Software

Tools to aid you

- Spell checking
- Fonts
- Polkit
- Firefox
- Google Chrome
- Discord
- Blu-ray
- Node.js (nvm)
- Kernel-based Virtual Machine (KVM)
- Folding@Home
- Timeshift
- GNOME Flatpaks

Spell checking

Hunspell is a spell checker and morphological analyzer library used by Firefox, Thunderbird, Chromium, LibreOffice and more.

Install the following packages to enable system-wide spell checking and hyphenation support (add languages for `hunspell` and `hyphen` at your discretion):

```
pacman -S hunspell hunspell-{de,en_US} hyphen hyphen-{de,en}
```

Fonts

For most desktop environments, a sufficient number of fonts is installed as dependencies. However, there's several additional packages for different styles and writing systems (latin vs. non-latin scripts). Arch Wiki has an extensive list of available fonts in both the repositories and the AUR. Installing the Noto font family also provides a vast coverage over a large array of scripts.

Configuration

Most applications read the font configuration provided by the `fontconfig` library. These configurations are written in XML and read from several different locations.

Location	Description
`/etc/fonts/fonts.conf`	Master configuration file **(not for editing!)**
`/etc/fonts/conf.d`	System-wide additional drop-in configuration files, hand-written or as symbolic links
`$XDG_CONFIG_HOME/fontconfig/fonts.conf`	Per-user config file
`$XDG_CONFIG_HOME/fontconfig/conf.d`	Per-user additional drop-in configuration files, hand-written or as symbolic links

Configuration files are read in and applied in lexical order. If you need rules applied in a specific order, make sure to prepend them with 2-digit numbers in the order you need.

A minimal `fontconfig` configuration file contains these headers:

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "urn:fontconfig:fonts.dtd">
<fontconfig>

  <!-- settings go here -->

</fontconfig>
```

Some font packages come with pre-defined rule sets, which are installed to `/usr/share/fontconfig/conf.avail/`. To apply them, it's best to create symbolic links to them in their respective drop-in configuration directories.

To apply them system-wide, link them from the `/etc/fonts/conf.d` directory:

```
cd /etc/fonts/conf.d
sudo ln -s /usr/share/fontconfig/conf.avail/70-no-bitmaps-except-emoji.conf
```

To apply them only to the currently logged in user, link them in the `$XDG_CONFIG_HOME/fontconfig/conf.d` directory:

> **HINT:** The environment variable `$XDG_CONFIG_HOME` should point to the `.config` sub-directory in your home directory. If it doesn't, use `$HOME/.config` instead for the examples or set it with `export`.

```
mkdir $XDG_CONFIG_HOME/fontconfig/conf.d
ln -s /usr/share/fontconfig/conf.avail/70-no-bitmaps-except-emoji.conf
$XDG_CONFIG_HOME/fontconfig/conf.d
```

Emoji Fonts

There are a few emoji fonts available on Arch.

Name	Package	Description
JoyPixels	`ttf-joypixels`	formerly EmojiOne, part of Emoji as a Service, proprietary
Noto Color Emoji	`noto-fonts-emoji`	Google open-source emoji font, color
Twemoji (Twitter Emoji)	`ttf-twemoji` (AUR)	Emoji for everyone, originally created by Twitter

Install your selected emoji font:

```
pacman -S noto-fonts-emoji
```

Applications requesting emoji to be displayed should pick up on the font after restarting them.

> **NOTE:** KDE sometimes applies emoji fonts incorrectly, either not showing them at all or showing the outline symbol version from a different font. You can fix this by installing `noto-color-emoji-fontconfig` from the AUR and creating a symbolic link to the configuration file as shown above.

Polkit

`polkit` is an application-level toolkit for defining and handling the policy that allows unprivileged processes to speak to privileged processes: It is a framework for centralizing the decision making process with respect to granting access to privileged operations for unprivileged applications.

Custom rules

Mount disks as user

Edit/create `/etc/polkit-1/rules.d/50-udisk.rules`

```
// Original rules: https://github.com/coldfix/udiskie/wiki/Permissions
// Changes: Added org.freedesktop.udisks2.filesystem-mount-system, as this is used by Dolphin.

polkit.addRule(function(action, subject) {
  var YES = polkit.Result.YES;
  // NOTE: there must be a comma at the end of each line except for the last:
  var permission = {
    // required for udisks1:
    "org.freedesktop.udisks.filesystem-mount": YES,
    "org.freedesktop.udisks.luks-unlock": YES,
    "org.freedesktop.udisks.drive-eject": YES,
    "org.freedesktop.udisks.drive-detach": YES,
    // required for udisks2:
    "org.freedesktop.udisks2.filesystem-mount": YES,
    "org.freedesktop.udisks2.encrypted-unlock": YES,
    "org.freedesktop.udisks2.eject-media": YES,
    "org.freedesktop.udisks2.power-off-drive": YES,
    // Dolphin specific
    "org.freedesktop.udisks2.filesystem-mount-system": YES,
    // required for udisks2 if using udiskie from another seat (e.g. systemd):
    "org.freedesktop.udisks2.filesystem-mount-other-seat": YES,
    "org.freedesktop.udisks2.filesystem-unmount-others": YES,
    "org.freedesktop.udisks2.encrypted-unlock-other-seat": YES,
```

```
    "org.freedesktop.udisks2.eject-media-other-seat": YES,
    "org.freedesktop.udisks2.power-off-drive-other-seat": YES
  };
  if (subject.isInGroup("storage")) {
    return permission[action.id];
  }
});
```

Firefox

Install Firefox via these packages (adjust for your desired locale):

```
pacman -S firefox firefox-i18n-de
```

Hardware Acceleration

Utilizing GPU hardware accelerated decoding of video content results in smoother playback of HD/4K content, while reducing CPU load and power draw (important to save on battery on laptops).

To ensure Firefox uses hardware decoding verify the following:

- The necessary VA-API drivers are installed (see: Graphics Cards)
- Navigate to `about:support` and verify that *Compositing* says *WebRender* (*WebRender Software* will **not** work)

Verify hardware video decoding

To verify Firefox is actually using VA-API to decode video you can launch it with the following command:

```
MOZ_LOG="FFmpegVideo:5" firefox 2>&1 | grep 'VA-API'
```

Start playing some video in Firefox and watch the logs on your terminal. If your log output reads something like the following video decoding via VA-API is working.

```
[RDD 97685: MediaPDecoder #1]: D/FFmpegVideo FFVPX: Initialising VA-API FFmpeg decoder
[RDD 97685: MediaPDecoder #2]: D/FFmpegVideo FFVPX:   VA-API FFmpeg init successful
[RDD 97685: MediaPDecoder #2]: D/FFmpegVideo FFVPX: Choosing FFmpeg pixel format for VA-API
video decoding.
[RDD 97685: MediaPDecoder #1]: D/FFmpegVideo FFVPX:   VA-API FFmpeg init successful
[RDD 97685: MediaPDecoder #2]: D/FFmpegVideo FFVPX: VA-API Got one frame output with pts=0
dts=0 duration=40000 opaque=-9223372036854775808
[RDD 97685: MediaPDecoder #1]: D/FFmpegVideo FFVPX: Initialising VA-API FFmpeg decoder
[RDD 97685: MediaPDecoder #1]: D/FFmpegVideo FFVPX:   VA-API FFmpeg init successful
[RDD 97685: MediaPDecoder #1]: D/FFmpegVideo FFVPX: VA-API Got one frame output with pts=40000
```

```
dts=40000 duration=40000 opaque=-9223372036854775808
[RDD 97685: MediaPDecoder #2]: D/FFmpegVideo FFVPX: VA-API Got one frame output with pts=80000
dts=80000 duration=40000 opaque=-9223372036854775808
[RDD 97685: MediaPDecoder #2]: D/FFmpegVideo FFVPX: VA-API Got one frame output with
pts=120000 dts=120000 duration=40000 opaque=-9223372036854775808
```

Customization

> **ATTENTION:** Firefox version 147 introduced support for the XDG Base Directory Specification. Firefox will not migrate old profiles to the new directory structure. If you've set up Firefox before version 147, the previous location for all things Firefox remains `~/.mozilla/`. This article assumes a fresh install.

Most customizations can be done in `about:config` from the browser UI. Settings that deviate from defaults are saved to `~/.config/mozilla/firefox/<user-profile>/prefs.js`.

It is possible to pre-set certain settings in a separate `user.js` file in the same directory to override defaults. Both files have the same syntax:

```
user_pref("setting.key.goes.here", value)
```

Autoplay in background

Firefox prevents autoplay for media of tabs that aren't currently active, which causes apps like Plex to take very long to skip to the next track after the current one has ended. The following setting in `about:config` can be used to disable this behavior:

Setting key	Value	Description
`media.block-autoplay-until-in-foreground`	`false`	Enable autoplay when tab is not currently active

Or via `user.js`:

```
user_pref("media.block-autoplay-until-in-foreground", false)
```

KDE Plasma Integration

For better integration of Firefox into the KDE Plasma desktop, install the Plasma Integration add-on either via the Mozilla Add-on page. It enables rich notifications support and download progress integration into the notification area of KDE Plasma.

To prevent duplicate entries in the Media Player widget or tray icon, set `media.hardwaremediakeys.enabled` to `false`. This disables the media entry from Firefox itself and only uses the one from the Plasma integration add-on.

Or via `user.js`:

```
user_pref("media.hardwaremediakeys.enabled", false)
```

XDG Portal Integrations

By default, Firefox uses GTK file and print dialogs, even on KDE. To change this to KDE native dialogs navigate to `about:config` and change the appropriate `widget.use-xdg-desktop-portal` settings to `1` (default is `2` which equates to auto-detection).

The settings are as follows:

Setting Key	Description
`widget.use-xdg-desktop-portal.file-picker`	Use file dialogs native to current desktop environment
`widget.use-xdg-desktop-portal.location`	Use GeoLocation services of current desktop environment
`widget.use-xdg-desktop-portal.mime-handler`	Use MIME handler of current desktop environment for opening files in external apps
`widget.use-xdg-desktop-portal.open-uri`	Use desktop environment for invoking local apps from websites
`widget.use-xdg-desktop-portal.settings`	Use desktop environment settings for dark/light mode among other things

Or via `user.js`:

```
user_pref("widget.use-xdg-desktop-portal.file-picker", 1);
user_pref("widget.use-xdg-desktop-portal.location", 1);
user_pref("widget.use-xdg-desktop-portal.mime-handler", 1);
user_pref("widget.use-xdg-desktop-portal.open-uri", 1);
user_pref("widget.use-xdg-desktop-portal.settings", 1);
```

Disable AI Integrations

Mozilla introduced multiple AI integrations, despite user pushback. To disable these set the following settings in `user.js`:

```
user_pref("browser.ml.chat.enabled", false);
user_pref("browser.ml.chat.menu", false);
user_pref("browser.ml.chat.page.footerBadge", false);
user_pref("browser.ml.chat.page.menuBadge", false);
user_pref("browser.ml.chat.page", false);
user_pref("browser.ml.enable", false);
user_pref("browser.ml.linkPreview.enabled", false);
user_pref("browser.ml.pageAssist.enabled", false);
user_pref("browser.ml.smartAssist.enabled", false);
user_pref("browser.search.visualSearch.featureGate", false);
user_pref("browser.tabs.groups.smart.enabled", false);
user_pref("browser.tabs.groups.smart.userEnabled", false);
user_pref("browser.urlbar.quicksuggest.mlEnabled", false);
user_pref("extensions.ml.enabled", false);
user_pref("pdfjs.enableAltText", false);
user_pref("places.semanticHistory.featureGate", false);
user_pref("sidebar.revamp", false);
```

Google Chrome

Install Google Chrome from AUR:

```
yay -S google-chrome
```

Tweaks

To enable hardware accelerated video decoding (with open source drivers) create a file at `~/.config/chrome-flags.conf` and add the following line in it:

```
--enable-features=VaapiVideoDecoder
```

Additionally, if you need to be able to share your screen wie WebRTC, you need to add the following line as well:

```
--enable-usermedia-screen-capturing
```

Furthermore, visit chrome://flags and set the following options to further tweak performance (use the search field to filter):

Setting key	Value	Description
`#enable-webrtc-pipewire-capturer`	`Enabled`	Uses PipeWire to capture the screen in Wayland sessions
`#enable-gpu-rasterization`	`Enabled`	Uses GPU for rasterization, boosting performance
`#enable-zero-copy`	`Enabled`	Accesses GPU memory directly, boosting performance
`#ozone-platform-hint`	`Auto`	Auto-detects which windowing system is currently in use (X11, Wayland)

Discord

Discord is a proprietary, cross-platform, all-in-one voice and text chat application.

Install Discord from the repositories:

```
pacman -S discord
```

Or the official Flatpak app:

```
flatpak install com.discordapp.Discord
```

Rich Presence with Flatpak

Discord provides a Unix socket file that games and applications use in order to show what game or media is currently being played.

The Flatpak version creates this socket in a different location than the repo package version. In order for games and apps to still be able to communicate with the Discord app, a symbolic link needs to be created in the usual place, pointing to the location of the socket file of the Flatpak version.

The most straightforward way is to create a symbolic link with a single command:

```
ln -sf $XDG_RUNTIME_DIR/{app/com.discordapp.Discord,}/discord-ipc-0
```

However, since `$XDG_RUNTIME_DIR` is a tmpfs in RAM, its contents will be discarded if the system is shut down or rebooted. To fix this, `systemd-tmpfiles` can be used to automatically re-create the link every time you log into your desktop session.

Create the directory `user-tmpfiles.d` in your user's `.config` directory:

```
mkdir -p ~/.config/user-tmpfiles.d
```

In the newly created directory, create a new config file:

```
nano ~/.config/user-tmpfiles.d/discord-ipc.conf
```

`systemd-tmpfiles` uses the information in the config file to create temporary files as long as the user is logged in and remove them once the user logs out.

The fields in this file are space separated. The fields are as follows:

Field	Description
Type	The type of file to create. `L` = symbolic link, `d` = directory, `f` = regular file.
Path	Path where to create the object. `%t` = signifier for user-specific temporary directory (`/run/user/[ID]`).
Mode	File permissions in octal notation. `-` means keep the default.
Owner	User who will own the file. `-` means keep the default.
Group	Group who will own the file. `-` means keep the default.
Time	Expiration time of the file, e.g. `2h` deletes the files after 2 hours. `-` means no expiration time.
Options	Parameters specific to the type used in the *type* field. When `L`, it's the target the link will point to.

In order for the link to point to the Discord socket of the Flatpak version, the resulting config file should look have this line:

```
L %t/discord-ipc-0 - - - - app/com.discordapp.Discord/discord-ipc-0
```

This will create a symbolic link at `/run/user/1000/discord-ipc-0` pointing to the actual socket file at `/run/user/1000/app/com.discordapp.Discord/discord-ipc-0` where the Flatpak version of Discord is listening on incoming connections from other apps and games.

`systemd-tmpfiles` will not pick up on the new config file automatically, it will however read it upon next log-in. Alternatively, run `systemd-tmpfiles --user --create` manually to have `systemd-tmpfiles` create the link for you.

Blu-ray

Playback

In order play Blu-Rays install the following packages:

```
sudo pacman -S libbluray libaacs
```

Additionally, a `KEYDB.cfg` file is needed. Download it from the FindVUK Online Database

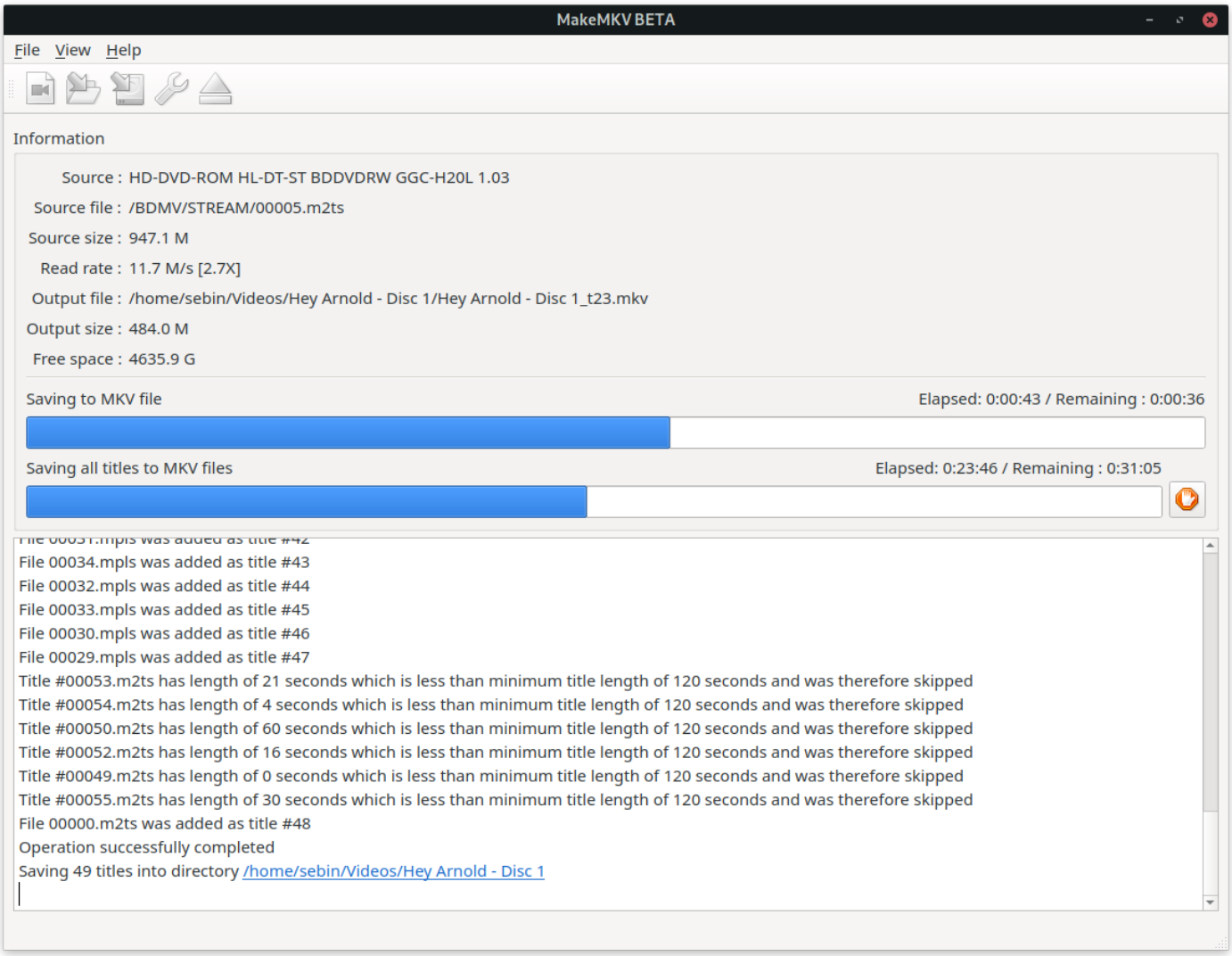
Extract the ZIP to `~/.config/aacs/`:

> **ATTENTION:** You may need to rename the `keydb.cfg` file to `KEYDB.cfg` (lower to upper case) for tooling to find it.

```
unzip keydb_eng.zip -d ~/.config/aacs/
```

After that use any Blu-Ray capable playback software, e.g. `vlc bluray:///dev/sr0` to play back Blu-Rays.

Ripping



In order to rip Blu-Rays install MakeMKV from the AUR:

```
yay -S makemkv
```

MakeMKV requires the `sg` (*SCSI generic (sg) driver*) kernel module to be loaded in order to recognize the drive. To load the module temporarily:

```
sudo modprobe sg
```

To have the kernel load the module on each boot:

```
echo sg | sudo tee > /etc/modules-load.d/sg.conf
```

Node.js (nvm)

Use the Node Version Manager (`nvm`) to install Node.js into your current user's path and switch Node.js versions on the fly.

Install `nvm` from the repositories:

```
pacman -S nvm
```

Include the init script `/usr/share/nvm/init-nvm.sh` into your shell configuration to load it each time you start your terminal:

```
# bash
echo 'source /usr/share/nvm/init-nvm.sh' >> ~/.bashrc


# zsh
echo 'source /usr/share/nvm/init-nvm.sh' >> ~/.zshrc
```

Restart your terminal to reload all init scripts and you should be able to use `nvm` to install a Node.js version of your choice:

```
nvm install 12
```

Migrating globally installed `npm` packages

When you install and switch to a different `nvm` managed version of Node.js (`nvm install 14` or `nvm use 16`) you may find that your globally installed `npm` packages (e.g. `svgo`) are no longer available until you switch back to the specific version of Node.js you have been using before the upgrade or switch.

This is because globally installed `npm` packages are installed for the specific version of Node.js you happen to be using at the time of installation and placed in a directory i.e. `~/.nvm/versions/node/v16.14.0/lib/node_modules`. When you install a different version, e.g. `17.2.0` the path to your Node.js installation changes to `~/.nvm/versions/node/v17.2.0/lib/node_modules`.

Use the `--reinstall-packages-from=<version>` option to carry over globally installed packages to the new Node.js installation.

You can either pass a specific version you want to reinstall globally installed packages from or use bash string expansion to reinstall from the currently active one in use:

```
nvm install <new version> --reinstall-packages-from=<old version>


nvm install 17 --reinstall-packages-from=$(node -v)
```

Kernel-based Virtual Machine (KVM)

Kernel-based Virtual Machine (KVM) is a full virtualization solution built into the Linux kernel. User space tools such as *libvirt* provide a standardized way to interface with virtualization engines, not only KVM but also Xen, OpenVZ and VirtualBox. Graphical tools like *virt-manager* allow for user-friendly management of virtual machines running on the local machine or on remote hosts running *libvirt*.

Preparation

KVM supports a wide range of guests, including Linux, BSD and even Windows. It's architecture allows for near-metal performance. To achieve this *KVM* utilizes hardware assisted virtualization technologies on the host machine's CPU (Intel VT, AMD-V).

On most desktop systems, these virtualization technologies are not enabled out of the box. To check if virtualization technologies are available on your system, use `lscpu`:

```
LC_ALL=C.UTF-8 lscpu | grep Virtualization
```

This will query the CPU's specifications and filter the output for the relevant virtualization feature section.

If the command produces **no output**, it indicates the system's CPU either does not support hardware assisted virtualization (unlikely if the machine's CPU was purchased in the last 10 years) or the feature is disabled in the machine's firmware options.

Hardware assisted virtualization technologies are a pre-requisite to being able to use KVM. Refer to your mainboard's user manual to learn how to enable the option.

> **NOTE:** Settings related to CPU virtualization features can sometimes be found in the "Overclocking" section of your firmware settings.

Also consider enabling IOMMU (Intel VT-d, AMD-Vi) for direct device pass-through.

>

> **NOTE:** On Intel-based systems, unless your kernel has the config option `CONFIG_INTEL_IOMMU_DEFAULT_ON` set (default is unset) you will also have to explicitly add `intel_iommu=on` to your kernel boot parameters.

Installation

The most common way to start using *KVM* is by installing *QEMU*, a generic and open source machine emulator and virtualizer. It utilizes *KVM* to achieve very good performance and can even emulate a different architecture from the one in the host machine. Though *KVM* is the most commonly used hypervisor for *QEMU*, it can also utilize other hypervisors, such as *Xen*, *OpenVZ* or *VirtualBox*.

Arch Linux offers several levels of completeness of the QEMU suite of emulators:

- `qemu-full`: installs the entirety of QEMU tools and libraries, capable of emulating many different systems and architectures.
- `qemu-desktop`: installs the essentials for running a QEMU environment for emulating x86_64 systems to run virtual machines on a desktop computer.
- `qemu-base`: the most basic QEMU environment intended for use on servers and headless environments.

Install the package that applies best for your use-case, e.g. if you plan on running virtual machines on your desktop computer:

```
sudo pacman -S qemu-desktop
```

QEMU itself does not provide graphical tools to set up and manage virtual machines. This is where *libvirt* comes in: it provides APIs for user-facing applications to offer graphical front-ends to users. One such application is *virt-manager* available from Arch repositories:

```
sudo pacman -S virt-manager libvirt
```

By default, only `root` can interface with *KVM* and thus the full system emulator provided by *QEMU* and *libvirt* will require elevated privileges every time you intend to run virtualized environments.

To allow your user to use virtual machines powered by *KVM* and *libvirt* add it to the `libvirt` group:

```
sudo usermod -aG libvirt $USER
```

> **NOTE:** The change will apply after logging out and back in again, or after a reboot. Feel free to continue the steps below until you actually start using QEMU.

Then, enable and start the *libvirt* daemon:

```
sudo systemctl enable --now libvirtd
```

Networking

For virtual machines to have network access, a network bridge needs to be created.

This is relatively easily achieved with `nmcli`, the CLI tool for *NetworkManager*:

```
# Create the new bridge interface
nmcli connection add \
      type bridge \
      ifname br0 \
      con-name "Bridge" \
      stp no

# Determine the current default route network adapter and add it to the bridge
DEFAULT_IF=$(ip r s default | awk '{print $5}')
nmcli connection add \
      type bridge-slave \
      ifname "$DEFAULT_IF" \
      con-name "Ethernet" \
      master br0

# Disable the old wired connection and bring up the bridge
nmcli connection down "Wired Connection 1"
nmcli connection up "Bridge"
```

To easily select the bridge network in a guest's network settings, create a small *libvirt* network XML definition file, e.g. as `br0.xml`:

```
<network>
  <name>br0</name>
  <forward mode='bridge'/>
  <bridge name='br0'/>
</network>
```

Import the definition file and set the network to autostart:

```
sudo virsh -c qemu:///system net-define br0.xml
sudo virsh -c qemu:///system net-autostart br0
```

Storage

Storage under *libvirt* is defined as **pools** which can be any of the following:

- a local directory
- a dedicated disk
- a pre-formatted block device
- an iSCSI target
- an LVM group
- a multi-path device enumerator (RAID device)
- an exported network directory
- a ZFS pool

The default storage pool is defined as a local directory at `/var/lib/libvirt/images`. This is where *libvirt* will store disk images for guests.

> **ATTENTION:** If your storage pool is on a copy-on-write file system, such as btrfs, it is recommended to disable CoW for that directory:
>
> ```
> sudo chattr +C /var/lib/libvirt/images
> ```

Storage pools can be created from within *virt-manager* or by writing custom XML definitions and importing them with `virsh`, in the case that the storage pool type is not exposed through the GUI or you need more fine-grained control over the specifications of the pool.

If you have a remote storage location that holds disk images, i.e. an exported NFS share with ISO images, it's possible to add it as a pool and mount it into the guest without the need to copy the images to your computer first.

The XML definition for a storage pool, named `remote-iso.xml` for example, could look something like this:

```
<pool type="netfs">
  <name>iso</name> <!-- name of the pool -->
  <source>
    <host name="dragonhoard"/> <!-- hostname/IP of remote machine -->
    <dir path="/mnt/user/downloads/ISOs"/> <!-- full path to images on remote machine -->
    <format type="auto"/>
```

```
  </source>
  <target>
    <path>/var/lib/libvirt/images/iso</path> <!-- mount point on local machine -->
  </target>
</pool>
```

In order for *libvirt* to successfully mount the network share, the mount point must exist prior to activating the pool:

```
sudo mkdir -p /var/lib/libvirt/images/iso
```

Finally, import the pool definition and set it to autostart:

```
sudo virsh -c qemu:///system pool-define remote-iso.xml
sudo virsh -c qemu:///system pool-autostart iso
```

When creating new virtual machines, the contents of the remote location are now easily selectable from within *virt-manager*'s creation wizard.

Folding@Home

Help scientists studying Alzheimer's, Huntington's, Parkinson's, and SARS-CoV-2 by simply running a piece of software on your computer. Add your computer to a network of millions of others around the world to form the world's largest distributed supercomputer.

Installlation

```
yay -S foldingathome opencl-amd
```

Configuration

Run `FAHClient --configure` as `root` to generate a configuration file at `/etc/foldingathome/config.xml`:

```
cd /etc/foldingathome
FAHClient --configure
```

Then start/enable the `foldingathome.service` systemd unit. NVIDIA users should also enable the `foldingathome-nvidia.service` systemd unit.

Example Configuration

```
<config>
  <!-- Slot Control -->
  <power v='FULL'/>

  <!-- User Information -->
  <passkey v='1234567890'/>
  <team v='45032'/>
  <user v='Registered_User_Name'/>

  <!-- Folding Slots -->
  <slot id='0' type='CPU'/>
```

```
  <slot id='1' type='GPU'/>
</config>
```

Timeshift

> **IMPORTANT:** Timeshift is **not a backup tool!** It only creates *local snapshots* of the system to roll back changes to the system. Do not rely on this mechanism to keep your data safe! Timeshift deletes the oldest snapshot when a new one is created and the maximum number of snapshots is reached. Furthermore, if the underlying file system is corrupted, the snapshots will be, too! Use a proper backup tool to keep your data safe on external data storage!

Timeshift helps create incremental snapshots of the file system at regular intervals, which can then be restored at a later date to undo all changes to the system.

It supports `rsync` snapshots for all filesystems, and uses the built-in snapshot features for Btrfs drives configured to use the `@` and `@home` subvolume layout for *root* and *home* directories respectively.

Installation

Timeshift is available from the Arch repos. It uses cron to make regularly scheduled backups. Install Timeshift with a cron daemon, e.g. `cronie`:

```
pacman -S timeshift cronie
```

Start and enable the cron scheduler for Timeshift to take regular snapshots:

```
sudo systemctl enable --now cronie
```

Finally, start Timeshift and complete the first time setup.

Automatic snapshots on system changes

In addition to Timeshift's periodic spanshots, `timeshift-autosnap` provides a `pacman` hook to create a manual snapshot every time packages are installed, upgraded or removed.

Install `timeshift-autosnap` from the AUR:

```
yay -S timeshift-autosnap
```

By default `timeshift-autosnap` only keeps 3 snapshots. To change this, edit `/etc/timeshift-autosnap.conf` and either set `deleteSnapshots` to `false` to never delete any snapshots or increase the number of `maxSnapshots`:

```
skipAutosnap=false
deleteSnapshots=true
maxSnapshots=7
updateGrub=true
snapshotDescription={timeshift-autosnap} {created before upgrade}
```

Prevent excessive snapshotting when using `yay`

By default, when installing or updating multiple packages from the AUR, `yay` first builds a package and immediately calls `pacman` to install it, before building and installing the next one on its list. This also means that the `timeshift-autosnap` hook is triggered **for each individual AUR package** built by `yay`, **including dependencies also installed from the AUR.**

This can have undesireable side-effects:

- `yay` will cause `timeshift-autosnap` to reach the `maxSnapshots` limit very quickly when installing multiple packages from the AUR, leaving you with snapshots with little to no meaningful changes between them
- if `deleteSnapshots` is set to `false` the amount of snapshots might quickly exhaust the usable space on the drive

To prevent this it is recommended to configure `yay` to:

1. not remove make dependencies after successfully built packages are installed
2. build all AUR packages first, install them all later
3. install AUR packages together with regular repo packages

By calling `yay` with the `--save` parameter, any options passed to it will be saved in a configuration file, e.g.:

```
yay --noremovemake --batchinstall --combinedupgrade --save
```

Next time you use `yay` to install, upgrade or remove packages it will read the generated config file at `~/.config/yay/config.json` and apply the options automatically without having to specify them during use.

GNOME Flatpaks

Core apps

Name	ID	Description
Calculator	`org.gnome.Calculator`	Perform arithmetic, scientific or financial calculations
Calendar	`org.gnome.Calendar`	Manage your schedule
Calls	`org.gnome.Calls`	Make phone and SIP calls
Camera	`org.gnome.Snapshot`	Take pictures and videos
Characters	`org.gnome.Characters`	Character map application
Clocks	`org.gnome.clocks`	Keep track of time
Color Profile Viewer	`org.gnome.ColorViewer`	Inspect and compare installed color profiles
Connections	`org.gnome.Connections`	View and use other desktops
Contacts	`org.gnome.Contacts`	Manage your contacts
Disk Usage Analyzer	`org.gnome.baobab`	Check folder sizes and available disk space
Document Scanner	`org.gnome.SimpleScan`	Make a digital copy of your photos and documents
Document Viewer	`org.gnome.Evince`	Document viewer for popular document formats
Extensions	`org.gnome.Extensions`	Manage your GNOME Extensions
Fonts	`org.gnome.font-viewer`	View fonts on your system
Image Viewer	`org.gnome.Loupe`	View images
Logs	`org.gnome.Logs`	View detailed event logs for the system
Maps	`org.gnome.Maps`	Find places around the world
Music	`org.gnome.Music`	Play and organize your music collection
Text Editor	`org.gnome.TextEditor`	Edit text files
Videos	`org.gnome.Totem`	Play movies

Name	ID	Description
Weather	`org.gnome.Weather`	Show weather conditions and forecast
Web	`org.gnome.Epiphany`	Browse the web

Internet

Name	ID	Description
Eolie	`org.gnome.Eolie`	Web browser
Evolution	`org.gnome.Evolution`	Manage your email, contacts and schedule
Fractal	`org.gnome.Fractal`	Chat on Matrix
Geary	`org.gnome.Geary`	Send and receive email
Polari	`org.gnome.Polari`	Talk to people on IRC

Multimedia

Name	ID	Description
Cheese	`org.gnome.Cheese`	Take photos and videos with your webcam, with fun graphical effects
Decibels	`org.gnome.Decibels`	Play audio files
EasyTAG	`org.gnome.EasyTAG`	Edit audio file metadata
Eye of GNOME	`org.gnome.eog`	Browse and rotate images
gThumb Image Viewer	`org.gnome.gThumb`	View and organize your images
Identity	`org.gnome.gitlab.YaLTeR.Identity`	Compare images and videos
Lollypop	`org.gnome.Lollypop`	Play and organize your music collection
Photos	`org.gnome.Photos`	Access, organize and share your photos on GNOME
Podcasts	`org.gnome.Podcasts`	Listen to your favorite shows
Rhythmbox	`org.gnome.Rhythmbox3`	Play and organize all your music
Shotwell	`org.gnome.Shotwell`	Digital photo organizer
Showtime	`org.gnome.Showtime`	Watch without distraction

Name	ID	Description
Sound Juicer	`org.gnome.SoundJuicer`	CD ripper with a clean interface and simple preferences
Sound Recorder	`org.gnome.SoundRecorder`	A simple, modern sound recorder for GNOME
Video Trimmer	`org.gnome.gitlab.YaLTeR.VideoTrimmer`	Trim videos quickly

Productivity

Name	ID	Description
Apostrophe	`org.gnome.gitlab.somas.Apostrophe`	Edit Markdown in style
Bookup	`org.gnome.gitlab.ilhooq.Bookup`	Streamline notes with Markdown!
Break Timer	`org.gnome.BreakTimer`	Computer break reminders for GNOME
Citations	`org.gnome.World.Citations`	Manage your bibliography
Endeavour	`org.gnome.Todo`	Manage your tasks
Fava	`org.gnome.gitlab.johannesjh.favagtk`	Do your finances using fava and beancount
Getting Things GNOME!	`org.gnome.GTG`	Personal tasks and TODO-list items organizer
Gnote	`org.gnome.Gnote`	A simple note-taking application
Hamster	`org.gnome.Hamster`	Personal time keeping tool
Iotas	`org.gnome.World.Iotas`	Simple note taking
Notes	`org.gnome.Notes`	Notes for GNOME
Papers	`org.gnome.Papers`	Read documents
Pinpoint	`org.gnome.Pinpoint`	Excellent presentations for hackers
Pulp	`org.gnome.gitlab.cheywood.Pulp`	Skim excessive feeds
Recipes	`org.gnome.Recipes`	GNOME loves to cook
Solanum	`org.gnome.Solanum`	Balance working time and break time
Translation Editor	`org.gnome.Gtranslator`	Translate and localize applications and libraries

Games

Name	ID	Description
Aisleriot Solitaire	`org.gnome.Aisleriot`	Play many different solitaire games
GNOME Chess	`org.gnome.Chess`	Play the classic two-player board game of chess
Crossword Editor	`org.gnome.Crosswords.Editor`	Create crossword puzzles
Crosswords	`org.gnome.Crosswords`	Solve crossword puzzles
Four-in-a-row	`org.gnome.Four-in-a-row`	Make lines of the same color to win
HexGL	`org.gnome.HexGL`	Space racing game
Hitori	`org.gnome.Hitori`	Play the Hitori puzzle game
GNOME Klotski	`org.gnome.Klotski`	Slide blocks to solve the puzzle
Lights Off	`org.gnome.LightsOff`	Turn off all the lights
Mahjongg	`org.gnome.Mahjongg`	Match tiles and clear the board
GNOME Mines	`org.gnome.Mines`	Clear hidden mines from a minefield
Nibbles	`org.gnome.Nibbles`	Guide a worm around a maze
Quadrapassel	`org.gnome.Quadrapassel`	Fit falling blocks together
Reversi	`org.gnome.Reversi`	Dominate the board in a classic reversi game, or play the reversed variant
GNOME Robots	`org.gnome.Robots`	Avoid the robots and make them crash into each other
GNOME Sudoku	`org.gnome.Sudoku`	Test yourself in the classic puzzle
Swell Foop	`org.gnome.SwellFoop`	Clear the screen by removing groups of colored and shaped tiles
Tali	`org.gnome.Tali`	Roll dice and score points
GNOME Taquin	`org.gnome.Taquin`	Slide tiles to their correct places
GNOME Tetravex	`org.gnome.Tetravex`	Reorder tiles to fit a square
GNOME 2048	`org.gnome.TwentyFortyEight`	Obtain the 2048 tile
Atomix	`org.gnome.atomix`	Build molecules out of single atoms
Five or More	`org.gnome.five-or-more`	Remove colored balls from the board by forming lines
gbrainy	`org.gnome.gbrainy`	gbrainy is a game to train memory, arithmetical, verbal and logical skills.
Convolution	`org.gnome.gitlab.bazylevnik0.Convolution`	Maze escaping game

Tools

Name	ID	Description
Brasero	`org.gnome.Brasero`	Create and copy CDs and DVDs
Buffer	`org.gnome.gitlab.cheywood.Buffer`	Embrace ephemeral text
Cowsay	`org.gnome.gitlab.Cowsay`	State of the art Cowsay generator
Déjà Dup Backups	`org.gnome.DejaDup`	Protect yourself from data loss
File Roller	`org.gnome.FileRoller`	Open, modify and create compressed archive files
Firmware	`org.gnome.Firmware`	Install firmware on devices
gedit	`org.gnome.gedit`	Text editor
GMetronome	`org.gnome.gitlab.dqpb.GMetronome`	Maintain a steady tempo
GNOME Network Displays	`org.gnome.NetworkDisplays`	Screencasting for GNOME
Keysign	`org.gnome.Keysign`	OpenPGP Keysigning helper
Passwords and Keys	`org.gnome.seahorse.Application`	Manage your passwords and encryption keys
Pika Backup	`org.gnome.World.PikaBackup`	Keep your data safe
Secrets	`org.gnome.World.Secrets`	Manage your passwords
Sushi	`org.gnome.NautilusPreviewer`	Provide a facility for quickly viewing different kinds of files

Software development

Name	ID	Description
Boxes	`org.gnome.Boxes`	Virtualization made simple
Builder	`org.gnome.Builder`	Create applications for GNOME
D-Spy	`org.gnome.dspy`	Analyze D-Bus connections
Devhelp	`org.gnome.Devhelp`	A developer tool for browsing and searching API documentation
GHex	`org.gnome.GHex`	Inspect and edit binary files
gitg	`org.gnome.gitg`	Graphical user interface for git
Glade	`org.gnome.Glade`	Create or open user interface designs for GTK+ applications
Meld	`org.gnome.meld`	Compare and merge your files