Intro to Easy Deployments with NixOps

Ben Sima[1]

2018-03-21 Wed

[1] ben@bsima.me

Abstract

NixOps is a declarative deployment tool and language for Linux. The same configuration for your local VM will also deploy to your AWS cluster, Digital Ocean droplets, or bare-metal data center. Configurations are versioned and can be rolled back with a single command. This talk will cover the basics of NixOps, the Nix configuration language, and provide resources for how to start using Nix today.

Follow along:

- Slides at `github.com/bsima/talks`
- Install Nix via `nixos.org`
- Install NixOps with `nix-env -i nixops`

What is Nix?

- declarative, purely functional language
- configuration DSL, ecosystem, libraries, package manager
- build system?
- started as a research project
- now a full Linux distro
- yes, it's actually used in production

Caveats

- Nix 2.0
- nixops is currenly single user
- everything is in development

Nix lang data types

(Run these examples in `nix-repl`)

```
n = 42                                  # variables
"Hello World, ${toString n}"            # strings
''                                      # multiline strings
Hello World.
The answer is ${toString n}.
''
[ 1 2 3 "hi" 4 ]                        # list
s = { attr = "value" }                  # sets (dictionary)
s.attr                                  # accessing attributes
./src/Makefile                          # first-class paths
```

Nix lang functions

```
nix-repl> (x: y: x+y) 2 3
5

nix-repl> add = x: y: x+y
nix-repl> add 2 3
5

nix-repl> mul = {x, y, ...}: x * y
nix-repl> mul {x = 5; y = 2; z = "hello"}
10

nix-repl> setKey = v: { key = v; }
nix-repl> setKey 42
{ key = 42; }
```

Nix hello world package

hello.nix

```
{ stdenv, fetchurl, perl }:

stdenv.mkDerivation {
  name = "hello-2.1.1";
  builder = ./builder.sh;
  src = fetchurl {
    url = ftp://ftp.nluug.nl/pub/gnu/hello/hello-2.1.1.tar.gz;
    sha256 = "1md7jsfd8pa45z73bz1kszpp01yw6x5ljkjk2hx7wl800any6
  };
  inherit perl;
}
```

Nix derivation

```
$ echo mycontent > myfile

$ nix-repl
nix-repl> d = derivation { system = "x86_64-linux";
                           builder = ./myfile;
                           name = "foo"; }
«derivation /nix/store/y4h73bmrc9ii5bxg6i7ck6hsf5gqv8ck-foo.drv

nix-repl> d.outPath
"/nix/store/hs0yi5n5nw6micqhy8l1igkbhqdkzqa1-foo"
```

3 main artifacts

expression -> derivation -> build product

Nix derivation

```
$ cat /nix/store/y4h73bmrc9ii5bxg6i7ck6hsf5gqv8ck-foo.drv
Derive(
  [ ("out","/nix/store/hs0yi5n5nw6micqhy8l1igkbhqdkzqa1-foo",""
  [], [ "/nix/store/xv2iccirbrvklck36f1g7vldn5v58vck-myfile" ],
  "x86_64-linux",
  "/nix/store/xv2iccirbrvklck36f1g7vldn5v58vck-myfile",
  [],
  [
   (
    "builder",
    "/nix/store/xv2iccirbrvklck36f1g7vldn5v58vck-myfile"
   ),
   ("name","foo"),
   ("out","/nix/store/hs0yi5n5nw6micqhy8l1igkbhqdkzqa1-foo"),
   ("system","x86_64-linux")
  ]
)
```

Nix store unique hashes

1. Compute hash of the file

```
$ nix-hash --type sha256 myfile
2bfef67de873c54551d884fdab3055d84d573e654efa79db3c0d7b98883
```

2. Build the string description

```
$ echo -n "source:sha256:2bfef67de873c54551d884fdab3055d84d
  > myfile.str
```

3. Compute final hash

```
$ nix-hash --type sha256 --truncate \
  --base32 --flat myfile.str
xv2iccirbrvklck36f1g7vldn5v58vck
```

Nix store tree

```
$ nix-store --query --tree /nix/store/y4h73bmrc9ii5bxg6i7ck6hsf
/nix/store/y4h73bmrc9ii5bxg6i7ck6hsf5gqv8ck-foo.drv
+---/nix/store/xv2iccirbrvklck36f1g7vldn5v58vck-myfile
```

haskell-src-1.0.2.0.drv store tree

```
$ nix-store --query --tree /nix/store/05gy5ywc...
```

NixOS configuration

/etc/nixos/configuration.nix

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

{
  imports = [ ./hardware-configuration.nix ];

  boot.loader.systemd-boot.enable = true;
  boot.loader.efi.canTouchEfiVariables = true;

  networking.hostName = "hal9000";
  networking.firewall.allowedTCPPorts = [ 22 ];
  environment.systemPackages = with pkgs; [ emacs ];
  fonts.fonts = with pkgs; [ google-fonts source-code-pro ];
  hardware.opengl.enable = true;
  services.openssh.enable = true;
}
```

NixOS – easy SSL!

```
services.fail2ban.enable = true;
services.nginx = {
  enable = true;
  virtualHosts."example.com" = {
    location."/".proxyPass = "http://127.0.0.1:9000";
    enableACME = true;
    forceSSL = true;
  };
};
```

NixOS binary cache

- https://cache.nixos.org
- nix-copy-closure

```
# on the server:
services.nix-serve = {
  enable = true;
  port = 5000; # default
  secretKeyFile = /var/nix-serve-key.pem;
};

# on the client:
nix.binaryCaches = [ "https://cache.example.org" ];
```

NixOS Hydra

- https://hydra.nixos.org/
- https://hydra.dhall-lang.org/
- https://github.com/TaktInc/hail

```
services.hydra = {
  enable = true;
};
```

Basic NixOps configuration (1)

In deployment.nix:

```
{
  webserver =
    { deployment.targetEnv = "virtualbox";
      services.httpd.enable = true;
      services.httpd.documentRoot = "/data";
      fileSystems."/data" =
        { fsType = "nfs4";
          device = "fileserver:/"; };
    };

  fileserver =
    { deployment.targetEnv = "virtualbox";
      services.nfs.server.enable = true;
      services.nfs.server.exports = "...";
    };
}
```

Basic NixOps configuration (2)

In your shell:

```
nixops create -d simple deployment.nix
nixops deploy -d simple
```

Building docker containers

- https://nixos.org/nixpkgs/manual/#sec-pkgs-dockerTools

```
buildImage {
  name = "redis";
  tag = "latest";
  fromImage = someBaseImage;
  fromImageName = null;
  fromImageTag = "latest";
  contents = pkgs.redis;
  runAsRoot = ''
    #!${stdenv.shell}
    mkdir -p /data
  '';
  config = {
    Cmd = [ "/bin/redis-server" ];
    WorkingDir = "/data";
    Volumes = {
      "/data" = {};
```

Help - Where do I go when I get stuck?

- IRC: `#nixos` on Freenode (I'm `bsima`)
- Manuals on nixos.org/nixos/support.html
- StackOverflow `nixos` and `nixops` tag
- `grep` source code on github.com/nixos/nixpkgs
- GiHub code search (surprisingly helpful)
- Cheatsheet: nixos.wiki/wiki/Cheatsheet
- Slides: github.com/bsima/talks