cardlay®

DevOps Enabling Your Team

Jacob Aae Mikkelsen

Agenda

- Goals
- Infrastructure
- Infrastructure as Code with Terraform
- Docker Orchestration with Rancher
- Deploying an App

Jacob Aae Mikkelsen

- Senior Software Architect at Cardlay A/S
- GR8Conf EU - Organizing Team Member
- External Associate Professor - University of Southern Denmark
- Groovy Ecosystem Nerd
-  @JacobAae
- Blogs The Grails Diary

Disclaimer

Some knowledge of infrastructure is assumed

Goals

- Define a cloud based infrastructure
  - as code
- Spin up the infrastructure from scratch
- Docker Orchestration tool
  - Deploy a demo app

Motivation

Devops

Devops definition

DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.

— theagileadmin.com/what-is-devops

Devops culture

Primary corollary

DevOps is also characterized by operations staff making use many of the same techniques as developers for their systems work.

— theagileadmin.com/what-is-devops

Reality

In reality, silos exist in IT and DevOps teams are interested in tearing down those silos.

Enable development team to handle deployment and configuration themselves

Requirement

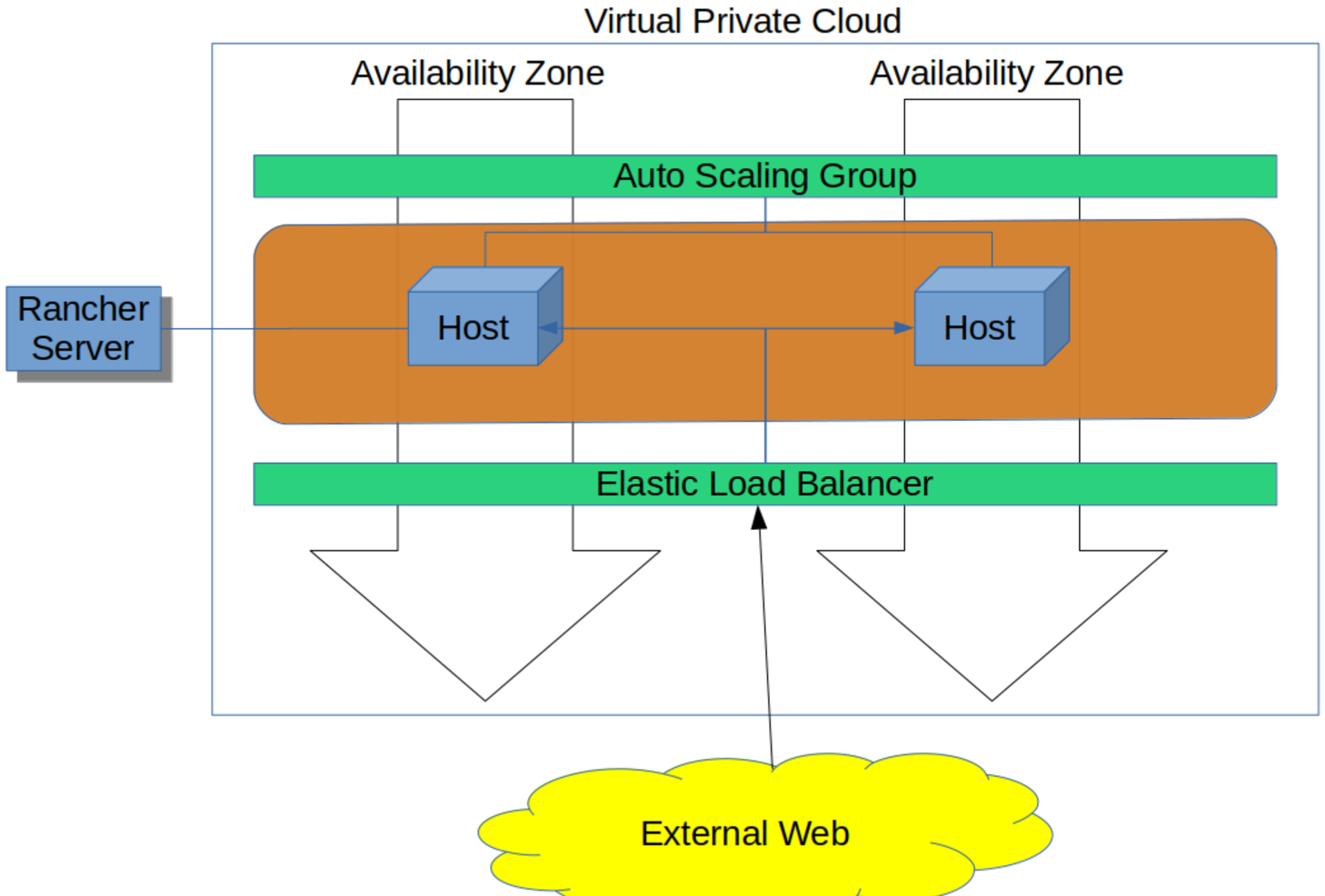
We need to understand the layers below our code

- Hosting platform
- Domain Names Administration (DNS)
- Network
- Automation of infrastructure

Infrastructure

Requirement

AWS VPC with hosts that can serve docker images Orchestrated by a Rancher Server


Virtual Private Cloud
Availability Zone
Availability Zone
Auto Scaling Group
Rancher Server
Host
Host
Elastic Load Balancer
External Web

Provisioning

- Single machine
- Single application server
- Single database server

Tools

- Ansible
- Chef
- Puppet

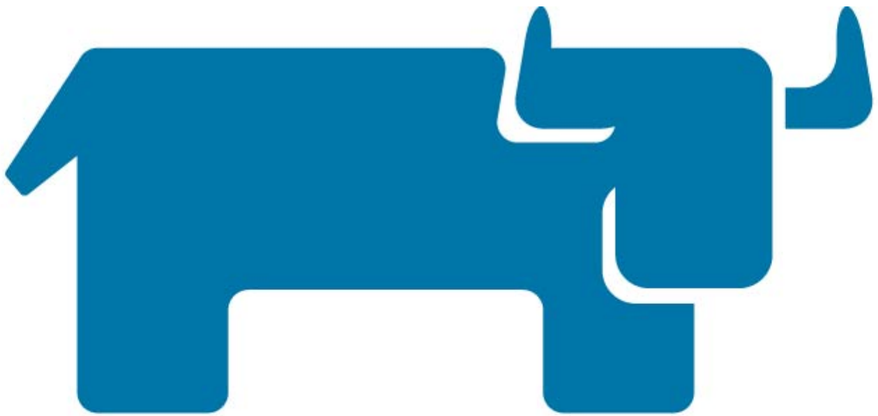
Orchestration

- Making all the singles mingle
- Connects the applikation server to a valid database server
- Networking
- Service discovery

Tools

- Terraform
- CloudFormation

Application Orchestration



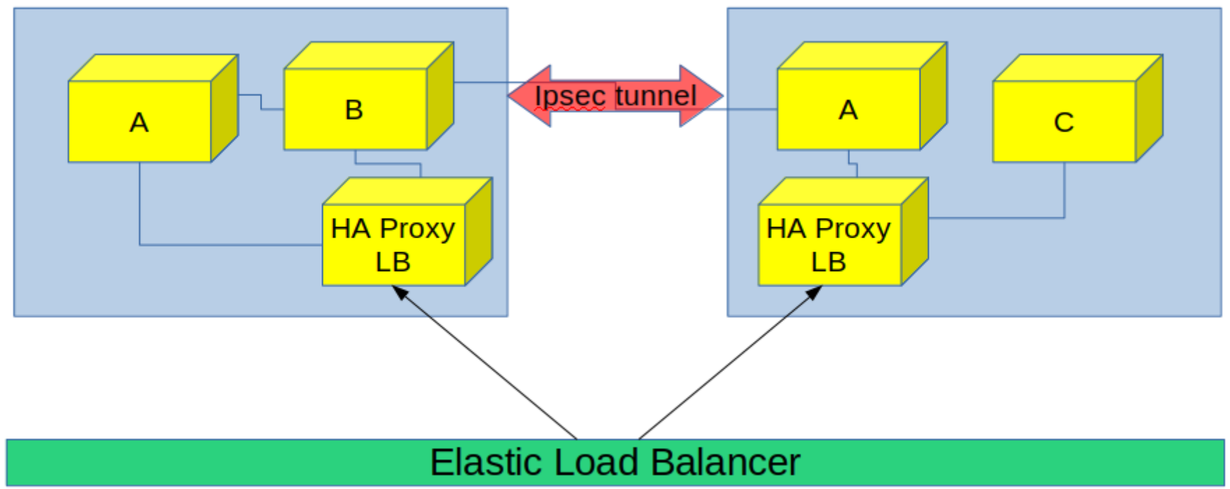
RANCHER

Rancher

Rancher Labs develops open source software that makes it easy to deploy and manage Docker containers and Kubernetes in production on any infrastructure.

— Rancher Website

Rancher Hosts



HashiCorp

Terraform

Terraform

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently.

— Terraform website

Key Features

- Infrastructure as Code
- Execution Plans
- Resource Graph
- Change Automation

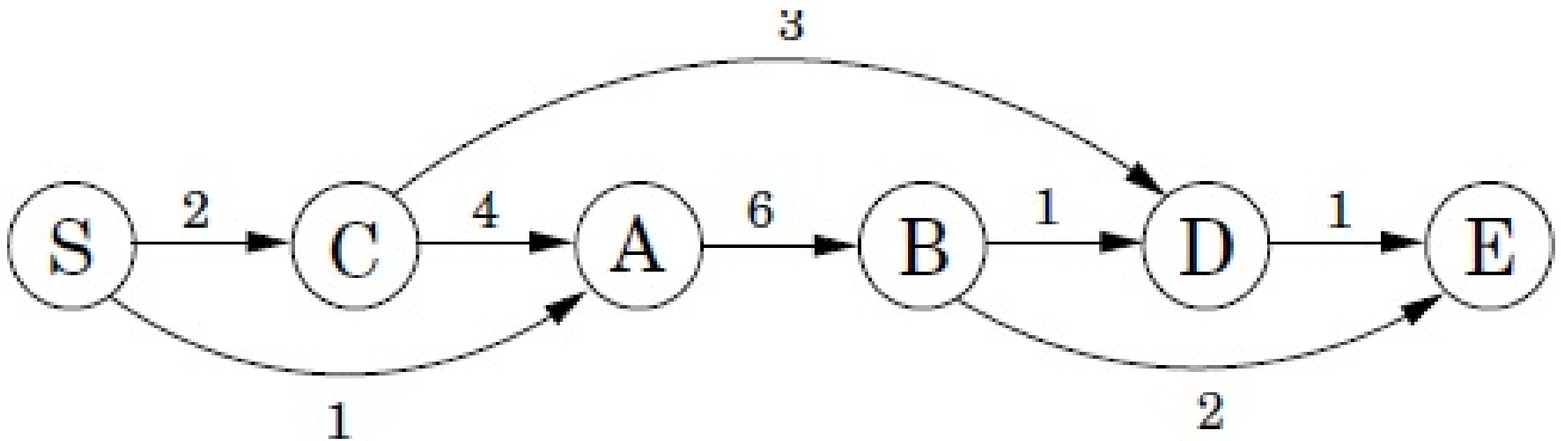
We get

- Repeatable
- Versioned
- Documented
- Automated
- Testable
- Shareable

Handling of infrastructure

How it Works

Terraform creates a D-A-G of tasks



Configuration

The set of files used to describe infrastructure in Terraform is simply known as a Terraform *configuration*.

Configuration

HashiCorp Configuration Language (HCL)

- Less verbose than JSON
- More concise than YAML
- Restricted subset (compared to programming language)
- Any tool can also accept JSON
- Allows comments

Key components

- Providers
- Resourses
- Provisioners

Providers

Account details for fx AWS

Resourses

Logical representation of "Physical" resource (physical item, even though it is a virtual server)

Defines the desired state of a resource

Provisioners

Post-creation "initialization" of resource

Has access to the current properties of a resource

Terraform files

All .tf /.tf.json files in working directory are loaded and appended (in alphabetical order)

Duplicate resources not allowed

Modules

Modules are reusable components that can take be configured with inputs and deliver output for use in other scripts

Sample Infrastructure

Structure

```
.
├── modules
│   └── host-workers
│       ├── workers.tf
│       ├── inputs.tf
│       └── outputs.tf
└── terraform
    ├── aws.tf
    ├── cluster.tf
    ├── dns.tf
    ├── terraform.tf
    ├── variables.tf
    └── vpc.tf
```

Structure comments

- **modules:** Shared code, here describing the configuration of a rancher host machine
- **terraform:** Terraform files for the cluster we are trying to create

aws.tf

```
provider "aws" {
  region  = "eu-west-1"
  profile = "gr8conf"
  // access_key = "${var.aws_access_key}"
  // secret_key = "${var.aws_secret_key}"
}
```

variables.tf (1)

```
variable "name" {
  description = "The name given to the cluster environment"
  default     = "gr8conf"
}

variable "vpc_cidr" {
  description = "The network CIDR."
  default     = "172.16.0.0/16"
}
```

variables.tf (2)

```
variable "cidrs_public_subnets" {
  description = "The CIDR ranges for public subnets."
  default     = ["172.16.0.0/24", "172.16.1.0/24", "172.16.2.0/24"]
}

variable "cidrs_private_subnet" {
  description = "CIDRs for private subnets."
  default     = ["172.16.128.0/24", "172.16.129.0/24", "172.16.130.0/24"]
}
```

vpc.tf (1)

```
module "vpc" {
  source                = "github.com/terraform-community-modules/tf_aws_vpc "
  name                  = "${var.name}-vpc"
  cidr                  = "${var.vpc_cidr}"
  private_subnets       = "${var.cidrs_private_subnet}"
  public_subnets        = "${var.cidrs_public_subnets}"
  enable_dns_hostnames = true
  enable_dns_support    = true
  azs                   = ["eu-west-1a", "eu-west-1b", "eu-west-1c"]
  enable_nat_gateway    = "true"

  tags {
    "Terraform" = "true"
    "Environment" = "GR8Conf"
  }
}
```

vpc.tf (2)

```
resource "aws_security_group" "vpc_sg_within" {
  name_prefix = "${var.name}"
  vpc_id = "${module.vpc.vpc_id}"
  ingress {
    from_port = 0
    to_port = 0
    protocol = "-1"
    self = true   }
  egress {
    from_port = 0
    to_port = 0
    protocol = "-1"
    self = true  }
  lifecycle {
    create_before_destroy = true }
}
```

vpc.tf (3)

```
resource "aws_security_group" "web_sg" {
  name_prefix = "${var.name}"
  vpc_id = "${module.vpc.vpc_id}"
  ingress {
    from_port = 80
    to_port = 80
    protocol = "tcp"
    cidr_blocks = [
      "0.0.0.0/0" // Restrict IP range access here
    ]
  }
}
```

vpc.tf (4)

```
resource "aws_eip" "nat" {
  vpc = true
}

resource "aws_nat_gateway" "nat" {
  allocation_id = "${aws_eip.nat.id}"
  subnet_id = "${element(module.vpc.public_subnets,  0)}"
}
```

Host Worker module

inputs.tf (1)

```
variable "host_ami" {
  description = "Ami for the host"
  default = "ami-75cbcb13" // rancheros-v1.0.1-hvm-1
}

variable "host_instance_type" {
  description = "The instance type for the hosts"
  default = "t2.micro"
}

variable "vpc_id" {
  description = "The VPC to launch resources in"
}

variable "name" {
  description = "The cluster name"
}
```

inputs.tf (2)

```
variable "host_subnet_ids" {
  description = "The subnets to launch the hosts in"
}

variable "host_security_group_ids" {
  description = "Additional security groups to apply to hosts"
  default = ""
}

variable "host_root_volume_size" {
  description = "The size of the root EBS volume in GB"
  default = 24
}
variable "loadbalancer_ids" {
  description = "The loadbalancers to attach to the auto scaling group"
  default = ""
}
```

inputs.tf (3)

```
variable "rancher_image" {
  description = "The Docker image to run Rancher from"
  default = "rancher/agent:v1.2.2"
}

variable "rancher_server_url" {
  description = "The URL for the Rancher server (including the version, i.e. rancher.gr8conf.org/v1) "
}

variable "rancher_env_token" {
  description = "The Rancher environment token hosts will join rancher with"
}

variable "rancher_host_labels" {
  description = "Comma separate k=v labels to apply to all rancher hosts"
  default = ""
}
```

inputs.tf (4)

```
variable "min_host_capacity" {
  description = "The miminum capacity for the auto scaling group"
  default = 1
}

variable "max_host_capacity" {
  description = "The maximum capacity for the auto scaling group"
  default = 4
}

variable "desired_host_capacity" {
  description = "The desired capacity for the auto scaling group"
  default = 1
}
```

inputs.tf (5)

```
variable "host_health_check_type" {
  description = "Whether to use EC2 or ELB healthchecks in the ELB"
  default = "EC2"
}

variable "host_health_check_grace_period" {
  description = "The grace period for autoscaling group health checks"
  default = 300
}
```

inputs.tf (6)

```
variable "host_profile" {
  description = "The IAM profile to assign to the instances"
  default = ""
}

variable "host_key_name" {
  description = "The EC2 KeyPair to use for the machine"
}
```

outputs.tf

```
output "hosts_security_group" {
  value = "${aws_security_group.worker_sg.id}"
}
```

workers.tf (1)

```
resource "aws_launch_configuration" "worker" {
  name_prefix = "terraform_worker_"
  image_id = "${var.host_ami}"
  instance_type = "${var.host_instance_type}"
  iam_instance_profile = "${var.host_profile}"
  security_groups = [
    "${compact(concat(list(aws_security_group.worker_sg.id), split( ",", var.host_security_group_ids)))}"
  ]
  associate_public_ip_address = false
  ebs_optimized = false // To enable tiny instances
  root_block_device {
    volume_type = "gp2"
    volume_size = "${var.host_root_volume_size}"
    delete_on_termination = true
  }
```

More on next slide

workers.tf (2)

```
  user_data = <<EOF
#cloud-config
rancher:
  services:
    rancher-agent1:
      image: ${var.rancher_image}
      environment:
        - CATTLE_AGENT_IP=$private_ipv4
        - CATTLE_HOST_LABELS=${join("&", split(",", var.rancher_host_labels))}
      command: ${var.rancher_server_url}/scripts/${var.rancher_env_token}
      volumes:
        - /var/run/docker.sock:/var/run/docker.sock
      privileged: true
EOF
  lifecycle {
    create_before_destroy = true
  }
}
```

Continued from previous slide

workers.tf (3)

```
resource "aws_autoscaling_group" "rancher" {
  max_size = "${var.max_host_capacity}"
  min_size = "${var.min_host_capacity}"
  desired_capacity = "${var.desired_host_capacity}"
  launch_configuration = "${aws_launch_configuration.worker.id}"
  health_check_type = "${var.host_health_check_type}"
  health_check_grace_period = "${var.host_health_check_grace_period}"
  load_balancers = [ "${compact(split(",", var.loadbalancer_ids))}" ]
  vpc_zone_identifier = [
    "${split(",", var.host_subnet_ids)}"
  ]
  tag {
    key = "Name"
    value = "${var.name}-host"
    propagate_at_launch = true
  }
}
```

workers.tf (4)

```
resource "aws_security_group" "worker_sg" {
  description = "Allow traffic to worker instances"
  vpc_id = "${var.vpc_id}"
}
```

workers.tf (5)

```
resource "aws_security_group_rule" "rancher_upd_500_ingress" {
  type = "ingress"
  from_port = 500
  to_port = 500
  protocol = "udp"
  security_group_id = "${aws_security_group.worker_sg.id}"
  self = true
}

resource "aws_security_group_rule" "rancher_upd_4500_ingress" {
  type = "ingress"
  from_port = 4500
  to_port = 4500
  protocol = "udp"
  security_group_id = "${aws_security_group.worker_sg.id}"
  self = true
}
```

workers.tf (6)

```
resource "aws_security_group_rule" "rancher_upd_500_egress" {
  type = "egress"
  from_port = 500
  to_port = 500
  protocol = "udp"
  security_group_id = "${aws_security_group.worker_sg.id}"
  self = true
}

resource "aws_security_group_rule" "rancher_upd_4500_egress" {
  type = "egress"
  from_port = 4500
  to_port = 4500
  protocol = "udp"
  security_group_id = "${aws_security_group.worker_sg.id}"
  self = true
}
```

workers.tf (7)

```
resource "aws_security_group_rule" "rancher_egress" {
  type = "egress"
  from_port = 0
  to_port = 0
  protocol = "-1"
  security_group_id = "${aws_security_group.worker_sg.id}"
  cidr_blocks = [
    "0.0.0.0/0"
  ]
}
```

Cluster

cluster.tf (1)

```
module "hosts" {
  source = "../modules/host-workers"
  name = "${var.name}-cluster"
  desired_host_capacity="2"
  host_key_name = "recovery"
  vpc_id = "${module.vpc.vpc_id}"
  host_subnet_ids = "${join(",", module.vpc.private_subnets)}"
  rancher_server_url = "https://rancher.grydeske.com/v1"
  rancher_env_token = "D93C1B9F627E1B7168AE:1483142400000:7lZFwjs9lDSQskK9fbXCPwiPL2g "
  rancher_host_labels = "region=eu-west-1,type.app=true,type.network=true "
  loadbalancer_ids = "${aws_elb.cluster-elb-public.id}"
  host_security_group_ids = "${aws_security_group.vpc_sg_within.id}"
  host_ami = "ami-75cbcb13"
}
```

cluster.tf (2)

```
resource "aws_elb" "cluster-elb-public" {
  subnets = ["${module.vpc.public_subnets}"]
  security_groups = [ "${aws_security_group.web_sg.id}", "${aws_security_group.vpc_sg_within.id}" ]
  listener {
    lb_port = 80
    lb_protocol = "HTTP"
    instance_port = 80
    instance_protocol = "HTTP"  }
  health_check {
    healthy_threshold = 2
    unhealthy_threshold = 2
    timeout = 5
    target = "TCP:80"
    interval = 10 }
  cross_zone_load_balancing = true
  tags { Cluster = "${var.name}"  }
}
```

DNS

dns.tf (1)

```
resource "aws_route53_zone" "gr8conf_domain" {
  lifecycle {
    prevent_destroy = true
  }
  name = "grydeske.org"
}

output "domain_ns_servers" {
  // There are 4 in all
  value = "\n${aws_route53_zone.gr8conf_domain.name_servers .0}\n${aws_route53_zone.gr8conf_domain.name_servers .1}\n${aws_route
}
```

cluster.tf (3)

```
resource "aws_route53_record" "dns-wildcard" {
  name = "*"
  zone_id = "${aws_route53_zone.gr8conf_domain.id}"
  type = "A"
  alias {
    name = "${aws_elb.cluster-elb-public.dns_name}"
    zone_id = "${aws_elb.cluster-elb-public.zone_id}"
    evaluate_target_health = true
  }
}
```

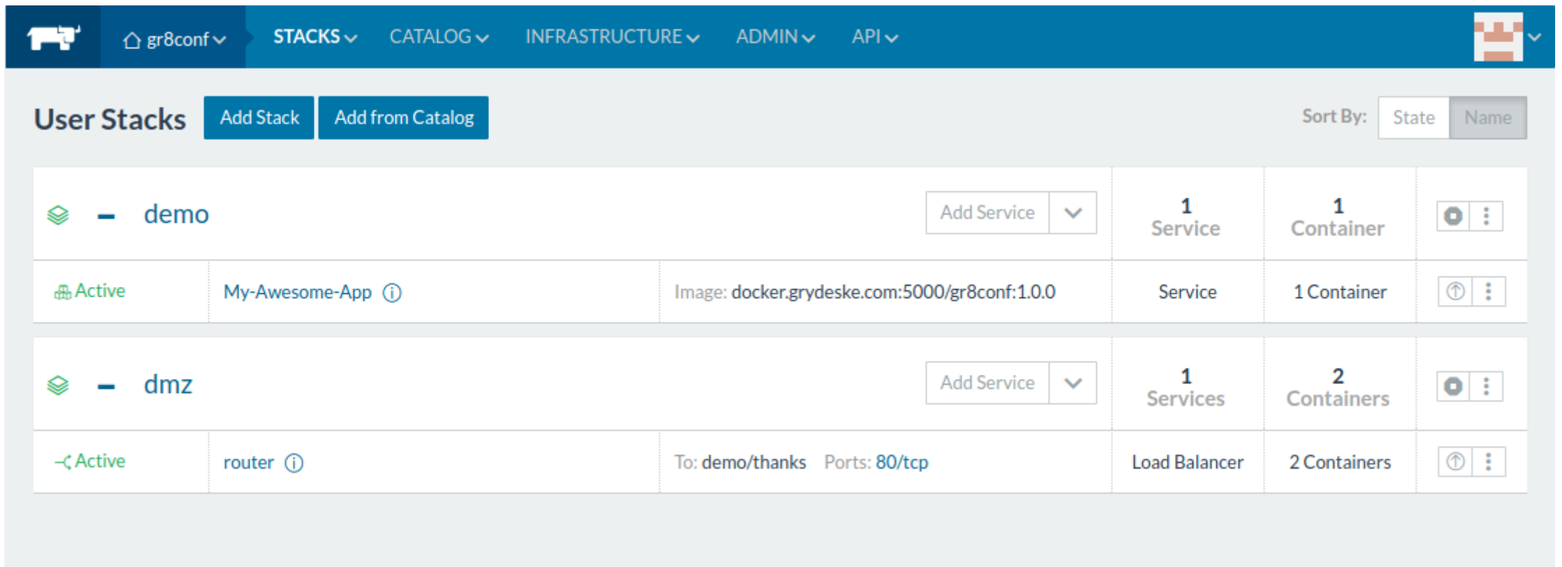
Lets spin up some infrastructure!

Deploying on Rancher

Rancher Features

- Environments
- Stacks
- Services

Rancher Features



Literature

- https://www.terraform.io
- http://www.oreilly.com/pub/e/3615
- http://rancher.com

Questions