Arch Linux ThinkPad T480s

Automated Arch Linux installation with LUKS encryption, btrfs, Hyprland.

What's Included

- Ventoy USB setup with install script, dotfiles, password store
- Automated installer: LUKS2, btrfs snapshots, systemd-boot
- Hyprland + Waybar + PipeWire

1. Create Ventoy USB

From any Linux:

```
curl -LO
↪  https://github.com/ventoy/Ventoy/releases/download/v1.1.10/ventoy-1.1.10-linux.tar.gz
tar xzf ventoy-1.1.10-linux.tar.gz
cd ventoy-1.1.10
sudo sh Ventoy2Disk.sh -i /dev/sdX
```

Copy to USB: - `archlinux-2026.01.01-x86_64.iso` - `arch-install.sh` - `dotfiles/` - `pass-store/`

If copying from macOS, clean up resource fork files:

```
dot_clean /Volumes/Ventoy/dotfiles
```

2. Install Arch

Boot USB -> Ventoy -> Arch ISO

```
mkdir -p /run/archusb
udevadm trigger                       # Initialize Ventoy device mapper
mount /dev/mapper/sda1 /run/archusb   # Use /dev/mapper/, not /dev/sda1
/run/archusb/arch-install.sh
```

> **Note:** Ventoy requires mounting via `/dev/mapper/sdX1`, not the raw block device. If `/dev/mapper/sda1` doesn't exist, run `dmsetup ls` to find the correct device.

When installation completes: - Do NOT remove USB - you need it for dotfiles in next step - Reboot and select HDD in boot menu (not USB)

3. Post-Install (First Boot)

Login as user. DO NOT start Hyprland yet.

Connect to internet and update

```
nmtui                  # Connect to WiFi
sudo pacman -Syu       # Update system
```

Mount USB (if using Ventoy stick)

```
sudo mkdir -p /run/archusb
sudo mount /dev/sda1 /run/archusb   # Direct mount (not /dev/mapper)
```

Back up default bash files

```
mv ~/.bashrc ~/.bashrc.default
mv ~/.bash_profile ~/.bash_profile.default
```

Clone and stow dotfiles

From USB:

```
cp -r /run/archusb/dotfiles ~/dotfiles
find ~/dotfiles -name '._*' -delete   # Remove macOS resource forks
```

Or from GitHub:

```
git clone https://github.com/benarcher2691/dotfiles_arch_2026.git ~/dotfiles
```

Then stow:

```
cd ~/dotfiles
stow alacritty bash git hypr mako nvim vim waybar yazi
```

Install yay (AUR helper)

```
cd /tmp
git clone https://aur.archlinux.org/yay-bin.git
cd yay-bin
makepkg -si --noconfirm
cd ~ && rm -rf /tmp/yay-bin

# Disable interactive prompts
yay -Y --diffmenu=false --editmenu=false --cleanmenu=false --removemake=yes
↪  --provides=false --combinedupgrade=false --save
```

Install LocalSend (for file transfer)

```
yay -S --noconfirm localsend-bin
```

Transfer GPG keys (via LocalSend)

On macOS, export:

```
gpg --export-secret-keys --armor > ~/Desktop/gpg-secret.asc
gpg --export-ownertrust > ~/Desktop/gpg-trust.txt
```

Send via LocalSend to Arch, then import:

```
gpg --import ~/Downloads/gpg-secret.asc
gpg --import-ownertrust ~/Downloads/gpg-trust.txt
rm ~/Downloads/gpg-secret.asc ~/Downloads/gpg-trust.txt
```

Transfer SSH public key

From macOS:

```
ssh-copy-id ben@<arch-ip>
```

Clone second-brain

```
git clone git@github.com:benarcher2691/second-brain.git ~/second-brain
```

Copy password store (from USB)

```
cp -r /run/archusb/pass-store ~/.password-store
```

Install AUR packages

```
yay -S --noconfirm arch-audit brave-bin blueman lazydocker obsidian rkhunter
↪  spotify-launcher swww yazi
```

Install sdkman and nvm

```
curl -s "https://get.sdkman.io" | bash    # Java version manager
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash   #
↪  Node version manager
```

Restart shell, then install versions as needed: - `sdk install java` (list available: `sdk list java`) - `nvm install --lts` (list available: `nvm ls-remote`)

4. Start Hyprland

```
start-hyprland
```

5. Network Printer Setup

CUPS and mDNS discovery are pre-installed. To add a network printer:

Discover the printer

```
avahi-browse -art | grep -i printer
```

This shows printers on the network with their mDNS hostnames (e.g., `BRN008077D04DB0.local`).

Get the IP address

```
avahi-resolve -n HOSTNAME.local
```

Install the driver (Brother example)

```
# Search for your model
yay -Ss brother | grep -i YOUR_MODEL

# Install (example for HL-2170W)
yay -S brother-hl2170w
```

Find the PPD name

```
lpinfo -m | grep -i YOUR_MODEL
```

Add the printer

```
lpadmin -p PrinterName -E -v "socket://PRINTER_IP:9100" -m YOUR_MODEL.ppd
```

Example for Brother HL-2170W:

```
lpadmin -p HL2170W -E -v "socket://192.168.1.8:9100" -m HL2170W.ppd
```

Test printing

```
echo "test" | lp -d PrinterName
lp -d PrinterName -P 1 document.pdf   # Print page 1 only
```

Web interface

CUPS web interface available at `http://localhost:631` for managing printers and jobs.

6. Additional Software

Media player

```
sudo pacman -S vlc vlc-plugin-ffmpeg vlc-plugin-mpeg2
```

Torrent client

```
sudo pacman -S transmission-gtk
xdg-mime default transmission-gtk.desktop x-scheme-handler/magnet
```

VPN

```
yay -S mullvad-vpn-bin   # Use -bin to avoid heavy Rust compilation
sudo systemctl enable --now mullvad-daemon
mullvad account login <account-number>
mullvad connect
```

> **Warning:** Mullvad's "early boot network blocker" can block ALL network traffic (including LAN) when the daemon starts but you're not connected to VPN. If networking stops after a Mullvad update, either connect to VPN or run `mullvad lockdown-mode set off`.

Claude Code CLI

```
yay -S claude-code
```

Docker and DevPod

```
# Install Docker
sudo pacman -S docker
sudo systemctl enable --now docker
sudo usermod -aG docker $USER
newgrp docker   # Activate group in current shell (or log out/in)

# Install DevPod (containerized dev environments)
yay -S devpod-bin
echo 'alias devpod="devpod-cli"' >> ~/.bashrc
source ~/.bashrc

# Set up Docker provider and disable IDE (use SSH instead)
devpod provider add docker
devpod ide use none

# Usage: start a dev environment and SSH into it
devpod up <repo-or-path>
devpod ssh <workspace-name>
```

fzf integration with bash

If you installed fzf, enable bash integration for enhanced history search:

```
# Add to ~/.bashrc
[ -f /usr/share/fzf/key-bindings.bash ] && source /usr/share/fzf/key-bindings.bash
[ -f /usr/share/fzf/completion.bash ] && source /usr/share/fzf/completion.bash
```

Then reload: `source ~/.bashrc`

Key bindings: - **Ctrl+R**: Interactive command history search - **Ctrl+T**: File/directory finder - **Alt+C**: Quick directory navigation

Alacritty vi mode

Alacritty has a built-in vi mode for navigating and copying terminal output without a mouse.

Enter vi mode: `Ctrl+Shift+Space`

Navigation: - `h/j/k/l`: Move cursor left/down/up/right - `w/b`: Jump forward/backward by word - `0/$`: Jump to start/end of line - `g/G`: Jump to top/bottom of scrollback - `Ctrl+u/Ctrl+d`: Page up/down

Selection: - `v`: Start character selection - `V`: Start line selection - `Ctrl+v`: Start block selection

Search: - `/`: Search forward - `?`: Search backward - `n`/`N`: Next/previous match

Copy: - `y`: Yank (copy) selection to clipboard

Exit vi mode: `Escape` or `Enter`

ThinkPad F10/F11/F12 media controls

F10 is mapped to `XF86Bluetooth` by default, which toggles Bluetooth at the kernel level (rfkill). To use F10-F12 as media keys with `playerctl`:

1. Copy the hwdb file from dotfiles:

```
sudo cp ~/dotfiles/hypr/etc/udev/hwdb.d/99-thinkpad-f10.hwdb /etc/udev/hwdb.d/
sudo systemd-hwdb update && sudo udevadm trigger
```

2. The Hyprland config already has the bindings:
   - **F10**: Previous track
   - **F11**: Play/Pause
   - **F12**: Next track

To find keysyms for other keys, use `wev` or `sudo evtest`.

7. Security Hardening

Based on Lynis audit recommendations.

Enable firewall

```
sudo systemctl enable --now ufw
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw enable
```

Enable time sync

```
sudo systemctl enable --now systemd-timesyncd
```

SSH hardening

Create `/etc/ssh/sshd_config.d/50-hardening.conf`:

```
sudo tee /etc/ssh/sshd_config.d/50-hardening.conf << 'EOF'
AllowTcpForwarding no
ClientAliveCountMax 2
LogLevel VERBOSE
MaxAuthTries 3
MaxSessions 2
TCPKeepAlive no
AllowAgentForwarding no
EOF

sudo systemctl restart sshd
```

Kernel hardening

Create `/etc/sysctl.d/99-security.conf`:

```
sudo tee /etc/sysctl.d/99-security.conf << 'EOF'
# Disable ICMP redirects
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
net.ipv4.conf.all.send_redirects = 0

# Log martian packets
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1

# Restrict kernel pointer access
kernel.kptr_restrict = 2

# Disable TTY line discipline autoload
dev.tty.ldisc_autoload = 0

# Protect FIFOs and regular files in world-writable directories
fs.protected_fifos = 2
fs.protected_regular = 2

# Disable core dumps for setuid programs
fs.suid_dumpable = 0

# Disable Magic SysRq key
kernel.sysrq = 0

# Disable unprivileged BPF
kernel.unprivileged_bpf_disabled = 1

# Harden BPF JIT compiler
net.core.bpf_jit_harden = 2
EOF

sudo sysctl --system
```

Security tools

Already installed via yay in section 3.

Run vulnerability scan: `arch-audit` Run rootkit scan: `sudo rkhunter --check`

Lynis security audit

```
sudo pacman -S lynis
sudo lynis audit system
```

Review the output for warnings and suggestions. Copy the results to Claude Code for analysis and recommendations.

8. Scripts

User scripts in `~/dotfiles/scripts/.local/bin/` (stowed to `~/.local/bin/`).

md2pdf

Convert Markdown to PDF with code line wrapping.

```
md2pdf README.md                # Creates README.pdf
md2pdf input.md output.pdf      # Custom output name
```

Uses pandoc with LaTeX, A4 paper, 2.5cm margins, and automatic code block wrapping.

github-backup

Mirror all GitHub repos for offline backup.

```
github-backup benarcher2691                    # Backup to
↪  ~/backups/github/benarcher2691/
github-backup myorg /mnt/external/github       # Custom backup location
```

Creates bare git mirrors with all branches, LFS objects, and wikis. Run periodically to keep backups current.

github-restore

Restore a repo from backup for offline work.

```
github-restore dotfiles             # Clone to ./dotfiles from backup
github-restore dotfiles ~/projects  # Clone to custom location
github-restore                      # List available backups
```

Clones from local backup and sets remote to GitHub. When online, `git push` goes to GitHub. Idempotent - safe to run on existing repos.

9. Encrypted Cloud Backup (rclone)

Sync Google Drive to an external USB drive with on-the-fly encryption using rclone.

One-time setup

1. Configure Google Drive remote

```
rclone config
```

- `n` - New remote
- Name: `gdrive`
- Storage: `drive` (Google Drive)
- Leave client_id and client_secret blank (use rclone's)
- Scope: `1` (full access)
- Leave root_folder_id and service_account_file blank
- `n` - No advanced config
- `y` - Auto config (opens browser for OAuth)
- `n` - Not a shared drive
- `y` - Confirm

2. Configure encrypted remote

```
rclone config
```

- `n` - New remote
- Name: `wd-crypt`
- Storage: `crypt` (Encrypt/Decrypt a remote)
- Remote: `/mnt/wd/gdrive-backup` (path where encrypted files are stored)
- Filename encryption: `standard`
- Directory name encryption: `1` (encrypt)
- `y` - Enter password
- Enter a strong password (save this in your password store!)
- `y` - Confirm password
- `n` - No password2
- `y` - Confirm

3. Save the encryption password

```
pass insert backup/wd-crypt
```

Without this password and the rclone config, the backup cannot be decrypted.

Mount USB drive

```
sudo mount --mkdir /dev/sdb1 /mnt/wd
sudo chown $USER:$USER /mnt/wd
```

> **Note:** For ext4 filesystems, use `chown` after mounting. The `uid`/`gid` mount options only work with FAT/exFAT/NTFS.

Sync Google Drive to encrypted backup

```
rclone sync gdrive: wd-crypt: --progress
```

This compares source and destination, then transfers only new/changed files. Files are encrypted as they're written to the USB drive.

Options: - `--dry-run` - Preview changes without writing - `--progress` - Show transfer progress - `--exclude "*.tmp"` - Skip certain files

Restore from backup

```
# List files (decrypted view)
rclone ls wd-crypt:

# Restore single file
rclone copy wd-crypt:Documents/file.pdf ~/Downloads/

# Restore entire backup
rclone copy wd-crypt: ~/restored-gdrive/
```

Backup the rclone config

The config file at `~/.config/rclone/rclone.conf` contains your encryption password (obfuscated). Back it up:

```
pass insert -m backup/rclone-config < ~/.config/rclone/rclone.conf
```

Or copy to a secure location on another device.