Downgrading or Upgrading Packages

When working with Flakes, you may encounter situations where you need to downgrade or upgrade certain packages to address bugs or compatibility issues. In Flakes, package versions and hash values are directly tied to the git commit of their flake input. To modify the package version, you need to lock the git commit of the flake input.

Here's an example of how you can add multiple nixpkgs inputs, each using a different git commit or branch:

nix

```nix
{
  description = "NixOS configuration of Ryan Yin";

  inputs = {
    # Default to the nixos-unstable branch
    nixpkgs.url = "github:nixos/nixpkgs/nixos-unstable";

    # Latest stable branch of nixpkgs, used for version rollback
    # The current latest version is 24.11
    nixpkgs-stable.url = "github:nixos/nixpkgs/nixos-24.11";

    # You can also use a specific git commit hash to lock the version
    nixpkgs-fd40cef8d.url = "github:nixos/nixpkgs/fd40cef8d797670e203a27a91e4b8e
  };

  outputs = inputs@{
    self,
    nixpkgs,
    nixpkgs-stable,
    nixpkgs-fd40cef8d,
    ...
  }: {
    nixosConfigurations = {
      my-nixos = nixpkgs.lib.nixosSystem rec {
        system = "x86_64-linux";

        # The `specialArgs` parameter passes the
        # non-default nixpkgs instances to other nix modules
        specialArgs = {
          # To use packages from nixpkgs-stable,
          # we configure some parameters for it first
```

```
            pkgs-stable = import nixpkgs-stable {
              # Refer to the `system` parameter from
              # the outer scope recursively
              inherit system;
              # To use Chrome, we need to allow the
              # installation of non-free software.
              config.allowUnfree = true;
            };
            pkgs-fd40cef8d = import nixpkgs-fd40cef8d {
              inherit system;
              config.allowUnfree = true;
            };
          };

          modules = [
            ./hosts/my-nixos

            # Omit other configurations...
          ];
        };
      };
    };
  }
```

In the above example, we have defined multiple nixpkgs inputs: `nixpkgs`, `nixpkgs-stable`, and `nixpkgs-fd40cef8d`. Each input corresponds to a different git commit or branch.

Next, you can refer to the packages from `pkgs-stable` or `pkgs-fd40cef8d` within your submodule. Here's an example of a Home Manager submodule:

nix

```
{
  pkgs,
  config,
  # Nix will search for and inject this parameter
  # from `specialArgs` in `flake.nix`
  pkgs-stable,
  # pkgs-fd40cef8d,
  ...
}:

```

```
{
  # Use packages from `pkgs-stable` instead of `pkgs`
  home.packages = with pkgs-stable; [
    firefox-wayland

    # Chrome Wayland support was broken on the nixos-unstable branch,
    # so we fallback to the stable branch for now.
    # Reference: https://github.com/swaywm/sway/issues/7562
    google-chrome
  ];

  programs.vscode = {
    enable = true;
    # Refer to vscode from `pkgs-stable` instead of `pkgs`
    package = pkgs-stable.vscode;
  };
}
```

Pinning a package version with an overlay

The above approach is perfect for application packages, but sometimes you need to replace libraries used by those packages. This is where Overlays shine! Overlays can edit or replace any attribute of a package, but for now we'll just pin a package to a different nixpkgs version. The main disadvantage of editing a dependency with an overlay is that your Nix installation will recompile all installed packages that depend on it, but your situation may require it for specific bug fixes.

nix

```
# overlays/mesa.nix
{ config, pkgs, lib, pkgs-fd40cef8d, ... }:
{
  nixpkgs.overlays = [
    # Overlay: Use `self` and `super` to express
    # the inheritance relationship
    (self: super: {
      mesa = pkgs-fd40cef8d.mesa;
    })
  ];
}
```

Applying the new configuration

By adjusting the configuration as shown above, you can deploy it using `sudo nixos-rebuild switch`. This will downgrade your Firefox/Chrome/VSCode versions to the ones corresponding to `nixpkgs-stable` or `nixpkgs-fd40cef8d`.

> According to 1000 instances of nixpkgs, it's not a good practice to use `import` in submodules or subflakes to customize `nixpkgs`. Each `import` creates a new instance of nixpkgs, which increases build time and memory usage as the configuration grows. To avoid this problem, we create all nixpkgs instances in `flake.nix`.
