1 Arch Linux

1.1 Mantainance

```
#check file size
du -sh .cache/
#remove a file
rm -rt .cache/
#delete what you don't need in .config file
```

specific mantainance:

```
#check the failed systems
systemctl --failed
#check the systemd journal
sudo journalctl -p 3-xb
#if the system doesn't boots then ctrl+alt+shift then timeshift -restore
#then update mirrors
#clar chache

#then to update the whole system use:
sudo pacman -Syyu
#to check system updates
sudo pacman -Syu
#if you wan't to remove all packages in the drive use
sudo pacman -Scc
#remove all unwanted dependencies
paru -Yc
#remove orphan packages
sudo pacman -Rns \$(pacman - Qdtq)
#sudo pacman -Syyy Syncrhonise data use "mirror1"
```

1.2 Print in arch linux

install packages: usbutils, lsusb, cups
use this to make cups usable

```
sudo systemct enable cups
sudo systemctl start cups
localhost:631

lp -d HP_Officejey_Pro_8600]
```

1.3 configure date and time

```
hwclock --set --date = "04/32/2021 19:00:00"
hwclock -hctosys
```

1.4 Configure wireless

```
#when entering an iso
iwctl
#then in the ui

#to list all available devices
device list
```

```
#to scan networks
station <device> scan

#to get newworks
station <device> get-network

#to connect to a network
station <device> connect "<name of network>"

#to check if the connection is staable
ping -c s 8.8.8.8

#don't forget before rebooting the iso run
pacman nmtui
```

from Arch Water Linux

```
# to acces the gui for the internet
nmtui
# solve temporary failure in name resolution
# change the /etc/resolve.conf file to nameserver 8.8.8.8

# restart the resolved daemon
sudo systemctl restart systemd-resolved.service
# check that the daemon is running and active
sudo systemctl status systemd-resolved.service
```

dwm basic configuration

```
#MODKEY + shift + q to restart X server
startx # to start the X server
```

1.5 mount devices

mount usb sticks:

```
#to mount a usb stick
mount /dev/sdb1 /mnt/<destination folder>
#to unmount a sub stick
umount /dev/sdb1
```

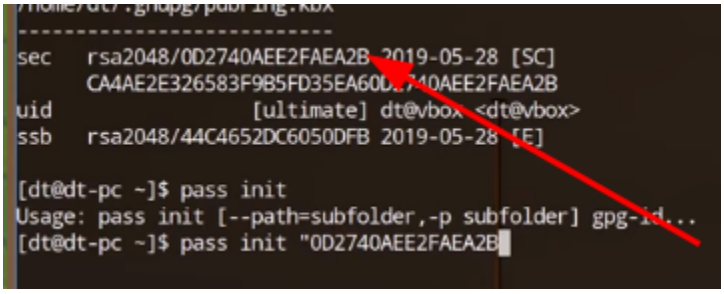
mount an android device:

```
#to mount and android device
simple-mtpfs --device 1 tablet/

#to unmount an android device
fusermount -u /tablet
```

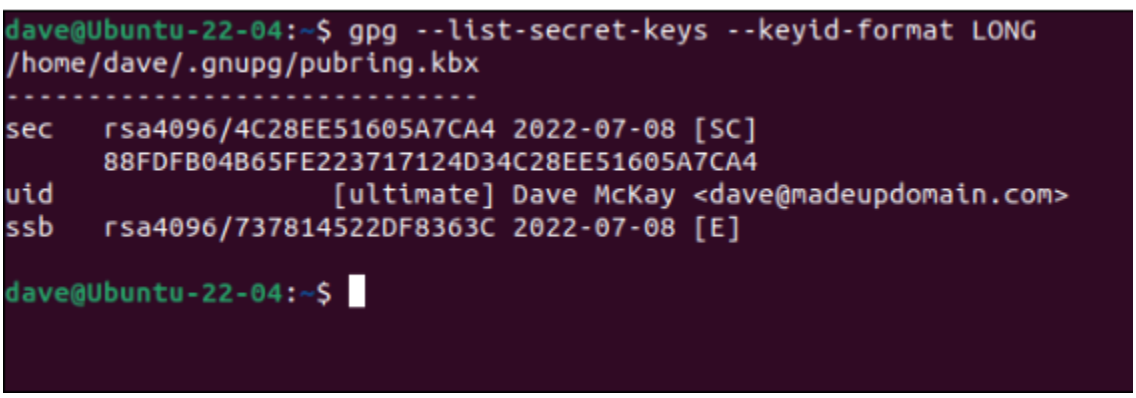
1.6 import export passwords from pass

export passwords:

```
# to list first the gpg keys
gpg --list-secret-keys --keyid-format LONG
```




```
# list key files
tree ~/.gnupg
# list the keys in long format
gpg --list-secret-keys --keyid-format LONG
```



list keys command output

- The "sec" (secret) line shows the number of bits in the encryption (4096 in this example), the key ID, the date the key was created, and "[SC]." The "S" means the key can be used for digital signatures and the "C" means it can be used for certification.
- The next line is the key fingerprint.
- The "uid" line holds the ID of the key's owner.
- The "ssb" line shows the secret subkey, when it was created, and "E." The "E" indicates it can be used for encryption.

```
# export public key
gpg --export --export-options backup --output public.gpg
# if you want to back up only the ones associated to one identity use
gpg --export --export-options backup --output public.gpg dave@madeupdomain.com
# export the secret keys
gpg --export-secret-keys --export-options backup --output private.gpg
# export the secret keys for one identity
gpg --export-secret-keys --export-options backup --output private.gpg dave@madeupdomain
# export ownerwhip trust
gpg --export-ownertrust > trust.gpg
```

```
# in a new computer
# import public keys
gpg --import public.gpg
# import private keys
gpg --import private.gpg
# import ownerwhip trust
gpg --import-ownertrust trust.gpg


# and that's it!
```

2 Install python version

```
# download the python version you need from https://www.python.org/downloads/source/
# unpack in the .local/src/pythonversions/pythonVersion.tgz
tar zxvf pythonVersion.tgz
cd pythonVersion
# Install the python version
./configure
make
sudo make install
make clean
# check python version
python[python_version] --version
# create a python environment using that python version
python[python_version] -m venv venv/
# source the environment
source venv/bin/activate
# for deactivating
deactivate
```

2.1 removing bloatware from android

```
# install the android developer tools
paru -S android-tools
# in your android enable developer options by about phone -> build number 7 times
# then enable usb debugging

# now in your linux sistem type in your terminal
adb devices # to see if device is succesfully connected
adb shell # to start the shell

# to delete an app
pm uninstall -k --user -0 (package-name)

# to see the names of apps use app inspector from the google store
```

3 deploy multiple applications to ec2 ubuntu instance

```
sudo apt install nginx
sudo apt install neovim
```

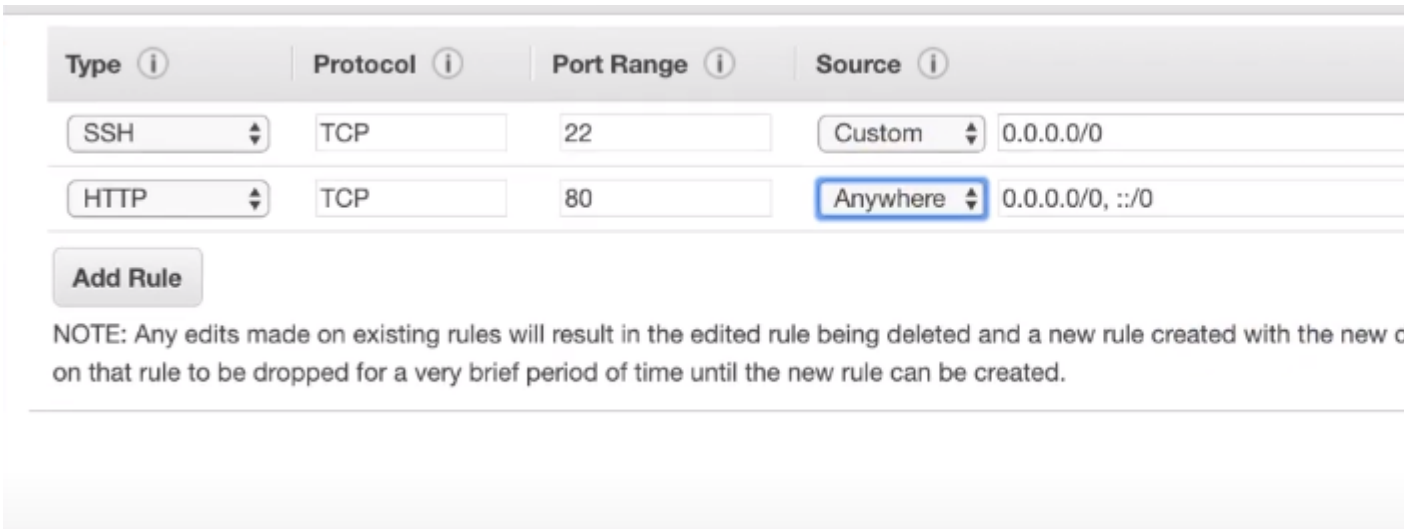
```
sudo mkdir -p /var/www/domainName.extension/html
sudo chown -R \$USER:\$USER /var/www/domainName.extension/html
sudo chmod -R 755 /var/www/domainName.extension


# custom edit the html
sudo nvim /var/www/domainName.extension/html/index.html


# edit the configuration
sudo nvim /etc/nginx/sites-available/domainName.extension

########## /etc/nginx/sites-available/domainName.extension ######
server \{
        listen 80;
        listen [::]:80;

        root /var/www/domainName.extension/html;
        index index.html index.htm index.nginx-debian.html;

        server_name domainName.extension www.domainName.extension;

        location / \{
                try_files \$uri \$uri/ =404;
        }

}
##########  ############

sudo ln -s /etc/nginx/sites-available/domainName.extension /etc/nginx/sites-enabled/

# edit the configuration
sudo nvim /etc/nginx.conf
# uncomment the server_names_hash_bucket_size 64;

# test the config
sudo nginx -t

# reestart the daemon
systemctl restart nginx
```

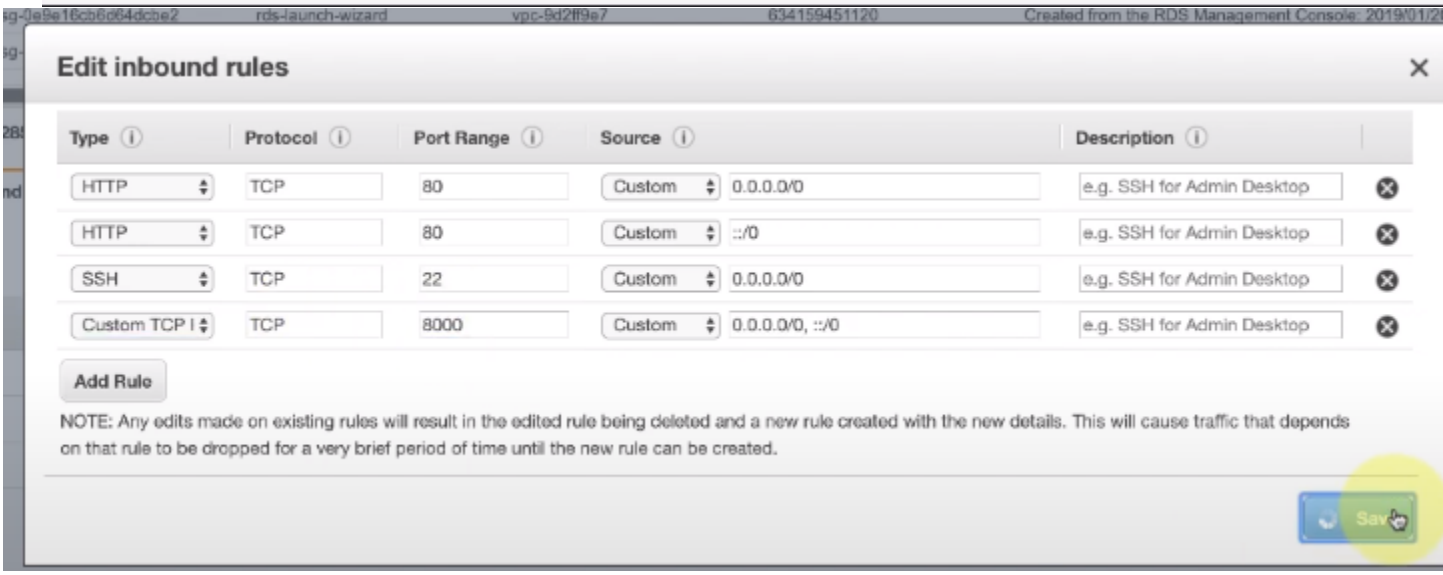
3.1 deploy python django aplication aws

```
## modify the django project

# settings.py

STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR,'static')
```

```
        MEDIA_URL = '/media/'
        MEDIA_ROOT = os.path.join(BASE_DIR,'media')


        ## now collect static
        source venv/bin/activate

        python manage.py collectstatic



        ### aws
# create an account in aws
# find ec2 and click on launch instance
# select ubunto server 18 free trial
# see all the instances you are running
# change the name of your instance (on the very left side of the row you can do that)
# you will be prompted to create a new id, so create the new keypair and save it to your linux machine
# the right click on the isntance id and click connect
# paste the ssh code in the folder were your keypair is, for example:
ssh -i "password-generator-django.pem" ubuntu@ec2-54-242-121-76.compute-1.amazonaws.com

# now that you are connected sudo update and upgrade the server
sudo apt-get update
sudo apt-get upgrade -y

# you will have to use gnix gnunicorn
python3 --version
python3 -m venv venv/
apt-get install python3-venv
python3 -m venv venv/

ls
# use the environment
source venv/bin/activate

# install django
pip3 install django


# install necessary packages for python
sudo apt install python3-dev buil-essential
sudo apt install libssl
sudo aptinstall libssl-dev
sudo apt install libmysqlcient-dev


# install the requirements
pip install -r requirements.txt

# install django-ckeditor
pip3 install django-ckeditor

# git clone your github django project
```

```
git clone url.git

cd url

# install modules for deploy
pip3 install gunicorn
sudo apt-get install -y nginx

# start nginx
sudo nginx

# configure security groups in for https and http
right click(on the instance row) -> networking -> change security group -> see the launch wizar asociat

click security groups -> launch wizard n -> inbound -> edit -> add rule
```






```
# to connect the gunicorn (which is the wsgi interface
```

```
# tell gunicorn to use the wsgi.py
cd project/project
gunicorn --bind 0.0.0.0:8000 project.wsgi:application
# after this you sould be able to acces your web app using the link
# use the port 8000 example: google.com:8000

# supervisor makes sure your application is running always
sudo pat-get install -y supervisor
# create a configuration for supervisor
cd /etc/supervisor/conf.d/
touch gunicorn.conf
sudo touch gunicorn.conf

# edit it
sudo nvim gunicorn.conf

# inside the file
[program:gunicorn]
directory=/home/ubuntu/project
command=/home/ubuntu/env/bin/gunicorn --workers 3 --bind unix:/home/ubuntu/project/app.sock
        testproject.wsgi:application

autostart=true
autorestart=true
stderr_logfile=/var/log/gunicorn.err.log
stdout_logfile=/var/log/gunicorn.out.log

[group:guni]
programs:gunicorn

# save and exti
# outside the file
sudo mkdir /var/log/gunicorn
sudo supervisorctl reread

sudo supervisorctl update

sudo supervisorctl status

# nginx configuration
cd ~
cd /etc/nginx/sites-available
sudo touch django.conf
sudo nvim django.conf

# paste the code

        server{
                listen 80;
                server_name ec2-3-91-188-252.compute-1.amazonaws.com;
                location / {
                        include proxy_params;
                        proxy_pass http://unix:/home/ubuntu/personal_portfolio/app.sock;
                }
```

```
                location /static/ {
                        autoindex on;
                        alias /home/ubuntu/personal_portfolio/static/;
                }
                location /media/ {
                        root /home/ubuntu/personal_portfolio/;
                }
                }


# save and exit
sudo nginx -t

# enable the link
sudo ln django.conf /etc/nginx/sites-enabled/


# save and exit
sudo nginx -t

sudo service nginx restart









######
## now for setting static files
#####

# in settings.py
STATIC_URL = '/static/'

# in the html you are doing
{% load staticfiles %}

# in the templates section
'DIRS':[os.path.join(BASE_DIR,'TestProject/templates')],

# open the server with the keys and cd
nvim /etc/nginx/sites-enabled/django.conf

# append

location /static/ {
        autoindex on;
        alias /home/ubuntu/ProjectFolder/MainProjectFolder/static/;
}

# outside the file
# open the nginx configuration to allow big pictures
```

```
cd /etc/nginx
sudo vim nginx.conf

#inside the http or server paste
client_max_body_size 4M;



sudo systemctl reload nginx ;


######
## now for setup the database with django
#####


# create database
dtaabase section ->
RDS ->
create database ->
mysql ->
only enable options eligible for free usage ->
next ->
select the specific mysql version ->
database instance offered on the free tier ->
allocated storage
configure the name,username,password etc
allow public accesibility
choose default existing vpc security groups
database name

# now get the latest code in your github
git pull
# configure the database for the server
DATABASES = {
        # name
        'default' : {
                'ENGINE':"django.db.backends.mysql",
                'NAME':"database_name",
                'USER':"database_user",
                # change this manually in the server
                "PASSWORD":"********",
                # click on the database, check endpoint & port for configuring
```

```
                'HOST','host',
                'PORT':'12312'
        }
}
# activate the environment
source venv/bin/activate

# install all the modules in the requirements
pip install django-mysql

# make the migrations
python manage.py makemigrations name_of_main_app
python manage.py migrate

# reestart the server
sudo supervisorctl reload

sudo service nginx restart
```

3.2 use nvim for graphical programs

```
# first you have to make it the default
# in .zshrc put
VISUAL=nvim
EDITOR=nvim

# then change the nvim.desktop in /usr/share/applications/nvim.desktop
and place
EXEC=nvimWrapper
TRY=nvimWrapper %F

# now create the wrapper in the bin like
st -e sh -c "nvim \$1"
```