NixOps User's Guide

Version 1.7

Eelco Dolstra

LogicBlox

Rob Vermaas

LogicBlox

Table of Contents

1. Introduction
2. Installation
3. Overview
    3.1. Deploying a VirtualBox VM
        3.1.1. Deploying multiple machines
    3.2. Deploying to a NixOS machine
    3.3. Deploying to Amazon EC2
    3.4. Deploying to Google Compute Engine
    3.5. Deploying to Microsoft Azure
        3.5.1. Prequisites
        3.5.2. Default resources
        3.5.3. Default subresources
        3.5.4. Backups
        3.5.5. Storage resources and key management
        3.5.6. Managing virtual machines without public IP addresses
        3.5.7. Resource names and IDs
        3.5.8. Azure limitations and quirks to be aware of:
    3.6. Deploying to Hetzner physical machines
    3.7. Deploying to Digital Ocean
    3.8. Deploying to Libvirtd (Qemu)
    3.9. Deploying Datadog resources
    3.10. Accessing machines
    3.11. Checking machine status
    3.12. Network special attributes
    3.13. Network arguments
    3.14. Managing keys
    3.15. Special NixOS module inputs
A. Command Reference
    nixops — deploy a set of NixOS machines
B. Configuration Options
    B.1. Machines
    B.2. AWS Resources
        B.2.1. EBS Volumes
        B.2.2. SQS Queues
        B.2.3. SNS Topics
        B.2.4. EC2 Keypairs
        B.2.5. EC2 Security Groups
        B.2.6. Elastic IPs
        B.2.7. S3 Buckets

B.2.8. IAM Roles
B.2.9. SSH Keypairs
B.2.10. CloudWatch Log Groups
B.2.11. CloudWatch Log Streams
B.2.12. Elastic File Systems
B.2.13. Elastic File System Mount Target
B.2.14. Vpc
B.2.15. Vpc Subnet
B.2.16. Vpc Route
B.2.17. Vpc Route Table
B.2.18. Vpc Route Table Association
B.2.19. Vpc Network Interface
B.2.20. Vpc Network Interface Attachment
B.2.21. Vpc Network Acl
B.2.22. Vpc Nat Gateway
B.2.23. Vpc Internet Gateway
B.2.24. Vpc Endpoint
B.2.25. Vpc Egress Only Internet Gateway
B.2.26. Vpc Dhcp Options
B.2.27. Vpc Customer Gateway
B.3. GCE Resources
B.3.1. GCE Disks
B.3.2. GCE Images
B.3.3. GCE Forwarding Rules
B.3.4. GCE HTTP Health Checks
B.3.5. GCE Networks
B.3.6. GCE Static IPs
B.3.7. GCE Target Pools
B.3.8. GSE Buckets
B.4. Azure Resources
B.4.1. Azure Availability Sets
B.4.2. Azure BLOB Containers
B.4.3. Azure BLOBs
B.4.4. Azure Directories
B.4.5. Azure DNS Record Sets
B.4.6. Azure DNS Zones
B.4.7. Azure ExpressRoute Circuits
B.4.8. Azure Files
B.4.9. Azure Gateway Connections
B.4.10. Azure Load Balancers
B.4.11. Azure Local Network Gateways
B.4.12. Azure Network Security Groups
B.4.13. Azure Queues
B.4.14. Azure Reserved IP Addresses
B.4.15. Azure Resource Groups
B.4.16. Azure Shares
B.4.17. Azure Storages
B.4.18. Azure Tables
B.4.19. Azure Traffic Manager Profiles
B.4.20. Azure Virtual Networks
B.4.21. Azure Virtual Network Gateways
B.5. Datadog Resources
B.5.1. Datadog Monitor Resource
B.5.2. Datadog Timeboard Resource
B.5.3. Datadog Screenboard Resource
BK. Hacking
BL. Release Notes
BL.1. Release 1.7 (April XX, 2019)
BL.2. Release 1.6.1 (Sep 14, 2018)
BL.3. Release 1.6 (Mar 28, 2018)
BL.4. Release 1.5.2 (Oct 29, 2017)
BL.5. Release 1.5.1 (Jul 5, 2017)
BL.6. Release 1.5 (Feb 16, 2017)

BL.7. Release 1.4 (Jul 11, 2016)
BL.8. Release 1.3.1 (January 14, 2016)
BL.9. Release 1.3 (September 28, 2015)
BL.10. Release 1.2 (April 30, 2014)
BL.11. Release 1.1.1 (October 2, 2013)
BL.12. Release 1.1 (September 9, 2013)
BL.13. Release 1.0.1 (July 11, 2013)
BL.14. Release 1.0 (June 18, 2013)

List of Examples

3.1. `trivial.nix`: logical network specification
3.2. `trivial-vbox.nix`: VirtualBox physical network specification
3.3. `load-balancer.nix`: logical network specification
3.4. `load-balancer-vbox.nix`: VirtualBox physical network specification
3.5. `trivial-nixos.nix`: NixOS target physical network specification
3.6. `load-balancer-ec2.nix`: EC2 physical network specification
3.7. `load-balancer-gce.nix`: GCE physical network specification
3.8.
3.9. `load-balancer-azure.nix`: Azure physical network specification
3.10. `trivial-digital-ocean.nix`: A trivial digital ocean setup
3.11. `datadog-timeboard.nix`: Datadog timeboard specification
3.12. `key-dependency.nix`: track key dependence with systemd
C.1.

Chapter 1. Introduction

NixOps is a tool for deploying NixOS machines in a network or cloud. It takes as input a declarative specification of a set of “logical” machines and then performs any necessary steps or actions to realise that specification: instantiate cloud machines, build and download dependencies, stop and start services, and so on. NixOps has several nice properties:

- It’s *declarative*: NixOps specifications state the desired configuration of the machines, and NixOps then figures out the actions necessary to realise that configuration. So there is no difference between doing a new deployment or doing a redeployment: the resulting machine configurations will be the same.

- It performs *fully automated* deployment. This is a good thing because it ensures that deployments are reproducible.

- It performs provisioning. Based on the given deployment specification, it will start missing virtual machines, create disk volumes, and so on.

- It’s based on the Nix package manager, which has a *purely functional* model that sets it apart from other package managers. Concretely this means that multiple versions of packages can coexist on a system, that packages can be upgraded or rolled back atomically, that dependency specifications can be guaranteed to be complete, and so on.

- It’s based on NixOS, which has a declarative approach to describing the desired configuration of a machine. This makes it an ideal basis for automated configuration management of sets of machines. NixOS also has desirable properties such as (nearly) atomic upgrades, the ability to roll back to previous configurations, and more.

- It’s *multi-cloud*. Machines in a single NixOps deployment can be deployed to different target environments. For instance, one logical machine can be deployed to a local “physical” machine, another to an automatically instantiated Amazon EC2 instance in the `eu-west-1` region, another in the `us-east-1` region, and so on. NixOps arranges the necessary network configuration to ensure that these machines can communicate securely with each other (e.g. by setting up encrypted tunnels).

- It supports *separation of “logical” and “physical” aspects* of a deployment. NixOps specifications are modular, and this makes it easy to separate the parts that say *what* logical machines should do from *where* they should do it. For instance, the former might say that machine X should run a PostgreSQL database and machine Y should run an Apache web server, while the latter might state that X should be instantiated as an EC2 `m1.large` machine while Y should be instantiated as an `m1.small`. We could also have a second physical specification that says that X and Y should both be instantiated as VirtualBox VMs on the developer’s workstation. So the same logical specification can easily be deployed to different environments.

- It uses a single formalism (the Nix expression language) for package management and system configuration management. This makes it very easy to add ad hoc packages to a deployment.

- It combines system configuration management and provisioning. Provisioning affects configuration management: for instance, if we instantiate an EC2 machine as part of a larger deployment, it may be necessary to put the IP address or hostname of that machine in a configuration file on another machine. NixOps takes care of this automatically.

- It can provision non-machine cloud resources such as Amazon S3 buckets and EC2 keypairs.

This manual describes how to install NixOps and how to use it. The appendix contains a copy of the NixOps manual page, which is also available by running man nixops.

Chapter 2. Installation

NixOps runs on Linux and Mac OS X. (It may also run on other platforms; the main prerequisite is that Nix runs on your platform.) Installing it requires the following steps:

1. Install the Nix package manager. It’s available from the Nix website in binary form for several platforms. Please refer to the installation instruction in the Nix manual for more details.

2. Install the latest version of NixOps.

```
$ nix-env -i nixops
```

Chapter 3. Overview

Table of Contents

3.1. Deploying a VirtualBox VM
    3.1.1. Deploying multiple machines
3.2. Deploying to a NixOS machine
3.3. Deploying to Amazon EC2
3.4. Deploying to Google Compute Engine
3.5. Deploying to Microsoft Azure
    3.5.1. Prequisites
    3.5.2. Default resources
    3.5.3. Default subresources
    3.5.4. Backups
    3.5.5. Storage resources and key management
    3.5.6. Managing virtual machines without public IP addresses
    3.5.7. Resource names and IDs
    3.5.8. Azure limitations and quirks to be aware of:

3.6. Deploying to Hetzner physical machines
3.7. Deploying to Digital Ocean
3.8. Deploying to Libvirtd (Qemu)
3.9. Deploying Datadog resources
3.10. Accessing machines
3.11. Checking machine status
3.12. Network special attributes
3.13. Network arguments
3.14. Managing keys
3.15. Special NixOS module inputs

This chapter gives a quick overview of how to use NixOps.

3.1. Deploying a VirtualBox VM

NixOps deploys machines on the basis of a declarative description of what those machines should do, and where they should be deployed to. These descriptions are specified in the *Nix expression language* used by the Nix package manager. Example 3.1 shows a minimal specification of a network consisting of only one logical machine named `webserver`.

Example 3.1. `trivial.nix`: logical network specification

```
{
  network.description = "Web server";

  webserver =
    { config, pkgs, ... }:
    { services.httpd.enable = true;
      services.httpd.adminAddr = "alice@example.org";
      services.httpd.documentRoot = "${pkgs.valgrind.doc}/share/doc/valgrind/html";
      networking.firewall.allowedTCPPorts = [ 80 ];
    };
}
```

This specification consists of a set of top-level attributes describing logical machines (namely `webserver`) and meta-information (namely `network.description`). Each attribute not named `network` describes a logical machine. The value of each logical machine attribute is a *NixOS configuration module*, which describes the desired configuration of the corresponding machine. Thus, the logical machine `webserver` should have the Apache httpd web server running, and its document root (rather arbitrarily for demonstration purposes) should be the documentation of the Valgrind package.

To deploy this machine, we also need to provide configuration options that tell NixOps to what environment it should be deployed. Example 3.2 specifies that `webserver` should be deployed as a VirtualBox instance. Note that for this to work the `vboxnet0` network has to exist - you can add it in the VirtualBox general settings under *Networks - Host-only Networks* if necessary. If you are running NixOps in a headless environment, then you should also add the option `deployment.virtualbox.headless = true;` to the configuration. Otherwise, VirtualBox will fail when it tries to open a graphical display on the host's desktop.

Example 3.2. `trivial-vbox.nix`: VirtualBox physical network specification

```
{
  webserver =
    { config, pkgs, ... }:
    { deployment.targetEnv = "virtualbox";
      deployment.virtualbox.memorySize = 1024; # megabytes
      deployment.virtualbox.vcpu = 2; # number of cpus
    };
}
```

Before we can deploy the network we need to use the command nixops create to create a *NixOps deployment* that contains any state associated with the deployment (such as information about instantiated VMs). At creation time, we need to specify the Nix expressions that constitute the complete deployment specification. So to create a deployment for deploying the Apache web server to VirtualBox, we would do:

```
$ nixops create ./trivial.nix ./trivial-vbox.nix -d trivial
33bced96-5f26-11e1-b9d7-9630d48abec1
```

Here `-d trivial` gives the symbolic name `trivial` to the deployment. Deployments can be identified in two ways: using the UUID printed by nixops create, or using the symbolic name you specified at creation time.

You can print a list of existing deployments using nixops list:

```
+--------------------------------------+-----------+--------------+------------+------------+
|                 UUID                 |    Name   | Description  | # Machines |    Type    |
+--------------------------------------+-----------+--------------+------------+------------+
| 33bced96-5f26-11e1-b9d7-9630d48abec1 |  trivial  |  Web server  |     0      |            |
+--------------------------------------+-----------+--------------+------------+------------+
```

The command nixops info shows the current deployment state:

```
$ nixops info -d trivial
Network UUID: 33bced96-5f26-11e1-b9d7-9630d48abec1
Network description: Web server

+-----------+--------+------------+-------------+------------+
|   Name    | Status |    Type    | Resource Id | IP address |
+-----------+--------+------------+-------------+------------+
| webserver |  New   | virtualbox |             |            |
+-----------+--------+------------+-------------+------------+
```

The machine status `New` indicates that the logical machine `webserver` hasn't been created yet. The `-d` option specifies which deployment to use; you can use the symbolic name (`-d trivial`) or the UUID (`-d 33bced96-5f26-11e1-b9d7-9630d48abec1`). You can also set the the environment variable `NIXOPS_DEPLOYMENT`.

The actual deployment is done by running nixops deploy:

```
$ nixops deploy -d trivial
creating VirtualBox VM ‘webserver’...
Virtual machine 'nixops-33bced96-5f26-11e1-b9d7-9630d48abec1-webserver' is created and registered
Clone hard disk created in format 'VDI'. UUID: 5a0b0771-7e03-4fab-9c2f-e95888b57db3
Waiting for VM "nixops-33bced96-5f26-11e1-b9d7-9630d48abec1-webserver" to power on...
VM "nixops-33bced96-5f26-11e1-b9d7-9630d48abec1-webserver" has been successfully started.
waiting for IP address of ‘webserver’......................... 192.168.56.101
waiting for SSH on ‘webserver’...
building all machine configurations...
building path(s) `/nix/store/ybrny9h744q8i3x026ccfmdav8qnw7pd-nixos-version'
building path(s) `/nix/store/zxw279xhl6l8yl94gnka8aqv1kkcrrd4-os-release'
fetching path `/nix/store/pn43d3llpsm3pc1ywaxccmw8pmzjqgz0-valgrind-3.7.0'...
…
copying closure to machine ‘webserver’...
copying 376 missing paths to ‘root@192.168.56.101’...
importing path `/nix/store/jfcs9xnfbmiwqs224sb0qqsybbfl3sab-linux-headers-2.6.35.14'
…
```

```
activating new configuration on machine ‘webserver’...
updating GRUB 2 menu...
activating the configuration...
…
starting new service ‘httpd’...
```

NixOps performs the following steps to do the deployment:

- It creates missing machines. In this case, a VirtualBox instance for the logical machine `webserver` is started. NixOps then waits to obtain its IP address.
- It builds the NixOS machine configurations locally. For instance, here Valgrind is built or downloaded because our machine configuration has a dependency on it.
- It copies the closure of each machine configuration to the corresponding machine.
- It activates the configuration on each machine. For instance, it starts the `httpd` systemd service on the `webserver` machine. This is the only step that has a visible effect; all prior steps do not affect the active configuration of the machines.

The nixops info command will show that a machine was created:

```
$ nixops info -d trivial
Network UUID: 33bced96-5f26-11e1-b9d7-9630d48abec1
Network description: Web server

+-----------+--------+------------+-----------------------------------------------------+--------
|   Name    | Status |    Type    |                     Resource Id                     |   IP adc
+-----------+--------+------------+-----------------------------------------------------+--------
| webserver |   Up   | virtualbox | nixops-33bced96-5f26-11e1-b9d7-9630d48abec1-machine | 192.168.
+-----------+--------+------------+-----------------------------------------------------+--------
```

Visit `http://192.168.56.101` in a web browser should now show the Valgrind documentation. You can also log in to the virtual machine as `root`:

```
$ nixops ssh -d trivial webserver
connecting to 192.168.56.101...
[root@webserver:~]#
```

The command nixops ssh is a convenience wrapper around ssh that passes the right IP address and SSH identity for the specified logical machine. (NixOps automatically creates a unique SSH key pair for communicating with each VirtualBox instance.)

Redeployment after making a change to the specification is simply a matter of running nixops deploy again. If we do this for the example, NixOps will notice that the `webserver` machine already exists and that most or all dependencies are already present, so it won’t create a new VirtualBox instance or need to build and copy a lot of dependencies. Thus redeployment typically only takes a few seconds:

```
$ time nixops deploy -d trivial
building all machine configurations...
copying closure to machine ‘webserver’...
activating new configuration on machine ‘webserver’...
real    0m3.700s
```

If you want to get rid of the virtual machines created by NixOps, you can run nixops destroy:

```
$ nixops destroy -d trivial
warning: are you sure you want to destroy VirtualBox VM ‘webserver’? (y/N) y
webserver> destroying VirtualBox VM...
webserver> 0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
webserver> 0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
```

You can use the option `--confirm` to confirm all questions. This is useful for automated deployment, but potentially dangerous.

3.1.1. Deploying multiple machines

A network consisting of only one logical machine is not very exciting. Example 3.3 shows a network consisting of three machines: a load balancer (named `proxy`) that uses Apache's `mod_proxy` to do reverse proxying, and two backend web servers (`backend1` and `backend2`) that serve the actual content. One important thing to note is that if you want to refer to another machine (e.g. in a configuration file), you can use a hostname equal to the logical name of the machine, as in the line

```
BalancerMember http://backend1 retry=0
```

This works because NixOps generates a `/etc/hosts` file that contains entries for all the logical machines in the network, mapping names to each machine's IP address. Also note that because the two backend machines have identical configurations, we can use a let-binding to define the configuration only once.

Example 3.3. `load-balancer.nix`: logical network specification

```
let

  backend =
    { config, pkgs, ... }:
    { services.httpd.enable = true;
      services.httpd.adminAddr = "alice@example.org";
      services.httpd.documentRoot = "${pkgs.valgrind.doc}/share/doc/valgrind/html";
      networking.firewall.allowedTCPPorts = [ 80 ];
    };

in

{
  network.description = "Load balancing network";

  proxy =
    { config, pkgs, nodes, ... }:
    { services.httpd.enable = true;
      services.httpd.adminAddr = "bob@example.org";
      services.httpd.extraModules = ["proxy_balancer" "lbmethod_byrequests"];
      services.httpd.extraConfig =
        ''
          <Proxy balancer://cluster>
            Allow from all
            BalancerMember http://backend1 retry=0
            BalancerMember http://backend2 retry=0
          </Proxy>
          ProxyPass         /    balancer://cluster/
          ProxyPassReverse  /    balancer://cluster/
        '';
      networking.firewall.allowedTCPPorts = [ 80 ];
    };

  backend1 = backend;
  backend2 = backend;
}
```

To deploy it, we need a physical specification, shown in Example 3.4. Deployment is as follows:

```
$ nixops create ./load-balancer.nix ./load-balancer-vbox.nix -d load-balancer-vbox
$ nixops deploy -d load-balancer-vbox
```

Note that NixOps creates and deploys the VMs in parallel to speed things up.

Example 3.4. `load-balancer-vbox.nix`: VirtualBox physical network specification

```
let
  vbox = { deployment.targetEnv = "virtualbox"; };
in
{ proxy    = vbox;
  backend1 = vbox;
  backend2 = vbox;
}
```

3.2. Deploying to a NixOS machine

To deploy to a machine that is already running NixOS, simply set `deployment.targetHost` to the IP address or host name of the machine, and leave `deployment.targetEnv` undefined. See Example 3.5.

Example 3.5. `trivial-nixos.nix`: NixOS target physical network specification

```
{
  webserver =
    { config, pkgs, ... }:
    { deployment.targetHost = "1.2.3.4";
    };
}
```

3.3. Deploying to Amazon EC2

Example 3.6 shows a physical specification that deploys the load balancer network to Amazon's Elastic Compute Cloud (EC2). It states that the three machines need to be instantiated in EC2 region `eu-west-1`. It also specifies a non-machine cloud resource: namely, the EC2 key pair to be used to access the machine via SSH. (It is possible to use manually created EC2 key pairs, but it's easier to let NixOps provision them.)

Example 3.6. `load-balancer-ec2.nix`: EC2 physical network specification

```
let

  region = "eu-west-1";
  accessKeyId = "dev"; # symbolic name looked up in ~/.ec2-keys or a ~/.aws/credentials profile n

  ec2 =
    { resources, ... }:
    { deployment.targetEnv = "ec2";
      deployment.ec2.accessKeyId = accessKeyId;
      deployment.ec2.region = region;
      deployment.ec2.instanceType = "m1.small";
      deployment.ec2.keyPair = resources.ec2KeyPairs.my-key-pair;
    };

in
```

```
{ proxy    = ec2;
  backend1 = ec2;
  backend2 = ec2;

  # Provision an EC2 key pair.
  resources.ec2KeyPairs.my-key-pair =
    { inherit region accessKeyId; };
}
```

Deployment is as follows:

```
$ nixops create ./load-balancer.nix ./load-balancer-ec2.nix -d load-balancer-ec2
$ nixops deploy -d load-balancer-ec2
my-key-pair> uploading EC2 key pair ‘charon-8e50b4b5-d7f9-11e2-b91c-23f8eaf468f4-my-key-pair’...
backend1...> creating EC2 instance (AMI ‘ami-8badbdff’, type ‘m1.small’, region ‘eu-west-1’)...
backend2...> creating EC2 instance (AMI ‘ami-8badbdff’, type ‘m1.small’, region ‘eu-west-1’)...
proxy......> creating EC2 instance (AMI ‘ami-8badbdff’, type ‘m1.small’, region ‘eu-west-1’)...
backend2...> waiting for IP address...
...
proxy......> activation finished successfully
backend2...> activation finished successfully
backend1...> activation finished successfully
```

Here NixOps has created an EC2 key pair and started three EBS-backed instances running the default NixOS AMI. Other than that, deployment is the same as for VirtualBox: NixOps builds the machine configurations, copies their closure over to the EC2 instances, and activates the new configurations.

The command nixops info shows all provisioned resources, not just machines:

```
$ nixops info -d load-balancer-ec2
...
+-------------+-----------------+----------------------------+-------------------------------------
| Name        |      Status     | Type                       | Resource Id
+-------------+-----------------+----------------------------+-------------------------------------
| backend1    | Up / Up-to-date | ec2 [eu-west-1a; m1.small] | i-0ec4bc43
| backend2    | Up / Up-to-date | ec2 [eu-west-1a; m1.small] | i-0cc4bc41
| proxy       | Up / Up-to-date | ec2 [eu-west-1a; m1.small] | i-08c4bc45
| my-key-pair | Up / Up-to-date | ec2-keypair [eu-west-1]    | charon-8e50b4b5-d7f9-11e2-b91c-23f8
+-------------+-----------------+----------------------------+-------------------------------------
```

The resources can be destroyed by running:

```
$ nixops destroy -d load-balancer-ec2
```

This terminates the EC2 instances and deletes the EC2 key pair.

Deployment to EC2 has some prerequisites.

- Obviously, you need an EC2 account.
- You need to add your AWS access key ID and secret key to the file `~/.ec2-keys`, as follows:

  ```
  AKIABOGUSACCESSKEY BOGUSSECRETACCESSKEY dev # my AWS development account
  ```

  Here `dev` is a symbolic name for the AWS account, which you can use in `deployment.ec2.accessKeyId`.

  Also you can use a standard way of storing credentials in a `~/.aws/credentials`:

```
[dev]
aws_access_key_id = AKIABOGUSACCESSKEY
aws_secret_access_key = BOGUSSECRETACCESSKEY
```

Profile name `dev` is the same as a previously mentioned symbolic name which you can set in `deployment.ec2.accessKeyId`. It is also possible to use an alternative credentials file by setting the `AWS_SHARED_CREDENTIALS_FILE` environment variable.

Alternatively, you can set the environment variables `EC2_ACCESS_KEY` and `EC2_SECRET_KEY`.

- If you want to use an SSH key pair created with the ec2-create-keypair command line tool or the AWS web interface, set `deployment.ec2.keyPair` to the name of the key pair, and set `deployment.ec2.privateKey` to the path of the private key:

  ```
      deployment.ec2.keyPair = "your-key-name";
      deployment.ec2.privateKey = "/path/to/your-key-name.pem";
  ```

  You can leave out `deployment.ec2.privateKey` option in case the key is findable by SSH through its normal mechanisms (e.g. it is listed in `~/.ssh/config` or was added to the ssh-agent)

- You need to ensure that your EC2 security groups are set up to allow (at the very least) SSH traffic from your network. By default, NixOps uses the security group `default`. You can set the option `deployment.ec2.securityGroups` to use other security groups:

  ```
  deployment.ec2.securityGroups = [ "allow-ssh" "allow-http" ];
  ```

- You need to set `deployment.ec2.region` to the EC2 region you want to deploy to. Note that key pairs and security groups are region-specific.

3.4. Deploying to Google Compute Engine

Example 3.7 shows a physical specification that deploys the load balancer network to Google Compute Engine(GCE). It states that the three machines need to be instantiated in GCE region `europe-west1-b`, based on the unstable branch of NixOS. It also specifies an alternative load balancer implemented using GCE Forwarding Rule.

Example 3.7. `load-balancer-gce.nix`: GCE physical network specification

```
let

  # change this as necessary or wipe and use ENV vars
  credentials = {
    project = "myproject";
    serviceAccount = "000000000000-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx@developer.gserviceaccount.cor
    accessKey = "/path/to/user/key.pem";
  };

  gce = { resources, ...}:  {
    networking.firewall.allowedTCPPorts = [ 80 ];
    deployment.targetEnv = "gce";
    deployment.gce = credentials // {
      region = "europe-west1-b";
      tags = [ "public-http" ];
      network = resources.gceNetworks.lb-net;
    };
  };

in {
```

```
  # create a network that allows SSH traffic(by default), pings
  # and HTTP traffic for machines tagged "public-http"
  resources.gceNetworks.lb-net = credentials // {
    addressRange = "192.168.4.0/24";
    firewall = {
      allow-http = {
        targetTags = [ "public-http" ];
        allowed.tcp = [ 80 ];
      };
      allow-ping.allowed.icmp = null;
    };
  };

  # by default, health check pings port 80, so we don't have to set anything
  resources.gceHTTPHealthChecks.plain-hc = credentials;

  resources.gceTargetPools.backends = { resources, nodes, ...}: credentials // {
    region = "europe-west1";
    healthCheck = resources.gceHTTPHealthChecks.plain-hc;
    machines = with nodes; [ backend1 backend2 ];
  };

  resources.gceForwardingRules.lb = { resources, ...}: credentials // {
    protocol = "TCP";
    region = "europe-west1";
    portRange = "80";
    targetPool = resources.gceTargetPools.backends;
    description = "Alternative HTTP Load Balancer";
  };

  proxy    = gce;
  backend1 = gce;
  backend2 = gce;

}
```

Deployment is as follows:

```
$ nixops create ./load-balancer.nix ./load-balancer-gce.nix -d load-balancer-gce
$ nixops deploy -d load-balancer-gce
bootstrap> creating GCE image 'n-588718b8099211e49d39b8e8560f8b58-bootstrap'...
lb-net..> Creating GCE network 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-lb-net'...
plain-hc> creating GCE HTTP health check 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-plain-hc'...
backends> creating GCE target pool 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-backends'...
lb-net..> Creating GCE firewall 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-lb-net-allow-ssh'...
lb-net..> Creating GCE firewall 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-lb-net-allow-ping'...
backends> updating the machine list of GCE target pool 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b5
lb-net..> Creating GCE firewall 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-lb-net-allow-http'...
proxy....> Creating GCE disk of auto GiB from image 'n-588718b8099211e49d39b8e8560f8b58-bootstrap'
backend1.> Creating GCE disk of auto GiB from image 'n-588718b8099211e49d39b8e8560f8b58-bootstrap'
backend2.> Creating GCE disk of auto GiB from image 'n-588718b8099211e49d39b8e8560f8b58-bootstrap'
lb......> creating GCE forwarding rule 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-lb'...done.
lb......> got IP: 146.148.16.5
backend2> creating GCE machine 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-backend2'...
proxy...> creating GCE machine 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-proxy'...
backend1> creating GCE machine 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-backend1'...
backend1> got IP: 130.211.95.195
backend2> got IP: 146.148.2.203
proxy...> got IP: 146.148.20.120
backend1> attaching GCE disk 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-backend1-root'...
backend1> waiting for SSH....
backend2> attaching GCE disk 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-backend2-root'...
backend2> waiting for SSH...
backend1> .
```

```
proxy...> attaching GCE disk 'nixops-588718b8-0992-11e4-9d39-b8e8560f8b58-proxy-root'...
...
proxy......> activation finished successfully
backend2...> activation finished successfully
backend1...> activation finished successfully
```

Here NixOps has created a GCE network, a health check, a load balancer, a bootstrap image based on the unstable branch of NixOS, 3 root disks for the instances and started three instances running the default NixOS image. Other than that, deployment is the same as for VirtualBox: NixOps builds the machine configurations, copies their closure over to the GCE instances, and activates the new configurations.

The command nixops info shows all provisioned resources, not just machines:

```
$ nixops info -d load-balancer-gce
...
+-----------+-----------------+------------------------------------+------------------------------
| Name      |      Status     | Type                               | Resource Id
+-----------+-----------------+------------------------------------+------------------------------
| backend1  | Up / Up-to-date | gce [europe-west1-b; g1-small]     | n-588718b8099211e49d39b8e856
| backend2  | Up / Up-to-date | gce [europe-west1-b; g1-small]     | n-588718b8099211e49d39b8e856
| proxy     | Up / Up-to-date | gce [europe-west1-b; g1-small]     | n-588718b8099211e49d39b8e856
| lb        | Up / Up-to-date | gce-forwarding-rule [europe-west1] | n-588718b8099211e49d39b8e856
| plain-hc  | Up / Up-to-date | gce-http-health-check [:80/]       | n-588718b8099211e49d39b8e856
| bootstrap | Up / Up-to-date | gce-image                          | n-588718b8099211e49d39b8e856
| lb-net    | Up / Up-to-date | gce-network [192.168.4.0/24]       | n-588718b8099211e49d39b8e856
| backends  | Up / Up-to-date | gce-target-pool [europe-west1]     | n-588718b8099211e49d39b8e856
+-----------+-----------------+------------------------------------+------------------------------
```

The resources can be destroyed by running:

```
$ nixops destroy -d load-balancer-gce
```

This terminates the GCE instances and deletes the alternative GCE-based load balancer.

Deployment to GCE has some prerequisites.

- Obviously, you need an GCE service account which you can create from the Developer Console.
- Once you've created a new GCE service account and downloaded the generated private key (in the PKCS12 format), you'll need to convert the key to PEM format by running the following command:

  ```
  $ openssl pkcs12 -in pkey.pkcs12 -passin pass:notasecret -nodes -nocerts | openssl rsa -out
  ```
- All GCE resources and instances must belong to a GCE project which you can create from the Developer Console. Alternatively, you could use a project you already have. Several deployments can coexist in a single project and with manually-created resources, as long as you don't exceed the quotas.
- You must ensure that the GCE service account you've created has sufficient permissions to manage resources in the project.
- You must supply the credentials(project, service account name and path to the key) via either `*.project`, `*.serviceAccount` and `*.accessKey` options or `GCE_PROJECT`, `GCE_SERVICE_ACCOUNT` and `ACCESS_KEY_PATH` environment variables. Options take precedence over environment variables and are per-resource/-instance.
- You need to ensure that GCE firewall is configured correctly. The `default` GCE network which is

created for each project and to which all instances belong by default, only allows SSH and internal traffic. Usually, this is not enough and you want to create a network managed by NixOps with custom firewall settings. By default, the NixOps-managed networks allow SSH traffic because it is absolutely required to manage the instances. In addition to allowing traffic based on IP and port ranges, firewall can also selectively enable traffic for instances with specific tags, such as `public-http` in the example, which is assigned to the instances you want to receive connections on port 80.

- Many resources are region- and zone-specific, and thus you need to set `*.region` options where applicable.

GCE limitations and quirks to be aware of.

- A bootstrap image needs to be created for each deployment because it is impossible to create public images. Default bootstrap image specification can be overriden by defining `resources.gceImages.bootstrap`. Additionally, the instance's `bootstrapImage` option can be used to specify an instance-specific bootstrap image.

  A solution is to create one's own image, by running the script in `<nixpkgs/nixos/maintainers/scripts/gce/create-gce.sh>`. Two things need to be done before running the script: <itemizedList> <listitem>Set the `BUCKET_NAME` environment variable to the target Google Storage bucket's name</listitem>. <listitem>Add permissions to that bucket for the "Compute Engine" service account (under the Google Cloud console, IAM & Administration, IAM)</listitem> </itemizedList> Then, add the corresponding resources (i.e. at the top level of the nixops deployment):

  Example 3.8.

  ```
  resources.gceImages.my-bootstrap = {
    name = "my-bootstrap";
    project = "…";
    serviceAccount = "…";
    accessKey = "…";
    sourceUri = "gs://my-bucket/nixos-image-18.03.git.fa98773-x86_64-linux.raw.tar.gz";
  };
  ```

- There's no "native" support for starting and stopping instances. NixOps emulates starting and stoping by creating and tearing down GCE instances, but preserving the disk contents.

  While this mostly just works, GCE ends up charging you a minimum of uptime (which was 10 minutes at the moment of writing this manual) thus too frequent start/stop cycling ends up expensive.

  Start/stop cycling of an instance which uses an ephemeral IP address often causes the IP address to change, which breaks certain features such as encrypted tunnels until repaired by `deploy`.

  Another important difference is that NixOps attempts to replicate the last known state of the instance(attached disks, tags). Thus, if the state was modified manually (e.g. via gcloud tool), such changes are lost in a start/stop cycle.

  Consider rebooting instead which doesn't have these limitations and, in addition, is faster.

- Creation, modification and deletion of resources and instances are not idempotent in GCE.

In practice, this means that if you hit Ctrl+C or an error happens, while NixOps is creating, destroying or otherwise changing the state of a resource, the state of the resource expected by NixOps and the actual state may diverge.

Usually, this doesn't cause too much trouble, but a good practice is to follow each failed or aborted deployment operation with a `deploy --check` run to detect and fix any state mismatch(es).

- The instances which are members of target pools need a constantly-running `configure-forwarding-rules` service, which is enabled by default, and is not otherwise required. Substantial RAM savings for a large deployment can be obtained by disabling the service if it isn't needed.

Migration of resources between zones and putting previously-existing resources under NixOps control.

- Disks can be migrated by making a snapshot and then initializing a new NixOps-managed disk from it, possibly, in another zone or region.

- Migrating an instance to another zone via backup functionality is currently impossible. It is still possible to create a new instance and migrate each disk by hand using snapshots.

- Putting a manually-created static IP resource under NixOps management is done this way: create a resource to temporarily hold the IP address, such as an instance or a forwarding rule; delete the static IP resource, which still leaves the IP address itself under your control thanks to the holding resource; create a new static IP address `with resources.gceStaticIPs.$NAME.ipAddress` set to the IP address of the holding resource; delete the holding resource after checking that the static IP resource has been correctly created and holds the original IP address. *You must practice the migration procedure on a test static IP resource.*

  If by accident or after ignoring the above advice, you lose control of a valuable IP address, you must act very fast and attempt to create a new static IP resource with `with resources.gceStaticIPs.$NAME.ipAddress` set to the IP address itself that you want to regain control over. If you are late and the IP address has been given to someone else, it still makes sense to repeately try reserving the address because most likely it is in use as an emphemeral one and thus will become available soon. Needless to say, you want to avoid a situation like this at all costs.

  IP addresses are region-specific and thus most likely can't be migrated to another region. It is impossible to migrate an IP address to another project without temporarily losing control over it.

3.5. Deploying to Microsoft Azure

> ***Warning*** *The Azure backend in Nixops is now disabled. See PR#1131*
>
> *For existing deployments, Azure backend is supported in Nixops up to release 1.6.1 only.*

Note: only ARM(Azure Resource Manager) mode is supported by this backend.

Example 3.9 shows a physical specification that deploys the load balancer network to Azure along with the absolute minimum of accessory resources that need to be created to be able to deploy virtual machines. It states that the three machines need to be instantiated in azure location `West US`. It also specifies an alternative load balancer implemented using a native Azure Load Balancer resource.

Example 3.9. `load-balancer-azure.nix`: Azure physical network specification

```
let

  # change this as necessary or delete and use ENV vars
  credentials = {
    subscriptionId = "00000000-0000-0000-0000-000000000000";
    authority = "https://login.windows.net/AUTHORITY.onmicrosoft.com";
    user = "user@AUTHORITY.onmicrosoft.com";
    password = "**********";
  };

  azure = { backendAddressPools ? [] }: { resources, ...}:  {
    deployment.targetEnv = "azure";
    deployment.azure = credentials // {
      location = "westus";
      size = "Standard_A0"; # minimal size that supports load balancing
      availabilitySet = resources.azureAvailabilitySets.set;
      networkInterfaces.default.backendAddressPools = backendAddressPools;
    };
  };

  azure_backend = {resources, ...}@args:
    azure { backendAddressPools = [{loadBalancer = resources.azureLoadBalancers.lb;}]; } args;

in {

  resources.azureReservedIPAddresses.lb-ip = credentials // {
    location = "West US";
  };

  resources.azureAvailabilitySets.set = credentials // {
    location = "westus";
  };

  resources.azureLoadBalancers.lb = {resources,...}: credentials // {
    location = "westus";
    frontendInterfaces.default.publicIpAddress = resources.azureReservedIPAddresses.lb-ip;
    loadBalancingRules.web = {
      frontendPort = 80;
      backendPort = 80;
    };
  };

  proxy    = azure {};
  backend1 = azure_backend;
  backend2 = azure_backend;

}
```

The deployment proceeds like this:

```
$ nixops create ./load-balancer.nix ./load-balancer-azure.nix -d load-balancer-azure
$ nixops deploy -d load-balancer-azure
...
def-group....................> creating Azure resource group 'nixops-71616e2e-c165-11e5-b910-b8e85
dn-westus....................> creating Azure virtual network 'nixops-71616e2e-c165-11e5-b910-b8e8
set..........................> creating Azure availability set 'nixops-71616e2e-c165-11e5-b910-b8e
lb-ip........................> creating Azure reserved IP address 'nixops-71616e2e-c165-11e5-b910-
def-storage-westus...........> creating Azure storage '71616e2ec165westus' in westus...
lb-ip........................> reserved IP address: 40.78.67.191
lb...........................> creating Azure load balancer 'nixops-71616e2e-c165-11e5-b910-b8e85
def-storage-westus...........> waiting for the storage to settle; this may take several minutes...
def-storage-westus...........> updating BLOB service properties of Azure storage '71616e2ec165west
def-storage-westus...........> updating queue service properties of Azure storage '71616e2ec165wes
def-storage-westus...........> updating table service properties of Azure storage '71616e2ec165wes
```

```
def-storage-westus-vhds......> creating Azure BLOB container 'nixops-71616e2e-c165-11e5-b910-b8e85
def-storage-westus-vhds-image> creating Azure BLOB 'nixops-71616e2e-c165-11e5-b910-b8e8560f8b58-un
def-storage-westus-vhds-image> updating properties of Azure BLOB 'nixops-71616e2e-c165-11e5-b910-b
backend2.....................> getting an IP address
proxy........................> getting an IP address
backend1.....................> getting an IP address
backend2.....................> creating a network interface
backend1.....................> creating a network interface
proxy........................> creating a network interface
backend1.....................> creating Azure machine 'nixops-71616e2e-c165-11e5-b910-b8e8560f8b58
backend2.....................> creating Azure machine 'nixops-71616e2e-c165-11e5-b910-b8e8560f8b58
proxy........................> creating Azure machine 'nixops-71616e2e-c165-11e5-b910-b8e8560f8b58
...
proxy......> activation finished successfully
backend2...> activation finished successfully
backend1...> activation finished successfully
```

Here NixOps has created a resource group, storage, container for blobs, root image blob, availability set, load balancer and started three instances running the default NixOS image. Other than that, deployment is the same as for VirtualBox: NixOps builds the machine configurations, copies their closure over to the Azure instances, and activates the new configurations.

The command nixops info shows all provisioned resources, not just machines:

```
$ nixops info -d load-balancer-azure
...
+-------------------------------+----------------+--------------------------------------+---------
| Name                          |     Status     | Type                                 | Resource
+-------------------------------+----------------+--------------------------------------+---------
| backend1                      | Up / Up-to-date | azure [westus; Standard_A0]         | nixops-7
| backend2                      | Up / Up-to-date | azure [westus; Standard_A0]         | nixops-7
| proxy                         | Up / Up-to-date | azure [westus; Standard_A0]         | nixops-7
| set                           | Up / Up-to-date | azure-availability-set [westus]     | nixops-7
| def-storage-westus-vhds-image | Up / Up-to-date | azure-blob                          | nixops-7
| def-storage-westus-vhds       | Up / Up-to-date | azure-blob-container                | nixops-7
| lb                            | Up / Up-to-date | azure-load-balancer [westus]        | nixops-7
| lb-ip                         | Up / Up-to-date | azure-reserved-ip-address [West US] | nixops-7
| def-group                     | Up / Up-to-date | azure-resource-group [westus]       | nixops-7
| def-storage-westus            | Up / Up-to-date | azure-storage [westus]              | 71616e2e
| dn-westus                     | Up / Up-to-date | azure-virtual-network [westus]      | nixops-7
+-------------------------------+----------------+--------------------------------------+---------
```

Opening http://40.78.60.145, http://40.78.58.17, http://40.78.59.32, or http://40.78.67.191 in a web browser should now show the Nixos homepage. Also, you can log into any of the machines as `root`:

```
$ nixops ssh -d load-balancer-azure backend1
connecting to 40.78.60.145...
[root@backend1:~]#
```

The resources can be destroyed by running:

```
$ nixops destroy -d load-balancer-azure
```

This terminates the Azure instances and deletes the alternative native load balancer.

3.5.1. Prequisites

- You need Azure credentials to authenticate requests. The authentication methods supported is using Azure Active Directory's application ID and key. You need to ensure your Azure account has an Active Directory, and add a application to it.

To create Active Directory's application guides: 1. Open "Cloud Shell". (The ">_" icon in right up.) 2. Input those. (See this link.

```
        $ az ad sp create-for-rbac
```

3. Go to "Azure Active Directory" (from left panel) 4. (Manage section) "App registrations" 5. Select created application. 6. Application ID can be get. (nixops appId) 7. "Keys" 8. Add key name, expiration period and click "save", copy the generate key as "nixops appKey"

- You must supply the credentials(subscription ID, authority URL, application ID, application Key) to your deployments via either `*.subscriptionId`, `*.authority`, `*.appId` and `*.appKey` options or `AZURE_SUBSCRIPTION_ID`, `AZURE_AUTHORITY_URL`, `AZURE_ACTIVE_DIR_APP_ID` and `AZURE_ACTIVE_DIR_APP_KEY` environment variables. Options take precedence over environment variables and are specified per resource/machine.

  Example credentials for application ID/application key authentication:

  ```
  credentials = {
    subscriptionId = "00000000-0000-0000-0000-000000000000";
    authority = "https://login.windows.net/YOURDIRECTORYNAME.onmicrosoft.com";
    appId = "44444444-4444-4444-4444-444444444444";
    appKey = "********************";
  };
  ```

  To get Service Principal credentials: 1. Go to "Azure Active Directory" (from left panel) 2. (Manage section) "App registrations" 3. "New application registration" 4. Type application name (needed later), and random SignOnURL or RedirectURL 5. Application ID is your Service Principal ID (nixops servicePrincipal) 6. Click on your application 7. "Keys" 8. Add key name, expiration period and click "save", copy the generate key as "nixops password" 9. Go to "Subscriptions" (from left panel) 10. Select your subscription 11. Select "Access control (IAM)" 12. "Add" 13. Select "role" (permissions that nixops needs) 14. Next type application name you just created 15. "Save" 16. Profit?

  Authority URL can also be specified as `https://login.windows.net/TENANT_ID`.

- You need to ensure that SSH ports of all machines are reachable either directly via machines' public IP addresses or via NAT rules on the public IP address of a load balancer.

3.5.2. Default resources

If a virtual machine specification omits `resourceGroup`, `storage`, `ephemeralDiskContainer`, `networkInterfaces.default.subnet.network` or `rootDiskImageBlob`, NixOps will automatically generate "default" resources. You can see them using nixops info command. This substantially reduces the boilerplate code for simple deployments without affecting the complex ones.

There's only one default resource group. Default storage accounts and networks are created in each datacenter location where they are needed.

Disk containers are created in each storage account that is used by a virtual machine with `ephemeralDiskContainer` left empty.

Root image BLOBs are created in each disk container that is used by a virtual machine with `rootDiskImageBlob` left empty.

The default root disk image BLOB resources can be set to mirror your custom image instead of the default NixOps-provided one using:

```
$ nixops set-args -d load-balancer-azure --argstr azure-image-url "http://mystorage.windows.net/im

$ nixops info -d load-balancer-azure
...
Nix arguments: azure-image-url = "http://mystorage.windows.net/images/nixos-custom.vhd"
...

$ nixops set-args -d load-balancer-azure --unset azure-image-url
```

3.5.3. Default subresources

Several Azure resources can have multiple subresources. For example, network can have several subnets. In each such case, a default subresource is created unless specified otherwise and referencing a subresource via its "parent" references the subresource named "default".

One such resource type is virtual network with its subnetworks. A virtual network specification

```
resources.azureVirtualNetworks.network = credentials // {
  location = "West EU";
};
```

is identical to

```
resources.azureVirtualNetworks.network = credentials // {
  location = "West EU";
  subnets.default = { ... };
};
```

and when referencing the default subnet

```
networkInterfaces.default.subnet.network = resources.azureVirtualNetworks.network;
```

is identical to:

```
networkInterfaces.default.subnet.network = resources.azureVirtualNetworks.network;
networkInterfaces.default.subnet.name = "default";
```

Another example is load balancer backend address pools. A load balancer has a default backend address pool:

```
backendAddressPools = [ "default" ];
```

and a virtual machine can join the default pool with

```
  networkInterfaces.default.backendAddressPools =
      [{loadBalancer = resources.azureLoadBalancers.lb;}];
```

instead of

```
  networkInterfaces.default.backendAddressPools =
      [{loadBalancer = resources.azureLoadBalancers.lb; name = "default"; }];
```

3.5.4. Backups

Backups are implemented as BLOB snapshots. Deleting a BLOB, also deletes all of its backups.

Backups are tracked by BLOB URLs and not disk names, so if an ephemeral disk changes its `mediaLink` property, it will be treated as a different/new disk for backup purposes. Renaming a disk, but keeping `mediaLink` property unchanged preserves backups.

3.5.5. Storage resources and key management

Each storage account has two access keys, any of which can be used to authenticate operations. The keys are automatically generated when a storage account is created and can be independently regenerated at any time using `azure-cli`. Running `deploy --check` on the storage account fetches the updated key(s).

`activeKey` property specifies which of the keys NixOps should use to authenticate storage operations. This allows you to regenerate the inactive key, and then switch to using it, providing for seamless key replacement.

All storage resources(containers, BLOBs, queues etc) allow you to explicitly specify the access key, but this is only useful if the storage account is not managed by NixOps. If the storage account is managed by NixOps, all you need is to specify the parent resource (storage account for containers, container for BLOBs etc) and NixOps will infer the storage account and active key automatically.

3.5.6. Managing virtual machines without public IP addresses

If a virtual machine doesn't have a public IP address (has ip.obtain set to false), NixOps is unable to reach the SSH port of the machine and manage it. However, if you route the SSH port of the machine via an inboud NAT rule to a load balancer frontend interface that has a public IP address, NixOps will automatically detect and use this to manage the machine.

In this example, NixOps will access the machine via `lb-ip:2201` :

```
resources.azureLoadBalancers.lb = {resources,...}: credentials // {
  location = "westus";
  frontendInterfaces.default.publicIpAddress = resources.azureReservedIPAddresses.lb-ip;
  inboundNatRules.machine3-ssh = {
    frontendPort = 2201;
    backendPort = 22;
  };
  inboundNatRules.machine4-ssh = {
    frontendPort = 2202;
    backendPort = 22;
  };
};

machine3.deployment.azure = {
  networkInterfaces.default ={
    ip.obtain = false;
    inboundNatRules = [{loadBalancer = resources.azureLoadBalancers.lb; name = "machine3-ssh";}];
  };
};
```

3.5.7. Resource names and IDs

The best way to specify a reference to a resource that is managed by NixOps is via `resources.azure*.resourceName`. However, if you need to reference a resource not managed by NixOps, you can do so by resource name or ID.

If the property description says "The name or resource of..." such as for resource groups and storages, the reference is by name:

```
  resources.azureResourceGroups.group = credentials // {
    name = "my-test-group";
    location = "West US";
  };

  resources.azureAvailabilitySets.set1 = {resources,...}: credentials // {
    resourceGroup = resources.azureResourceGroups.group;
    location = "West US";
  };

  resources.azureAvailabilitySets.set2 = credentials // {
    resourceGroup = "my-test-group";
    location = "West US";
  };
```

If the property description says "The Azure Resource Id or NixOps resource...", the reference is by full Azure resource ID:

```
resources.azureVirtualNetworks.network = credentials // {
  name = "test-network";
  location = "West US";
  addressSpace = [ "10.1.0.0/16" "10.4.0.0/16" ];
  subnets = {
    default.addressPrefix = "10.1.11.0/24";
    GatewaySubnet.addressPrefix = "10.1.10.0/24";
  };
};

resources.azureVirtualNetworkGateways.gateway1 = {resources,...}: credentials // {
  location = "West US";
  gatewayType = "RouteBased";
  subnet.network = resources.azureVirtualNetworks.network;
  subnet.name = "GatewaySubnet";
};

resources.azureVirtualNetworkGateways.gateway2 = credentials // {
  location = "West US";
  gatewayType = "RouteBased";
  subnet.network = "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups" +
                   "/nixops-00000000-0000-0000-0000-000000000000-def-group/providers" +
                   "/Microsoft.Network/virtualNetworks/test-network";
  subnet.name = "GatewaySubnet";
};
```

3.5.8. Azure limitations and quirks to be aware of:

- You can replace the root disk of a VM, but root disks of different VMs aren't interchangeable because Azure only allows provisioning info to be supplied when creating a new root disk.
- BLOB URLs must use HTTPS protocol.
- BLOB MD5 hash reported by Azure is not reliable. It is just another piece of user-provided metadata and can't be used to check BLOB contents.
- BLOB type cannot be changed via copy operation. If a BLOB is created by copying another BLOB(using `copyFromBlob` property), and `blobType` property value doesn't match the type of the source BLOB, `deploy --check` will keep complaining until you change `blobType` property value to match the source BLOB type.
- Virtual machines that are members of a load balancer backend pool must belong to an availability set, and must belong to the same set.
- Drivers of older linux versions don't automatically detect removed disks and rescan has to be

triggered manually. While NixOps tries to do it for you, if it fails, you can run `sg_scan /dev/disk/by-lun/X` to drop the removed disk.

- Removal of a disk which is currently in use by dm-mapper(eg via cryptsetup), creates a rather broken state: device node is not completely released, reattaching the disk doesn't fix anything. This can be fixed with a reboot. You can trigger such a state for example if your current working dir is within the disk being unmounted, which prevents umount from actually releasing the underlying dm-mapper disk, which prevents dm-mapper from releasing the Azure disk device.

- Creation, modification and deletion of resources and instances are not idempotent in Azure.

  In practice, this means that if you hit Ctrl+C or an error happens while NixOps is creating, destroying or otherwise changing the state of a resource, the state of the resource expected by NixOps and the actual state may diverge. Usually, this doesn't cause too much trouble, but a good practice is to follow each failed or aborted deployment operation with a `deploy --check` run to detect and fix any state mismatch(es).

- Resource creation and deletion operations take time to settle during which the resource is in a transient state. You are most likely to encounter this if you abort resource creation and run `deploy --check`, which will silently wait for the resource to settle.

  In certain circumstances you may encounter "resource failed to settle" error, which means that waiting for the resource to settle timed out. This shouldn't happen often as all operations have sensible timeouts. You are most likely to hit this if an aborted creation of a resource is followed by `deploy --check` which has a small timeout or if Azure gets unusually slow due to maintenance events.

- Sometimes deploying or updating a resource doesn't result in an error and instead the resource enters a "Failed" state. `deploy --check` complains when it encounters failed resources. Depending on the cause, redeploying the resource or deploying it with known good or fixed parameters resolves this problem.

- You may get "Failed getting access" authentication error when deleting a resource even if you have specified correct credentials. This happens because NixOps copies the credentials to the internal state database only on nixops deploy and the state database currently stores outdated credentials for example because you were asked to change your password by Azure. This can be resolved by a nixops deploy run.

- Azure resource names are case-insensitive and must only be unique within their container(resource group, share etc), while NixOps resource names are case-sensitive and global. NixOps has a check for resource uniqueness which usually catches the naming clashes.

- If you are getting `socket.gaierror: [Errno -2] Name or service not known` when dealing with storage services(BLOBs, containers etc), you need to flush DNS cache or wait a little bit. Azure storage API calls are issued using the subdomain name which exists only if storage exists. Thus, running `deploy --check` for a container before its storage is created causes a DNS resolution failure to be cached for some time even after you create the storage.

- If a storage is deleted, containers and BLOBs can no longer authenticate and can't differentiate between a network failure and missing storage, so such a situation is not handled automatically by NixOps to avoid causing damage. Getting out of this ambiguous state requires either (re-)deployment of the storage or manual deletion of the affected NixOps resources.

- Queues, Tables, BLOB Containers and Shares disappear instantly on deletion, but it takes some time for deletes to settle. You will get an error if you try re-creating such a resource too soon after deletion. Usually, storage resources settle within several seconds.

- `deploy --check` has no way to retrieve container ACL, so be careful with manual changes.

3.6. Deploying to Hetzner physical machines

In order to deploy to Hetzner machines, you need to have a valid account to their server management interface, called the Robot. This account is *only* used for the initial deployment and the destruction of a machine. In particular the initial deployment creates a separate Robot sub-account (Hetzner calls this the "Admin login" because you'd give it to your server's sysadmin) just for the machine that's going to be created, so a person who has access to your deployment will only have access to the machines within the deployment and not *all* machines that are associated with your main Robot account. When destroying a machine, the separate admin account is removed as well.

When you have 2-factor authentication enabled for your main Robot account, NixOps cannot create sub-accounts for you because the Hetzner API doesn't support 2-factor auth (as of writing). In that case you have to create the sub-accounts manually in the Robot UI, set `deployment.hetzner.createSubAccount` to `false`, and tell NixOps about each machine's sub-account credentials as described below.

Of course you need machines where you can deploy to, which can only be ordered by the Robot's web interface. In the expression of the NixOps network, you reference these machines by setting `deployment.hetzner.mainIPv4` to the corresponding main IP address, to be found in the list of the `Server` tab in the Robot.

Partitioning of a machine is currently done by using Anaconda's Kickstart format. By default, it consists of two disks with two swap partitions, one on each disk and one big ext4 array with RAID1, similiar to the default layout Hetzner is using for installing their Debian machines. If you want to change the default, you can use `deployment.hetzner.partitions` to change the default layout. For example to install a machine with btrfs:

```
{
  example = {
    deployment.targetEnv = "hetzner";
    deployment.hetzner.mainIPv4 = "1.2.3.4";
    deployment.hetzner.partitions = ''
      clearpart --all --initlabel --drives=sda,sdb

      part swap1 --recommended --label=swap1 --fstype=swap --ondisk=sda
      part swap2 --recommended --label=swap2 --fstype=swap --ondisk=sdb

      part btrfs.1 --grow --ondisk=sda
      part btrfs.2 --grow --ondisk=sdb

      btrfs / --data=1 --metadata=1 --label=root btrfs.1 btrfs.2
    '';
  };
}
```

This will install NixOS on a machine with the main IP *`1.2.3.4`*, using a swap partition for each drive and use everything else for a single btrfs volume.

In the previous example, there is no occurrence of `deployment.hetzner.robotUser` and `deployment.hetzner.robotPass`, you can set the credentials to your main Robot account (or each machine's sub-account account, if `deployment.hetzner.createSubAccount` is `false`) there. However it is recommended to use the environment variables `HETZNER_ROBOT_USER` and `HETZNER_ROBOT_PASS`, as you only need them for initial deployment and destruction. If `deployment.hetzner.createSubAccount` is `false`, you can't use `HETZNER_ROBOT_USER` because each machine will have a different user name, but you can still use `HETZNER_ROBOT_PASS`.

3.7. Deploying to Digital Ocean

Example 3.10 shows how to run a `512m` digital ocean instance in the `ams2` region, with IPv6 support enabled. We only support droplet creation and destruction at the moment. This example assumes you have the `DIGITAL_OCEAN_AUTH_TOKEN` set with an authentication token, obtained from the Digital Ocean console. The token can also be provided via the `deployment.digitalOcean.authToken` option.

Note that we rely on a ssh key resource with the hard-coded name `ssh-key`. Providing your own key is not supported yet.

Example 3.10. `trivial-digital-ocean.nix`: A trivial digital ocean setup

```
{
  resources.sshKeyPairs.ssh-key = {};

  machine = { config, pkgs, ... }: {
    services.nginx.enable = true;
    services.openssh.enable = true;

    deployment.targetEnv = "digitalOcean";
    deployment.digitalOcean.enableIpv6 = true;
    deployment.digitalOcean.region = "ams2";
    deployment.digitalOcean.size = "512mb";
  };
}
```

To install we first start a Ubuntu instance, and then overwrite it with NixOS via a modified version of nixos-infect . `nixos-infect` itself uses the undocumented `NIXOS_LUSTRATE` under the hood.

3.8. Deploying to Libvirtd (Qemu)

In order to use libvirtd backend, a couple of manual steps need to be taken. Libvirtd backend is currently supported only on NixOS.

Configure your host NixOS machine to enable libvirtd daemon, add your user to libvirtd group and change firewall not to filter DHCP packets.

```
  virtualisation.libvirtd.enable = true;
  users.extraUsers.myuser.extraGroups = [ "libvirtd" ];
  networking.firewall.checkReversePath = false;
```

Next we have to make sure our user has access to create images by executing:

```
  $ sudo mkdir /var/lib/libvirt/images
  $ sudo chgrp libvirtd /var/lib/libvirt/images
  $ sudo chmod g+w /var/lib/libvirt/images
```

We're ready to create the deployment, start by creating `example.nix`:

```
{
  example = { config, pkgs, lib, ... }: {
  };
}
```

and libvirtd specification `example-libvirtd.nix`:

```
{
  example = {
    deployment.targetEnv = "libvirtd";
  };
}
```

Finally, let's deploy it with NixOps:

```
  $ nixops create -d example-libvirtd ./example.nix ./example-libvirtd.nix
  $ nixops deploy -d example-libvirtd
```

Graphics Display and Console *It's possible to connect a VNC viewer to the guest to see the graphics display (X11) or the framebuffer console.*

To do this, ensure the `deployment.libvirtd.headless` option is set to `false` (the default). Then use the `virsh vncdisplay` command to get a VNC connection string to pass to your VNC viewer.

Serial console: *If you want to access the serial console of the guest (`virsh console`) we also need the following:*

```

boot.kernelParams = [ "console=ttyS0,115200" ];
deployment.libvirtd.extraDevicesXML = ''
  <serial type='pty'>
    <target port='0'/>
  </serial>
  <console type='pty'>
    <target type='serial' port='0'/>
  </console>
'';
```

Tip *In order to log in you have to set a (root) password.*

3.9. Deploying Datadog resources

NixOps allows deploying Datadog resources (monitors, timeboards, screenboards) using a declarative description. Before deploying Datadog resources, you need to generate an api_key and app_key from here. The following is a minimal specification of a datadog resource deployment which takes a host as an argument

Warning *Note that if you don't specify the api_key/app_key options, they will be defaulted to the environment variables DATADOG_API_KEY and DATADOG_APP_KEY.*

Example 3.11. `datadog-timeboard.nix`: Datadog timeboard specification

```
  { host
   , ...
  }:
  let
    app_key = "...";
    api_key = "...";
  in
  {
    resources.datadogTimeboards.host-timeboard = { config, ...}:
    {
      appKey = app_key;
      apiKey = api_key;
      title = "Timeboard created using NixOps";
      description = "Timeboard created using NixOps";
      templateVariables = [
        {
          name = "host";
          prefix = "host";
          default = "${host}";
        }
      ];
      graphs = [
        {
          title = "system.disk.free";
          definition = builtins.toJSON {
            requests= [
              {
                type= "line";
                conditional_formats= [];
                aggregator= "avg";
                q= "avg:system.disk.free{device:/dev/dm-0,host:${host}}";
              }
            ];
            viz= "timeseries";
          };
        }
      ];
    };
  }
```

In this example, the graph definition is a JSON string, which can be customized by following the JSON graphing documentation. This is similar for the monitor options and screenboard widgets which are defined using a JSON string as well.

Once deployed, the deployment specification would be:

```
$ nixops deploy -d timeboards
host-timeboard> creating datadog timeboard 'Timeboard created using NixOps...'
building all machine configurations...
timeboards> closures copied successfully
timeboards> deployment finished successfully

$ nixops info -d timeboards
...
Nix arguments: host = "myhost"

+----------------+-----------------+-------------------+------------------------------------------
| Name           |      Status     | Type              | Resource Id
+----------------+-----------------+-------------------+------------------------------------------
| host-timeboard | Up / Up-to-date | datadog-timeboard | Timeboard created using NixOps [ /dash/26
+----------------+-----------------+-------------------+------------------------------------------
```

3.10. Accessing machines

We have seen above that you can login to individual machines by doing `nixops ssh` *`name`*, where *`name`* is the name of the machine.

It's also possible to perform a command on all machines:

```
$ nixops ssh-for-each -d load-balancer-ec2 -- df /tmp
backend1...> /dev/xvdb      153899044 192084 145889336   1% /tmp
proxy......> /dev/xvdb      153899044 192084 145889336   1% /tmp
backend2...> /dev/xvdb      153899044 192084 145889336   1% /tmp
```

By default, the command is executed sequentially on each machine. You can add the flag `-p` to execute it in parallel.

3.11. Checking machine status

The command nixops check checks the status of each machine in a deployment. It verifies that the machine still exists (i.e. hasn't been destroyed outside of NixOps), is up (i.e. the instance has been started) and is reachable via SSH. It also checks that any attached disks (such as EBS volumes) are not in a failed state, and prints the names of any systemd units that are in a failed state.

For example, for the 3-machine EC2 network shown above, it might show:

```
$ nixops check -d load-balancer-ec2
+----------+--------+-----+-----------+----------+----------------+---------------+-------+
| Name     | Exists | Up  | Reachable | Disks OK | Load avg.      | Failed units  | Notes |
+----------+--------+-----+-----------+----------+----------------+---------------+-------+
| backend1 | Yes    | Yes | Yes       | Yes      | 0.03 0.03 0.05 | httpd.service |       |
| backend2 | Yes    | No  | N/A       | N/A      |                |               |       |
| proxy    | Yes    | Yes | Yes       | Yes      | 0.00 0.01 0.05 |               |       |
+----------+--------+-----+-----------+----------+----------------+---------------+-------+
```

This indicates that Apache httpd has failed on `backend1` and that machine `backend2` is not running at all. In this situation, you should run nixops deploy --check to repair the deployment.

3.12. Network special attributes

It is possible to define special options for the whole network. For example:

```
{
  network = {
    description = "staging environment";
    enableRollback = true;
  };

  defaults = {
    imports = [ ./common.nix ];
  };

  machine = { ... }: {};
}
```

Each attribute is explained below:

`defaults`
  Applies given NixOS module to all machines defined in the network.

`network.description`
  A sentence describing the purpose of the network for easier comparison when running nixops list

`network.enableRollback`

If `true`, each deployment creates a new profile generation to able to run nixops rollback. Defaults to `false`.

3.13. Network arguments

In NixOps you can pass in arguments from outside the nix expression. The network file can be a nix function, which takes a set of arguments which are passed in externally and can be used to change configuration values, or even to generate a variable number of machines in the network.

Here is an example of a network with network arguments:

```
{ maintenance ? false
}:
{
  machine =
    { config, pkgs, ... }:
    { services.httpd.enable = maintenance;
      ...
    };
}
```

This network has a *maintenance* argument that defaults to `false`. This value can be used inside the network expression to set NixOS option, in this case whether or not Apache HTTPD should be enabled on the system.

You can pass network arguments using the `set-args` nixops command. For example, if we want to set the `maintenance` argument to `true` in the previous example, you can run:

```
  $ nixops set-args --arg maintenance true -d <name>
```

The arguments that have been set will show up:

```
$ nixops info -d argtest
Network name: argtest
Network UUID: 634d6273-f9f6-11e2-a004-15393537e5ff
Network description: Unnamed NixOps network
Nix expressions: .../network-arguments.nix
Nix arguments: maintenance = true

+---------+---------------+------+-------------+------------+
| Name    |     Status    | Type | Resource Id | IP address |
+---------+---------------+------+-------------+------------+
| machine | Missing / New | none |             |            |
+---------+---------------+------+-------------+------------+
```

Running `nixops deploy` after changing the arguments will deploy the new configuration.

3.14. Managing keys

Files in `/nix/store/` are readable by every user on that host, so storing secret keys embedded in nix derivations is insecure. To address this, nixops provides the configuration option `deployment.keys`, which nixops manages separately from the main configuration derivation for each machine.

Add a key to a machine like so.

```
{
  machine =
    { config, pkgs, ... }:
    {
      deployment.keys.my-secret.text = "shhh this is a secret";
      deployment.keys.my-secret.user = "myuser";
      deployment.keys.my-secret.group = "wheel";
      deployment.keys.my-secret.permissions = "0640";
    };
}
```

This will create a file `/run/keys/my-secret` with the specified contents, ownership, and permissions.

Among the key options, only `text` is required. The `user` and `group` options both default to `"root"`, and `permissions` defaults to `"0600"`.

Keys from `deployment.keys` are stored under `/run/` on a temporary filesystem and will not persist across a reboot. To send a rebooted machine its keys, use nixops send-keys. Note that all nixops commands implicitly upload keys when appropriate, so manually sending keys should only be necessary after an unattended reboot.

If you have a custom service that depends on a key from `deployment.keys`, you can opt to let systemd track that dependency. Each key gets a corresponding systemd service `"${keyname}-key.service"` which is active while the key is present, and otherwise inactive when the key is absent. See Example 3.12 for how to set this up.

Example 3.12. `key-dependency.nix`: track key dependence with systemd

```
{
  machine =
    { config, pkgs, ... }:
    {
      deployment.keys.my-secret.text = "shhh this is a secret";

      systemd.services.my-service = {
        after = [ "my-secret-key.service" ];
        wants = [ "my-secret-key.service" ];
        script = ''
          export MY_SECRET=$(cat /run/keys/my-secret)
          run-my-program
        '';
      };
    };
}
```

These dependencies will ensure that the service is only started when the keys it requires are present. For example, after a reboot, the services will be delayed until the keys are available, and systemctl status and friends will lead you to the cause.

3.15. Special NixOS module inputs

In deployments with multiple machines, it is often convenient to access the configuration of another node in the same network, e.g. if you want to store a port number only once.

This is possible by using the extra NixOS module input `nodes`.

```
{
  network.description = "Gollum server and reverse proxy";

  gollum =
    { config, pkgs, ... }:
    {
      services.gollum = {
        enable = true;
        port = 40273;
      };
      networking.firewall.allowedTCPPorts = [ config.services.gollum.port ];
    };

  reverseproxy =
    { config, pkgs, nodes, ... }:
    let
      gollumPort = nodes.gollum.config.services.gollum.port;
    in
    {
      services.nginx = {
        enable = true;
        virtualHosts."wiki.example.net".locations."/" = {
          proxyPass = "http://gollum:${toString gollumPort}";
        };
      };
      networking.firewall.allowedTCPPorts = [ 80 ];
    };
}
```

Moving the port number to a different value is now without the risk of an inconsistent deployment.

Aditional module inputs are

- `name`: The name of the machine.
- `uuid`: The NixOps UUID of the deployment.
- `resources`: NixOps resources associated with the deployment.

Appendix A. Command Reference

Table of Contents

nixops — deploy a set of NixOS machines

Name

nixops — deploy a set of NixOS machines

Synopsis

`nixops` { `--version` | `--help` | *`command`* [*`arguments`*...] } [{ `--state` | `-s` } *`statefile`*] [{ `--deployment` | `-d` } *`uuid-or-name`*] [`--confirm`] [`--debug`]

Description

NixOps is a tool for deploying NixOS machines in a network or cloud.

Common options

`--state`, `-s`
: Path to the state file that contains the deployments. It defaults to the value of the `NIXOPS_STATE` environment variable, or `~/.nixops/deployments.nixops` if that one is not defined. It must have extension `.nixops`. The state file is actually a SQLite database that can be inspected using the sqlite3 command (for example, `sqlite3 deployments.nixops .dump`). If it does not exist, it is created automatically.

`--deployment`, `-d`
: UUID or symbolic name of the deployment on which to operate. Defaults to the value of the `NIXOPS_DEPLOYMENT` environment variable.

`--confirm`
: Automatically confirm “dangerous” actions, such as terminating EC2 instances or deleting EBS volumes. Without this option, you will be asked to confirm each dangerous action interactively.

`--debug`
: Turn on debugging output. In particular, this causes NixOps to print a Python stack trace if an unhandled exception occurs.

`--help`
: Print a brief summary of NixOps’s command line syntax.

`--version`
: Print NixOps’s version number.

Common options passed along to Nix

`-I`
: Append a directory to the Nix search path.

`--max-jobs`
: Set maximum number of concurrent Nix builds.

`--cores`
: Sets the value of the NIX_BUILD_CORES environment variable in the invocation of builders

`--keep-going`
: Keep going after failed builds.

`--keep-failed`
: Keep temporary directories of failed builds.

`--show-trace`
: Print a Nix stack trace if evaluation fails.

`--fallback`
: Fall back on installation from source.

`--option`
: Set a Nix option.

`--read-only-mode`
: Run Nix evaluations in read-only mode.

Environment variables

NIXOPS_STATE
The location of the state file if `--state` is not used. It defaults to `~/.nixops/deployments.nixops`.

NIXOPS_DEPLOYMENT
UUID or symbolic name of the deployment on which to operate. Can be overridden using the `-d` option.

EC2_ACCESS_KEY, AWS_ACCESS_KEY_ID
AWS Access Key ID used to communicate with the Amazon EC2 cloud. Used if `deployment.ec2.accessKeyId` is not set in an EC2 machine's configuration.

EC2_SECRET_KEY, AWS_SECRET_ACCESS_KEY
AWS Secret Access Key used to communicate with the Amazon EC2 cloud. It is only used if no secret key corresponding to the AWS Access Key ID is defined in `~/.ec2-keys` or `~/.aws/credentials`.

AWS_SHARED_CREDENTIALS_FILE
Alternative path to the the shared credentials file, which is located in `~/.aws/credentials` by default.

HETZNER_ROBOT_USER, HETZNER_ROBOT_PASS
Username and password used to access the Robot for Hetzner deployments.

GCE_PROJECT
GCE Project which should own the resources in the Google Compute Engine deployment. Used if `deployment.gce.project` is not set in a GCE machine configuration and if `resources.$TYPE.$NAME.project` is not set in a GCE resource specification.

GCE_SERVICE_ACCOUNT, ACCESS_KEY_PATH
GCE Service Account ID and the path to the corresponding private key in .pem format which should be used to manage the Google Compute Engine deployment. Used if `deployment.gce.serviceAccount` and `deployment.gce.accessKey` are not set in a GCE machine configuration and if `resources.$TYPE.$NAME.serviceAccount` and `resources.$TYPE.$NAME.accessKey` are not set in a GCE resource specification.

AZURE_SUBSCRIPTION_ID, AZURE_AUTHORITY_URL, AZURE_USER, AZURE_SERVICE_PRINCIPAL, AZURE_PASSWORD
Azure subscription ID, authority URL, user, service principal and password. Used if not set in an Azure machine deployment configuration via `deployment.azure.subscriptionId`, `deployment.azure.authority`, `deployment.azure.user`, `deployment.azure.servicePrincipal` and `deployment.azure.password`, and if not set in an Azure resource specification via `resources.$TYPE.$NAME.subscriptionId`, `resources.$TYPE.$NAME.authority`, `resources.$TYPE.$NAME.user`, `resources.$TYPE.$NAME.servicePrincipal` and `resources.$TYPE.$NAME.password`.

Files

~/.ec2-keys
This file maps AWS Access Key IDs to their corresponding Secret Access Keys. Each line must consist of an Access Key IDs, a Secret Access Keys and an optional symbolic identifier, separated by whitespace. Comments starting with `#` are stripped. An example:

```
AKIABOGUSACCESSKEY BOGUSSECRETACCESSKEY dev # AWS development account
AKIABOGUSPRODACCESSKEY BOGUSPRODSECRETACCESSKEY prod # AWS production account
```

The identifier can be used instead of actual Access Key IDs in `deployment.ec2.accessKeyId`, e.g.

```
deployment.ec2.accessKeyId = "prod";
```

This is useful if you have an AWS account with multiple user accounts and you don't want to hard-code an Access Key ID in a NixOps specification.

`~/.aws/credentials`
: This file pairs AWS Access Key IDs with their corresponding Secret Access Keys under symbolic profile names. It consists of sections marked by profile names. Sections contain newline-separated "assignments" of "variables" `aws_access_key_id` and `aws_secret_access_key` to a desired Access Key ID and a Secret Access Key, respectively, e.g.:

```
[dev]
aws_access_key_id = AKIABOGUSACCESSKEY
aws_secret_access_key = BOGUSSECRETACCESSKEY

[prod]
aws_access_key_id = AKIABOGUSPRODACCESSKEY
aws_secret_access_key = BOGUSPRODSECRETACCESSKEY
```

Symbolic profile names are specified in `deployment.ec2.accessKeyId`, e.g.:

```
deployment.ec2.accessKeyId = "prod";
```

If an actual Access Key IDs is used in `deployment.ec2.accessKeyId` its corresponding Secret Access Key is looked up under `[default]` profile name. Location of credentials file can be customized by setting the `AWS_SHARED_CREDENTIALS_FILE` environment variable.

Command `nixops create`

Synopsis

`nixops create` *`nixexprs`*... [`-I` *`path`* ...]

Description

This command creates a new deployment state record in NixOps's database. The paths of the Nix expressions that specify the desired deployment (*`nixexprs`*) are stored in the state file. The UUID of the new deployment is printed on standard output.

Options

`-I` ***`path`***
: Add *`path`* to the Nix expression search path for all future evaluations of the deployment specification. NixOps stores *`path`* in the state file. This option may be given multiple times. See the description of the `-I` option in nix-instantiate(1) for details.

`--deployment`, `-d`
: Set the symbolic name of the new deployment to the given string. The name can be used to refer to the deployment by passing the option `-d` *`name`* or the environment variable `NIXOPS_DEPLOYMENT=`*`name`* to subsequent NixOps invocations. This is typically more convenient than using the deployment's UUID. However, names are not required to be unique; if you create multiple deployments with the same name, NixOps will complain.

Examples

To create a deployment with symbolic name `foo`, and then perform the actual deployment:

```
$ nixops create expr1.nix expr2.nix -d foo
created deployment ‘32b06868-d27c-11e2-a055-81d7beb7925e’

$ nixops deploy -d foo
```

Command `nixops modify`

Synopsis

`nixops modify` *nixexprs*... [{ `--name` | `-n` } *name*] [`-I` *path* ...]

Description

This command modifies an existing deployment. The options are the same as for nixops create. The symbolic name of the deployment can be changed using the `--name` flag.

Examples

To change the Nix expressions specifying the deployment, and rename it from `foo` to `bar`:

```
$ nixops modify -d foo -n bar expr3.nix expr4.nix
```

Note that `-d` identifies the existing deployment, while `-n` specifies its new name.

Command `nixops clone`

Synopsis

`nixops clone` [{ `--name` | `-n` } *name*]

Description

This command clones an existing deployment; that is, it creates a new deployment that has the same deployment specification and parameters, but a different UUID and (optionally) name. Note that nixops clone does not currently clone the state of the machines in the existing deployment. Thus, when you first run nixops deploy on the cloned deployment, NixOps will create new instances from scratch.

Examples

To create a new deployment `bar` by cloning the deployment `foo`:

```
$ nixops clone -d foo -n bar
```

Command `nixops delete`

Synopsis

`nixops delete` [`--all`] [`--force`]

Description

This command deletes a deployment from the state file. NixOps will normally refuse to delete the deployment if any resources belonging to the deployment (such as virtual machines) still exist. You must run nixops destroy first to get rid of any such resources. However, if you pass `--force`, NixOps will forget about any still-existing resources; this should be used with caution.

If the `--all` flag is given, all deployments in the state file are deleted.

Examples

To delete the deployment named `foo`:

```
$ nixops delete -d foo
```

Command `nixops deploy`

Synopsis

`nixops deploy` [`--kill-obsolete` | `-k`] [`--dry-run`] [`--repair`] [`--create-only`] [`--build-only`] [`--copy-only`] [`--check`] [`--allow-reboot`] [`--force-reboot`] [`--allow-recreate`] [`--include` *`machine-name`*...] [`--exclude` *`machine-name`*...] [`-I` *`path`* ...] [`--max-concurrent-copy` *`N`*]

Description

This command deploys a set of machines on the basis of the specification described by the Nix expressions given in the preceding nixops create call. It creates missing virtual machines, builds each machine configuration, copies the closure of each configuration to the corresponding machine, uploads any keys described in `deployment.keys`, and activates the new configuration.

Options

`--kill-obsolete`, `-k`
: Destroy (terminate) virtual machines that were previously created as part of this deployment, but are obsolete because they are no longer mentioned in the deployment specification. This happens if you remove a machine from the specification after having run nixops deploy to create it. Without this flag, such obsolete machines are left untouched.

`--dry-run`
: Dry run; show what would be done by this command without actually doing it.

`--repair`
: Use --repair when calling nix-build. This is useful for repairing the nix store when some inconsistency is found and nix-copy-closure is failing as a result. Note that this option only works in nix setups that run without the nix daemon.

`--create-only`
: Exit after creating any missing machines. Nothing is built and no existing machines are touched.

`--build-only`
: Just build the configuration locally; don’t create or deploy any machines. Note that this may fail if the configuration refers to information only known after machines have been created (such as IP addresses).

`--copy-only`
: Exit after creating missing machines, building the configuration and copying closures to the target machines; i.e., do everything except activate the new configuration.

`--check`
: Normally, NixOps assumes that the deployment state of machines doesn't change behind its back. For instance, it assumes that a VirtualBox VM, once started, will continue to run unless you run nixops destroy to terminate it. If this is not the case, e.g., because you shut down or destroyed a machine through other means, you should pass the `--check` option to tell NixOps to verify its current knowledge.

`--allow-reboot`
: Allow NixOps to reboot the instance if necessary. For instance, if you change the type of an EC2 instance, NixOps must stop, modify and restart the instance to effectuate this change.

`--force-reboot`
: Reboot the machine to activate the new configuration (using nixos-rebuild boot).

`--allow-recreate`
: Recreate resources that have disappeared (e.g. destroyed through mechanisms outside of NixOps). Without this flag, NixOps will print an error if a resource that should exist no longer does.

`--include` ***`machine-name...`***
: Only operate on the machines explicitly mentioned here, excluding other machines.

`--exclude` ***`machine-name...`***
: Only operate on the machines that are *not* mentioned here.

`-I` ***`path`***
: Add *`path`* to the Nix expression search path. This option may be given multiple times and takes precedence over the `-I` flags used in the preceding nixops create invocation. See the description of the `-I` option in nix-instantiate(1) for details.

`--max-concurrent-copy` ***`N`***
: Use at most *`N`* concurrent nix-copy-closure processes to deploy closures to the target machines. *`N`* defaults to 5.

Examples

To deploy all machines:

```
$ nixops deploy
```

To deploy only the logical machines `foo` and `bar`, checking whether their recorded deployment state is correct:

```
$ nixops deploy --check --include foo bar
```

To create any missing machines (except `foo`) without doing anything else:

```
$ nixops deploy --create-only --exclude foo
```

Command `nixops destroy`

Synopsis

`nixops destroy` [`--all`] [`--include` *`machine-name`*...] [`--exclude` *`machine-name`*...]

Description

This command destroys (terminates) all virtual machines previously created as part of this deployment, and similarly deletes all disk volumes if they're marked as "delete on termination". Unless you pass the `--confirm` option, you will be asked to approve every machine destruction.

This command has no effect on machines that cannot be destroyed automatically; for instance, machines in the `none` target environment (such as physical machines, or virtual machines not created by NixOps).

Options

`--all`
: Destroy all deployments.

`--include` *`machine-name...`*
: Only destroy the machines listed here.

`--exclude` *`machine-name...`*
: Destroy all machines except the ones listed here.

Examples

To destroy all machines:

```
$ nixops destroy
```

To destroy the machine named `foo`:

```
$ nixops destroy --include foo
```

Command `nixops stop`

Synopsis

`nixops stop` [`--include` *`machine-name`*...] [`--exclude` *`machine-name`*...]

Description

This command stops (shuts down) all non-obsolete machines that can be automatically started. This includes EC2 and VirtualBox machines, but not machines using the `none` backend (because NixOps doesn't know how to start them automatically).

Options

`--include` *`machine-name...`*
: Only stop the machines listed here.

`--exclude` *`machine-name...`*
: Stop all machines except the ones listed here.

Examples

To stop all machines that support being stopped:

```
$ nixops stop
```

Command `nixops start`

Synopsis

`nixops start` [`--include` *`machine-name`*...] [`--exclude` *`machine-name`*...]

Description

This command starts all non-obsolete machines previously stopped using nixops stop.

Options

`--include` ***`machine-name...`***
: Only start the machines listed here.

`--exclude` ***`machine-name...`***
: Start all machines except the ones listed here.

Examples

To start all machines that were previously stopped:

```
$ nixops start
```

Command `nixops list`

Synopsis

`nixops list`

Description

This command prints information about all deployments in the database: the UUID, the name, the description, the number of running or stopped machines, and the types of those machines.

Examples

```
$ nixops list
+--------------------------------------+------------------------+------------------------+--------
|                 UUID                 |          Name          |      Description       | # Machi
+--------------------------------------+------------------------+------------------------+--------
| 80dc8e11-287d-11e2-b05a-a810fd2f513f |          test          |      Test network      |     4
| 79fe0e26-d1ec-11e1-8ba3-a1d56c8a5447 |   nixos-systemd-test   | Unnamed NixOps network |     1
| 742c2a4f-0817-11e2-9889-49d70558c59e |       xorg-test        | NixOS X11 Updates Test |      0
+--------------------------------------+------------------------+------------------------+--------
```

Command nixops info

Synopsis

`nixops info` [`--all`] [`--plain`] [`--no-eval`]

Description

This command prints some information about the current state of the deployment. For each machine, it prints:

- The logical name of the machine.
- Its state, which is one of `New` (not deployed yet), `Up` (created and up to date), `Outdated` (created but not up to date with the current configuration, e.g. due to use of the `--exclude` option to nixops deploy) and `Obsolete` (created but no longer present in the configuration).
- The type of the machine (i.e. the value of `deployment.targetEnv`, such as `ec2`). For EC2 machines, it also shows the machine’s region or availability zone.
- The virtual machine identifier, if applicable. For EC2 machines, this is the instance ID. For VirtualBox VMs, it’s the virtual machine name.
- The IP address of the machine. This is its public IP address, if it has one, or its private IP address otherwise. (For instance, VirtualBox machines only have a private IP address.)

Options

`--all`
Print information about all resources in all known deployments, rather than in a specific deployment.

`--plain`
Print the information in a more easily parsed format where columns are separated by tab characters and there are no column headers.

`--no-eval`
Do not evaluate the deployment specification. Note that as a consequence the “Status” field in the output will show all machines as “Obsolete” (since the effective deployment specification is empty).

Examples

```
$ nixops info -d foo
Network name: test
Network UUID: 80dc8e11-287d-11e2-b05a-a810fd2f513f
Network description: Test network
Nix expressions: /home/alice/test-network.nix

+----------+-----------------+------------------------------+------------+-----------------+
|   Name   |      Status     |             Type             |   VM Id    |    IP address   |
+----------+-----------------+------------------------------+------------+-----------------+
| backend0 |  Up / Outdated  | ec2 [us-east-1b; m2.2xlarge] | i-905e9def |   23.23.12.249  |
| backend1 |  Up / Outdated  | ec2 [us-east-1b; m2.2xlarge] | i-925e9ded |  184.73.128.122 |
| backend2 |  Up / Obsolete  | ec2 [us-east-1b; m2.2xlarge] | i-885e9df7 | 204.236.192.216 |
| frontend | Up / Up-to-date |  ec2 [us-east-1c; m1.large]  | i-945e9deb |  23.23.161.169  |
+----------+-----------------+------------------------------+------------+-----------------+
```

Command `nixops check`

Synopsis

`nixops check` [`--all`]

Description

This command checks and prints the status of each machine in the deployment. For instance, for an EC2 machine, it will ask EC2 whether the machine is running or stopped. If a machine is supposed to be up, NixOps will try to connect to the machine via SSH and get the current load average statistics.

Options

`--all`
Check all machines in all known deployments, rather than in a specific deployment.

Examples

For a running VirtualBox instance, NixOps will print something like:

```
$ nixops status
machine> VM state is ‘running’
machine> pinging SSH... up [1.03 0.34 0.12]
```

For a stopped EC2 instance, NixOps might show:

```
machine> instance state is ‘stopped’
```

Command `nixops ssh`

Synopsis

`nixops ssh` [*`username`*@]*`machine`* [*`command`* [*`args`*...]]

Description

This command opens an SSH connection to the specified machine and executes the specified command. If no command is specified, an interactive shell is started.

Options

`--include-keys`
Include the public SSH host keys into .ssh/known_hosts for all machines in the imported network.

Examples

To start a shell on machine `foo`:

```
$ nixops ssh foo
```

To run Emacs on machine `bar`:

```
$ nixops ssh bar -- -X emacs
```

Passes `-X` (“enable X11 forwarding”) to SSH.

Command `nixops ssh-for-each`

Synopsis

`nixops ssh-for-each` [`--parallel` | `-p`] [`--include` *`machine-name`*...] [`--exclude` *`machine-name`*...] [*`command`* [*`args`*...]]

Description

This operation executes the specified shell command on all non-obsolete machines.

Options

`--parallel`
: Execute the command on each machine in parallel. The default is to do each machine sequentially.

`--include` ***`machine-name...`***
: Execute the command only on the machines listed here.

`--exclude` ***`machine-name...`***
: Execute the command on all machines except the ones listed here.

Examples

To reboot all machines in parallel:

```
$ nixops ssh-for-each -p reboot
```

Command `nixops mount`

Synopsis

`nixops mount` [{ `--option` | `-o` } *`option`* ...] [*`username`*@]*`machine`* [:[*`remote`*]] *`local`*

Description

This command mounts the directory *`remote`* in the file system of the specified machine onto the directory *`local`* in the local file system. If `:`*`remote`* is omitted, the entire remote file system is mounted. If you specify an empty path (i.e. `:`), then the home directory of the specified user is mounted. If no user is specified, `root` is assumed.

This command is implemented using sshfs, so you must have sshfs installed and the `fuse` kernel module loaded.

Options

`--option` / **`-o`** ***`opt`***
: Pass additional options to sshfs. See sshfs(1) for details.

Examples

To mount the entire file system of machine `foo` onto the local directory `~/mnt`:

```
$ nixops mount foo ~/mnt

$ ls -l ~/mnt
total 72
drwxr-xr-x 1 root  root   4096 Jan 15 11:44 bin
drwx------ 1 root  root   4096 Jan 14 17:15 boot
…
```

To mount the home directory of user `alice`:

```
$ nixops mount alice@foo: ~/mnt
```

To mount a specific directory, passing the option `transform_symlinks` to ensure that absolute symlinks in the remote file system work properly:

```
$ nixops mount foo:/data ~/mnt -o transform_symlinks
```

Command `nixops reboot`

Synopsis

`nixops reboot` [`--include` *`machine-name`*...] [`--exclude` *`machine-name`*...] [`--no-wait`] [*`command`* [*`args`*...]]

Description

This command reboots all non-obsolete machines in parallel.

Options

`--include` ***`machine-name...`***
: Only reboot the machines listed here.

`--exclude` ***`machine-name...`***
: Reboot all machines except the ones listed here.

`--no-wait`
: Do not wait until the machines have finished rebooting.

Examples

To reboot all machines except `foo` and wait until they're up again, that is, are reachable via SSH again:

```
$ nixops reboot --exclude foo
```

Command `nixops backup`

Synopsis

`nixops backup` [`--include` *`machine-name`*...] [`--exclude` *`machine-name`*...]

Description

This command makes a backup of all persistent disks of all machines. Currently this is only implemented for EC2 EBS instances/volumes.

Options

`--include` ***`machine-name...`***
: Only backup the persistent disks of the machines listed here.

`--exclude` ***`machine-name...`***
: Backup the persistent disks of all machines except the ones listed here.

Examples

To backup the persistent disks of all machines:

```
$ nixops backup
```

Command `nixops restore`

Synopsis

`nixops restore` [`--include` *`machine-name`*...] [`--exclude` *`machine-name`*...] [`--backup-id` *`backup-id`*...]

Description

This command restores a machine to a backup.

Options

`--include` ***`machine-name...`***
: Only backup the persistent disks of the machines listed here.

`--exclude` ***`machine-name...`***
: Restore the persistent disks of all machines to a given backup except the ones listed here.

`--devices` *`device-name...`*
: Restore only the persistent disks which are mapped to the specified device names.

`--backup-id`*`backup-id`*
: Restore the persistent disks of all machines to a given backup except the ones listed here.

Examples

To list the available backups and restore the persistent disks of all machines to a given backup:

```
            $ nixops backup-status
            $ nixops restore --backup-id 20120803151302
```

Restore the persistent disks at device /dev/xvdf of all machines to a given backup:

```
            $ nixops restore --devices /dev/xvdf --backup-id 20120803151302
```

Command `nixops show-option`

Synopsis

`nixops show-option` [`--xml`] *`machine`* *`option`*

Description

This command prints the value of the specified NixOS configuration option for the specified machine.

Examples

```
$ nixops show-option machine services.xserver.enable
false

$ nixops show-option --xml machine boot.initrd.availableKernelModules
<?xml version='1.0' encoding='utf-8'?>
<expr>
  <list>
    <string value="md_mod" />
    <string value="raid0" />
    …
  </list>
</expr>
```

Command `nixops set-args`

Synopsis

`nixops set-args` [`--arg` *`name`* *`value`* ...] [`--argstr` *`name`* *`value`* ...] [`--unset` *`name`* ...]

Description

This command persistently sets arguments to be passed to the deployment specification.

Options

`--arg` *`name`* *`value`*
: Set the function argument *`name`* to *`value`*, where the latter is an arbitrary Nix expression.

`--argstr` *`name`* *`value`*
: Like `--arg`, but the value is a literal string rather than a Nix expression. Thus, `--argstr name value` is equivalent to `--arg name \"value\"`.

`--unset` *`name`*
: Remove a previously set function argument.

Examples

Consider the following deployment specification (`servers.nix`):

```
{ nrMachines, active }:

with import <nixpkgs/pkgs/lib>;

let

  makeMachine = n: nameValuePair "webserver-${toString n}"
    ({ config, pkgs, ... }:
    { deployment.targetEnv = "virtualbox";
      services.httpd.enable = active;
      services.httpd.adminAddr = "foo@example.org";
    });

in listToAttrs (map makeMachine (range 1 nrMachines))
```

This specifies a network of *`nrMachines`* identical VirtualBox VMs that run the Apache web server if *`active`* is set. To create 10 machines without Apache:

```
$ nixops create servers.nix
$ nixops set-args --arg nrMachines 10 --arg active false
$ nixops deploy
```

Next we can enable Apache on the existing machines:

```
$ nixops set-args --arg active true
$ nixops deploy
```

or provision additional machines:

```
$ nixops set-args --arg nrMachines 20
$ nixops deploy
```

Command `nixops show-console-output`

Synopsis

`nixops show-console-output` *`machine`*

Description

This command prints the console output of the specified machine, if available. Currently this is only supported for the EC2 backend.

Examples

```
$ nixops show-console-output machine
Xen Minimal OS!
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Linux version 3.2.36 (nixbld@) (gcc version 4.6.3 (GCC) ) #1 SMP Fri Jan 4 16:07:14
…
```

Command `nixops export`

Synopsis

`nixops export` [`--all`]

Description

This command exports the state of the specified deployment, or all deployments if `--all` is given, as a JSON represention to standard output. The deployment(s) can be imported into another state file using nixops import.

Examples

To export a specific deployment, and import it into the state file `other.nixops`:

```
$ nixops export -d foo > foo.json
$ nixops import -s other.nixops < foo.json
added deployment ‘2bbaddca-01cb-11e2-88b2-19d91ca51c50’
```

If desired, you can then remove the deployment from the old state file:

```
$ nixops delete -d foo --force
```

To export all deployments:

```
$ nixops export --all > all.json
```

Command `nixops import`

Synopsis

`nixops import` [`--include-keys`]

Description

This command creates deployments from the state data exported by nixops export. The state is read from standard input. See nixops export for examples.

Command `nixops send-keys`

Synopsis

`nixops send-keys` [`--include` *machine-name*...] [`--exclude` *machine-name*...]

Description

This command uploads the keys described in `deployment.keys` to remote machines in the `/run/keys/` directory.

Keys are *not* persisted across reboots by default. If a machine reboot is triggered from outside `nixops`, it will need nixops send-keys to repopulate its keys.

Note that nixops deploy does an implicit send-keys where appropriate, so manually sending keys is only necessary after unattended reboots.

Options

`--include` *machine-name...*
: Only operate on the machines explicitly mentioned here, excluding other machines.

`--exclude` *machine-name...*
: Only operate on the machines that are *not* mentioned here.

Appendix B. Configuration Options

Table of Contents

B.1. Machines
B.2. AWS Resources
    B.2.1. EBS Volumes
    B.2.2. SQS Queues
    B.2.3. SNS Topics
    B.2.4. EC2 Keypairs
    B.2.5. EC2 Security Groups
    B.2.6. Elastic IPs
    B.2.7. S3 Buckets
    B.2.8. IAM Roles
    B.2.9. SSH Keypairs
    B.2.10. CloudWatch Log Groups
    B.2.11. CloudWatch Log Streams
    B.2.12. Elastic File Systems
    B.2.13. Elastic File System Mount Target
    B.2.14. Vpc
    B.2.15. Vpc Subnet
    B.2.16. Vpc Route
    B.2.17. Vpc Route Table
    B.2.18. Vpc Route Table Association
    B.2.19. Vpc Network Interface
    B.2.20. Vpc Network Interface Attachment
    B.2.21. Vpc Network Acl
    B.2.22. Vpc Nat Gateway
    B.2.23. Vpc Internet Gateway
    B.2.24. Vpc Endpoint
    B.2.25. Vpc Egress Only Internet Gateway

B.2.26. Vpc Dhcp Options
B.2.27. Vpc Customer Gateway
B.3. GCE Resources
B.3.1. GCE Disks
B.3.2. GCE Images
B.3.3. GCE Forwarding Rules
B.3.4. GCE HTTP Health Checks
B.3.5. GCE Networks
B.3.6. GCE Static IPs
B.3.7. GCE Target Pools
B.3.8. GSE Buckets
B.4. Azure Resources
B.4.1. Azure Availability Sets
B.4.2. Azure BLOB Containers
B.4.3. Azure BLOBs
B.4.4. Azure Directories
B.4.5. Azure DNS Record Sets
B.4.6. Azure DNS Zones
B.4.7. Azure ExpressRoute Circuits
B.4.8. Azure Files
B.4.9. Azure Gateway Connections
B.4.10. Azure Load Balancers
B.4.11. Azure Local Network Gateways
B.4.12. Azure Network Security Groups
B.4.13. Azure Queues
B.4.14. Azure Reserved IP Addresses
B.4.15. Azure Resource Groups
B.4.16. Azure Shares
B.4.17. Azure Storages
B.4.18. Azure Tables
B.4.19. Azure Traffic Manager Profiles
B.4.20. Azure Virtual Networks
B.4.21. Azure Virtual Network Gateways
B.5. Datadog Resources
B.5.1. Datadog Monitor Resource
B.5.2. Datadog Timeboard Resource
B.5.3. Datadog Screenboard Resource

B.1. Machines

NixOps adds several options to the NixOS machine configuration system. For the standard NixOS configuration options, please see the NixOS manual or the configuration.nix(5) man page.

Appendix C. Configuration Options

`deployment.alwaysActivate`

Always run the activation script, no matter whether the configuration has changed (the default). This behaviour can be enforced even if it's set to `false` using the command line option `--always-activate` on deployment.

If this is set to `false`, activation is done only if the new system profile doesn't match the previous one.

Type: boolean

Default: `true`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/options.nix`

`deployment.autoLuks`

The LUKS volumes to be created. The name of each attribute set specifies the name of the LUKS volume; thus, the resulting device will be named `/dev/mapper/`*`name`*.

Type: attribute set of submodules

Default: `{ }`

Example: `{ secretdisk = { device = "/dev/xvdf"; passphrase = "foobar"; } ; }`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/auto-luks.nix`

`deployment.autoLuks.<name>.autoFormat`

If the underlying device does not currently contain a filesystem (as determined by blkid, then automatically initialise it using cryptsetup luksFormat.

Type: boolean

Default: `false`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/auto-luks.nix`

`deployment.autoLuks.<name>.cipher`

The cipher used to encrypt the volume.

Type: string

Default: `"aes-cbc-essiv:sha256"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/auto-luks.nix`

`deployment.autoLuks.<name>.device`

The underlying (encrypted) device.

Type: string

Example: `"/dev/xvdg"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/auto-luks.nix`

`deployment.autoLuks.<name>.keySize`

The size in bits of the encryption key.

Type: signed integer

Default: `128`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/auto-luks.nix

`deployment.autoLuks.<name>.passphrase`

The passphrase (key file) used to decrypt the key to access the volume. If left empty, a passphrase is generated automatically; this passphrase is lost when you destroy the machine or underlying device, unless you copy it from NixOps's state file. Note that unless `deployment.storeKeysOnMachine` is set to `false`, the passphrase is stored in the Nix store of the instance, so an attacker who gains access to the disk containing the store can subsequently decrypt the encrypted volume.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/auto-luks.nix

`deployment.autoRaid0`

The RAID-0 volumes to be created. The name of each attribute set specifies the name of both the volume group and the logical volume; thus, the resulting device will be named `/dev/name/name`.

Type: attribute set of submodules

Default: `{ }`

Example: `{ bigdisk = { devices = [ "/dev/xvdg" "/dev/xvdh" ] ; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/auto-raid0.nix

`deployment.autoRaid0.<name>.devices`

The underlying devices to be combined into a RAID-0 volume.

Type: list of strings

Example: `[ "/dev/xvdg" "/dev/xvdh" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/auto-raid0.nix

`deployment.azure.appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.availabilitySet`

The Azure Resource Id or NixOps resource of the Azure availability set to place the machine into. Azure Virtual Machines specified in the same availability set are allocated to different hardware nodes to maximize availability.

Type: null or string or resource of type ‘azure-availability-set’

Default: `null`

Example: `"resources.azureVirtualNetworks.myset"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.blockDeviceMapping`

Block device mapping.

Type: attribute set of submodules

Default: `{ }`

Example: `{ /dev/disk/by-lun/1 = { mediaLink = "http://mystorage.blob.core.windows.net/mycontainer/machine-disk"; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.blockDeviceMapping.<name>.cipher`

The cipher used to encrypt the disk.

Type: string

Default: `"aes-cbc-essiv:sha256"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.blockDeviceMapping.<name>.encrypt`

Whether the Azure disk should be encrypted using LUKS.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.blockDeviceMapping.<name>.hostCaching`

Specifies the platform caching behavior of data disk blob for read/write efficiency. The default vault is None. Possible values are: None, ReadOnly, ReadWrite.

Type: one of "None", "ReadOnly", "ReadWrite"

Default: `"None"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.blockDeviceMapping.<name>.isEphemeral`

Whether the disk is ephemeral. Emphemeral disk BLOBs are automatically created and destroyed by NixOps as needed. The user has an option to keep the BLOB with contents after the virtual machine is destroyed. Ephemeral disk names need to be unique only among the other ephemeral disks of the virtual machine.

Type: boolean

Default: `true`

Example: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.blockDeviceMapping.<name>.keySize`

The size of the encryption key.

Type: signed integer

Default: `128`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.blockDeviceMapping.<name>.lun`

Logical Unit Number (LUN) location for the data disk in the virtual machine. Required if the disk is created via fileSystems.X.azure attrset. The disk will appear as /dev/disk/by-lun/*. Must be unique. Valid values are: 0-31. LUN value must be less than the maximum number of allowed disks for the virtual machine size.

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.blockDeviceMapping.<name>.mediaLink`

The location of the BLOB in the Azure BLOB store to store the ephemeral disk contents. The BLOB location must belong to a storage account in the same subscription as the virtual machine. If the BLOB doesn't exist, it will be created.

Type: null or string

Default: `null`

Example: `"http://mystorage.blob.core.windows.net/mycontainer/machine-disk"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.blockDeviceMapping.<name>.name`

The short name of the disk to create.

Type: null or string

Default: `null`

Example: `"data"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.blockDeviceMapping.<name>.passphrase`

The passphrase (key file) used to decrypt the key to access the device. If left empty, a passphrase is generated automatically; this passphrase is lost when you destroy the machine or remove the volume, unless you copy it from NixOps's state file. Note that the passphrase is stored in the Nix store of the instance, so an attacker who gains access to the Azure disk or instance store that contains the Nix store can subsequently decrypt the encrypted volume.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.blockDeviceMapping.<name>.size`

Volume size (in gigabytes) for automatically created Azure disks. This option value is ignored if the disk BLOB already exists.

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.ephemeralDiskContainer`

Azure BLOB container name or resource in which to create the ephemeral disks that don't specify mediaLink explicitly.

Type: string or resource of type ‘azure-blob-container’

Example: `"resources.azureBlobContainers.container"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.location`

The Azure data center location where the virtual machine should be created.

Type: string

Example: `"westus"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.machineName`

The Azure machine `Name`.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"custom-machine-name"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.backendAddressPools`

List of Azure load balancer backend address pools to join.

Type: list of submodules

Default: `[ ]`

Example: `[ { loadBalancer = "resources.azureLoadBalancers.mybalancer"; name = "website"; } ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.backendAddressPools.*.loadBalancer`

The Azure Resource Id or NixOps resource of the Azure load balancer to attach the interface to.

Type: string or resource of type ‘azure-load-balancer’

Example: `"resources.azureLoadBalancers.mybalancer"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.backendAddressPools.*.name`

The name of the Azure load balancer Backend Address Pool to join.

Type: unspecified

Default: `"default"`

Example: `"website"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.inboundNatRules`

List of Azure load balancer inbound NAT rules to use.

Type: list of submodules

Default: `[ ]`

Example: `[ { loadBalancer = "resources.azureLoadBalancers.mybalancer"; name = "admin-machine-ssh"; } ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.inboundNatRules.*.loadBalancer`

The Azure Resource Id or NixOps resource of the Azure load balancer to attach the interface to.

Type: string or resource of type ‘azure-load-balancer’

Example: `"resources.azureLoadBalancers.mybalancer"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.inboundNatRules.*.name`

The name of the Azure load balancer Inbound NAT Rule to use.

Type: unspecified

Example: `"admin-machine-ssh"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.ip.allocationMethod`

Dynamically-allocated IP address changes if the associated VM is deallocated, deleted, re-created, stopped and may change in certain other circumstances. Statically-allocated IP address stays the same regardless of what happens to the VM, but is billed for regardless of whether the VM is active and usable.

Type: one of "Dynamic", "Static"

Default: `"Dynamic"`

Example: `"Static"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.ip.domainNameLabel`

The concatenation of the domain name label and the regionalized DNS zone make up the fully qualified domain name associated with the public IP address. If a domain name label is specified, an A DNS record is created for the public IP in the Microsoft Azure DNS system. Example FQDN: mylabel.northus.cloudapp.azure.com.

Type: null or string

Default: `null`

Example: `"mylabel"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.ip.obtain`

Whether to obtain a dedicated public IP for the interface.

Type: boolean

Default: `true`

Example: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.ip.resource`

The Azure Resource Id or NixOps resource of an Azure reserved IP address resource to use for the network interface. To use a reserved IP, you must set ip.obtain to false.

Type: null or string or resource of type ‘azure-reserved-ip-address’

Default: `null`

Example: `"my-reserved-ip"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.securityGroup`

The Azure Resource Id or NixOps resource of the Azure network security group to associate to the interface.

Type: null or string or resource of type ‘azure-network-security-group’

Default: `null`

Example: `"resources.azureSecurityGroups.my-security-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.subnet.name`

Azure virtual subnetwork name to attach the network interface to.

Type: string

Default: `"default"`

Example: `"my-subnet"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.networkInterfaces.default.subnet.network`

The Azure Resource Id or NixOps resource of the Azure virtual network to attach the network interface to.

Type: string or resource of type ‘azure-virtual-network’

Example: `"resources.azureVirtualNetworks.mynetwork"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.resourceGroup`

Azure resource group name or resource to create the machine in.

Type: string or resource of type ‘azure-resource-group’

Example: `"resources.azureResourceGroups.mygroup"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.rootDiskImageBlob`

Bootstrap image BLOB URL, name or resource. Must reside on the same storage as VM disks.

Type: string or resource of type ‘azure-blob’

Example: `"nresources.azureBlobs.image-blob"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.size`

The size of the virtual machine to allocate.

Type: string

Default: `"Basic_A0"`

Example: `"Standard_A0"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.storage`

Azure storage service name or resource to use to manage the disk BLOBs.

Type: string or resource of type ‘azure-storage’

Example: `"resources.azureStorages.mystorage"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.azure.usePrivateIpAddress`

If instance is in a subnet/VPC whether to use the private IP address for ssh connections to this host. Defaults to false due to networkInterfaces.default.ip.obtain defaulting to true.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`deployment.container.host`

The NixOS machine on which this container is to be instantiated.

Type: string or a machine

Default: `"localhost"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/container.nix

`deployment.digitalOcean.authToken`

The API auth token. We're checking the environment for `DIGITAL_OCEAN_AUTH_TOKEN` first and if that is not set we try this auth token.

Type: string

Default: `""`

Example: `"8b2f4e96af3997853bfd4cd8998958eab871d9614e35d63fab45a5ddf981c4da"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/digital-ocean.nix

`deployment.digitalOcean.enableIpv6`

Whether to enable IPv6 support on the droplet.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/digital-ocean.nix

`deployment.digitalOcean.region`

The region. See https://status.digitalocean.com/ for a list of regions.

Type: string

Default: `""`

Example: `"nyc3"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/digital-ocean.nix

`deployment.digitalOcean.size`

The size identifier between `512mb` and `64gb`. The supported size IDs for a region can be queried via API: https://developers.digitalocean.com/documentation/v2/#list-all-sizes

Type: string

Example: `"512mb"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/digital-ocean.nix

`deployment.ec2.accessKeyId`

The AWS Access Key ID. If left empty, it defaults to the contents of the environment variables

`EC2_ACCESS_KEY` or `AWS_ACCESS_KEY_ID` (in that order). The corresponding Secret Access Key is not specified in the deployment model, but looked up in the file `~/.ec2-keys`, which should specify, on each line, an Access Key ID followed by the corresponding Secret Access Key. If the lookup was unsuccessful it is continued in the standard AWS tools `~/.aws/credentials` file. If it does not appear in these files, the environment variables `EC2_SECRET_KEY` or `AWS_SECRET_ACCESS_KEY` are used.

Type: string

Default: `""`

Example: `"AKIABOGUSACCESSKEY"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.ami`

EC2 identifier of the AMI disk image used in the virtual machine. This must be a NixOS image providing SSH access.

Type: string

Example: `"ami-00000000"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.associatePublicIpAddress`

If instance in a subnet/VPC, whether to associate a public IP address with the instance.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.blockDeviceMapping`

Block device mapping.

`/dev/sd[a-e]` or `/dev/xvd[a-e]` must be ephemeral devices.

With the following instances, EBS volumes are exposed as NVMe block devices: C5, C5d, i3.metal, M5, and M5d (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/device_naming.html). For these instances volumes should be attached as `/dev/nvme[1-26]n1`, there should be no hole in numbering.

Example C.1.

{ machine = { deployment.ec2.blockDeviceMapping."/dev/nvme1n1".size = 1; deployment.ec2.blockDeviceMapping."/dev/nvme3n1".size = 1; # this device will be attached as /dev/nvme2n1, you should use /dev/nvme2n1 }; }

Type: attribute set of submodules

Default: `{ }`

Example: `{ /dev/xvdb = { disk = "ephemeral0"; } ; /dev/xvdg = { disk = "vol-00000000"; } ; }`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`deployment.ec2.blockDeviceMapping.<name>.cipher`
The cipher used to encrypt the disk.

Type: string

Default: `"aes-cbc-essiv:sha256"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`deployment.ec2.blockDeviceMapping.<name>.deleteOnTermination`
For automatically created EBS volumes, determines whether the volume should be deleted on instance termination.

Type: boolean

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`deployment.ec2.blockDeviceMapping.<name>.disk`
EC2 identifier of the disk to be mounted. This can be an ephemeral disk (e.g. `ephemeral0`), a snapshot ID (e.g. `snap-00000000`) or a volume ID (e.g. `vol-00000000`). Leave empty to create an EBS volume automatically. It can also be an EBS resource (e.g. `resources.ebsVolumes.big-disk`).

Type: string or resource of type ‘ebs-volume’

Default: `""`

Example: `"vol-00000000"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`deployment.ec2.blockDeviceMapping.<name>.encrypt`
Whether the EBS volume should be encrypted using LUKS.

Type: boolean

Default: `false`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`deployment.ec2.blockDeviceMapping.<name>.encryptionType`
Whether the EBS volume should be encrypted using LUKS or on the underlying EBS volume (Amazon EBS feature). Possible values are "luks" (default) and "ebs".

Type: one of "luks", "ebs"

Default: `"luks"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.blockDeviceMapping.<name>.fsType`

Filesystem type for automatically created EBS volumes.

Type: string

Default: `"ext4"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.blockDeviceMapping.<name>.iops`

The provisioned IOPS you want to associate with this EBS volume.

Type: signed integer

Default: `0`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ebs-options.nix

`deployment.ec2.blockDeviceMapping.<name>.keySize`

The size of the encryption key.

Type: signed integer

Default: `128`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.blockDeviceMapping.<name>.passphrase`

The passphrase (key file) used to decrypt the key to access the device. If left empty, a passphrase is generated automatically; this passphrase is lost when you destroy the machine or remove the volume, unless you copy it from NixOps's state file. Note that the passphrase is stored in the Nix store of the instance, so an attacker who gains access to the EBS volume or instance store that contains the Nix store can subsequently decrypt the encrypted volume.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.blockDeviceMapping.<name>.size`

Volume size (in gigabytes). This may be left unset if you are creating the volume from a snapshot, in which case the size of the volume will be equal to the size of the snapshot. However, you can set a size larger than the snapshot, allowing the volume to be larger than the snapshot from which it is created.

Type: signed integer

Example: `100`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ebs-options.nix

`deployment.ec2.blockDeviceMapping.<name>.volumeType`

The volume type for the EBS volume, which must be one of `"standard"` (a magnetic volume), `"io1"` (a provisioned IOPS SSD volume) or `"gp2"` (a general purpose SSD volume). `"st1"` (a throughput optimized HDD volume). `"sc1"` (a cold HDD volume).

Type: one of "standard", "io1", "gp2", "st1", "sc1"

Default: `"standard"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ebs-options.nix

`deployment.ec2.ebsBoot`

Whether you want to boot from an EBS-backed AMI. Only EBS-backed instances can be stopped and restarted, and attach other EBS volumes at boot time. This option determines the selection of the default AMI; if you explicitly specify `deployment.ec2.ami`, it has no effect.

Type: boolean

Default: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.ebsInitialRootDiskSize`

Preferred size (G) of the root disk of the EBS-backed instance. By default, EBS-backed images have a size determined by the AMI. Only supported on creation of the instance.

Type: signed integer

Default: `0`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.ebsOptimized`

Whether the EC2 instance should be created as an EBS Optimized instance.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.elasticIPv4`

Elastic IPv4 address to be associated with this machine.

Type: string or resource of type ‘elastic-ip’

Default: `""`

Example: `"123.1.123.123"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.instanceId`

EC2 instance ID (set by NixOps).

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.instanceProfile`

The name of the IAM Instance Profile (IIP) to associate with the instances.

Type: string

Default: `""`

Example: `"rolename"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.instanceType`

EC2 instance type. See http://aws.amazon.com/ec2/instance-types/ for a list of valid Amazon EC2 instance types.

Type: string

Default: `"m1.small"`

Example: `"m1.large"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.keyPair`

Name of the SSH key pair to be used to communicate securely with the instance. Key pairs can be created using the ec2-add-keypair command.

Type: string or resource of type ‘ec2-keypair’

Example: `"my-keypair"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.physicalProperties`

Attribute set containing number of CPUs and memory available to the machine.

Type: unspecified

Default: `{ }`

Example: `{ cores = 4; memory = 14985; }`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`deployment.ec2.placementGroup`

Placement group for the instance.

Type: string or resource of type ‘ec2-placement-group’

Default: `""`

Example: `"my-cluster"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`deployment.ec2.privateKey`

Path of the SSH private key file corresponding with `deployment.ec2.keyPair`. NixOps will use this private key if set; otherwise, the key must be findable by SSH through its normal mechanisms (e.g. it should be listed in `~/.ssh/config` or added to the ssh-agent).

Type: string

Default: `""`

Example: `"/home/alice/.ssh/id_rsa-my-keypair"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`deployment.ec2.region`

AWS region in which the instance is to be deployed. This option only applies when using EC2. It implicitly sets `deployment.ec2.ami`.

Type: string

Default: `""`

Example: `"us-east-1"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`deployment.ec2.securityGroupIds`

Security Group IDs for the instance. Necessary if starting an instance inside a VPC/subnet. In the non-default VPC, security groups needs to be specified by ID and not name.

Type: list of strings

Default: `[ "default" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.securityGroups`

Security groups for the instance. These determine the firewall rules applied to the instance.

Type: list of string or resource of type ‘ec2-security-group’s

Default: `[ "default" ]`

Example: `[ "my-group" "my-other-group" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.spotInstancePrice`

Price (in dollar cents per hour) to use for spot instances request for the machine. If the value is equal to 0 (default), then spot instances are not used.

Type: signed integer

Default: `0`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.spotInstanceTimeout`

The duration (in seconds) that the spot instance request is valid. If the request cannot be satisfied in this amount of time, the request will be cancelled automatically, and NixOps will fail with an error message. The default (0) is no timeout.

Type: signed integer

Default: `0`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.subnetId`

The subnet inside a VPC to launch the instance in.

Type: string or resource of type ‘vpc-subnet’

Default: `""`

Example: `"subnet-00000000"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.tags`

Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.usePrivateIpAddress`

If instance is in a subnet/VPC whether to use the private IP address for ssh connections to this host. Defaults to true in the case that you are deploying into a subnet but not associating a public ip address.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.ec2.zone`

The EC2 availability zone in which the instance should be created. If not specified, a zone is selected automatically.

Type: string

Default: `""`

Example: `"us-east-1c"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`deployment.encryptedLinksTo`

NixOps will set up an encrypted tunnel (via SSH) to the machines listed here. Since this is a two-way (peer to peer) connection, it is not necessary to set this option on both endpoints. NixOps will set up `/etc/hosts` so that the host names of the machines listed here resolve to the IP addresses of the tunnels. It will also add the alias *`machine`*`-encrypted` for each machine.

Type: list of strings

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/options.nix

`deployment.gce.accessKey`

The path to GCE Service Account key. If left empty, it defaults to the contents of the environment variable `ACCESS_KEY_PATH`.

Type: string or path

Default: `""`

Example:

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.blockDeviceMapping`

Block device mapping.

Type: attribute set of submodules

Default: `{ }`

Example: `{ /dev/sda = { image = "bootstrap-img"; } ; /dev/sdb = { disk = "vol-d04895b8"; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.blockDeviceMapping.<name>.bootDisk`

Should the instance boot from this disk.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.blockDeviceMapping.<name>.cipher`

The cipher used to encrypt the disk.

Type: string

Default: `"aes-cbc-essiv:sha256"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.blockDeviceMapping.<name>.deleteOnTermination`

For automatically created GCE disks, determines whether the disk should be deleted on instance destruction.

Type: boolean

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.blockDeviceMapping.<name>.disk`

GCE Disk resource or name of a disk not managed by NixOps to be mounted.

Type: null or string or resource of type ‘gce-disk’

Default: `null`

Example: `"resources.gceDisks.exampleDisk"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.blockDeviceMapping.<name>.diskType`

The disk storage type (standard/ssd).

Type: string

Default: `"standard"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.blockDeviceMapping.<name>.disk_name`

Name of the GCE disk to create.

Type: null or string

Default: `null`

Example: `"machine-persistent-disk2"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.blockDeviceMapping.<name>.encrypt`

Whether the GCE disk should be encrypted using LUKS.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.blockDeviceMapping.<name>.image`

The image name or resource from which to create the GCE disk. If not specified, an empty disk is created. Changing the image name has no effect if the disk already exists.

Type: null or string or resource of type ‘gce-image’

Default: `null`

Example: `"image-432"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.blockDeviceMapping.<name>.keySize`

The size of the encryption key.

Type: signed integer

Default: `128`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix`

`deployment.gce.blockDeviceMapping.<name>.passphrase`

The passphrase (key file) used to decrypt the key to access the device. If left empty, a passphrase is generated automatically; this passphrase is lost when you destroy the machine or remove the volume, unless you copy it from NixOps's state file. Note that the passphrase is stored in the Nix store of the instance, so an attacker who gains access to the GCE disk or instance store that contains the Nix store can subsequently decrypt the encrypted volume.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix`

`deployment.gce.blockDeviceMapping.<name>.readOnly`

Should the disk be attached to the instance as read-only.

Type: boolean

Default: `false`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix`

`deployment.gce.blockDeviceMapping.<name>.size`

Volume size (in gigabytes) for automatically created GCE disks. This may be left unset if you are creating the disk from a snapshot or image, in which case the size of the disk will be equal to the size of the snapshot or image. You can set a size larger than the snapshot or image, allowing the disk to be larger than the snapshot from which it is created.

Type: null or signed integer

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix`

`deployment.gce.blockDeviceMapping.<name>.snapshot`

The snapshot name from which to create the GCE disk. If not specified, an empty disk is created. Changing the snapshot name has no effect if the disk already exists.

Type: null or string

Default: `null`

Example: `"snapshot-432"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix`

`deployment.gce.bootstrapImage`

Bootstrap image name or resource to use to create the root disk of the instance.

Type: string or resource of type ‘gce-image’

Default: `{ }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.canIpForward`
Allows the instance to send and receive packets with non-matching destination or source IPs.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.instanceServiceAccount`
A service account with its specified scopes, authorized for this instance.

Type: submodule

Default: `{ }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.instanceServiceAccount.email`
Email address of the service account. If not given, Google Compute Engine default service account is used.

Type: string

Default: `"default"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.instanceServiceAccount.scopes`
The list of scopes to be made available for this service account.

Type: list of strings

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.instanceType`
GCE instance type. See https://developers.google.com/compute/pricing for a list of valid instance types.

Type: string

Default: `"g1-small"`

Example: `"n1-standard-1"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.ipAddress`

GCE Static IP address resource to bind to or the name of an IP address not managed by NixOps.

Type: null or string or resource of type ‘gce-static-ip’

Default: `null`

Example: `"resources.gceStaticIPs.exampleIP"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.labels`

A set of key/value label pairs to assign to the instance.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.machineName`

The GCE Instance `Name`.

Type: string

Default: `"n-<uuid>-<name>"`

Example: `"custom-machine-name"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.metadata`

Metadata to assign to the instance. These are available to the instance via the metadata server. Some metadata keys such as "startup-script" are reserved by GCE and can influence the instance.

Type: attribute set of strings

Default: `{ }`

Example: `{ loglevel = "warn"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.network`

The GCE Network to make the instance a part of. Can be either a gceNetworks resource or a name of a network not managed by NixOps.

Type: null or string or resource of type ‘gce-network’

Default: `null`

Example: `"resources.gceNetworks.verySecureNetwork"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.project`

The GCE project which should own the instance. If left empty, it defaults to the contents of the environment variable `GCE_PROJECT`.

Type: string

Default: `""`

Example: `"myproject"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.region`

The GCE datacenter in which the instance should be created.

Type: string

Example: `"europe-west1-b"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.rootDiskSize`

Root disk size(in gigabytes). Leave unset to be the same as `bootstrapImage` size.

Type: null or signed integer

Default: `null`

Example: `200`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.rootDiskType`

The root disk storage type (standard/ssd).

Type: string

Default: `"standard"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix`

`deployment.gce.scheduling.automaticRestart`

Whether the Instance should be automatically restarted when it is terminated by Google Compute Engine (not terminated by user).

Type: boolean

Default: `true`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix`

`deployment.gce.scheduling.onHostMaintenance`

Defines the maintenance behavior for this instance. For more information, see https://developers.google.com/compute/docs/instances#onhostmaintenance.

Allowed values are: "MIGRATE" to let GCE automatically migrate your instances out of the way of maintenance events and "TERMINATE" to allow GCE to terminate and restart the instance.

Type: string

Default: `"MIGRATE"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix`

`deployment.gce.scheduling.preemptible`

Whether the instance is preemptible. For more information, see https://developers.google.com/compute/docs/instances#onhostmaintenance.

Type: boolean

Default: `false`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix`

`deployment.gce.serviceAccount`

The GCE Service Account Email. If left empty, it defaults to the contents of the environment variable `GCE_SERVICE_ACCOUNT`.

Type: string

Default: `""`

Example: `"12345-asdf@developer.gserviceaccount.com"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix`

`deployment.gce.subnet`

Specifies the subnet that the instances will be part of.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.gce.tags`

Tags to assign to the instance. These can be used in firewall and networking rules and are additionally available as metadata.

Type: list of strings

Default: `[ ]`

Example: `[ "random" "tags" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`deployment.hasFastConnection`

If set to `true`, whole closure will be copied using just \`nix-copy-closure\`.

If set to `false`, closure will be copied first using binary substitution. Addtionally, any missing derivations copied with \`nix-copy-closure\` will be done using `--gzip` flag.

Some backends set this value to `true`.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/options.nix

`deployment.hetzner.createSubAccount`

Whether NixOps should create a Hetzner "Admin account" (a sub-account that allows to manage this single machine).

You must disable this when your Hetzner main account is protected with 2-factor authentication, as the Hetzner webservice API does not support 2-factor auth.

When this is disabled, you must manually create the sub-account for each machine in the Hetzner Robot UI before running NixOps.

When this is disabled, NixOps assumes that the credentials for the sub-account are those given with the \`robotUser\` and \`robotPass\` options. If those are left empty, the values of the environment variables `HETZNER_ROBOT_USER` and `HETZNER_ROBOT_PASS` are used instead.

Note that if you have more than one Hetzner and \`createSubAccount = false\`, it does not make sense to use `HETZNER_ROBOT_USER` because Hetzner (as of writing) enforces a different sub-account user name for each server, so you should use \`robotUser\` per machine instead of using the environment variable. But you may use the environment variable for the password if you set the sub-account passwords to be identical.

Type: boolean

Default: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/hetzner.nix

`deployment.hetzner.mainIPv4`

Main IP address identifying the server.

Type: null or string

Default: `null`

Example: `"78.46.1.93"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/hetzner.nix

`deployment.hetzner.partitions`

Specify layout of partitions and file systems using Anacondas Kickstart format. For possible options and commands, please have a look at:

http://fedoraproject.org/wiki/Anaconda/Kickstart

Type: strings concatenated with "\n"

Default:

```
''
clearpart --all --initlabel --drives=sda,sdb

part swap1 --recommended --label=swap1 --fstype=swap --ondisk=sda
part swap2 --recommended --label=swap2 --fstype=swap --ondisk=sdb

part raid.1 --grow --ondisk=sda
part raid.2 --grow --ondisk=sdb

raid / --level=1 --device=md0 --fstype=ext4 --label=root raid.1 raid.2
''
```

Example:

```
''
# Example for partitioning on a vServer:
clearpart --all --initlabel --drives=vda
part swap --recommended --label=swap --fstype=swap --ondisk=vda
part / --fstype=ext4 --label=root --grow --ondisk=vda
''
```

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/hetzner.nix

`deployment.hetzner.robotPass`

Password of the Hetzner robot account.

If left empty, the value of the environment variable `HETZNER_ROBOT_PASS` is used instead.

Type: null or string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/hetzner.nix

`deployment.hetzner.robotUser`

Username of the Hetzner robot account.

If left empty, the value of the environment variable `HETZNER_ROBOT_USER` is used instead.

Type: null or string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/hetzner.nix

`deployment.keys`

The set of keys to be deployed to the machine. Each attribute maps a key name to a file that can be accessed as *`destDir`*`/`*`name`*, where `destDir` defaults to `/run/keys`. Thus, `{ password.text = "foobar"; }` causes a file *`destDir`*`/password` to be created with contents `foobar`. The directory *`destDir`* is only accessible to root and the `keys` group, so keep in mind to add any users that need to have access to a particular key to this group.

Each key also gets a systemd service *`name`*`-key.service` which is active while the key is present and inactive while the key is absent. Thus, `{ password.text = "foobar"; }` gets a `password-key.service`.

Type: attribute set of string or key optionss

Default: `{ }`

Example: `{ password = { text = "foobar"; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/keys.nix

`deployment.keys.<name>.destDir`

When specified, this allows changing the destDir directory of the key file from its default value of `/run/keys`.

This directory will be created, its permissions changed to `0750` and ownership to `root:keys`.

Type: path

Default: `"/run/keys"`

`deployment.keys.<name>.group`

The group that will be set for the key file.

Type: string

Default: `"root"`

`deployment.keys.<name>.keyFile`

When non-null, contents of the specified file will be deployed to the specified key on the target machine. If the key name is *`password`* and `/foo/bar` is set here, the contents of the file *destDir/password* deployed will be the same as local file `/foo/bar`.

Since no serialization/deserialization of key contents is involved, there are no limits on that content: null bytes, invalid Unicode, `/dev/random` output -- anything goes.

NOTE: Either `text` or `keyFile` have to be set.

Type: null or path

Default: `null`

`deployment.keys.<name>.permissions`

The default permissions to set for the key file, needs to be in the format accepted by chmod(1).

Type: string

Default: `"0600"`

Example: `"0640"`

`deployment.keys.<name>.text`

When non-null, this designates the text that the key should contain. So if the key name is *password* and `foobar` is set here, the contents of the file *destDir/password* will be `foobar`.

NOTE: Either `text` or `keyFile` have to be set.

Type: null or string

Default: `null`

Example: `"super secret stuff"`

`deployment.keys.<name>.user`

The user which will be the owner of the key file.

Type: string

Default: `"root"`

`deployment.libvirtd.baseImage`

The disk is created using the specified disk image as a base.

Type: null or path

Default: `null`

Example: `"/home/alice/base-disk.qcow2"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.baseImageSize`

The size (G) of base image of virtual machine.

Type: signed integer

Default: `10`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.cmdline`

Specify the kernel cmdline (valid only with the kernel setting).

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.domainType`

Specify the type of libvirt domain to create (see '$ virsh capabilities | grep domain' for valid domain types

Type: string

Default: `"kvm"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.extraDevicesXML`

Additional XML appended at the end of device tag in domain xml. See https://libvirt.org/formatdomain.html

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.extraDomainXML`

Additional XML appended at the end of domain xml. See https://libvirt.org/formatdomain.html

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.headless`

If set VM is started in headless mode, i.e., without a visible display on the host's desktop.

Type: unspecified

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.imageDir`

Directory to store VM image files. Note that it should be writable both by you and by libvirtd daemon.

Type: path

Default: `"/var/lib/libvirt/images"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.initrd`

Specify the kernel initrd (valid only with the kernel setting).

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.kernel`

Specify the host kernel to launch (valid for kvm).

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.memorySize`

Memory size (M) of virtual machine.

Type: signed integer

Default: `512`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.networks`

Names of libvirt networks to attach the VM to.

Type: list of strings

Default: `[ "default" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.libvirtd.vcpu`

Number of Virtual CPUs.

Type: signed integer

Default: `1`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/libvirtd.nix

`deployment.owners`

List of email addresses of the owners of the machines. Used to send email on performing certain actions.

Type: list of strings

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/options.nix

`deployment.route53.accessKeyId`

The AWS Access Key ID. If left empty, it defaults to the contents of the environment variables `EC2_ACCESS_KEY` or `AWS_ACCESS_KEY_ID` (in that order). The corresponding Secret Access Key is not specified in the deployment model, but looked up in the file `~/.ec2-keys`, which should specify, on each line, an Access Key ID followed by the corresponding Secret Access Key. If the lookup was unsuccessful it is continued in the standard AWS tools `~/.aws/credentials` file. If it does not appear in these files, the environment variables `EC2_SECRET_KEY` or `AWS_SECRET_ACCESS_KEY` are used.

Type: string

Default: `""`

Example: `"AKIABOGUSACCESSKEY"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/route53.nix

`deployment.route53.hostName`

The DNS hostname to bind the public IP address to.

Type: string

Default: `""`

Example: `"test.x.logicblox.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/route53.nix

`deployment.route53.ttl`

The time to live (TTL) for the A record created for the specified DNS hostname.

Type: signed integer

Default: `300`

Example: `300`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/route53.nix

`deployment.route53.usePublicDNSName`

Whether to create a CNAME record with the instance's public DNS name. This will resolve inside AWS to a private IP and outside AWS to the public IP.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/route53.nix

`deployment.storeKeysOnMachine`

If true, secret information such as LUKS encryption keys or SSL private keys is stored on the root disk of the machine, allowing the machine to do unattended reboots. If false, secrets are not stored; NixOps supplies them to the machine at mount time. This means that a reboot will not complete entirely until you run nixops deploy or nixops send-keys.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/keys.nix

`deployment.targetEnv`

This option specifies the type of the environment in which the machine is to be deployed by NixOps. Currently, it can have the following values. `"none"` means deploying to a pre-existing physical or virtual NixOS machine, reachable via SSH under the hostname or IP address specified in `deployment.targetHost`. `"ec2"` means that a virtual machine should be instantiated in an Amazon EC2-compatible cloud environment (see `deployment.ec2.*`). `"virtualbox"` causes a VirtualBox VM to be created on your machine. (This requires VirtualBox to be configured on your system.)

Type: string

Default: `"none"`

Example: `"ec2"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/options.nix

`deployment.targetHost`

This option specifies the hostname or IP address to be used by NixOps to execute remote deployment operations.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/options.nix

`deployment.targetPort`

This option specifies the SSH port to be used by NixOps to execute remote deployment operations.

Type: signed integer

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/options.nix

`deployment.virtualbox.disks`

Definition of the virtual disks attached to this instance. The root disk is called `deployment.virtualbox.disks.disk1`.

Type: attribute set of submodules

Default: `{ }`

Example: `{ big-disk = { port = 1; size = 1048576; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/virtualbox.nix

`deployment.virtualbox.disks.<name>.baseImage`

If set, this disk is created as a clone of the specified disk image.

Type: null or path

Default: `null`

Example: `"/home/alice/base-disk.vdi"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/virtualbox.nix

`deployment.virtualbox.disks.<name>.port`

SATA port number to which the disk is attached.

Type: signed integer

Example: `1`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/virtualbox.nix

`deployment.virtualbox.disks.<name>.size`

Size (in megabytes) of this disk.

Type: signed integer

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/virtualbox.nix

`deployment.virtualbox.headless`

If set, the VirtualBox instance is started in headless mode, i.e., without a visible display on the host's desktop.

Type: unspecified

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/virtualbox.nix

`deployment.virtualbox.memorySize`

Memory size (M) of virtual machine.

Type: signed integer

Default: `512`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/virtualbox.nix

`deployment.virtualbox.sharedFolders`

Definition of the host folders that should be shared with this instance.

Type: attribute set of submodules

Default: `{ }`

Example: `{ home = { hostPath = "/home"; readOnly = false; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/virtualbox.nix

`deployment.virtualbox.sharedFolders.<name>.hostPath`

The path of the host directory that should be shared to the guest

Type: string

Example: `"/home"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/virtualbox.nix

`deployment.virtualbox.sharedFolders.<name>.readOnly`

Specifies if the shared folder should be read-only for the guest

Type: boolean

Default: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/virtualbox.nix

`deployment.virtualbox.vcpu`

Number of Virtual CPUs. Left unspecified if not provided.

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/virtualbox.nix

`deployment.virtualbox.vmFlags`

Arbitrary string arguments to append to the modifyvm command.

Type: list of Concatenated strings

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/virtualbox.nix

`fileSystems`

NixOps extends NixOS' `fileSystem` option to allow convenient attaching of EC2 volumes.

Type: list or attribute set of submodules

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix
/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix
/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix
/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/doc/manual/dummy.nix

`fileSystems.<name?>.azure`

Azure disk to be attached to this mount point. This is a shorthand for defining a separate `deployment.azure.blockDeviceMapping` attribute.

Type: null or submodule

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`fileSystems.<name?>.azure.cipher`

The cipher used to encrypt the disk.

Type: string

Default: `"aes-cbc-essiv:sha256"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`fileSystems.<name?>.azure.encrypt`

Whether the Azure disk should be encrypted using LUKS.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`fileSystems.<name?>.azure.hostCaching`

Specifies the platform caching behavior of data disk blob for read/write efficiency. The default vault is None. Possible values are: None, ReadOnly, ReadWrite.

Type: one of "None", "ReadOnly", "ReadWrite"

Default: `"None"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix`

`fileSystems.<name?>.azure.isEphemeral`

Whether the disk is ephemeral. Emphemeral disk BLOBs are automatically created and destroyed by NixOps as needed. The user has an option to keep the BLOB with contents after the virtual machine is destroyed. Ephemeral disk names need to be unique only among the other ephemeral disks of the virtual machine.

Type: boolean

Default: `true`

Example: `false`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix`

`fileSystems.<name?>.azure.keySize`

The size of the encryption key.

Type: signed integer

Default: `128`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix`

`fileSystems.<name?>.azure.lun`

Logical Unit Number (LUN) location for the data disk in the virtual machine. Required if the disk is created via fileSystems.X.azure attrset. The disk will appear as /dev/disk/by-lun/*. Must be unique. Valid values are: 0-31. LUN value must be less than the maximum number of allowed disks for the virtual machine size.

Type: null or signed integer

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix`

`fileSystems.<name?>.azure.mediaLink`

The location of the BLOB in the Azure BLOB store to store the ephemeral disk contents. The BLOB location must belong to a storage account in the same subscription as the virtual machine. If the BLOB doesn't exist, it will be created.

Type: null or string

Default: `null`

Example: `"http://mystorage.blob.core.windows.net/mycontainer/machine-disk"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`fileSystems.<name?>.azure.name`

The short name of the disk to create.

Type: null or string

Default: `null`

Example: `"data"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`fileSystems.<name?>.azure.passphrase`

The passphrase (key file) used to decrypt the key to access the device. If left empty, a passphrase is generated automatically; this passphrase is lost when you destroy the machine or remove the volume, unless you copy it from NixOps's state file. Note that the passphrase is stored in the Nix store of the instance, so an attacker who gains access to the Azure disk or instance store that contains the Nix store can subsequently decrypt the encrypted volume.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`fileSystems.<name?>.azure.size`

Volume size (in gigabytes) for automatically created Azure disks. This option value is ignored if the disk BLOB already exists.

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure.nix

`fileSystems.<name?>.ec2`

EC2 disk to be attached to this mount point. This is shorthand for defining a separate `deployment.ec2.blockDeviceMapping` attribute.

Type: null or submodule

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`fileSystems.<name?>.ec2.cipher`

The cipher used to encrypt the disk.

Type: string

Default: `"aes-cbc-essiv:sha256"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`fileSystems.<name?>.ec2.deleteOnTermination`
For automatically created EBS volumes, determines whether the volume should be deleted on instance termination.

Type: boolean

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`fileSystems.<name?>.ec2.disk`
EC2 identifier of the disk to be mounted. This can be an ephemeral disk (e.g. `ephemeral0`), a snapshot ID (e.g. `snap-00000000`) or a volume ID (e.g. `vol-00000000`). Leave empty to create an EBS volume automatically. It can also be an EBS resource (e.g. `resources.ebsVolumes.big-disk`).

Type: string or resource of type ‘ebs-volume’

Default: `""`

Example: `"vol-00000000"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`fileSystems.<name?>.ec2.encrypt`
Whether the EBS volume should be encrypted using LUKS.

Type: boolean

Default: `false`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`fileSystems.<name?>.ec2.encryptionType`
Whether the EBS volume should be encrypted using LUKS or on the underlying EBS volume (Amazon EBS feature). Possible values are "luks" (default) and "ebs".

Type: one of "luks", "ebs"

Default: `"luks"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix`

`fileSystems.<name?>.ec2.fsType`
Filesystem type for automatically created EBS volumes.

Type: string

Default: `"ext4"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`fileSystems.<name?>.ec2.iops`

The provisioned IOPS you want to associate with this EBS volume.

Type: signed integer

Default: `0`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ebs-options.nix

`fileSystems.<name?>.ec2.keySize`

The size of the encryption key.

Type: signed integer

Default: `128`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`fileSystems.<name?>.ec2.passphrase`

The passphrase (key file) used to decrypt the key to access the device. If left empty, a passphrase is generated automatically; this passphrase is lost when you destroy the machine or remove the volume, unless you copy it from NixOps's state file. Note that the passphrase is stored in the Nix store of the instance, so an attacker who gains access to the EBS volume or instance store that contains the Nix store can subsequently decrypt the encrypted volume.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2.nix

`fileSystems.<name?>.ec2.size`

Volume size (in gigabytes). This may be left unset if you are creating the volume from a snapshot, in which case the size of the volume will be equal to the size of the snapshot. However, you can set a size larger than the snapshot, allowing the volume to be larger than the snapshot from which it is created.

Type: signed integer

Example: `100`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ebs-options.nix

`fileSystems.<name?>.ec2.volumeType`

The volume type for the EBS volume, which must be one of `"standard"` (a magnetic volume), `"io1"` (a provisioned IOPS SSD volume) or `"gp2"` (a general purpose SSD volume). `"st1"` (a throughput optimized HDD volume). `"sc1"` (a cold HDD volume).

Type: one of "standard", "io1", "gp2", "st1", "sc1"

Default: `"standard"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ebs-options.nix

`fileSystems.<name?>.gce`

GCE disk to be attached to this mount point. This is shorthand for defining a separate `deployment.gce.blockDeviceMapping` attribute.

Type: null or submodule

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.bootDisk`

Should the instance boot from this disk.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.cipher`

The cipher used to encrypt the disk.

Type: string

Default: `"aes-cbc-essiv:sha256"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.deleteOnTermination`

For automatically created GCE disks, determines whether the disk should be deleted on instance destruction.

Type: boolean

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.disk`

GCE Disk resource or name of a disk not managed by NixOps to be mounted.

Type: null or string or resource of type ‘gce-disk’

Default: `null`

Example: `"resources.gceDisks.exampleDisk"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.diskType`

The disk storage type (standard/ssd).

Type: string

Default: `"standard"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.disk_name`

Name of the GCE disk to create.

Type: null or string

Default: `null`

Example: `"machine-persistent-disk2"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.encrypt`

Whether the GCE disk should be encrypted using LUKS.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.image`

The image name or resource from which to create the GCE disk. If not specified, an empty disk is created. Changing the image name has no effect if the disk already exists.

Type: null or string or resource of type ‘gce-image’

Default: `null`

Example: `"image-432"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.keySize`

The size of the encryption key.

Type: signed integer

Default: `128`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.passphrase`

The passphrase (key file) used to decrypt the key to access the device. If left empty, a passphrase is generated automatically; this passphrase is lost when you destroy the machine or remove the volume, unless you copy it from NixOps's state file. Note that the passphrase is stored in the Nix store of the instance, so an attacker who gains access to the GCE disk or instance store that contains the Nix store can subsequently decrypt the encrypted volume.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.readOnly`

Should the disk be attached to the instance as read-only.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.size`

Volume size (in gigabytes) for automatically created GCE disks. This may be left unset if you are creating the disk from a snapshot or image, in which case the size of the disk will be equal to the size of the snapshot or image. You can set a size larger than the snapshot or image, allowing the disk to be larger than the snapshot from which it is created.

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`fileSystems.<name?>.gce.snapshot`

The snapshot name from which to create the GCE disk. If not specified, an empty disk is created. Changing the snapshot name has no effect if the disk already exists.

Type: null or string

Default: `null`

Example: `"snapshot-432"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce.nix

`networking.p2pTunnels.ssh`

A set of peer-to-peer tunnels set up automatically over SSH.

Type: attribute set of submodules

Default: `{ }`

Example: `{ tunnel1 = { localIPv4 = "172.16.12.1"; localTunnel = 0; privateKey = "/root/.ssh/id_vpn"; remoteIPv4 = "172.16.12.2"; remoteTunnel = 1; target = "192.0.2.1"; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ssh-tunnel.nix

`networking.p2pTunnels.ssh.<name>.localIPv4`

IPv4 address of the local endpoint of the tunnel.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ssh-tunnel.nix

`networking.p2pTunnels.ssh.<name>.localTunnel`

Local tunnel device number.

Type: signed integer

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ssh-tunnel.nix

`networking.p2pTunnels.ssh.<name>.privateKey`

Path to the private key file used to connect to the remote machine.

Type: path

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ssh-tunnel.nix

`networking.p2pTunnels.ssh.<name>.remoteIPv4`

IPv4 address of the remote endpoint of the tunnel.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ssh-tunnel.nix

`networking.p2pTunnels.ssh.<name>.remoteTunnel`

Remote tunnel device number.

Type: signed integer

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ssh-tunnel.nix

`networking.p2pTunnels.ssh.<name>.target`

Host name or IP address of the remote machine.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ssh-tunnel.nix

`networking.p2pTunnels.ssh.<name>.targetPort`

Port number that SSH listens to on the remote machine.

Type: signed integer

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ssh-tunnel.nix

`networking.privateIPv4`

IPv4 address of this machine within in the logical network. This address can be used by other machines in the logical network to reach this machine. However, it need not be visible to the outside (i.e., publicly routable).

Type: string

Example: `"10.1.2.3"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/options.nix

`networking.publicIPv4`

Publicly routable IPv4 address of this machine.

Type: null or string

Default: `null`

Example: `"198.51.100.123"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/options.nix

`networking.vpnPublicKey`

Public key of the machine's VPN key (set by nixops)

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/options.nix

B.2. AWS Resources

This section lists resource types associated with the Amazon Web Services (AWS) cloud computing environment.

B.2.1. EBS Volumes

An Amazon EBS volume is defined by setting `resources.ebsVolumes.name` to an attribute set containing values for the following options.

Appendix D. Configuration Options

`accessKeyId`
: The AWS Access Key ID.

  Type: string

  Default: `""`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ebs-volume.nix

`iops`
: The provisioned IOPS you want to associate with this EBS volume.

  Type: signed integer

  Default: `0`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ebs-options.nix

`region`
: AWS region.

  Type: string

  Example: `"us-east-1"`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ebs-volume.nix

`size`
: Volume size (in gigabytes). This may be left unset if you are creating the volume from a snapshot, in which case the size of the volume will be equal to the size of the snapshot. However, you can set a size larger than the snapshot, allowing the volume to be larger than the snapshot from which it is created.

  Type: signed integer

  Example: `100`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ebs-options.nix

`snapshot`

The snapshot ID from which this volume will be created. If not specified, an empty volume is created. Changing the snapshot ID has no effect if the volume already exists.

Type: string

Default: `""`

Example: `"snap-1cbda474"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ebs-volume.nix

`tags`

Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ebs-volume.nix

`volumeType`

The volume type for the EBS volume, which must be one of `"standard"` (a magnetic volume), `"io1"` (a provisioned IOPS SSD volume) or `"gp2"` (a general purpose SSD volume). `"st1"` (a throughput optimized HDD volume). `"sc1"` (a cold HDD volume).

Type: one of "standard", "io1", "gp2", "st1", "sc1"

Default: `"standard"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ebs-options.nix

`zone`

The EC2 availability zone in which the volume should be created.

Type: string

Example: `"us-east-1c"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ebs-volume.nix

B.2.2. SQS Queues

An Amazon SQS queue is defined by setting `resources.sqsQueues.`*`name`* to an attribute set containing values for the following options.

Appendix E. Configuration Options

`accessKeyId`
: The AWS Access Key ID.

  Type: string

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sqs-queue.nix

`arn`
: Amazon Resource Name (ARN) of the queue. This is set by NixOps.

  Type: string

  Default: `""`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sqs-queue.nix

`name`
: Name of the SQS queue.

  Type: string

  Default: `"charon-<uuid>-<name>"`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sqs-queue.nix

`region`
: AWS region.

  Type: string

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sqs-queue.nix

`url`
: URL of the queue. This is set by NixOps.

  Type: string

  Default: `""`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sqs-queue.nix

`visibilityTimeout`
: The time interval in seconds after a message has been received until it becomes visible again.

  Type: signed integer

  Default: `30`

  Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sqs-queue.nix

B.2.3. SNS Topics

An Amazon SNS topic is defined by setting `resources.snsTopics.`*`name`* to an attribute set containing values for the following options.

Appendix F. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sns-topic.nix

`arn`

Amazon Resource Name (ARN) of the SNS topic. This is set by NixOps.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sns-topic.nix

`displayName`

Display name of the topic

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sns-topic.nix

`name`

Name of the SNS topic.

Type: string

Default: `"charon-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sns-topic.nix

`policy`

Policy to apply to the SNS topic.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sns-topic.nix

`region`

AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sns-topic.nix

`subscriptions`

List of subscriptions to apply to the topic.

Type: list of submodules

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sns-topic.nix

`subscriptions.*.endpoint`

The endpoint to send data to.

Type: string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sns-topic.nix

`subscriptions.*.protocol`

The protocol to use.

Type: string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/sns-topic.nix

B.2.4. EC2 Keypairs

An Amazon EC2 keypair is defined by setting `resources.ec2KeyPairs.`*`name`* to an attribute set containing values for the following options.

Appendix G. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-keypair.nix`

`name`

Name of the EC2 key pair.

Type: string

Default: `"charon-<uuid>-<name>"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-keypair.nix`

`region`

AWS region.

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-keypair.nix`

B.2.5. EC2 Security Groups

An Amazon Security Group is defined by setting `resources.ec2SecurityGroups.`*`name`* to an attribute set containing values for the following options.

Appendix H. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix`

`description`

Informational description of the security group.

Type: string

Default: `"NixOps-provisioned group <name>"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix`

`groupId`

The security group ID. This is set by NixOps.

Type: null or string

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix`

`name`

Name of the security group.

Type: string

Default: `"charon-<uuid>-<name>"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix`

`region`

AWS region.

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix`

`rules`

The security group's rules.

Type: list of submodules

Default: `{ }`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix`

`rules.*.codeNumber`

ICMP code number (ICMP only, -1 for all).

Type: null or signed integer

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix`

`rules.*.fromPort`

The bottom of the allowed port range for this rule (TCP/UDP only).

Type: null or signed integer

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix`

`rules.*.protocol`

The protocol (tcp, udp, or icmp) that this rule describes. Use "-1" to specify All.

Type: string

Default: `"tcp"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix

`rules.*.sourceGroup.groupName`

The name of the source security group (if allowing all instances in a group access instead of an IP range).

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix

`rules.*.sourceGroup.ownerId`

The AWS account ID that owns the source security group.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix

`rules.*.sourceIp`

The source IP range (CIDR notation).

Can also be a reference to ElasticIP resource, which will be suffixed with /32 CIDR notation.

Type: null or string or resource of type ‘elastic-ip’

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix

`rules.*.toPort`

The top of the allowed port range for this rule (TCP/UDP only).

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix

`rules.*.typeNumber`

ICMP type number (ICMP only, -1 for all).

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix

`vpcId`

The VPC ID to create security group in (default is not set, uses default VPC in EC2-VPC account, in EC2-Classic accounts no VPC is set).

Type: null or string or resource of type ‘vpc’

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ec2-security-group.nix

B.2.6. Elastic IPs

An Amazon Elastic IP is defined by setting `resources.elasticIPs.`*`name`* to an attribute set containing values for the following options.

Appendix I. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-ip.nix

`address`

The elastic IP address, set by NixOps.

Type: string

Default: `"_UNKNOWN_ELASTIC_IP_"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-ip.nix

`region`

AWS region.

Type: string

Example: `"us-east-1"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-ip.nix`

`vpc`

Whether to allocate the address for use with instances in a VPC.

Type: boolean

Default: `false`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-ip.nix`

B.2.7. S3 Buckets

An Amazon S3 bucket is defined by setting `resources.s3Buckets.`*`name`* to an attribute set containing values for the following options.

Appendix J. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/s3-bucket.nix`

`arn`

Amazon Resource Name (ARN) of the S3 bucket. This is set by NixOps.

Type: string

Default: `"arn:aws:s3:::charon-<uuid>-<name>"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/s3-bucket.nix`

`lifeCycle`

The JSON lifecycle management string to apply to the bucket.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/s3-bucket.nix`

`name`

Name of the S3 bucket.

Type: string

Default: `"charon-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/s3-bucket.nix

`persistOnDestroy`

If set to true nixops destroy won't delete the bucket on destroy.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/s3-bucket.nix

`policy`

The JSON Policy string to apply to the bucket.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/s3-bucket.nix

`region`

Amazon S3 region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/s3-bucket.nix

`versioning`

Whether to enable S3 versioning or not. Valid values are 'Enabled' or 'Suspended'

Type: one of "Suspended", "Enabled"

Default: `"Suspended"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/s3-bucket.nix

`website.enabled`

Whether to serve the S3 bucket as public website.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/s3-bucket.nix

`website.errorDocument`

The S3 key to serve when response is an error.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/s3-bucket.nix

`website.suffix`

A suffix that is appended to a request that is for a directory on the website endpoint.

Type: string

Default: `"index.html"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/s3-bucket.nix

B.2.8. IAM Roles

An Amazon IAM role is defined by setting `resources.iamRoles.name` to an attribute set containing values for the following options.

Appendix K. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/iam-role.nix

`assumeRolePolicy`

The IAM AssumeRole policy definition (in JSON format). Empty string (default) uses the existing Assume Role Policy.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/iam-role.nix

`name`

Name of the IAM role.

Type: string

Default: `"charon-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/iam-role.nix

`policy`

The IAM policy definition (in JSON format).

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/iam-role.nix

B.2.9. SSH Keypairs

An SSH keypair is defined by setting `resources.sshKeyPairs.`*`name`* to an empty attribute set. You can access the generated keypair by using the following options.

Appendix L. Configuration Options

`privateKey`

The generated private key.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ssh-keypair.nix

`publicKey`

The generated public SSH key.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/ssh-keypair.nix

B.2.10. CloudWatch Log Groups

A CloudWatch Log Group is defined by setting `resources.cloudwatchLogGroups.`*`name`* to an attribute set containing values for the following options.

Appendix M. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/cloudwatch-log-group.nix

`arn`
Amazon Resource Name (ARN) of the cloudwatch log group. This is set by NixOps.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/cloudwatch-log-group.nix

`name`
Name of the cloudwatch log group.

Type: string

Default: `"charon-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/cloudwatch-log-group.nix

`region`
AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/cloudwatch-log-group.nix

`retentionInDays`
How long to store log data in a log group

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/cloudwatch-log-group.nix

B.2.11. CloudWatch Log Streams

A CloudWatch Log Stream is defined by setting `resources.cloudwatchLogStreams.`*`name`* to an attribute set containing values for the following options.

Appendix N. Configuration Options

`accessKeyId`
The AWS Access Key ID.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/cloudwatch-log-stream.nix

`arn`

Amazon Resource Name (ARN) of the cloudwatch log stream. This is set by NixOps.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/cloudwatch-log-stream.nix

`logGroupName`

The name of the log group under which the log stream is to be created.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/cloudwatch-log-stream.nix

`name`

Name of the cloudwatch log stream.

Type: string

Default: `"charon-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/cloudwatch-log-stream.nix

`region`

AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/cloudwatch-log-stream.nix

B.2.12. Elastic File Systems

An Elastic File System is defined by setting `resources.elasticFileSystems.`*`name`* to an attribute set containing values for the following options.

Appendix O. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-file-system.nix

`region`
: AWS region.

  Type: string

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-file-system.nix

`tags`
: Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

  Type: attribute set of strings

  Default: `{ }`

  Example: `{ foo = "bar"; xyzzy = "bla"; }`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-file-system.nix

B.2.13. Elastic File System Mount Target

An Elastic File System Mount Target is defined by setting `resources.resources.elasticFileSystemMountTargets.`*`name`* to an attribute set containing values for the following options.

Appendix P. Configuration Options

`accessKeyId`
: The AWS Access Key ID.

  Type: string

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-file-system-mount-target.nix

`fileSystem`
: The Elastic File System to which this mount target refers.

  Type: string or resource of type ‘elastic-file-system’

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-file-system-mount-target.nix

`ipAddress`
: The IP address of the mount target in the subnet. If unspecified, EC2 will automatically assign an address.

  Type: null or string

  Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-file-system-mount-target.nix

`region`
AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-file-system-mount-target.nix

`securityGroups`
The EC2 security groups associated with the mount target's network interface.

Type: list of strings

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-file-system-mount-target.nix

`subnet`
The EC2 subnet in which to create this mount target.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-file-system-mount-target.nix

`tags`
Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/elastic-file-system-mount-target.nix

B.2.14. Vpc

A Vpc is defined by setting `resources.vpc` *`.name`* to an attribute set containing values for the following options.

Appendix Q. Configuration Options

`accessKeyId`
The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix`

`amazonProvidedIpv6CidrBlock`
Requests an Amazon-provided IPv6 CIDR block with a /56 prefix length for the VPC. You cannot specify the range of IP addresses, or the size of the CIDR block.

Type: boolean

Default: `false`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc.nix`

`cidrBlock`
The CIDR block for the VPC

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc.nix`

`enableClassicLink`
Enables a VPC for ClassicLink. You can then link EC2-Classic instances to your ClassicLink-enabled VPC to allow communication over private IP addresses. You cannot enable your VPC for ClassicLink if any of your VPC’s route tables have existing routes for address ranges within the 10.0.0.0/8 IP address range , excluding local routes for VPCs in the 10.0.0.0/16 and 10.1.0.0/16 IP address ranges.

Type: boolean

Default: `false`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc.nix`

`enableDnsHostnames`
Specifies whether DNS hostnames are provided for the instances launched in this VPC. You can only set this attribute to true if EnableDnsSupport is also true.

Type: boolean

Default: `false`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc.nix`

`enableDnsSupport`
Specifies whether the DNS server provided by Amazon is enabled for the VPC.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc.nix

`instanceTenancy`

The supported tenancy options for instances launched into the VPC. Valid values are "default" and "dedicated".

Type: string

Default: `"default"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc.nix

`name`

Name of the VPC.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc.nix

`region`

AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`tags`

Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc.nix

`vpcId`

The VPC id generated from AWS. This is set by NixOps

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc.nix

B.2.15. Vpc Subnet

A Vpc Subnet is defined by setting `resources.vpcSubnets .`*`name`* to an attribute set containing values for the following options.

Appendix R. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`cidrBlock`

The CIDR block for the VPC subnet

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-subnet.nix

`ipv6CidrBlock`

The IPv6 network range for the subnet, in CIDR notation. The subnet size must use a /64 prefix length.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-subnet.nix

`mapPublicIpOnLaunch`

Indicates whether instances launched into the subnet should be assigned a public IP in launch. Default is false.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-subnet.nix

`name`

Name of the subnet VPC.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-subnet.nix

`region`
AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`subnetId`
The VPC subnet id generated from AWS. This is set by NixOps

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-subnet.nix

`tags`
Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-subnet.nix

`vpcId`
The ID of the VPC where the subnet will be created

Type: string or resource of type ‘vpc’

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-subnet.nix

`zone`
The availability zone for the VPC subnet. By default AWS selects one for you.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-subnet.nix

B.2.16. Vpc Route

A Vpc Route is defined by setting `resources.vpcRoutes` *`.name`* to an attribute set containing values for the following options.

Appendix S. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix`

`destinationCidrBlock`

The IPv4 CIDR address block used for the destination match.

Type: null or string

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route.nix`

`destinationIpv6CidrBlock`

The IPv6 CIDR block used for the destination match.

Type: null or string

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route.nix`

`egressOnlyInternetGatewayId`

[IPv6 traffic only] The ID of an egress-only Internet gateway.

Type: null or string or resource of type ‘vpc-egress-only-internet-gateway’

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route.nix`

`gatewayId`

The ID of an Internet gateway or virtual private gateway attached to your VPC.

Type: null or string or resource of type ‘vpc-internet-gateway’

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route.nix`

`instanceId`

The ID of a NAT instance in your VPC. The operation fails if you specify an instance ID unless exactly one network interface is attached.

Type: null or string or EC2 machine

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route.nix`

`name`

Name of the VPC route.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route.nix`

`natGatewayId`

The ID of a NAT gateway.

Type: null or string or resource of type ‘vpc-nat-gateway’

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route.nix`

`networkInterfaceId`

The ID of a network interface.

Type: null or string or resource of type ‘vpc-network-interface’

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route.nix`

`region`

AWS region.

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix`

`routeTableId`

The ID of the VPC route table

Type: string or resource of type ‘vpc-route-table’

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route.nix`

B.2.17. Vpc Route Table

A Vpc Route Table is defined by setting `resources.vpcRouteTables` *`.name`* to an attribute set containing values for the following options.

Appendix T. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix`

`name`

Name of the VPC route table.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route-table.nix`

`propagatingVgws`

A list of VPN gateways for propagation.

Type: list of string or resource of type ‘aws-vpn-gateway’s

Default: `[ ]`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route-table.nix`

`region`

AWS region.

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix`

`tags`

Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route-table.nix

`vpcId`

The ID of the VPC where the route table will be created

Type: string or resource of type ‘vpc’

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route-table.nix

B.2.18. Vpc Route Table Association

A Vpc Route Table Association is defined by setting `resources.vpcRouteTableAssociations .`*name* to an attribute set containing values for the following options.

Appendix U. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`name`

Name of the VPC route table association.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route-table-association.nix

`region`

AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

routeTableId
: The ID of the VPC route table

  Type: string or resource of type ‘vpc-route-table’

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route-table-association.nix

subnetId
: The ID of the VPC subnet where the route table will be associated

  Type: string or resource of type ‘vpc-subnet’

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-route-table-association.nix

B.2.19. Vpc Network Interface

A Vpc Network Interface is defined by setting `resources.vpcNetworkInterfaces` *.name* to an attribute set containing values for the following options.

Appendix V. Configuration Options

accessKeyId
: The AWS Access Key ID.

  Type: string

  Default: `""`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

description
: A description for the network interface.

  Type: string

  Default: `""`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface.nix

name
: Name of the VPC network interface.

  Type: string

  Default: `"nixops-<uuid>-<name>"`

  Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface.nix

`primaryPrivateIpAddress`

The primary private IPv4 address of the network interface. If you don't specify an IPv4 address, Amazon EC2 selects one for you from the subnet's IPv4 CIDR range.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface.nix

`privateIpAddresses`

One or more secondary private IPv4 addresses.

Type: list of strings

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface.nix

`region`

AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`secondaryPrivateIpAddressCount`

The number of secondary private IPv4 addresses to assign to a network interface. When you specify a number of secondary IPv4 addresses, Amazon EC2 selects these IP addresses within the subnet's IPv4 CIDR range. You can't specify this option and specify privateIpAddresses in the same time.

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface.nix

`securityGroups`

The IDs of one or more security groups.

Type: list of string or resource of type ‘ec2-security-group’s

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface.nix

`sourceDestCheck`

Indicates whether source/destination checking is enabled. Default value is true.

Type: boolean

Default: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface.nix

`subnetId`

Subnet Id to create the ENI in.

Type: string or resource of type ‘vpc-subnet’

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface.nix

`tags`

Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface.nix

B.2.20. Vpc Network Interface Attachment

A Vpc Network Interface Attachment is defined by setting `resources.vpcNetworkInterfaceAttachments.name` to an attribute set containing values for the following options.

Appendix W. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`deviceIndex`

The index of the device for the network interface attachment.

Type: signed integer

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface-attachment.nix

`instanceId`

ID of the instance to attach to.

Type: string or EC2 machine

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface-attachment.nix

`name`

Name of the VPC network interface attachment.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface-attachment.nix

`networkInterfaceId`

ENI ID to attach to.

Type: string or resource of type ‘vpc-network-interface’

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-interface-attachment.nix

`region`

AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

B.2.21. Vpc Network Acl

A Vpc Network Acl is defined by setting `resources.vpcNetworkAcls` .*name* to an attribute set containing values for the following options.

Appendix X. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`entries`

The network ACL entries

Type: list of submodules

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`entries.*.cidrBlock`

The IPv4 network range to allow or deny, in CIDR notation.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`entries.*.egress`

Indicates whether this is an egress rule (rule is applied to traffic leaving the subnet).

Type: boolean

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`entries.*.fromPort`

The first port in the range.

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`entries.*.icmpCode`

The ICMP type code to be used.

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`entries.*.icmpType`

The ICMP type to be used.

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`entries.*.ipv6CidrBlock`

The IPv6 network range to allow or deny, in CIDR notation.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`entries.*.protocol`

The protocol to match. If using the -1 'all' protocol, you must specify a from and to port of 0.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`entries.*.ruleAction`

The action to take. Can be either "allow" or "deny".

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`entries.*.ruleNumber`

The rule number of the entry. ACL entries are processed in asceding order by rule number.

Type: signed integer

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`entries.*.toPort`

The last port in the range.

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`name`

Name of the DHCP options set.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`networkAclId`

The network ACL id generated from AWS. This is set by NixOps

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`region`

AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`subnetIds`

A list of subnet IDs to apply to the ACL to.

Type: list of string or resource of type ‘vpc-subnet’s

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`tags`

Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

`vpcId`

The Id of the associated VPC.

Type: string or resource of type ‘vpc’

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-network-acl.nix

B.2.22. Vpc Nat Gateway

A Vpc Nat Gateway is defined by setting `resources.vpcNatGateways .name` to an attribute set containing values for the following options.

Appendix Y. Configuration Options

`accessKeyId`
: The AWS Access Key ID.

  Type: string

  Default: `""`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`allocationId`
: The allocation ID of the elastic IP address.

  Type: string or resource of type ‘elastic-ip’

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-nat-gateway.nix

`name`
: Name of the VPC NAT gateway.

  Type: string

  Default: `"nixops-<uuid>-<name>"`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-nat-gateway.nix

`region`
: AWS region.

  Type: string

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`subnetId`
: The ID of the VPC subnet where the NAT gateway will be created

  Type: string or resource of type ‘vpc-subnet’

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-nat-gateway.nix

B.2.23. Vpc Internet Gateway

A Vpc Internet Gateway is defined by setting `resources.vpcInternetGateways .name` to an attribute set containing values for the following options.

Appendix Z. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`name`

Name of the VPC internet gateway.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-internet-gateway.nix

`region`

AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`tags`

Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-internet-gateway.nix

`vpcId`

The ID of the VPC where the internet gateway will be created

Type: string or resource of type ‘vpc’

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-internet-gateway.nix

B.2.24. Vpc Endpoint

A Vpc Endpoint is defined by setting `resources.vpcEndpoints .name` to an attribute set containing values for the following options.

Appendix AA. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix`

`name`

Name of the VPC endpoint.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-endpoint.nix`

`policy`

A policy to attach to the endpoint that controls access to the service.

Type: null or string

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-endpoint.nix`

`region`

AWS region.

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix`

`routeTableIds`

One or more route table IDs.

Type: list of string or resource of type ‘vpc-route-table’s

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-endpoint.nix

`serviceName`

The AWS service name, in the form com.amazonaws.region.service.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-endpoint.nix

`vpcId`

The ID of the VPC where the endpoint will be created.

Type: string or resource of type ‘vpc’

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-endpoint.nix

B.2.25. Vpc Egress Only Internet Gateway

A Vpc Egress Only Internet Gateway is defined by setting `resources.vpcEgressOnlyInternetGateways.`*`name`* to an attribute set containing values for the following options.

Appendix AB. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`name`

Name of the VPC egress only internet gateway.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-egress-only-internet-gateway.nix

`region`

AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`vpcId`

The ID of the VPC where the internet gateway will be created

Type: string or resource of type ‘vpc’

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-egress-only-internet-gateway.nix`

B.2.26. Vpc Dhcp Options

A Vpc Dhcp Options is defined by setting `resources.vpcDhcpOptions` *`.name`* to an attribute set containing values for the following options.

Appendix AC. Configuration Options

`accessKeyId`

The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix`

`domainName`

If you're using AmazonProvidedDNS in us-east-1, specify ec2.internal. If you're using another region specify region.compute.internal (e.g ap-northeast-1.compute.internal). Otherwise specify a domain name e.g MyCompany.com. This value is used to complete unqualified DNS hostnames.

Type: null or string

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-dhcp-options.nix`

`domainNameServers`

The IP addresses of up to 4 domain name servers, or AmazonProvidedDNS.

Type: null or list of strings

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-dhcp-options.nix`

`name`

Name of the DHCP options set.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-dhcp-options.nix`

`netbiosNameServers`
The IP addresses of up to 4 NetBIOS name servers.

Type: null or list of strings

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-dhcp-options.nix`

`netbiosNodeType`
The NetBIOS node type (1,2,4 or 8).

Type: null or signed integer

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-dhcp-options.nix`

`ntpServers`
The IP addresses of up to 4 Network Time Protocol (NTP) servers.

Type: null or list of strings

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-dhcp-options.nix`

`region`
AWS region.

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix`

`tags`
Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-dhcp-options.nix

`vpcId`
The ID of the VPC used to associate the DHCP options to.

Type: string or resource of type ‘vpc’

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-dhcp-options.nix

B.2.27. Vpc Customer Gateway

A Vpc Customer Gateway is defined by setting `resources.vpcCustomerGateways .name` to an attribute set containing values for the following options.

Appendix AD. Configuration Options

`accessKeyId`
The AWS Access Key ID.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`bgpAsn`
For devices that support BGP, the customer gateway's BGP ASN.

Type: signed integer

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-customer-gateway.nix

`name`
Name of the VPC customer gateway.

Type: string

Default: `"nixops-<uuid>-<name>"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-customer-gateway.nix

`publicIp`
The Internet-routable IP address for the customer gateway's outside interface. The address must be static.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-customer-gateway.nix

`region`

AWS region.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/common-ec2-auth-options.nix

`tags`

Tags assigned to the instance. Each tag name can be at most 128 characters, and each tag value can be at most 256 characters. There can be at most 10 tags.

Type: attribute set of strings

Default: `{ }`

Example: `{ foo = "bar"; xyzzy = "bla"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-customer-gateway.nix

`type`

The type of VPN connection that this customer gateway supports (ipsec.1).

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/vpc-customer-gateway.nix

B.3. GCE Resources

This section lists resource types associated with the Google Compute Engine (GCE) cloud computing environment.

B.3.1. GCE Disks

A GCE Disk is defined by setting `resources.gceDisks.`*`name`* to an attribute set containing values for the following options.

Appendix AE. Configuration Options

`accessKey`

The path to GCE Service Account key. If left empty, it defaults to the contents of the environment variable `ACCESS_KEY_PATH`.

Type: string or path

Default: `""`

Example:

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-disk.nix

diskType

The disk storage type (standard/ssd).

Type: string

Default: `"standard"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-disk.nix

image

The image name or resource from which this disk will be created. If not specified, an empty disk is created. Changing the image name has no effect if the disk already exists.

Type: null or string or resource of type ‘gce-image’

Default: `null`

Example: `"image-2cfda297"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-disk.nix

name

Description of the GCE disk. This is the `Name` tag of the disk.

Type: string

Default: `"n-<uuid>-<name>"`

Example: `"big-fat-disk"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-disk.nix

project

The GCE project which should own the disk. If left empty, it defaults to the contents of the environment variable `GCE_PROJECT`.

Type: string

Default: `""`

Example: `"myproject"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-disk.nix

region

The GCE datacenter in which the disk should be created.

Type: string

Example: `"europe-west1-b"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-disk.nix

`serviceAccount`

The GCE Service Account Email. If left empty, it defaults to the contents of the environment variable `GCE_SERVICE_ACCOUNT`.

Type: string

Default: `""`

Example: `"12345-asdf@developer.gserviceaccount.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-disk.nix

`size`

Disk size (in gigabytes). This may be left unset if you are creating the disk from a snapshot or image, in which case the size of the disk will be equal to the size of the snapshot or image. You can set a size larger than the snapshot or image, allowing the disk to be larger than the snapshot from which it is created.

Type: null or signed integer

Default: `null`

Example: `100`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-disk.nix

`snapshot`

The snapshot name from which this disk will be created. If not specified, an empty disk is created. Changing the snapshot name has no effect if the disk already exists.

Type: null or string

Default: `null`

Example: `"snap-1cbda474"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-disk.nix

B.3.2. GCE Images

A GCE Image is defined by setting `resources.gceImages.`*`name`* to an attribute set containing values for the following options.

Appendix AF. Configuration Options

`accessKey`

The path to GCE Service Account key. If left empty, it defaults to the contents of the environment variable `ACCESS_KEY_PATH`.

Type: string or path

Default: `""`

Example:

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-image.nix

`description`

An optional textual description of the image.

Type: null or string

Default: `null`

Example: `"bootstrap image for the DB node"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-image.nix

`name`

Description of the GCE image. This is the `Name` tag of the image.

Type: string

Default: `"n-<uuid>-<name>"`

Example: `"my-bootstrap-image"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-image.nix

`project`

The GCE project which should own the image. If left empty, it defaults to the contents of the environment variable `GCE_PROJECT`.

Type: string

Default: `""`

Example: `"myproject"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-image.nix

`serviceAccount`

The GCE Service Account Email. If left empty, it defaults to the contents of the environment variable `GCE_SERVICE_ACCOUNT`.

Type: string

Default: `""`

Example: `"12345-asdf@developer.gserviceaccount.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-image.nix

`sourceUri`

The full Google Cloud Storage URL where the disk image is stored.

Type: string

Example: `"gs://nixos-images/nixos-14.10pre-git-x86_64-linux.raw.tar.gz"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-image.nix

B.3.3. GCE Forwarding Rules

A GCE Forwarding Rule is defined by setting `resources.gceForwardingRules.`*`name`* to an attribute set containing values for the following options.

Appendix AG. Configuration Options

`accessKey`

The path to GCE Service Account key. If left empty, it defaults to the contents of the environment variable `ACCESS_KEY_PATH`.

Type: string or path

Default: `""`

Example:

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-forwarding-rule.nix

`description`

An optional textual description of the Fowarding Rule.

Type: null or string

Default: `null`

Example: `"load balancer for the public site"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-forwarding-rule.nix

`ipAddress`

GCE Static IP address resource to bind to or the name of an IP address not managed by NixOps. If left unset, an ephemeral(random) IP address will be assigned on deployment.

Type: null or string or resource of type ‘gce-static-ip’

Default: `null`

Example: `"resources.gceStaticIPs.exampleIP"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-forwarding-rule.nix

`name`

Description of the GCE Forwarding Rule. This is the `Name` tag of the rule.

Type: string

Default: `"n-<uuid>-<name>"`

Example: `"my-public-ip"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-forwarding-rule.nix

`portRange`

If protocol is TCP or UDP, packets addressed to ports in the specified range will be forwarded to the target.

Leave unset to forward all ports.

Type: null or string

Default: `null`

Example: `"1-1000"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-forwarding-rule.nix

`project`

The GCE project which should own the forwarding rule. If left empty, it defaults to the contents of the environment variable `GCE_PROJECT`.

Type: string

Default: `""`

Example: `"myproject"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-forwarding-rule.nix

`protocol`

The IP protocol to which this rule applies.

Acceptable values are: "AH": Specifies the IP Authentication Header protocol. "ESP": Specifies the IP Encapsulating Security Payload protocol. "SCTP": Specifies the Stream Control Transmission Protocol. "TCP": Specifies the Transmission Control Protocol. "UDP": Specifies the User Datagram Protocol.

Type: string

Example: `"TCP"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-forwarding-rule.nix

`publicIPv4`
The assigned IP address of this forwarding rule. This is set by NixOps to the ephemeral IP address of the resource if ipAddress wasn't set, otherwise it should be the same as ipAddress.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-forwarding-rule.nix

`region`
The GCE region to which the forwarding rule should belong.

Type: string

Example: `"europe-west1"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-forwarding-rule.nix

`serviceAccount`
The GCE Service Account Email. If left empty, it defaults to the contents of the environment variable `GCE_SERVICE_ACCOUNT`.

Type: string

Default: `""`

Example: `"12345-asdf@developer.gserviceaccount.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-forwarding-rule.nix

`targetPool`
GCE Target Pool resource to receive the matched traffic or the name of a target pool not managed by NixOps.

Type: string or resource of type ‘gce-target-pool’

Example: `"resources.gceStaticIPs.exampleIP"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-forwarding-rule.nix

B.3.4. GCE HTTP Health Checks

A GCE HTTP Health Check is defined by setting `resources.gceHTTPHealthChecks.name` to an attribute set containing values for the following options.

Appendix AH. Configuration Options

`accessKey`

The path to GCE Service Account key. If left empty, it defaults to the contents of the environment variable `ACCESS_KEY_PATH`.

Type: string or path

Default: `""`

Example:

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

`checkInterval`

How often (in seconds) to send a health check.

Type: signed integer

Default: `5`

Example: `20`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

`description`

An optional textual description of the HTTP Health Check.

Type: null or string

Default: `null`

Example: `"health check for databases"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

`healthyThreshold`

An unhealthy VM will be marked healthy after this many consecutive successes.

Type: signed integer

Default: `2`

Example: `4`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

`host`

The value of the host header in the HTTP health check request. If left unset(default value), the public IP on behalf of which this health check is performed will be used.

Type: null or string

Default: `null`

Example: `"healthcheckhost.org"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

`name`

Description of the GCE HTTP Health Check. This is the `Name` tag of the health check.

Type: string

Default: `"n-<uuid>-<name>"`

Example: `"my-health-check"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

`path`

The request path of the HTTP health check request.

Type: string

Default: `"/"`

Example: `"/is_healthy"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

`port`

The TCP port number for the HTTP health check request.

Type: signed integer

Default: `80`

Example: `8080`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

`project`

The GCE project which should own the HTTP health check. If left empty, it defaults to the contents of the environment variable `GCE_PROJECT`.

Type: string

Default: `""`

Example: `"myproject"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

`serviceAccount`

The GCE Service Account Email. If left empty, it defaults to the contents of the environment variable `GCE_SERVICE_ACCOUNT`.

Type: string

Default: `""`

Example: `"12345-asdf@developer.gserviceaccount.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

`timeout`

How long (in seconds) to wait before claiming failure.

Type: signed integer

Default: `5`

Example: `20`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

`unhealthyThreshold`

A so-far healthy VM will be marked unhealthy after this many consecutive failures.

Type: signed integer

Default: `2`

Example: `4`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-http-health-check.nix

B.3.5. GCE Networks

A GCE Network is defined by setting `resources.gceNetworks.`*`name`* to an attribute set containing values for the following options.

Appendix AI. Configuration Options

`accessKey`

The path to GCE Service Account key. If left empty, it defaults to the contents of the environment variable `ACCESS_KEY_PATH`.

Type: string or path

Default: `""`

Example:

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-network.nix

`addressRange`

The range of internal addresses that are legal on this network. This range is a CIDR specification.

Type: string

Example: `"192.168.0.0/16"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-network.nix

`firewall`

Firewall rules.

Type: attribute set of submodules

Default: `{ allow-ssh = { allowed = { tcp = [ 22 ] ; } ; } ; }`

Example: `{ allow-http = { allowed = { tcp = [ 80 ] ; } ; sourceRanges = [ "0.0.0.0/0" ] ; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-network.nix

`firewall.<name>.allowed`

Allowed protocols and ports. Setting protocol to null for example "icmp = null" allows all connections made using the protocol to proceed.";

Type: attribute set of null or list of string or signed integerss

Example: `{ icmp = null; tcp = [ 80 ] ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-network.nix

`firewall.<name>.sourceRanges`

The address blocks that this rule applies to, expressed in CIDR format. An inbound connection is allowed if either the range or the tag of the source matches the `sourceRanges` or `sourceTags`. As a convenience, leaving this option unset is equivalent to setting it to ["0.0.0.0/0"].

Type: null or list of strings

Default: `null`

Example: `[ "192.168.0.0/16" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-network.nix

`firewall.<name>.sourceTags`

A list of instance tags which this rule applies to. Can be set in addition to `sourceRanges`. An inbound connection is allowed if either the range or the tag of the source matches the `sourceRanges` or `sourceTags`.

Don't forget to set `sourceRanges` to [] or at least a more restrictive range because the default setting makes `sourceTags` irrelevant.

Type: list of strings

Default: `[ ]`

Example: `[ "admin" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-network.nix

`firewall.<name>.targetTags`

A list of instance tags indicating sets of instances located on the network which may make network connections as specified in `allowed`. If no `targetTags` are specified, the firewall rule applies to all instances on the network.

Type: list of strings

Default: `[ ]`

Example: `[ "public-http" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-network.nix

`name`

Description of the GCE Network. This is the `Name` tag of the network.

Type: string

Default: `"n-<uuid>-<name>"`

Example: `"my-custom-network"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-network.nix

`project`

The GCE project which should own the network. If left empty, it defaults to the contents of the environment variable `GCE_PROJECT`.

Type: string

Default: `""`

Example: `"myproject"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-network.nix

`serviceAccount`

The GCE Service Account Email. If left empty, it defaults to the contents of the environment variable `GCE_SERVICE_ACCOUNT`.

Type: string

Default: `""`

Example: `"12345-asdf@developer.gserviceaccount.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-network.nix

B.3.6. GCE Static IPs

A GCE Static IP is defined by setting `resources.gceStaticIPs.`*`name`* to an attribute set containing values for the following options.

Appendix AJ. Configuration Options

`accessKey`

The path to GCE Service Account key. If left empty, it defaults to the contents of the environment variable `ACCESS_KEY_PATH`.

Type: string or path

Default: `""`

Example:

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-static-ip.nix

`ipAddress`

The specific ephemeral IP address to promote to a static one.

This lets you permanently reserve an ephemeral address used by one of resources to preserve it across machine teardowns or reassign it to another resource. Changing value of, setting or unsetting this option has no effect once the address resource is deployed, thus you can't lose the static IP unless you explicitly destroy it.

Type: null or string

Default: `null`

Example: `"123.123.123.123"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-static-ip.nix

`name`

Description of the GCE static IP address. This is the `Name` tag of the address.

Type: string

Default: `"n-<uuid>-<name>"`

Example: `"my-public-ip"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-static-ip.nix

`project`

The GCE project which should own the IP address. If left empty, it defaults to the contents of the environment variable `GCE_PROJECT`.

Type: string

Default: `""`

Example: `"myproject"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-static-ip.nix

`publicIPv4`

The static IP address assigned. This is set by NixOps to the ephemeral IP address of the resource if ipAddress wasn't set, otherwise it should be the same as ipAddress.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-static-ip.nix

`region`

The GCE region to which the IP address should be bound.

Type: string

Example: `"europe-west1"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-static-ip.nix

`serviceAccount`

The GCE Service Account Email. If left empty, it defaults to the contents of the environment variable `GCE_SERVICE_ACCOUNT`.

Type: string

Default: `""`

Example: `"12345-asdf@developer.gserviceaccount.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-static-ip.nix

B.3.7. GCE Target Pools

A GCE Target Pool is defined by setting `resources.gceTargetPools.name` to an attribute set containing values for the following options.

Appendix AK. Configuration Options

`accessKey`

The path to GCE Service Account key. If left empty, it defaults to the contents of the environment variable `ACCESS_KEY_PATH`.

Type: string or path

Default: `""`

Example:

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-target-pool.nix

`healthCheck`

GCE HTTP Health Check resource or name of a HTTP Health Check resource not managed by NixOps.

A member VM in this pool is considered healthy if and only if the specified health checks passes. Unset health check means all member virtual machines will be considered healthy at all times but the health status of this target pool will be marked as unhealthy to indicate that no health checks are being performed.

Type: null or string or resource of type ‘gce-http-health-check’

Default: `null`

Example: `"resources.gceHTTPHealthChecks.my-check"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-target-pool.nix

`machines`

The list of machine resources or fully-qualified GCE Node URLs to add to this pool.

Type: list of string or GCE machines

Default: `[ ]`

Example: `[ "machines.httpserver1" "machines.httpserver2" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-target-pool.nix

`name`

Description of the GCE Target Pool. This is the `Name` tag of the target pool.

Type: string

Default: `"n-<uuid>-<name>"`

Example: `"my-target-pool"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-target-pool.nix`

`project`
The GCE project which should own the target pool. If left empty, it defaults to the contents of the environment variable `GCE_PROJECT`.

Type: string

Default: `""`

Example: `"myproject"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-target-pool.nix`

`region`
The GCE region to where the GCE Target Pool instances should reside.

Type: string

Example: `"europe-west1"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-target-pool.nix`

`serviceAccount`
The GCE Service Account Email. If left empty, it defaults to the contents of the environment variable `GCE_SERVICE_ACCOUNT`.

Type: string

Default: `""`

Example: `"12345-asdf@developer.gserviceaccount.com"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gce-target-pool.nix`

B.3.8. GSE Buckets

A GSE Bucket is defined by setting `resources.gseBuckets.`*`name`* to an attribute set containing values for the following options.

Appendix AL. Configuration Options

`accessKey`

The path to GCE Service Account key. If left empty, it defaults to the contents of the environment variable `ACCESS_KEY_PATH`.

Type: string or path

Default: `""`

Example:

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`cors`

Cross-Origin Resource Sharing configuration.

Type: list of submodules

Default: `[ ]`

Example: `[ { maxAgeSeconds = 100; methods = [ "GET" "PUT" ] ; origins = [ "http://site.com" "http://site.org" ] ; responseHeaders = [ "header1" "header2" ] ; } ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`cors.*.maxAgeSeconds`

The value, in seconds, to return in the Access-Control-Max-Age header used in preflight responses.

Type: null or signed integer

Default: `3600`

Example: `360`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`cors.*.methods`

The list of HTTP methods on which to include CORS response headers, (GET, OPTIONS, POST, etc). Note: "*" is permitted in the list, and means "any method".

Type: list of strings

Default: `[ ]`

Example: `[ "GET" "POST" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`cors.*.origins`

The list of Origins eligible to receive CORS response headers. Note: "*" is permitted in the list, and means "any Origin".

Type: list of strings

Default: `[ ]`

Example: `[ "http://example.org" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`cors.*.responseHeaders`

The list of HTTP headers other than the simple response headers to give permission for the user-agent to share across domains.

Type: list of strings

Default: `[ ]`

Example: `[ "FIXME" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`lifecycle`

Object Lifecycle Configuration for the bucket contents.

Type: list of submodules

Default: `[ ]`

Example: `[ { conditions = { age = 40; } ; } ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`lifecycle.*.action`

The action to perform when all conditions are met. Currently only "Delete" is supported by GCE.

Type: string

Default: `"Delete"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`lifecycle.*.conditions.age`

This condition is satisfied when an object reaches the specified age (in days).

Type: null or signed integer

Default: `null`

Example: `365`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`lifecycle.*.conditions.createdBefore`

This condition is satisfied when an object is created before midnight of the specified date in UTC.

Type: null or string

Default: `null`

Example: `"2013-01-10"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`lifecycle.*.conditions.isLive`

Relevant only for versioned objects. If the value is true, this condition matches the live objects; if the value is false, it matches archived objects.

Type: null or boolean

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`lifecycle.*.conditions.numberOfNewerVersions`

Relevant only for versioned objects. If the value is N, this condition is satisfied when there are at least N versions (including the live version) newer than this version of the object. For live objects, the number of newer versions is considered to be 0. For the most recent archived version, the number of newer versions is 1 (or 0 if there is no live object), and so on.

Type: null or signed integer

Default: `null`

Example: `3`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`location`

Object data for objects in the bucket resides in physical storage within this region. Defaults to US. See the developer's guide for the authoritative list.

Type: string

Default: `"US"`

Example: `"EU"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`logging.logBucket`

The destination bucket where the current bucket's logs should be placed.

FIXME: is this a bucket name or a fully-qualified url?

Type: null or string or resource of type ‘gse-bucket’

Default: `null`

Example: `"resources.gseBuckets.logBucket"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`logging.logObjectPrefix`

A prefix for log object names.

Type: null or string

Default: `null`

Example: `"log"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`name`

This is the `Name` tag of the bucket.

Type: string

Default: `"n-<uuid>-<name>"`

Example: `"my-bucket"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`project`

The GCE project which should own the bucket. If left empty, it defaults to the contents of the environment variable `GCE_PROJECT`.

Type: string

Default: `""`

Example: `"myproject"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`serviceAccount`

The GCE Service Account Email. If left empty, it defaults to the contents of the environment variable `GCE_SERVICE_ACCOUNT`.

Type: string

Default: `""`

Example: `"12345-asdf@developer.gserviceaccount.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`storageClass`

This defines how objects in the bucket are stored and determines the SLA and the cost of storage. Typical values are STANDARD and DURABLE_REDUCED_AVAILABILITY. See the developer's guide for the authoritative list.

Type: string

Default: `"STANDARD"`

Example: `"DURABLE_REDUCED_AVAILABILITY"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`versioning.enabled`

While set to true, versioning is fully enabled for this bucket.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`website.mainPageSuffix`

Behaves as the bucket's directory index where missing objects are treated as potential directories.

For example, with mainPageSuffix main_page_suffix configured to be index.html, a GET request for http://example.com would retrieve http://example.com/index.html, and a GET request for http://example.com/photos would retrieve http://example.com/photos/index.html.

Type: null or string

Default: `null`

Example: `"index.html"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

`website.notFoundPage`

Serve this object on request for a non-existent object.

Type: null or string

Default: `null`

Example: `"404.html"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/gse-bucket.nix

B.4. Azure Resources

> ***Warning*** *The Azure backend in Nixops is now disabled. See PR#1131*
>
> *For existing deployments, Azure backend is supported in Nixops up to release 1.6.1 only.*

This section lists resource types associated with the Microsoft Azure (Azure) cloud computing environment.

B.4.1. Azure Availability Sets

An Azure Availability Set is defined by setting `resources.azureAvailabilitySets.name` to an attribute set containing values for the following options.

Appendix AM. Configuration Options

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-availability-set.nix

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-availability-set.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-availability-set.nix

identifierUri
The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-availability-set.nix

location
The Azure data center location where the availability set should be created.

Type: string

Example: `"westus"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-availability-set.nix

name
Name of the Azure availability set.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-availability-set"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-availability-set.nix

platformFaultDomainCount
The number of update domains that are used. A single hardware failure can only affect virtual machines in one fault domain. A maximum of 3 fault domains can be used.

Type: signed integer

Default: `3`

Example: `3`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-availability-set.nix

platformUpdateDomainCount
The number of update domains that are used. Only one of the update domains can be rebooted or unavailable at once during planned maintenance. A maximum of 20 update domains can be used.

Type: signed integer

Default: `5`

Example: `10`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-availability-set.nix

`resourceGroup`

The name or resource of an Azure resource group to create the availability set in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-availability-set.nix

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-availability-set.nix

`tags`

Tag name/value pairs to associate with the availability set.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-availability-set.nix

B.4.2. Azure BLOB Containers

An Azure BLOB Container is defined by setting `resources.azureBlobContainers.`*`name`* to an attribute set containing values for the following options.

Appendix AN. Configuration Options

`accessKey`

Access key for the storage service if not managed by NixOps.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob-container.nix

`acl.blobPublicAccess`

Permissions for the container: null(private), 'container'(anonymous clients can enumerate and read all BLOBs) or 'blob'(anonymous clients can read but can't enumerate BLOBs in the container).

Type: one of <null>, "container", "blob"

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob-container.nix

`acl.signedIdentifiers`

An attribute set of Signed Identifiers and the corresponding access policies that may be used with Shared Access Signatures.

Type: attribute set of submodules

Default: `{ }`

Example: `{ MTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTI= = { expiry = "2013-11-27T08:49:37.0000000Z"; permissions = "raud"; start = "2013-11-26T08:49:37.0000000Z"; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob-container.nix

`acl.signedIdentifiers.<name>.expiry`

Access policy expiry UTC date/time in a valid ISO 8061 format. Supported ISO 8061 formats include the following: YYYY-MM-DD, YYYY-MM-DDThh:mmTZD, YYYY-MM-DDThh:mm:ssTZD, YYYY-MM-DDThh:mm:ss.ffffffTZD

Type: string

Example: `"2013-11-26T08:49:37.0000000Z"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob-container.nix

`acl.signedIdentifiers.<name>.permissions`

Abbreviated permission list.

Type: string

Example: `"raud"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob-container.nix

`acl.signedIdentifiers.<name>.start`

Access policy start UTC date/time in a valid ISO 8061 format. Supported ISO 8061 formats include the following: YYYY-MM-DD, YYYY-MM-DDThh:mmTZD, YYYY-MM-DDThh:mm:ssTZD, YYYY-MM-DDThh:mm:ss.ffffffTZD

Type: string

Example: `"2013-11-26T08:49:37.0000000Z"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob-container.nix

`metadata`

Metadata name/value pairs to associate with the container.

Type: attribute set of strings

Default: `{ }`

Example: `{ loglevel = "warn"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob-container.nix

`name`

Description of the Azure BLOB container. Must include only lower-case characters. This is the `Name` tag of the container.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-blob-container"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob-container.nix

`storage`

The name or resource of an Azure storage in which the container is to be created.

Type: string or resource of type ‘azure-storage’

Example: `"xxx-my-storage"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob-container.nix

B.4.3. Azure BLOBs

An Azure BLOB is defined by setting `resources.azureBlobs.`*`name`* to an attribute set containing values for the following options.

Appendix AO. Configuration Options

`accessKey`

Access key for the storage service if the container is not managed by NixOps.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix

`blobType`

BLOB type: BlockBlob or PageBlob.

Type: one of "BlockBlob", "PageBlob"

Default: `"BlockBlob"`

Example: `"PageBlob"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix

`cacheControl`

The Blob service stores this value but does not use or modify it.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix

`container`

The name or resource of an Azure BLOB container in which the BLOB is to be stored.

Type: string or resource of type ‘azure-blob-container’

Example: `"xxx-my-container"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix

`contentDisposition`

The Content-Disposition response header field conveys additional information about how to process the response payload, and also can be used to attach additional metadata. For example, if set to "attachment", Content-Disposition indicates that the user-agent should not display the response, but instead show a Save As dialog.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix

`contentEncoding`

Specifies which content encodings have been applied to the blob. This value is returned to the client when the Get Blob (REST API) operation is performed on the blob resource. The client can use this value when returned to decode the blob content.

Type: null or string

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix`

`contentLanguage`

Specifies the natural languages used by this resource.

Type: null or string

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix`

`contentType`

The MIME content type of the BLOB.

Type: null or string

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix`

`copyFromBlob`

Create the BLOB by copying the contents of an existing one. Any BLOB in your subscription or a publicly-accessible BLOB in another subscription can be copied.

Type: null or string

Default: `null`

Example: `"https://myaccount.blob.core.windows.net/mycontainer/myblob"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix`

`filePath`

Path to the local file to upload.

Type: null or string

Default: `null`

Example: `"path/to/source/file"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix`

`metadata`

Metadata name/value pairs to associate with the BLOB.

Type: attribute set of strings

Default: `{ }`

Example: `{ loglevel = "warn"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix

`name`

Description of the Azure BLOB. This is the `Name` tag of the BLOB.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-blob"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix

`storage`

The name or resource of an Azure storage if the container is not managed by NixOps.

Type: null or string or resource of type ‘azure-storage’

Default: `null`

Example: `"xxx-my-storage"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-blob.nix

B.4.4. Azure Directories

An Azure Directory is defined by setting `resources.azureDirectories.`*`name`* to an attribute set containing values for the following options.

Appendix AP. Configuration Options

`accessKey`

Access key for the storage service if not managed by NixOps.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-directory.nix

`metadata`
: Metadata name/value pairs to associate with the directory.

  Type: attribute set of strings

  Default: `{ }`

  Example: `{ loglevel = "warn"; }`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-directory.nix

`name`
: Description of the Azure directory. This is the `Name` tag of the directory.

  Type: string

  Default: `"nixops-<uuid>-<name>"`

  Example: `"my-directory"`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-directory.nix

`parentDirectory`
: The name or resource of an Azure directory in which the directory is to be created. Must specify at least one of parentDirectory or share.

  Type: null or string or resource of type ‘azure-directory’

  Default: `null`

  Example: `"xxx-my-directory"`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-directory.nix

`parentDirectoryPath`
: The path to the parent directory in which the directory is to be created. Should only be used if the parent directory is not managed by NixOps. Must also specify Azure share.

  Type: null or string

  Default: `null`

  Example: `"dir1/dir2"`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-directory.nix

`share`
: The name or resource of an Azure share in which the directory is to be created. Must specify at least one of parentDirectory or share.

  Type: null or string or resource of type ‘azure-share’

Default: `null`

Example: `"xxx-my-share"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-directory.nix

`storage`

The name or resource of an Azure storage in which the directory is to be created. Optional if parentDirectory or share are managed by NixOps.

Type: null or string or resource of type ‘azure-storage’

Default: `null`

Example: `"xxx-my-storage"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-directory.nix

B.4.5. Azure DNS Record Sets

An Azure DNS Record Set is defined by setting `resources.azureDNSRecordSets.`*`name`* to an attribute set containing values for the following options.

Appendix AQ. Configuration Options

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-record-set.nix

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-record-set.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable

`AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-record-set.nix

`dnsZone`

The Azure Resource Id or NixOps resource of the DNS zone to create the record set in.

Type: string or resource of type ‘azure-dns-zone’

Example: `"resources.azureDNSZones.test-com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-record-set.nix

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-record-set.nix

`name`

Name of the Azure DNS record set. Use "@" for RecordSets at the apex of the zone (e.g. SOA/NS).

Type: string

Example: `"test.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-record-set.nix

`properties`

Record properties depending on record type. See Azure documentation for DNS record sets.

Type: unspecified

Example: `{ CNAMERecord = { cname = "test.com"; } ; TTL = 300; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-record-set.nix

`recordType`
: DNS record type. Allowed values are: A, AAAA, CNAME, MX, SOA, NS, SRV, TXT.

  Type: string

  Example: `"CNAME"`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-record-set.nix

`subscriptionId`
: The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

  Type: string

  Default: `""`

  Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-record-set.nix

`tags`
: Tag name/value pairs to associate with the DNS record set.

  Type: attribute set of strings

  Default: `{ }`

  Example: `{ environment = "production"; }`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-record-set.nix

B.4.6. Azure DNS Zones

An Azure DNS Zone is defined by setting `resources.azureDNSZones.`*`name`* to an attribute set containing values for the following options.

Appendix AR. Configuration Options

`appId`
: The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

  Type: string

  Default: `""`

  Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

  Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-zone.nix

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-zone.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-zone.nix

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-zone.nix

`name`

Name of the Azure DNS zone.

Type: string

Example: `"test.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-zone.nix

`resourceGroup`

The name or resource of an Azure resource group to create the DNS zone in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-zone.nix`

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-zone.nix`

`tags`

Tag name/value pairs to associate with the DNS zone.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-dns-zone.nix`

B.4.7. Azure ExpressRoute Circuits

An Azure ExpressRoute Circuit is defined by setting `resources.azureExpressRouteCircuits.`*`name`* to an attribute set containing values for the following options.

Appendix AS. Configuration Options

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix`

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

`bandwidth`

Value of ExpressRoute circuit bandwidth in Mbps. This must match one of the bandwidths offered for the chosen service provider from the list returned by "azure network express-route provider list".

Type: signed integer

Example: `100`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

`location`

The Azure data center location where the ExpressRoute circuit should be created.

Type: string

Example: `"westus"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

`name`

Name of the Azure ExpressRoute circuit.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-express-route-circuit"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

peeringLocation

Peering location for the ExpressRoute Circuit. This must match one of the peering locations for the chosen service provider from the list returned by "azure network express-route provider list".

Type: string

Example: `"Amsterdam"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

peerings

Attribute set of BGP peering properties. The property list and allowed values deepend on the peering type. See Azure ExpressRoute documentation for more info.

Type: attribute set of attribute sets

Default: `{ }`

Example: `{ AzurePublicPeering = { peerASN = 100; peeringType = "AzurePublicPeering"; primaryPeerAddressPrefix = "192.168.1.0/30"; secondaryPeerAddressPrefix = "192.168.2.0/30"; vlanId = 200; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

resourceGroup

The name or resource of an Azure resource group to create the ExpressRoute circuit in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

serviceProviderName

The name of the service provider. This must match the provider name returned by "azure network express-route provider list".

Type: string

Example: `"FakeProvider"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

`sku.family`

The family of the SKU of the ExpressRoute circuit.

Type: one of "MeteredData", "UnlimitedData"

Example: `"UnlimitedData"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

`sku.tier`

The tier of the SKU of the ExpressRoute circuit.

Type: one of "Standard", "Premium"

Example: `"Premium"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

`tags`

Tag name/value pairs to associate with the ExpressRoute circuit.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-express-route-circuit.nix

B.4.8. Azure Files

An Azure File is defined by setting `resources.azureFiles.`*`name`* to an attribute set containing values for the following options.

Appendix AT. Configuration Options

`accessKey`

Access key for the storage service if the container is not managed by NixOps.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`cacheControl`

The File service stores this value but does not use or modify it.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`contentDisposition`

The Content-Disposition response header field conveys additional information about how to process the response payload, and also can be used to attach additional metadata. For example, if set to "attachment", Content-Disposition indicates that the user-agent should not display the response, but instead show a Save As dialog.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`contentEncoding`

Specifies which content encodings have been applied to the file. This value is returned to the client when the Get File operation is performed on the file resource and can be used to decode the file content.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`contentLanguage`

Specifies the natural languages used by this resource.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`contentType`

The MIME content type of the file.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`directory`

The name or resource of an Azure directory in which the file is to be created. If not specified, the file will be created in the root of the share. Must specify at least one of directory or share.

Type: null or string or resource of type ‘azure-directory’

Default: `null`

Example: `"xxx-my-directory"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`directoryPath`

The path to the directory in which the file is to be created. If not specified, the file will be created in the root of the share. Must also specify Azure share.

Type: null or string

Default: `null`

Example: `"dir1/dir2"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`filePath`

Path to the local file to upload.

Type: string

Example: `"path/to/source/file"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`metadata`

Metadata name/value pairs to associate with the File.

Type: attribute set of strings

Default: `{ }`

Example: `{ loglevel = "warn"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`name`

Description of the Azure file. This is the `Name` tag of the file.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-file"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`share`

The name or resource of an Azure share in which the file is to be stored. Must specify at least one of directory or share.

Type: null or string or resource of type ‘azure-share’

Default: `null`

Example: `"xxx-my-share"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

`storage`

The name or resource of an Azure storage if the share is not managed by NixOps.

Type: null or string or resource of type ‘azure-storage’

Default: `null`

Example: `"xxx-my-storage"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-file.nix

B.4.9. Azure Gateway Connections

An Azure Gateway Connection is defined by setting `resources.azureGatewayConnections.`*`name`* to an attribute set containing values for the following options.

Appendix AU. Configuration Options

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix`

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix`

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix`

`connectionType`

The connection type of the virtual network gateway connection.

Type: string

Example: `"Vnet2Vnet"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix`

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix`

`localNetworkGateway2`

The Azure Resource Id or NixOps resource of the second local network gateway in the connection.

Type: null or string or resource of type ‘azure-local-network-gateway’

Default: `null`

Example: `"xxx-my-vnet-gateway"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix

`location`
The Azure data center location where the virtual network gateway connection should be created.

Type: string

Example: `"westus"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix

`name`
Name of the Azure virtual network gateway connection.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-gateway-connection"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix

`resourceGroup`
The name or resource of an Azure resource group to create the virtual network gateway connection in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix

`routingWeight`
The routing weight of the virtual network gateway connection.

Type: signed integer

Example: `10`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix

`sharedKey`
IPSec shared key for the connection. Leave empty to generate automaticaly.

Type: null or string

Default: `null`

Example: `"wNEf6Vkw0Ijx2vNvdQohbZtDCaoDYqE8"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix`

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix`

`tags`

Tag name/value pairs to associate with the virtual network gateway connection.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix`

`virtualNetworkGateway1`

The Azure Resource Id or NixOps resource of the first virtual network gateway in the connection.

Type: null or string or resource of type ‘azure-virtual-network-gateway’

Default: `null`

Example: `"xxx-my-vnet-gateway"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix`

`virtualNetworkGateway2`

The Azure Resource Id or NixOps resource of the second virtual network gateway in the connection.

Type: null or string or resource of type ‘azure-virtual-network-gateway’

Default: `null`

Example: `"xxx-my-vnet-gateway"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-gateway-connection.nix`

B.4.10. Azure Load Balancers

An Azure Load Balancer is defined by setting `resources.azureLoadBalancers.name` to an attribute set containing values for the following options.

Appendix AV. Configuration Options

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix`

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix`

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix`

`backendAddressPools`

The list of names of backend address pools to create

Type: list of strings

Default: `[ "default" ]`

Example: `[ "website" "db" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`frontendInterfaces`

An attribute set of frontend network interfaces.

Type: attribute set of submodules

Default: `{ }`

Example: `{ default = { publicIpAddress = "my-reserved-address"; subnet = { network = "my-virtual-network"; } ; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`frontendInterfaces.<name>.privateIpAddress`

The static private IP address to reserve for the load balancer frontend interface. The address must be in the address space of `subnet`. Leave empty to auto-assign.

Type: null or string

Default: `null`

Example: `"10.10.10.10"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`frontendInterfaces.<name>.publicIpAddress`

The Azure Resource Id or NixOps resource of an Azure reserved IP address resource to use for the frontend interface. Leave empty to create an internal load balancer interface.

Type: null or string or resource of type ‘azure-reserved-ip-address’

Default: `null`

Example: `"my-reserved-ip"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`frontendInterfaces.<name>.subnet.name`

The name of the subnet of `network` in which to obtain the private IP address.

Type: string

Default: `"default"`

Example: `"my-subnet"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`frontendInterfaces.<name>.subnet.network`

The Azure Resource Id or NixOps resource of an Azure virtual network that contains the subnet.

Type: null or string or resource of type ‘azure-virtual-network’

Default: `null`

Example: `"my-network"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`inboundNatRules`

An attribute set of inbound NAT rules.

Type: attribute set of submodules

Default: `{ }`

Example: `{ admin-ssh = { backendPort = 22; frontendPort = 2201; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`inboundNatRules.<name>.backendPort`

The port used for internal connections on the endpoint. Possible values range between 1 and 65535, inclusive.

Type: signed integer

Example: `80`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`inboundNatRules.<name>.enableFloatingIp`

Floating IP is pertinent to failover scenarios: a "floating" IP is reassigned to a secondary server in case the primary server fails. Floating IP is required for SQL AlwaysOn.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`inboundNatRules.<name>.frontendInterface`

The name of a frontend interface over which this Inbound NAT Rule operates.

Type: string

Default: `"default"`

Example: `"webservers"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`inboundNatRules.<name>.frontendPort`

The port for the external endpoint. Port numbers for each Rule must be unique within the Load Balancer. Possible values range between 1 and 65535, inclusive.

Type: signed integer

Example: `80`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`inboundNatRules.<name>.idleTimeout`

Specifies the timeout in minutes for the Tcp idle connection. The value can be set between 4 and 30 minutes. This property is only used when the protocol is set to `Tcp`.

Type: signed integer

Default: `4`

Example: `30`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`inboundNatRules.<name>.protocol`

The transport protocol for the external endpoint. Possible values are Udp or Tcp.

Type: one of "Tcp", "Udp"

Default: `"Tcp"`

Example: `"Udp"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`loadBalancingRules`

An attribute set of load balancer rules.

Type: attribute set of submodules

Default: `{ }`

Example: `{ website = { backendPort = 8080; frontendPort = 80; probe = "web"; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`loadBalancingRules.<name>.backendAddressPool`
The name of a backend address pool over which this Load Balancing Rule operates.

Type: string

Default: `"default"`

Example: `"webservers"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`loadBalancingRules.<name>.backendPort`
The port used for internal connections on the endpoint. Possible values range between 1 and 65535, inclusive.

Type: signed integer

Example: `80`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`loadBalancingRules.<name>.enableFloatingIp`
Floating IP is pertinent to failover scenarios: a "floating" IP is reassigned to a secondary server in case the primary server fails. Floating IP is required for SQL AlwaysOn.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`loadBalancingRules.<name>.frontendInterface`
The name of a frontend interface over which this Load Balancing Rule operates.

Type: string

Default: `"default"`

Example: `"webservers"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`loadBalancingRules.<name>.frontendPort`

The port for the external endpoint. Port numbers for each Rule must be unique within the Load Balancer. Possible values range between 1 and 65535, inclusive.

Type: signed integer

Example: `80`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`loadBalancingRules.<name>.idleTimeout`

Specifies the timeout in minutes for the Tcp idle connection. The value can be set between 4 and 30 minutes. This property is only used when the protocol is set to `Tcp`.

Type: signed integer

Default: `4`

Example: `30`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`loadBalancingRules.<name>.loadDistribution`

Specifies the load balancing distribution type to be used by the Load Balancer Rule. Possible values are: Default - The load balancer is configured to use a 5 tuple hash to map traffic to available servers; SourceIP - The load balancer is configured to use a 2 tuple hash to map traffic to available servers; SourceIPProtocol - The load balancer is configured to use a 3 tuple hash to map traffic to available servers.

Type: one of "Default", "SourceIP", "SourceIPProtocol"

Default: `"Default"`

Example: `"SourceIP"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`loadBalancingRules.<name>.probe`

The name of a probe used by this Load Balancing Rule.

Type: null or string

Default: `null`

Example: `"webservers"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`loadBalancingRules.<name>.protocol`

The transport protocol for the external endpoint. Possible values are Udp or Tcp.

Type: one of "Tcp", "Udp"

Default: `"Tcp"`

Example: `"Udp"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`location`

The Azure data center location where the load balancer should be created.

Type: string

Example: `"westus"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`name`

Name of the Azure load balancer.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-network"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`probes`

An attribute set of load balancer probes

Type: attribute set of submodules

Default: `{ }`

Example: `{ web = { path = "/is-alive"; port = 8080; protocol = "http"; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`probes.<name>.interval`

The interval, in seconds, between probes to the backend endpoint for health status. The minimum allowed value is 5.

Type: signed integer

Default: `15`

Example: `5`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`probes.<name>.numberOfProbes`

The number of failed probe attempts after which the backend endpoint is removed from rotation. The default value is 2. NumberOfProbes multiplied by interval value must be greater or equal to 10. Endpoints are returned to rotation when at least one probe is successful.

Type: signed integer

Default: `2`

Example: `5`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`probes.<name>.path`

The URI used for requesting health status from the backend endpoint. Used if protocol is set to http.

Type: null or string

Default: `null`

Example: `"/is-up"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`probes.<name>.port`

Port on which the Probe queries the backend endpoint. Possible values range from 1 to 65535, inclusive.

Type: signed integer

Example: `80`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`probes.<name>.protocol`

Specifies the protocol of the probe request. Possible values are Http or Tcp. If Tcp is specified, a received ACK is required for the probe to be successful. If Http is specified, a 200 OK response from the specified URI is required for the probe to be successful.

Type: one of "Tcp", "Http"

Default: `"Tcp"`

Example: `"Http"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`resourceGroup`

The name or resource of an Azure resource group to create the load balancer in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

`tags`

Tag name/value pairs to associate with the load balancer.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-load-balancer.nix

B.4.11. Azure Local Network Gateways

An Azure Local Network Gateway is defined by setting `resources.azureLocalNetworkGateways.`*`name`* to an attribute set containing values for the following options.

Appendix AW. Configuration Options

`addressSpace`

List the address prefixes in CIDR notation of the local network site. Traffic addressed at these prefixes will be routed to the local network site.

Type: list of strings

Example: `"10.1.0.0/24"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-local-network-gateway.nix

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-local-network-gateway.nix

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-local-network-gateway.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-local-network-gateway.nix

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-local-network-gateway.nix

`ipAddress`

The public IP address of the local network gateway.

Type: string

Example: `"20.20.20.20"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-local-network-gateway.nix

`location`

The Azure data center location where the local network gateway should be created.

Type: string

Example: `"westus"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-local-network-gateway.nix

`name`

Name of the Azure local network gateway.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-local-network-gateway"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-local-network-gateway.nix

`resourceGroup`

The name or resource of an Azure resource group to create the local network gateway in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-local-network-gateway.nix

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-local-network-gateway.nix

`tags`

Tag name/value pairs to associate with the local network gateway.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-local-network-gateway.nix

B.4.12. Azure Network Security Groups

An Azure Network Security Group is defined by setting `resources.azureSecurityGroups.`*`name`* to an attribute set containing values for the following options.

Appendix AX. Configuration Options

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`location`

The Azure data center location where the network security group should be created.

Type: string

Example: `"westus"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`name`

Name of the Azure network security group.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-security-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`resourceGroup`

The name or resource of an Azure resource group to create the network security group in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`securityRules`

An attribute set of security rules.

Type: attribute set of submodules

Default: `{ }`

Example: `{ allow-ssh = { access = "Allow"; description = "Allow SSH"; destinationAddressPrefix = "*"; destinationPortRange = "22"; direction = "Inbound"; priority = 2000; protocol = "Tcp"; sourceAddressPrefix = "Internet"; sourcePortRange = "*"; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`securityRules.<name>.access`

Specifies whether network traffic is allowed or denied. Possible values are "Allow" and "Deny".

Type: one of "Allow", "Deny"

Example: `"Allow"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`securityRules.<name>.description`

A description for this rule. Restricted to 140 characters.

Type: string

Default: `""`

Example: `"Allow SSH"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`securityRules.<name>.destinationAddressPrefix`

CIDR or destination IP range or * to match any IP. Tags such as "VirtualNetwork", "AzureLoadBalancer" and "Internet" can also be used.

Type: string

Example: `"Internet"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`securityRules.<name>.destinationPortRange`

Destination Port or Range. Integer or range between 0 and 65535 or * to match any.

Type: string

Example: `"22"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`securityRules.<name>.direction`

The direction specifies if rule will be evaluated on incoming or outgoing traffic. Possible values are "Inbound" and "Outbound".

Type: one of "Inbound", "Outbound"

Example: `"Inbound"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`securityRules.<name>.priority`

Specifies the priority of the rule. The value can be between 100 and 4096. The priority number must be unique for each rule in the collection. The lower the priority number, the higher the priority of the rule.

Type: signed integer

Example: `2000`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`securityRules.<name>.protocol`

Network protocol this rule applies to. Can be Tcp, Udp or * to match both.

Type: one of "Tcp", "Udp", "*"

Example: `"Udp"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`securityRules.<name>.sourceAddressPrefix`

CIDR or source IP range or * to match any IP. Tags such as "VirtualNetwork", "AzureLoadBalancer" and "Internet" can also be used.

Type: string

Example: `"Internet"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`securityRules.<name>.sourcePortRange`

Source Port or Range. Integer or range between 0 and 65535 or * to match any.

Type: string

Example: `"22"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix

`tags`

Tag name/value pairs to associate with the network security group.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-network-security-group.nix`

B.4.13. Azure Queues

An Azure Queue is defined by setting `resources.azureQueues.name` to an attribute set containing values for the following options.

Appendix AY. Configuration Options

`accessKey`

Access key for the storage service if not managed by NixOps.

Type: null or string

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-queue.nix`

`acl.signedIdentifiers`

An attribute set of Signed Identifiers and the corresponding access policies that may be used with Shared Access Signatures.

Type: attribute set of submodules

Default: `{ }`

Example: `{ MTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTI= = { expiry = "2013-11-27T08:49:37.0000000Z"; permissions = "raud"; start = "2013-11-26T08:49:37.0000000Z"; } ; }`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-queue.nix`

`acl.signedIdentifiers.<name>.expiry`

Access policy expiry UTC date/time in a valid ISO 8061 format. Supported ISO 8061 formats include the following: YYYY-MM-DD, YYYY-MM-DDThh:mmTZD, YYYY-MM-DDThh:mm:ssTZD, YYYY-MM-DDThh:mm:ss.ffffffTZD

Type: string

Example: `"2013-11-26T08:49:37.0000000Z"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-queue.nix

`acl.signedIdentifiers.<name>.permissions`

Abbreviated permission list.

Type: string

Example: `"raud"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-queue.nix

`acl.signedIdentifiers.<name>.start`

Access policy start UTC date/time in a valid ISO 8061 format. Supported ISO 8061 formats include the following: YYYY-MM-DD, YYYY-MM-DDThh:mmTZD, YYYY-MM-DDThh:mm:ssTZD, YYYY-MM-DDThh:mm:ss.ffffffTZD

Type: string

Example: `"2013-11-26T08:49:37.0000000Z"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-queue.nix

`metadata`

Metadata name/value pairs to associate with the queue.

Type: attribute set of strings

Default: `{ }`

Example: `{ loglevel = "warn"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-queue.nix

`name`

Description of the Azure queue. This is the `Name` tag of the queue.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-queue"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-queue.nix

`storage`

The name or resource of an Azure storage in which the queue is to be created.

Type: string or resource of type ‘azure-storage’

Example: `"xxx-my-storage"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-queue.nix

B.4.14. Azure Reserved IP Addresses

An Azure Reserved IP Address is defined by setting `resources.azureReservedIPAddresses.`*`name`* to an attribute set containing values for the following options.

Appendix AZ. Configuration Options

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

`domainNameLabel`

The concatenation of the domain name label and the regionalized DNS zone make up the fully qualified domain name associated with the public IP address. If a domain name label is specified, an A DNS record is created for the public IP in the Microsoft Azure DNS system. Example FQDN: mylabel.northus.cloudapp.azure.com.

Type: null or string

Default: `null`

Example: `"mylabel"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

`idleTimeout`

The timeout for the TCP idle connection. The value can be set between 4 and 30 minutes.

Type: signed integer

Default: `4`

Example: `30`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

`location`

The Azure data center where the reserved IP address should be located.

Type: string

Example: `"West US"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

`name`

Description of the Azure reserved IP address. This is the `Name` tag of the address.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-public-ip"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

`resourceGroup`

The name or resource of an Azure resource group to create the IP address in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

`reverseFqdn`

A fully qualified domain name that resolves to this public IP address. If the reverseFqdn is specified, then a PTR DNS record is created pointing from the IP address in the in-addr.arpa domain to the reverse FQDN.

Type: null or string

Default: `null`

Example: `"mydomain.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

`tags`

Tag name/value pairs to associate with the IP address.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-reserved-ip-address.nix

B.4.15. Azure Resource Groups

An Azure Resource Group is defined by setting `resources.azureResourceGroups.`*`name`* to an attribute set containing values for the following options.

Appendix BA. Configuration Options

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-resource-group.nix

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-resource-group.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-resource-group.nix

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-resource-group.nix

`location`

The Azure data center location where the resource group should be created.

Type: string

Example: `"westus"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-resource-group.nix`

`name`

Description of the Azure Resource Group. This is the `Name` tag of the group.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-resource-group"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-resource-group.nix`

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-resource-group.nix`

`tags`

Tag name/value pairs to associate with the resource group.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-resource-group.nix`

B.4.16. Azure Shares

An Azure Share is defined by setting `resources.azureShares.`*`name`* to an attribute set containing values for the following options.

Appendix BB. Configuration Options

`accessKey`

Access key for the storage service if not managed by NixOps.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-share.nix

`metadata`

Metadata name/value pairs to associate with the share.

Type: attribute set of strings

Default: `{ }`

Example: `{ loglevel = "warn"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-share.nix

`name`

Description of the Azure share. This is the `Name` tag of the share.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-share"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-share.nix

`storage`

The name or resource of an Azure storage in which the share is to be created.

Type: string or resource of type ‘azure-storage’

Example: `"xxx-my-storage"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-share.nix

B.4.17. Azure Storages

An Azure Storage is defined by setting `resources.azureStorages.`*`name`* to an attribute set containing values for the following options.

Appendix BC. Configuration Options

`accountType`

Specifies whether the account supports locally-redundant storage, geo-redundant storage, zone-redundant storage, or read access geo-redundant storage. Possible values are: Standard_LRS, Standard_ZRS, Standard_GRS, Standard_RAGRS, Premium_LRS

Type: string

Default: `"Standard_LRS"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`activeKey`

Specifies which of the access keys should be used by containers, tables and queues. The keys provide the same access, but can be independently regenerated which allows seamless key replacement. Possible values are: primary, secondary.

Type: string

Default: `"primary"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.hourMetrics.enable`
Whether metrics are enabled for the service.

Type: boolean

Default: `true`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.hourMetrics.includeAPIs`
Whether metrics should generate summary statistics for called API operations.

Type: boolean

Default: `true`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.hourMetrics.retentionPolicy.days`
Indicates the number of days that metrics or logging data is retained. All data older than this value will be deleted.

Type: signed integer

Default: `7`

Example: `3`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.hourMetrics.retentionPolicy.enable`
Whether a retention policy is enabled for the service.

Type: boolean

Default: `true`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.logging.delete`
Whether delete requests should be logged.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.logging.read`
Whether read requests should be logged.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.logging.retentionPolicy.days`
Indicates the number of days that metrics or logging data is retained. All data older than this value will be deleted.

Type: signed integer

Default: `7`

Example: `3`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.logging.retentionPolicy.enable`
Whether a retention policy is enabled for the service.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.logging.write`
Whether write requests should be logged.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.minuteMetrics.enable`

Whether metrics are enabled for the service.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.minuteMetrics.includeAPIs`

Whether metrics should generate summary statistics for called API operations.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.minuteMetrics.retentionPolicy.days`

Indicates the number of days that metrics or logging data is retained. All data older than this value will be deleted.

Type: signed integer

Default: `7`

Example: `3`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`blobService.minuteMetrics.retentionPolicy.enable`

Whether a retention policy is enabled for the service.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`customDomain`

User domain assigned to the storage account. Name is the CNAME source.

Type: string

Default: `""`

Example: `"mydomain.org"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix`

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix`

`location`

The Azure data center location where the storage should be created.

Type: string

Example: `"westus"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix`

`name`

Name of the Azure storage account. Must be globally-unique, between 3 and 24 characters in length, and must consist of numbers and lower-case letters only.

Type: string

Example: `"my-storage"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix`

`queueService.hourMetrics.enable`

Whether metrics are enabled for the service.

Type: boolean

Default: `true`

Example: `true`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix`

`queueService.hourMetrics.includeAPIs`

Whether metrics should generate summary statistics for called API operations.

Type: boolean

Default: `true`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`queueService.hourMetrics.retentionPolicy.days`

Indicates the number of days that metrics or logging data is retained. All data older than this value will be deleted.

Type: signed integer

Default: `7`

Example: `3`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`queueService.hourMetrics.retentionPolicy.enable`

Whether a retention policy is enabled for the service.

Type: boolean

Default: `true`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`queueService.logging.delete`

Whether delete requests should be logged.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`queueService.logging.read`

Whether read requests should be logged.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`queueService.logging.retentionPolicy.days`

Indicates the number of days that metrics or logging data is retained. All data older than this value will be deleted.

Type: signed integer

Default: `7`

Example: `3`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`queueService.logging.retentionPolicy.enable`

Whether a retention policy is enabled for the service.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`queueService.logging.write`

Whether write requests should be logged.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`queueService.minuteMetrics.enable`

Whether metrics are enabled for the service.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`queueService.minuteMetrics.includeAPIs`

Whether metrics should generate summary statistics for called API operations.

Type: boolean

Default: `false`

Example: `true`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix`

`queueService.minuteMetrics.retentionPolicy.days`

Indicates the number of days that metrics or logging data is retained. All data older than this value will be deleted.

Type: signed integer

Default: `7`

Example: `3`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix`

`queueService.minuteMetrics.retentionPolicy.enable`

Whether a retention policy is enabled for the service.

Type: boolean

Default: `false`

Example: `true`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix`

`resourceGroup`

The name or resource of an Azure resource group to create the storage in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix`

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix`

`tableService.hourMetrics.enable`

Whether metrics are enabled for the service.

Type: boolean

Default: `true`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.hourMetrics.includeAPIs`
Whether metrics should generate summary statistics for called API operations.

Type: boolean

Default: `true`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.hourMetrics.retentionPolicy.days`
Indicates the number of days that metrics or logging data is retained. All data older than this value will be deleted.

Type: signed integer

Default: `7`

Example: `3`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.hourMetrics.retentionPolicy.enable`
Whether a retention policy is enabled for the service.

Type: boolean

Default: `true`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.logging.delete`
Whether delete requests should be logged.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.logging.read`

Whether read requests should be logged.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.logging.retentionPolicy.days`

Indicates the number of days that metrics or logging data is retained. All data older than this value will be deleted.

Type: signed integer

Default: `7`

Example: `3`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.logging.retentionPolicy.enable`

Whether a retention policy is enabled for the service.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.logging.write`

Whether write requests should be logged.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.minuteMetrics.enable`

Whether metrics are enabled for the service.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.minuteMetrics.includeAPIs`
Whether metrics should generate summary statistics for called API operations.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.minuteMetrics.retentionPolicy.days`
Indicates the number of days that metrics or logging data is retained. All data older than this value will be deleted.

Type: signed integer

Default: `7`

Example: `3`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tableService.minuteMetrics.retentionPolicy.enable`
Whether a retention policy is enabled for the service.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

`tags`
Tag name/value pairs to associate with the storage.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-storage.nix

B.4.18. Azure Tables

An Azure Table is defined by setting `resources.azureTables.name` to an attribute set containing values for the following options.

Appendix BD. Configuration Options

`accessKey`

Access key for the storage service if not managed by NixOps.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-table.nix

`acl.signedIdentifiers`

An attribute set of Signed Identifiers and the corresponding access policies that may be used with Shared Access Signatures.

Type: attribute set of submodules

Default: `{ }`

Example: `{ MTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTI= = { expiry = "2013-11-27T08:49:37.0000000Z"; permissions = "raud"; start = "2013-11-26T08:49:37.0000000Z"; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-table.nix

`acl.signedIdentifiers.<name>.expiry`

Access policy expiry UTC date/time in a valid ISO 8061 format. Supported ISO 8061 formats include the following: YYYY-MM-DD, YYYY-MM-DDThh:mmTZD, YYYY-MM-DDThh:mm:ssTZD, YYYY-MM-DDThh:mm:ss.ffffffTZD

Type: string

Example: `"2013-11-26T08:49:37.0000000Z"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-table.nix

`acl.signedIdentifiers.<name>.permissions`

Abbreviated permission list.

Type: string

Example: `"raud"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-table.nix

`acl.signedIdentifiers.<name>.start`

Access policy start UTC date/time in a valid ISO 8061 format. Supported ISO 8061 formats include the following: YYYY-MM-DD, YYYY-MM-DDThh:mmTZD, YYYY-MM-DDThh:mm:ssTZD, YYYY-MM-DDThh:mm:ss.ffffffTZD

Type: string

Example: `"2013-11-26T08:49:37.0000000Z"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-table.nix

`name`

Description of the Azure table. The name must not contain dashes. This is the `Name` tag of the table.

Type: string

Default: `"nixops<uuid><name>"`

Example: `"mytable"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-table.nix

`storage`

The name or resource of an Azure storage in which the table is to be created.

Type: string or resource of type ‘azure-storage’

Example: `"xxx-my-storage"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-table.nix

B.4.19. Azure Traffic Manager Profiles

An Azure Traffic Manager Profile is defined by setting `resources.azureTrafficManagerProfiles.`*`name`* to an attribute set containing values for the following options.

Appendix BE. Configuration Options

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`dns.relativeName`

Specifies the relative DNS name provided by this Traffic Manager profile. This value is combined with the DNS domain name used by Azure Traffic Manager to form the fully-qualified domain name (FQDN) of the profile.

Type: string

Example: `"myservice"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`dns.ttl`

Specifies the DNS Time-to-Live (TTL), in seconds. This informs the Local DNS resolvers and DNS clients how long to cache DNS responses provided by this Traffic Manager profile. Possible values are 30...999,999.

Type: signed integer

Example: `30`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`enable`

Whether to enable the Traffic Manager profile.

Type: boolean

Default: `true`

Example: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`endpoints`

An attribute set of endpoints

Type: attribute set of submodules

Default: `{ }`

Example: `{ west_us_endpoint = { location = "westus"; target = "westus.sample.org"; } ; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`endpoints.<name>.enable`

Whether to enable the endpoint. If the endpoint is Enabled, it is probed for endpoint health and is included in the traffic routing method.

Type: boolean

Default: `true`

Example: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`endpoints.<name>.location`

Specifies the location of the endpoint. Must be specified for endpoints when using the 'Performance' traffic routing method.

Type: null or string

Default: `null`

Example: `"westus"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`endpoints.<name>.priority`

Specifies the priority of this endpoint when using the 'priority' traffic routing method. Priority must lie in the range 1...1000. Lower values represent higher priority. No two endpoints can share the same priority value.

Type: null or signed integer

Default: `null`

Example: `1000`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`endpoints.<name>.target`

The fully-qualified DNS name of the endpoint. Traffic Manager returns this value in DNS responses to direct traffic to this endpoint.

Type: string

Example: `"myendpoint.sample.org"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`endpoints.<name>.weight`

Specifies the weight assigned by Traffic Manager to the endpoint. This is only used if the Traffic Manager profile is configured to use the 'weighted' traffic routing method. Possible values are from 1 to 1000.

Type: null or signed integer

Default: `null`

Example: `1000`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`monitor.path`

Specifies the path relative to the endpoint domain name used to probe for endpoint health.

Type: string

Default: `"/"`

Example: `"/alive"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`monitor.port`

Specifies the TCP port used to monitor endpoint health. Possible values are 1...65535

Type: signed integer

Default: `80`

Example: `8080`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`monitor.protocol`

Specifies the protocol to use to monitor endpoint health.

Type: one of "HTTP", "HTTPS"

Default: `"HTTP"`

Example: `"HTTPS"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`name`

Name of the Azure Traffic Manager profile.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-traffic-manager-profile"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`resourceGroup`

The name or resource of an Azure resource group to create the Traffic Manager profile in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`tags`
  Tag name/value pairs to associate with the Traffic Manager profile.

  Type: attribute set of strings

  Default: `{ }`

  Example: `{ environment = "production"; }`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

`trafficRoutingMethod`
  Specifies the traffic routing method, used to determine which endpoint is returned in response to incoming DNS queries.

  Type: one of "Performance", "Weighted", "Priority"

  Default: `"Performance"`

  Example: `"Priority"`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-traffic-manager-profile.nix

B.4.20. Azure Virtual Networks

An Azure Virtual Network is defined by setting `resources.azureVirtualNetworks.`*`name`* to an attribute set containing values for the following options.

Appendix BF. Configuration Options

`addressSpace`
  The list of address blocks reserved for this virtual network in CIDR notation.

  Type: list of strings

  Example: `[ "10.1.0.0/16" "10.3.0.0/16" ]`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix

`appId`
  The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

  Type: string

  Default: `""`

  Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

  Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix

`dnsServers`

List of DNS servers IP addresses to provide via DHCP. Leave empty to provide the default Azure DNS servers.

Type: null or list of strings

Default: `[ ]`

Example: `[ "8.8.8.8" "8.8.4.4" ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix

`location`

The Azure data center location where the virtual network should be created.

Type: string

Example: `"westus"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix`

`name`

Name of the Azure virtual network.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-network"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix`

`resourceGroup`

The name or resource of an Azure resource group to create the network in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix`

`subnets`

An attribute set of subnets

Type: attribute set of submodules

Example: `{ }`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix`

`subnets.<name>.addressPrefix`

Address prefix for the subnet in CIDR notation.

Type: string

Example: `"10.1.0.0/24"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix`

`subnets.<name>.securityGroup`

The Azure Resource Id or NixOps resource of the Azure network security group to apply to all NICs in the subnet.

Type: null or string or resource of type ‘azure-network-security-group’

Default: `null`

Example: `"resources.azureSecurityGroups.my-security-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix

`tags`

Tag name/value pairs to associate with the virtual network.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network.nix

B.4.21. Azure Virtual Network Gateways

An Azure Virtual Network Gateway is defined by setting `resources.azureVirtualNetworkGateways.`*`name`* to an attribute set containing values for the following options.

Appendix BG. Configuration Options

`appId`

The ID of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_ID`.

Type: string

Default: `""`

Example: `"aaaaaaaa-0000-aaaa-0000-aaaaaaaaaaaa"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix

`appKey`

The secret value of registrated application in Azure Active Directory. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_KEY`.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix

`authority`

The Azure Authority URL. If left empty, it defaults to the contents of the environment variable `AZURE_AUTHORITY_URL`.

Type: string

Default: `""`

Example: `"https://login.windows.net/ACTIVE_DIRECTORY_TENANT.onmicrosoft.com"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix

`bgpEnabled`

Whether BGP is enabled for this virtual network gateway or not.

Type: boolean

Default: `false`

Example: `true`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix

`gatewaySize`

The size of the virtual network gateway.

Type: one of "Default", "HighPerformance"

Default: `"Default"`

Example: `"HighPerformance"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix

`gatewayType`

The type of the virtual network gateway: RouteBased or PolicyBased.

Type: string

Example: `"RouteBased"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix

`identifierUri`

The URI that identifies the resource for which the token is valid. If left empty, it defaults to the contents of the environment variable `AZURE_ACTIVE_DIR_APP_IDENTIFIER_URI`.

Type: string

Default: `"https://management.azure.com/"`

Example: `"https://management.azure.com/"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix

`location`

The Azure data center location where the virtual network gateway should be created.

Type: string

Example: `"westus"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix

`name`

Name of the Azure virtual network gateway.

Type: string

Default: `"nixops-<uuid>-<name>"`

Example: `"my-virtual-network-gateway"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix

`resourceGroup`

The name or resource of an Azure resource group to create the virtual network gateway in.

Type: string or resource of type ‘azure-resource-group’

Example: `"xxx-my-group"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix

`subnet.name`

The name of the subnet of `network` to use as the gateway subnet.

Type: string

Default: `"default"`

Example: `"my-subnet"`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix

`subnet.network`

The Azure Resource Id or NixOps resource of an Azure virtual network that contains the gateway subnet.

Type: null or string or resource of type ‘azure-virtual-network’

Default: `null`

Example: `"my-network"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix`

`subscriptionId`

The Azure Subscription ID. If left empty, it defaults to the contents of the environment variable `AZURE_SUBSCRIPTION_ID`.

Type: string

Default: `""`

Example: `"f1ce4500-ab06-495a-8d59-a7cfe9e46dae"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix`

`tags`

Tag name/value pairs to associate with the virtual network gateway.

Type: attribute set of strings

Default: `{ }`

Example: `{ environment = "production"; }`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/azure-virtual-network-gateway.nix`

B.5. Datadog Resources

B.5.1. Datadog Monitor Resource

A Datadog monitor is defined by setting `resources.datadogMonitors.`*`name`* to an attribute set containing values for the following options.

Appendix BH. Configuration Options

`apiKey`

The Datadog API Key.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-monitor.nix`

`appKey`

The Datadog APP Key.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-monitor.nix`

`message`

Message to send for a set of users.

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-monitor.nix`

`monitorOptions`

A dictionary of options for the monitor.

See the API documentation for more details about the available options http://docs.datadoghq.com/api/#monitors

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-monitor.nix`

`monitorTags`

A list of tags to associate with your monitor.

Type: list of strings

Default: `[ ]`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-monitor.nix`

`name`

Name of the alert which will show up in the subject line of the email.

Type: string

Default: `"datadog-monitor-<uuid>-<name>"`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-monitor.nix`

`query`

The query that defines the monitor.

See the datadog API documentation for more details about query creation http://docs.datadoghq.com/api/#monitors

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-monitor.nix`

`silenced`

dictionary of scopes to timestamps or None. Each scope will be muted until the given POSIX timestamp or forever if the value is None.

Examples:

To mute the alert completely: {'*': None}

To mute role:db for a short time: {'role:db': 1412798116}

Type: null or string

Default: `null`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-monitor.nix`

`type`

Type of the datadog resource chosen from: "metric alert" "service check" "event alert".

Type: string

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-monitor.nix`

B.5.2. Datadog Timeboard Resource

A Datadog timeboard is defined by setting `resources.dataogTimeboards.`*`name`* to an attribute set containing values for the following options.

Appendix BI. Configuration Options

`apiKey`

The Datadog API Key.

Type: string

Default: `""`

Declared by:

`/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix`

`appKey`

The Datadog App Key.

Type: string

Default: `""`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix

`description`

A description of the timeboard's content.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix

`graphs`

A list of graph definitions

Type: list of submodules

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix

`graphs.*.definition`

The graph definition.

See datadog JSON graphing documentation for more details http://docs.datadoghq.com/graphingjson/

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix

`graphs.*.title`

The name of the graph.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix

`readOnly`

The read-only status of the timeboard.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix

`templateVariables`

A list of template variables for using Dashboard templating.

Type: list of submodules

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix

`templateVariables.*.default`

The default value for the template variable on dashboard load

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix

`templateVariables.*.name`

The name of the variable.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix

`templateVariables.*.prefix`

The tag prefix associated with the variable. Only tags with this prefix will appear in the variable dropdown.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix

`title`

The title of the timeboard.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-timeboard.nix

B.5.3. Datadog Screenboard Resource

A Datadog screenboard is defined by setting `resources.dataogScreenboards.`*`name`* to an attribute set containing values for the following options.

Appendix BJ. Configuration Options

`apiKey`
: The Datadog API Key.

  Type: string

  Default: `""`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

`appKey`
: The Datadog APP Key.

  Type: string

  Default: `""`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

`boardTitle`
: The name of the dashboard.

  Type: string

  Default: `""`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

`description`
: A description of the dashboard's content.

  Type: string

  Default: `""`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

`height`
: Height in pixels.

  Type: null or signed integer

  Default: `null`

  Declared by:

  /nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

`readOnly`
: The read-only status of the screenboard.

Type: boolean

Default: `false`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

`templateVariables`

A list of template variables for using Dashboard templating.

Type: list of submodules

Default: `[ ]`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

`templateVariables.*.default`

The default value for the template variable on dashboard load

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

`templateVariables.*.name`

The name of the variable.

Type: string

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

`templateVariables.*.prefix`

The tag prefix associated with the variable. Only tags with this prefix will appear in the variable dropdown.

Type: null or string

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

`widgets`

A list of widget definitions.

See the datadog screenboard API for more details on creating screenboard widgets http://docs.datadoghq.com/api/screenboards/

Type: list of strings

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

`width`

Screenboard width in pixels.

Type: null or signed integer

Default: `null`

Declared by:

/nix/store/70nravlngyfahrs0pb49wkwb1h70avay-source/nix/datadog-screenboard.nix

Appendix BK. Hacking

This section provides some notes on how to hack on NixOps. To get the latest version of NixOps from GitHub:

```
$ git clone git://github.com/NixOS/nixops.git
$ cd nixops
```

To build it and its dependencies:

```
$ nix-build release.nix -A build.x86_64-linux
```

The resulting NixOps can be run as `./result/bin/nixops`.

To build all dependencies and start a shell in which all environment variables (such as `PYTHONPATH`) are set up so that those dependencies can be found:

```
$ nix-shell release.nix -A build.x86_64-linux --exclude tarball
$ echo $PYTHONPATH
/nix/store/yzj6p5f7iyh247pwxrg97y3klm6d0cni-python-2.7.3/lib/python2.7/site-packages:...
```

You can then run NixOps in your source tree as follows:

```
$ nixops
```

To run the tests, do

```
$ python2 tests.py
```

Note that some of the tests involve the creation of EC2 resources and thus cost money. You must set the environment variable `EC2_ACCESS_KEY` and (optionally) `EC2_SECRET_KEY`. (If the latter is not set, it will be looked up in `~/.ec2-keys` or in `~/.aws/credentials`, as described in Section 3.3.) To run a specific test, run `python2 tests.py` *`test-name`*, e.g. To run all tests in `./tests/functional/test_encrypted_links.py`

```
$ python2 tests.py tests.functional.test_encrypted_links
```

To run only one test in `tests/functional/test_encrypted_links.py`

```
$ python2 tests.py tests.functional.test_encrypted_links:TestEncryptedLinks.test_deploy
```

To filter on which backends you want to run functional tests against, you can filter on one or more tags. To run e.g. only the virtualbox tests, run:

```
$ python2 tests.py tests.functional -A vbox
```

There are also a few NixOS VM tests. These can be run as follows:

```
$ nix-build release.nix -A tests.none_backend
```

Some useful snippets to debug nixops: Logging

```
# this will not work, because sys.stdout is substituted with log file
print('asdf')

# this will work
self.log('asdf')
from __future__ import print_function; import sys; print('asfd', file=sys.__stdout__)
import sys; import pprint; pprint.pprint(some_structure, stream=sys.__stdout__)
```

To set breakpoint use

```
import sys; import pdb; pdb.Pdb(stdout=sys.__stdout__).set_trace()
```

Appendix BL. Release Notes

Table of Contents

BL.1. Release 1.7 (April XX, 2019)
BL.2. Release 1.6.1 (Sep 14, 2018)
BL.3. Release 1.6 (Mar 28, 2018)
BL.4. Release 1.5.2 (Oct 29, 2017)
BL.5. Release 1.5.1 (Jul 5, 2017)
BL.6. Release 1.5 (Feb 16, 2017)
BL.7. Release 1.4 (Jul 11, 2016)
BL.8. Release 1.3.1 (January 14, 2016)
BL.9. Release 1.3 (September 28, 2015)
BL.10. Release 1.2 (April 30, 2014)
BL.11. Release 1.1.1 (October 2, 2013)
BL.12. Release 1.1 (September 9, 2013)
BL.13. Release 1.0.1 (July 11, 2013)
BL.14. Release 1.0 (June 18, 2013)

BL.1. Release 1.7 (April XX, 2019)

- General

  > ***Warning*** *Azure backend is now disabled after the updates to Azure's Python libraries in NixOS 19.03. Please see PR#1131 for more details.*
  >
  > *Existing Azure deployments should use NixOps release 1.6.1. We hope to revive the Azure support in the future once the API compatibility issues are resolved.*

  - Mitigation for ssh StrictHostKeyChecking=no issue.
  - Fix nixops info --plain output.
  - Documentation fixes: add AWS VPC resources and fix some outdated command outputs.

- Addition of Hashicorp's Vault AppRole resource.

- AWS
  - Add more auto retries to api calls to prevent eventual consistency issues.
  - Fix `nixops check` with NVMe devices.
  - Route53: normalize DNS hostname.
  - S3: support bucket lifecycle configuration as well as versioning.
  - S3: introduce persistOnDestroy for S3 buckets which allows keeping the bucket during a destroy for later usage
  - Fix backup-status output when backup is performed on a subset of devices.
- Datadog
  - add tags for Datadog monitors
- GCE
  - Fix machines being leaked when running destroy after a stop operation.
  - make sure the machine exists before attempting a destroy.
- Hetzner
  - Remove usage of local commands for network configuration.

    > ***Warning*** *Note that this is incompatible with NixOS versions prior to 18.03, see release-notes.*
- VirtualBox
  - added NixOS 18.09/19.03 images.
  - handle deleted VMs from outside NixOps.

This release has contributions from Amine Chikhaoui, Assassinkin, aszlig, Aymen Memni, Chaker Benhamed, Chawki Cheikch, David Kleuker, Domen Kožar, Dorra Hadrich, dzanot, Eelco Dolstra, Jörg Thalheim, Kosyrev Serge, Max Wilson, Michael Bishop, Niklas Hambüchen, Pierre Bourdon, PsyanticY, Robert Hensing.

BL.2. Release 1.6.1 (Sep 14, 2018)

- General
  - Fix the deployment of machines with a large number of keys.
  - Show exit code of configuration activation script, when it is non-zero.
  - Ignore evaluation errors in destroy and delete operations.
  - Removed top-level Exception catch-all

  - Minor bugfixes.

- AWS

  - Automatically retry certain API calls.

  - Fixed deployment errors when `deployment.route53.hostName` contains uppercase letters.

  - Support for GCE routes.

  - Support attaching NVMe disks.

- GCE

  - Add labels for GCE volumes and snapshots.

  - Add option to enable IP forwarding.

- VirtualBox

  - Use images from nixpkgs if available.

This release has contributions from Amine Chikhaoui, aszlig, Aymen Memni, Chaker Benhamed, Domen Kožar, Eelco Dolstra, Justin Humm, Michael Bishop, Niklas Hambüchen, Rob Vermaas, Sergei Khoma.

BL.3. Release 1.6 (Mar 28, 2018)

- General

  - JSON output option for `show-option` command.

  - Added experimental `--show-plan` to `deploy` command. Only works for VPC resources currently.

- Backend: libvirtd

  - Added support for custom kernel/initrd/cmdline, for easier kernel testing/developing.

  - Fail early when defining domain.

  - Support NixOS 18.03

- Backend: AWS/EC2

  - Allow changing security groups for instances that were deployed with a default VPC (no explicit subnetId/vpc)

  - Make sure EC2 keypair not destroyed when it is in use, instead produce error.

  - Support for separate Route53 resources.

  - Support CloudWatch metrics and alarms.

  - Support updating IAM instance profile of an existing instance.

  - Support VPC resources.

    - RDS: allow multiple security groups.
    - Allow S3 buckets to be configured as websites.
    - Fix issue where S3 bucket policy was only set on initial deploy.
- Backend: Datadog
    - Support sending start/finish of deploy and destroy events.
    - Support setting downtime during deployment.
- Backend: Azure
    - Fix Azure access instructions.
- Backend: Google Compute
    - Add support for labelling GCE instances
    - Minor fixes to make GCE backend more consistent with backends such as EC2.
    - Fix attaching existing volumes to instances.
    - Implemented `show-physical --backup` for GCE, similar to EC2.
    - Prevent google-instance-setup service from replacing the host key deployed by NixOps.
    - Allow instances to be created inside VPC subnets.

This release has contributions from Adam Scott, Amine Chikhaoui, Anthony Cowley, Brian Olsen, Daniel Kuehn, David McFarland, Domen Kožar, Eelco Dolstra, Glenn Searby, Graham Christensen, Masato Yonekawa, Maarten Hoogendoorn, Matthieu Coudron, Maximilian Bosch, Michael Bishop, Niklas Hambüchen, Oussama Elkaceh, Pierre-Étienne Meunier, Peter Jones, Rob Vermaas, Samuel Leathers, Shea Levy, Tomasz Czyż, Vaibhav Sagar.

BL.4. Release 1.5.2 (Oct 29, 2017)

- General
    - This release has various minor bug and documentation fixes.
    - #703: don't ask for known host if file doesn't exist.
    - Deprecated --evaluate-only for --dry-run.
- Backend: libvirtd
    - Added domainType option.
    - Make the libvirt images readable only by their owner/group.
    - Create "persistent" instead of "transient" domains, this ensures that nixops deployments/VMs survive a reboot.
    - Stop using disk backing file and use self contained images.

- Backend: EC2
  - #652, allow securityGroups of Elastic File System mount target to be set.
  - #709: allow Elastic IP resource for security group sourceIP attribute.
- Backend: Azure
  - Use Azure images from nixpkgs, if they are available.
- Backend: Google Compute
  - Use Google Compute images from nixpkgs, if they are available.

This release has contributions from Andreas Rammhold, Bjørn Forsman, Chris Van Vranken, Corbin, Daniel Ehlers, Domen Kožar, Johannes Bornhold, John M. Harris, Jr, Kevin Quick, Kosyrev Serge, Marius Bergmann, Nadrieril, Rob Vermaas, Vlad Ki.

BL.5. Release 1.5.1 (Jul 5, 2017)

- General
  - This release has various minor bug and documentation fixes.
- Backend: None
  - #661: Added *deployment.keys.*.keyFile* option to provide keys from local files, rather than from text literals.
  - #664: Added *deployment.keys.*.destDir* and *deployment.keys.*.path* options to give more control over where the deployment keys are stored on the deployed machine.
- Backend: Datadog
  - Show URL for dashboards and timeboards in info output.
- Backend: Hetzner
  - Added option to disable creation of sub-accounts.
- Backend: Google Compute
  - Added option to set service account for an instance.
  - Added option to use preemptible option when creating an instance.
- Backend: Digital Ocean
  - Added option to support IPv6 on Digital Ocean.

This release has contributions from Albert Peschar, Amine Chikhaoui, aszlig, Clemens Fruhwirth, Domen Kožar, Drew Hess, Eelco Dolstra, Igor Pashev, Johannes Bornhold, Kosyrev Serge, Leon Isenberg, Maarten Hoogendoorn, Nadrieril Feneanar, Niklas Hambüchen, Philip Patsch, Rob Vermaas, Sven Slootweg.

BL.6. Release 1.5 (Feb 16, 2017)

- General
    - Various minor documentation and bug fixes
    - #508: Implementation of SSH tunnels has been rewritten to use iproute in stead of netttools
    - #400: The ownership of keys is now implemented after user/group creation
    - #216: Added *--keep-days* option for cleaning up backups
    - #594: NixOps statefile is now created with stricter permissions
    - Use types.submodule instead of deprecated types.optionSet
    - #566: Support setting deployment.hasFastConnection
    - Support for "nixops deploy --evaluate-only"
- Backend: None
    - Create /etc/hosts
- Backend: Amazon Web Services
    - Support for Elastic File Systems
    - Support latest EBS volume types
    - Support for Simple Notification Service
    - Support for Cloudwatch Logs resources
    - Support loading credentials from ~/.aws/credentials (AWS default)
    - Use HVM as default virtualization type (all new instance types are HVM)
    - #550: Fix sporadic error "Error binding parameter 0 - probably unsupported type"
- Backend: Datadog
    - Support provisioning Datadog Monitors
    - Support provisioning Datadog Dashboards
- Backend: Hetzner
    - #564: Binary cache substitutions didn't work because of certificate errors
- Backend: VirtualBox
    - Support dots in machine names
    - Added vcpu option
- Backend: Libvirtd

    - Documentation typo fixes
- Backend: Digital Ocean
    - Initial support for Digital Ocean to deploy machines

This release has contributions from Amine Chikhaoui, Anders Papitto, aszlig, Aycan iRiCAN, Christian Kauhaus, Corbin Simpson, Domen Kožar, Eelco Dolstra, Evgeny Egorochkin, Igor Pashev, Maarten Hoogendoorn, Nathan Zadoks, Pascal Wittmann, Renzo Carbonaram, Rob Vermaas, Ruslan Babayev, Susan Potter and Danylo Hlynskyi.

BL.7. Release 1.4 (Jul 11, 2016)

- General
    - Added *show-arguments* command to query nixops arguments that are defined in the nix expressions
    - Added *--dry-activate* option to the deploy command, to see what services will be stopped/started/restarted.
    - Added *--fallback* option to the deploy command to match the same flag on nix-build.
    - Added *--cores* option to the deploy command to match the same flag on nix-build.
- Backend: None
- Amazon EC2
    - Use hvm-s3 AMIs when appropriate
    - Allow EBS optimized flag to be changed (needs --allow-reboot)
    - Allow to recover from spot instance kill, when using external volume defined as resource (resources.ebsVolumes)
    - When disassociating an elastic IP, make sure to check the current instance is the one who is currently associated with it, in case someone else has 'stolen' the elastic IP
    - Use generated list for deployment.ec2.physicalProperties, based on Amazon Pricing listing
    - EC2 AMI registry has been moved the the nixpkgs repository
    - Allow a timeout on spot instance creation
    - Allow updating security groups on running instances in a VPC
    - Support x1 instances
- Backend: Azure
    - New Azure Cloud backend contributed by Evgeny Egorochkin
- Backend: VirtualBox
    - Respect deployment.virtualbox.disks.*.size for images with a baseImage

- Allow overriding the VirtualBox base image size for disk1

- Libvirt
  - Improve logging messages
  - #345: Use qemu-system-x86_64 instead of qemu-kvm for non-NixOS support
  - add extraDomainXML NixOS option
  - add extraDevicesXML NixOS option
  - add vcpu NixOS option

This release has contributions from Amine Chikhaoui, aszlig, Cireo, Domen Kožar, Eelco Dolstra, Eric Sagnes, Falco Peijnenburg, Graham Christensen, Kevin Cox, Kirill Boltaev, Mathias Schreck, Michael Weiss, Brian Zach Abe, Pablo Costa, Peter Hoeg, Renzo Carbonara, Rob Vermaas, Ryan Artecona, Tobias Pflug, Tom Hunger, Vesa Kaihlavirta, Danylo Hlynskyi.

BL.8. Release 1.3.1 (January 14, 2016)

- General
  - #340: "too long for Unix domain socket" error
  - #335: Use the correct port when setting up an SSH tunnel
  - #336: Add support for non-machine IP resources in /etc/hosts
  - Fix determining system.stateVersion
  - ssh_util: Reconnect on dead SSH master socket
  - #379: Remove reference to `jobs` attribute in NixOS
- Backend: None
  - Pass deployment.targetPort to ssh for none backend
  - #361: don't use _ssh_private_key if its corresponding public key hasn't been deployed yet
- Amazon EC2
  - Allow specifying assumeRolePolicy for IAM roles
  - Add vpcId option to EC2 security group resources
  - Allow VPC security groups to refer to sec. group names (within the same sec. group) as well as group ids
  - Prevent vpc calls to be made if only security group ids are being used (instead of names)
  - Use correct credentials for VPC API calls
  - Fix "creating EC2 instance (... region ‘None’)" when recreating missing instance

    - Allow keeping volumes while destroying deployment

- VirtualBox
    - #359: Change sbin/mount.vboxsf to bin/mount.vboxsf
- Hetzner
    - #349: Don't create /root/.ssh/authorized_keys
    - #348: Fixup and refactor Hetzner backend tests
    - hetzner-bootstrap: Fix wrapping Nix inside chroot
    - hetzner-bootstrap: Allow to easily enter chroot
- Libvirt
    - #374: Add headless mode
    - #374: Use more reliable method to retrieve IP address
    - #374: Nicer error message for missing images dir
    - #374: Be able to specify xml for devices

This release has contributions from aszlig, Bas van Dijk, Domen Kožar, Eelco Dolstra, Kevin Cox, Paul Liu, Robin Gloster, Rob Vermaas, Russell O'Connor, Tristan Helmich and Yves Parès (Ywen)

BL.9. Release 1.3 (September 28, 2015)

- General
    - NixOps now requires NixOS 14.12 and up.
    - Machines in NixOps network now have access to the deployment name, uuid and its arguments, by means of the `deployment.name`, `deployment.uuid` and `deployment.arguments` options.
    - Support for <...> paths in network spec filenames, e.g. you can use: `nixops create '<nixops/templates/container.nix>'`.
    - Support ‘username@machine’ for `nixops scp`
- Amazon EC2
    - Support for the latest EC2 instance types, including t2 and c4 instance.
    - Support Amazon EBS SSD disks.
    - Instances can be placed in an EC2 placement group. This allows instances to be grouped in a low-latency 10 Gbps network.
    - Allow starting EC2 instances in a VPC subnet.
    - More robust handling of spot instance creation.

    - Support for setting bucket policies on S3 buckets created by NixOps.
    - Route53 support now uses CNAME to public DNS hostname, in stead of A record to the public IP address.
    - Support Amazon RDS instances.
- Google Cloud
    - Instances
    - Disks
    - Images
    - Load balancer, HTTP health check, Target pools and forwarding rules.
    - Static IPs
    - New backend for Google Cloud Platform. It includes support for the following resources:
- VirtualBox
    - VirtualBox 5.0 is required for the VirtualBox backend.
- NixOS container
    - New backend for NixOS containers.
- Libvirt
    - New backend for libvirt using QEMU/KVM.

This release has contributions from Andreas Herrmann, Andrew Murray, aszlig, Aycan iRiCAN, Bas van Dijk, Ben Moseley, Bjørn Forsman, Boris Sukholitko, Bruce Adams, Chris Forno, Dan Steeves, David Guibert, Domen Kožar, Eelco Dolstra, Evgeny Egorochkin, Leroy Hopson, Michael Alyn Miller, Michael Fellinger, Ossi Herrala, Rene Donner, Rickard Nilsson, Rob Vermaas, Russell O'Connor, Shea Levy, Tomasz Kontusz, Tom Hunger, Trenton Strong, Trent Strong, Vladimir Kirillov, William Roe.

BL.10. Release 1.2 (April 30, 2014)

- General
    - NixOps now requires NixOS 13.10 and up.
    - Add `--all` option to `nixops destroy`, `nixops delete` and `nixops ssh-for-each`.
    - The `-d` option now matches based on prefix for convenience when the specified uuid/id is not found.
    - Resources can now be accessed via direct reference, i.e. you can use `securityGroups = [ resources.ec2SecurityGroups.foo ];` in stead of `securityGroups = [ resources.ec2SecurityGroups.foo.name ];`.
    - Changed default value of `deployment.storeKeysOnMachine` to false, which is the more secure option. This can prevent unattended reboot from finishing, as keys will need to be pushed

to the machine.

- Amazon EC2
  - Support provisioning of elastic IP addresses.
  - Support provisioning of EC2 security groups.
  - Support all HVM instance types.
  - Support `ap-southeast-1` region.
  - Better handling of errors in pushing Route53 records.
  - Support using ARN's for applying instance profiles to EC2 instances. This allows cross-account API access.
  - Base HVM image was updated to allow using all emphemeral devices.
  - Instance ID is now available in nix through the `deployment.ec2.instanceId` option, set by nixops.
  - Support independent provisioning of EBS volumes. Previously, EBS volumes could only be created as part of an EC2 instance, meaning their lifetime was tied to the instance and they could not be managed separately. Now they can be provisioned independently, e.g.:

    ```
    resources.ebsVolumes.bigdata =
      { name = "My Big Fat Data";
        region = "eu-west-1";
        zone = "eu-west-1a";
        accessKeyId = "...";
        size = 1000;
      };
    ```

  - To allow cross-account API access, the *deployment.ec2.instanceProfile* option can now be set to either a name (previous behaviour) or an Amazon Resource Names (ARN) of the instance profile you want to apply.
- Hetzner
  - Always hard reset on destroying machine.
  - Support for Hetzner vServers.
  - Disabled root password by default.
  - Fix hard reset for rebooting to rescue mode.. This is particularly useful if you have a dead server and want to put it in rescue mode. Now it's possible to do that simply by running:

    ```
    nixops reboot --hard --rescue --include=deadmachine
    ```

- VirtualBox
  - Require VirtualBox >= 4.3.0.
  - Support for shared folders in VirtualBox. You can mount host folder on the guest by setting the deployment.virtualbox.sharedFolders option.

    - Allow destroy if the VM is gone already

This release has contributions from aszlig, Corey O'Connor, Domen Kožar, Eelco Dolstra, Michael Stone, Oliver Charles, Rickard Nilsson, Rob Vermaas, Shea Levy and Vladimir Kirillov.

BL.11. Release 1.1.1 (October 2, 2013)

This a minor bugfix release.

- Added a command-line option *--include-keys* to allow importing SSH public host keys, of the machines that will be imported, to the *.ssh/known_hosts* of the user.
- Fixed a bug that prevented switching the *deployment.storeKeysOnMachine* option value.
- On non-EC2 systems, NixOps will generate ECDSA SSH host key pairs instead of DSA from now on.
- VirtualBox deployments use generated SSH host keypairs.
- For all machines which nixops generates an SSH host keypair for, it will add the SSH public host key to the known_hosts configuration of all machines in the network.
- For EC2 deployments, if the nixops expression specifies a set of security groups for a machine that is different from the security groups applied to the existing machine, it will produce a warning that the change cannot be made.
- For EC2 deployments, disks that are not supposed to be attached to the machine are detached only after system activation has been completed. Previously this was done before, but that could lead to volumes not being able to detach without needing to stop the machine.
- Added a command-line option *--repair* as a convient way to pass this option, which allows repairing of broken or changed paths in the nix store, to nix-build calls that nixops performs. Note that this option only works in nix setups that run without the nix daemon.

This release has contributions from aszlig, Ricardo Correia, Eelco Dolstra, Rob Vermaas.

BL.12. Release 1.1 (September 9, 2013)

- Backend for Hetzner, a German data center provider. More information and a demo video can be found here.
- When using the `deployment.keys.*` options, the keys in /run/keys are now created with mode 600.
- Fixed bug where EBS snapshots name tag was overridden by the instance name tag.
- The nixops executable now has the default OpenSSH from nixpkgs in its PATH now by default, to work around issues with left-over SSH master connections on older version of OpenSSH, such as the version that is installed by default on CentOS.
- A new resource type has been introduced to generate sets of SSH public/private keys.
- Support for spot instances in the EC2 backend. By specifying the `deployment.ec2.spotInstancePrice` option for a machine, you can set the spot instance price in

cents. NixOps will wait 10 minutes for a spot instance to be fulfilled, if not, then it will error out for that machine.

BL.13. Release 1.0.1 (July 11, 2013)

This is a minor bugfix release.

- Reduce parallelism for running EC2 backups, to prevent hammering the AWS API in case of many disks.
- Propagate the instance tags to the EBS volumes (except for Name tag, which is overriden with a detailed description of the volume and its use).

BL.14. Release 1.0 (June 18, 2013)

Initial release.