MocaccinoOS

Desktop Edition — Documentation

Source: https://www.mocaccino.org/docs/

Generated: February 2026

Mocaccino Desktop is a Gentoo-based Linux distribution oriented towards Desktop systems. It uses Luet as its package manager and delivers software as self-contained layers — making installation and upgrades as simple as possible.

Table of Contents

1. Documentation Overview

https://www.mocaccino.org/docs/

Welcome to the MocaccinoOS Desktop documentation. MocaccinoOS Desktop is a Gentoo-based Linux distribution using **Luet** as a package manager, which is static and uses containers to build packages.

Main Features

- Simple to install and use — get a full desktop up and running in minutes
- Applications ship as self-contained layers — install and uninstall without dependency headaches
- Rolling release with LTS and mainline kernel options — always up to date
- Wide desktop environment support: KDE Plasma, GNOME, XFCE, MATE, Hyprland, and more
- Gaming-ready out of the box: Steam, gamemode, VKBasalt, and Proton support
- Luet package manager with both a command-line interface and a graphical tool (Vajo)
- Flatpak, Snap, AppImage, and Wine supported for maximum software compatibility

2. Mocaccino Desktop

https://www.mocaccino.org/docs/desktop/

Mocaccino Desktop is a Gentoo-based distribution (derived from Sabayon) oriented towards Desktop systems. It contains installable "apps" referred to internally as "layers" to install common suite of packages needed to bootstrap a pure and simple OS.

A user should be able to install KDE Plasma by running `luet install layers/plasma` and nothing else. All micropackages are abstracted away and included in a layer installed as a single package — think Android apps: Install and uninstall should be that simple.

Why MOS Desktop is Different?

For the user, MOS Desktop is a pure and simple OS. Applications bundle all required dependencies in order to run, or share common layers used between them (e.g. MATE, GNOME, GTK-based apps). This allows OTA-alike updates without struggling with package dependencies.

From a developer standpoint, MOS takes a unique approach on package building, allowing developers to iterate locally changes to packages easily, using Docker to build packages locally or Kubernetes in a cluster.

2.1 Download and Install

https://www.mocaccino.org/docs/desktop/install/

Download

https://www.mocaccino.org/docs/desktop/install/download/

Official releases can be downloaded from the **Github repository**:
https://github.com/mocaccinoOS/mocaccino/releases

Login

The live images should log you into the graphical environment automatically. If you require a password for the default user:

- Default username and password is **mocaccino**

Installation

https://www.mocaccino.org/docs/desktop/install/installation/

This guide walks you through installing MocaccinoOS to your hard drive using the **Calamares graphical installer**. It applies to all editions including KDE Plasma, GNOME, XFCE, and MATE.

Step 1: Boot into the Live Environment
After downloading the MocaccinoOS ISO and writing it to a USB stick, boot your computer from the USB. You'll be greeted with the live session desktop. From here you can explore the live environment or go straight to installation.

Step 2: Launch the Calamares Installer
Look for the installer icon labeled "Install MocaccinoOS" on the desktop or in the application menu. Click to start the installation process.

Step 3: Choose Your Language
Select your preferred language for the installation process. This will also set the system language after installation. Click Next to continue.

Step 4: Select Your Location and Time Zone
Choose your region and time zone. This sets your system clock and affects time/date display. Click Next to proceed.

Step 5: Keyboard Layout
Choose the keyboard layout that matches your hardware or preferences. You can test your keyboard in the input field provided. Click Next when ready.

Step 6: Partition Your Disk
Choose how to partition your hard drive. Options: Erase disk (wipes entire disk), Replace a partition, or Manual partitioning. Make sure to back up important data before continuing. Click Next once you've made your choice.

Step 7: Create a User Account
Enter your full name, desired username, and password. Optionally choose to log in automatically or use the same password for the administrator account. Click Next to continue.

Step 8: Summary
Review the installation summary. Double-check partition choices and user information. If correct, click Install and confirm when prompted.

Step 9: Installation Progress
Calamares will now install MocaccinoOS to your disk. This can take a few minutes.

Step 10: Installation Complete
Once finished, you'll see a confirmation screen. You can choose to restart immediately or stay in the live session. After rebooting, log into your new MocaccinoOS system using the username and password you created. Remove the installation media when prompted and reboot.

2.2 Desktop Environments

https://www.mocaccino.org/docs/desktop/desktop_environments/

MocaccinoOS supports a wide range of desktop environments, all installable as single layer packages using Luet. After installing a desktop environment, restart your system and select it from the login screen.

KDE Plasma

```
sudo luet install layers/plasma layers/kde-apps-minimal apps/discover
```

The `layers/plasma` package ships with the SDDM login manager. To enable it:

```
sudo systemctl enable sddm --force
```

GNOME

```
sudo luet install layers/gnome apps/gnome-software
```

The `layers/gnome` package ships with the GDM login manager. To enable it:

```
sudo systemctl enable gdm --force
```

MATE

```
sudo luet install layers/mate themes/mate
```

XFCE

```
sudo luet install layers/xfce
```

LXQT

```
sudo luet install layers/lxqt
```

Cinnamon

```
sudo luet install layers/cinnamon
```

Enlightenment

```
sudo luet install layers/enlightenment
```

Trinity

```
sudo luet install layers/trinity
```

Hyprland

Hyprland is available in the community repository (install it first if you haven't). It includes the Kitty terminal emulator:

```
sudo luet install layers/hyprland apps/kitty
```

Fluxbox

```
sudo luet install layers/fluxbox
```

COSMIC

```
sudo luet install layers/cosmic
```

Login Managers

If you prefer **LightDM** as your login manager:

```
sudo luet install apps/lightdm sudo systemctl enable lightdm --force
```

If you prefer **Slim**:

```
sudo luet install apps/slim sudo systemctl enable slim --force
```

2.3 Package Management (Luet)

https://www.mocaccino.org/docs/desktop/luet/

MocaccinoOS uses **Luet** as a package manager. Learn how to work with Luet in the command shell or by using Vajo, the GUI/TUI tool.

Using the Command Shell

https://www.mocaccino.org/docs/desktop/luet/luet_cli_commands/

Common luet commands for package management:

`luet install <package>` — Install a package
`luet uninstall <package>` — Remove a package
`luet upgrade` — Upgrade all installed packages
`luet search <query>` — Search for packages in enabled repositories
`luet search --installed <query>` — Search among installed packages
`luet repo list` — List installed repositories
`luet repo update` — Update repository metadata

Vajo — GUI/TUI for Luet

https://www.mocaccino.org/docs/desktop/luet/vajo/

Vajo is MocaccinoOS's graphical and terminal-based frontend for the Luet package manager. It lets you browse, install, remove, and update packages without needing to use the command line directly.
To install Vajo:

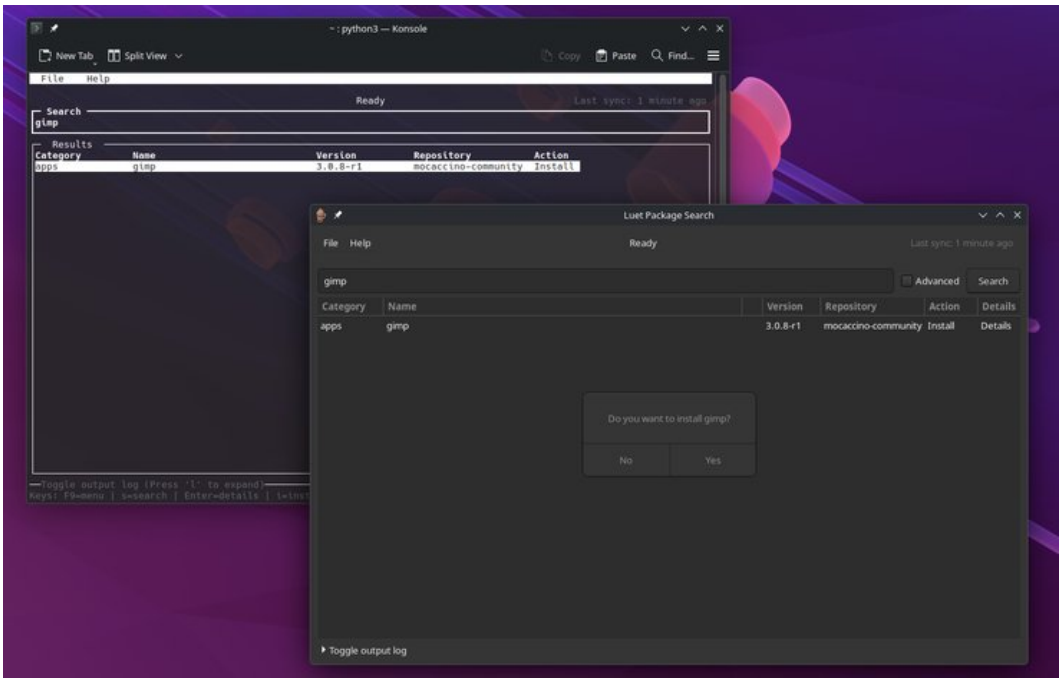
```
sudo luet install apps/vajo
```

Vajo ships in two modes:

- **GUI mode** — a full graphical application for desktop users. Launch it from your application menu or by running `vajo-gui`.
- **TUI mode** — an interactive ncurses terminal interface for users who prefer working in a terminal. Run `vajo-tui` to launch it.

Both modes provide the same core functionality:

- Browse and search available packages across all enabled repositories
- Install and remove packages with a single click or keypress
- Upgrade the entire system
- View installed packages and their details
- Manage repositories



Vajo TUI (left) and GUI (right) showing a search for GIMP

Repositories

https://www.mocaccino.org/docs/desktop/luet/repositories/

By default MocaccinoOS ships with the **mocaccino-desktop-stable** repository installed. This repository has all the core layers and desktop environments.

Community Repository

Any additional software compiled and packaged by the MocaccinoOS project is available in the community repository. For example, neovim, alacritty, or gimp require installing the community repository first:

```
$ sudo luet install repository/mocaccino-community-stable
```

Identify Installed Repositories

```
$ sudo luet repo list
```

Switch Between Stable and Development Repositories

■ **Warning:** Do not mix development and stable repositories! Be sure to only have installed stable or the development repositories.

To remove development repositories:

```
$ luet uninstall -y repository/mocaccino-extra repository/mocaccino-desktop
repository/mocaccino-os-commons
```

To enable stable repositories:

```
$ luet install -y --nodeps repository/mocaccino-extra-stable
repository/mocaccino-desktop-stable repository/mocaccino-os-commons-stable
```

2.4 Software Installation

https://www.mocaccino.org/docs/desktop/software/

Different ways to install and configure software on MocaccinoOS Desktop.

Docker

https://www.mocaccino.org/docs/desktop/software/docker/

Docker allows you to run applications in isolated containers. Install Docker and enable the service:

```
sudo luet install container/docker systemd-service/dockerd sudo systemctl enable docker sudo systemctl start docker
```

If you plan to use Docker as a regular user (not just root), also install these packages:

```
sudo luet install entity/docker acct-group/docker
```

Then add your user to the docker group so you can run docker commands without sudo:

```
sudo usermod -aG docker $USER
```

Log out and back in for the group change to take effect. Verify Docker is working:

```
docker run hello-world
```

Example: Apache Webserver with Docker

https://www.mocaccino.org/docs/desktop/software/docker/apache_with_docker/

A practical example of running a web server inside a Docker container.
Create the webserver document root in your home directory:

```
mkdir ~/website
```

Pull the Apache `httpd:2.4` container from Docker Hub and map the document root:

```
sudo docker run -dit --name my-web -p 8080:80 \ -v ~/website/:/usr/local/apache2/htdocs/ httpd:2.4
```

Confirm the container is running:

```
sudo docker ps
```

Create a simple homepage:

```
cat <<EOF >> ~/website/index.html <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>My website rocks!</title> </head> <body> <h1>Welcome to my homepage!</h1> </body> </html> EOF
```

Open http://localhost:8080/ in your browser to see the result.
When done, stop and clean up the container:

```
sudo docker stop my-web sudo docker rm my-web sudo docker image remove httpd:2.4
```

Flatpak

https://www.mocaccino.org/docs/desktop/software/flatpak/

MocaccinoOS Desktop supports Flatpak, and its usage is encouraged for packages like Spotify, Telegram, and others not found in the default desktop repository or community repository.

To install Flatpak, run as root:

```
sudo luet install apps/flatpak
```

Once Flatpak is installed, add the Flathub repository from your user account:

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

To install an app, for example Spotify:

```
flatpak install Spotify
```

Search

Search for applications across all added remotes:

```
flatpak search gimp
```

Running Applications

Run an installed application by its ID:

```
flatpak run org.gimp.GIMP
```

Updating

Update all installed Flatpak applications and runtimes:

```
flatpak update
```

Accessing External Paths

Flatpak runs applications in a sandbox that prevents access to host content. To expose a path to a specific app:

```
flatpak override --user --filesystem=/mnt org.app.Id
```

Flatseal is a GUI permissions manager offering simple point-and-click permission controls for Flatpak apps. Install it from Flathub: `flatpak install flathub com.github.tchx84.Flatseal`

Snap

https://www.mocaccino.org/docs/desktop/software/snap/

■ **Warning:** Using Snap is currently experimental on MocaccinoOS. Flatpak is the preferred alternative.

Besides the preferred Flatpak, MocaccinoOS also offers Snap support.

Installing and Configuring

To install Snap, run as root:

```
luet install apps/snapd
```

Once installed, enable the required service:

```
systemctl enable --now snapd.socket
```

AppArmor

Snap's fallback behavior without AppArmor is flaky, and some snaps won't work at all. AppArmor is pulled in as a dependency of `apps/snapd` so it is already on the system. Enable the AppArmor services:

```
systemctl enable --now apparmor.service snapd.apparmor.service
```

Enable the required boot parameters by editing `/etc/default/grub` and appending these to `GRUB_CMDLINE_LINUX`:

```
GRUB_CMDLINE_LINUX="apparmor=1 security=apparmor"
```

After editing, regenerate the GRUB config and restart:

```
sudo grub-mkconfig -o /boot/grub/grub.cfg
```

Using Snap

Install an application, for example Spotify:

```
sudo snap install spotify
```

Search for applications:

```
snap find spotify
```

Run an installed application:

```
snap run spotify
```

Update all installed Snap applications:

```
snap refresh
```

AppImage

https://www.mocaccino.org/docs/desktop/software/appimage/

AppImages are self-contained applications that run on any Linux distribution. Download an AppImage, make it executable, and run it directly. No installation required.

```
chmod +x application.AppImage ./application.AppImage
```

Whalebrew

https://www.mocaccino.org/docs/desktop/software/whalebrew/

Whalebrew allows you to install Docker images as if they were native applications.

Wine

https://www.mocaccino.org/docs/desktop/software/wine/

Wine is a compatibility layer that allows you to run many Microsoft Windows applications on Linux by translating Windows API calls into native Linux calls — without fully emulating Windows. This means Windows programs can often run nearly as fast as native Linux software.

Why use Wine: For Windows-only applications or games with no native Linux equivalent. It avoids the overhead of full virtualization, keeping resource use low. You don't need to dual-boot or run a full Windows VM.

Key limitations: Wine does not guarantee perfect compatibility — some applications run flawlessly, others only partially or not at all. For 32-bit applications on a 64-bit system, 32-bit compatibility libraries are needed. Drivers, low-level kernel integrations, and copy-protected software often don't work under Wine.

Wine is available in the **community repository**. If you have enabled this repository, install it with:

```
sudo luet install apps/wine-staging
```

The `apps/wine-staging` layer bundles Wine (Staging) with necessary 32-bit/multilib support, plus optional helper tools Winetricks and Protontricks — useful for installing common Windows runtimes (fonts, DirectX/3D libraries, .NET, etc.).

Configure Wine (initializes your default Wine prefix, creates a fake "C:" drive and registry):

```
winecfg
```

Run a Windows program:

```
wine /path/to/program.exe
```

For games or complex applications relying on 3D graphics, consider using **Lutris** (see Gaming section) which manages Wine versions and dependencies for you.

Waydroid

https://www.mocaccino.org/docs/desktop/software/waydroid/

Waydroid allows running Android applications on Linux through a container-based approach. It requires a Wayland session to work.

2.5 Gaming

https://www.mocaccino.org/docs/desktop/gaming/

Steam

We recommend installing Steam directly from the desktop repository. You can also use the Flatpak version, but that may give problems because it runs in a sandbox.

```
sudo luet install apps/steam
```

Gamemode

Gamemode is bundled in the `apps/steam` package — installing Steam gives you Gamemode automatically. To launch a game with Gamemode, right-click the game in the Steam Library, select Properties, and enter this in the Launch Options box:

```
gamemoderun %command%
```

VKBasalt

VKBasalt is a post-processing layer for Vulkan that enhances graphics. It only works with Vulkan games, including all Proton games. VKBasalt is also bundled in the `apps/steam` package. To enable it for a game, add this to Launch Options:

```
ENABLE_VKBASALT=1 %command%
```

For non-Steam games, run from the terminal:

```
ENABLE_VKBASALT=1 mygame
```

Additional Steam Library on NTFS

If you share an NTFS partition between Windows and MocaccinoOS containing a Steam library, this can cause issues launching games. Due to NTFS, filenames with characters invalid on Windows (such as a colon `:`) cause disk errors. Fix by symlinking the `/compatdata` folder to a Linux partition:

```
mkdir -p ~/.steam/steam/steamapps/compatdata ln -s ~/.steam/steam/steamapps/compatdata
/media/gamedisk/Steam/steamapps/
```

■ **Warning:** If a /compatdata folder already exists on the mounted NTFS disk BEFORE creating the symlink, DELETE IT first!

Proton GE

Proton GE is a fork of Valve's Proton by GloriousEggroll (Thomas Crider, who works at Red Hat). It includes additional patches and bleeding-edge technology that can increase performance and compatibility.
Proton GE is available in the **community repository**. If you have enabled this repository, install it with:

```
sudo luet install apps/proton-ge-custom-bin
```

After installation, restart Steam. Proton GE will then appear in the compatibility settings for any game.

Heroic Games Launcher

For installing and playing games from the Epic Games Store or GOG, we recommend **Heroic Games Launcher**. Install it via Flatpak:

```
flatpak install flathub com.heroicgameslauncher.hgl
```

Lutris

Lutris is an alternative gaming platform for Linux that manages Wine versions and game-specific configurations. It is available in the **community repository**. If you have enabled this repository, install it with:

```
sudo luet install apps/lutris
```

Nvidia Drivers

MocaccinoOS does not come with Nvidia drivers pre-installed. If you require proprietary drivers:

```
sudo luet install kernel-modules/nvidia-drivers-lts
```

MangoHud

MangoHud is a Vulkan and OpenGL overlay for monitoring FPS, temperatures, CPU/GPU load, and more. It is available in the **community repository**. If you have enabled this repository, install it with:

```
sudo luet install apps/mangohud
```

2.6 Printing

https://www.mocaccino.org/docs/desktop/printing/

CUPS (Common Unix Printing System) is pre-installed on MocaccinoOS, so you can start adding printers right away without any additional setup.

Adding a Printer via the CUPS Web Interface

CUPS includes a built-in web interface for managing printers. Open your browser and navigate to `http://localhost:631`.
Go to **Administration** → **Add Printer**. Authenticate with your system username and password (your account must have administrator privileges). CUPS will scan for available devices — select your printer from the list, or choose the connection type manually if it is not detected.
Fill in the printer details:

- **Name**: a short identifier with no spaces (e.g. `HP_OfficeJet`)
- **Description**: a human-friendly label (e.g. `HP OfficeJet Pro 9015`)
- **Location**: optional, useful in shared environments (e.g. `Office Room 2`)

Click **Continue**, then select a driver (CUPS will suggest one based on the detected model — or upload a PPD file from your manufacturer). Click **Add Printer**, then **Set Default Options** to configure paper size, quality, and duplex settings.

Network & Shared Printers

Network Printer (IPP / AirPrint): Most modern network printers support IPP. In the CUPS Web UI, go to **Administration** → **Add Printer**, select **Internet Printing Protocol (IPP)**, and enter the URI:

```
ipp://<printer-ip-address>/ipp/print
```

Printer shared by another Linux machine (CUPS-to-CUPS): It may be discovered automatically under **Discovered Network Printers**. If not, add it manually:

```
ipp://<host-ip>:631/printers/<printer-name>
```

Printer shared by a Windows machine (Samba/SMB): Select **Windows Printer via SAMBA** as the connection type and enter:

```
smb://<host-ip>/<printer-name>
```

Enter Windows credentials when prompted, then complete driver selection as usual.

Managing Printers

Manage all printers from `http://localhost:631` under the **Printers** tab. From there you can pause/resume a printer, set a default, view and cancel print jobs, modify options, or remove a printer.

Printing on KDE Plasma

KDE Plasma integrates with CUPS through the **Print Manager** tool. Open **System Settings** → **Printers** and authenticate. Click **Add Printer**, select your printer, accept or choose a driver, give it a name and description, then click **Finish**. Return to **System Settings** → **Printers** at any time to manage jobs or set a default.

Printing on GNOME

Open **Settings** → **Printers**, click **Unlock** and authenticate. Click the **+** button to add a printer — GNOME will show detected printers. USB printers are often added automatically when connected. Click the **settings (■)** button next to an existing printer to modify options, set it as default, view jobs, or remove it.

2.7 Network Configuration

https://www.mocaccino.org/docs/desktop/network/

MocaccinoOS uses **Network Manager** to manage network connections by default. It handles wired and wireless connections automatically and can be controlled via the `nmcli` command line tool, a graphical applet in your desktop environment, or the `nmtui` text interface.

Checking Status

```
nmcli general status # verify NM is running nmcli device status # list all interfaces nmcli
connection show # list configured profiles
```

Wi-Fi

```
nmcli device wifi list # scan for networks nmcli device wifi connect <SSID> password <password>
# connect nmcli device disconnect <interface> # disconnect nmcli radio wifi on / off # enable
or disable Wi-Fi
```

Wired (Ethernet)

Wired connections are configured automatically via DHCP when a cable is plugged in. To bring a connection up or down manually:

```
nmcli connection up <connection-name> nmcli connection down <connection-name>
```

Static IP

Assign a static IP to an existing connection:

```
nmcli connection modify <connection-name> \ ipv4.method manual \ ipv4.addresses 192.168.1.50/24
\ ipv4.gateway 192.168.1.1 \ ipv4.dns 192.168.1.1 nmcli connection up <connection-name>
```

To revert back to DHCP:

```
nmcli connection modify <connection-name> ipv4.method auto nmcli connection up
<connection-name>
```

nmtui

For a simple text-based interface, Network Manager includes `nmtui` — a menu-driven interface where you can add, edit, and activate connections without needing to remember `nmcli` syntax:

```
nmtui
```

Switching to systemd-networkd

For users who prefer a more minimal setup, **systemd-networkd** is available as an alternative.
Disable Network Manager and enable systemd-networkd:

```
systemctl stop NetworkManager systemctl disable NetworkManager systemctl enable --now
systemd-networkd systemctl enable --now systemd-resolved
```

Find your interface name:

```
networkctl
```

This will display output like:

```
IDX LINK TYPE OPERATIONAL SETUP 1 lo loopback carrier unmanaged 2 enp3s0 ether routable
configured 3 enp2s0 ether off unmanaged
```

Create a configuration file for your interface (replace `enp3s0` with your interface name):

```
nano /etc/systemd/network/00-enp3s0.network
```

Static IP:

```
[Match] Name=enp3s0 [Network] Address=192.168.1.50/24 Gateway=192.168.1.1 DNS=192.168.1.1
```

DHCP:

```
[Match] Name=enp3s0 [Network] DHCP=yes
```

After saving the file, restart the service to apply changes:

```
systemctl restart systemd-networkd
```

2.8 File Sharing

https://www.mocaccino.org/docs/desktop/file-sharing/

MocaccinoOS Desktop provides several protocols for sharing files over a network: Samba (for Windows/macOS compatibility), NFS (for Linux-to-Linux sharing), and SFTP (secure transfer over SSH).

Samba Configuration

https://www.mocaccino.org/docs/desktop/file-sharing/samba/

Samba allows MocaccinoOS to share files and printers with Windows, macOS, and other Linux systems using the SMB protocol. The core packages are already installed — you only need to configure it.

Configure Global Settings

MocaccinoOS provides a default config file. Copy it before making any changes:

```
sudo cp /etc/samba/smb.conf.default /etc/samba/smb.conf
```

The config file is now at `/etc/samba/smb.conf`. The default `[global]` section works for most simple setups.

Define a Share

To create a shared folder, add a section to the end of `/etc/samba/smb.conf`:

```
[Shared_Docs] comment = Collaborative Document Sharing path = /mnt/storage/shared_data
browsable = yes guest ok = no read only = no create mask = 0775 directory mask = 0775 valid
users = @staff, username
```

Key options: `path` — the directory to share. `guest ok = no` requires authentication. `valid users` restricts access to specific users or groups (prefix groups with @).

Add Samba Users

Samba maintains its own password database separate from Linux. Any user who needs access must be added:

```
sudo smbpasswd -a username
```

Start and Enable Samba

```
sudo systemctl enable smb sudo systemctl start smb
```

NFS Configuration

https://www.mocaccino.org/docs/desktop/file-sharing/nfs/

NFS (Network File System) is ideal for sharing directories between Linux systems on a local network. Install the required utilities:

```
luet install net-fs/nfs-utils
```

To share a directory, add it to `/etc/exports`:

```
/srv/share 192.168.1.0/24(rw,sync,no_subtree_check)
```

Apply the export and enable the service:

```
sudo exportfs -ra sudo systemctl enable nfs-server sudo systemctl start nfs-server
```

On the client, mount the share:

```
sudo mount server-ip:/srv/share /mnt/nfs
```

To mount automatically on boot, add to `/etc/fstab`:

```
server-ip:/srv/share /mnt/nfs nfs defaults 0 0
```

SFTP Configuration

https://www.mocaccino.org/docs/desktop/file-sharing/sftp/

SFTP (Secure File Transfer Protocol) transfers files over an encrypted SSH connection. MocaccinoOS includes OpenSSH, so no additional installation is needed.
Enable and start the SSH service:

```
sudo systemctl enable sshd sudo systemctl start sshd
```

Connect from a remote machine:

```
sftp username@your-ip-address
```

For graphical SFTP access, use a file manager that supports SFTP (e.g. Dolphin, Nautilus, Thunar) or a dedicated client like FileZilla.

2.9 Internationalization

https://www.mocaccino.org/docs/desktop/internationalization/

MocaccinoOS supports internationalization features which are not installed by default but can be added at any time. The following features are available:

- Input methods (Chinese, Japanese, Korean, etc.) — see Fcitx or IBus sections below
- Fonts for additional scripts (CJK, emoji, etc.)
- Locale and language data

Input Method — Fcitx

https://www.mocaccino.org/docs/desktop/internationalization/input_method/fcitx/

Fcitx is an input method framework with support for Chinese, Japanese, Korean, and other languages. Install the Fcitx layer:

```
luet install layers/fcitx
```

The `layers/fcitx` package includes fcitx-based input methods including Chinese table and phonetic engines.

Input Method — IBus

https://www.mocaccino.org/docs/desktop/internationalization/input_method/ibus/

MocaccinoOS provides the IBus input method framework as an alternative to Fcitx. IBus is commonly used for Asian language input and integrates well with GTK applications and GNOME-based environments.

Install IBus:

```
luet install layers/ibus
```

This installs the IBus framework along with Hangul, Pinyin, and table-based input methods.

Enable in KDE Plasma

Open **System Settings** → **Input Devices** → **Virtual Keyboard**, select **IBus**, then log out and back in.

Configure Input Methods

Run the IBus setup tool to add your desired input methods (e.g. Hangul, Pinyin, Table):

```
ibus-setup
```

If needed on non-Plasma or X11 sessions, ensure these environment variables are set:

```
export GTK_IM_MODULE=ibus export QT_IM_MODULE=ibus export XMODIFIERS=@im=ibus
```

IBus works in both X11 and Wayland sessions. GTK applications have native integration; Qt is supported via IBus modules.

Spell Checking Dictionaries

https://www.mocaccino.org/docs/desktop/internationalization/spellcheck/

MocaccinoOS provides multilingual spell-checking dictionaries via the `myspell-dicts` package group. These dictionaries are used by applications such as LibreOffice, Firefox, Chromium, Kate, KWrite, and other GTK and Qt applications with spell checking support.

Install all dictionaries:

```
luet install app/myspell-dicts
```

This installs MySpell dictionaries for many languages, including: Afrikaans, Bulgarian, Catalan, Czech, Danish, German, Greek, English, Esperanto, Spanish, Estonian, French, Irish, Galician, Hebrew, Croatian,

Hungarian, Icelandic, Italian, Kazakh, Khmer, Lithuanian, Latvian, Macedonian, Malay, Norwegian (Bokmål/Nynorsk), Dutch, Polish, Portuguese (EU/BR), Romanian, Russian, Slovak, Slovenian, Swedish, and Ukrainian.

Usage

After installation, supported applications automatically detect available dictionaries. Select your language via **Tools** → **Spell Checking Language** or by right-clicking and choosing **Language**. No additional configuration is required — multiple languages can be active simultaneously.

2.10 Virtualization

https://www.mocaccino.org/docs/desktop/virtualization/

MocaccinoOS supports virtualization using KVM/QEMU and libvirt.

Virt-manager

https://www.mocaccino.org/docs/desktop/virtualization/virt-manager/

Virt-manager is a graphical interface for managing virtual machines via libvirt. Install it:

```
luet install apps/virt-manager
```

Virt-manager provides a desktop user interface for managing virtual machines through libvirt, allowing you to create, configure, and control virtual machines from a GUI.

2.11 Performance Optimizations

https://www.mocaccino.org/docs/desktop/performance/

MocaccinoOS includes optimized system and network settings by default. These tweaks enhance networking, memory management, and filesystem efficiency. Some settings are enabled by default; additional examples are provided for users who want further customization.

■ **Warning:** Do NOT edit /usr/lib/sysctl.d/70-mocaccino.conf — it will be overwritten on system updates. To make custom changes permanent, create /etc/sysctl.d/99-custom.conf instead.

Default Optimizations (Already Enabled)

The following settings are active out of the box:

```
net.core.default_qdisc=fq # FQ queuing: reduces bufferbloat net.ipv4.tcp_congestion_control=bbr
# BBR: low-latency congestion control vm.swappiness=10 # Reduce swap usage for responsiveness
vm.dirty_ratio=20 # Prevent stutters via disk writeback vm.dirty_background_ratio=5
vm.vfs_cache_pressure=50 # Better I/O performance fs.inotify.max_user_watches=524288 # For
IDEs, Docker, game engines fs.inotify.max_user_instances=1024
```

Customizing TCP Buffer Sizes

■ **Warning:** Only use ONE of the following profiles at a time — do not mix them.

Low Latency (Gaming, Video Calls):

```
net.core.rmem_max=4194304 net.core.wmem_max=4194304 net.ipv4.tcp_rmem=4096 87380 4194304
net.ipv4.tcp_wmem=4096 16384 4194304
```

High Bandwidth (File Transfers, Streaming):

```
net.core.rmem_max=16777216 net.core.wmem_max=16777216 net.ipv4.tcp_rmem=4096 87380 16777216
net.ipv4.tcp_wmem=4096 65536 16777216
```

Copy your chosen settings into `/etc/sysctl.d/99-custom.conf` to make them permanent.

Adjusting vm.max_map_count

The default `vm.max_map_count=65530` works for most users. Some applications (Elasticsearch, game engines, AI tools) may need a higher limit:

```
echo "vm.max_map_count=262144" >> /etc/sysctl.d/99-custom.conf sudo sysctl --system
```

Recommended values: 65530 (default), 262144 (databases/Elasticsearch), 1048576+ (high-performance workloads). Note: the game DayZ requires 1048576.

Applying Changes

After modifying `/etc/sysctl.d/99-custom.conf`, apply immediately without rebooting:

```
sudo sysctl --system
```

To verify a setting:

```
sysctl net.ipv4.tcp_congestion_control
```

To restore defaults, delete your custom file and re-apply:

```
sudo rm /etc/sysctl.d/99-custom.conf sudo sysctl --system
```

3. General Section

https://www.mocaccino.org/docs/general/

■ **Warning:** Note: Mocaccino and Luet are under active development. Some features may change between updates. It is always a good idea to back up important data before making significant system changes.

This section covers system configuration topics that apply to all MocaccinoOS installations.

3.1 Switching Kernels

https://www.mocaccino.org/docs/general/kernels/

MocaccinoOS images ship with the latest LTS kernel by default. It is possible to switch kernels by running `mos kernel-switcher`.

■ **Warning:** WARNING: Due to a known bug, after switching kernels you may end up without a working initramfs, causing the system to fail to boot. Make sure you have the proper kernel/mocaccino-lts-initramfs or kernel/mocaccino-initramfs installed BEFORE rebooting! See: https://github.com/mocaccinoOS/mocaccino/issues/109

List Available Kernels

```
mos kernel-switcher list
```

Switch to Main Kernel

To switch to the main kernel (updated more frequently):

```
mos kernel-switcher switch kernel/mocaccino-full
```

Switch to LTS Kernel

To switch to the LTS kernel:

```
mos kernel-switcher switch kernel/mocaccino-lts-full
```

3.2 System Config Files

https://www.mocaccino.org/docs/general/system-config/

MocaccinoOS Desktop automatically protects `/etc` from overwriting user-override changes. However, upgrades might require overwriting and upgrading specific default configuration. The `mos` CLI embeds a

`config-update` tool compatible with `etc-update` and `dispatch-conf`.

Check for Pending Config Changes

To show a summary of config changes that require merging:

```
$ mos config-update check Files with unmerged config files: 1 - /etc/test/foo (2 unmerged
config files)
```

Merging Configuration

To merge configurations interactively:

```
mos config-update update
```

To check and merge all files one by one:

```
mos config-update update -a
```

Auto-merging

To auto-merge all configurations without interactive prompts:

```
mos config-update update --interactive=false
```

4. Contribution Guidelines

https://www.mocaccino.org/docs/contribution-guidelines/

Contributing to MocaccinoOS. The project is open-source and community-driven. Contributions are welcome in all forms:

- **Bug Reports:** Open an issue at https://github.com/mocaccinoOS/mocaccino/issues
- **Pull Requests:** Fork the relevant repository and submit a PR
- **Documentation:** Help improve the docs by editing pages on GitHub
- **Package Requests:** Request new packages for the community repository
- **Package Contributions:** Submit PRs to the community repository

When contributing to MocaccinoOS Desktop packages, make sure to:

- Read the Luet documentation on package building
- Follow the existing package structure and conventions
- Test changes on a test system before submitting
- Revbump reverse dependencies when making breaking changes

Key repositories:

- Main repository: https://github.com/mocaccinoOS/mocaccino
- Desktop packages: https://github.com/mocaccinoOS/desktop
- Community repository: https://github.com/mocaccinoOS/community-repository
- Website/Docs: https://github.com/MocaccinoOS/website

5. Support / Donate

https://www.mocaccino.org/docs/donate/

MocaccinoOS is currently supported by community donations. While development continues, funding helps sustain long-term work.

MocaccinoOS builds on the incredible work of the Gentoo Linux community. All packages are compiled on the project's own infrastructure powered by Kairos.

If you find MocaccinoOS useful and would like to help keep the project healthy, consider supporting via donations.

Donate page: https://www.mocaccino.org/docs/donate/

Appendix: Quick Reference

Common Luet Commands

`luet install <package>`
Install a package

`luet uninstall <package>`
Uninstall a package

`luet upgrade`
Upgrade all packages

`luet search <query>`
Search packages in repositories

`luet search --installed <query>`
Search installed packages

`luet repo list`
List installed repositories

`luet repo update`
Update repository metadata

`luet cleanup`
Clean up cache and downloaded packages

`luet install repository/<repo-name>`
Add a repository

`luet uninstall repository/<repo-name>`
Remove a repository

Key Repositories

- **mocaccino-desktop-stable** — Core layers and desktop environments (installed by default)
- **mocaccino-community-stable** — Community packages (neovim, alacritty, gimp, etc.)
- **mocaccino-extra-stable** — Extra MocaccinoOS packages
- **luet** — Official Luet repository

Useful Links

- **Documentation**: https://www.mocaccino.org/docs/
- **Issue Tracker**: https://github.com/mocaccinoOS/mocaccino/issues
- **Packages**: https://packages.mocaccino.org
- **Community**: https://www.mocaccino.org/community/
- **GitHub Organization**: https://github.com/mocaccinoOS
- **Luet Documentation**: https://luet-lab.github.io/docs/