиться на версию до коммита с hello.html |
| git reset <кусок изменённого объекта, например HEAD hello.html> | убрать из индексации часть изменений (после случайного git add .) |
| git log --oneline | посмотреть лог по 1 строчке |
|  |  |
|  |  |
|  |  |
| настройки подключения |  |
| git config --global user.name "AntAlMakarov" | установить имя своего юзера |
| git config --global user.email "AntAlMakarov@greenatom.ru" | установить email своего юзера |
|  |  |
|  |  |
| git **remote** add origin https://gitlab.rosatom.local/rpa\_dev/error\_handler.git | подключиться к репозиторию "error\_handler", созданному на гит |
| git remote add origin https://github.com/Antohnio123/Antohnio.git | подключиться к моему репозиторию |
| git **remote set-url origin** new.git.url/here | подключиться к другому репозиторию |
| git clone https://github.com/aanikin/Python | скачать папку с чужого репозитория |
| git remote add aanikin https://github.com/aanikin/Python | подключение к репозиторию АА Аникина, где перед адресом стоит ИМЯ! |
| PS D:\Python\aanikin\Python> git add Practice/a.makarov/test.py | добавление папки, либо можно добавить всё |
| git pull aanikin master --allow-unrelated-histories | скачать данные с репозитория АА Аникина |
| PS D:\Python\aanikin\Python> git push -u aanikin --all | ПОСЛЕ pull закинуть измененные свои файлы |
|  |  |
|  |  |
| Анализ |  |
| git status | посмотреть, что назначено к выгрузке в Локалку |
| git **remote** show origin | то же самое |
| git remote -v | узнать, к каким репозиториям подключен |
| git config --get remote.origin.url | увидеть, какой онлайн-репозиторий выбран как origin |
| git branch -a | увидеть все ветки |
| git log --pretty=oneline | Проверить историю коммитов |
| git tag | Посмотреть список имеющихся тегов |
| commits |  |
| git tag stable-1 | создать «легковесный» тэг, связанный с последним  коммитом. Если тэг уже есть, то еще один создан не будет. |
| git tag stable-2 f292ef5 | пометить определенный коммит. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| SSH-key |  |
| Open a terminal on Linux or macOS, or Git Bash / WSL on Windows.  Generate a new ED25519 SSH key pair:  ssh-keygen -t ed25519 -C "email@example.com"  Or, if you want to use RSA:  ssh-keygen -t rsa -b 4096 -C "email@example.com"  It will generate the key in => C:\Users\yourname.ssh directory.  Copy the public key and paste in the gitlab location.  ssh-agent $(ssh-add C:\Users\youname\.ssh\id\_rsa; git clone git@gitlab.com:xyz/SpringBootStarter.git) | |
| В Винде надо удалить старый пароль и логин в Windows Credentials |  |
|  |  |
| ВЕТКИ: |  |
| git status | показать, на какой ветке мы сейчас и в каком состоянии |
| **git branch -a** | увидеть все ветки |
| git checkout -b admin | создать ветку admin и сразу на неё переключиться |
| **git checkout forum** | переключиться на ветку форума |
| git checkout 2a36c1b | переключиться на версию с указанным ID |
| git revert 7f6b500 | отменить коммит с указанным ID |
| git reset 9cf99cc --hard | нормально отбэкапиться к ID версии |
| git merge forum | слить текущую ветку и указанную (forum) |
| git branch -d <имя ветки> | удалить ветку за ненадобностью или после merge с другой. |
| git branch --set-upstream-to=master tocheck | ветка tocheck теперь тречит ветку master (нужно для оттречивания веток на сторонних ресурсах) |
| git push origin <имя ветки> | запушить на репо origin ветку <имя ветки> |
|  |  |
| Merge | Слияние веток |
| $ git checkout <feature>  $ git merge master  (or)  $ git merge master <feature> |  |
|  |  |
| Rebase | Перебазирвоание одной ветки в другую (мастера в <feature>). ОСТОРОЖНО! Перезаписывает историю, срывает связи с удаленными ветками (требуется новый пуш) |
| git checkout <feature> |  |
| git rebase -i master | Это откроет редактор, перечислив все коммиты, которые будут перемещены. Это точно определяет, как будет выглядеть ветка после выполнения перемещения.  Упорядочивая объекты, вы можете сделать историю такой, как захотите. Вы можете использовать команды fixup, squash, edit, и так далее |
|  |  |
| .GITIGNORE |  |
|  | Файл должен иметь формат «.gitignore» без имени. Винда не даст такой создать, но можно сделать «.gitignore.» с точкой в конце, которая исчезнет автоматически. |
| / | Если шаблон начинается со слеша, то он соответствует файлам и каталогам только в корне хранилища.  Если шаблон не начинается со слэша, он соответствует файлам и каталогам в любом каталоге или подкаталоге. |
| \* | Символ звездочки соответствует нулю или более символам |
| \*\* | Два соседних символа звездочки соответствуют любому файлу или нулю или более каталогам. Когда за ним следует косая черта (/), она соответствует только каталогам |
| ? | Вопросительный знак соответствует любому отдельному символу |
| […] | Любой ИЗ символов в скобках |
| \*.[oa] | file.o  file.a  dsfgfdg.o |
| \*.[!oa] | Если первым знаком стоит восклицательный, он зачит отрицание - любой символ КРОМЕ указанных в скобках  file.s  file.1  file.0 |
|  |  |
| !.gitignore  !script.pl  !template.latex | …But not these files... |
| !\*/ | ...even if they are in subdirectories |
| !\*/a/b/file1.txt  !\*/a/b/c/\* | if the files to be tracked are in subdirectories |
| файлы |  |
| /files/data.zip  /server.log  /uploads/users/data/info.xls | Конкретные файлы |
| \*.zip  \*.log  \*.pdf  \*.xls | Игнорирование по типу файла, будут игнорироваться в АБСОЛЮТНО всех директориях |
| config.php | Игнорирование конкретного файла ТОЛЬКО в корне проекта  (корнём считается расположение файла .gitignore)  Например НЕ БУДЕТ проигнорирован файл /db/config.php |
| /params/config.php | Игнорирование конкретного файла ТОЛЬКО в указанной директории |
| Папки |  |
| /images/\* | Игнорирование всех файлов и папок ТОЛЬКО в конкретной директории корень/images  (включая поддиректории и файлы в них)  Например /images/user.jpg, /images/company/logo.png  # НЕ БУДУТ проигнорированы файлы и папки /prod/images/user.jpg |
|  |  |
| /private/\*.html | Игнорирование ВСЕХ html-файлов в ОДНОЙ КОНКРЕТНОЙ директории(НЕ ВКЛЮЧАЯ поддиректории)  # Например /private/index.html  # НЕ БУДУТ проигнорированы файлы в /private/ivan/index.html |
| /private/\*\*/\*.html | Игнорирование ВСЕХ html-файлов в КОНКРЕТНОЙ директории ВКЛЮЧАЯ поддиректории  # Например /private/info.html, /private/users/ivan/info.html |
| /secret/\*  !/secret/free.txt | Исключение из игнорирования  # Игнорирование ВСЕХ файлов и папок внутри директории /secret,  # за исключением файла /secret/free.txt, он не будет проигнорирован |
| \!readme!.txt | Игнорирование файла с именем, содержащим спецсимволы |
| /images/h?/\*.jp?g | Игнорирование всех JPG и JPEG файлов внутри директорий,  которые начинаются на "h" и МОГУТ содержать ещё один символ после  # Например /images/h4/user.jpg, /images/h/company.jpeg |
|  |  |
| CI/CD | https://youtu.be/jkwJCR0rqDc |
| https://docs.gitlab.com/ee/ci/yaml/ | Доки по написанию yml-файлов |
| **image:** docker:stable  **services:** - docker:dind    **stages:** - pre-build  - build  **.base:** *# tags:* *# - docker-shell* **before\_script:** - docker login -u gitlab-ci-token -p $CI\_JOB\_TOKEN $CI\_REGISTRY  **script:** - BRANCH\_TAG=$CI\_COMMIT\_REF\_NAME  - docker pull $CI\_REGISTRY\_IMAGE:latest || true  - docker pull $CI\_REGISTRY\_IMAGE:$BRANCH\_TAG || true *# Сделано для того, чтобы не подтягивать каждый раз предыдущие коммиты* - docker build --cache-from $CI\_REGISTRY\_IMAGE:$BRANCH\_TAG --tag $CI\_REGISTRY\_IMAGE:$CI\_COMMIT\_REF\_SLUG . *#$CI\_REGISTRY\_IMAGE:$BRANCH\_TAG .*  *#- docker-compose up -d*  *#- docker push $CI\_REGISTRY\_IMAGE/$FOLDER:$BRANCH\_TAG* - docker push $CI\_REGISTRY\_IMAGE:$CI\_COMMIT\_REF\_SLUG  **build:master:**  **extends:** .base  **stage:** build  **variables:**  **CI\_COMMIT\_REF\_NAME:** latest  **only:** - master  **pre\_build:features:**  **when:** manual  **extends:** .base  **stage:** pre-build  **except:** - master | *# Это сервис, позволяющий запускать докер внутри докера внутри докера...* *Git Runner бежит в докере, а внутри мы билдим свой докер имадж. Получается матрёшка докеров. dind позволяет это.* |
| <https://docs.gitlab.com/ee/ci/variables/predefined_variables.html> | Набор предопределённых переменных в CI/CD.  Можно вводить и свои дополнительно! |
| $CI\_REGISTRY |  |
| $CI\_REGISTRY\_IMAGE | Докер-имадж в GitLab registry |
| $CI\_COMMIT\_REF\_NAME | The branch or tag name for which project is built. |
| $CI\_COMMIT\_REF\_SLUG | CI\_COMMIT\_REF\_NAME in lowercase, shortened to 63 bytes, and with everything except 0-9 and a-z replaced with -. No leading / trailing -. Use in URLs, host names and domain names. |
| $CI\_JOB\_TOKEN | A token to authenticate with certain API endpoints. The token is valid as long as the job is running. |
| $CI\_JOB\_STAGE | текущий стейдж джобы (stage) |
|  |  |
| - echo "$CI\_JOB\_STAGE" | вывести в консоль сообщение со значением переменной |
| .deploy\_common:  # common config HERE  deploy\_master\_CD:  extends: .deploy\_common  only:  refs:  - master  deploy\_manual:  extends: .deploy\_common  when: manual |  |
| deploy:  rules:  - if: '$CI\_COMMIT\_REF\_NAME == "master"'  - when: manual | УСЛОВИЯ запуска лучше ставить через rules, НЕ через only/except |
|  |  |
|  |  |