



NixOS 25.05 released

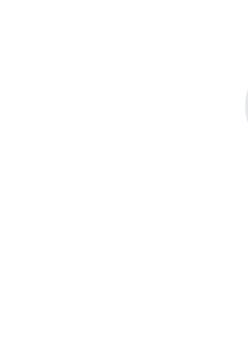
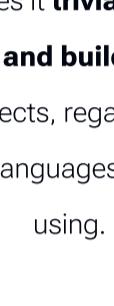
[Read Announcement →](#)

Declarative builds and deployments.

Nix is a tool that takes a unique approach to package management and system configuration. Learn how to make reproducible, declarative and reliable systems.

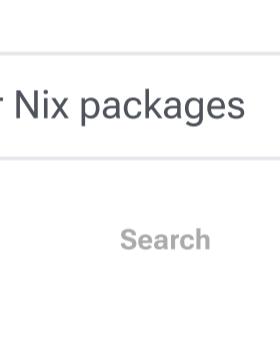
[Download](#)[Get started](#)

```
$ # Typing "nix-shell -p ..." each time can be tedious. We can do better.  
$ # We can write everything down in shell.nix  
$ cat -n shell.nix  
 1 {  
 2   pkgs ? import <nixpkgs> { }, # here we import the nixpkgs packages  
 3 }:  
 4 # mkShell is a helper function  
 5 pkgs.mkShell {  
 6   name = "dev-environment"; # that requires a name  
 7   buildInputs = [  
 8     # and a list of packages  
 9     pkgs.nodejs  
10   ];  
11   shellHook = ''  
12   # bash to run when you enter the shell  
13   echo "Start developing..."  
14   '';  
15 }  
$ # This snippet provides a developer environment named "dev-environment"  
$ # with the node command line tool available  
$ # and which will print "Start developing..." on first start.  
$ # To enter this developer environment, run:  
$ nix-shell  
00:00
```



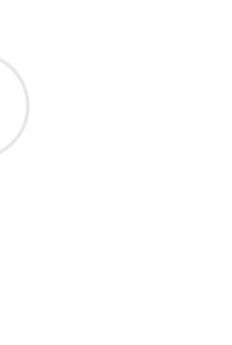
Reproducible

Nix builds packages in isolation from each other. This ensures that they are reproducible and don't have undeclared dependencies, so **if a package works on one machine, it will also work on another.**



Declarative

Nix makes it **trivial to share development and build environments** for your projects, regardless of what programming languages and tools you're using.



Reliable

Nix ensures that installing or upgrading one package **cannot break other packages**. It allows you to **roll back to previous versions**, and ensures that no package is in an inconsistent state during an upgrade.

Choose from over 120 000 Packages

The Nix Packages collection ([Nixpkgs](#)) offers a large selection of packages for the Nix package manager.

Search

Try new tools without fear

```
$ # Lets see if python is present on the system  
$ python --version  
python: command not found  
$ # Use nix-shell to create a shell environment with python  
$ nix-shell -p python3  
(nix-shell) $ python --version  
Python 3.7.7  
(nix-shell) $ # YAAAY! Python is available  
(nix-shell) $ exit  
$ # And this is how you create on demand environments
```



00:00

Multiple languages, one tool

```
$ # Lets create an environment with multiple packages  
$ nix-shell -p python3 nodejs go rustc  
(nix-shell) $ node --version  
v10.20.1  
(nix-shell) $ go version  
go version go1.14.1 linux/amd64  
(nix-shell) $ rustc --version  
rustc 1.42.0  
(nix-shell) $ # Imagine how easy  
(nix-shell) $ exit  
$ # And POOF, just like that you are back to your normal environment  
$ # playing aro
```



00:00

Declarative dev-environments

```
2   pkgs ? import <nixpkgs> { }, # here we import the nixpkgs  
3 }:  
4 # mkShell is a helper function  
5 pkgs.mkShell {  
6   name = "dev-environment"; # that requires a name  
7   buildInputs = [  
8     # and a list of packages  
9     pkgs.python3  
10    pkgs.python3Packages.virtualenv  
11    pkgs.nodejs  
12    pkgs.yarn  
13  ];  
14  shellHook = ''  
15  # bash to run when you enter the shell  
16  echo "Start developing..."  
17  '';  
18 }  
$ # Pause the video to read and understand the shell.nix  
$ # To enter the dev-environment simply run:  
$ nix-shell  
Start developing...  
(nix-shell) $ python --version  
Python 3.7.7  
(nix-shell) $ v:
```



00:00

Minimal docker images

```
$ # We all love docker. But over time it can become tedious to write reliable docker files.  
$ # What if you could use the power of Nix to build Docker images  
$ cat -n docker.nix  
 1 {  
 2   pkgs ? import <nixpkgs> { system = "x86_64-linux"; }, #  
set  
 3 }:  
 4 # helper to build Docker image  
 5 pkgs.dockerTools.buildLayeredImage {  
 6   name = "nix-hello"; # give docker image a name  
 7   tag = "latest"; # provide a tag  
 8   contents = [ pkgs.hello ]; # packages in docker image  
 9 }  
$ # Pause the video to read and understand the docker.nix  
$ # Now we build a Docker image
```



00:00

Declarative cloud images

```
$ # How hard would it be to build and configure an Amazon EC2 image  
$ # Let us configure a Nginx to serve a "Welcome to nginx!" page,  
$ # valid SSL certificate (via LetsEncrypt) and recommended security  
$ cat -n amazon.nix  
 1 { pkgs, ... :  
 2 {  
 3   security.acme.acceptTerms = true;  
 4   security.acme.email = "nix@example.com";  
 5   services.nginx = {  
 6     enable = true;  
 7     recommendedGzipSettings = true;  
 8     recommendedOptimizedCaching = true;  
 9     recommendedProxySettings = true;  
10    recommendedTlsSettings = true;  
11    virtualHosts."example.com" = {  
12      enableACME = true;  
13      forceSSL = true;  
14      locations."/".root = "${pkgs.nginx}/html";  
15    };  
16  };  
17 }  
$ # Pause the video to understand the nixfile  
$ # To enter the dev-environment simply run:  
$ nix-shell  
Start developing...  
(nix-shell) $ python --version  
Python 3.7.7  
(nix-shell) $ v:
```



00:00

Test your configurations

```
$ # In this example we will look into how to test your NixOS configuration  
$ # We will create a simple configuration with the 'hello' package  
$ # system wide and check that the 'hello' world binary works.  
$ cat -n test.nix  
 1 {  
 2   pkgs ? import <nixpkgs> { },  
 3 }:  
 4 pkgs.nixosTest {  
 5   name = "example-test";  
 6   # virtual machine with one package installed system wide  
 7   machine = { pkgs, ... :  
 8     environment.system.packages = [ pkgs.hello ];  
 9   };  
10  testScript = ''  
11  # run hello on machine and check for output  
12  machine.succeed('Hello, world!')  
13  # test is a simple python script  
14  '';  
15 }  
$ # Pause the video to understand the nixfile  
$ # To enter the dev-environment simply run:  
$ nix-shell  
Start developing...  
(nix-shell) $ python --version  
Python 3.7.7  
(nix-shell) $ v:
```



00:00

The Project

[Channel Status](#)[Packages search](#)[Options search](#)[Reproducible Builds Status](#)[Security](#)[Summer of Nix](#)

Get in Touch

[Forum](#)[Matrix Chat](#)[Commercial support](#)

Contribute

[Contributing Guide](#)[Donate](#)

Stay up to Date

[Blog](#)[Research & Publications](#)[NixOS Weekly \(Archive\)](#)