Host Your Own Nix Binary Cache Server

Introduction

The Nix binary cache is an implementation of the Nix Store that stores data on a remote server rather than locally, facilitating the sharing of binary caches across multiple machines.

The official Nix binary cache server only provides binaries built with standard parameters. If you've customized build parameters or are using packages outside of Nixpkgs, Nix won't find the corresponding binary cache, resulting in local builds.

Relying solely on your local Nix Store `/nix/store` can be cumbersome, as you'd need to rebuild all your custom packages on each machine, which can be time-consuming and memory-intensive. This situation is exacerbated on lower-performance platforms like Raspberry Pi.

This document will show you how to set up your own Nix binary cache server using an S3 service (like MinIO) to share build results across machines and address the aforementioned issues.

# Prerequisites

1. A NixOS host
2. Deployed MinIO server
   1. If not, you can follow MinIO's [official deployment guide]().
3. The MinIO server needs a valid TLS digital certificate, which can be public or private. This example will use `https://minio.homelab.local` with a private certificate.
4. Install `minio-client`

Generating a Password

```bash
nix run nixpkgs#pwgen -- -c -n -y -s -B 32 1
# => oenu1Yuch3rohz2ahveid0koo4giecho
```

Setting Up the MinIO Client

Install the MinIO command-line client `mc`.

```nix
{ pkgs, ... }:
{
  environment.systemPackages = with pkgs; [
    minio-client # Alternatives for ls, cp, mkdir, diff, and rsync commands for
  ];
}
```

Create `~/.mc/config.json` with the following content (replace the key parameters with your own):

```json
{
  "version": "10",
  "aliases": {
    "s3": {
      "url": "https://s3.homelab.local",
      "accessKey": "minio",
      "secretKey": "oenu1Yuch3rohz2ahveid0koo4giecho",
      "api": "s3v4",
      "path": "auto"
    }
  }
}
```

Since Nix will interact directly with the S3 bucket, we need to configure S3 credentials for all machines that require access to the Nix binary cache.

Create `~/.aws/credentials` with the following content (replace `<nixbuildersecret>` with the password generated by the `pwgen` command).

```conf
[nixbuilder]
aws_access_key_id=nixbuilder
aws_secret_access_key=<nixbuildersecret>
```

Setting Up S3 Bucket as Binary Cache

Create the `nix-cache` bucket using the minio client:

```bash
mc mb s3/nix-cache
```

Create the `nixbuilder` user for MinIO and assign it a password:

```bash
mc admin user add s3 nixbuilder <PASSWORD>
```

Create a file named `nix-cache-write.json` in the current working directory with the following content:

```json
{
  "Id": "AuthenticatedWrite",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AuthenticatedWrite",
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::nix-cache", "arn:aws:s3:::nix-cache/*"],
      "Principal": "nixbuilder"
    }

  ]
}
```

Now, create a policy for uploading files to S3 using the `nix-cache-write.json` file:

```bash
mc admin policy create s3 nix-cache-write nix-cache-write.json
```

Associate the S3 policy we just created with the `nixbuilder` user:

```bash
mc admin policy attach s3 nix-cache-write -user nixbuilder
```

Allow anonymous users to download files without authentication, so all Nix servers can pull data directly from this S3 cache:

```bash
mc anonymous set download s3/nix-cache
```

Finally, add the `nix-cache-info` file to the S3 bucket root directory, as Nix requires this file to record some information related to the binary cache:

```bash
cat > nix-cache-info <<EOF
StoreDir: /nix/store
WantMassQuery: 1
Priority: 40
EOF
# Copy `nix-cache-info` to the S3 bucket
mc cp ./nix-cache-info s3/nix-cache/nix-cache-info
```

Generating Signature Key Pair

As mentioned earlier, the Nix binary cache uses a public key signature mechanism to verify the origin and integrity of the data, so we need to generate a key pair for our Nix build machine to sign the binary cache. The key name is arbitrary, but NixOS developers strongly recommend using the cache domain followed by an integer, so if the key needs to be revoked or regenerated, you can simply increment the integer at the end.

bash
```bash
nix key generate-secret --key-name s3.homelab.local-1 > ~/.config/nix/secret.key
nix key convert-secret-to-public < ~/.config/nix/secret.key > ~/.config/nix/publ
cat ~/.config/nix/public.key
# => s3.homelab.local-1:m0J/oDlLEuG6ezc6MzmpLCN2MYjssO3NMIlr9JdxkTs=
```

Using S3 Binary Cache in `flake.nix`

Add the following to your `configuration.nix` or any custom NixOS module:

nix
```nix
{
  nix = {
    settings = {
      # The substituter will be appended to the default substituters when fetchi
      extra-substituters = [
        "https://s3.homelab.local/nix-cache/"
      ];
      extra-trusted-public-keys = [
        "s3.homelab.local-1:m0J/oDlLEuG6ezc6MzmpLCN2MYjssO3NMIlr9JdxkTs="
      ];
    };
  };
}
```

Rebuild the system to start using our newly created S3 binary cache:

bash
```bash
sudo nixos-rebuild switch --upgrade --flake .#<HOST>
```

Pushing Store Paths to Binary Cache

Sign some paths in the local store.

```bash
nix store sign --recursive --key-file ~/.config/nix/secret.key /run/current-syst
```

Copy these paths to the cache:

```bash
nix copy --to 's3://nix-cache?profile=nixbuilder&endpoint=s3.homelab.local' /run
```

Adding Automatic Object Expiration Policy

```bash
mc ilm rule add s3/nix-cache --expire-days "DAYS"
# For example: mc ilm rule add s3/nix-cache --expire-days "7"
```

This will set an expiration policy for objects in the S3 bucket, ensuring that they are automatically removed after a specified number of days.

This is useful for keeping the cache size manageable and ensuring that outdated binaries are not stored indefinitely.

References

- Blog post by Jeff on Nix binary caches
- Binary cache in the NixOS wiki
- Serving a Nox store via S3 in the NixOS manual
- Serving a Nix store via HTTP in the NixOS manual

Loading comments...