NixOps: Declarative Provisioning and Deployment

Eelco Dolstra Rob Vermaas Shea Levy

LogicBlox Inc.

RELENG 2013

May 20, 2013, San Francisco

LOGICBLOX®

NixOS

Overview

NixOS

Previous work:

- Nix: a purely functional package manager
- NixOS: a Linux distribution with a declarative configuration management model

This talk:

- NixOps: a tool for declarative provisioning and deployment of networks of NixOS machines

Overview

NixOS

Previous work:

- Nix: a purely functional package manager
- NixOS: a Linux distribution with a declarative configuration management model

This talk:

- NixOps: a tool for declarative provisioning and deployment of networks of NixOS machines

Overview

NixOS

Previous work:

- Nix: a purely functional package manager
- NixOS: a Linux distribution with a declarative configuration management model

This talk:

- NixOps: a tool for declarative provisioning and deployment of networks of NixOS machines

Nix: Purely functional package management

Main idea: store all packages in isolation from each other:

`/nix/store/rpdqxnilb0cg...-firefox-3.5.4`

Paths contain a 160-bit **cryptographic hash** of **all** inputs used to build the package.

Advantages:

- Atomic upgrades
- Rollbacks
- Reproducible
- Multiple versions
- Correct dependencies
- Source-based

```
/nix/store
├── l9w6773m1msy...-openssh-4.6
│   ├── bin
│   │   └── ssh
│   └── sbin
│       └── sshd
├── smkabrbibqv7...-openssl-0.9.8
│   └── lib
│       └── libssl.so.0.9.8
├── c6jbqm2mc0a7...-zlib-1.2.3
│   └── lib
│       └── libz.so.1.2.3
└── im276akmsrhv...-glibc-2.5
    └── lib
        └── libc.so.6
```

NixOS

A Linux distribution that builds all static parts of a system using Nix:

- Packages
- Configuration files
- Systemd units
- Boot scripts
- ...

Advantages:

- Reproducible
- Transactional upgrades
- Rollbacks
- Multi-user package management

```
/nix/store
├─ l9w6773m1msy...-openssh-4.6
│  ├─ bin
│  │  └─ ssh
│  └─ sbin
│     └─ sshd
├─ 21gbj37rhibx...-sshd_config
├─ dz0snsw724pf...-sshd.service
└─ ...
```

NixOS configuration

/etc/nixos/configuration.nix

```
{
  boot.loader.grub.bootDevice = "/dev/sda";
  fileSystems."/".device = "/dev/sda1";
  services.sshd.enable = true;
  services.postgresql.enable = true;
  services.httpd.enable = true;
  services.httpd.documentRoot = ...;
}
```

NixOps — Logical configuration

NixOps extends the NixOS approach to networks of machines.

logical.nix

```
{
  database =
    { services.postgresql.enable = true;
    };

  webserver =
    { services.httpd.enable = true;
      services.httpd.documentRoot = ...;
    };
}
```

NixOps — Physical configuration

physical-vbox.nix

```
{
  database =
    { deployment.targetEnv = "virtualbox";
    };

  webserver =
    { deployment.targetEnv = "virtualbox";
    };
}
```

NixOps — Physical configuration

physical-ec2.nix

```
{
  database =
    { deployment.targetEnv = "ec2";
      deployment.ec2.region = "us-east-1";
    };

  webserver =
    { deployment.targetEnv = "ec2";
      deployment.ec2.region = "eu-west-1";
    };
}
```

NixOps — Deploying

```
$ nixops create -n foo \
    ./logical.nix ./physical-vbox.nix
$ nixops deploy -d foo
```

This will:

- Create all machines
- Build/download all dependencies
- Upload them to the machines
- Activate any necessary services

NixOps — Redeploying

Just edit the spec and do

```
$ nixops deploy -d foo
```

This will:

- Create new machines
- Destroy obsolete machines
- Rebuild new dependencies
- Restart changed services, start new services, stop obsolete services

Conclusion

NixOps is a tool for provisioning and deploying networks of NixOS Linux machines.

- Declarative: System figures out what needs to be done to realize a change to the spec.
- Integrated provisioning and deployment.
- Allows abstracting over target environment.

More info: `http://nixos.org`, `https://github.com/NixOS/nixops`

Question

Is declarative the way to go? Isn't it easier to just hack up a imperative deployment script?