# 4. Шпаргалки по технологиям

В этой книге собраны инструкции по различным инструментам и технологиям

- Git
  - Git: Основы
  - Git: Работа с ветками
- SSH
  - Local port forwarding
  - Скачивание и загрузка файлов по SSH
  - SSH as proxy
- SAMBA
  - Монтирование самбы на Ubuntu server
- Databases
  - Postgres database dump and restore
  - DB Zoo
- WireShark
- ffmpeg/ffprobe

# Git

# Git: Основы

## Определение

Git - система версионирования текстовых файлов. С помощью неё удобно отслеживать сделанные изменения, переключаться на старую версию файлов проекта, совмещать изменения сделанные разными людьми и просматривать разницу между текстовыми файлами.

## Основные понятия

- commit - зафиксированное состояние (и как действие: фиксация изменения)
- commit message - сообщение, поясняющее сделанные изменения
- branch - ответвление состояния из другого состояния. Ветка.
- push - процесс отправки локальных изменений на удалённый сервер
- pull - процесс скачивания изменения с удалённого сервера
- fetch - обновление истории коммитов на локальной машине с сервера (без скачивания самих изменений)
- checkout - процесс переключения на определённое состояние (на коммит/тэг/ветку)
- tag - пометка на коммите (например версия: v1.2.3)

## Базовая работа с git

## Для удобной работы с git используются следующие файлы:

Для того чтобы вы могли пулить и пушить не вводя пароль каждый раз, можно создать два файла, которые будут хранить эти данные. Однако, если вы работайте на многопользовательском сервере, то будьте аккуратнее чтобы к вашему юзеру не было доступа у других.

Однако пароль из-за требований безопасности не рекомендуется использовать. Вместо пароля используются более гибкие временные пароли - токены. Сгенерируйте токены на странице в GitLab (`https://gitlab.example.com/-/user_settings/ssh_keys`) или GitHub (https://github.com/settings/tokens) с нужными правами доступа. (Для новичков подойдёт классический токен - он проще настраивается)

**Итак, создаём файлы в пользовательской директории `/home/you_user` для linux или `C:\Users\you_user` для windows:**

- `.gitconfig` - файл, информация из которого используется в коммитах (имя из этого файла будет отображаться в коммитах).
  Пример заполнения:

  ```
  [credential]
      helper = store # указание что токен нужно брать из .git-credentials
  [user]
      name = Firstname Lastname
      email = your-email@gmail.com
  ```

- `.git-credentials` - файл с реквизитами доступа к git-серверам (файл с токенами). В этом файле могут быть credentials для разных удалённых серверов одновременно.
  Пример с одним сервером:

  ```
  https://username:ghp_some0very0very0very0very0long0token@github.com
  ```

  формат заполенения: каждая строчка состоит из: `https://username:token@domain.name`.
  Важное примечание: если в токене есть символ `@`, то vscode ломает аутентификацию. Лучше перегенерируйте токен.

**Опционально можно создать дополнительные файлы:**

- `.gitignore` - в папке юзера лежит глобальный gitignore, то есть применяющийся ко всем проектам git. (В папке проекта - локальный для проекта). Советую его заполнить чтобы случайно не закоммитить пароли или виртуальные окружения в любых проектах:

  ```
  .idea
  .idea/
  .vscode
  .ipynb_checkpoints
  __pycache__
  venv/
  .venv/

  clients/
  statistic/
  _statistic/
  userdata/
  logs/

  *.env*
  *.conf*
  ```

```
*.logs*
*.log*

*privatekey*
test.txt
text.txt
ignore/
```

## Версионирование

Для продукта чаще всего используют семантическое версионирование vX.Y.Z (например v1.7.42)

Вкратце договорённость такая:

- X - мажорная версия, растёт при потере обратной совместимости API (под API тут понимается в том числе взаимосвязи между модулями одного приложения)
- Y - минорная версия, растёт при добавлении новой функциональности
- Z - патч/багфикс-версия, растёт при починке багов без новой функциональности

> ZeroVer - софт, который имеет нулевую мажорную версию принято считать не готовым для использования, тем не менее, такого софта много. Старайтесь соблюдать правило: если считаете софт не готовым для использования пользователями, то придерживайтесь нулевой мажорной версии.

**Тэги**

Теги используются для отметки специальных состояний проекта, например релиза или конкретной версии. Если вы используйте семантическое версионирование, то ваш тэг будет выглдяеть примерно так `v1.2.3`

# Читайте далее:

Git: Работа с ветками

# Git: Работа с ветками

**master / main / default / trunk** - стандартные названия основных ветвей. Стабильные ветки, из которых обычно выпускают релизы.

**dev / development** - основная ветка разработки, туда вносятся текущие изменения, не считается стабильной. Если изменений вносится немного (поправить пару строчек / заменить конфиг / обновить библиотеку с сохранением совместимости) и эти изменения безопасны допустимо добавление коммитов непосредственно в dev-ветвь. Во всех остальных случаях стоит использовать feature-branch.

**feature-branch / ticket-branch** - ветка, унаследованная от develop в которую вносятся точечные изменения, например добавление фичи, исправление бага. Тикет бранчи именуются с номером соответствующего тикета/задачи. Так можно по названию ветки найти задачу и понять цель изменений. И наоборот: по тикету найти ветку. В этой ветке может существовать частично нерабочее состояние репозитория.

## Полный классический Git flow

Screenshot 2024-10-05 211241.png

## Более простая схема для роли developer

Screenshot 2024-10-05 211246.png

## Процесс работы в feature-branch

Screenshot 2024-10-05 211252.png

## Безопасное удаление ветвей

Если неслитые ветки удалять через интерфейс гитлаба/гитхаба, то все изменения в этих ветвях навсегда будут утеряны. Чтобы такого не происходило, можно перед удалением

ветки навесить на неё тег: (например тег archive/назв.ветки)

```
git clone <url репозитория>    (если нет локальной копии)
cd <репозиторий>

git checkout <имя ветки для закрытия>
git tag archive/<имя ветки для закрытия> <имя ветки для закрытия>
git checkout <любая другая ветка>
git branch -D <имя ветки для закрытия>
git branch -d -r origin/<имя ветки для закрытия>
git push --tags
git push -d origin <имя ветки для закрытия>
```

# Как работает MR

На самом деле при merge/MR происходит чуть более сложная логика чем просто наложение diff разных коммитов последовательно друг на друга. Подробнее можно почитать здесь.

В настройках репозитория можно выбрать метод для merge:

Screenshot 2024-10-05 211819.png

# SSH

SSH

# Local port forwarding

## Command explanation

To forward port of remote host use the following command

```
ssh -L <LOCAL_PORT>:<TUNNEL_IP>:<DESTINATION_PORT> <user>@<server_ip>
```

-L stands for local adress because we are tunneling connection on local machine, TUNNEL_IP is 127.0.0.1

```
ssh -L <LOCAL_PORT>:127.0.0.1:<DESTINATION_PORT> <user>@<server_ip>
```

## Real life example

- I have a server with ip *69.69.69.69*
- I have user on a server called *gnumik*
- My public key is recognized by this server and I can establish ssh connection like this:

```
ssh gnumik@69.69.69.69
```

- I have postgresql data base running on the server on port *5432*
- I want to have access to this postgre instance from my local machine using local port *5433*
- I need to execute the following command on my local machine:

```
ssh -L 5433:127.0.0.1:5432 gnumik@69.69.69.69
```

***P.S.*** I use arch btw

SSH

# Скачивание и загрузка файлов по SSH

**Скачать с сервера**

```
scp -r -P 22 -i ~/.ssh/privkey user@192.168.1.7:/home/myuser/from "C:\localfolder"
```

**Залить на сервер**

```
scp -r -P 22 -i ~/.ssh/privkey "C:\localfolder" user@192.168.1.7:/home/myuser/to
```

В случае если нужно передать только один файл, уберите опцию `-r`

SSH

# SSH as proxy

**ssh-сервер в качестве прокси для браузера**

```
ssh -i ~/.ssh/privkey -p 22 myuser@192.168.1.7 -D 2141 -N


start chrome --proxy-server="socks5://localhost:2141"
```

# SAMBA

Файловый сервер и сетевые диски

SAMBA

# Монтирование самбы на Ubuntu server

Чтобы подключить удаленную папку через SMB (версии 2 и выше) к Ubuntu с помощью консоли Bash, выполните следующие шаги:

## 1. Установите необходимые пакеты

```
sudo apt update
sudo apt install cifs-utils
```

## 2. Создайте точку монтирования

Создайте папку, куда будет монтироваться удаленный ресурс:

```
sudo mkdir -p /mnt/smb-share
```

## 3. Подключите удаленную папку

Используйте команду `mount` для подключения:

```
sudo mount -t cifs -o username=USERNAME,password=PASSWORD,vers=2.0 //SERVER_ADDRESS/SHARE_NAME /mnt/smb-share
```

- `USERNAME` — имя пользователя для доступа к SMB-ресурсу.
- `PASSWORD` — пароль пользователя.
- `SERVER_ADDRESS` — IP-адрес или имя хоста SMB-сервера.
- `SHARE_NAME` — имя общего ресурса на сервере.
- `vers=2.0` — указание версии протокола SMB (можно заменить на `vers=3.0`, если требуется версия 3).

Если вы хотите ввести пароль вручную или оставить его в безопасности, не указывайте его в команде, и вам будет предложено ввести его после выполнения команды.

# 4. Проверьте подключение

Проверьте доступ к монтированной папке:

```
ls /mnt/smb-share
```

# 5. Автоматическое монтирование (опционально)

Чтобы подключение выполнялось автоматически при загрузке системы:

1. Откройте файл `/etc/fstab` для редактирования:

   ```
   sudo nano /etc/fstab
   ```

2. Добавьте строку:

   ```
   //SERVER_ADDRESS/SHARE_NAME /mnt/smb-share cifs
   username=USERNAME,password=PASSWORD,vers=2.0 0 0
   ```

3. Сохраните файл и примонтируйте ресурсы:

   ```
   sudo mount -a
   ```

Если хотите хранить пароль отдельно, создайте файл `/etc/samba/credentials`:

```
sudo nano /etc/samba/credentials
```

Содержимое:

```
username=USERNAME
password=PASSWORD
```

Защитите файл:

```
sudo chmod 600 /etc/samba/credentials
```

И обновите запись в `/etc/fstab`:

```
//SERVER_ADDRESS/SHARE_NAME /mnt/smb-share cifs credentials=/etc/samba/credentials,vers=2.0 0 0
```

Теперь монтирование будет происходить безопасно и автоматически.

# Авто

Чтобы монтирование SMB-папки выполнялось автоматически при загрузке системы, выполните следующие шаги:

---

# 1. **Настройка файла `/etc/fstab`**

Откройте файл `/etc/fstab` для редактирования:

```
sudo nano /etc/fstab
```

Добавьте строку в конец файла:

```
//SERVER_ADDRESS/SHARE_NAME /mnt/smb-share cifs
credentials=/etc/samba/credentials,vers=2.0,iocharset=utf8 0 0
```

## Расшифровка параметров:

- `//SERVER_ADDRESS/SHARE_NAME` — путь к SMB-ресурсу (например, `//192.168.1.10/share`).
- `/mnt/smb-share` — локальная точка монтирования.
- `cifs` — тип файловой системы.
- `credentials=/etc/samba/credentials` — путь к файлу с учетными данными.
- `vers=2.0` — версия протокола SMB (можно использовать `vers=3.0`, если сервер поддерживает SMBv3).
- `iocharset=utf8` — кодировка для корректного отображения имен файлов.
- `0 0` — параметры для автоматического монтирования.

---

# 2. **Создание файла с учетными данными**

Создайте файл для хранения имени пользователя и пароля:

```
sudo nano /etc/samba/credentials
```

Добавьте следующие строки:

```
username=USERNAME
password=PASSWORD
```

Сохраните файл и закройте редактор.

### Защитите файл от несанкционированного доступа:

```
sudo chmod 600 /etc/samba/credentials
```

---

# 3. **Проверка монтирования**

После настройки, выполните команду:

```
sudo mount -a
```

Если папка примонтировалась без ошибок, настройка выполнена правильно.

---

# 4. **Перезагрузка системы**

Перезагрузите компьютер, чтобы убедиться, что папка монтируется автоматически:

```
sudo reboot
```

После загрузки проверьте содержимое точки монтирования:

```
ls /mnt/smb-share
```

---

Теперь папка будет автоматически подключаться при каждой загрузке системы.

# Databases

Базы данных

Databases

# Postgres database dump and restore

В этой короткой инструкции описывается как сделать бэкап базы данных PostgreSQL и восстановить его (на другом сервере);

Предполагается что PostgreSQL развёрнута в docker контейнере.

## Dump

! Осторожно, если для докер-контейнера с БД не настроен volume который сохраняет состояние базы данных при перезапуске контейнера, то есть шанс потерять всё. (Но надюсь он настроен, потому что иначе жить нельзя).

Для начала добавьте в `docker-compose.yml` в раздел volumes новый volume чтобы было удобно забрать с сервера дамп базы данных.

```
  volumes:
    - ./dumps/:/dumps/
```

Преезапустите контейнер:

```
docker compose down
docker compose up -d
```

Войдите внутрь контейнера с помощью команды (Выйти из контейнера: `ctrl+d`):

```
docker exec -it postgres_db bash
```

Сделайте дамп вашей базы данных

```
pg_dump database_name > dumps/db.sql
```

## Перенос

**Старый сервер.** Файл будет создан в примонтированной директории `dumps`. Перенос файла дампа на новый сервер можно выполнить любым удобным способом, scp или winSCP (GUI для

scp).

**Новый сервер.** Проделайте те же действия с `docker-compose.yml` для создания volume и перезапустите контейнер. В директории рядом с `docker-compose.yml` должна создастся папка `dumps`. Перенесите файл дампа туда.

# Restore

Войдите внутрь контейнера с помощью команды (Выйти из контейнера: `ctrl+d`):

```
docker exec -it postgres_db bash
```

Тут существует два варианта.

- Вариант 1. Если база данных чистая, то выполните команду:

  ```
  psql -d database_name < dumps/db.sql
  ```

- Вариант 2. Если база данных не чистая или вы сделали что-то не так и psql не даёт восстановить из дампа. Самый простой способ дропнуть базу и создать её заного.

  ```
  psql -c "DROP DATABASE database_name;"
  psql -c "CREATE DATABASE database_name;"
  ```

  После чего загрузите базу данных из дампа:

  ```
  psql -d database_name < dumps/db.sql
  ```

Поздравляю, вы великолепны!

# DB Zoo

databases.jpg

# WireShark

Wireshark-cheatsheet-1.png

Wireshark-cheatsheet-2.png

# ffmpeg/ffprobe

1. Trim video
   `ffmpeg -i input.mp4  -ss 00:00:00 -to 00:00:15 -c:v copy -c:a copy output.mp4`
   This command trim video from 00:00:00 to 00:00:15 (final duration = 15 sec).
   The `-c:v copy -c:a copy` commands copy the original audio and video without re-encoding.
   **CAUTION!** This way may cause add black frames in the head of a video.
   **How to deal with the black screen**
   The black screen at the beginning of your video is bound to spoil this lovely story. Here's how to avoid a black screen
   Use the "-avoid_negative_ts make_zero"  parameter
   What you do here is to pass that part that reads "zero" to avoid any negative section. i.e "make_zero" to "avoid_negative_ts"
   Hence, the video timestamp will be shifted to "0". So, if the time you specified didn't fall into the rightful place, it will automatically be corrected to begin at the "start" point.
   `ffmpeg -i input.mp4  -ss 00:00:15 -to 00:00:20 -c:v copy -c:a copy -avoid_negative_ts make_zero output.mp4`

   Another way to deal with black screen problem is replace seek parameter
   `ffmpeg -ss 00:00:15 -i input.mov -to 00:00:20 -c copy -avoid_negative_ts make_zero output.mov`
   So, to workaround this effect run a command with no copy video and audio. For example:
   `ffmpeg -i input.mp4 -ss 00:00:00 -to 00:00:15 output.mp4`
2. **Removing the first seconds** `ffmpeg -ss 00:05 -i input.mp4 -c:v libx264 -c:a aac output.mp4`

**Removing the first seconds without re-encoding** `ffmpeg -ss 00:05 -i input.mp4 -c:v copy -c:a copy output.mp4`

3. Speed up video
   `ffmpeg -i input.mp4 -r 60 -vf "setpts=0.5*PTS" output.mp4`
4. Shrink big MP4 video file to smaller sizes
   - **3.6 Gb** to **556 Mb**, great quality □□
   `ffmpeg -i input.mp4 -vcodec h264 -acodec mp2 output.mp4`
   - **3.6 Gb** to **62 Mb**, quality "good enough"/acceptable □□
   `ffmpeg -i input.mp4 -s 1280x720 -acodec copy -y output.mp4`
   - **3.6 Gb** to **30 Mb**, very shitty quality □□
   `ffmpeg -i input.mp4 -vcodec h264 -b:v 1000k -acodec mp3 output.mp4`
   - CPU transcoding to 720p with hevc codec
   `ffmpeg -i input.mov -s 1280x720 -vcodec hevc -acodec copy -y output_720p.mov`
   - GPU transcoding to 720p with hevc codec
   `ffmpeg -y -vsync 0 -hwaccel cuda -hwaccel_output_format cuda -i input.mov -vf scale_cuda=1280:720 -c:a copy -c:v hevc_nvenc output_720p_hevc_nvenc.mov`
5. Create video from images

   MacOS: Add pixel format.
   `ffmpeg -framerate 10 -i filename-%03d.jpg -pix_fmt yuv420p output.mp4`

`ffmpeg -framerate 10 -pattern_type  glob -i "filename-*.jpg" -pix_fmt yuv420p output.mp4`

6. Concatenate videos
   Create txt file with list of videos:
   `for f in *.mov; do echo "file '$f'" >> videos.txt; done`
   Concat videos:
   `ffmpeg -f concat -i videos.txt -c copy output_[concated.mov](http://concated.mov/)`
7. Get 1 frame from video
   1. With the seek option
      `ffmpeg -ss 00:00:01 -i input.mov -frames:v 1 output.jpeg`
   2. With select filter
      `ffmpeg -i input.mov -vf "select=eq(n\,34)" -frames:v 1 output.jpeg`
8. Get every N frame from video
   - It will get `1 frame every 10 seconds` instead of `1 frame every 1 second ffmpeg -i out1.avi -r 0.1 -f image2 image-%3d.jpeg`
   - If you really what every 10th frame from video then you can use `select` with `modulo 10` `ffmpeg -i out1.mp4 -vf "select=not(mod(n\,10))" -vsync vfr image_%03d.jpg`
9. Get fps from video
   `ffprobe -v error -select_streams v:0 -show_entries stream=r_frame_rate -of default=noprint_wrappers=1:nokey=1 input/training_sessions/140048_15sec/140048_4_2023-05-24_15.14.42_15sec_sync.mov | bc -l`
10. Convert mov to mp4 `ffmpeg -i input.mov -crf 0 output.mp4`
11. Obtain certain information about video file `ffprobe -v error -select_streams v:0 -show_entries stream=width,height,duration,bit_rate -of default=noprint_wrappers=1 input.mp4`
12. Convert mp3 to wav `ffmpeg -i file.mp3 -acodec pcm_s16le -ac 1 -ar 22050 file.wav` This command generate wav file with shorter duration
13. Show audio file info `ffmpeg -i file.wav -hide_banner`
14. **How to generate a sine wave with ffmpeg?** `ffmpeg -f lavfi -i "sine=frequency=15000:sample_rate=48000:duration=0.5" -c:a mp3 -b:a 192k -af "volume=40dB" 15kHz40.mp3`
15. **Changing volume**

To change the audio volume, you may use FFmpeg's volume audio filter.

If we want our volume to be half of the input volume:

`ffmpeg -i input.wav -filter:a "volume=0.5" output.wav`

150% of current volume:

`ffmpeg -i input.wav -filter:a "volume=1.5" output.wav`

You can also use decibel measures. To increase the volume by 10dB:

`ffmpeg -i input.wav -filter:a "volume=10dB" output.wav`

To reduce the volume, use a negative value:

`ffmpeg -i input.wav -filter:a "volume=-5dB" output.wav`

Note that the `volume` filter only **adjusts** the volume. It does not **set** the volume. To set or otherwise normalize the volume of a stream, see the sections below.

16. Concat two audio files `ffmpeg -i file1.mp3 -i file2.mp3 -filter_complex "[0:a][1:a]concat=n=2:v=0:a=1" final.mp3`
17. Encode a high quality MP3 or MP4 audio from a movie file `ffmpeg -i sample.avi -q:a 0 -map a sample.mp3`
18. Trim audio file
    `ffmpeg -i a.mp3 -ss 00:00:00 -to 00:05:00 b.mp3`