Deploying with morph

Introduction to Morph

Morph is a simple, stateless deployment tool for NixOS. Essentially, it's a way to manage one or more servers remotely from a central NixOS configuration, deploying system updates and configuration changes over SSH.

You write configuration for Morph with the same language, syntax and tools as you would use for a local NixOS configuration; in fact, there is very little difference in format between a Morph and a NixOS configuration. In many cases, it's nothing more than an extra object wrapped around the system configurations.

For example, consider the following (simplified) NixOS system configuration, derived from an example in the Morph repository:

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

  services.nginx.enable = true;

  fileSystems = {
    "/" = {
      label = "nixos";
      fsType = "ext4";
    };
  };
}
```

The equivalent Morph configuration would look like this:

> This example is simplified, and you should not actually try deploy this to a server! It will break your connection, as the example configuration does not include an SSH daemon.

```
{
  network = {
    pkgs = import (builtins.fetchTarball "https://github.com/NixOS/nixpkgs/archive/nixos-unstable.tar.gz") {};
    description = "Basic Morph configuration";
  };
```

```
  "server01.example.com" = { pkgs }: {
    boot.loader.systemd-boot.enable = true;
    boot.loader.efi.canTouchEfiVariables = true;

    services.nginx.enable = true;

    fileSystems = {
      "/" = {
        label = "nixos";
        fsType = "ext4";
      };
    };
  };
}
```

As you can see, the configuration actually didn't change at all - all we've done is add a wrapper object which specifies the default `nixpkgs` source, and the hostname to deploy the configuration on. This is all we need to have a working Morph deployment!

Aside from pushing updates to your server(s), Morph does a few other essential things:

- **Health checks**, ie. verifying that a deployment succeeded by checking that expected services are available afterwards.
- **Uploading secrets,** such as application keys and credentials that should not be in the Nix store.

However, that's about all that it does. This is usually all you need for simpler deployments, but if you need more complex procedures, then a different deployment tool may be a better choice.

> It is easy to change your deployment tool later on, because Morph syntax and configuration structure is so close to that of NixOS itself! This means it's completely okay to start with Morph, and then move to something else if it turns out not to be sufficient.

Stateless?

Some deployment tools, even those for NixOS, are stateful; they keep a database on your own computer of the current state of each server, then each time they are run, they calculate what changed and push those changes to the server. One of the downsides of this model is that you must always deploy to a server from the same computer, since the database can only exist in one place at a time. However, this kind of state can be necessary if the tool also needs to manage

resources such as IaaS services.

When you are just deploying to a standard Linux system like a VPS or bare-metal dedicated server, though, the added complexity and limitations of a stateful deployment tool are not really worth it, and it's often much simpler to use a stateless deployment tool instead. These don't need to keep local state, and always ensure that the remote system is in the exact configuration that your configuration file specifies. Morph is one of those tools.