Fedora Workstation: Things to do after installation (Apps, Settings, and Tweaks)

Please feel free to raise any comments or issues on the website's Github repository. Pull requests are very much appreciated.

In the following I will go through my post installation steps, i.e. which settings I choose and which apps I install and use.

Basic Steps

Go through welcome screen

This is self-explanatory. Usually I already set up Online Accounts for Nextcloud.

Get Thunderbolt Dock to work and adjust monitors

I use a Thunderbolt Dock (DELL TB16) with three monitors, which is great but also a bit tricky to set up. The most important step is to check in "Settings-Privacy-Thunderbolt", whether "Dell Thunderbord Cable" and "Dell Thunderbord Dock" are both set to "Authorized". You may need to go into your BIOS to make some changes as outlined on the Dell TB16 Archwiki. I noticed that sometimes I just need to plug the USB-C cable in and out a couple of times or turn it around to make it work. A reboot might help as well. Once it works, it usually keeps working, so this is annoying just for a short time :-) Once it works, I can arrange my three monitors in "Settings-Display".

Wayland or Xorg

By Default Wayland is enabled. If you have a Nvidia card this is not working well, so you would have to disable it. Also MATLAB seems to work better on Xorg than Wayland, but I still need to test this. Anyways, on my Dell XPS 13 I usually stick to Wayland, whereas on my Dell Precision 7520 I disable it by uncommenting `WaylandEnable=false` and adding `DefaultSession=gnome-xorg.desktop` to the [daemon] section of /etc/gdm/custom.conf:

```
sudo nano /etc/gdm/custom.conf
# [daemon]
# WaylandEnable=false
# DefaultSession=gnome-xorg.desktop
```

Next time you reboot the system it will boot into an Xorg Gnome session.

Nvidia

If you have a Nvidia card, run Xorg and the following:

```
modinfo -F version nvidia
sudo dnf update -y # and reboot if you are not on the latest kernel
sudo dnf install -y akmod-nvidia # rhel/centos users can use kmod-nvidia instead
sudo dnf install -y xorg-x11-drv-nvidia-cuda #optional for cuda/nvdec/nvenc support
sudo dnf install -y xorg-x11-drv-nvidia-cuda-libs
sudo dnf install -y vdpauinfo libva-vdpau-driver libva-utils
sudo dnf install -y vulkan
modinfo -F version nvidia
```

DNF flags

I add some flags to the dnf conf file to speed it up:

```
echo 'fastestmirror=1' | sudo tee -a /etc/dnf/dnf.conf
echo 'max_parallel_downloads=10' | sudo tee -a /etc/dnf/dnf.conf
echo 'deltarpm=true' | sudo tee -a /etc/dnf/dnf.conf
cat /etc/dnf/dnf.conf
# [main]
# gpgcheck=1
# installonly_limit=3
```

```
# installonly_limit=3
# clean_requirements_on_remove=True
# best=False
# skip_if_unavailable=True
# fastestmirror=1
# max_parallel_downloads=10
# deltarpm=true
```

Set hostname

By default my machine is called localhost; hence, I rename it for better accessability on the network:

```
hostnamectl set-hostname fedora
```

Check locales

Let's check if the locales and timezone is correctly set:

```
localectl status
# System Locale: LANG=de_DE.UTF-8
# VC Keymap: de-nodeadkeys
# X11 Layout: de
# X11 Variant: nodeadkeys
timedatectl
# Local time: Di 2020-11-03 13:58:54 CET
# Universal time: Di 2020-11-03 12:58:54 UTC
# RTC time: Di 2020-11-03 12:58:54
# Time zone: Europe/Berlin (CET, +0100)
# System clock synchronized: yes
# NTP service: active
# RTC in local TZ: no
```

Looks good, if not see the help file on the two commands or change locales and timezone in Gnome-Settings.

btrfs filesystem optimizations

Fedora has not optimized the mount options for btrfs yet. I have found that there is *some* general agreement on the following mount options if you are on a SSD or NVME:

- `ssd`: use SSD specific options for optimal use on SSD and NVME
- `noatime`: prevent frequent disk writes by instructing the Linux kernel not to store the last access time of files and folders
- `space_cache`: allows btrfs to store free space cache on the disk to make caching of a block group much quicker
- `commit=120`: time interval in which data is written to the filesystem (value of 120 is taken from Manjaro's minimal iso)
- `compress=zstd`: allows to specify the compression algorithm which we want to use. btrfs provides lzo, zstd and zlib compression algorithms. Based on some Phoronix test cases, zstd seems to be the better performing candidate.
- `discard=async`: Btrfs Async Discard Support Looks To Be Ready For Linux 5.6

So add these options to your btrfs subvolume mount points in your fstab:

```
sudo nano /etc/fstab
# UUID=47faf958-b80a-43e1-a36f-ca5a932474f7 /                       btrfs   subvol=root,x-systemd.device-timeou
# UUID=04ae92cd-717c-4aaf-bb24-58001be8d334 /boot                   ext4    defaults        1 2
# UUID=C17B-722D                          /boot/efi               vfat    umask=0077,shortname=winnt 0 2
# UUID=47faf958-b80a-43e1-a36f-ca5a932474f7 /home                   btrfs   subvol=home,x-systemd.device-timeou
# UUID=47faf958-b80a-43e1-a36f-ca5a932474f7 /btrfs_pool             btrfs   subvolid=5,x-systemd.device-timeout
sudo mkdir -p /btrfs_pool
sudo mount -a
```

Note that I also add a mountpoint for the btrfs root filesystem (this has always id 5) for easy access of all my subvolumes in `/btrfs_pool`. You would need to restart to make use of the new options. I usually first run updates and restart prior to restoring my backups, such that my restored files are using the optimized mount options such as compression.

Furthermore, as I am using btrfs discard support, let's check whether the `discard` option is passed on in

`/etc/crypttab` (as I am using LUKS to encrypt my drives):

```
sudo nano /etc/crypttab
# luks-fcc669e7-32d5-43b2-ba03-2db6a7f5b33d UUID=fcc669e7-32d5-43b2-ba03-2db6a7f5b33d none discard
```

As both fstrim and discard=async mount option can peacefully co-exist, I also enable `fstrim.timer`:

```
sudo systemctl enable fstrim.timer
```

Install updates and reboot

```
sudo dnf upgrade --refresh
sudo dnf check
sudo dnf autoremove
sudo fwupdmgr get-devices
sudo fwupdmgr refresh --force
sudo fwupdmgr get-updates
sudo fwupdmgr update
sudo reboot now
```

Fish - A Friendly Interactive Shell

I am trying out the Fish shell, due to its user-friendly features, so I install it and make it my default shell:

```
sudo dnf install -y fish util-linux-user
chsh -s /usr/bin/fish
```

You will need to log out and back in for this change to take effect. Lastly, I want to add the ~/.local/bin to my $PATH persistently in Fish:

```
mkdir -p /home/$USER/.local/bin
set -Ua fish_user_paths /home/$USER/.local/bin
```

Also I make sure that it is in my $PATH also on bash:

```
bash -c 'echo $PATH'
#/home/$USER/.local/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin
```

If it isn't then I make the necessary changes in my `.bashrc`, see below.

Gnome Extensions and Tweaks

Let's install the extensions app, Gnome Tweaks, and some extensions:

```
sudo dnf install -y gnome-extensions-app gnome-tweaks
sudo dnf install -y gnome-shell-extension-appindicator
```

I install Sound Input & Output Device Chooser using Firefox. Lastly, I also like Pop Shell, see below how to install it.

In Gnome Tweaks I make the following changes:

- Disable "Suspend when laptop lid is closed" in General
- Disable "Activities Overview Hot Corner" in Top Bar
- Enable "Weekday" and "Date" in "Top Bar"
- Enable Battery Percentage (also possible in Gnome Settings - Power)
- Check Autostart programs

Additional repositories

I enable third party repositories by going into Software -> Software Repositories -> Third Party Repositories -> Enable All. I go through the list and enable all the repositories I think I need such as RPM Fusion NVIDIA Driver. Then I run

```
sudo dnf install -y  https://download1.rpmfusion.org/free/fedora/rpmfusion-free-release-$(rpm -E %fedora).noarc
sudo dnf install -y https://download1.rpmfusion.org/nonfree/fedora/rpmfusion-nonfree-release-$(rpm -E %fedora).
```

to enable the RPM Fusion free and nonfree repositories. Afterwards I run

```
sudo dnf upgrade --refresh
sudo dnf groupupdate core
sudo dnf install -y rpmfusion-free-release-tainted
sudo dnf install -y dnf-plugins-core
```

Checkout `sudo dnf grouplist -v` to see available groups you might be interested in.

Flatpak support

Flatpak is installed by default on Fedora Workstation, but one needs to enable the Flathub store:

```
flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
flatpak update
```

Snap support

Enabling snap support boils down to running the following commands:

```
sudo dnf install -y snapd
sudo ln -s /var/lib/snapd/snap /snap # for classic snap support
sudo reboot now
```

The restart is needed to ensure snap's paths are updated correctly. After the reboot, check whether there are any updates:

```
sudo snap refresh
```

Restore from Backup

I mount my LUKS encrypted backup storage drive using nautilus (simply click on it in the file manager). Then let's use rsync to copy over my files and important configuration scripts:

```
export BACKUP=/run/media/$USER/NAME_OR_UUID_BACKUP_DRIVE/@home/$USER/
sudo rsync -avuP $BACKUP/Desktop ~/
sudo rsync -avuP $BACKUP/Documents ~/
sudo rsync -avuP $BACKUP/Downloads ~/
sudo rsync -avuP $BACKUP/Music ~/
sudo rsync -avuP $BACKUP/Pictures ~/
sudo rsync -avuP $BACKUP/Templates ~/
sudo rsync -avuP $BACKUP/Videos ~/
sudo rsync -avuP $BACKUP/.ssh ~/
sudo rsync -avuP $BACKUP/.gnupg ~/

sudo rsync -avuP $BACKUP/.local/share/applications ~/.local/share/
sudo rsync -avuP $BACKUP/.gitconfig ~/
sudo rsync -avuP $BACKUP/.gitkraken ~/
sudo rsync -avuP $BACKUP/.config/Nextcloud ~/.config/

sudo rsync -avuP $BACKUP/dynare ~/
sudo rsync -avuP $BACKUP/.dynare ~/
sudo rsync -avuP $BACKUP/Images ~/
sudo rsync -avuP $BACKUP/SofortUpload ~/
sudo rsync -avuP $BACKUP/Work ~/
sudo rsync -avuP $BACKUP/Zotero ~/
sudo rsync -avuP $BACKUP/MATLAB ~/
sudo rsync -avuP $BACKUP/.matlab ~/

sudo chown -R $USER:$USER /home/$USER # make sure I own everything
```

SSH keys

If I want to create a new SSH key, I run e.g.:

```
ssh-keygen -t ed25519 -C "fedora-on-precision"
```

Usually, however, I restore my `.ssh` folder from my backup (see above). Either way, afterwards, one needs to add the file containing your key, usually `id_rsa` or `id_ed25519`, to the ssh-agent:

```
eval "$(ssh-agent -s)" #works in bash
```

```
eval (ssh-agent -c) #works in fish
ssh-add ~/.ssh/id_ed25519
```

Don't forget to add your public key to GitHub, Gitlab, Servers, etc.

Pop theme

I love the experience and theming of Gnome in Pop!_OS, so I make Fedora look and behave similarly:

Install Pop-Shell Tiling Extension

```
sudo dnf install -y gnome-shell-extension-pop-shell
```

Logout and Login, then activate it in the Extensions App (I usually don't activate Native Window Placement) and you get an icon in your system tray. Turn on Tiling by clicking on the icon. Note that this will overwrite several Keyboard shortcuts, which is for me a good thing as I am quite used to the shortcuts in Pop!_OS. If you want to be able to view these shortcuts in the icon in the tray, run the following:

```
# Pop shell keyboard shortcuts
sudo dnf install -y make cargo rust gtk3-devel
git clone https://github.com/pop-os/shell-shortcuts /home/$USER/fedora/pop-theme/shell-shortcuts
cd /home/$USER/fedora/pop-theme/shell-shortcuts
make
sudo make install
pop-shell-shortcuts
```

Pop GTK theme

The following installs the Pop!_OS GTK theme:

```
sudo dnf install -y sassc meson glib2-devel
git clone https://github.com/pop-os/gtk-theme /home/$USER/fedora/pop-theme/gtk-theme
cd /home/$USER/fedora/pop-theme/gtk-theme
meson build && cd build
ninja
sudo ninja install

gsettings set org.gnome.desktop.interface gtk-theme "Pop"
```

Pop icon theme

The following installs the Pop!_OS icon theme:

```
git clone https://github.com/pop-os/icon-theme /home/$USER/fedora/pop-theme/icon-theme
cd /home/$USER/fedora/pop-theme/icon-theme
meson build
sudo ninja -C "build" install

gsettings set org.gnome.desktop.interface icon-theme "Pop"
gsettings set org.gnome.desktop.interface cursor-theme "Pop"
```

Pop fonts

For fonts, install

```
sudo dnf install -y fira-code-fonts 'mozilla-fira*' 'google-roboto*'
```

and then go into Gnome Tweaks and make the following changes in Fonts:

- Interface Text: Fira Sans Book 10
- Document Text: Roboto Slab Regular 11
- Monospace Text: Fira Mono Regular 11
- Legacy Window Titles: Fira Sans SemiBold 10
- Hinting: Slight
- Antialiasing: Standard (greyscale)
- Scaling Factor: 1.00

Pop Gnome Terminal Theme

Open gnome-terminal, go to Preferences and choose the Theme Import to Default in the Global tab

Open gnome-terminal, go to Preferences and change the `Theme variant` to `Default` in the Global tab.
Then create a new Profile called `Pop` with the following settings:

- Text
  - Custom font: Fira Mono 12
  - Deactivate Terminal bell
- Colors
  - Deactivate Use colors from system theme
  - Built-in schemes: Custom
  - Default color: Text #F2F2F2 | Background: #333333
  - Bold color (unchecked) #73C48F
  - Cursor color (checked): Text #49B9C7 | Background: #F6F6F6
  - Highlight color (checked): Text #FFFFFF | Background: #48B9C7
  - Uncheck Transparend background
  - Palette colors:
    - 0: #333333 1: #CC0000 2: #4E9A06 3: #C4A000 4: #3465A4 5: #75507B 6: #06989A 7: #D3D7CF
    - 8: #88807C 9: #F15D22 10: #73C48F 11: #FFCE51 12: #48B9C7 13: #AD7FA8 14: #34E2E2 15: #EEEEEC
  - Uncheck Show bold text in bright colors

Right click on the Pop profile and set as default.

Lastly, we need to append some things to `PS1` in our `.bashrc` to get the green prompt and some other neat colors in the terminal. Mine looks like this:

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific environment
if ! [[ "$PATH" =~ "$HOME/.local/bin:$HOME/bin:" ]]
then
    PATH="$HOME/.local/bin:$HOME/bin:$PATH"
fi
export PATH

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions

# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
    xterm-color|*-256color) color_prompt=yes;;
esac

# uncomment for a colored prompt, if the terminal has the capability; turned
# off by default to not distract the user: the focus in a terminal window
# should be on the output of commands, not on the prompt
#force_color_prompt=yes

if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
        # We have color support; assume it's compliant with Ecma-48
        # (ISO/IEC-6429). (Lack of such support is extremely rare, and such
        # a case would tend to support setf rather than setaf.)
        color_prompt=yes
    else
        color_prompt=
    fi
fi

if [ "$color_prompt" = yes ]; then
    PS1='\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
else
    PS1='\u@\h:\w\$ '
fi
```

```
fi
unset color_prompt force_color_prompt

# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
    PS1="\[\e]0;\u@\h: \w\a\]$PS1"
    ;;
*)
    ;;
esac

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi
```

Security steps with Yubikey

I have two Yubikeys and use them

- as second-factor for all admin/sudo tasks using authselect
- to unlock my luks encrypted partitions using mkinitcpio-ykfde
- for my private GPG key using the smart card capabilities of the yubikey

For this I need to install several packages:

```
sudo dnf install -y yubikey-manager # some common packages
# Insert the yubikey
ykman info # your key should be recognized
# Device type: YubiKey 5 NFC
# Serial number:
# Firmware version: 5.1.2
# Form factor: Keychain (USB-A)
# Enabled USB interfaces: OTP+FIDO+CCID
# NFC interface is enabled.
#
# Applications  USB     NFC
# OTP           Enabled Enabled
# FIDO U2F      Enabled Enabled
# OpenPGP       Enabled Enabled
# PIV           Enabled Enabled
# OATH          Enabled Enabled
# FIDO2         Enabled Enabled
```

Make sure that OpenPGP and PIV are enabled on both Yubikeys as shown above.

Yubikey: two-factor authentication for admin/sudo password

Let's set up the Yubikeys as second-factor for everything related to sudo using the pam.d module and authselect:

```
sudo dnf install -y pam-u2f pamu2fcfg
sudo authselect select sssd with-pam-u2f-2fa without-nullok
mkdir ~/.config/Yubico
pamu2fcfg > ~/.config/Yubico/u2f_keys # When your device begins flashing, touch the metal contact to confirm th
pamu2fcfg -n >> ~/.config/Yubico/u2f_keys # Do the same with your backup device
```

Important: before you close the terminal, open a new one and check whether you can do `sudo echo test`:

```
sudo echo test
# Please touch the device.
# [sudo] Passwort for $USER
```

Yubikey: two-factor authentication for luks

Let's set up the Yubikeys as second-factor to unlock the luks partition. If you have brand new keys, then

create a new key on them (BE CAREFUL TO NOT OVERWRITE, I.E. IF YOU HAVE ALREADY DONE THIS, DON'T RUN THIS COMMAND AGAIN):

```
ykpersonalize -2 -ochal-resp -ochal-hmac -ohmac-lt64 -oserial-api-visible
```

Now we can enroll both yubikeys to the luks partition using mkinitcpio-ykfde.

```
sudo dnf install -y libyubikey-devel ykpers-devel iniparser-devel libarchive-devel cryptsetup-devel python-mark
git clone https://github.com/eworm-de/mkinitcpio-ykfde.git /home/$USER/fedora/mkinitcpio-ykfde
cd /home/$USER/fedora/mkinitcpio-ykfde
make MD=markdown_py
sudo make install-dracut
```

Get the serial number(s) from your Yubikey(s):

```
sudo ykinfo -s #with first yubikey
# serial: 9600174
sudo ykinfo -s #with second yubikey
# serial: 9243118
```

and write them down. Get the luks mapping name by looking into the crypttab (it's the first entry in each line):

```
sudo cat /etc/crypttab
# luks-6e7e8f26-4f38-468e-aa2c-9ddaaad4aedf UUID=6e7e8f26-4f38-468e-aa2c-9ddaaad4aedf none discard
```

Now, open `/etc/ykfde.conf` with a text editor and add your mapping name and serial numbers to the bottom of the file and save it:

```
sudo nano /etc/ykfde.conf
# [general]
# device name = luks-6e7e8f26-4f38-468e-aa2c-9ddaaad4aedf
# second factor = yes
#
# [9600174]
# luks slot = 1
#
# [9243118]
# luks slot = 2
```

Make sure to choose the appropriate LUKS slot; they are numbered from 0 and slot 0 is used by default and contains your passphrase chosen in the installer. Better check it with `cryptsetup luksDump /dev/vda3` if you have other slots occupied.

Now, add your Yubikey with your 2-factor password of choice:

```
## Put in your first yubikey
sudo ykfde --ask-new-2nd-factor
# Please give new second factor: ## put in a password of your choice
# Please give new second factor for verification: ## put in a password of your choice again
# Please give existing LUKS passphrase: ## put in your existing LUKS passphrase

## Put in your second yubikey
sudo ykfde --ask-new-2nd-factor
# Please give new second factor: ## put in a password of your choice
# Please give new second factor for verification: ## put in a password of your choice again
# Please give existing LUKS passphrase: ## put in your existing LUKS passphrase
```

Set up your Yubikey challenges to run at boot with these two commands:

```
sudo ykfde-cpio
sudo dracut -f
```

and update your GRUB bootloader configuration:

```
echo 'GRUB_EARLY_INITRD_LINUX_CUSTOM="ykfde-challenges.img"' | sudo tee -a /etc/default/grub
sudo grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg
```

Reboot and check whether you can unlock your luks partitions using your Yubikey and the chosen 2nd-factor password. Try to unlock with and without the Yubikeys to make sure everything works as you

expect.

Now, as I have a fully encrypted luks system and am the sole user of my computer, I can turn on automatic login in the settings.

Yubikey: private GPG key

Let's use the private GPG key on the Yubikey (a tutorial on how to put it there is taken from Heise or YubiKey-Guide). Installing some packages first:

```
sudo dnf install -y gpg gnupg2 gnupg-pkcs11-scd pcsc-tools opensc pcsc-lite-ccid
sudo systemctl start pcscd
sudo systemctl enable pcscd
```

Insert your yubikey and check whether the card is readable by gpg:

```
gpg --card-status
```

If there is a `No such device` message, then try restarting pcscd by running:

```
sudo systemctl restart pcscd
```

This is a known bug, that is, if you ever run into the issue that `gpg --card-status` does not find your Yubikey, simply run `sudo systemctl restart pcscd` and it'll work.

My public key is given in a file called `/home/$USER/.gnupg/public.asc`:

```
cd ~/.gnupg
gpg --import public.asc #this is my public key, my private one is on my yubikey
export KEYID=91E724BF17A73F6D
gpg --edit-key $KEYID
  trust
  5
  y
  quit
echo "This is an encrypted message" | gpg --encrypt --armor --recipient $KEYID -o encrypted.txt
gpg --decrypt --armor encrypted.txt
```

Apps

Browser

Firefox

I used to use Firefox for almost all of my browsing which is installed by default with the following:

- Extensions
  - Bitwarden
  - Disable HTML5 Autoplay
  - GNOME Shell-Integration
  - HTTPS Everywhere
  - uBlock Origin
- Plugins
  - OpenH264-Videocodec
  - Widevine Content Decryption Module
- Theme: firefox-gnome-theme:

```
git clone https://github.com/rafaelmardojai/firefox-gnome-theme/ /home/$USER/fedora/firefox-gnome-theme
cd /home/$USER/fedora/firefox-gnome-theme
./scripts/install.sh
```

Vivaldi

I am in the process of switching to Vivaldi:

```
sudo dnf install -y dnf-utils
sudo dnf config-manager --add-repo https://repo.vivaldi.com/archive/vivaldi-fedora.repo
sudo dnf install -y vivaldi-stable
```

I use the following Extensions installable from the Chrome store:

- Bitwarden
- GNOME Shell-Integration
- uBlock Origin

Google Chrome

If I ever need Google Chrome, then I enable the repo in the software manager and install it via the software shop.

Profile-sync-daemon

This neat little utility improves your browsing experience:

```
sudo dnf install -y profile-sync-daemon
psd
# First time running psd so please edit /home/$USER/.config/psd/psd.conf to your liking and run again
nano /home/$USER/.config/psd/psd.conf
# Close your browser now
systemctl --user enable psd.service
systemctl --user start psd.service
systemctl --user status psd.service
psd preview
```

System utilities

Flatseal

Flatseal is a great tool to check or change the permissions of your flatpaks:

```
flatpak install -y flatseal
```

Timeshift

For Timeshift, you would need to change the subvolume layout of Fedora 33. I have described these steps in my Fedora 33 installation guide.

Virtual machines: Quickemu and other stuff

Fedora by default has KVM, Qemu, virt-manager and gnome-boxes set up; however, I have found a much easier tool for most virtualization tasks: Quickqemu which uses the snap package `Qemu-virgil`:

```
git clone https://github.com/wmutschl/quickemu /home/$USER/fedora/quickemu
cd /home/$USER/fedora/quickemu
sudo dnf install -y bsd-games wget
sudo snap install qemu-virgil
sudo snap refresh qemu-virgil --edge
sudo snap connect qemu-virgil:audio-record
sudo snap connect qemu-virgil:kvm
sudo snap connect qemu-virgil:raw-usb
sudo snap connect qemu-virgil:removable-media

mkdir -p /home/$USER/.local/bin
ln -s /home/$USER/fedora/quickemu/quickemu /home/$USER/.local/bin/quickemu
```

I keep the conf files for my virtual machines on an external SSD.

Networking

Dropbox

Unfortunately, I still have some use case for Dropbox:

```
sudo dnf install -y dropbox nautilus-dropbox
```

Open dropbox and set it up, check options.

Nextcloud

I have all my files synced to my own Nextcloud server, so I need the sync client:

```
sudo dnf install -y nextcloud-client nextcloud-client-nautilus
```

Open Nextcloud and set it up. Recheck options and note to ignore hidden files once the first folder sync is set up.

I get two anoying issues with Nextcloud, which will probably be fixed in the future. For now the following works for me:

1. If you have many subfolders (which I do), there are not enough inotify-watches and Nextcloud does not sync instantenously but only periodically. This can be solved by

```
sudo -i
echo 'fs.inotify.max_user_watches = 524288' >> /etc/sysctl.conf
sysctl -p
```

The same issue happens with Visual Studio Code and the workaround is taken from there instructions page.

2. If you use X11 instead of Wayland, the app indicator icon does not show if I enable autostart of Nextcloud in its settings menu. The problem is, that while the nextcloud client is actually running after being autostarted, there is no tray icon (I use the 'KStatusNotifierItem/AppIndicator Support' extension). Whereas, if I start the client manually after logging in (without autostart or after killing the autostarted instance), the icon is there. For anyone experiencing this issue the workaround is to delay the autostart. That is, make the following changes to the .desktop file which resides in the subdirectory `~/.config/autostart` of the users home directory:

```
nano /home/$USER/.config/autostart/com.nextcloud.desktopclient.nextcloud.desktop
# [Desktop Entry]
# Categories=Utility;X-SuSE-SyncUtility;
# Type=Application
# Exec=bash -c 'sleep 5 && nextcloud'
# Name=Nextcloud desktop sync client
# Comment=Nextcloud desktop synchronization client
# GenericName=Folder Sync
# Icon=Nextcloud
# Keywords=Nextcloud;syncing;file;sharing;
# X-GNOME-Autostart-Delay=3
```

What it does is simply waiting 3+5 seconds before launching the client. Your mileage may vary - perhaps you need to give it more time if your startup takes longer than mine.

Alternatively, you might install TopIcons Plus Gnome Extension in addition to the KStatusNotifierItem/AppIndicator Support Extension. I set the 'Icon size' to 18 in the settings of TopIcons Plus, the 'Tray horizontal alignment' to 'Right' and 'Tray offset' to 1, see also Mattermost.

OpenConnect and OpenVPN

```
sudo dnf install -y openconnect NetworkManager-openconnect NetworkManager-openconnect-gnome
sudo dnf install -y openvpn NetworkManager-openvpn NetworkManager-openvpn-gnome
```

Go to Settings-Network-VPN and add openconnect for my university VPN and openvpn for ProtonVPN, check connections.

Remote desktop

To access our University remote Windows desktop session:

```
sudo dnf install -y rdesktop
echo "rdesktop -g 1680x900 wiwi-farm.uni-muenster.de -r disk:home=/home/$USER/ -u "WIWI\w_muts01" &" > ~/wiwi.s
chmod +x wiwi.sh
cat <<EOF > ~/.local/share/applications/wiwi.desktop
[Desktop Entry]
Name=WIWI Terminal Server
Comment=WIWI Terminal Server wiwi-farm
Keywords=WIWI;RDP;
Exec=/home/$USER/wiwi.sh
Icon=preferences-desktop-remote-desktop
Terminal=false
MimeType=application/x-remote-connection;x-scheme-handler/vnc;
Type=Application
StartupNotify=true
```

```
Categories=Network;RemoteAccess;
EOF
```

Note that this also adds a shortcut to the menu.

Coding

git related packages:

git and git-lfs are very important tools for me; as a GUI I like to use GitKraken:

```
sudo dnf install -y git git-lfs
git-lfs install
flatpak install -y gitkraken
```

The flatpak version of GitKraken works perfectly. Open GitKraken and set up Accounts and Settings (or restore from Backup see above). Note that for the flatpak version, one needs to add the following Custom Terminal Command: `flatpak-spawn --host gnome-terminal %d` to be able to open the repository quickly in the terminal.

Dynare related packages

I am a developer of Dynare and need these packages to compile it from source and run it optimally on a Fedora-based system:

```
# Minimal packages (use --disable-doc and --disable-octave flags)
sudo dnf install -y gcc gcc-g++ gfortran boost-devel gsl-devel lapack lapack-devel matio-devel openblas openbla
# Octave packages (use --disable-doc flag)
sudo dnf install -y octave octave-devel octave-statistics octave-io octave-optim octave-control
# Documentation packages
sudo dnf install -y texlive-scheme-minimal texlive-collection-publishers texlive-collection-latexextra texlive-
```

Next, we need to compile `slicot` and `x13as` from source as it is not packaged in Fedora (yet):

```
# slicot
mkdir -p /home/$USER/dynare/slicot
cd /home/$USER/dynare/slicot
wget https://deb.debian.org/debian/pool/main/s/slicot/slicot_5.0+20101122.orig.tar.gz
tar xf slicot_5.0+20101122.orig.tar.gz
cd slicot-5.0+20101122
make FORTRAN=gfortran OPTS="-O2 -fPIC -fdefault-integer-8" LOADER=gfortran lib
mkdir -p /home/$USER/dynare/slicot/lib
cp slicot.a /home/$USER/dynare/slicot/lib/libslicot64_pic.a #for MATLAB
cp slicot.a /home/$USER/dynare/slicot/lib/libslicot_pic.a #for octave

# x13as
mkdir -p /home/$USER/dynare/x13as
cd /home/$USER/dynare/x13as
wget https://www.census.gov/ts/x13as/unix/x13assrc_V1.1_B39.tar.gz
tar xf x13assrc_V1.1_B39.tar.gz
sed -i "s|-static| |" makefile.gf # this removes '-static' in the makefile.gf
make -f makefile.gf FFLAGS="-O2 -std=legacy" PROGRAM=x13as
mkdir -p /home/$USER/.local/bin
ln -s /home/$USER/dynare/x13as/x13as /home/$USER/.local/bin/x13as
```

Lastly, after installing MATLAB, I am compiling several versions of Dynare from source:

```
git clone --recurse-submodules --single-branch --branch 4.5 https://git.dynare.org/dynare/dynare.git /home/$USE
cd /home/$USER/dynare/stable-4.5
autoreconf -si
./configure --with-slicot=/home/$USER/dynare/slicot --with-matlab=/home/$USER/MATLAB/R2020b MATLAB_VERSION=R202
make -j$(($(nproc)+1)) #rule of thumb: one more than CPUs as shown by e.g. lscpu, better check

git clone --recurse-submodules --single-branch --branch 4.6 https://git.dynare.org/dynare/dynare.git /home/$USE
cd /home/$USER/dynare/stable-4.6
autoreconf -si
./configure --with-slicot=/home/$USER/dynare/slicot --with-matlab=/home/$USER/MATLAB/R2020b MATLAB_VERSION=R202
make -j$(($(nproc)+1)) #rule of thumb: one more than CPUs as shown by e.g. lscpu, better check

git clone --recurse-submodules --single-branch --branch master https://git.dynare.org/dynare/dynare.git /home/$
cd /home/$USER/dynare/unstable
autoreconf -si
./configure --with-slicot=/home/$USER/dynare/slicot --with-matlab=/home/$USER/MATLAB/R2020b
make -j$(($(nproc)+1)) #rule of thumb: one more than CPUs as shown by e.g. lscpu, better check
```

MATLAB

I have a license for MATLAB R2020a and R2020b; which I both install. In the following I will focus on the latest, but mention some details for the other version if needed.

First download the installation files from Mathworks, then unzip the installation files. The normal installer does not work for me as I am getting a "terminate called after throwing an instance of 'std::runtime_error'; what(): Unable to launch the MATLAB Window application Aborted" error, so I run the legacy installer:

```
unzip matlab_R2020b_glnxa64.zip -d /home/$USER/matlab_installer/R2020b
/home/$USER/matlab_installer/R2020b/bin/glnxa64/install_unix_legacy
```

Note that I don't run this as root as I am installing it into my home folder under `/home/$USER/MATLAB/R2020b` and will create the symbolic links myself. If you want to install MATLAB into the default directory `/usr/local/MATLAB/R2020b` you have to run the installer as root.

Unfortunately, there is no `matlab-support` package as for Debian-based systems, so I need to do several things by hand using the Arch Wiki on MATLAB as a main reference.

0. Make sure you have ownership of the configuration and installation files (probably not necessary)

```
mkdir -p /home/$USER/.matlab
sudo chown -R $USER:$USER /home/$USER/.matlab
sudo chown -R $USER:$USER /home/$USER/MATLAB
```

1. Create a symbolic link In order to open MATLAB from the terminal, create a symbolic link into your .local/bin folder:

```
mkdir -p /home/$USER/.local/bin
ln -s /home/$USER/MATLAB/R2020b/bin/matlab /home/$USER/.local/bin/matlab
```

Now open MATLAB in the terminal and activate your license. Make note of any errors and warnings in the terminal such as

```
matlab
# Gtk-Message: 21:10:59.44: Failed to load module "canberra-gtk-module"
# Gtk-Message: 21:10:59.44: Failed to load module "pk-gtk-module"
exit
```

Inside MATLAB try opening Settings and the Add-Ons Manager and make note of any errors. For instance, I get all sorts of `matlab.internal.webwindow` and `matlab.internal.cef.webwindow` when I try to open e.g. the AddOns window. We will solve this later, for now exit MATLAB again.

2. Create desktop entry

First download a MATLAB icon from here or here. Then we create a desktop entry for the local user:

```
cat << EOF > /home/$USER/.local/share/applications/matlabR2020b.desktop
[Desktop Entry]
Version=1.0
Type=Application
Terminal=false
Exec=/home/$USER/MATLAB/R2020b/bin/matlab -desktop
Name=MATLAB R2020b
Icon=/home/wmutschl/matlabicon_128.png
Categories=Development;Math;Science
Comment=Scientific computing environment
StartupNotify=true
StartupWMClass=sun-awt-X11-XFramePeer
MimeType=text/x-matlab
EOF
```

3. Use system libraries instead of MATLAB's shipped libraries Renaming or excluding some libraries from MATLAB solves a whole bunch of issues (actually almost all), so let's do it! First, install minimal development tools:

```
sudo dnf install -y gcc gcc-g++ gfortran
```

Then put the shipped libraries from MATLAB into exclude folders:

```
export MATLAB_ROOT=/home/wmutschl/MATLAB/R2020b
cd $MATLAB_ROOT/sys/os/glnxa64
mkdir -p exclude
mv libgcc_s* exclude/
mv libgfortran* exclude/ #only for R2020b, not for R2020a
mv libquadmath* exclude/
mv libstdc++* exclude/

cd $MATLAB_ROOT/bin/glnxa64
mkdir -p exclude
mv libfreetype* exclude/
mv libcrypto* exclude/

cd $MATLAB_ROOT/cefclient/sys/glib/glnxa64 || cd $MATLAB_ROOT/cefclient/sys/os/glnxa64 #the path has changed in
mkdir -p exclude
mv libgio* exclude/
mv libglib* exclude/
mv libgmodule* exclude/
mv libgobject* exclude/
mv libgthread* exclude/
```

4. OpenGL acceleration (and some other graphics issues)

Force Mesa to use the old driver which seems to be more performant on Gnome and solves many graphics issues in MATLAB like OpenGL acceleration:

```
echo 'MESA_LOADER_DRIVER_OVERRIDE=i965' | sudo tee -a /etc/environment
```

Open a new terminal (or better reboot) and check the following:

```
glxinfo | grep "direct rendering"
#should be yes
matlab -nodesktop -nosplash -r "opengl info; exit" | grep Software
#should be false
```

If you run into a shared resources-for-x11-graphics bug, this can be solved by

```
echo "-Djogl.disable.openglarbcontext=1" > /home/$USER/MATLAB/R2020b/bin/glnxa64/java.opts
```

5. gtk modules

This is more of a cosmetic issue if you run MATLAB from the terminal you get some warnings. To get rid of them:

```
sudo dnf install -y libcanberra-gtk2 PackageKit-gtk3-module

sudo bash -c 'cat > /etc/ld.so.conf.d/gtk2.conf << 'EOF'
/usr/lib64/gtk-2.0/modules
EOF'
sudo bash -c 'cat > /etc/ld.so.conf.d/gtk3.conf << 'EOF'
/usr/lib64/gtk-3.0/modules
EOF'
sudo ldconfig
matlab
#no more warnings
```

6. Help browser issues

The quick help (F1) only displays the help message once after that it stops working for me, unfortunately. I don't have a solution yet..., maybe `webutils.htmlrenderer('basic');` Clicking on Open Help Browser does give me the relevant help page, so this is not severe for me.

If MATLAB works as it should, I change some settings to use Windows type shortcuts on the Keyboard, add `mod` files as supported extensions. Also I do not use MATLAB's source control capabilities and enable antialiasing in the fonts section.

R

For teaching and data analysis there is nothing better than R and RStudio:

```
sudo dnf install -y R rstudio-desktop
```

Open rstudio, set it up to your liking.

Java via Openjdk

Install the default OpenJDK Runtime Environment:

```
sudo dnf install -y java-latest-openjdk
java -version
```

Visual Studio Code

I am in the process of transitioning all my coding to Visual Studio code:

```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
sudo sh -c 'echo -e "[code]\nname=Visual Studio Code\nbaseurl=https://packages.microsoft.com/yumrepos/vscode\ne
sudo dnf check-update
sudo dnf install -y code
```

I sync my settings and extensions inside VScode. Similar to Nextcloud there is an error labeled "Visual Studio Code is unable to watch for file changes in this large workspace" (error ENOSPC) which has to do with the limit of inotify. The workaround (if you haven't done so already) is to run:

```
sudo -i
echo 'fs.inotify.max_user_watches = 524288' >> /etc/sysctl.conf
sysctl -p
```

The same issue happens with Nextcloud.

Text-processing

Hugo

My website uses the Academic Template for Hugo, which is based on Go. As I need the extended version I don't install hugo from the repo, but instead download the official release binary from Github:

```
sudo dnf install -y golang #dependency I need

export HUGOVER=`curl --silent "https://api.github.com/repos/gohugoio/hugo/releases/latest" | grep -Po '"tag_nam
wget https://github.com/gohugoio/hugo/releases/download/v${HUGOVER:1}/hugo_extended_${HUGOVER:1}_Linux-64bit.ta
tar -xvf hugo_extended_${HUGOVER:1}_Linux-64bit.tar.gz hugo
rm hugo_extended_${HUGOVER:1}_Linux-64bit.tar.gz
mv hugo ~/.local/bin/hugo
hugo version
# Hugo Static Site Generator v0.78.2-959724F0/extended linux/amd64 BuildDate: ...
```

Latex related packages

I write all my papers and presentations with Latex using either TexStudio or VScode as editors:

```
sudo dnf install -y texlive-scheme-full
sudo dnf install -y texstudio
```

Open texstudio and set it up.

Microsoft Fonts

Sometimes I get documents which require fonts from Microsoft:

```
sudo dnf install -y curl cabextract xorg-x11-font-utils fontconfig
sudo rpm -i https://downloads.sourceforge.net/project/mscorefonts2/rpms/msttcore-fonts-installer-2.6-1.noarch.r
```

Masterpdf

I have purchased a license for Master PDF in case I need advanced PDF editing tools:

```
flatpak install -y masterpdf
```

Open masterpdf and enter license. Also I use flatseal to give the app full access to my home folder.

Softmaker Office

I have a personal license for Softmaker Office, which needs to be installed via its own repo:

```
sudo rpm --import https://shop.softmaker.com/repo/linux-repo-public.key
sudo sh -c 'echo -e "[SoftMaker_Office_Repository]\nname=SoftMaker Office Repository\nenabled=1\nautorefresh=1\
sudo dnf check-update
sudo dnf install -y softmaker-office-2018
```

Open it and enter license.

Zotero

Zotero is great to keep track of the literature I use in my research and teaching. I install it via a flatpak:

```
flatpak install -y zotero
```

Open zotero, log in to account, install extension better-bibtex and sync.

Communication

Mattermost

Our Dynare team communication is happening via Mattermost which can be installed via flatpak:

```
flatpak install -y Mattermost
```

Unfortunately, I still have an issue with the tray icon as it is only shown when turning the KStatusNotifierItem/AppIndicator Support Extension off and on again. However, what works for me is to additionally install the TopIcons Plus Gnome Extension. I set the 'Icon size' to 18 in the settings of TopIcons Plus, the 'Tray horizontal alignment' to 'Right' and 'Tray offset' to 1.

Skype

Skype can be installed either via snap or flatpak. I find the flatpak version works better with the system tray icons:

```
flatpak install -y skype
```

Open skype, log in and set up audio and video.

Zoom

Zoom can be installed either via snap or flatpak. I find the flatpak version works better with the system tray icons:

```
flatpak install -y zoom
```

Open zoom, log in and set up audio and video.

Multimedia

VLC

The best video player:

```
sudo dnf install -y vlc
```

Open it and check whether it works.

Multimedia Codecs

If you have VLC installed, you should be fine as it has builtin support for all relevant audio and video codecs. In other cases, I have found that the following commands install all required stuff for Audio and Video:

```
sudo dnf groupupdate sound-and-video
sudo dnf install -y libdvdcss
sudo dnf install -y gstreamer1-plugins-{bad-\*,good-\*,ugly-\*,base} gstreamer1-libav --exclude=gstreamer1-plug
sudo dnf install -y lame\* --exclude=lame-devel
sudo dnf group upgrade --with-optional Multimedia
```

For OpenH264 in Firefox I run:

```
sudo dnf config-manager --set-enabled fedora-cisco-openh264
sudo dnf install -y gstreamer1-plugin-openh264 mozilla-openh264
```

Afterwards you need to open Firefox, go to menu → Add-ons → Plugins and enable OpenH264 plugin. You can do a simple test whether your H.264 works in RTC on this page (check Require H.264 video).

OBS

I like that the snap version has all popular extensions included, so I use it:

```
sudo snap install obs-studio --edge
sudo snap connect obs-studio:audio-record
sudo snap connect obs-studio:avahi-control
sudo snap connect obs-studio:camera
sudo snap connect obs-studio:jack1
sudo snap connect obs-studio:joystick
sudo snap connect obs-studio:removable-media
```

Open OBS and set it up, import your scenes, etc.

Gnome Settings

- Set up Wifi, Ethernet and VPN
- Turn off bluetooth
- Change wallpaper
- Automatically delete recent files and trash
- Turn of screen after 15 min
- Turn on night mode
- Add online account for Nextcloud and Fedora
- Deactivate system sounds, mute mic
- Turn of suspend, shutdown for power button
- Turn on natural scrolling for mouse touchpad
- Go through keyboard shortcuts and adapt, I also add custom ones:
  - `xkill` on `CTRL + ALT + X`
  - `gnome-terminal` on `CTRL + ALT + T`
- Change clock to 24h format
- Display battery as percentage
- Check your default programs

Other stuff

- Bookmarks for netdrives: Using `CTRL + L` in nautilus, I can open the following links inside nautilus and add bookmarks to these drives for easy access:
  - university netdrive: `davs://w_muts01@wiwi-webdav.uni-muenster.de/`
  - university cluster `sftp://w_muts01@palma2c.uni-muenster.de`
  - personal homepage `sftp://mutschler.eu`
- Reorder Favorites: I like to reorder the favorites on the gnome launcher (when one hits the `SUPER`) key
- Go through all programs: Hit `META + A` and go through all programs, decide whether you need them or uninstall these
- Check autostart programs in Gnome Tweaks
- In the file manager preferences I enable “Sort folders before files
- Click on the clock and set the location for your weather forecast

PREVIOUS

Pop!_OS: Things to do after installation (Apps, Settings, and Tweaks)

NEXT

Ubuntu Desktop: Things to do after installation (Apps, Settings, and Tweaks)