Stylix

Stylix is a NixOS module which applies the same colour scheme, font and wallpaper to a range of applications and desktop environments.

What's this?

base16.nix allows you to import colours from base16 into Nix code. Stylix takes this a step further:

- Automatically colours and changes the font of apps
- Sets your wallpaper
- Exports the colour scheme to be used manually for anything we missed
- Can also generate themes based on an image

For those not familiar with NixOS and Home Manager:

- NixOS is a Linux distribution
- Home Manager is a program which runs anywhere
- Both use the Nix language and package manager
- Both let you install programs and change settings via code

Stylix supports either NixOS + Home Manager, or Home Manager on its own. Certain features are only available with NixOS.

Resources

Please refer to the Stylix book for instructions and a list of supported apps.

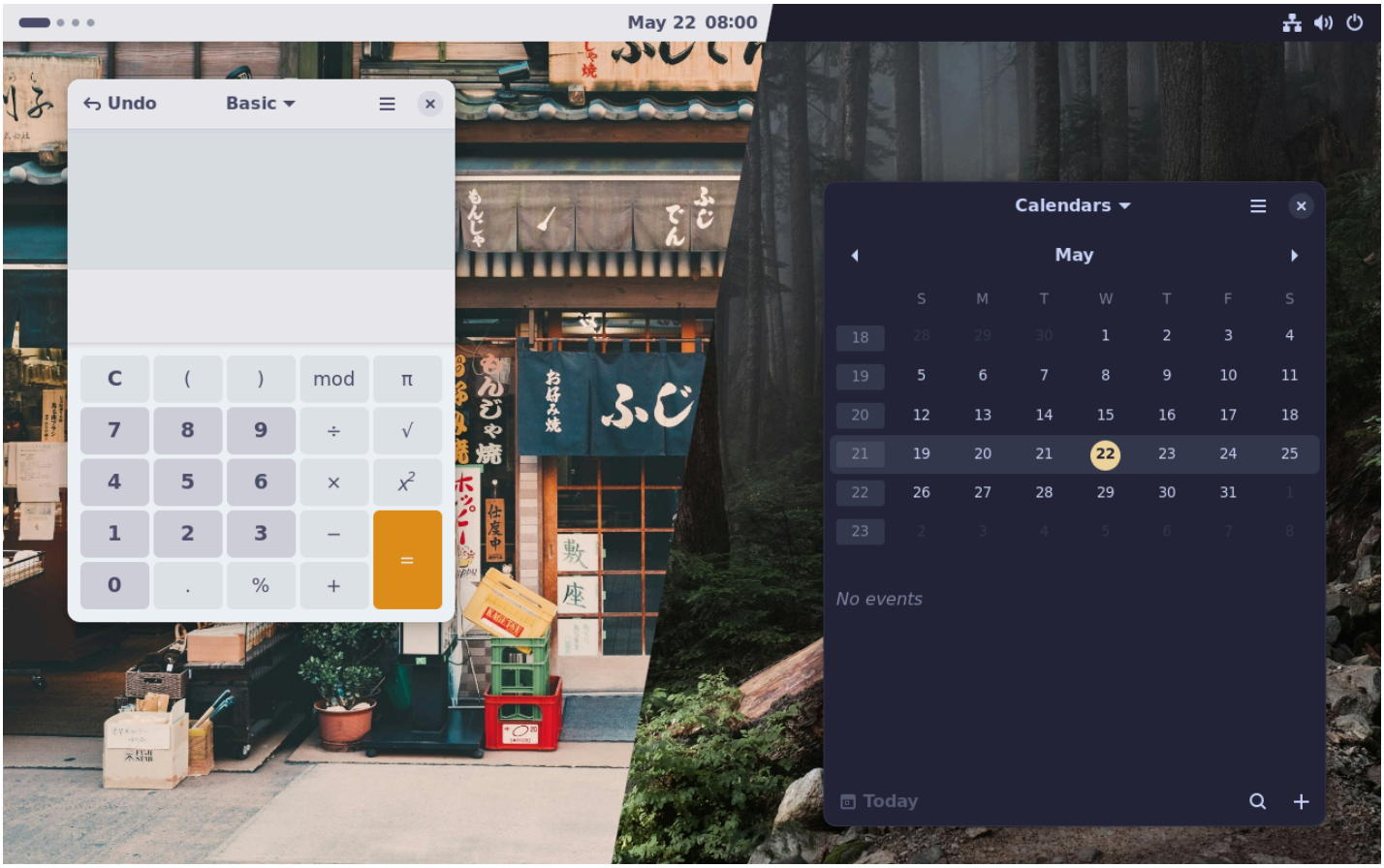
For a visual guide, watch the *Ricing Linux Has Never Been Easier | NixOS + Stylix* YouTube video by Vimjoyer.

> **Note**
>
> It's now necessary to include `stylix.enable = true` in your configuration for any other settings to take effect. This is not mentioned in the video linked above.

If you have any questions, you are welcome to join our Matrix room, or ask on GitHub Discussions.

Example configurations

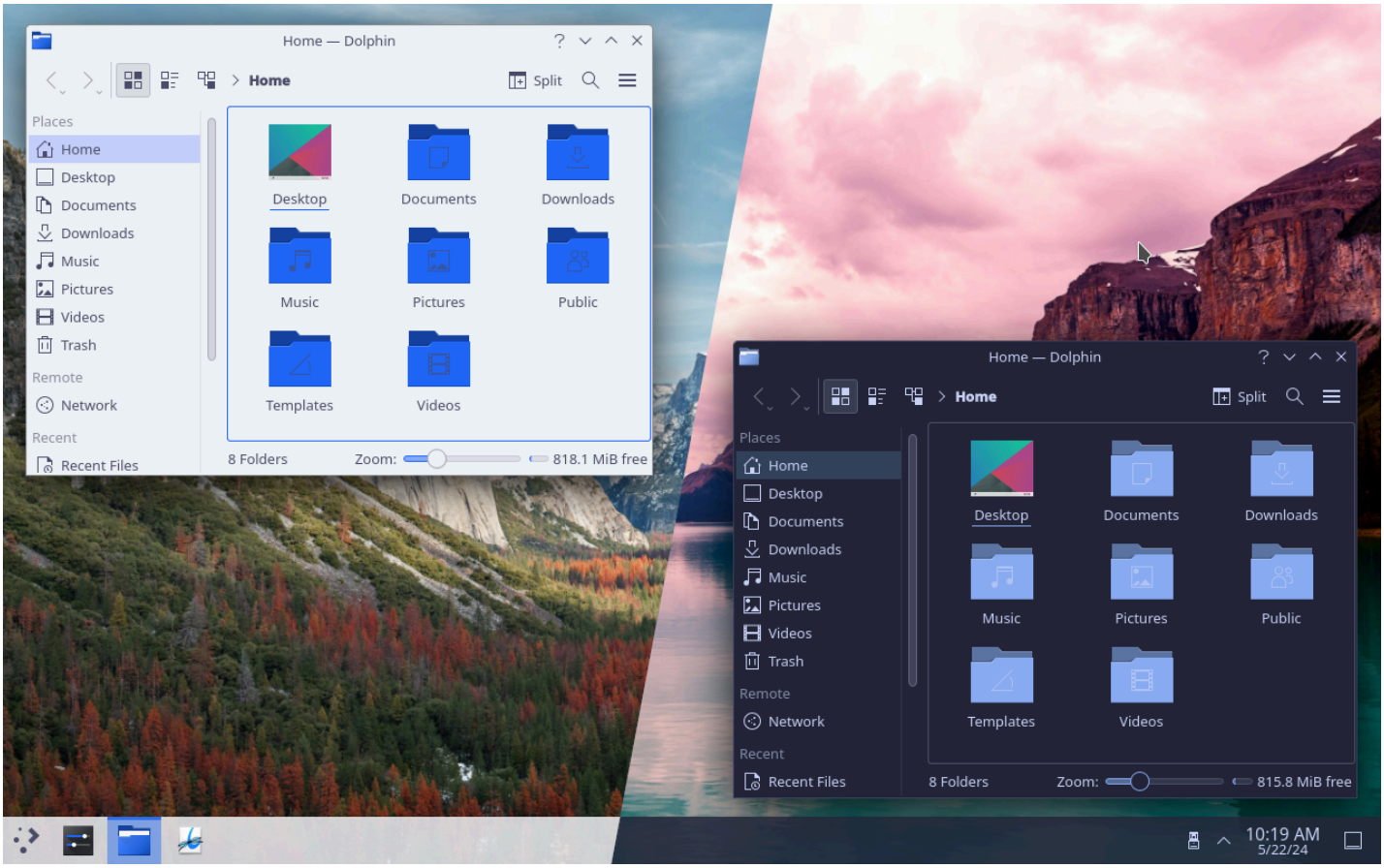
GNOME 46



Photos by Clay Banks and Derrick Cooper.

Try a live demo of this theme by running `nix run github:danth/stylix#testbed-gnome-light` or `nix run github:danth/stylix#testbed-gnome-dark`.

KDE Plasma 5



Photos by Aniket Deole and Tom Gainor.

KDE theming is still a work in progress - so some manual steps may be needed to apply the settings completely.

Installation

NixOS

You can install Stylix into your NixOS configuration using Flakes. This will provide theming for system level programs such as bootloaders, splash screens, and display managers.

```
{
  inputs = {
    nixpkgs.url = "github:NixOS/nixpkgs/nixos-unstable";
    stylix.url = "github:danth/stylix";
  };

  outputs = { nixpkgs, stylix, ... }: {
    nixosConfigurations."«hostname»" = nixpkgs.lib.nixosSystem {
      system = "x86_64-linux";
      modules = [ stylix.nixosModules.stylix ./configuration.nix ];
    };
  };
}
```

Minimal `flake.nix` for a NixOS configuration.

Many applications cannot be configured system wide, so Stylix will also need Home Manager to be able to change their settings within your home directory.

Installing Home Manager as a NixOS module is highly recommended if you don't use it already. This will combine it with your existing configuration, so you don't need to run any extra commands when you rebuild, and the theme you set in NixOS will automatically be used for Home Manager too.

When Stylix is installed to a NixOS configuration, it will automatically set up its Home Manager modules if it detects that Home Manager is available. You can theoretically use it without installing Home Manager, however most features will be unavailable.

nix-darwin

You can install Stylix into your nix-darwin configuration in a similar fashion to NixOS via Flakes.

```
{
  inputs = {
    darwin = {
      url = "github:LnL7/nix-darwin";
      inputs.nixpkgs.follows = "nixpkgs";
    };
    nixpkgs.url = "github:NixOS/nixpkgs/nixos-unstable";
    stylix.url = "github:danth/stylix";
  };

  outputs = { darwin, nixpkgs, stylix, ... }: {
    darwinConfigurations."«hostname»" = darwin.lib.darwinSystem {
      system = "aarch64-darwin";
      modules = [ stylix.darwinModules.stylix ./configuration.nix ];
    };
  };
}
```

Minimal `flake.nix` for a nix-darwin configuration.

While this won't have an effect on the looks of MacOS, since we don't have the controls to theme it like we do NixOS, it will automatically set up the Home Manager modules for you.

Home Manager

If you would prefer to use the standalone version of Home Manager, you can install Stylix directly into your Home Manager configuration instead. This could be useful if you are on another operating system, or a machine which is managed by someone else.

```
{
  inputs = {
    nixpkgs.url = "github:NixOS/nixpkgs/nixos-unstable";
    home-manager.url = "github:nix-community/home-manager";
    stylix.url = "github:danth/stylix";
  };

  outputs = { nixpkgs, home-manager, stylix, ... }: {
    homeConfigurations."«username»" = home-manager.lib.homeManagerConfiguration {
      pkgs = nixpkgs.legacyPackages.x86_64-linux;
      modules = [ stylix.homeManagerModules.stylix ./home.nix ];
    };
  };
}
```

Minimal `flake.nix` for a Home Manager configuration.

If you choose to use both NixOS and Home Manager but configure them separately, you will need to copy the settings described below into both of your configurations, as keeping them separate means that they cannot follow each other automatically.

Without flakes

If you haven't enabled flakes yet or don't want to use this feature, `default.nix` re-exports all the flake outputs, without requiring flakes to be enabled. This means that once you have a copy of this repo, using either a local checkout, niv, or any other method, you can import it to get the NixOS module as the `nixosModules.stylix` attribute and the Home Manager module as the `homeManagerModules.stylix` attribute.

```
let
  stylix = pkgs.fetchFromGitHub {
      owner = "danth";
      repo = "stylix";
      rev = "...";
      sha256 = "...";
  };
in {
    imports = [ (import stylix).homeManagerModules.stylix ];

    stylix = {
      enable = true;
      image = ./wallpaper.jpg;
    };
}
```

Example usage of the Home Manager module without flakes.

Configuration

Enable

To enable the Stylix module, declare:

```
{
  stylix.enable = true;
}
```

Note

The global enable option was recently added, so you may come across old examples which don't include it. No other settings will take effect unless `stylix.enable` is set to `true`.

Wallpaper

To start theming, you need to set a wallpaper image.

```
{
  stylix.image = ./wallpaper.png;
}
```

The option accepts derivations as well as paths, so you can fetch an image directly from the internet:

```
{
  stylix.image = pkgs.fetchurl {
    url = "https://www.pixelstalk.net/wp-content/uploads/2016/05/Epic-Anime-Awesome-
Wallpapers.jpg";
    sha256 = "enQo3wqhgf0FEPHj2coOCvo7DuZv+x5rL/WIo4qPI50=";
  };
}
```

Color scheme

Generated schemes

If you only set a wallpaper, Stylix will use a genetic algorithm to create a color scheme. The quality of these schemes can vary, but more colorful images tend to have better results.

You can force a light or dark scheme using the polarity option:

```
{
  stylix.polarity = "dark";
}
```

The current scheme can be previewed in a web browser at either `/etc/stylix/palette.html` for NixOS, or `~/.config/stylix/palette.html` for Home Manager.

Handmade schemes

If you prefer a handmade color scheme, you can choose anything from the Tinted Theming repository:

```
{
  stylix.base16Scheme = "${pkgs.base16-schemes}/share/themes/gruvbox-dark-hard.yaml";
}
```

This option also accepts other files and formats supported by `mkSchemeAttrs`.

Overriding

For convenience, it is possible to override parts of `stylix.base16Scheme` using `stylix.override`. Anything that base16.nix accepts as override is valid.

When using both the Home Manager and NixOS modules, both the system overrides and the user-provided one are used in the user configuration if `stylix.base16Scheme` is not changed in the user config. If that is the case, only the user override is used.

Extending

When passing colors to unsupported targets or creating custom modules, it is possible to access values from the configured color scheme through `config.lib.stylix.colors`. An overview of the available values is shown below.

```
config.lib.stylix.colors = {
  base08 = "ff0000";
  base08-hex-r = "ff";
  base08-dec-r = "0.996094";
  # ...
  red = "ff0000";
  # ...
  withHashtag = {
    base08 = "#ff0000";
    # ...
  };
};
```

This attrset is generated by `mkSchemeAttrs` from `base16.nix`. Refer to the documentation for more info.

For more complex configurations you may find it simpler to use mustache templates to generate output files. See base16.nix documentation for usage examples.

Fonts

The default combination of fonts is:

```
{
  stylix.fonts = {
    serif = {
      package = pkgs.dejavu_fonts;
      name = "DejaVu Serif";
    };

    sansSerif = {
      package = pkgs.dejavu_fonts;
      name = "DejaVu Sans";
    };

    monospace = {
      package = pkgs.dejavu_fonts;
      name = "DejaVu Sans Mono";
    };

    emoji = {
      package = pkgs.noto-fonts-emoji;
      name = "Noto Color Emoji";
    };
  };
}
```

These can be changed as you like.

To make things look more uniform, you could replace the serif font with the sans-serif font:

```
{
  stylix.fonts.serif = config.stylix.fonts.sansSerif;
}
```

Or even choose monospace for everything:

```
{
  stylix.fonts = {
    serif = config.stylix.fonts.monospace;
    sansSerif = config.stylix.fonts.monospace;
    emoji = config.stylix.fonts.monospace;
  };
}
```

Home Manager inheritance

By default, if Home Manager is used as part of NixOS, then Stylix will be automatically installed for all users, and the NixOS theme will become their default settings.

This is convenient for single-user systems, since you can configure everything once at the system level and it will automatically carry over. For multi-user systems, you can override the settings within Home Manager to select a different theme for each user.

You may prefer to disable inheritance entirely, and set up the Home Manager version of Stylix yourself if required. Refer to the options `stylix.homeManagerIntegration.autoImport` and `stylix.homeManagerIntegration.followSystem` to customize this.

Note

There is a special case involving the `stylix.base16Scheme` option:

If the wallpaper in a Home Manager configuration is changed, then Home Manager will stop inheriting the color scheme from NixOS. This allows Home Manager configurations to use the automatic palette generator without being overridden.

Similarly, `stylix.override` is not inherited if the color scheme is different.

Turning targets on and off

In Stylix terms, a target is anything which can have colors, fonts or a wallpaper applied to it. Each module in this repository should correspond to a target of the same name.

Each target has an option like `stylix.targets.«target».enable` to turn its styling on or off. Normally, it's turned on automatically when the target is installed. You can set `stylix.autoEnable = false` to opt out of this behaviour, in which case you'll need to manually enable each target you want to be styled.

Targets are different between Home Manager and NixOS, and sometimes available in both cases. If both are available, it is always correct to enable both. The reference pages have a list of targets for NixOS and Home Manager respectively.

Tips and tricks

Adjusting the brightness and contrast of a background image

If you want to use a background image for your desktop but find it too bright or distracting, you can use the `imagemagick` package to dim the image, or adjust its brightness and contrast to suit your preference.

Here's an example Nix expression that takes an input image, applies a brightness/contrast adjustment to it, and saves the result as a new image file:

```
{ pkgs, ... }:

let
  inputImage = ./path/to/image.jpg;
  brightness = -30;
  contrast = 0;
  fillColor = "black"
in
{
  stylix.image = pkgs.runCommand "dimmed-background.png" { } ''
    ${pkgs.imagemagick}/bin/convert "${inputImage}" -brightness-contrast
${brightness},${contrast} -fill ${fillColor} $out
  '';
}
```

Dynamic wallpaper generation based on selected theme

With imagemagick, you can also dynamically generate wallpapers based on the selected theme. Similarly, you can use a template image and repaint it for the current theme.

```
{ pkgs, ... }:

let
  theme = "${pkgs.base16-schemes}/share/themes/catppuccin-latte.yaml";
  wallpaper = pkgs.runCommand "image.png" {} ''
        COLOR=$(${pkgs.yq}/bin/yq -r .base00 ${theme})
        COLOR="#"$COLOR
        ${pkgs.imagemagick}/bin/magick convert -size 1920x1080 xc:$COLOR $out
  '';
in {
  stylix = {
    image = wallpaper;
    base16Scheme = theme;
  };
}
```

Which is neatly implemented as a single function in `lib.stylix.pixel`:

```
{ pkgs, config, ... }:

{
  stylix = {
    image = config.lib.stylix.pixel "base0A";
    base16Scheme = "${pkgs.base16-schemes}/share/themes/catppuccin-latte.yaml";
  };
}
```

NixOS options

The following options can only be set in a NixOS configuration.

stylix.enable

Whether to enable Stylix.

When this is `false`, all theming is disabled and all other options are ignored.

Type: boolean

Default: `false`

Example: `true`

stylix.autoEnable

Whether to enable targets by default.

When this is `false`, all targets are disabled unless explicitly enabled.

When this is `true`, most targets are enabled by default. A small number remain off by default, because they require further manual setup, or they are only applicable in specific circumstances which cannot be detected automatically.

Type: boolean

Default: `true`

Example: `false`

stylix.base16Scheme

A scheme following the base16 standard.

This can be a path to a file, a string of YAML, or an attribute set.

Type: path or strings concatenated with “\n” or (attribute set)

Default: The colors used in the theming.

Those are automatically selected from the background image by default, but could be overridden manually.

stylix.cursor.package

Package providing the cursor theme.

Type: package

Default: `<derivation vanilla-dmz-0.4.5>`

stylix.cursor.name

The cursor name within the package.

Type: string

Default: `"Vanilla-DMZ"`

stylix.cursor.size

The cursor size.

Type: signed integer

Default: `32`

stylix.fonts.packages

A list of all the font packages that will be installed.

Type: list of package *(read only)*

stylix.fonts.emoji

Emoji font.

Type: submodule

Default:

```
{
  name = "Noto Color Emoji";
  package = <derivation noto-fonts-color-emoji-2.042>;
}
```

stylix.fonts.emoji.package

Package providing the font.

Type: package

stylix.fonts.emoji.name

Name of the font within the package.

Type: string

stylix.fonts.monospace

Monospace font.

Type: submodule

Default:

```
{
  name = "DejaVu Sans Mono";
  package = <derivation dejavu-fonts-2.37>;
}
```

stylix.fonts.monospace.package

Package providing the font.

Type: package

stylix.fonts.monospace.name

Name of the font within the package.

Type: string

stylix.fonts.sansSerif

Sans-serif font.

Type: submodule

Default:

```
{
  name = "DejaVu Sans";
  package = <derivation dejavu-fonts-2.37>;
}
```

stylix.fonts.sansSerif.package

Package providing the font.

Type: package

stylix.fonts.sansSerif.name

Name of the font within the package.

Type: string

stylix.fonts.serif

Serif font.

Type: submodule

Default:

```
{
  name = "DejaVu Serif";
  package = <derivation dejavu-fonts-2.37>;
}
```

stylix.fonts.serif.package

Package providing the font.

Type: package

stylix.fonts.serif.name

Name of the font within the package.

Type: string

stylix.fonts.sizes.applications

The font size used by applications.

Type: unsigned integer, meaning >=0

Default: 12

stylix.fonts.sizes.desktop

The font size used in window titles/bars/widgets elements of the desktop.

Type: unsigned integer, meaning >=0

Default: `10`

stylix.fonts.sizes.popups

The font size for notifications/popups and in general overlay elements of the desktop.

Type: unsigned integer, meaning >=0

Default: `10`

stylix.fonts.sizes.terminal

The font size for terminals/text editors.

Type: unsigned integer, meaning >=0

Default: `12`

stylix.homeManagerIntegration.autoImport

Whether to import Stylix automatically for every Home Manager user.

This only works if you are using `home-manager.users.«name»` within your NixOS configuration, rather than running Home Manager independently.

Type: boolean

Default: `true`

Example: `false`

stylix.homeManagerIntegration.followSystem

When this option is `true`, Home Manager configurations will follow the NixOS configuration by default, rather than using the standard default settings.

This only applies to Home Manager configurations managed by `stylix.homeManagerIntegration.autoImport`.

Type: boolean

Default: `true`

Example: `false`

stylix.image

Wallpaper image.

This is set as the background of your desktop environment, if possible, and used to generate a colour scheme if you don't set one manually.

Type: path or package convertible to it

stylix.imageScalingMode

Wallpaper scaling mode;

This is the scaling mode your wallpaper image will use assuming it doesnt fix your monitor perfectly

Type: one of "stretch", "fill", "fit", "center", "tile"

Default: `"fill"`

stylix.opacity.applications

The opacity of the windows of applications, the amount of applications supported is currently limited

Type: floating point number

Default: `1.0`

stylix.opacity.desktop

The opacity of the windows of bars/widgets, the amount of applications supported is currently limited

Type: floating point number

Default: `1.0`

stylix.opacity.popups

The opacity of the windows of notifications/popups, the amount of applications supported is currently limited

Type: floating point number

Default: `1.0`

stylix.opacity.terminal

The opacity of the windows of terminals, this works across all terminals supported by stylix

Type: floating point number

Default: `1.0`

stylix.override

An override that will be applied to stylix.base16Scheme when generating config.lib.stylix.colors.

Takes anything that a scheme generated by base16nix can take as argument to override.

Type: attribute set

Default: `{ }`

stylix.polarity

Use this option to force a light or dark theme.

By default we will select whichever is ranked better by the genetic algorithm. This aims to get good contrast between the foreground and background, as well as some variety in the highlight colours.

Type: one of “either”, “light”, “dark”

Default: `"either"`

stylix.targets.chromium.enable

Whether to enable theming for Chromium, Google Chrome and Brave.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.console.enable

Whether to enable theming for the Linux kernel console.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.feh.enable

Whether to enable theming for the desktop background using Feh.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.fish.enable

Whether to enable theming for Fish.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.gnome.enable

Whether to enable theming for GNOME and GDM.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.grub.enable

Whether to enable theming for GRUB.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.grub.useImage

Whether to use your wallpaper image as the GRUB background.

Type: boolean

Default: `false`

stylix.targets.gtk.enable

Whether to enable theming for all GTK3, GTK4 and Libadwaita apps.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.kmscon.enable

Whether to enable theming for the kmscon virtual console.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.lightdm.enable

Whether to enable theming for LightDM.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.nixos-icons.enable

Whether to enable theming for the NixOS logo.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.nixvim.enable

Whether to enable theming for nixvim.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.nixvim.transparentBackground.main

Whether to enable background transparency for the main NeoVim window.

Type: boolean

Default: `false`

Example: `true`

stylix.targets.nixvim.transparentBackground.signColumn

Whether to enable background transparency for the NeoVim sign column.

Type: boolean

Default: `false`

Example: `true`

stylix.targets.plymouth.enable

Whether to enable theming for the Plymouth boot screen.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.plymouth.logo

Logo to be used on the boot screen.

Type: path or package

Default: NixOS logo

stylix.targets.plymouth.logoAnimated

Whether to apply a spinning animation to the logo.

Disabling this allows the use of logos which don't have rotational symmetry.

Type: boolean

Default: `true`

Home Manager options

The following options can only be set in a Home Manager configuration.

If you combined Home Manager with your NixOS configuration, write these options within a Home Manager section, either for all users:

```
home-manager.sharedModules = [{
  stylix.targets.xyz.enable = false;
}];
```

Or for a specific user:

```
home-manager.users.«name» = {
  stylix.targets.xyz.enable = false;
};
```

Read more about per-user themes.

stylix.enable

Whether to enable Stylix.

When this is `false`, all theming is disabled and all other options are ignored.

Type: boolean

Default: `false`

Example: `true`

stylix.autoEnable

Whether to enable targets by default.

When this is `false`, all targets are disabled unless explicitly enabled.

When this is `true`, most targets are enabled by default. A small number remain off by default, because they require further manual setup, or they are only applicable in specific

circumstances which cannot be detected automatically.

Type: boolean

Default: `true`

Example: `false`

stylix.base16Scheme

A scheme following the base16 standard.

This can be a path to a file, a string of YAML, or an attribute set.

Type: path or strings concatenated with “\n” or (attribute set)

Default: The colors used in the theming.

Those are automatically selected from the background image by default, but could be overridden manually.

stylix.cursor.package

Package providing the cursor theme.

Type: package

Default: `<derivation vanilla-dmz-0.4.5>`

stylix.cursor.name

The cursor name within the package.

Type: string

Default: `"Vanilla-DMZ"`

stylix.cursor.size

The cursor size.

Type: signed integer

Default: 32

stylix.fonts.packages

A list of all the font packages that will be installed.

Type: list of package *(read only)*

stylix.fonts.emoji

Emoji font.

Type: submodule

Default:

```
{
  name = "Noto Color Emoji";
  package = <derivation noto-fonts-color-emoji-2.042>;
}
```

stylix.fonts.emoji.package

Package providing the font.

Type: package

stylix.fonts.emoji.name

Name of the font within the package.

Type: string

stylix.fonts.monospace

Monospace font.

Type: submodule

Default:

```
{
  name = "DejaVu Sans Mono";
  package = <derivation dejavu-fonts-2.37>;
}
```

stylix.fonts.monospace.package

Package providing the font.

Type: package

stylix.fonts.monospace.name

Name of the font within the package.

Type: string

stylix.fonts.sansSerif

Sans-serif font.

Type: submodule

Default:

```
{
  name = "DejaVu Sans";
  package = <derivation dejavu-fonts-2.37>;
}
```

stylix.fonts.sansSerif.package

Package providing the font.

Type: package

stylix.fonts.sansSerif.name

Name of the font within the package.

Type: string

stylix.fonts.serif

Serif font.

Type: submodule

Default:

```
{
  name = "DejaVu Serif";
  package = <derivation dejavu-fonts-2.37>;
}
```

stylix.fonts.serif.package

Package providing the font.

Type: package

stylix.fonts.serif.name

Name of the font within the package.

Type: string

stylix.fonts.sizes.applications

The font size used by applications.

Type: unsigned integer, meaning >=0

Default: `12`

stylix.fonts.sizes.desktop

The font size used in window titles/bars/widgets elements of the desktop.

Type: unsigned integer, meaning >=0

Default: `10`

stylix.fonts.sizes.popups

The font size for notifications/popups and in general overlay elements of the desktop.

Type: unsigned integer, meaning >=0

Default: `10`

stylix.fonts.sizes.terminal

The font size for terminals/text editors.

Type: unsigned integer, meaning >=0

Default: `12`

stylix.image

Wallpaper image.

This is set as the background of your desktop environment, if possible, and used to generate a colour scheme if you don't set one manually.

Type: path or package convertible to it

stylix.imageScalingMode

Wallpaper scaling mode;

This is the scaling mode your wallpaper image will use assuming it doesnt fix your monitor perfectly

Type: one of "stretch", "fill", "fit", "center", "tile"

Default: `"fill"`

stylix.opacity.applications

The opacity of the windows of applications, the amount of applications supported is currently limited

Type: floating point number

Default: `1.0`

stylix.opacity.desktop

The opacity of the windows of bars/widgets, the amount of applications supported is currently limited

Type: floating point number

Default: `1.0`

stylix.opacity.popups

The opacity of the windows of notifications/popups, the amount of applications supported is currently limited

Type: floating point number

Default: `1.0`

stylix.opacity.terminal

The opacity of the windows of terminals, this works across all terminals supported by stylix

Type: floating point number

Default: `1.0`

stylix.override

An override that will be applied to stylix.base16Scheme when generating config.lib.stylix.colors.

Takes anything that a scheme generated by base16nix can take as argument to override.

Type: attribute set

Default: `{ }`

stylix.polarity

Use this option to force a light or dark theme.

By default we will select whichever is ranked better by the genetic algorithm. This aims to get good contrast between the foreground and background, as well as some variety in the highlight colours.

Type: one of “either”, “light”, “dark”

Default: `"either"`

stylix.targets.alacritty.enable

Whether to enable theming for Alacritty.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.avizo.enable

Whether to enable theming for Avizo.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.bat.enable

Whether to enable theming for Bat.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.bemenu.enable

Whether to enable theming for bemenu.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.bemenu.alternate

Whether to use alternating colours.

Type: boolean

Default: `false`

stylix.targets.bemenu.fontSize

Font size used for bemenu.

Type: null or signed integer

Default: `10`

stylix.targets.bspwm.enable

Whether to enable theming for bspwm.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.btop.enable

Whether to enable theming for btop.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.dunst.enable

Whether to enable theming for Dunst.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.emacs.enable

Whether to enable theming for Emacs.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.feh.enable

Whether to enable theming for the desktop background using Feh.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.firefox.enable

Whether to enable theming for Firefox.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.firefox.profileNames

The Firefox profile names to apply styling on.

Type: list of string

Default: `[ ]`

stylix.targets.fish.enable

Whether to enable theming for Fish.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.foot.enable

Whether to enable theming for Foot.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.fuzzel.enable

Whether to enable theming for Fuzzel.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.fzf.enable

Whether to enable theming for Fzf.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.gedit.enable

Whether to enable theming for GEdit.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.gitui.enable

Whether to enable theming for GitUI.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.gnome.enable

Whether to enable theming for GNOME.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.gtk.enable

Whether to enable theming for all GTK3, GTK4 and Libadwaita apps.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.gtk.extraCss

Extra code added to `gtk-3.0/gtk.css` and `gtk-4.0/gtk.css`.

Type: strings concatenated with "\n"

Default: `""`

Example:

```
''
  // Remove rounded corners
  window.background { border-radius: 0; }
''
```

stylix.targets.helix.enable

Whether to enable theming for Helix.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.hyprland.enable

Whether to enable theming for Hyprland.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.hyprpaper.enable

Whether to enable theming for Hyprpaper.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.i3.enable

Whether to enable theming for i3.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.k9s.enable

Whether to enable theming for k9s.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.kde.enable

Whether to enable theming for KDE.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.kitty.enable

Whether to enable theming for Kitty.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.kitty.variant256Colors

Whether to use the 256-color variant rather than the default combination of colors.

Type: boolean

Default: `false`

stylix.targets.lazygit.enable

Whether to enable theming for lazygit.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.mako.enable

Whether to enable theming for Mako.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.mangohud.enable

Whether to enable theming for mangohud.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.ncspot.enable

Whether to enable theming for Ncspot.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.neovim.enable

Whether to enable theming for Neovim.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.neovim.transparentBackground.main

Whether to enable background transparency for the main Neovim window.

Type: boolean

Default: `false`

Example: `true`

stylix.targets.neovim.transparentBackground.signColumn

Whether to enable background transparency for the Neovim sign column.

Type: boolean

Default: `false`

Example: `true`

stylix.targets.nixvim.enable

Whether to enable theming for nixvim.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.nixvim.transparentBackground.main

Whether to enable background transparency for the main NeoVim window.

Type: boolean

Default: `false`

Example: `true`

stylix.targets.nixvim.transparentBackground.signColumn

Whether to enable background transparency for the NeoVim sign column.

Type: boolean

Default: `false`

Example: `true`

stylix.targets.nushell.enable

Whether to enable theming for Nushell.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.qutebrowser.enable

Whether to enable theming for Qutebrowser.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.rofi.enable

Whether to enable theming for Rofi.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.sway.enable

Whether to enable theming for Sway.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.swaylock.enable

Whether to enable theming for Swaylock.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.swaylock.useImage

Whether to use your wallpaper image for the Swaylock background. If this is disabled, a plain color will be used instead.

Type: boolean

Default: `true`

stylix.targets.sxiv.enable

Whether to enable theming for Sxiv.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.tmux.enable

Whether to enable theming for Tmux.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.tofi.enable

Whether to enable theming for Tofi.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.vesktop.enable

Whether to enable theming for Vesktop.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.vim.enable

Whether to enable theming for Vim.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.vscode.enable

Whether to enable theming for VSCode.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.waybar.enable

Whether to enable theming for Waybar.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.waybar.enableCenterBackColors

enables background colors on the center of the bar

Type: boolean

Default: `false`

stylix.targets.waybar.enableLeftBackColors

enables background colors on the left side of the bar

Type: boolean

Default: `false`

stylix.targets.waybar.enableRightBackColors

enables background colors on the right side of the bar

Type: boolean

Default: `false`

stylix.targets.wezterm.enable

Whether to enable theming for wezterm.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.wofi.enable

Whether to enable theming for wofi.

Type: boolean

Default: `false`

Example: `true`

stylix.targets.wpaperd.enable

Whether to enable theming for wpaperd.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.xfce.enable

Whether to enable theming for Xfce.

Type: boolean

Default: `false`

Example: `true`

stylix.targets.xresources.enable

Whether to enable theming for Xresources.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.yazi.enable

Whether to enable theming for Yazi.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.zathura.enable

Whether to enable theming for Zathura.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

stylix.targets.zellij.enable

Whether to enable theming for zellij.

Type: boolean

Default: same as `stylix.autoEnable`

Example: `false`

Contributing to Stylix

Commit messages

To keep things consistent, commit messages should follow a format similar to Nixpkgs:

```
«scope»: «summary»

«motivation for change»
```

Where the scope is one of:

Scope	Purpose
`ci`	Changes to GitHub Actions workflows.
`doc`	Changes to the website, `README.md`, and so on.
`stylix`	Changes in the `stylix` directory, `flake.nix`, and other global code.
Name of target	Changes to code for a particular target.
`treewide`	Changes across many targets.

The scope is meant to indicate which area of the code was changed. Specifying the type of change, such as `feat` or `fix`, is not necessary. Dependency updates should use whichever scope they are related to.

The summary should start with a lowercase letter, and should not end with punctuation.

Most commits to `master` will also include a pull request number in brackets after the summary. GitHub adds this automatically when creating a squash merge.

Old commits

Commits before 2024 did not follow any particular format. Some have emojis from GitMoji.

Adding modules

Development setup

Currently the easiest way to test Stylix is to use the new code in your actual configuration.

You might find it useful to change the flake reference in your configuration from `github:danth/stylix` to `git+file:/home/user/path/to/stylix` so that you don't need to push your changes to GitHub during testing.

Then, remember to run `nix flake lock --update-input stylix` to refresh the flake each time you make an edit.

Nix only reads files which are tracked by Git, so you also need to `git add «file»` after creating a new file.

Module naming

Modules should be named like `modules/«name»/«platform».nix`. For example, `modules/avizo/hm.nix` is a Home Manager module which themes Avizo.

The following platforms are supported:

- NixOS (`nixos`)
- Home Manager (`hm`)
- Nix-Darwin (`darwin`)

Correctly named modules will be imported automatically.

Other files needed by the module can also be stored within the `modules/«name»` folder, using any name which is not on the list above.

Module template

All modules should have an enable option created using `mkEnableTarget`. This is similar to `mkEnableOption` from the standard library, however it integrates with `stylix.enable` and

`stylix.autoEnable` and generates more specific documentation.

A general format for modules is shown below.

```
{ config, lib, ... }:

{
  options.stylix.targets.«name».enable =
    config.lib.stylix.mkEnableTarget "«human readable name»" true;

  config = lib.mkIf (config.stylix.enable && config.stylix.targets.«name».enable) {
    programs.«name».backgroundColor = config.lib.stylix.colors.base00;
  };
}
```

The human readable name will be inserted into the following sentence:

Whether to enable theming for «human readable name».

If your module will touch options outside of `programs.«name»` or `services.«name»`, it should include an additional condition in `mkIf` to prevent any effects when the target is not installed.

The boolean value after `mkEnableTarget` should be changed to `false` if one of the following applies:

- The module requires further manual setup to work correctly.
- There is no reliable way to detect whether the target is installed, *and* enabling it unconditionally would cause problems.

How to apply colors

Refer to the style guide to see how colors are named, and where to use each one.

The colors are exported under `config.lib.stylix.colors`, which originates from `mkSchemeAttrs`.

You can use the values directly:

```
{
  environment.variables.MY_APPLICATION_COLOR = config.lib.stylix.colors.base05;
}
```

Or you can create a Mustache template and use it as a function. This returns a derivation which builds the template.

```
{
  environment.variables.MY_APPLICATION_CONFIG_FILE =
    let configFile = config.lib.stylix.colors {
      template = ./config.toml.mustache;
      extension = ".toml";
    };
    in "${configFile}";
}
```

Setting options through an existing NixOS or Home Manager module is preferable to generating whole files, since users will have the option of overriding things individually.

Also note that reading generated files with `builtins.readFile` can be very slow and should be avoided.

How to apply other things

For everything else, like fonts and wallpapers, you can just take option values directly from `config`. See the reference pages for a list of options.

Testbeds

Stylix provides a suite of virtual machines which can be used to test and preview themes without installing the target to your live system.

These can be particularly helpful for:

- Working on targets before the login screen, since you can avoid closing your editor to see the result.
- Developing for a different desktop environment than the one you normally use.
- Reducing the risk of breaking your system while reviewing pull requests.

Testbeds are also built by GitHub Actions for every pull request. This is less beneficial compared to running them yourself, since it cannot visually check the theme, however it can catch build failures which may have been missed otherwise.

Creation

New testbeds are defined by creating a file called `testbed.nix` within the folder for the corresponding target. This file will automatically be loaded as a NixOS module, with options such as `stylix.image` already defined. The module should include any options necessary to install the target and any supporting software - for example, a window manager.

If the target can only be used through Home Manager, you can write a Home Manager module within the NixOS module using the following format:

```
{
    home-manager.sharedModules = [{
        # Write Home Manager options here
    }];
}
```

Using `home-manager.sharedModules` is preferred over `home-manager.users.guest` since it allows us to easily change the username or add additional users in the future.

Once the module is complete, use `git add` to track the file, then the new packages will be available to use.

Usage

You can list the available testbeds by running this command from anywhere within the repository:

```
user@host:~$ nix flake show
github:danth/stylix
└───packages
    └───x86_64-linux
        ├───docs: package 'stylix-book'
        ├───palette-generator: package 'palette-generator'
        ├───testbed-gnome-dark: package 'testbed-gnome-dark'
        ├───testbed-gnome-light: package 'testbed-gnome-light'
        ├───testbed-kde-dark: package 'testbed-kde-dark'
        └───testbed-kde-light: package 'testbed-kde-light'
```

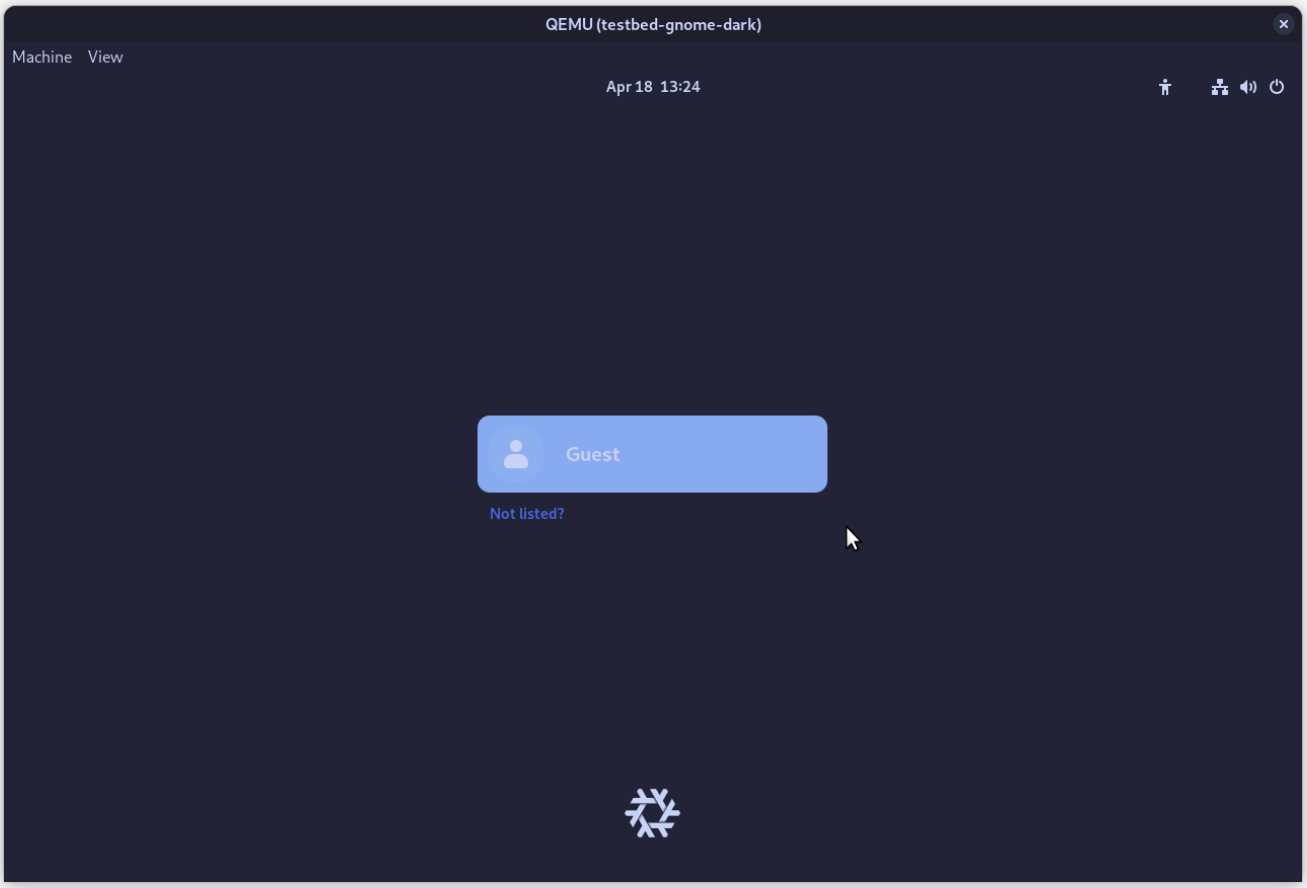
(This has been edited down to only the relevant parts.)

To start a testbed, each of which is named in the format `testbed-«target»-«polarity»`, run the following command:

```
user@host:~$ nix run .#testbed-«target»-«polarity»
```

Any package with a name not fitting the given format is not a testbed, and may behave differently with this command, or not work at all.

Once the virtual machine starts, a window should open, similar to the screenshot below. The contents of the virtual machine will vary depending on the target you selected earlier.



If the testbed includes a login screen, the guest user should log in automatically when selected. Depending on the software used, you may still be presented with a password prompt - in which case you can leave it blank and proceed by pressing enter.

Style guide

The base16 style guide is generally targeted towards text editors. Stylix aims to support a variety of other applications, and as such it requires its own guide to keep colours consistent. Towards this goal we will define several common types of applications and how to style each of them using the available colours.

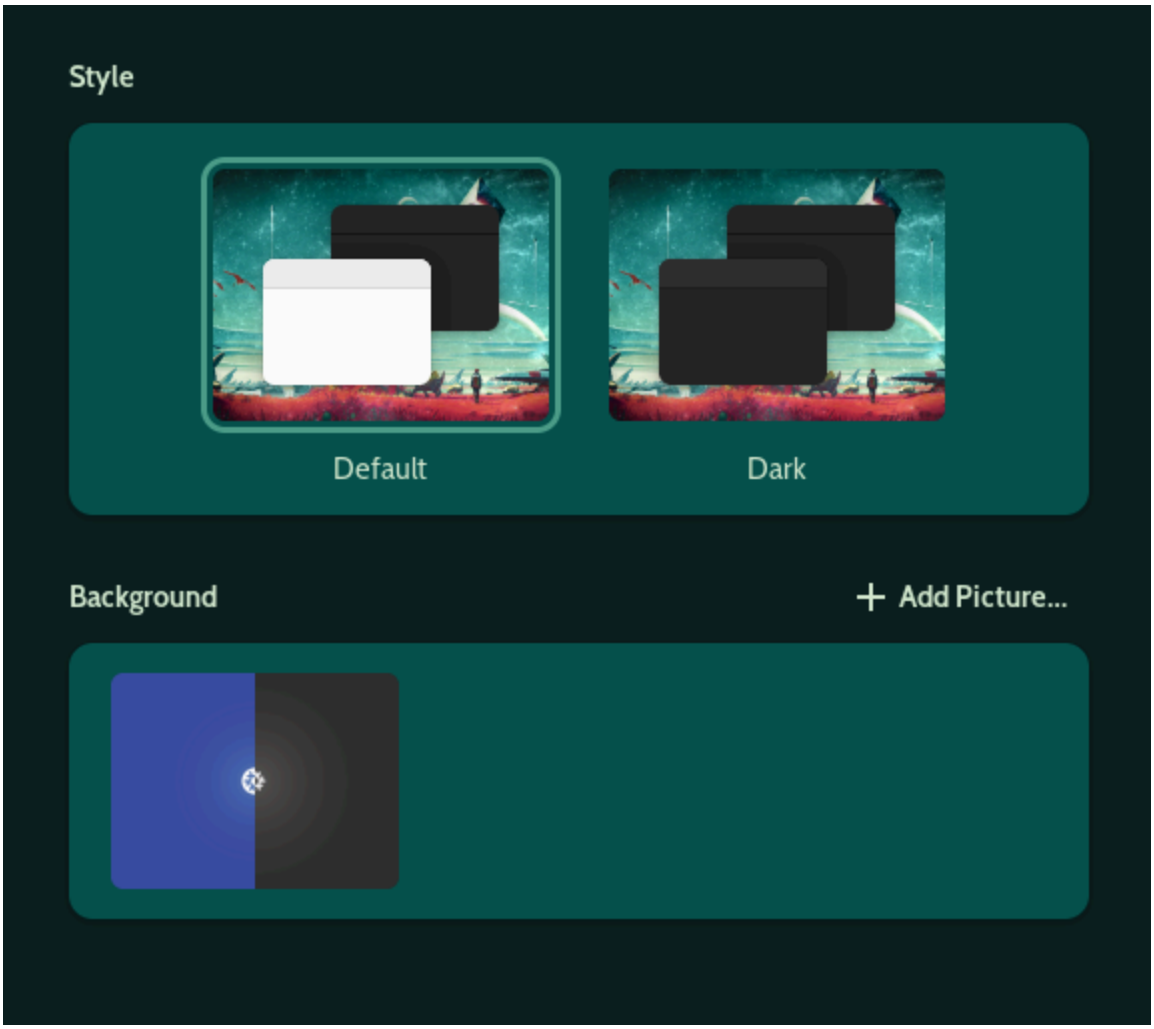
Please keep in mind that this is a general guide; there will be several applications that don't fit into any of the groups below. In this case it is up to the committer to make sure said application fits in stylistically with the rest of the themed applications.

It is also important to note that this is a growing theming guide and when theming an application and you find the guide to be lacking in any way in terms of direction, you are encouraged to open an issue regarding what you would like to see added to the style guide.

Terms

Alternate

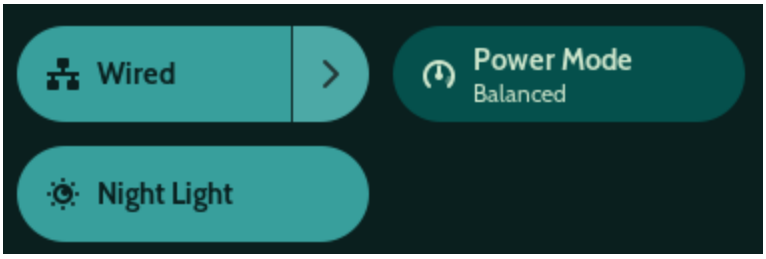
An alternate color should be used when something needs to look separate while not being drastically different. The smaller or less common element should use the alternate color.



For example, each section in this settings menu uses the alternate background color to separate it from the rest of the window, which is using the default background.

On/Off

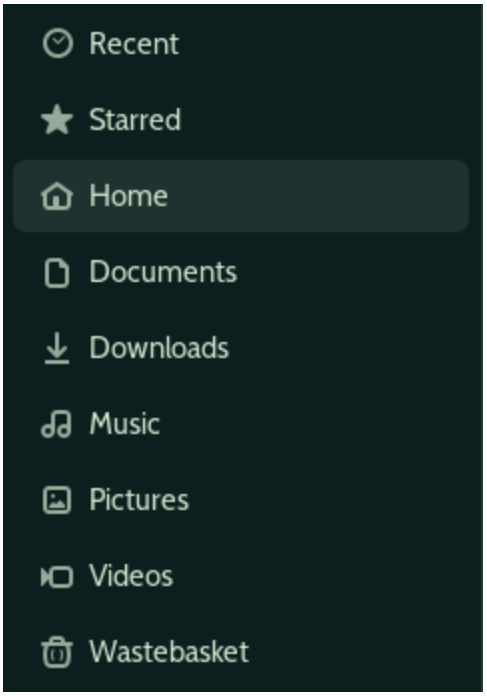
This is for toggles or simple status indicators which have an obvious on and off state.



In the screenshot above the Wired and Night Light buttons are on, Power Mode is off.

Lists and selections

A list of items to select between, such as tabs in a web browser. The selection is the currently active item, or there could be multiple selected depending on the use case.



General colors

- Default background: base00
- Alternate background: base01
- Selection background: base02
- Default text: base05
- Alternate text: base04
- Warning: base0A
- Urgent: base09
- Error: base08

Window Managers

Window Managers arrange windows and provide decorations like title bars and borders. Examples include Sway and i3.

This does not include applications bundled with the desktop environment such as file managers, which would fall into the general category. Desktop helpers such as taskbars and

menus are not technically part of the window manager, although they're often configured in the same place.

An urgent window is one which is grabbing for attention - Windows shows this by a flashing orange taskbar icon.

- Unfocused window border: base03
- Focused window border: base0D
- Unfocused window border in group: base03
- Focused window border in group: base0D
- Urgent window border: base08
- Window title text: base05

Notifications and Popups

Notifications and popups are any application overlay intended to be displayed over other applications. Examples include the mako notification daemon and avizo.

- Window border: base0D
- Low urgency background color: base06
- Low urgency text color: base0A
- High urgency background color: base0F
- High urgency text color: base08
- Incomplete part of progress bar: base01
- Complete part of progress bar: base02

Desktop Helpers

Applications that fall under this group are applications that complement the window management facilities of whatever window manager the user is using. Examples of this include waybar and polybar, as well as the similar programs that are part of KDE and GNOME.

Light text color widgets

Refer to general colors above.

Dark text color widgets

These widgets use a different text color than usual to ensure it's still readable when the background is more vibrant.

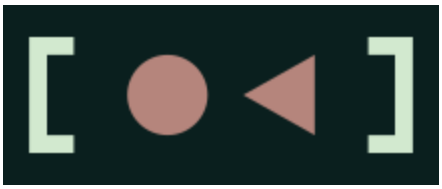
- Default text color: base00
- Alternate text color: base01
- Item on background color: base0E
- Item off background color: base0D
- Alternate item on background color: base09
- Alternate item off background color: base02
- List unselected background: base0D
- List selected background: base03

Images

For creating modified versions of logos, icons, etc; where we would rather the colors be similar to the original.

Note that the colors provided by the scheme won't necessarily match the names given below, although most handmade schemes do.

- Background color: base00
- Alternate background color: base01
- Main color: base05
- Alternate main color: base04
- Red: base08
- Orange: base09
- Yellow: base0A
- Green: base0B
- Cyan: base0C
- Blue: base0D
- Purple: base0E
- Brown: base0F



Example of a modified systemd logo. The square brackets are using the main color, which is usually be white or black depending on the polarity of the scheme.

Text Editors/Viewers

Text editors are any application that can view or edit source code. Examples include vim, helix, and bat.

For these please refer to the official base16 style guide.