# Setting up post-build hooks

## Contents



This guide shows how to use the Nix `post-build-hook` configuration option to automatically upload build results to an S3-compatible binary cache.

## Implementation caveats

This is a simple and working example, but it is not suitable for all use cases.

The post-build hook program runs after each executed build, and blocks the build loop. The build loop exits if the hook program fails.

Concretely, this implementation will make Nix slow or unusable when the network connection is slow or unreliable. A more advanced implementation might pass the store paths to a user-supplied daemon or queue for processing the store paths outside of the build loop.

This tutorial assumes you have configured an S3-compatible binary cache, and that the `root` user's default AWS profile can upload to the bucket.

Use `nix-store --generate-binary-cache-key` to create a pair of cryptographic keys. You will sign paths with the private key, and distribute the public key for verifying the authenticity of the paths.

Skip to main content

```
$ nix-store --generate-binary-cache-key example-nix-cache-1 /etc/nix/key.private /e
$ cat /etc/nix/key.public
example-nix-cache-1:1/cKDz3QCCOmwcztD2eV6Coggp6rqc9DGjWv7C0G+rM=
```

Configure Nix to use a custom binary cache on any machine that will access the bucket. For example, add the cache URL to **`substituters`** and the public key to **`trusted-public-keys`** in `nix.conf`:

```
substituters = https://cache.nixos.org/ s3://example-nix-cache
trusted-public-keys = cache.nixos.org-1:6NCHdD59X431o0gWypbMrAURkbJ16ZPMQFGspcDShjY
```

Machines that build for the cache must sign derivations using the private key. The path to the file containing the private key you just generated must be added to the **`secret-key-files`** setting for those machines:

```
secret-key-files = /etc/nix/key.private
```

Write the following script to `/etc/nix/upload-to-cache.sh`:

```
#!/bin/sh
set -eu
set -f # disable globbing
export IFS=' '
echo "Uploading paths" $OUT_PATHS
exec nix copy --to "s3://example-nix-cache" $OUT_PATHS
```

The `$OUT_PATHS` variable is a space-separated list of Nix store paths. In this case, we expect and want the shell to perform word splitting to make each output path its own argument to `nix store sign`. Nix guarantees the paths will not contain any spaces, however a store path might contain glob characters. The `set -f` disables globbing in the shell.

Make sure the hook program is executable by the `root` user:

```
# chmod +x /etc/nix/upload-to-cache.sh
```

Set the **`post-build-hook`** configuration option on the local machine to run the hook:

```
post-build-hook = /etc/nix/upload-to-cache.sh
```

Skip to main content

```
pkill nix-daemon
```

Build any derivation, for example:

```
$ nix-build -E '(import <nixpkgs> {}).writeText "example" (builtins.toString builti
this derivation will be built:
  /nix/store/s4pnfbkalzy5qz57qs6yybna8wylkig6-example.drv
building '/nix/store/s4pnfbkalzy5qz57qs6yybna8wylkig6-example.drv'...
running post-build-hook '/home/grahamc/projects/github.com/NixOS/nix/post-hook.sh'.
post-build-hook: Signing paths /nix/store/ibcyipq5gf91838ldx40mjsp0b8w9n18-example
post-build-hook: Uploading paths /nix/store/ibcyipq5gf91838ldx40mjsp0b8w9n18-exampl
/nix/store/ibcyipq5gf91838ldx40mjsp0b8w9n18-example
```

To check that the hook took effect, delete the path from the store, and try substituting it from the binary cache:

```
$ rm ./result
$ nix-store --delete /nix/store/ibcyipq5gf91838ldx40mjsp0b8w9n18-example
$ nix-store --realise /nix/store/ibcyipq5gf91838ldx40mjsp0b8w9n18-example
copying path '/nix/store/m8bmqwrch6l3h8s0k3d673xpmipcdpsa-example from 's3://exampl
warning: you did not specify '--add-root'; the result might be removed by the garba
/nix/store/m8bmqwrch6l3h8s0k3d673xpmipcdpsa-example
```

You have configured Nix to automatically sign and upload every local build to a remote S3-compatible binary cache.

Before deploying this to production, be sure to consider the implementation caveats.