My NixOS-Powered Homelab

Josh Lee, Altinity

My path to NixOS

Nix-Shell? Home-Manager? Flakes?

```
$ nix-shell -p clickhouse
```

Nix-Shell? Home-Manager? Flakes?

```
$ nix-shell -p clickhouse
```

Join me in the Cloud-Native Databases Devroom later today,
or Monitoring & Observability tomorrow afternoon…

How do I make templated VMs without having to store multi-GB images?

How can I share parameters between VMs?

How do I keep track of the state of each VM?

Why NixOS?

Why NixOS?

- "Forced" GitOps
- Self-documenting
- Portable and reproducible
- Uses familiar tools (systemd, docker)

Why Nix?

- One language for all tools (except k8s?)
- Functional ❤️
- Nix-Modules system is awesome

Agenda

1. NixOS Generators
2. VM-specific Configurations
3. Managing Multiple Hosts
4. Managing Workloads & Services

NixOS Generators

NixOS Generators

```
$ nixos-generate -c proxmox -f configuration.nix
```

VM-specific Configuration

VM Specific Configuration

```
imports = [
  (modulesPath + "/profiles/qemu-guest.nix")
];
```

VM Specific Configuration

```
imports = [
  (modulesPath + "/profiles/qemu-guest.nix")
];

services.qemuGuest.enable = lib.mkDefault true;
```

VM Specific Configuration

```
boot.loader.grub.enable = true;
boot.loader.grub.devices = [ "nodev" ];
```

(Important!) VM Specific Configuration

```
boot.growPartition = true;
```

Enable Remote Updates

```
nix.settings.trusted-users = [ "root" "@wheel" ];
nix.settings.experimental-features = [
    "nix-command"
    "flakes"
];
```

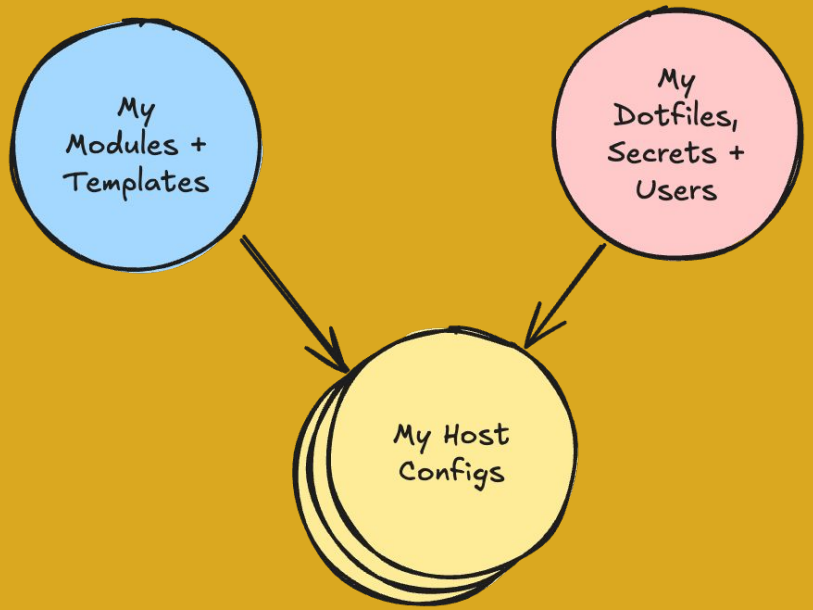
Use Remote Updates

```
$ nixos-rebuild --target-host user@remote-host
--use-remote-sudo
```

Managing the Menagerie

```
hostsDir = ./nix-hosts;
readHost = file: import (hostsDir + ("/" + file));
hostFiles = lib.filter (file: lib.hasSuffix ".nix" file) (lib.attrNames (builtins.readDir
hostsDir));
hostDefinitions = builtins.map (file: readHost file) hostFiles;
makeSystem = host: lib.nixosSystem {
   system = "x86_64-linux";
   modules = [ host ];
};
systems = builtins.map makeSystem hostDefinitions;
configurations = lib.listToAttrs (builtins.map (host: {
   name = host.config.networking.hostName;
   value = host;
}) systems);
```

Managing the Menagerie



Per-host:

Only override what's needed...

```
lib.mkDefault
lib.mkForce
```

Per-host:

Put system.stateVersion in your most-specific configuration file.

Commit it!

What about Workloads?

Deploying Services

```
{ config, lib, ... }:
with lib;
let
    cfg = config.home-cloud.monitoring;
    ports = {
        grafana = 2342;
        prometheus = 9000;
        node_exporter = 9100;
    };
in
{
    options.home-cloud.monitoring = {
        enable = mkEnableOption "monitoring";
    };
    config = mkIf cfg.enable {
        networking.firewall = {
            allowedTCPPorts = [ 80 443 ];
        };

        services.cadvisor.enable = true;
         services.prometheus = {
            ...
            port = ports.node_exporter;
          };
        };
     };
  };
}
```

Deploying Docker Services

Deploying Docker Services

```
{ config, lib, ... }:

with lib;
let

cfg = config.home-cloud.portainer;
in
{
  options.home-cloud.
portainer = {

enable = lib.mkEnableOption "portainer";
  };

config = lib.mkIf cfg.enable {
    virtualisation.oci-containers.
containers = {

portainer = {

image = "portainer/portainer-ce";

ports = [

"8000:8000"

"9443:9443"
        ];

volumes = [

"appdata:/data"

"/var/run/podman/podman.sock:/var/run/docker.sock:Z"
        ];

extraOptions = [

"--privileged"
        ];
      };
    };
  };
}
```

My Tailgate Config

Tailgate (Tailnet Router) Config

```
networking.
hostName = "tailgate";
boot.growPartition = true;

services.tailscale.
enable = true;
services.tailscale.useRoutingFeatures = "both";
system.stateVersion = "23.11";
```

Lessons Learned & Next Steps

Building an OCI Stack

Building an OCI Stack

1. Container Runtime
2. Ingress
3. DNS
4. Shared Storage
5. Monitoring

Oops, I'm building a Kubernetes...

Remember to KISS

Promox VMs vs Nix MicroVMs vs Containers

When should I actually do this?

When should I actually do this?

1. Tight coupling between layers
2. Edge computing
3. Quick experiments
4. Ephemeral VMs

Q&A

Thank You!

Chat with me:
https://altinity.com/slack

Follow me:
https://www.linkedin.com/in/joshuamlee/