Command line tips and tricks

- Apt / apt-get
- Bash tools
- Docker
- Git
- IPtables
- macOS / OS X
- macOS software setup
- Mounting
- Network
- OpenSSL
- Pkcs11-tool
- PKI
- Resolve / DNS
- Skype
- SQL
- ssh, bash, vim configs
- Tips & Tricks
- Vagrant
- Windows
- GPG

Apt / apt-get

Install error

```
Unpacking xroad-signer (7.6.2-1.ubuntu22.04) over (7.5.1-1.ubuntu22.04) ...
dpkg: error processing archive /var/cache/apt/archives/xroad-signer_7.6.2-
1.ubuntu22.04_amd64.deb (--unpack):
unable to open '/etc/xroad/conf.d/signer-console-logback.xml.dpkg-new': No such file or
directory
No apport report written because the error message indicates an issue on the local system
```

Solution

```
rm -rf  /var/cache/apt/archives/xroad-signer_7.6.2-1.ubuntu22.04_amd64.deb
apt --fix-broken install
```

Find out from which deb package a file is

```
# dpkg -S p11tool
gnutls-bin: /usr/bin/p11tool
gnutls-bin: /usr/share/man/man1/p11tool.1.gz
```

Show info about the package

`apt show <package-name>`

List available updates

`apt list --upgradable`

Do everything

`apt update && apt dist-upgrade -y && apt autoremove -y && apt autoclean -y && lsof | grep lib | grep DEL && init 6`

List the package versions available from all your sources

`apt-cache madison <package name>`

Search package from local cache

`apt-cache search <package name>`

Bash tools

Awk

First and third column from file, seperated by spaces
`awk -F " " '{print $1,$3'} result.txt`

Print from line 156 to line 170
`awk 'NR >= 156 && NR <= 170' log.txt`

Cat

Copy Data to Other File
`cat file1 > file2`

Combine Files And Sort Alphabetically
`cat file1 file2 | sort > file3`

Create new file that consists of the contents of file1, followed by keyboard.
`cat file1 -> file2`

Multi line to singe line
`cat multiline.txt | tr -d '\n' > oneline.txt`

Cut

First and third column from file, seperated by spaces
`cut -d' ' -f1,3 result.txt`

Find

Search for a file
`find location -name test.zip`
`find / -name phppgadmin.conf`

Search filetype
`find / -i name "*.txt"`

Accessed less than 3m ago
`find / -amin -3`

Accessed two days ago
`find / -time -2`

Smaller than 5MB, bigger than 2MB
`find / -size -5M -and -size +2M`

Remove all .DS_Store files
`find . -name '*.DS_Store' -type f -delete`

Find files older than 7 days

```
find <dir> -type f -mtime +7
```

Mtime

+n More than *n*.
n Exactly *n*.
-n Less than *n*.

```
You can write -mtime 6 or -mtime -6 or -mtime +6:
Using 6 without sign means "equal to 6 days old – so modified between 'now - 6 * 86400' and
'now - 7 * 86400'" (because fractional days are discarded).
Using -6 means "less than 6 days old – so modified on or after 'now - 6 * 86400'".
Using +6 means "more than 6 days old – so modified on or before 'now - 7 * 86400'" (where the
7 is a little unexpected, perhaps).
```

Grep

Flag	Description
-i	Ignore Case
-l	Print name of each file which contains a match
-v	Return all lines which don't match the pattern
-c	Print a count of matching lines
-n	Print the line nr before each line that matches

List all files in directory where file name contains "test"
`ls | grep "test"`
`grep "" -l "test"`

To search for a line containing text hello.gif in any .txt files
`grep "hello\.gif" *.txt`

Exactly and solely "apple"
`grep -x "apple" list.txt`

Everything else but not "apple"
`grep -v "apple" fruitlist.txt`

Match insensitive
`grep -i "fruit" fruitlist.txt`

Match apple and Apple
`grep "[Aa]pple" fruitlist.txt`

Grep Terminal History
`history | grep "ionic"`

Output non duplicates
`grep -vFf file1 file2`

iPerf3

```
# on receiver
iperf3 -s -p 7575
# on client
iperf3 -c <receiver-ip> -p 7575
```

Links

Create symbolic link
`ln -s /source /destination`

Remove link
`unlink /destination/script.sh`

Netcat

Scan ports
`nc -vvvnz -w5 185.31.92.195 5577 5500`

Netstat

`sudo netstat -lntp`
l - listening ports
n - no hostnames
t - tcp
p - processes

SSH

ssh-key to machines
`ssh-copy-id <ip-or-hostname>`

Tar

Archive a directory / File
`tar -zcvf archive.tar.gz test/`

Unarchive

```
tar -zxvf archive.tar.gz
tar -zxvf archive.tar.gz -C <output-dir>
tar xopf thing.tar
```

TCP

`tcpdump -n -v -i eth0 -s 0 port 80 -w /home/$USER/tcpdump.pcap`
-n don't convert addresses to names
-v verbose
-i interface eth0
-s snapshot length

Users

Remove user from group

`passwd -d <username>`

Reload bash user
`. ~/.bash_profile`

Display all users or groups
`compgen -u`
`compgen -g`

Add user to sudoers
`sudo usermod -aG sudo qrl`

Zip

Compress a directory / File
`zip -r <test.zip> <folder/file>`

Decompress
`unzip <test.zip> -d <destination_folder>`

Wget

Save website

```
wget --mirror --convert-links --adjust-extension --page-requisites --reject-regex "report" --
no-warc-compression --warc-file="kalaralli" https://qrl.ee/kalaralli/
```

Docker

Read compose logs

```
docker-compose logs -f --tail=100
```

Volume in dir

```
- ./name:/name
# For macOS
- ${PWD}/name:/name
```

Force restart service with compose

`docker-compose up -d --force-recreate <service>`

Port forwarding

<host_port>:<container_port>

Build image

`docker build -t <docker-username>/<image-name>:<tag> .`

Push image to docker hub registry

`docker image push <docker-username>/image-name`

Docker-compose port forwarding

`host-port:container-port`

Rebuild container

`docker-compose up -d --no-deps --build --force-recreate <service_name>`
--no-deps - Don't start linked services
--build - Build images before starting containers
--no-cache
--force-recreate

Updating

update single image `docker-compose pull deluge`
update all images `docker-compose pull`
update single container `docker-compose up -d deluge`
update all necessary containers `docker-compose up -d`

remove unnecessary images `docker image prune`

Get container image version

`docker inspect <container name> | grep -i version`

Git

Push commits to another branch

```
git push origin <branch with new changes>:<branch you are pushing to>
```

List git config settings

`git config --list`

Change git user for single repository

`git config user.name "John Smith"`

Change git user email globally

`git config --global user.email "john@smith.com"`

Checkout to old commit

`git checkout <commit hash>`

Change git case sensitivity(default is true)

`git config core.ignorecase false`

Restore to certain commit

`git reset --hard f170a0c`
`git push -f`

Change origin URL

`git remote -v`
`git remote set-url origin <url>`

Reset one file

`git checkout HEAD -- <files>`

Clone specific branch

`git clone -b <branch> <repo-url>`

Show commit diff

`git show <commit-hash>`

IPtables

```
# src(me) -> dest
iptables -A OUTPUT -p tcp --match multiport --dports 80,443 -d 10.11.12.55 -j ACCEPT -m comment
--comment 'X-tee turvaserver'

# dst -> src(me)
iptables -A INPUT -p tcp --match multiport --dports 80,443 -s 10.11.12.55 -j ACCEPT -m comment --
comment 'X-tee turvaserver'

# list by line numbers
iptables -L --line-numbers

# reegli nr 24 kustutamine
iptables -D INPUT 24

# reegli lisamine reale 53
iptables -I OUTPUT 53 -p tcp --match multiport --dports 80,443 -d 10.11.12.55 -j ACCEPT -m
comment --comment 'X-tee turvaserver'
```

macOS / OS X

Mac install sshpass

```
brew tap esolitos/ipa
brew install sshpass
```

Vagrant bash completion

`brew install vagrant-completion`

add to .bash_profile:

```
if [ -f `brew --prefix`/etc/bash_completion.d/vagrant ]; then
    source `brew --prefix`/etc/bash_completion.d/vagrant
fi
```

Ignore .DS_Store globally

`echo ".DS_Store" > ~/.gitignore_global`
`git config --global core.excludesfile ~/.gitignore_global`

Update macOS DB File

`/usr/libexec/locate.updatedb`

Launch VS Code from Terminal

In VS Code do `⇧⌘P` and type `Shell Command: Install 'code' command in PATH`
In Terminal, reload user `source ~/.bash_profile`

Find process PID by port

`lsof -i tcp:8080`

Mac equivalent to netstat -lntp

`netstat -p tcp -van | grep LISTEN`

Reset Final Cut Pro Trial

`mv -v ~/Library/Application\ Support/.ffuserdata ~/.Trash`

macOS software setup

Install first

- docker - https://docs.docker.com/docker-for-mac/install/
- iterm2 - https://iterm2.com/
- fork - https://git-fork.com/
- homebrew - https://brew.sh/
- postman - https://www.postman.com/
- skype - https://www.skype.com/en/get-skype/
- skype for business - https://www.microsoft.com/en-us/download/details.aspx?id=54108
- visual studio code - https://code.visualstudio.com/download

Install with brew

ansible

`brew install ansible`

vagrant

`brew tap hashicorp/tap`
`brew install vagrant`

ansible-lint

`brew install ansible-lint`

bash-completion

https://sourabhbajaj.com/mac-setup/BashCompletion/

vagrant-completion

`brew install vagrant-completion`

```
if [ -f `brew --prefix`/etc/bash_completion.d/vagrant ]; then
    source `brew --prefix`/etc/bash_completion.d/vagrant
fi
```

sshpass

`brew tap esolitos/ipa`
`brew install sshpass`

terraform and terragrunt

`brew install terraform terragrunt`

htop

brew install htop

wget

brew install wget

whois

brew install whois

Install from App Store

- rocketchat

Backup

```
export PC_NAME=nimi && \
cd ~ && \
tar -zcf $PC_NAME-downloads-`date +"%Y-%m-%d_%H-%M"`.tar.gz ~/Downloads/ && \
tar -zcf $PC_NAME-desktop-`date +"%Y-%m-%d_%H-%M"`.tar.gz ~/Desktop/ && \
tar -zcf $PC_NAME-lab-`date +"%Y-%m-%d_%H-%M"`.tar.gz ~/LAB/
```

Mounting

Get info of the disk(s)

```
fdisk -l
ls -l /dev/disk/by-uuid/
```

Manual USB mount

```
mount /dev/sdb1 /mnt/usb-ssd -o uid=qrl,gid=qrl,iocharset=utf8,nofail
```

Auto USB mount (/etc/fstab)

```
UUID=6403-240D /mnt/usb-ssd exfat uid=qrl,gid=qrl,iocharset=utf8,nofail 0 0
```

Auto NAS mount (/etc/fstab)

```
//192.168.1.109/usbdisk1 /mnt/nas-usb cifs
username=qrl,password=qrl,iocharset=utf8,uid=1001,gid=1002,nofail 0 0
```

Network

Newer Debian

```
resolvectl status
service systemd-resolved restart
```

Older Debian

```
systemd-resolve --status
systemctl restart systemd-resolved.service
```

OpenSSL

Get sha256 fingerprint of cert

```
openssl x509 -noout -sha256 -fingerprint -in <file>
```

Utf8 output

```
-nameopt utf8
```

Convert RSA DER public key to PEM

```
openssl rsa -pubin -inform DER -in slot3_pub.key -outform PEM -out slot3_pub.pem
```

Convert DER certificate to PEM

```
openssl x509 -inform der -in cert.der -out cert.pem
```

Convert DER CSR to PEM

```
openssl req -inform der -in der-csr.csr -out pem-csr.pem
```

Get pub key of CSR

```
openssl req -pubkey -noout -in <csr-file>.csr
```

Get pub key from certificate

```
openssl x509 -pubkey -noout -in <cert-file>.crt
```

Modulus of Pub Key

```
openssl rsa -noout -text -modulus -in [public_key_filename]
```

Modulus of Certificate

```
openssl x509 -noout -text -modulus -in [certificate_filename]
```

View chained certificate

```
openssl storeutl -text -noout -certs CAFile.pem
```

View CSR

```
openssl req -noout -text -in <csr-file>.csr
```

View PEM certificate

```
openssl x509 -in cert.pem -text -noout
```

View RSA public key

```
openssl rsa -pubin -in slot3_pub.pem -text
```

Pkcs11-tool

Install pkcs11-tool

```
apt install opensc
```

Get hsm info

```
pkcs11-tool --module /opt/nfast/toolkits/pkcs11/libcknfast.so --show-info
```

List available slots

```
pkcs11-tool --module /opt/nfast/toolkits/pkcs11/libcknfast.so --list-slots
```

List objects on specified slot

```
pkcs11-tool --module /opt/nfast/toolkits/pkcs11/libcknfast.so --list-objects --slot 0x1f655491
```

List certificates on specified slot

```
pkcs11-tool --module /opt/nfast/toolkits/pkcs11/libcknfast.so --list-objects --slot 0x1f655491
--type cert
```

Export public key

```
pkcs11-tool --module /usr/lib/libcs_pkcs11_R2.so --login --slot 3 --read-object --type pubkey
--id <id> --output-file slot3_pub.key
```

Export certificate(needs partitiob/slot password)

```
pkcs11-tool --module /opt/nfast/toolkits/pkcs11/libcknfast.so --login --slot 0x1f55492 --read-
object --type cert --id <id> --output-file cert.der
```

Thales-Other

```
pkcs11-tool --module /usr/lib/libCryptoki2_64.so --list-slots
pkcs11-tool --module /usr/lib/libCryptoki2_64.so --list-objects --login --login-type user --
token-label <label>
pkcs11-tool --module /usr/lib/libCryptoki2_64.so --list-objects -r --type cert -l --login-type
user --token-label <label>
```

PKI

PKI - public key infrastructure

Signer
- Kasutades x hash algoritmi, nt SHA luuakse andmete/data põhjal hash
- Kasutades privaatset võtit ja x krüptograafia algoritmi, nt RSA, loodud hash krüpteeritakse
- Kirja pannakse ka täpne aeg, millal hash loodi/krüpteeriti ehk ajatempel
- Kaasa pannakse sertifikaat, mis tõestab, et antud privaatse võtme omanik on see, kes ta väidab, et ta on. Samuti sisaldab sertifikaat signeerija avalikku võtit
- Sertifikaat saadakse nt CA avalikust LDAP-st
- Originaal andmetega/dataga pannakse kaasa 1) krüpteeritud hash 2) täpne aeg 3) sertifikaat

Verifier
- Kasutades avalikku/pub võtit sertifikaadi seest ja x krüptograafia algoritmi, edastatud hash dekrüpteeritakse
- Kasutades x hash algoritmi, luuakse andmete/data põhjal hash
- Võrreldakse andmete/data põhjal loodud hashi ja dekrüpteeritud hashi. Kui need on samad, on allkiri õige ehk esialgset datat/andmeid ei ole muudetud

Eestis signeerides id-kaardi, m-id või smart-id’ga võetakse isiku sertifikaat avalikust LDAP-st kuna sertifikaat ei sisalda isiklikku ega tundliku infot.

PKI involves using a digital certificate for identity verification.

Root cert
Intermediate cert
End user/entity cert

Certificate
The certificate is used to confirm that the public key belongs to the specific organization or identity.
- issued by a CA
- contains the public key for a digital signature and specifies the identity associated with the key, such as the name of an organization
- CA acts as the guarantor
- Digital certificates must be issued by a trusted authority and are only valid for a specified time. They are required in order to create a digital signature
-

SHA - hash algo
one way function, turning an input of any size into a fixed-length output

RSA - encrypt algo
encrypting an input into an output that can then be decrypted

It uses a different key for encryption (the public one) than for decryption (the private one). This can therefore be used to receive encrypted messages from others - you can publish your public key, but only you with the private key can then decrypt the messages that have been encrypted with it.

If you reverse the keys for RSA, it can be used to generate a digital signature - by encrypting something with your private key, anyone can decrypt it with the public key and, if they are sure the public key belongs to you, then they have confidence that you were the one who encrypted the original. This is normally done in conjunction with a hash function - you hash your input, then encrypt that with your private key, giving a digital signature of a fixed length for your input message.

One more note is that hash algorithms, like SHA-1, can compute digests given data of any length as input. Asymmetric algorithms, like RSA, are limited in the length of data they can transform. For that reason, the original message is rarely signed with RSA, and instead the SHA-1 digest of the original message is signed. The recipient of the message and its signature computes the SHA-1 digest of the message, then decrypts the signature with the sender's public key and verifies that the digests exactly match

The keys work inversely to each other. Encrypted something with your public key? Decrypt it with your private key. Conversely, if you encrypted something with your private key, you decrypt it with your public. Such is the nature of asymmetric cryptography.

Encryption with the private key is used to prove authenticity. If person 1 encrypts a message with their own private key then person 2 can decrypt it with person 1's public key, which proves that person 1 originated the message since it could only have been encrypted with their private key.

https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq
https://stackoverflow.com/questions/733692/sha1-vs-rsa-whats-the-difference-between-them

Resolve / DNS

Ubuntu 22

```
resolvectl status
```

Ubuntu 18

```
systemd-resolve --status
```

Skype

List all members & roles in a chat

`/showmembers`

Make chat history visible to everyone

`/set options +HISTORY_DISCLOSED`

Change user roles in a chat

`/setrole johnsmith MASTER`

SQL

PSQL get version

```
psql -h <IP/DNS> -U <user> -d <database> -c 'SELECT VERSION()'
```

PSQL test connection

Install packages postgresql-client-common and postgresql-client-10

```
psql -h <host> -p <port> -U <username> -d <database> -c '\c'
```

Database size

```
SELECT pg_size_pretty( pg_database_size('<database>') );
```

Table sizes

```
select table_name, pg_size_pretty( pg_relation_size(quote_ident(table_name)) )
from information_schema.tables
where table_schema = 'public'
order by pg_relation_size(quote_ident(table_name)) desc;
```

Mysql change user password

```
mysql -u username-p
use mysql;
update user set password=PASSWORD('your_new_password') where User='username';
flush privileges;
quit
```

Postgres automatically fit records to the width of the screen(add to ~/.psqlrc to start psql with it every time)

```
\x auto
```

Postgres query from bash

```
psql -U <username> -d <database> -c "select * from ppl;"
```

Mssql list all tables

```
select * from sys.all_objects where type = 'U'
```

Oracle list all users

```
select username from dba_users
```

ssh, bash, vim configs

SSH

```
# disable host key check
Host *
    StrictHostKeyChecking no
    ServerAliveInterval 30

Host test123
  User usr
  Hostname qrl.ee
  Port 2222
  IdentityFile ~/.ssh/my-priv-key.pem
  LocalForward 4000 localhost:4000
```

BASH

```
alias ll='ls -larthF'
HISTCONTROL=erasedups
HISTFILESIZE=
HISTSIZE=
```

VIM .vimrc

```
colorscheme desert
syntax on
:highlight Comment ctermfg=green
set mouse=v
```

Tips & Tricks

Rsync

```
rsync -avh -log-file=$HOME/.rsyncd.log --delete /mnt/nas/ /mnt/mybook/nas-backup/ --dry-run


-a = archive (preserve permissions, symbolic links, etc.)
-v = verbose
-h = human-readable
--delete = delete files in destination that no longer exist in source
--dry-run = simulate the process without making changes
```

Create ssh key

```
ssh-keygen -t ed25519 -C "comment" -f <keys-name>
```

Last commands exit code

```
echo $?
```

Delete only directories

```
rm -rf `ls -d */`
find . -maxdepth 1 -mindepth 1 -type d -exec rm -rf '{}' \;
```

Disable swap

```
cat /proc/sys/vm/swappiness
sysctl vm.swappiness=0


vim /etc/sysctl.conf
vm.swappiness=0
```

Extract multiple part rar file

```
unrar x <filename>.rar
```

sudo !! - will instantly run the previously run command but with sudo prefix

UHD - usb hsm device

Upgrade all outdates pip packages

```
pip list --outdated
pip install -U `pip list --outdated | awk 'NR>2 {print $1}'`
```

Clean /boot & /usr disk, freeing inodes

First try

```
apt autoremove
apt autoremove --purge
purge-old-kernels
```

If unsuccessful, then

```
# Check your kernel
uname -r
cat /proc/version

# List installed kernels, headers, modules
dpkg --list 'linux-*' | grep ^ii
apt list --installed linux-*

# Remove and purge unnecessary ones
apt remove --purge linux-{headers,image,modules,modules-extra}-4.15.0-1*
apt remove --purge linux-{headers,image,modules,modules-extra}-4.15.0-20[0-6]

certain numbers/names - {1,2,3}
certain range - [1-3]

# Check iu packages
dpkg --list 'linux-*' | grep "^iU"

# Remove if necessary
apt remove --purge $(dpkg -l | grep "^iU" | awk '{print $2}')
```

JMXterm example

https://www.googlecloudcommunity.com/gc/Cloud-Product-Articles/How-to-Read-JMX-Statistics-via-The-Shell/ta-p/77629

```
java -jar jmxterm-1.0.2-uber.jar
jvms
```

```
open <PID>
beans
get -b java.lang:type=Memory *
get used -b java.lang:type=Memory HeapMemoryUsage
```

Kick ssh session

```
who -u
echo "HAHA" | write user1 pts/2
kill -9 24995
```

putty pub2 to linux format

`ssh-keygen -i -f <pub2>`

restart network interface

`ip link set ens160 down && ip link set ens160 up`

journalctl vacuum

```
journalctl --disk-usage
journalctl --rotate
journalctl -D /var/log/journal/ --vacuum-size=100M
journalctl --verify
```

Measure command or script execution time

`time <command or script>`

Create user wo password

`adduser <user> --disabled-password --force-badname`

Add user to group

`usermod -a -G <group> <user>`

Remove user from group

`gpasswd -d <user> <group>`

Change shell to Bash

`sudo usermod -s /bin/bash $USER`

Switch iTerm2 to Bash

`chsh -s /bin/zsh`

Get pid info

`cat /proc/pid/cmdline`

Get key from port 80

`apt-key adv --keyserver hkp://keys.gnupg.net:80 --recv "E775D097"`

Sudo without password

create group, for ex named wheel
`sudo visudo` and add the line below to it
`wheel     ALL=(ALL) NOPASSWD:ALL`
run `usermod -a -G wheel $USER`

Fix locale

```
open /etc/environment


LANGUAGE="en_US.UTF-8"
LANG="en_US.UTF-8"
LC_ALL="en_US.UTF-8"


locale-gen en_US.UTF-8
```

See lib-s waiting to be "updated"

`lsof | grep lib | grep DEL`

Attach NAS

Mac

Finder -> Go -> Connect to Server `afp://LS520DE2a5.local/<dir>`

Linux

mkdir /mnt/nas-plex

add to /etc/fstab

```
//{{ nas_host }}/Plex /mnt/nas-plex username={{ nas_username }},password={{ nas_password }},iocharset=utf8,uid=1000,gid=1000
```

mount -a

Win

Computer -> add a network location -> `\\hostname\dir`

My external IP

`curl ifconfig.me`

View CSR

`openssl req -text -noout -verify -in CSR.csr`

View certificate

`openssl x509 -in certificate.crt -text -noout`

Copy ssh key to machine

`ssh-copy-id <hostname or IP>`

Measure command execution time

`time <command>`

List All Directories

`ls -d */`

Move Directory One Lv Down

`mv Math/ ../`

Directory Size

`du -hs <dir_name>`

Check Storage

`df -h`

Show Linux distro info

`lsb_release -a`

Show certificate info

`openssl x509 -in certificate.crt -text -noout`

Delete all mail

`mail`
`d *`
`q`

Show cpu stats / list cores

`lscpu`
`cat /proc/cpuinfo`
`cat /proc/cpuinfo | grep processor | wc -l`

Select default editor for logged in user

`select-editor`

Count process open file handlers

`ls /proc/my_pid/fd | wc -l`

Traceroute

Linux `mtr qrl.ee`
Windows `tracert qrl.ee`

Count open files (The value does not apply to a root user, so if you want to tally with the lsof output, exclude files opened by root)

`lsof|wc -l`

Open files limit by OS

`cat /proc/sys/fs/file-max`

Create trash 10MB base64 filled file

`base64 /dev/urandom | head -c 10000000 > file.txt`

Create trash 1MB text filled file

`yes "HelloWorld" | head --byte=1100000 > 1MB.txt`

Virtualbox shared folder

Go to Devices option of your VM and click on the Shared Folders option and add the required_folder you want to share.
Now in your file system (root) you can see media/sf_required_folder.
But by default the Guest User won't have access to this folder.
So to grant access to this folder you need to add Guest user to the group vboxsf.
For this, `sudo adduser Guest_user vboxsf` to give access.
Now have the access.

Vagrant

Basic commands

```
vagrant up
vagrant provision
vagrant destroy

vagrant ssh
vagrant status
```

Example Vagrantfile

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.box = "ubuntu/bionic64"
  config.ssh.insert_key = false
  config.vm.hostname = "rpi.setup.local"

  config.vm.provider :virtualbox do |v|
    v.memory = 1024
    v.cpus = 1
  end

  config.vm.define "rpi.setup" do |instance|
    instance.vm.hostname = "rpi.setup.local"
    # Set forwarded ports from virtualbox instance
    #instance.vm.network "forwarded_port", guest: 80, host: 80

    # Ansible provisioning.
    instance.vm.provision "ansible" do |ansible|
      ansible.playbook = "site.yml" # playbook reference
      ansible.become = true
      ansible.ask_vault_pass = true # --ask-vault-pass
```

```
      #ansible.vault_password_file = "~/.vault_pass_local.txt" # vault password in a file
      ansible.groups = {
        "all:vars" => { "ansible_python_interpreter" => "/usr/bin/python3" },
        "proxy"    => [ "rpi.setup" ]
      }
      # Enable ansible verbosity
      #ansible.verbose  = "vvvv"
    end
  end
end
```

Windows

Test IP and port connection

```
Test-NetConnection <IP> -port <port>
```

GPG

Show fingerprint

```
gpg --show-keys xroad.pub
```

Show ID

```
gpg --import xroad.pub --dry-run
```

Add gpg key from url

```
apt-key adv --fetch-keys <url>
```