NixOS Package Management

isomo[1]

2025-10-01

[1] github/jiahaoxiang2000

Overview

Package Management Strategies

In this NixOS configuration, we use three different approaches to manage software packages:

Package Management Strategies

In this NixOS configuration, we use three different approaches to manage software packages:

Type 1: Standard Packages

Packages from the official nixpkgs unstable channel, directly available through the Nix package manager.

Package Management Strategies

Type 2: Custom Configured Packages

Packages that require manual overrides or custom configuration to work properly in our environment.

Package Management Strategies

Type 2: Custom Configured Packages

Packages that require manual overrides or custom configuration to work properly in our environment.

Type 3: Self-Packaged Software

Custom packages we maintain ourselves, imported as flake inputs from GitHub repositories.

Package Types

Type 1: Standard Packages

Most packages come directly from nixpkgs unstable channel:

```
home.packages = with pkgs; [
  # Development environment
  pipx nodejs rustc rustup gcc uv gh
];
```

Type 1: Standard Packages

Most packages come directly from nixpkgs unstable channel:

```
home.packages = with pkgs; [
  # Development environment
  pipx nodejs rustc rustup gcc uv gh
];
```

These packages are:

- Available in the NixOS binary cache
- No custom configuration needed
- Updated through channel updates

Type 2: Custom Configured Packages

VSCode with Wayland Support

Custom override to use version 1.104.2 with proper Wayland integration

```
((vscode.overrideAttrs (oldAttrs: {
  version = "1.104.2";
  src = pkgs.fetchurl {
    name = "VSCode_1.104.2_linux-x64.tar.gz";
    url = "https://update.code.visualstudio.com/1.104.2/
linux-x64/stable";
    sha256 = "0zgsR0nk9zsOeEcKhrmAFbAhvKKFNsC8fXjCnxFcndE=";
  };
})).override {
  commandLineArgs = [
    "--enable-features=UseOzonePlatform"
    "--ozone-platform=wayland"
    "--enable-wayland-ime"
    "--wayland-text-input-version=3"
  ];
})
```

Created PR #447688[1] to fix upstream, but using manual config for immediate availability.

[1]https://github.com/NixOS/nixpkgs/pull/447688

Type 3: Self-Packaged Software

Custom packages maintained in separate repositories:

```
# In flake.nix inputs:
inputs = {
  blivedm_rs = {
    url = "github:jiahaoxiang2000/blivedm_rs";
    inputs.nixpkgs.follows = "nixpkgs";
  };
  danmu-tts = {
    url = "github:jiahaoxiang2000/danmu-tts";
    inputs.nixpkgs.follows = "nixpkgs";
  };
};

# In home.nix:
home.packages = with pkgs; [
  inputs.blivedm_rs.packages.${pkgs.system}.default
  inputs.danmu-tts.packages.${pkgs.system}.default
];
```

These packages:

- Are not in the official nixpkgs channel
- Built from source (not cached)
- Updated by changing flake input versions

Package Categories

Live Streaming

```
home.packages = with pkgs; [
  # Custom packages for Bilibili live streaming
  inputs.blivedm_rs.packages.${pkgs.system}.default
  inputs.danmu-tts.packages.${pkgs.system}.default
];
programs.obs-studio = {
  enable = true;
  plugins = with pkgs.obs-studio-plugins; [
    wlrobs  # Wayland screen capture
  ];
};
```

IDE & Development Environment

```
home.packages = with pkgs; [
  neovim         # Highly customizable terminal editor
  vscode         # Custom configured for Wayland
  rustc rustup   # Rust development
  nodejs         # JavaScript runtime
  gcc            # C/C++ compiler
  uv             # Python package manager
  pipx           # Python CLI tools
  gh             # GitHub CLI
  gnumake        # Build automation tool
];
```

Desktop Environment: River WM

Wayland-native desktop on River window manager:

```
# Terminal & Launcher
foot        # Wayland-native terminal
wmenu       # Application launcher
# Input & Clipboard
xremap      # Keyboard remapping
wl-clipboard # Wayland clipboard utilities
# System Control
pavucontrol # Audio settings
wlr-randr   # Display configuration
```

```
# Status & Notifications
i3bar-river      # Status bar for River
i3status-rust    # Rust-based status generator
dunst            # Notification daemon

# Screenshots & Security
hyprshot         # Screenshot tool
hyprlock         # Screen lock
```

Writing & Typography

```
home.packages = with pkgs; [
  # Fonts
  source-han-serif  # Chinese serif font
  source-han-sans   # Chinese sans-serif font
  source-han-mono   # Chinese monospace font
  font-awesome      # Icon font
  # Typesetting
  texliveFull       # Complete TeX Live distribution
  typst             # Modern typesetting system
  tinymist          # Typst language server
];
```

Multimedia & Utilities

```
home.packages = with pkgs; [
  kdePackages.kdenlive      # Professional video editor
  mpv                       # Versatile media player
  netease-cloud-music-gtk   # Netease Cloud Music
  qqmusic                   # QQ Music
  kdePackages.okular        # PDF reader
  killall                   # Process management
  p7zip                     # 7z compression
  fastfetch                 # System information
  jq                        # JSON processor
];
```

System Configuration

System-Level Setup

Key system configurations in configuration.nix:

```
# Performance kernel
boot.kernelPackages = pkgs.linuxPackages_zen;
# NVIDIA graphics
services.xserver.videoDrivers = [ "nvidia" ];
hardware.nvidia.package =
  config.boot.kernelPackages.nvidiaPackages.stable;
# Chinese input method
i18n.inputMethod = {
  type = "fcitx5";
  enable = true;
  fcitx5.addons = [
    fcitx5-chinese-addons
    fcitx5-pinyin-moegirl
    fcitx5-pinyin-zhwiki
  ];
};
```

- Zen kernel: Optimized for desktop performance
- NVIDIA: Proprietary drivers for GPU acceleration
- Fcitx5: Chinese input with cloud pinyin

Audio & Display

```
# Modern audio stack
services.pipewire = {
  enable = true;
  pulse.enable = true;  # PulseAudio compatibility
};
# Wayland portals
xdg.portal = {
  enable = true;
  wlr.enable = true;  # wlroots portal for screen sharing
};
# Display manager
```

```
services.displayManager.sddm = {
  enable = true;
  wayland.enable = true;
};
# Window manager
programs.river.enable = true;
```

Full Wayland stack with PipeWire audio and proper portal support for screen sharing.

Package Management Optimization

```
# Use Tsinghua mirror for faster downloads
nix.settings.substituters = lib.mkForce [
  "https://mirrors.tuna.tsinghua.edu.cn/nix-channels/store"
];
# Automatic garbage collection
nix.gc = {
  automatic = true;
  dates = "weekly";
  options = "--delete-older-than 30d";
};
```

```nix
# Store optimization
nix.optimise = {
  automatic = true;
  dates = [ "weekly" ];
};
# Limit boot entries
boot.loader.systemd-boot.configurationLimit = 10;
```

Summary

Three-Tier Package Strategy

Our NixOS configuration demonstrates a flexible package management approach: using **standard packages** for most software (Type 1) with fast updates from binary cache, **custom overrides** for specific requirements (Type 2) providing immediate fixes for critical issues, and **self-maintained packages** for specialized tools (Type 3) giving full control over custom software.

Total packages: 50+ across 6 major categories
System: River WM + NixOS unstable + Zen kernel