Setting up an HTTP binary cache

Contents

A binary cache stores pre-built Nix store objects and provides them to other machines over the network. Any machine with a Nix store can be a binary cache for other machines.

Introduction

In this tutorial you will set up a Nix binary cache that will serve store objects from a NixOS machine over HTTP or HTTPS.

What will you learn?

You'll learn how to:

- Set up signing keys for your cache
- Enable the right services on the NixOS machine serving the cache
- Check that the setup works as intended

What do you need?

Skip to main content

- SSH access to a NixOS machine to use as a cache

  If you're new to NixOS, learn about the module system and configure your first system with NixOS virtual machines.

- (optional) A public IP and DNS domain

  If you don't host yourself, check NixOS friendly hosters on the NixOS Wiki. Follow the tutorial on Provisioning remote machines via SSH to deploy your NixOS configuration.

For a cache on a local network, we assume:

- The hostname is `cache` (replace it with yours, or an IP address)
- The host serves store objects via HTTP on port 80 (this is the default)

For a publicly accessible cache, we assume:

- The domain name is `cache.example.com` (replace it with yours)
- The host serves store objects via HTTPS on port 443 (this is the default)

How long will it take?

- 25 minutes

Set up services

For the NixOS machine hosting the cache, create a new configuration module in `binary-cache.nix`:

```
{ config, ... }:

{
  services.nix-serve = {
    enable = true;
    secretKeyFile = "/var/secrets/cache-private-key.pem";
  };

  services.nginx = {
    enable = true;
    recommendedProxySettings = true;
    virtualHosts.cache = {
      locations."/".proxyPass = "http://${config.services.nix-serve.bindAddress}:${
    };
  };
```

Skip to main content

```
  ];
}
```

The options under `services.nix-serve` configure the binary cache service.

`nix-serve` doesn't support IPv6 or SSL/HTTPS. The `services.nginx` options are used to set up a proxy, which does support IPv6, to handle requests to the hostname `cache`.

Important

There is an optional HTTPS section at the end of this tutorial.

Add the new NixOS module to the existing machine configuration:

```
{ config, ... }:

{
  imports = [
    ./binary-cache.nix
  ];

  # ...
}
```

From your local machine, deploy the new configuration:

```
nixos-rebuild switch --no-flake --target-host root@cache
```

Note

The binary cache daemon will report errors because there is no secret key file, yet.

Generate a signing key pair

A pair of private and public keys is required to ensure that the store objects in the cache are authentic.

To generate a key pair for the binary cache, replace the example hostname `cache.example.com` with your hostname:

Skip to main content

```
nix-store --generate-binary-cache-key cache.example.com cache-private-key.pem cache
```

`cache-private-key.pem` will be used by the binary cache daemon to sign the binaries as they are served. Copy it to the location configured in `services.nix-serve.secretKeyFile` on the machine hosting the cache:

```
scp cache-private-key.pem root@cache:/var/secrets/cache-private-key.pem
```

Up until now, the binary cache daemon was in a restart loop due to the missing secret key file. Check that it now works correctly:

```
ssh root@cache systemctl status nix-serve.service
```

Important

Configure Nix to use a custom binary cache using `cache-public-key.pem` on your local machine.

Test availability

The following steps check if everything is set up correctly and may help with identifying problems.

Check general availability

Test if the binary cache, reverse proxy, and firewall rules work as intended by querying the cache:

```
$ curl http://cache/nix-cache-info
StoreDir: /nix/store
WantMassQuery: 1
Priority: 30
```

Skip to main content

Check store object signing

To test if store objects are signed correctly, inspect the metadata of a sample derivation. On the binary cache host, build the `hello` package and get the `.narinfo` file from the cache:

```
$ hash=$(nix-build '<nixpkgs>' -A pkgs.hello | awk -F '/' '{print $4}' | awk -F '-'
$ curl "http://cache/$hash.narinfo" | grep "Sig: "
...
Sig: cache.example.org:GyBFzocLAeLEFd0hr2noK84VzPUw0ArCNYEnrm1YXakdsC5FkO2Bkj2JH8Xj
```

Make sure that the output contains this line prefixed with `Sig:` and shows the public key you generated.

Serving the binary cache via HTTPS

If the binary cache is publicly accessible, it is possible to enforce HTTPS with Let's Encrypt SSL certificates. Edit your `binary-cache.nix` like this and make sure to replace the example URL and mail address with yours:

```
   services.nginx = {
     enable = true;
     recommendedProxySettings = true;
-    virtualHosts.cache = {
+    virtualHosts."cache.example.com" = {
+      enableACME = true;
+      forceSSL = true;
       locations."/".proxyPass = "http://${config.services.nix-serve.bindAddress}:$
     };
   };

+  security.acme = {
+    acceptTerms = true;
+    certs = {
+      "cache.example.com".email = "you@example.com";
+    };
+  };

   networking.firewall.allowedTCPPorts = [
     config.services.nginx.defaultHTTPListenPort
+    config.services.nginx.defaultSSLListenPort
   ];
```

Rebuild the system to deploy these changes:

```
nixos-rebuild switch --no-flake --target-host root@cache.example.com
```

Skip to main content

Next steps

If your binary cache is already a remote build machine, it will serve all store objects in its Nix store.

- Configure Nix to use a custom binary cache using the binary cache's hostname and the generated public key
- Setting up post-build hooks to upload store objects to the binary cache
- Setting up distributed builds

To save storage space, please refer to the following NixOS configuration attributes:

- `nix.gc`: Options for automatic garbage collection
- `nix.optimise`: Options for periodic Nix store optimisation

Alternatives

- `nix-serve-ng`: A drop-in replacement for `nix-serve`, written in Haskell
- The SSH Store, Experimental SSH Store, and the S3 Binary Cache Store can also be used to serve a cache. There are many commercial providers for S3-compatible storage, for example:
  - Amazon S3
  - Tigris
  - Cloudflare R2
- attic: Nix binary cache server backed by an S3-compatible storage provider
- Cachix: Nix binary cache as a service

References

- Nix Manual on HTTP Binary Cache Store
- `services.nix-serve` module options
- `services.nginx` module options