My Ubuntu bash Setup and Ricing

Here is my configuration for the Linux UBUNTU terminal.

🔁 Take it as a **backup** and easy way to transition from one PC or VM to another.

Table of contents

What does ricing mean?

In the context of terminal configuration, ricing refers to the practice of customizing and beautifying the appearance and functionality of the terminal and other elements of the desktop environment.

The term originally comes from the custom car culture, where rice was used to describe modifying cars with flashy but often unnecessary features.

In the tech world, ricing has been adopted to describe the process of making a system look aesthetically pleasing and unique.

Installing commands

> ✖ Do not execute the commands below as a script
>
> 🧑‍💻 Some may expect user input.
>
> ⚠ The installation might end with an error, so take them **individually** and analyze the reusult.

Task	apt	apt-get	snap	dpkg

Task	apt	apt-get	snap	dpkg
Install Package Manager	Pre-installed	Pre-installed	`sudo apt install snapd`	Pre-installed
Update Package Manager	`sudo apt update`	`sudo apt-get update`	`sudo snap refresh`	`sudo apt update`
Update All Packages	`sudo apt upgrade`	`sudo apt-get upgrade`	`sudo snap refresh`	`sudo apt upgrade`
Install a Package	`sudo apt install <package>`	`sudo apt-get install <package>`	`sudo snap install <package>`	`sudo dpkg -i <package>.deb`
List All Packages	`apt list --installed`	`apt-get list --installed`	`snap list`	`dpkg --list`.
Get Version of a Package	`apt show <package>`	`apt-cache policy <package>`	`snap info <package>`	`dpkg -s <package>`
Delete a Package	`sudo apt remove <package>`	`sudo apt-get remove <package>`	`sudo snap remove <package>`	`sudo dpkg -r <package>`

Task	cargo	pip/pip3	npm
Install Package Manager	`sudo apt install cargo` or `curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh`	`sudo apt install python-pip` (for pip), `sudo apt install python3-pip` (for pip3)	`sudo apt install npm`
Update Package Manager	`cargo install-update -a`	Not applicable (managed by Python installer)	`npm install -g npm@latest`
Update All Packages	`cargo install-update -a`	`pip list --outdated` + `pip install --upgrade <package>`	`npm update -g`
Install a Package	`cargo install <package>`	`pip install <package>` (pip for Python 2), `pip3 install <package>` (pip for Python 3)	`npm install <package>`
List All Packages	`cargo install --list`	`pip list` or `pip3 list`	`npm list -g --depth=0`

Task	cargo	pip/pip3	npm
Get Version of a Package	`cargo search <package>` or `cargo --version`	`pip show <package>` or `pip3 show <package>`	`npm list <package> -g`
Delete a Package	`cargo uninstall <package>`	`pip uninstall <package>` or `pip3 uninstall <package>`	`npm uninstall -g <package>`

```
sudo apt update
sudo apt install snapd

sudo apt install google-chrome-stable


sudo apt install xfce4-terminal     # terminal emulator (for background
image)


# Please see oficial Cargo Rust installation:
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
source $HOME/.cargo/env     # place it in ~/.bashrc



sudo apt install ripgrep
cargo install ripgrep       # cargo alternative (will install `rg` command)

sudo apt install locate
sudo apt install fd-find
sudo apt install fzf        # fuzzy finder
sudo apt install sl         # Steam Locomotive
sudo apt install cmatrix



sudo apt install htop       # process viewer
sudo apt install bpytop     # resource monitor that shows usage and stats


# downloading instructions for multiple OS: https://snapcraft.io/zellij
# downloading instructions for Ubuntu:
https://snapcraft.io/install/zellij/ubuntu
sudo snap install zellij --classic                  # a better TMUX
# cargo install zellij                              # alternative

sudo apt install nano                               # text editor
sudo snap install helix --classic                   # modal text editor
# cargo install helix                               # alternative
```

```
sudo snap install code --classic                    # VS Code IDE
sudo snap install intellij-idea-ultimate --classic  # IDE for Java, Scala
sudo snap install rustrover --classic               # IDE for Rust
sudo snap install clion --classic                   # IDE for C/C++

sudo snap install onefetch  # info about GIT REPO
sudo apt-get install neofetch                       # info about OS distro



sudo apt install fd-find                            # better find command


# downloading instructions: https://github.com/cowboy8625/rusty-rain.git


# A HTTP Test Tool: https://hurl.dev/player.html?id=hurl&speed=3
cargo install hurl

cargo install rusty-rain                            # a CMatrix clone in
Rust

sudo apt install exa                                # a modern replacement
for ls
# Advice: place in `~/.bashrc`: alias lsc=exa


cargo install lsd                                   # colorfull ls
cargo install bat                                   # collor appealing cat

cargo install tokei                                 # code statistics


sudo apt install colortest


sudo snap install discord
sudo snapp install spotify



git config --user.name='TrifanBogdan24'
git config --user.email=''      # my email
```

💾 Configuration File (`~/.bashrc`)

In order for the following changes to be persistent over time (restarting the terminal) the modifications are made in a configuration file, saved locally, on the disk.

> 💾 In my case, `~/.bashrc`.

In this file, the PS1 environment variable and some suggestive **aliases** are set.

Each time a bash terminal is opened, all these instructions are executed, and the **aliases** will be accessible in every such terminal session.

```
$ nano -l ~/.bashrc
```

Copy and paste the following code at the end of the configuration file:

```
# at the end of `~/.bashrc`:


# token for committing to my private repos
export GITHUB_TOKEN='my token'      # personal, sensitive info
alias github_token="echo $GITHUB_TOKEN"


alias chrome='google-chrome &> /dev/null &'
alias youtube='open https://www.youtube.com/ &> /dev/null'          # opens
YouTube in web browser
alias chatgpt='open https://chatgpt.com/ &> /dev/null'              # opens
ChatGpt in web browser

alias periodic-table='npx periodic-table-cli'
alias world-map='telnet mapscii.me'
alias recent-files='ls -ltrh'
alias cmd-help='compgen -c | fzf | xargs man'
alias ascii_colors='colortest-16b'
alias hacking-terminal='docker run --rm -it bcbcarl/hollywood'       #
`CTRL-C` and `exit` to stop


                                        # escapes anotether alias
alias git_reset_last_commit="git reset --hard \$(git log | grep 'commit'
awk 'NR==1 {print $2}')"

alias git_delete_last_commit="git reset --soft HEAD~1 && git push -f
origin"


git_rename_last_commit() {
    # `$1` = arg 1 = new commit message
    git commit --ammend -m $1
    git push -f origin
}




# opens IDEs in current directory
```

```
alias vscode='code .'
alias open-rustrover='rustrover . &> /dev/null &'
alias open-intellij='intellij-idea-ultimate . &> /dev/null &'
alias open-clion='clion . &> /dev/null &'

alias ip='ip -c'        # colored command




# colored manual page
man() {
    LESS_TERMCAP_mb=$'\e[1;34m'   \
    LESS_TERMCAP_md=$'\e[1;32m'   \
    LESS_TERMCAP_so=$'\e[1;33m'   \
    LESS_TERMCAP_us=$'\e[1;4;31m' \
    LESS_TERMCAP_me=$'\e[0m'      \
    LESS_TERMCAP_se=$'\e[0m'      \
    LESS_TERMCAP_ue=$'\e[0m'      \
    command man "$@"
}



# for PS1 prompt variable
# get current branch in git repo
function parse_git_branch() {
    BRANCH=$(git branch 2> /dev/null | sed -e '/^[^*]/d' -e 's/* \
(.*\)/\1/')
    if [ ! "${BRANCH}" == "" ]; then
        STAT=$(parse_git_dirty)
        if [ "${BRANCH}" == "master" ]; then
            echo -e "[\e[32mgit: master${STAT}\e[0m]"
        elif [ "${BRANCH}" == "main" ]; then
            echo -e "[\e[32mgit: main${STAT}\e[0m]"
        else
            echo "[git: ${BRANCH}${STAT}]"
        fi
    else
        echo ""
    fi
}



# for PS1 prompt variable
# get current status of git repo
function parse_git_dirty {
    status=`git status 2>&1 | tee`
    dirty=`echo -n "${status}" 2> /dev/null | grep "modified:" &>
/dev/null; echo "$?"`
    untracked=`echo -n "${status}" 2> /dev/null | grep "Untracked files" &>
/dev/null; echo "$?"`
    ahead=`echo -n "${status}" 2> /dev/null | grep "Your branch is ahead
```

```
of" &> /dev/null; echo "$?"`
    newfile=`echo -n "${status}" 2> /dev/null | grep "new file:" &>
/dev/null; echo "$?"`
    renamed=`echo -n "${status}" 2> /dev/null | grep "renamed:" &>
/dev/null; echo "$?"`
    deleted=`echo -n "${status}" 2> /dev/null | grep "deleted:" &>
/dev/null; echo "$?"`
    bits=''
    if [ "${renamed}" == "0" ]; then
        bits=">${bits}"
    fi
    if [ "${ahead}" == "0" ]; then
        bits="*${bits}"
    fi
    if [ "${newfile}" == "0" ]; then
        bits="+${bits}"
    fi
    if [ "${untracked}" == "0" ]; then
        bits="?${bits}"
    fi
    if [ "${deleted}" == "0" ]; then
        bits="x${bits}"
    fi
    if [ "${dirty}" == "0" ]; then
        bits="!${bits}"
    fi
    if [ ! "${bits}" == "" ]; then
        echo " ${bits}"
    else
        echo ""
    fi
}




# daca nu iti place cum arata terminalul, comenteaza linia de mai jos
# PS1='\[\e[1;39m\](\[\e[0;0m\] \[\e[1;34m\]\u\[\e[0;0m\]\[\e[1;39m\]@\
[\e[0m\]\[\e[1;34m\]\h\[\e[0m\]\[\e[1;39m\] )\[\e[0m\] \[\e[1;39m\]: [ \
[\e[1;96m\]\w\[\e[0m\] \[\e[1;39m\]]\[\e[0m\] \[\e[1;39m\]$\[\e[0m\]\n\
[\e[1;96m\]$\[\e[0m\] '


# without GIT REPO
# PS1='\[\e[1;39m\](\[\e[0;0m\] \[\e[1;34m\]\u\[\e[0;0m\]\[\e[1;39m\]@\
[\e[0m\]\[\e[1;34m\]\h\[\e[0m\]\[\e[1;39m\] )\[\e[0m\] \[\e[1;39m\]: [ \
[\e[1;96m\]\w\[\e[0m\] \[\e[1;39m\]]\[\e[0m\] \[\e[1;39m\]\[\e[0m\] \
[\e[1;39m\]$\[\e[0m\] $(printf "%.0s." {1..10})\[\e[0m\] $(date "+%T") \n\
[\e[1;96m\]$\[\e[0m\] '

# info about GIT REPO
PS1='\[\e[1;39m\](\[\e[0;0m\] \[\e[1;34m\]\u\[\e[0;0m\]\[\e[1;39m\]@\
```

```
[\e[0m\]\[\e[1;34m\]h\[\e[0m\]\[\e[1;39m\] )\[\e[0m\] \[\e[1;39m\]: [ \
[\e[1;96m\]\w\[\e[0m\] \[\e[1;39m\]]\[\e[0m\] \
[\e[1;39m\]$(parse_git_branch)\[\e[0m\] \[\e[1;39m\]$\[\e[0m\] $(printf
"%.0s." {1..10})\[\e[0m\] $(date "+%T") \n\[\e[1;96m\]$\[\e[0m\] '



# displaying the logo of Arch Linux
# neofetch --ascii_distro arch


# the below command will display a logo of Ubuntu Linux
# neofetch --ascii_distro ubuntu --ascii_colors 4 7 --colors 6 7 7 6 7 7


# Kali Linux
# neofetch --ascii_distro kali --colors 6 7 7 6 7


neofetch --ascii_distro ubuntu --ascii_colors 4 7 --colors 6 7 7 6 7 7
```

```
$ source ~/.bashrc
$ reset
```

⚙ Basic Terminal Customizations

⚙: Teminal -> Three Horizontal Bars -> Preferences -> Unnamed -> Colors -> Background -> `#0D0324`

⚙: Terminal -> Three Horizontal Bars -> Preferences -> Unnamed -> Text -> Cursor shape -> `I-Beam`

⚙: Terminal -> Three Horizontal Bars -> Preferences -> Unnamed -> Text -> Cursor blinking -> `Enable`

Nerd Fonts

> 🌐 Fonts: https://www.nerdfonts.com/font-downloads
>
> 🌐 Also see: https://www.nerdfonts.com/

```
$ cd ~/Downloads/
$ mkdir nerd-fonts-helper-dir

$ touch nerd_font_downloader.sh
$ chmod +x nerd_font_downloader.sh
$ nano -l nerd_font_downloader.sh       # text editor
```

```
#!/bin/bash

# download_nerd_font.sh

nr_args=$#

if [[ $nr_args -ne 1 ]]; then
    echo "Err: the script expects only a single argument: the URL of the
font"
    echo "See fonts at: https://www.nerdfonts.com/font-downloads"
    exit 1
fi



URL=$1

wget $URL -O nerd-fonts.zip
unzip nerd-fonts.zip -d nerd-fonts
mkdir -p ~/.local/share/fonts
mv nerd-fonts/* ~/.local/share/fonts/
rm -rf nerd-fonts nerd-fonts.zip
fc-cache -fv
```

```
$ # it might not work (be aware of the versions of the fonts)
$ ./nerd_font_downloader.sh https://github.com/ryanoasis/nerd-
fonts/releases/download/v3.2.1/InconsolataGo.zip
```

> 📥 Script to install a single nerd font, being given an URL: scripts/download_nerd_font.sh
>
> 📥 Script to install all single nerd fonts: scripts/install_all_nerd_fonts.sh

My favourite Nerd Fonts:

- 0xProto Nerd Font
- 3270 Nerd Font
- CommitMono Nerd Font
- JetBrainsMono Nerd Font
- M+ Nerd Font
- RobotoMono Nerd Font
- Ubuntu Nerd Font
- UbuntuMono Nerd Font
- UbuntuSans Nerd Font
- VictorMono Nerd Font

Setting Nerd Fonts

---

1. In `Ubuntu` terminal: ⚙: Terminal -> Three Horizontal Bars -> Preferences -> Text -> Check the box for `Custom font` and search for a **Nerd font**
2. In `VS Code` terminal: ⚙: Open `VS Code` -> `CTRL ,` -> Search for `terminal integrated font` -> type a **Nerd font** (some might not work well)

Configurable Terminal Prompt (`oh my posh`)

🌐 `oh my posh` official website: https://ohmyposh.dev/

```
$ curl -s https://ohmyposh.dev/install.sh | sudo bash -s

$  mkdir -p ~/.poshthemes
```

🌐 `oh my posh` themes: https://ohmyposh.dev/docs/themes

> All these themes work only in `Nerd Fonts`
>
> You have to set **terminal's font** for each `IDE` you use.
>
> Some fonts migth not work well in `IDE`s.

> 📥 Installing all `oh my posh` themes: scripts/install_all_prompt_themes.sh

My favourite themes:

> Setting `alias` for favourite color themes: cripts/alias_fav_posh_themes.sh
>
> For persistance, place the code in `~/.bashrc` and use the following command: `source ~/.bashrc`

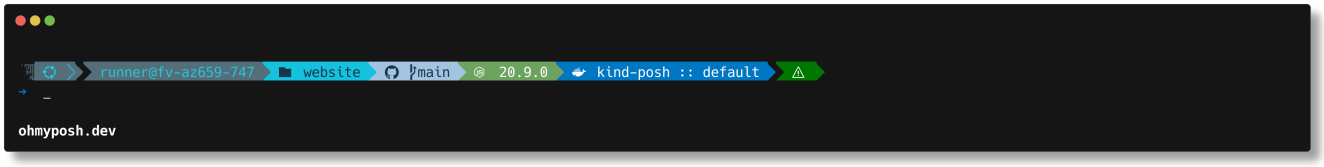
- atomic


```
# Installing the theme
wget https://raw.githubusercontent.com/JanDeDobbeleer/oh-my-
posh/main/themes/atomic.omp.json -O ~/.poshthemes/atomic.omp.json

# Setting the prompt
eval "$(oh-my-posh init bash --config ~/.poshthemes/atomic.omp.json)"
```

- blue-owl

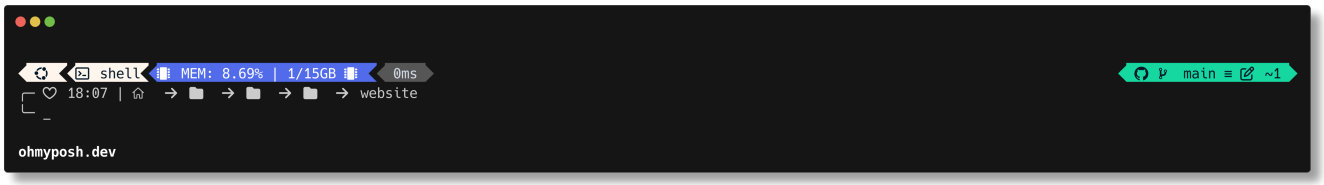

```
# Installing the theme
wget https://raw.githubusercontent.com/JanDeDobbeleer/oh-my-
posh/main/themes/blue-owl.omp.json -O ~/.poshthemes/blue-owl.omp.json

# Setting the prompt
eval "$(oh-my-posh init bash --config ~/.poshthemes/blue-owl.omp.json)"
```

- blueish




```
# Installing the theme
wget https://raw.githubusercontent.com/JanDeDobbeleer/oh-my-
posh/main/themes/blueish.omp.json -O ~/.poshthemes/blueish.omp.json

# Setting the prompt
eval "$(oh-my-posh init bash --config ~/.poshthemes/blueish.omp.json)"
```

- clean-detailed




```
# Installing the theme
wget https://raw.githubusercontent.com/JanDeDobbeleer/oh-my-
posh/main/themes/clean-detailed.omp.json -O ~/.poshthemes/clean-
detailed.omp.json

# Setting the prompt
eval "$(oh-my-posh init bash --config ~/.poshthemes/clean-
detailed.omp.json)"
```

- kali


```
# Installing the theme
wget https://raw.githubusercontent.com/JanDeDobbeleer/oh-my-
posh/main/themes/kali.omp.json -O ~/.poshthemes/kali.omp.json

# Setting the prompt
eval "$(oh-my-posh init bash --config ~/.poshthemes/kali.omp.json)"
```

- powerlevel10k_modern


```
# Installing the theme
wget https://raw.githubusercontent.com/JanDeDobbeleer/oh-my-
posh/main/themes/powerlevel10k_modern.omp.json -O
~/.poshthemes/powerlevel10k_modern.omp.json

# Setting the prompt
eval "$(oh-my-posh init bash --config
~/.poshthemes/powerlevel10k_modern.omp.json)"
```

- powerlevel10k_rainbow


```
# Installing the theme
wget https://raw.githubusercontent.com/JanDeDobbeleer/oh-my-
posh/main/themes/powerlevel10k_rainbow.omp.json -O
~/.poshthemes/powerlevel10k_rainbow.omp.json

eval "$(oh-my-posh init bash --config
~/.poshthemes/powerlevel10k_rainbow.omp.json)"
```

- quick-term


```
# Installing the theme
wget https://raw.githubusercontent.com/JanDeDobbeleer/oh-my-
posh/main/themes/quick-term.omp.json -O ~/.poshthemes/quick-term.omp.json


# Setting the prompt
eval "$(oh-my-posh init bash --config ~/.poshthemes/quick-term.omp.json)"
```

> I find `quick-term` to be the most suitable for me.
>
> So, the line `eval "$(oh-my-posh init bash --config ~/.poshthemes/quick-term.omp.json)"` will be inlcuded at the end of the configuration file `~/.bashrc`

 Xfce Terminal Emulator

---

Installilling a Terminal Emulator (Xfce)

For this task, we will use a `Terminal Emulator`, since the built-in console does not support background images.

Show applications (a square of 9 dots in right lower corner) -> search for `Ubuntu Software` -> start typing `Xfce Terminal` -> install **Terminal Emulator**.

> Alternative `shell` command:

```
# Terminal Emulator that enables setting a background image
sudo apt install xfce4-terminal
```

`Xfce` command for creating a new **window**/**tab**:

- New **window**: `xfce4-terminal`
- New **tab**: `xfce4-terminal --tab` (in an existing **window**, otherwise it creates a **window** with two **tabs**)

```
xfce4-terminal &        # new window
xfce4-terminal --tab &  # new tab inside of window
```

⚙ Basic setup for `Xfce Terminal`

⚙ `Xfce Terminal` -> `Edit` -> `Preferences...` -> Cursor shape -> `I-Beam`

⚙ `Xfce Terminal` -> `Edit` -> `Preferences...` -> Cursor shape -> check ☑ `Cursor blinks` box

⚙ `Xfce Terminal` -> `Edit` -> `Preferences...` -> Cursor shape -> check ☑ `Automatically copy selection to clipboard` box

⚙ `Xfce Terminal` -> `Edit` -> `Preferences...` -> Cursor shape -> check ☐ `Show unsafe paste dialog` box

🖼 Setting the background image in `Xfce Terminal`

Open `Xfce Terminal` -> Bar -> `Edit` -> `Preferences...` -> Appearance -> Background -> select `Background image` and provide a path to the `File:` field.

> Also, in order for the prompt to work, set a `Nerd Font`

🖧 DNS (Domain Name Server)

🌐What is `DNS`

> `DNS` is an Internet protocol that maps `URL`s (Uniform Resource Locators) or domain names to `IP` addresses. This process in essential because while humans find it easier to remember and use domain names, like www.example.com, computers and network devices use `IP` addresses (like `192.0.2.1` to identify each other on the internet).

Useful links:

- Known `DNS` providers: https://adguard-dns.io/kb/general/dns-providers/
- `DNS` filtering: https://adguard-dns.io/kb/general/dns-filtering/

> I use the following `DNS IP`s: `94.140.14.15` and `94.140.15.16`

⚙ Setting `DNS`

> Using the `GUI` of `Ubuntu`: https://phoenixnap.com/kb/ubuntu-dns-nameservers

I don't recommend the above link, but it's just in case.

Let's be professional and use the terminal 🧑‍💻.

```
$ cat /etc/resolv.conf
# This is /run/systemd/resolve/stub-resolv.conf managed by man:systemd-
resolved(8).
# Do not edit.
#
# This file might be symlinked as /etc/resolv.conf. If you're looking at
# /etc/resolv.conf and seeing this text, you have followed the symlink.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "resolvectl status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs should typically not access this file directly, but
only
# through the symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in
a
# different way, replace this symlink by a static file or a different
symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes
of
# operation for /etc/resolv.conf.

nameserver 127.0.0.53
options edns0 trust-ad
search .
```

> 👉 Notice that `/etc/resolv.conf` is a **symbolic link** to another configuration file, **in this case**, `/run/systemd/resolve/stub-resolv.conf`
>
> 📢 Also notice the comment `# Do not edit`

Since `/etc/resolv.conf` is managed by `systemd-resolved` and is **symlinked** to `/run/systemd/resolve/stub-resolv.conf`, the `DNS` configuration must be updated using the `systemd-resolved` **configuration file**.

```
$ sudo nano -l /etc/systemd/resolved.conf
```

Uncomment and **set** the `DNS` and `FallbackDNS` lines with desired DNS servers. For example:

```
[Resolve]
DNS=94.140.14.15 94.140.15.16
FallbackDNS=8.8.8.8 8.8.4.4 1.1.1.1 1.0.0.1
```

```
$ sudo systemctl restart systemd-resolved
$ resolvectl status
Global
           Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
    resolv.conf mode: stub
         DNS Servers: 94.140.14.15 94.140.15.16
Fallback DNS Servers: 8.8.8.8 8.8.4.4 1.1.1.1 1.0.0.1
```

> An alternative to `resolvectl status` could be `resolvectl dns`:

```
$ resolvectl dns
Global: 94.140.14.15 94.140.15.16
Link 2 (enp1s0): 94.140.14.15 94.140.15.16
Link 3 (wlp2s0): 8.8.8.8 1.1.1.1
Link 4 (docker0):
```

> The `FallbackDNS` entries are optional and will be used if the primary DNS servers are unreachable.

!⚠ Troubleshooting `DNS` configuration

> If there are errors at setting **global** `DNS` configure the **interfaces**, otherwise skip this part.

If it appears that the `DNS` setting for specific network interfaces (`enp1s0` and `wlp2s0`) might not be using the global `DNS` server, here is the solution:

1. Step 1: Clear Link-Specific `DNS` Settings

```
$ sudo resolvectl dns enp1s0 94.140.14.15 94.140.15.16
$ sudo resolvectl dns wlp2s0 94.140.14.15 94.140.15.16
```

2. Step 2: Verify the `DNS` Configuration

```
$ resolvectl status
```

3. Step 3: Update Network Manager Configuration (if applicable)

If you're using NetworkManager, it might override systemd-resolved settings. You can update the DNS configuration in NetworkManager.

```
$ sudo nano /etc/NetworkManager/NetworkManager.conf
```

Add the following lines (if not already present):

```
[main]
dns=systemd-resolved
```

Restart NetworkManager:

```
$ sudo systemctl restart NetworkManager
```

4. Step 4: Ensure /etc/resolv.conf is Symlinked Correctly

```
sudo ln -sf /run/systemd/resolve/stub-resolv.conf /etc/resolv.conf
```

5. Step 5: Recheck the Status

```
$ resolvectl status
```

🗑 Uninstalling Utilities

Deleting a command is as simple as installing it, the only difference in the one-liner is a word specified to the package manager.

> 🗑 `remove` will take place of `install` argument.

```
rustup self uninstall           # uninstall cargo

sudo apt remove xfce4-terminal

sudo apt remove cmatrix
sudo snap remove code
sudo snap remove intellij-idea-ultimate
sudo snap remove rustrover
sudo snap remove clion

sudo apt remove nano
sudo snap remove helix
# cargo uninstall helix         # if installed with cargo

sudo snap remove zellij
# cargo uninstall zellij        # if installed with cargo


cargo uninstall ripgrep         # rg command

sudo apt remove fd-find
```

```
sudo snap remove onefetch
sudo apt-get remove neofetch


sudo snap remove discord
sudo snap remove spotify

sudo snap remove exa

cargo remove hurl
cargo uninstall bat
cargo uninstall lsd
cargo uninstall rusty-rain
cargo uninstall tokei

# unsetting git info
git config --global --unset user.name
git config --global --unset user.email
```