Introduction

Colmena is a simple, stateless NixOS deployment tool modeled after NixOps and morph, written in Rust. It's a thin wrapper over Nix commands like `nix-instantiate` and `nix-copy-closure`, and supports parallel deployment.

Interested? Get started here!

```
$ colmena apply --on @tag-a
[INFO ] Enumerating nodes...
[INFO ] Selected 7 out of 45 hosts.
  (...) ✅ 0s Build successful
  sigma 🕘 7s copying path
'/nix/store/h6qpk8rwm3dh3zsl1wlj1jharzf8aw9f-unit-haigha-
agent.service' to 'ssh://root@sigma.redacted'...
  theta ✅ 7s Activation successful
  gamma 🕘 8s Starting...
  alpha ✅ 1s Activation successful
epsilon 🕘 7s copying path
'/nix/store/fhh4rfixny8b21l6jqzk7nqwxva5k20h-nixos-system-epsilon-
20.09pre-git' to 'ssh://root@epsilon.redacted'...
   beta 🕘 7s removing obsolete file
/boot/kernels/z28ayg10kpnlrz0s2qrb9pzv82lc20s2-initrd-linux-5.4.89-
initrd
  kappa ✅ 2s Activation successful
```

You are currently reading **the unstable version** of the Colmena Manual, built against the tip of the development branch. Features described here will eventually become a part of **version 0.5**.

Links

- GitHub
- Deployment Options Reference
- Matrix Chat (#colmena:nixos.org)

Tutorial

Installation

To install the latest development version to the user profile, use the following command:

```
nix-env -if https://github.com/zhaofengli/colmena/tarball/main
```

To install the latest stable version, read the corresponding Manual for instructions.

Unstable Binary Cache

A public binary cache is available at https://colmena.cachix.org, courtesy of Cachix. This binary cache contains unstable versions of Colmena built by GitHub Actions.

Basic Configuration

If you use Nix Flakes, follow the Flake version here.

Colmena should work with your existing NixOps and morph configurations with minimal modification (see Migrating from NixOps/morph).

Here is a sample `hive.nix` with two nodes, with some common configurations applied to both nodes:

```
{
  meta = {
    # Override to pin the Nixpkgs version (recommended). This option
    # accepts one of the following:
    # - A path to a Nixpkgs checkout
    # - The Nixpkgs lambda (e.g., import <nixpkgs>)
    # - An initialized Nixpkgs attribute set
    nixpkgs = <nixpkgs>;

    # You can also override Nixpkgs by node!
    nodeNixpkgs = {
      node-b = ./another-nixos-checkout;
    };

    # If your Colmena host has nix configured to allow for remote builds
```

```nix
    # (for nix-daemon, your user being included in trusted-users)
    # you can set a machines file that will be passed to the underlying
    # nix-store command during derivation realization as a builders option.
    # For example, if you support multiple orginizations each with their own
    # build machine(s) you can ensure that builds only take place on your
    # local machine and/or the machines specified in this file.
    # machinesFile = ./machines.client-a;
  };

  defaults = { pkgs, ... }: {
    # This module will be imported by all hosts
    environment.systemPackages = with pkgs; [
      vim wget curl
    ];

    # By default, Colmena will replace unknown remote profile
    # (unknown means the profile isn't in the nix store on the
    # host running Colmena) during apply (with the default goal,
    # boot, and switch).
    # If you share a hive with others, or use multiple machines,
    # and are not careful to always commit/push/pull changes
    # you can accidentaly overwrite a remote profile so in those
    # scenarios you might want to change this default to false.
    # deployment.replaceUnknownProfiles = true;
  };

  host-a = { name, nodes, ... }: {
    # The name and nodes parameters are supported in Colmena,
    # allowing you to reference configurations in other nodes.
    networking.hostName = name;
    time.timeZone = nodes.host-b.config.time.timeZone;

    boot.loader.grub.device = "/dev/sda";
    fileSystems."/" = {
      device = "/dev/sda1";
      fsType = "ext4";
    };
  };

  host-b = {
    # Like NixOps and Morph, Colmena will attempt to connect to
    # the remote host using the attribute name by default. You
    # can override it like:
    deployment.targetHost = "host-b.mydomain.tld";

    # It's also possible to override the target SSH port.
    # For further customization, use the SSH_CONFIG_FILE
    # environment variable to specify a ssh_config file.
    deployment.targetPort = 1234;

    # Override the default for this target host
    deployment.replaceUnknownProfiles = false;

    # You can filter hosts by tags with --on @tag-a,@tag-b.
```

```
      # In this example, you can deploy to hosts with the "web" tag using:
      #    colmena apply --on @web
      # You can use globs in tag matching as well:
      #    colmena apply --on '@infra-*'
      deployment.tags = [ "web" "infra-lax" ];

      time.timeZone = "America/Los_Angeles";

      boot.loader.grub.device = "/dev/sda";
      fileSystems."/" = {
        device = "/dev/sda1";
        fsType = "ext4";
      };
    };
  }
```

The full set of `deployment` options can be found here.

Now you are ready to use Colmena! To build the configuration:

```
colmena build
```

To build and deploy to all nodes:

```
colmena apply
```

Next Steps

- Head to the Features section to see what else Colmena can do.
- Read more about options available in Colmena in the Reference section.

Usage with Flakes

Installation

To quickly try Colmena out, use the following command to enter an ephemeral environment with the latest development version of `colmena`:

```
nix shell github:zhaofengli/colmena
```

To install Colmena to the user profile, use the following command:

```
nix-env -if https://github.com/zhaofengli/colmena/tarball/main
```

You can also add `github:zhaofengli/colmena` as an input in your Flake and add the `colmena` package to your `devShell`.

Unstable Binary Cache

A public binary cache is available at https://colmena.cachix.org, courtesy of Cachix. This binary cache contains unstable versions of Colmena built by GitHub Actions.

Basic Configuration

Colmena reads the `colmenaHive` output in your Flake, generated with `colmena.lib.makeHive`.

Here is a short example:

```
{
  inputs = {
    nixpkgs.url = "github:NixOS/nixpkgs/nixos-unstable";
    colmena.url = "github:zhaofengli/colmena";
  };
  outputs = { nixpkgs, colmena, ... }: {
    colmenaHive = colmena.lib.makeHive {
      meta = {
        nixpkgs = import nixpkgs {
          system = "x86_64-linux";
          overlays = [];
        };
      };

      host-a = { name, nodes, pkgs, ... }: {
        boot.isContainer = true;
        time.timeZone = nodes.host-b.config.time.timeZone;
      };
      host-b = {
        deployment = {
          targetHost = "somehost.tld";
          targetPort = 1234;
          targetUser = "luser";
        };
        boot.isContainer = true;
        time.timeZone = "America/Los_Angeles";
      };
    };
  };
}
```

The full set of `deployment` options can be found here. You can also check out the example in the main tutorial for some inspiration.

Now you are ready to use Colmena! To build the configuration:

```
colmena build
```

To build and deploy to all nodes:

```
colmena apply
```

Migrating to Direct Flake Evaluation

error: flake 'git+file:///path/to/flake' does not provide attribute 'packages.x86_64-

```
linux.colmenaHive', 'legacyPackages.x86_64-linux.colmenaHive' or 'colmenaHive'
```

Colmena now uses `nix eval` to evaluate flakes. Your flake needs to depend on Colmena itself as an input and expose a new output called `colmenaHive`:

```
  {
    inputs = {
+     # ADDED: Colmena input
+     colmena.url = "github:zhaofengli/colmena";

      # ... Rest of configuration ...
    };
    outputs = { self, colmena, ... }: {
+     # ADDED: New colmenaHive output
+     colmenaHive = colmena.lib.makeHive self.outputs.colmena;

      # Your existing colmena output
      colmena = {
        # ... Rest of configuration ...
      };
    };
  }
```

Using Legacy Flake Evaluation (Deprecated)

By default, Colmena uses `nix eval` to evaluate your flake. If you need to use the old evaluation method based on `nix-instantiate` and `builtins.getFlake`, add the `--legacy-flake-eval` flag. The legacy flake evaluator uses the `colmena` output and does not work purely on Nix 2.21+.

Next Steps

- Head to the Features section to see what else Colmena can do.
- Read more about options available in Colmena in the Reference section.

Migrating from NixOps/morph

Colmena should work with existing NixOps and morph configurations with minimal modification. That said, there are a few things to look out for:

Colmena deploys to *existing* NixOS hosts

Unlike NixOps which can be configured to manage the entire lifecycles of NixOS machines (e.g., spinning up AWS EC2 instances), Colmena can only deploy to hosts already running NixOS.

`network` vs `meta`

Colmena accepts a set of options to configure the deployment itself as `meta`. For NixOps compatibility, it also accepts `network` as an alias so you don't have to change your existing configuration.

Pinning Nixpkgs

You can pin the nixpkgs version by setting `meta.nixpkgs` (or `network.nixpkgs` if you use the alias). This is required if you use Flakes. The options accepts one of the following:

- Path to a Nixpkgs checkout *(not supported in Flakes)*
    - Example: `./nixpkgs`
- The Nixpkgs lambda returned by importing its `default.nix` *(not supported in Flakes)*
    - Example: `import ./nixpkgs`
- A fully initialized Nixpkgs attribute set
    - Example: `import ./nixpkgs { system = "x86_64-linux"; }`

Features

This section introduces the main features in Colmena:

- **Node Tagging** - Deploying to a subset of tagged nodes
- **Local Deployment** - Deploying to the host running Colmena itself
- **Secrets** - Deploying sensitive files separate from the main configuration
- **Ad Hoc Evaluation** - Evaluating a Nix expression with access to your configuration
- **Parallelism** - Controlling how Colmena parallelizes the deployment process
- **Remote Builds** - Building system profiles on remote machines

Node Tagging

With node tags, you can quickly select a subset of nodes for deployment. You can specify tags using the `deployment.tags` option:

```
{
  alpha = { pkgs, ... }: {
    deployment.tags = [ "web" "infra-lax" ];

    # ... Rest of configuration ...
  };
  beta = { pkgs, ... }: {
    deployment.tags = [ "infra-sfo" ];

    # ... Rest of configuration ...
  };
}
```

You can filter hosts by tags or names with `--on`, which accepts a comma-separated list of node names or @tags.

To select all nodes with `web`:

```
$ colmena apply --on @web
```

Wildcards are supported as well. To select all nodes with a tag beginning with `infra-`:

```
$ colmena apply --on '@infra-*'
```

(Note the quotes around the argument)

Local Deployment

For some machines, you may still want to stick with the manual `nixos-rebuild` -type of workflow. Colmena allows you to build and activate configurations on the host running Colmena itself, provided that:

1. The node must be running NixOS.
2. The node must have `deployment.allowLocalDeployment` set to `true` .
3. The node's *attribute name* must match the hostname of the machine.

If you invoke `apply-local` with `--sudo` , Colmena will attempt to elevate privileges with `sudo` if it's not run as root. You may also find it helpful to set `deployment.targetHost` to `null` if you don't intend to deploy to the host via SSH.

As an example, the following `hive.nix` includes a node (`laptop`) that is meant to be only deployed with `apply-local` :

```nix
{
  meta = {
    nixpkgs = ./deps/nixpkgs-stable;

    # I'd like to use the unstable version of Nixpkgs on
    # my desktop machines.
    nodeNixpkgs = {
      laptop = ./deps/nixpkgs-unstable;
    };
  };

  # This attribute name must match the output of `hostname` on your machine
  laptop = { name, nodes, ... }: {
    networking.hostName = "laptop";

    deployment = {
      # Allow local deployment with `colmena apply-local`
      allowLocalDeployment = true;

      # Disable SSH deployment. This node will be skipped in a
      # normal`colmena apply`.
      targetHost = null;
    };

    # ... Rest of configuration ...
  };

  server-a = { pkgs, ... }: {
    # This node will use the default Nixpkgs checkout specified
    # in `meta.nixpkgs`.

    # ... Rest of configuration ...
  };
}
```

On `laptop`, run `colmena apply-local --sudo` to activate the configuration.

Secrets

Colmena allows you to upload secret files that will not be stored in the Nix store to nodes. It implements a subset of the `deployment.keys` options supported by NixOps.

For example, to deploy DNS-01 credentials for use with `security.acme`:

```
{
  shared-box = {
    security.acme.certs."my-site.tld".credentialsFile = "/run/keys/acme-
credentials.secret";
    deployment.keys."acme-credentials.secret" = {
      # Alternatively, `text` (string) or `keyFile` (path to file)
      # may be specified.
      keyCommand = [ "vault" "read" "-field=env" "secret/dns01" ];

      destDir = "/run/keys";        # Default: /run/keys
      user = "acme";                # Default: root
      group = "nginx";              # Default: root
      permissions = "0640";         # Default: 0600

      uploadAt = "pre-activation"; # Default: pre-activation, Alternative:
post-activation
    };
    # Rest of configuration...
  };
}
```

Take note that if you use the default path (`/run/keys`), the secret files are only stored in-memory and will not survive reboots. To upload your secrets without performing a full deployment, use `colmena upload-keys` .

Key Services

For each secret file deployed using `deployment.keys` , a systemd service with the name of `${name}-key.service` is created (`acme-credentials.secret-key.service` for the example above). This unit is only active when the corresponding file is present, allowing you to set up dependencies for services requiring secret files to function.

Key Permissions

The `/run/keys` directory is owned by the `keys` group. If you are using a systemd service running as a non-root user, you will likely need to add:

```
SupplementaryGroups = [ "keys" ];
```

to your service configuration.

Flakes

If you are using flakes, Nix will copy the entire flake (everything tracked by git) into the Nix store during evaluation. This means that all files as checked out by git are world-readable, including the ones managed by filter-based encryption tools like `git-crypt`. To use `deployment.keys.<name>.keyFile` with flakes without having the secrets copied to the Nix store, a quoted absolute path can be used.

Ad Hoc Evaluation

Sometimes you may want to extract values from your Hive configuration for consumption in another program (e.g., OctoDNS). To do that, create a `.nix` file with a lambda:

```
{ nodes, pkgs, lib, ... }:
# Feels like a NixOS module - But you can return any JSON-serializable value
lib.attrsets.mapAttrs (k: v: v.config.deployment.targetHost) nodes
```

Then you can obtain a JSON output with:

```
$ colmena eval target-hosts.nix
{"alpha":"fd12:3456::1","beta":"fd12:3456::2"}
```

You can also specify an expression directly on the command line:

```
$ colmena eval -E '{ nodes, pkgs, lib, ... }: ...'
```

Instantiation

You may directly instantiate an expression that evaluates to a derivation:

```
$ colmena eval --instantiate -E '{ nodes, ... }:
nodes.alpha.config.boot.kernelPackages.kernel'
/nix/store/7ggmhnwvywrqcd1z2sdpan8afz55sw7z-linux-5.14.14.drv
```

Interactive REPL

To explore the configurations interactively, start a REPL session with `colmena repl`:

```
$ colmena repl
[INFO ] Using flake: git+file:///home/user/cluster
Welcome to Nix 2.10.3. Type :? for help.

Loading installable ''...
Added 3 variables.
nix-repl> nodes.alpha.config.deployment.targetHost
"fd12:3456::1"
```

Parallelism

Colmena is built from the ground up to support parallel deployments. Evaluation, build, and deployment of node configurations can happen at the same time.

Configuration

The parallelism of Colmena can be controlled through two flags:

`--limit <number>`

Number of hosts to deploy at once in the final step (pushing closures and activating new profiles). The default value is 10.

`--eval-node-limit <number>`

By default, Colmena will automatically determine the maximum number of nodes to evaluate at the same time according to available RAM. This flag allows you to set the limit to a predetermined value.

Parallel Evaluation (Experimental)

By default, Colmena evaluates nodes in batches according to available RAM using Nix's built-in single-threaded evaluator. Experimental support is available for using nix-eval-jobs as the evaluator.

When nix-eval-jobs is enabled via `--evaluator streaming`, evaluation is parallelized with deployment processes kicked off as individual nodes finish evaluating.

Remote Builds

If the host running Colmena is not powerful enough, consider offloading the actual builds to remote machines. Colmena supports two ways to achieve this:

Using Colmena's `deployment.buildOnTarget`

If you set `deployment.buildOnTarget = true;` for a node, then the actual build process will be initiated on the node itself. Colmena will evaluate the configuration locally before copying the derivations to the target node. You can temporarily enable this for all nodes by passing `--build-on-target` on the command line, or disable it with `--no-build-on-target`.

This is most useful in scenarios where the machine running Colmena is bandwidth-constrained, or it's inconvenient to configure designated builders beforehand. With this method, the build results will *not* be copied back to the local machine or otherwise shared across the target nodes. If you have custom packages used on multiple nodes, the work required to build those packages will be duplicated across the nodes.

Using the native distributed build feature in Nix

When distributed build is enabled, Nix will transparently forward builds to the configured builders. After the builds are done, Nix will copy the results back to the local machine.

Builders can either be configured globally or in your configuration with `meta.machinesFile`.

Examples

This section contains examples of setups people commonly use:

- **Multi-Architecture Deployments** - Deploying to hosts running a foreign architecture

Multi-Architecture Deployments

You can deploy to hosts running different architectures with a single configuration. There are two ways to achieve this:

Using `binfmt` Emulation

On Linux hosts, you can run builds through transparent binary emulation using QEMU and binfmt-misc.

This following example sets up binfmt, allowing an X86-64 host (`laptop`) to build derivations for an AArch64 host (`rpi`) through QEMU:

```
{
  # The NixOS machine you are running Colmena on (x86_64-linux)
  laptop = { pkgs, ... }: {
    # Enable binfmt emulation for aarch64-linux
    boot.binfmt.emulatedSystems = [ "aarch64-linux" ];

    # ... Rest of configuration ...
  };

  # The remote machine running a foreign architecture (aarch64-linux)
  rpi = { pkgs, ... }: {
    # Override nixpkgs architecture
    nixpkgs.system = "aarch64-linux";

    # ... Rest of configuration ...
  };
}
```

(For Flake users, the above attribute set is the value of `outputs.colmena` *)*

First, deploy the local configuration with `colmena apply-local --sudo` . For more information on what is required on the local system, see Local Deployment.

After the new configuration is activated, binfmt emulation will be set up on the local machine. You can then deploy to the `rpi` node with `colmena apply --on rpi` .

Building Remotely

If the remote nodes are powerful enough, you may also execute builds on them directly. See Remote Builds for more details.

Reference

You are currently reading **the unstable version** of the Colmena Manual, built against the tip of the development branch.

This section contains detailed listings of options and parameters accepted by Colmena:

Deployment Options

You are currently reading **the unstable version** of the Colmena Manual, built against the tip of the development branch.

Colmena adds a set of extra options that can be used in your NixOS configurations under the `deployment` prefix.

deployment.allowLocalDeployment

Allow the configuration to be applied locally on the host running Colmena.

For local deployment to work, all of the following must be true:

- The node must be running NixOS.
- The node must have deployment.allowLocalDeployment set to true.
- The node's networking.hostName must match the hostname.

To apply the configurations locally, run `colmena apply-local`. You can also set deployment.targetHost to null if the nost is not accessible over SSH (only local deployment will be possible).

Type: boolean

Default: `false`

deployment.buildOnTarget

Whether to build the system profiles on the target node itself.

When enabled, Colmena will copy the derivation to the target node and initiate the build there. This avoids copying back the build results involved with the native distributed build feature. Furthermore, the `build` goal will be equivalent to the `push` goal. Since builds happen on the target node, the results are automatically "pushed" and won't exist in the local Nix store.

You can temporarily override per-node settings by passing `--build-on-target` (enable for all nodes) or `--no-build-on-target` (disable for all nodes) on the command line.

Type: boolean

Default: `false`

deployment.keys

A set of secrets to be deployed to the node.

Secrets are transferred to the node out-of-band and never ends up in the Nix store.

Type: attribute set of (submodule)

Default: `{ }`

deployment.keys.<name>.destDir

Destination directory on the host.

Type: absolute path

Default: `"/run/keys"`

deployment.keys.<name>.group

The group that will own the file.

Type: string

Default: `"root"`

deployment.keys.<name>.keyCommand

Command to run to generate the key. One of `text` , `keyCommand` and `keyFile` must be set.

Type: null or (list of string)

Default: `null`

deployment.keys.<name>.keyFile

Path of the local file to read the key from. One of `text` , `keyCommand` and `keyFile` must be set.

Type: null or absolute path

Default: `null`

deployment.keys.<name>.name

File name of the key.

Type: string

Default: `"‹name›"`

deployment.keys.<name>.permissions

Permissions to set for the file.

Type: string

Default: `"0600"`

deployment.keys.<name>.text

Content of the key. One of `text` , `keyCommand` and `keyFile` must be set.

Type: null or string

Default: `null`

deployment.keys.<name>.uploadAt

When to upload the keys.

- pre-activation (default): Upload the keys before activating the new system profile.
- post-activation: Upload the keys after successfully activating the new system profile.

For `colmena upload-keys`, all keys are uploaded at the same time regardless of the configuration here.

Type: one of “pre-activation”, “post-activation”

Default: `"pre-activation"`

deployment.keys.<name>.user

The group that will own the file.

Type: string

Default: `"root"`

deployment.privilegeEscalationCommand

Command to use to elevate privileges when activating the new profiles on SSH hosts.

This is used on SSH hosts when `deployment.targetUser` is not `root`. The user must be allowed to use the command non-interactively.

Type: list of string

Default:

```
[
  "sudo"
  "-H"
  "--"
]
```

deployment.replaceUnknownProfiles

Allow a configuration to be applied to a host running a profile we have no knowledge of. By setting this option to false, you reduce the likelyhood of rolling back changes made via another Colmena user.

Unknown profiles are usually the result of either:

- The node had a profile applied, locally or by another Colmena.
- The host running Colmena garbage-collecting the profile.

To force profile replacement on all targeted nodes during apply, use the flag `--force-replace-unknown-profiles`.

Type: boolean

Default: `true`

deployment.sshOptions

Extra SSH options to pass to the SSH command.

Type: list of string

Default: `[ ]`

deployment.tags

A list of tags for the node.

Can be used to select a group of nodes for deployment.

Type: list of string

Default: `[ ]`

deployment.targetHost

The target SSH node for deployment.

By default, the node’s attribute name will be used. If set to null, only local deployment will be supported.

Type: null or string

Default: `"nixos"`

deployment.targetPort

The target SSH port for deployment.

By default, the port is the standard port (22) or taken from your ssh_config.

Type: null or (unsigned integer, meaning >=0)

Default: `null`

deployment.targetUser

The user to use to log into the remote node. If set to null, the target user will not be specified in SSH invocations.

Type: null or string

Default: `"root"`

Meta Options

You are currently reading **the unstable version** of the Colmena Manual, built against the tip of the development branch.

The following is a list of options that can be added to the `meta` attribute set.

For compatibility with NixOps, you may name it `network` instead of `meta`. However, you cannot specify both at the same time.

allowApplyAll

Whether to allow deployments without a node filter set.

If set to false, a node filter must be specified with `--on` when deploying.

It helps prevent accidental deployments to the entire cluster when tags are used (e.g., `@production` and `@staging`).

Type: boolean

Default: `true`

description

A short description for the configuration.

Type: string

Default: `"A Colmena Hive"`

machinesFile

Use the machines listed in this file when building this hive configuration.

If your Colmena host has nix configured to allow for remote builds (for nix-daemon, your user being included in trusted-users) you can set a machines file that will be passed to the underlying nix-store command during derivation realization as a builders option. For example, if you support multiple orginizations each with their own build machine(s) you can ensure that builds only take place on your local machine and/or the machines specified in this file.

See https://nixos.org/manual/nix/stable/advanced-topics/distributed-builds for the machine specification format.

This option is ignored when builds are initiated on the remote nodes themselves via `deployment.buildOnTarget` or `--build-on-target`. To still use the Nix distributed build functionality, configure the builders on the target nodes with `nix.buildMachines`.

Type: null or absolute path

Default: `null`

name

The name of the configuration.

Type: string

Default: `"hive"`

nixpkgs

The pinned Nixpkgs package set. Accepts one of the following:

- A path to a Nixpkgs checkout
- The Nixpkgs lambda (e.g., import <nixpkgs>)
- An initialized Nixpkgs attribute set

This option must be specified when using Flakes.

Type: unspecified value

Default: `null`

nodeNixpkgs

Node-specific Nixpkgs pins.

Type: attribute set of unspecified value

Default: `{ }`

nodeSpecialArgs

Node-specific special args.

Type: attribute set of unspecified value

Default: `{ }`

specialArgs

A set of special arguments to be passed to NixOS modules.

This will be merged into the `specialArgs` used to evaluate the NixOS configurations.

Type: attribute set of unspecified value

Default: `{ }`

Command Line Options

You are currently reading **the unstable version** of the Colmena Manual, built against the tip of the development branch.

The following are the help messages that will be printed when you invoke any sub-command with `--help`:

`colmena`

```
NixOS deployment tool

Colmena helps you deploy to multiple hosts running NixOS.
For more details, read the manual at <https://colmena.cli.rs/0.5>.


Note: You are using a pre-release version of Colmena, so the
supported options may be different from
what's in the manual.

Usage: colmena [OPTIONS] <COMMAND>

Commands:
  apply        Apply configurations on remote machines
  apply-local  Apply configurations on the local machine
  build        Build configurations but not push to remote machines
  eval         Evaluate an expression using the complete
configuration
  upload-keys  Upload keys to remote hosts
  exec         Run a command on remote machines
  repl         Start an interactive REPL with the complete
configuration
  nix-info     Show information about the current Nix installation
  help         Print this message or the help of the given
subcommand(s)

Options:
      --show-trace
          Show debug information for Nix commands

          Passes --show-trace to Nix commands

      --impure
          Allow impure expressions

          Passes --impure to Nix commands
      --nix-option <NAME> <VALUE>
          Passes an arbitrary option to Nix commands

          This only works when building locally.

  -f, --config <CONFIG>
          If this argument is not specified, Colmena will search
upwards from the current working
          directory for a file named "flake.nix" or "hive.nix". This
behavior is disabled if
          --config/-f is given explicitly.

          For a sample configuration, check the manual at
<https://colmena.cli.rs/0.5>.

  -h, --help
          Print help (see a summary with '-h')

  -V, --version
          Print version

      --color <WHEN>
          When to colorize the output

          By default, Colmena enables colorized output when the
terminal supports it.

          It's also possible to specify the preference using
environment variables. See
          <https://bixense.com/clicolors>.

          [default: auto]

          Possible values:
          - auto:   Detect automatically
          - always: Always display colors
          - never:  Never display colors
```

colmena apply

Apply configurations on remote machines

```
Usage: colmena apply [OPTIONS] [GOAL]

Arguments:
  [GOAL]
          Deployment goal
          Same as the targets for switch-to-configuration, with the
following extra pseudo-goals:

          - build: Only build the system profiles - push: Only copy
the closures to remote nodes -
          keys: Only upload the keys to the remote nodes

          `switch` is the default goal unless `--reboot` is passed,
in which case `boot` is the
          default.

          [default: switch]

          Possible values:
          - build:        Build the configurations only
          - push:         Push the closures only
          - switch:       Make the configuration the boot default and
activate now
          - boot:         Make the configuration the boot default
          - test:         Activate the configuration, but don't make
it the boot default
          - dry-activate: Show what would be done if this
configuration were activated
          - upload-keys:  Only upload keys

Options:
      --eval-node-limit <LIMIT>
          Evaluation node limit

          Limits the maximum number of hosts to be evaluated at once.
The evaluation process is
          RAM-intensive. The default behavior is to limit the maximum
number of hosts evaluated at
          the same time based on naive heuristics.

          Set to 0 to disable the limit.

          [default: auto]

  -p, --parallel <LIMIT>
          Deploy parallelism limit

          Limits the maximum number of hosts to be deployed in
parallel.

          Set to 0 to disable parallelism limit.

          [default: 10]

      --show-trace
```

```
          Show debug information for Nix commands

          Passes --show-trace to Nix commands

      --impure
          Allow impure expressions

          Passes --impure to Nix commands

      --keep-result
          Create GC roots for built profiles.

          The built system profiles will be added as GC roots so that
they will not be removed by
          the garbage collector. The links will be created under
`.gcroots` in the directory the
          Hive configuration is located.

      --nix-option <NAME> <VALUE>
          Passes an arbitrary option to Nix commands

          This only works when building locally.

  -v, --verbose
          Be verbose

          Deactivates the progress spinner and prints every line of
output.

      --no-keys
          Do not upload keys

          By default, Colmena will upload secret keys set in
`deployment.keys` before deploying the
          new profile on a node. To upload keys without building or
deploying the rest of the
          configuration, use `colmena upload-keys`.

      --reboot
          Reboot nodes after activation

          Reboots nodes after activation and waits for them to come
back up.

      --no-substitute
          Do not use substitutes

          Disables the use of substituters when copying closures to
the remote host.

      --no-gzip
```

```
          Do not use gzip

          Disables the use of gzip when copying closures to the
remote host.

      --build-on-target
          Build the system profiles on the target nodes

          If enabled, the system profiles will be built on the target
nodes themselves, not on the
          host running Colmena. This overrides per-node preferences
set in
          `deployment.buildOnTarget`. To temporarily disable remote
build on all nodes, use
          `--no-build-on-target`.

      --force-replace-unknown-profiles
          Ignore all targeted nodes
`deployment.replaceUnknownProfiles` setting

          If `deployment.replaceUnknownProfiles` is set for a target,
using this switch will treat
          `deployment.replaceUnknownProfiles` as though it was set to
`true` and perform unknown
          profile replacement.

      --evaluator <EVALUATOR>
          The evaluator to use (experimental)

          If set to `chunked` (default), evaluation of nodes will
happen in batches. If set to
          `streaming`, the experimental streaming evaluator (nix-
eval-jobs) will be used and nodes
          will be evaluated in parallel.

          This is an experimental feature.

          [default: chunked]
          [possible values: chunked, streaming]

      --on <NODES>
          Node selector

          Select a list of nodes to deploy to. The list is comma-
separated and globs are supported.
          To match tags, prepend the filter by @. Valid examples:

          - host1,host2,host3 - edge-* - edge-*,core-* - @a-
tag,@tags-can-have-*

  -f, --config <CONFIG>
```

```
          If this argument is not specified, Colmena will search 
upwards from the current working
          directory for a file named "flake.nix" or "hive.nix". This 
behavior is disabled if
          --config/-f is given explicitly.

          For a sample configuration, check the manual at 
<https://colmena.cli.rs/0.5>.

  -h, --help
          Print help (see a summary with '-h')

      --color <WHEN>
          When to colorize the output

          By default, Colmena enables colorized output when the 
terminal supports it.

          It's also possible to specify the preference using 
environment variables. See
          <https://bixense.com/clicolors>.

          [default: auto]

          Possible values:
          - auto:   Detect automatically
          - always: Always display colors
          - never:  Never display colors
```

colmena apply-local

Apply configurations on the local machine

```
Usage: colmena apply-local [OPTIONS] [GOAL]

Arguments:
  [GOAL]
          Deployment goal

          Same as the targets for switch-to-configuration. "push" is 
noop in apply-local.

          [default: switch]

          Possible values:
          - build:        Build the configurations only
          - push:         Push the closures only
          - switch:       Make the configuration the boot default and
```

activate now
          - **boot:**   Make the configuration the boot default
          - **test:**   Activate the configuration, but don't make it the boot default
          - **dry-activate:** Show what would be done if this configuration were activated
          - **upload-keys:** Only upload keys

Options:

      --sudo
          Attempt to escalate privileges if not run as root

      --show-trace
          Show debug information for Nix commands

          Passes --show-trace to Nix commands

  -v, --verbose
          Be verbose

          Deactivates the progress spinner and prints every line of output.

      --impure
          Allow impure expressions

          Passes --impure to Nix commands

      --no-keys
          Do not deploy keys

          Do not deploy secret keys set in `deployment.keys`. By default, Colmena will deploy keys
          set in `deployment.keys` before activating the profile on this host.

      --nix-option <NAME> <VALUE>
          Passes an arbitrary option to Nix commands

          This only works when building locally.

      --node <NODE>
          Override the node name to use

  -f, --config <CONFIG>
          If this argument is not specified, Colmena will search upwards from the current working
          directory for a file named "flake.nix" or "hive.nix". This behavior is disabled if
          --config/-f is given explicitly.

```
          For a sample configuration, check the manual at
<https://colmena.cli.rs/0.5>.

  -h, --help
          Print help (see a summary with '-h')

      --color <WHEN>
          When to colorize the output

          By default, Colmena enables colorized output when the
terminal supports it.

          It's also possible to specify the preference using
environment variables. See
          <https://bixense.com/clicolors>.

          [default: auto]

          Possible values:
          - auto:   Detect automatically
          - always: Always display colors
          - never:  Never display colors
```

colmena build

Build configurations but not push to remote machines

This subcommand behaves as if you invoked `apply` with the `build` goal.

```
Usage: colmena build [OPTIONS]

Options:
      --eval-node-limit <LIMIT>
          Evaluation node limit

          Limits the maximum number of hosts to be evaluated at once.
The evaluation process is
          RAM-intensive. The default behavior is to limit the maximum
number of hosts evaluated at
          the same time based on naive heuristics.

          Set to 0 to disable the limit.

          [default: auto]

  -p, --parallel <LIMIT>
          Deploy parallelism limit
          Limits the maximum number of hosts to be deployed in
parallel.

          Set to 0 to disable parallelism limit.

          [default: 10]

      --show-trace
          Show debug information for Nix commands

          Passes --show-trace to Nix commands

      --impure
          Allow impure expressions

          Passes --impure to Nix commands

      --keep-result
          Create GC roots for built profiles.

          The built system profiles will be added as GC roots so that
they will not be removed by
          the garbage collector. The links will be created under
`.gcroots` in the directory the
          Hive configuration is located.

      --nix-option <NAME> <VALUE>
          Passes an arbitrary option to Nix commands

          This only works when building locally.

  -v, --verbose
          Be verbose

          Deactivates the progress spinner and prints every line of
output.

      --no-keys
          Do not upload keys

          By default, Colmena will upload secret keys set in
`deployment.keys` before deploying the
          new profile on a node. To upload keys without building or
deploying the rest of the
          configuration, use `colmena upload-keys`.

      --reboot
          Reboot nodes after activation

          Reboots nodes after activation and waits for them to come
```

```
back up.

      --no-substitute
          Do not use substitutes

          Disables the use of substituters when copying closures to
the remote host.

      --no-gzip
          Do not use gzip

          Disables the use of gzip when copying closures to the
remote host.

      --build-on-target
          Build the system profiles on the target nodes

          If enabled, the system profiles will be built on the target
nodes themselves, not on the
          host running Colmena. This overrides per-node preferences
set in
          `deployment.buildOnTarget`. To temporarily disable remote
build on all nodes, use
          `--no-build-on-target`.

      --force-replace-unknown-profiles
          Ignore all targeted nodes
`deployment.replaceUnknownProfiles` setting

          If `deployment.replaceUnknownProfiles` is set for a target,
using this switch will treat
          `deployment.replaceUnknownProfiles` as though it was set to
`true` and perform unknown
          profile replacement.

      --evaluator <EVALUATOR>
          The evaluator to use (experimental)

          If set to `chunked` (default), evaluation of nodes will
happen in batches. If set to
          `streaming`, the experimental streaming evaluator (nix-
eval-jobs) will be used and nodes
          will be evaluated in parallel.

          This is an experimental feature.

          [default: chunked]
          [possible values: chunked, streaming]

      --on <NODES>
          Node selector
          Select a list of nodes to deploy to. The list is comma-
separated and globs are supported.
          To match tags, prepend the filter by @. Valid examples:

          - host1,host2,host3 - edge-* - edge-*,core-* - @a-
tag,@tags-can-have-*

  -f, --config <CONFIG>
          If this argument is not specified, Colmena will search
upwards from the current working
          directory for a file named "flake.nix" or "hive.nix". This
behavior is disabled if
          --config/-f is given explicitly.

          For a sample configuration, check the manual at
<https://colmena.cli.rs/0.5>.

  -h, --help
          Print help (see a summary with '-h')

      --color <WHEN>
          When to colorize the output

          By default, Colmena enables colorized output when the
terminal supports it.

          It's also possible to specify the preference using
environment variables. See
          <https://bixense.com/clicolors>.

          [default: auto]

          Possible values:
          - auto:   Detect automatically
          - always: Always display colors
          - never:  Never display colors
```

colmena upload-keys

Upload keys to remote hosts

This subcommand behaves as if you invoked `apply` with the pseudo `keys` goal.

Usage: **colmena upload-keys** [OPTIONS]

Options:

```
      --eval-node-limit <LIMIT>
          Evaluation node limit

          Limits the maximum number of hosts to be evaluated at once.
The evaluation process is
          RAM-intensive. The default behavior is to limit the maximum
number of hosts evaluated at
          the same time based on naive heuristics.

          Set to 0 to disable the limit.

          [default: auto]

  -p, --parallel <LIMIT>
          Deploy parallelism limit

          Limits the maximum number of hosts to be deployed in
parallel.

          Set to 0 to disable parallelism limit.

          [default: 10]

      --show-trace
          Show debug information for Nix commands

          Passes --show-trace to Nix commands

      --impure
          Allow impure expressions

          Passes --impure to Nix commands

      --keep-result
          Create GC roots for built profiles.

          The built system profiles will be added as GC roots so that
they will not be removed by
          the garbage collector. The links will be created under
`.gcroots` in the directory the
          Hive configuration is located.

      --nix-option <NAME> <VALUE>
          Passes an arbitrary option to Nix commands

          This only works when building locally.

  -v, --verbose
          Be verbose

          Deactivates the progress spinner and prints every line of
```

```
output.

       --no-keys
           Do not upload keys

           By default, Colmena will upload secret keys set in
`deployment.keys` before deploying the
           new profile on a node. To upload keys without building or
deploying the rest of the
           configuration, use `colmena upload-keys`.

       --reboot
           Reboot nodes after activation

           Reboots nodes after activation and waits for them to come
back up.

       --no-substitute
           Do not use substitutes

           Disables the use of substituters when copying closures to
the remote host.

       --no-gzip
           Do not use gzip

           Disables the use of gzip when copying closures to the
remote host.

       --build-on-target
           Build the system profiles on the target nodes

           If enabled, the system profiles will be built on the target
nodes themselves, not on the
           host running Colmena. This overrides per-node preferences
set in
           `deployment.buildOnTarget`. To temporarily disable remote
build on all nodes, use
           `--no-build-on-target`.

       --force-replace-unknown-profiles
           Ignore all targeted nodes
`deployment.replaceUnknownProfiles` setting

           If `deployment.replaceUnknownProfiles` is set for a target,
using this switch will treat
           `deployment.replaceUnknownProfiles` as though it was set to
`true` and perform unknown
           profile replacement.

       --evaluator <EVALUATOR>
```

```
          The evaluator to use (experimental)

          If set to `chunked` (default), evaluation of nodes will
happen in batches. If set to
          `streaming`, the experimental streaming evaluator (nix-
eval-jobs) will be used and nodes
          will be evaluated in parallel.

          This is an experimental feature.

          [default: chunked]
          [possible values: chunked, streaming]

      --on <NODES>
          Node selector

          Select a list of nodes to deploy to. The list is comma-
separated and globs are supported.
          To match tags, prepend the filter by @. Valid examples:

          - host1,host2,host3 - edge-* - edge-*,core-* - @a-
tag,@tags-can-have-*

  -f, --config <CONFIG>
          If this argument is not specified, Colmena will search
upwards from the current working
          directory for a file named "flake.nix" or "hive.nix". This
behavior is disabled if
          --config/-f is given explicitly.

          For a sample configuration, check the manual at
<https://colmena.cli.rs/0.5>.

  -h, --help
          Print help (see a summary with '-h')

      --color <WHEN>
          When to colorize the output

          By default, Colmena enables colorized output when the
terminal supports it.

          It's also possible to specify the preference using
environment variables. See
          <https://bixense.com/clicolors>.

          [default: auto]

          Possible values:
          - auto:   Detect automatically
          - always: Always display colors
```

- **never:** Never display colors

colmena eval

Evaluate an expression using the complete configuration

Your expression should take an attribute set with keys `pkgs`, `lib` and `nodes` (like a NixOS
module) and return a JSON-serializable value. For example, to retrieve the configuration of one
node, you may write something like:

```
{ nodes, ... }: nodes.node-a.config.networking.hostName
```

Usage: **colmena eval** [OPTIONS] [FILE]

Arguments:
  [FILE]
          The .nix file containing the expression

Options:
  -E <EXPRESSION>
          The Nix expression

      --instantiate
          Actually instantiate the expression

      --show-trace
          Show debug information for Nix commands

          Passes --show-trace to Nix commands

      --impure
          Allow impure expressions

          Passes --impure to Nix commands

      --nix-option <NAME> <VALUE>
          Passes an arbitrary option to Nix commands

          This only works when building locally.

  -f, --config <CONFIG>
          If this argument is not specified, Colmena will search upwards from the current working
          directory for a file named "flake.nix" or "hive.nix". This behavior is disabled if
          --config/-f is given explicitly.

```
          For a sample configuration, check the manual at
<https://colmena.cli.rs/0.5>.

  -h, --help
          Print help (see a summary with '-h')

      --color <WHEN>
          When to colorize the output

          By default, Colmena enables colorized output when the
terminal supports it.

          It's also possible to specify the preference using
environment variables. See
          <https://bixense.com/clicolors>.

          [default: auto]

          Possible values:
          - auto:   Detect automatically
          - always: Always display colors
          - never:  Never display colors
```

colmena exec

Run a command on remote machines

```
Usage: colmena exec [OPTIONS] <COMMAND>...

Arguments:
  <COMMAND>...
          Command to run

          It's recommended to use -- to separate Colmena options from
the command to run. For
          example:

          colmena exec --on @routers -- tcpdump -vni any ip[9] == 89

Options:
  -p, --parallel <LIMIT>
          Deploy parallelism limit

          Limits the maximum number of hosts to run the command in
parallel.

          In `colmena exec`, the parallelism limit is disabled (0) by
```

```
default.

          [default: 0]

      --show-trace
          Show debug information for Nix commands

          Passes --show-trace to Nix commands

  -v, --verbose
          Be verbose

          Deactivates the progress spinner and prints every line of
output.

      --impure
          Allow impure expressions

          Passes --impure to Nix commands

      --on <NODES>
          Node selector

          Select a list of nodes to deploy to. The list is comma-
separated and globs are supported.
          To match tags, prepend the filter by @. Valid examples:

          - host1,host2,host3 - edge-* - edge-*,core-* - @a-
tag,@tags-can-have-*

      --nix-option <NAME> <VALUE>
          Passes an arbitrary option to Nix commands

          This only works when building locally.

  -f, --config <CONFIG>
          If this argument is not specified, Colmena will search
upwards from the current working
          directory for a file named "flake.nix" or "hive.nix". This
behavior is disabled if
          --config/-f is given explicitly.

          For a sample configuration, check the manual at
<https://colmena.cli.rs/0.5>.

  -h, --help
          Print help (see a summary with '-h')

      --color <WHEN>
          When to colorize the output
          By default, Colmena enables colorized output when the
terminal supports it.

          It's also possible to specify the preference using
environment variables. See
          <https://bixense.com/clicolors>.

          [default: auto]

          Possible values:
          - auto:   Detect automatically
          - always: Always display colors
          - never:  Never display colors
```

colmena nix-info

```
Show information about the current Nix installation

Usage: colmena nix-info [OPTIONS]

Options:
      --show-trace
          Show debug information for Nix commands

          Passes --show-trace to Nix commands

      --impure
          Allow impure expressions

          Passes --impure to Nix commands

      --nix-option <NAME> <VALUE>
          Passes an arbitrary option to Nix commands

          This only works when building locally.

  -f, --config <CONFIG>
          If this argument is not specified, Colmena will search
upwards from the current working
          directory for a file named "flake.nix" or "hive.nix". This
behavior is disabled if
          --config/-f is given explicitly.

          For a sample configuration, check the manual at
<https://colmena.cli.rs/0.5>.

  -h, --help
          Print help (see a summary with '-h')

      --color <WHEN>
          When to colorize the output

          By default, Colmena enables colorized output when the
terminal supports it.

          It's also possible to specify the preference using
environment variables. See
          <https://bixense.com/clicolors>.

          [default: auto]

          Possible values:
          - auto:   Detect automatically
          - always: Always display colors
          - never:  Never display colors
```

colmena repl

```
Start an interactive REPL with the complete configuration

In the REPL, you can inspect the configuration interactively with tab
completion. The node
configurations are accessible under the `nodes` attribute set.

Usage: colmena repl [OPTIONS]

Options:
      --show-trace
          Show debug information for Nix commands

          Passes --show-trace to Nix commands

      --impure
          Allow impure expressions

          Passes --impure to Nix commands

      --nix-option <NAME> <VALUE>
          Passes an arbitrary option to Nix commands

          This only works when building locally.

  -f, --config <CONFIG>
          If this argument is not specified, Colmena will search
upwards from the current working
          directory for a file named "flake.nix" or "hive.nix". This
behavior is disabled if
          --config/-f is given explicitly.

          For a sample configuration, check the manual at
<https://colmena.cli.rs/0.5>.

  -h, --help
          Print help (see a summary with '-h')

      --color <WHEN>
          When to colorize the output

          By default, Colmena enables colorized output when the
terminal supports it.

          It's also possible to specify the preference using
environment variables. See
          <https://bixense.com/clicolors>.

          [default: auto]
```

Possible values:

- **auto**:   Detect automatically
- **always**: Always display colors
- **never**: Never display colors

Release Notes

Release 0.4.0 (2023/05/14)

- Flake evaluation is now actually pure by default. To enable impure expressions, pass `--impure`.
- `--reboot` is added to trigger a reboot and wait for the node to come back up.
- The target user is no longer explicitly set when `deployment.targetUser` is null (#91).
- In `apply-local`, we now only escalate privileges during activation (#85).
- Impure overlays are no longer imported by default if a path is specified in `meta.nixpkgs` (#39)
- GC roots are now created right after the builds are complete, as opposed to after activation.
- The `meta.allowApplyAll` option has been added. If set to false, deployments without a node filter (`--on`) are disallowed (#95).
- The `--no-substitutes` option under the `apply` subcommand has been renamed to `--no-substitute` (#59).
- The `meta.nodeSpecialArgs` option has been added. It allows specifying node-specific `specialArgs` passed to NixOS modules (#100).
- The `repl` subcommand has been added. It allows you to start an interactive REPL with access to the complete node configurations.
- The default goal for `colmena apply` is now `boot` if `--reboot` is specified, and `switch` otherwise (#113).
- Post-activation keys are now uploaded after the reboot if `--reboot` is specified (#113).
- Flake-enabled deployments now use the new SSH store protocol (`ssh-ng://`).

Release 0.3.2 (2022/09/29)

- Fixed: Key services were using the deprecated `inotifyTools` alias removed from `nixos-unstable` (NixOS/nixpkgs#192681).

Release 0.3.1 (2022/08/18)

- Fixed: Streaming evaluation fails for node names containing periods (#92)
- Fixed: Streaming evaluation fails in non-flake deployments with relative paths (#107)
- Fixed: `colmena apply-local` returning non-zero exit code when successful (#111)

Release 0.3.0 (2022/04/27)

- Remote builds are now supported (#33).
- Streaming evaluation powered by nix-eval-jobs is now available as an experimental feature (`--evaluator streaming`).
- Colmena can now run on macOS to deploy to NixOS hosts using remote building.
- It's now possible to configure output colorization via the CLI and environment variables. Colmena follows the clicolors standard.
- A systemd unit (`${name}-key.service`) is now created for each secret file deployed using `deployment.keys` (#48).
- Node enumeration is now faster if you do not filter against tags with `--on @tag-name` .
- The main deployment logic has been rewritten to be cleaner and easier to follow.
- There are now end-to-end tests to ensure that the development branch is actually functional as a whole at all times.

Release 0.2.2 (2022/03/08)

This bugfix release fixes NixOS detection so `apply-local` works with the latest changes in `nixos-unstable` (#63). Additionally, `--no-keys` was fixed in `apply-local` .

Release 0.2.1 (2022/01/26)

This bugfix release fixes the issue (#50) where sandboxed documentation builds fail when using the unstable Nixpkgs channel.

Release 0.2.0 (2021/11/18)

This is release 0.2.0, the first stable release of Colmena!

Colmena is a simple, stateless NixOS deployment tool modeled after NixOps and morph, built from the ground up to support parallel deployments.

This release contains the following features:

- Node Tagging
- Local Deployment
- Secrets
- Ad Hoc Evaluation
- Nix Flakes Support
- Parallelism

We now have a User Manual at https://colmena.cli.rs/0.2 containing tutorials, sample configurations as well as a complete listing of supported deployment options.

Contributing

Contributions are welcome!

Code

You can checkout the source code and submit pull requests at the GitHub repo.

By contributing code to Colmena, you agree that your contributions will be available under the MIT License.

Issues & Feature Requests

Bumped into a problem? We would like to fix it! Please open a new issue at the GitHub repo.