

Server to Cloud Migrations

How we run server products on stateless cloud services



@CRAIGKNOTT92



Server to Cloud Migrations

How we run server products on stateless cloud services



@CRAIGKNOTT92

OUR MISSION

We believe behind every great human achievement, there is a **team**.
Our mission is to unleash the potential of every team.





More than 100 teams

~600 core services

Hundreds of changes/day

Multiple products

Many, many, many customers



Customer Support Technology



Customer Support Technology (SRE)



What is Jira Datacenter

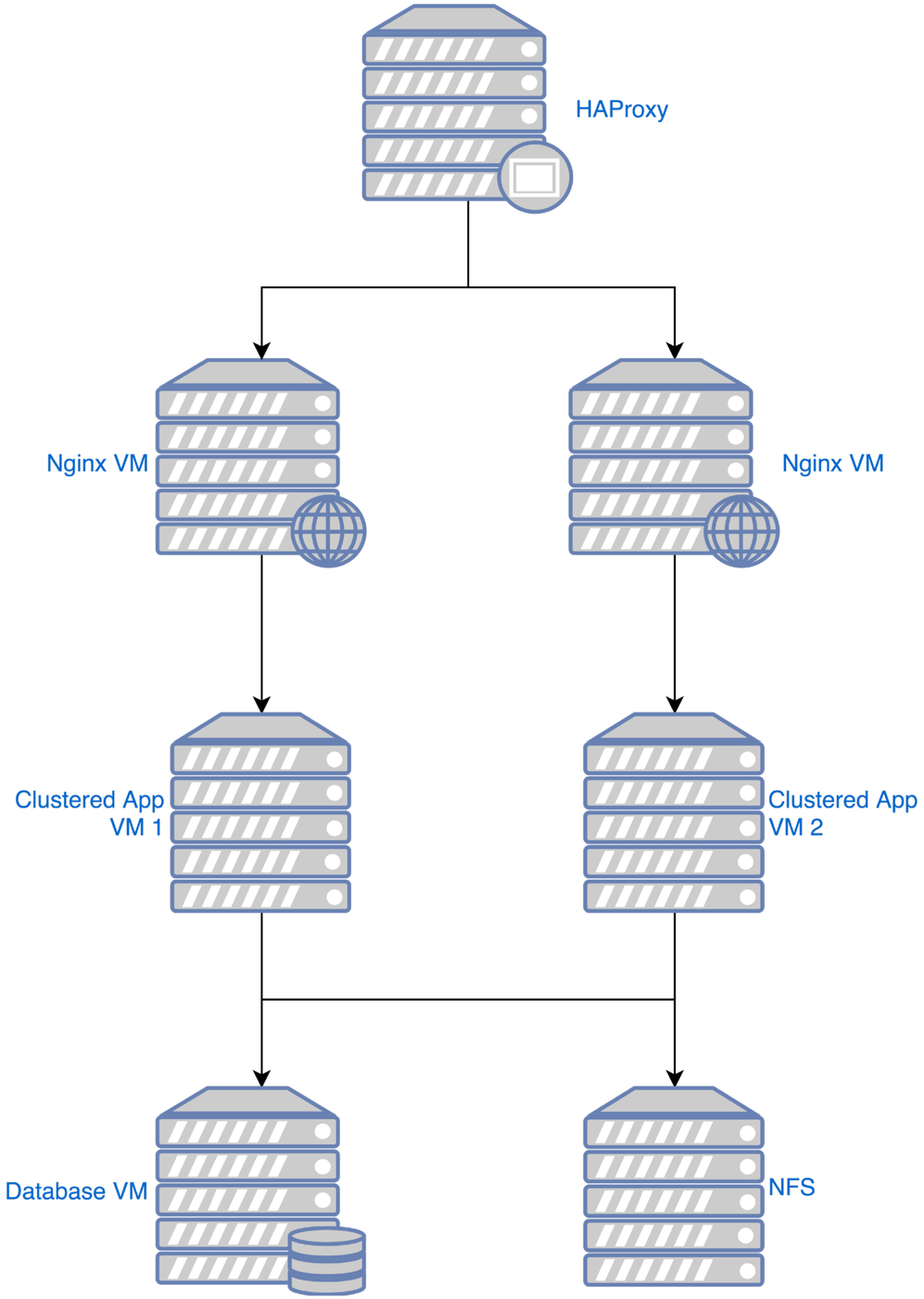
Once upon a time...



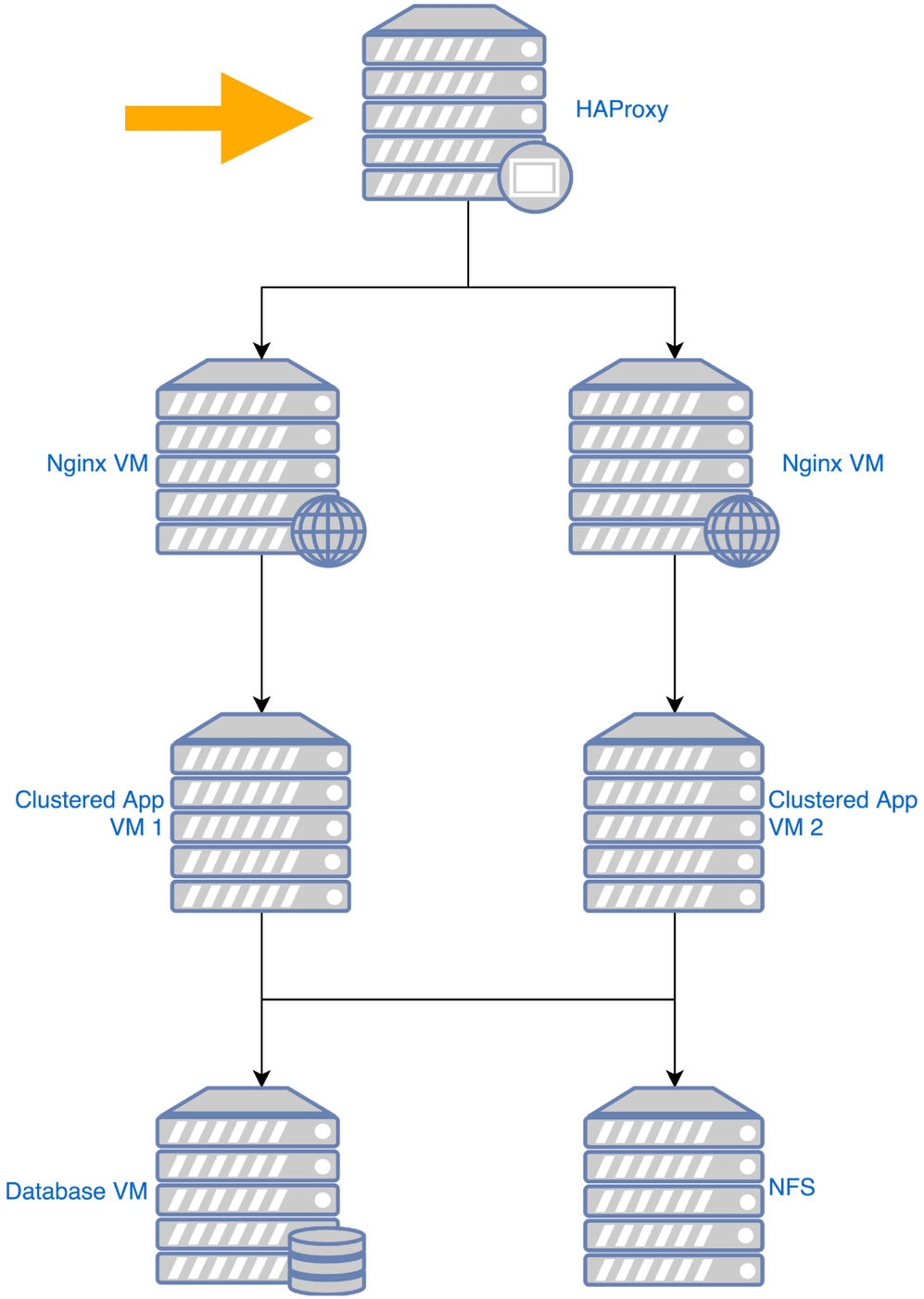
“

Sorry, we can’t support this anymore. You’ll have to run it yourself… Somewhere else

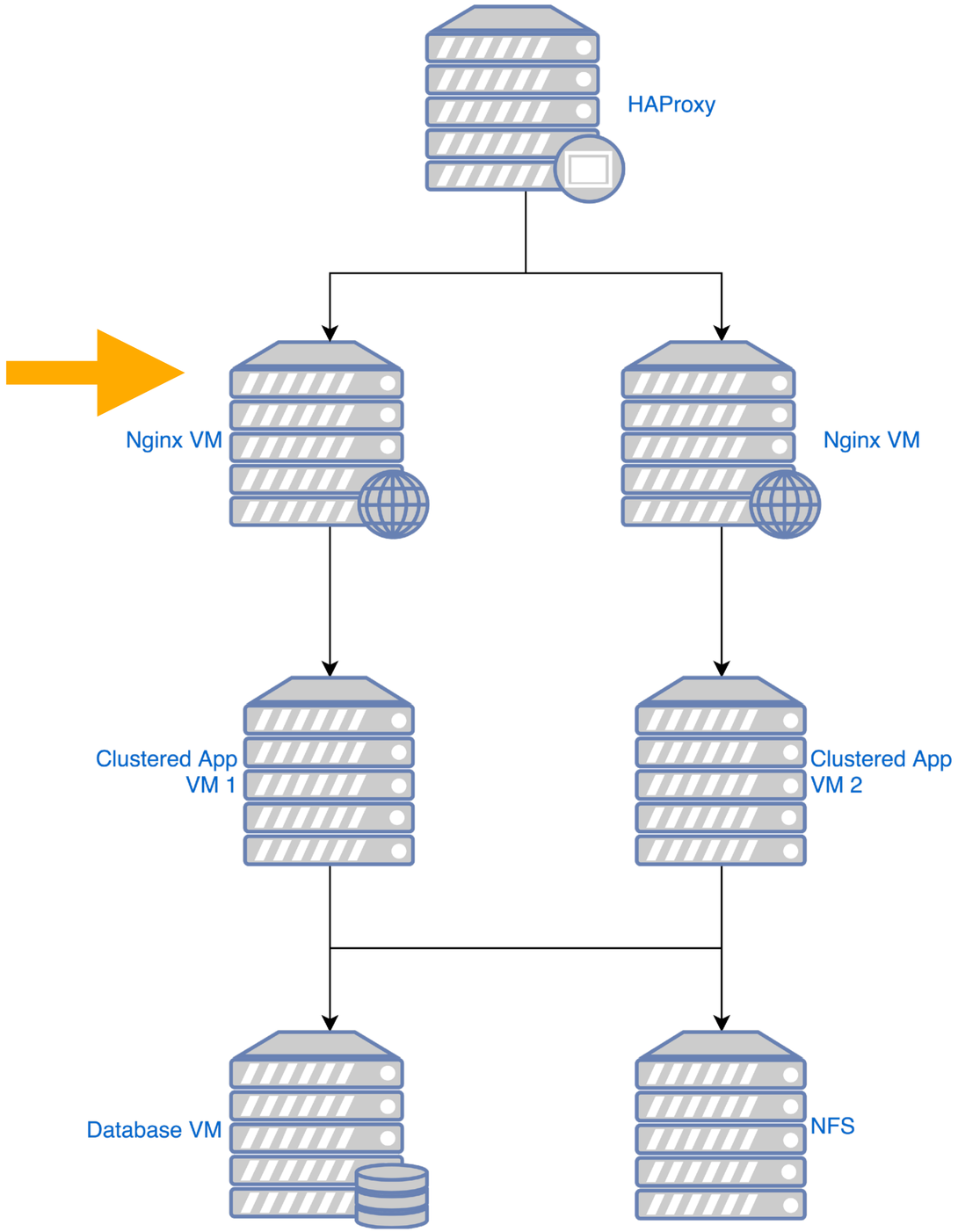
VM PLATFORM MANAGER



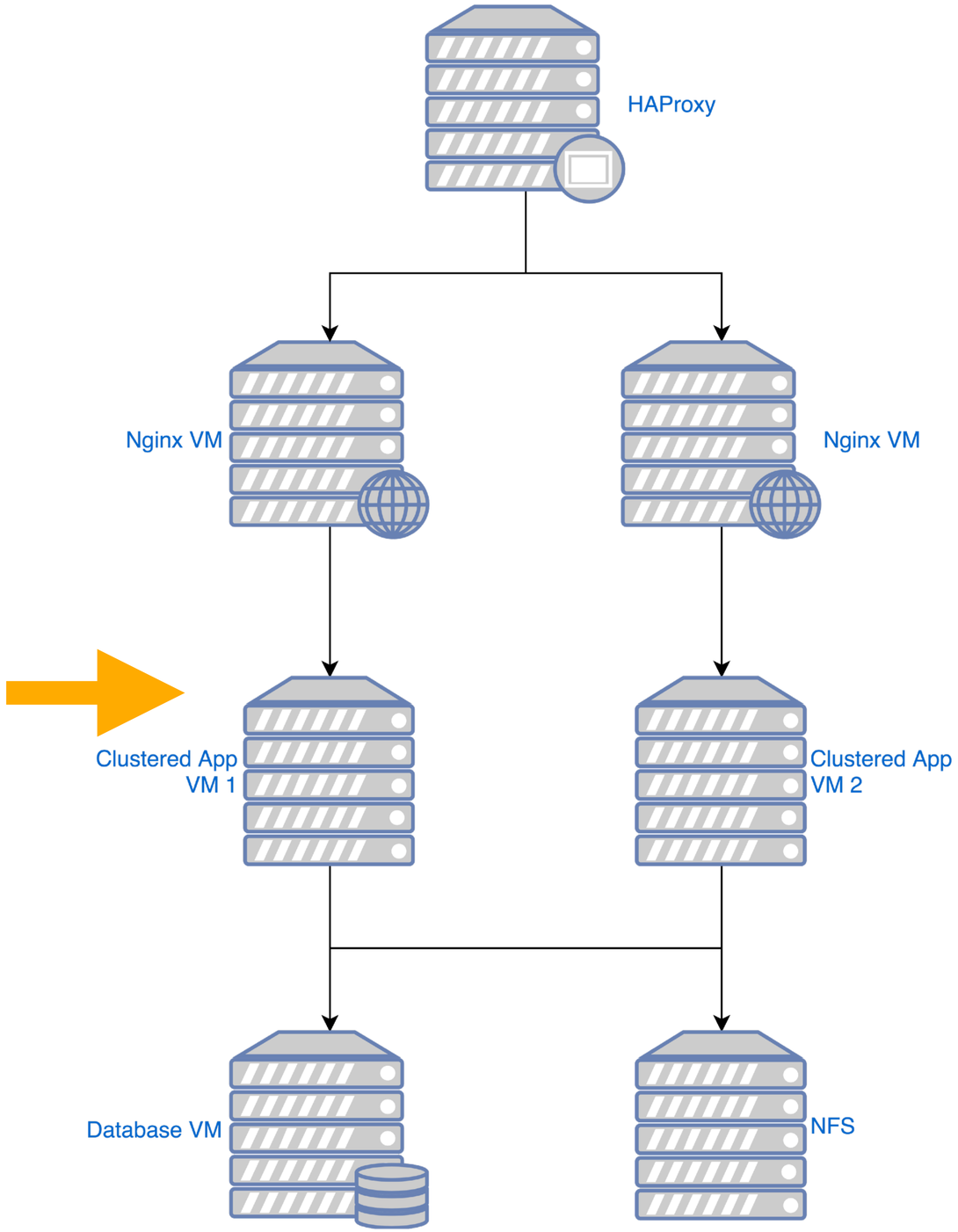
Architecture



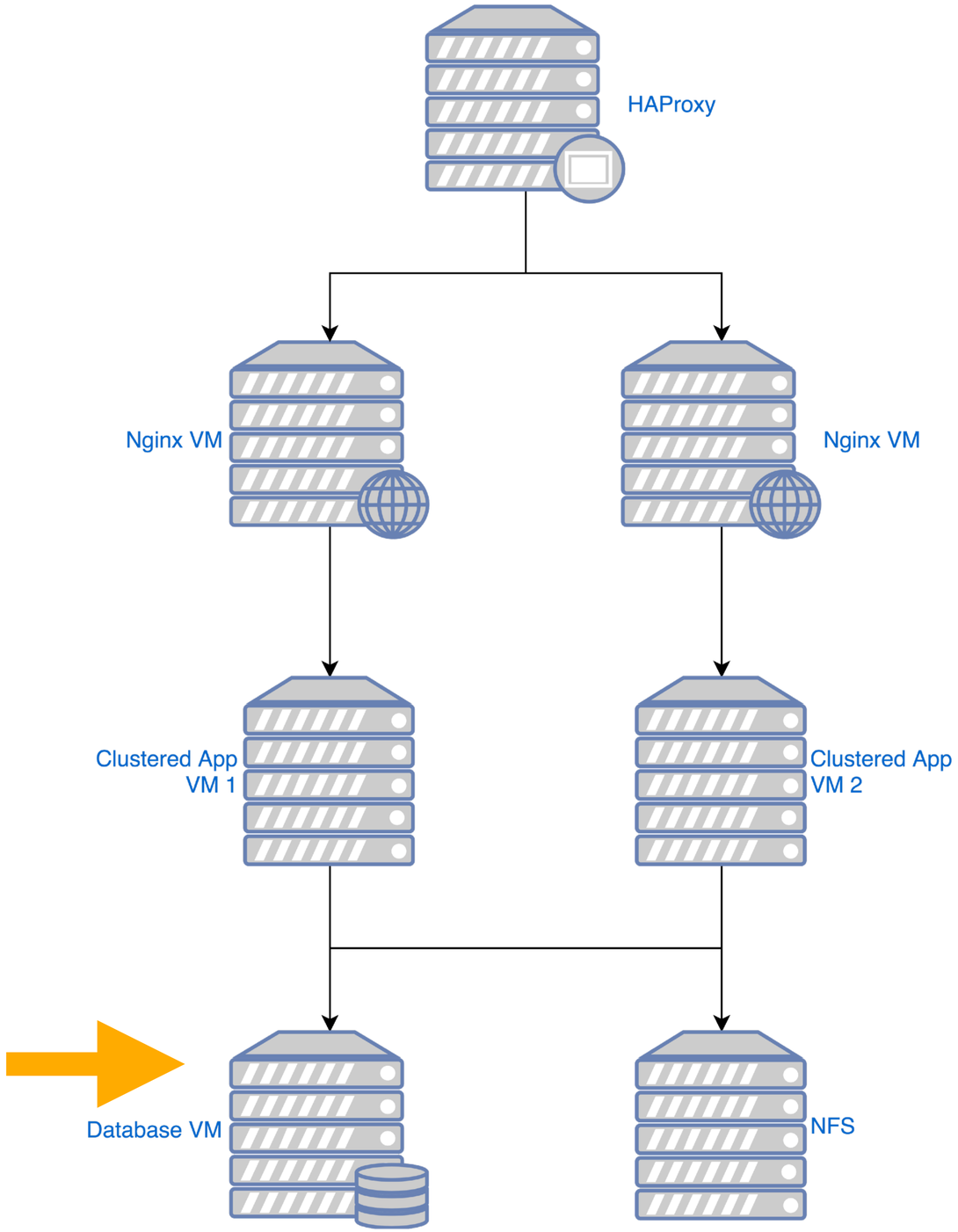
Architecture



Architecture



Architecture



Architecture

WHAT'S THE PROBLEM?

WHAT'S THE PROBLEM?
INDIVIDUALLY MANAGED

WHAT'S THE PROBLEM? INDIVIDUALLY MANAGED CLUSTER INCONSISTENCIES

WHAT'S THE PROBLEM?
INDIVIDUALLY MANAGED CLUSTER INCONSISTENCIES UNSTABLE

WHAT'S THE PROBLEM?
INDIVIDUALLY MANAGED CLUSTER INCONSISTENCIES UNSTABLE

WHAT ARE WE GOING TO DO?

Migrate to Cloud Services!

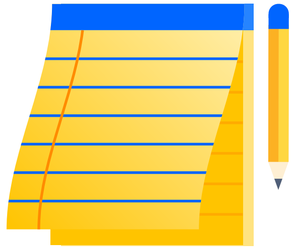


Core Requirements



Easy to use

Maintain and improve upon



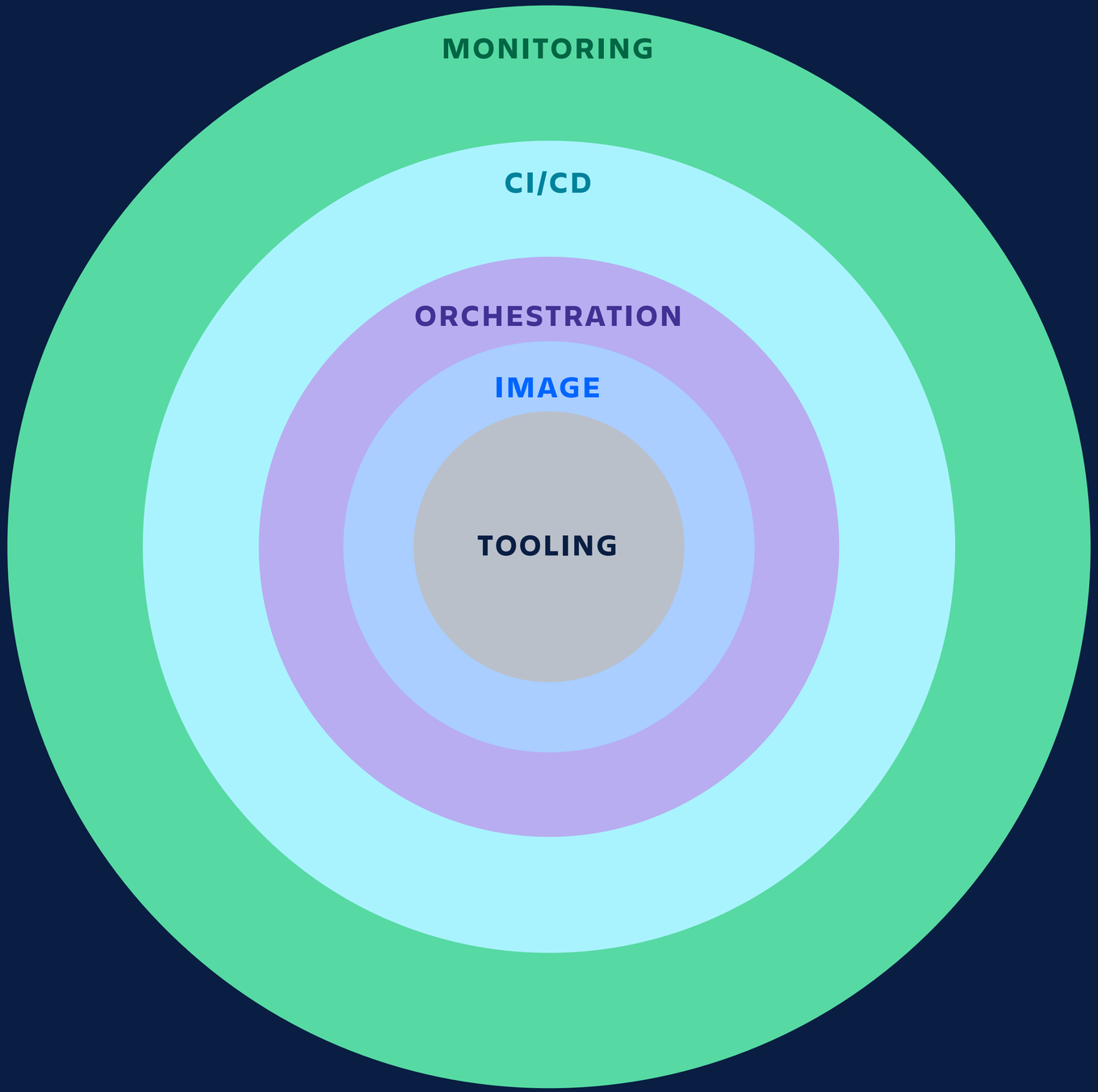
Auditing

All changes are peer reviewed or have some audit trail



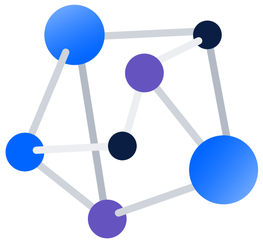
Image Management

Instead of server management


MONITORING
CI/CD
ORCHESTRATION
IMAGE
TOOLING

TOOLING

What tools do we need



Orchestration



Image Creation



Configuration



Orchestration

The automated arrangement, coordination, and management of computer systems, middleware, and services

CLOUDFORMATION

- Support for all AWS resources on release
- State Management

TERRAFORM

- Support for multiple providers
- Open Source
- Modularization
- Change Verification

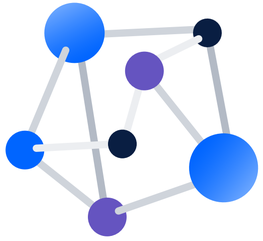
CLOUDFORMATION

- Support for all AWS resources on release
- State Management

TERRAFORM

- Support for multiple providers
- Open Source
- Modularization
- Change Verification

What tools do we need



Terraform



Image Creation



Configuration

Packer!

Machine Images

The perfect tool for creating machine images in a repeatable, consistent and peer reviewed way

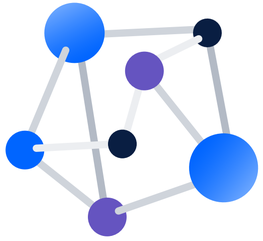
Multiple Providers

Just like terraform supports multiple providers, Packer can create images for multiple providers.

Open Source

Missing a feature? Work with the open source community to implement it.

What tools do we need



Terraform

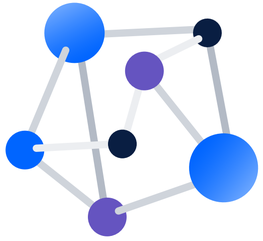


Packer



Configuration

What tools do we need



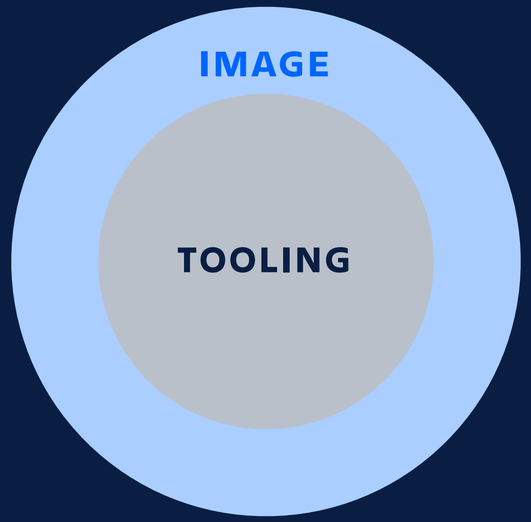
Terraform



Packer



Ansible


IMAGE
TOOLING

AMI Creation With Packer

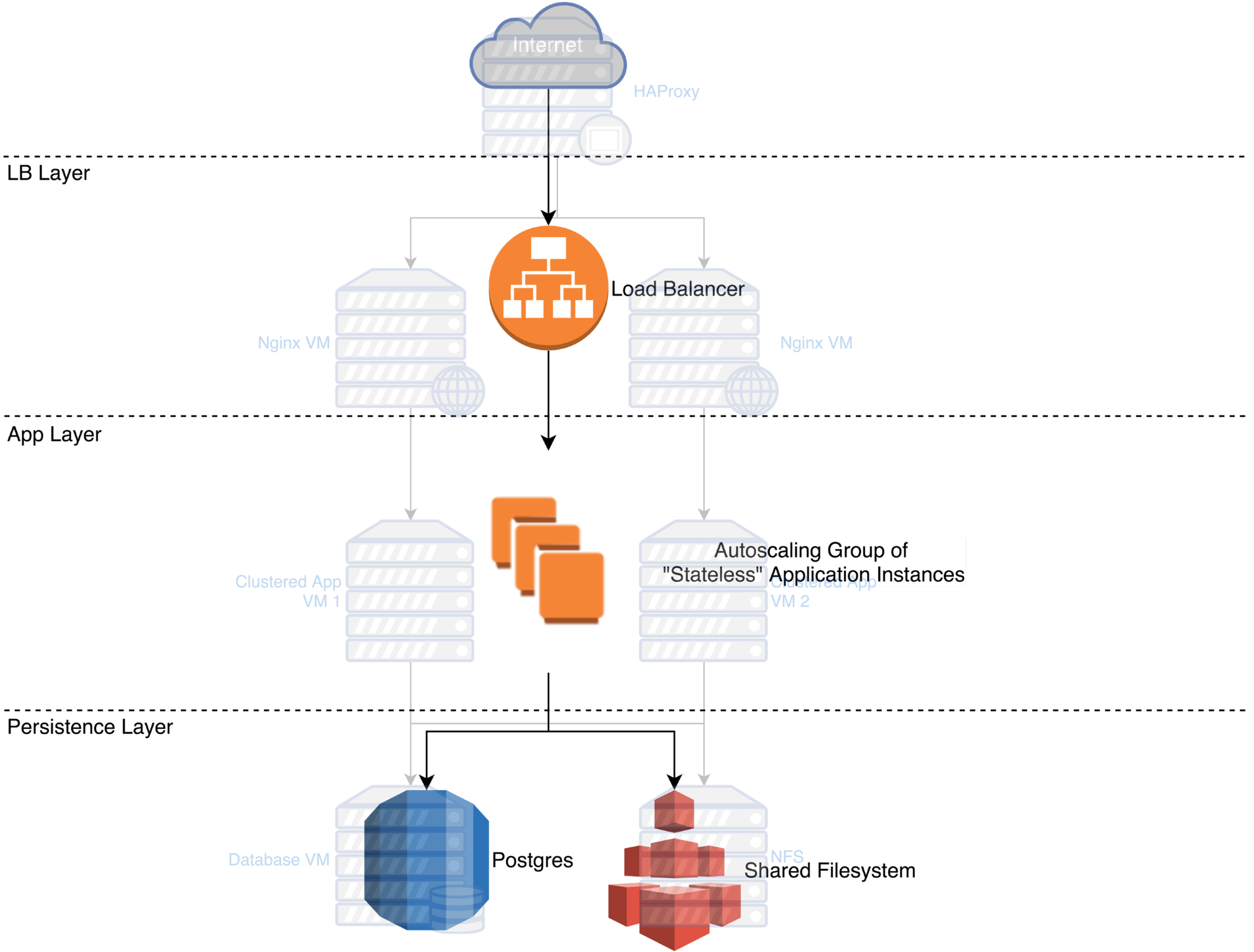
packer.io

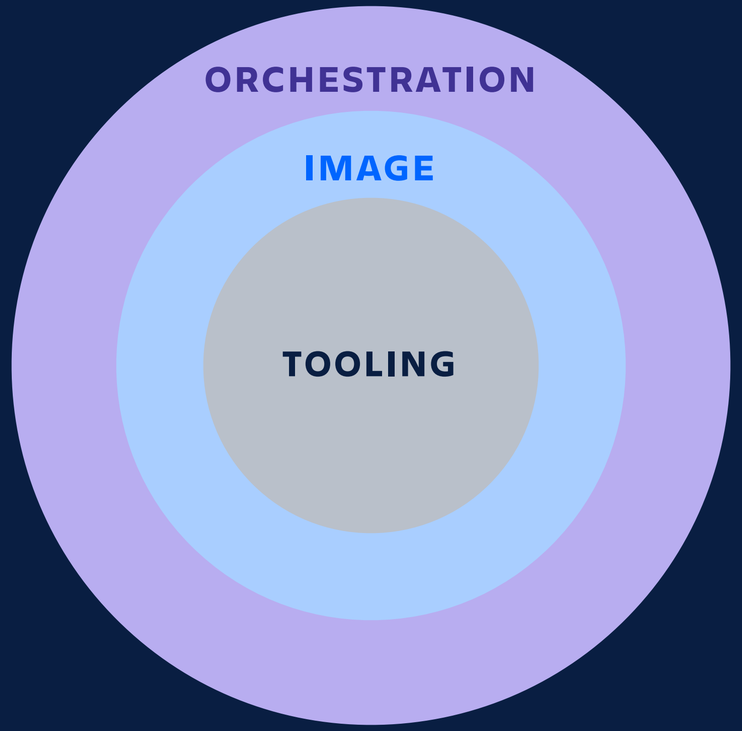
```
{
 “description”: “Our Application”,
 “variables”: {},
 “builders”: [{
   “type”: “amazon-ebs”,
   “access_key”: “{{user `aws_access_key`}}”
   “secret_key”: “{{user `aws_secret_key`}}”
   “region”: “{{user `region`}}”
    …
 }]
}
```

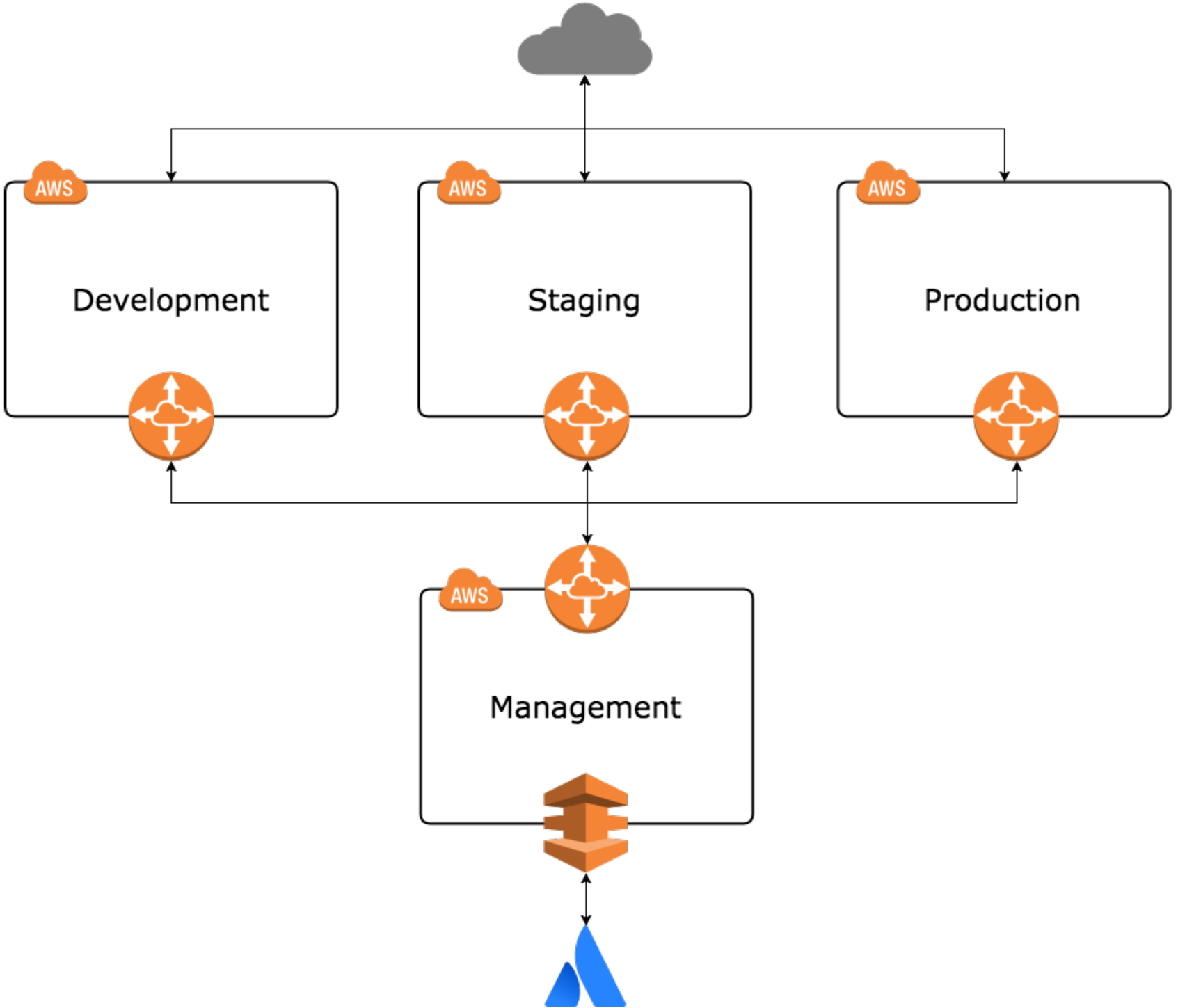
AMI Creation With Packer

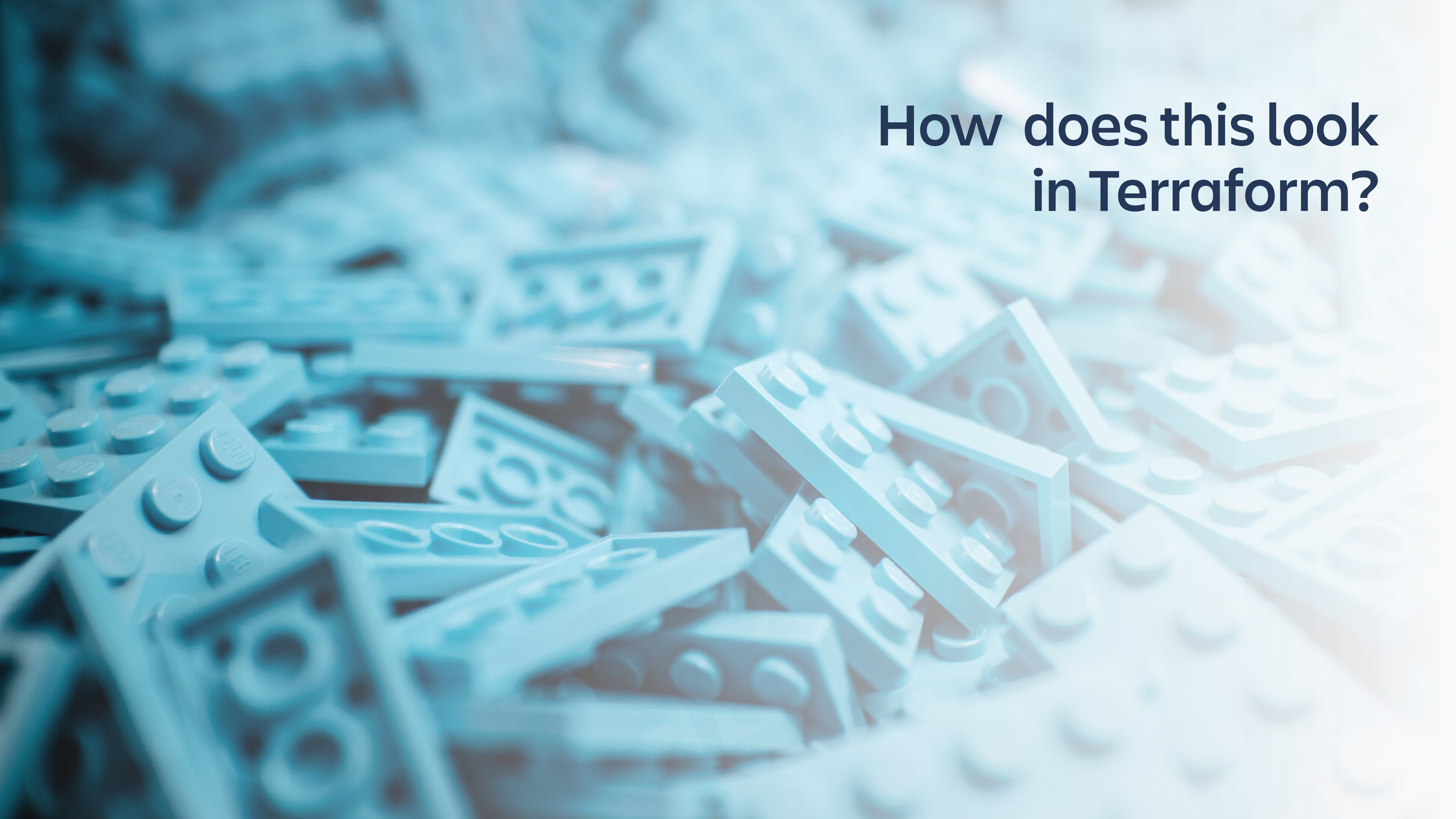
packer.io

```
{
“provisioners”: [{
    “type”: “ansible”,
    “playbook_file”: “../ansible/base.yml”,
      …
 }],
}
```


Internet
HAProxy
LB Layer
Load Balancer
Nginx VM
Nginx VM
App Layer
Autoscaling Group of
"Stateless" Application Instances
Clustered App
VM 1
Clustered App
VM 2
Persistence Layer
Database VM
Postgres
NFS
Shared Filesystem


ORCHESTRATION
IMAGE
TOOLING


AWS
Development
AWS
Staging
AWS
Production
AWS
Management



How does this look in Terraform?

Terraform Structure

Overview

Module Layout

Environments

```
environments
- development
- staging
- production
- management
modules
- app_tier
- db_tier
- vpc
- vpc_peering
```

Terraform Structure

Overview

Module Layout

Environments

```
app_tier
- load_balancing.tf
- autoscaling.tf
- iam.tf
- security_groups.tf
- outputs.tf
- variables.tf
vpc
- main.tf
- outputs.tf
- variables.tf
```

Terraform Structure

Overview

Module Layout

Environments

```
app_tier
- load_balancing.tf
- autoscaling.tf
- iam.tf
- security_groups.tf
- outputs.tf
- variables.tf
vpc
- main.tf
- outputs.tf
- variables.tf
```

```
output “vpc_id” {
  value = “${module.vpc.id}”
}
```

```
variable “vpc_cidr” {
  description = “VPC CIDR”
}
```

Terraform Structure

Overview

Module Layout

Environments

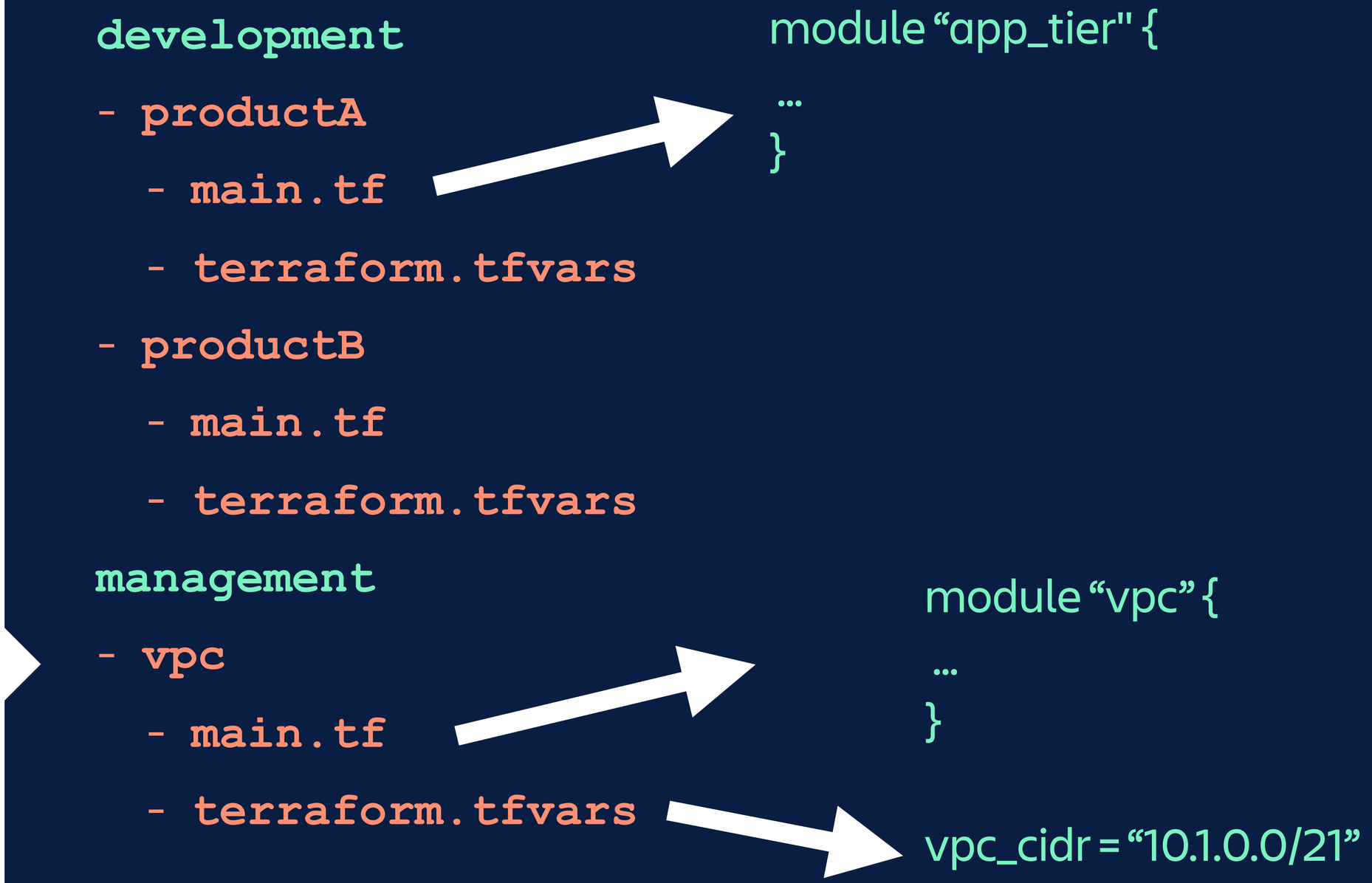
```
development
- productA
  - main.tf
  - terraform.tfvars
- productB
  - main.tf
  - terraform.tfvars
management
- vpc
  - main.tf
  - terraform.tfvars
```

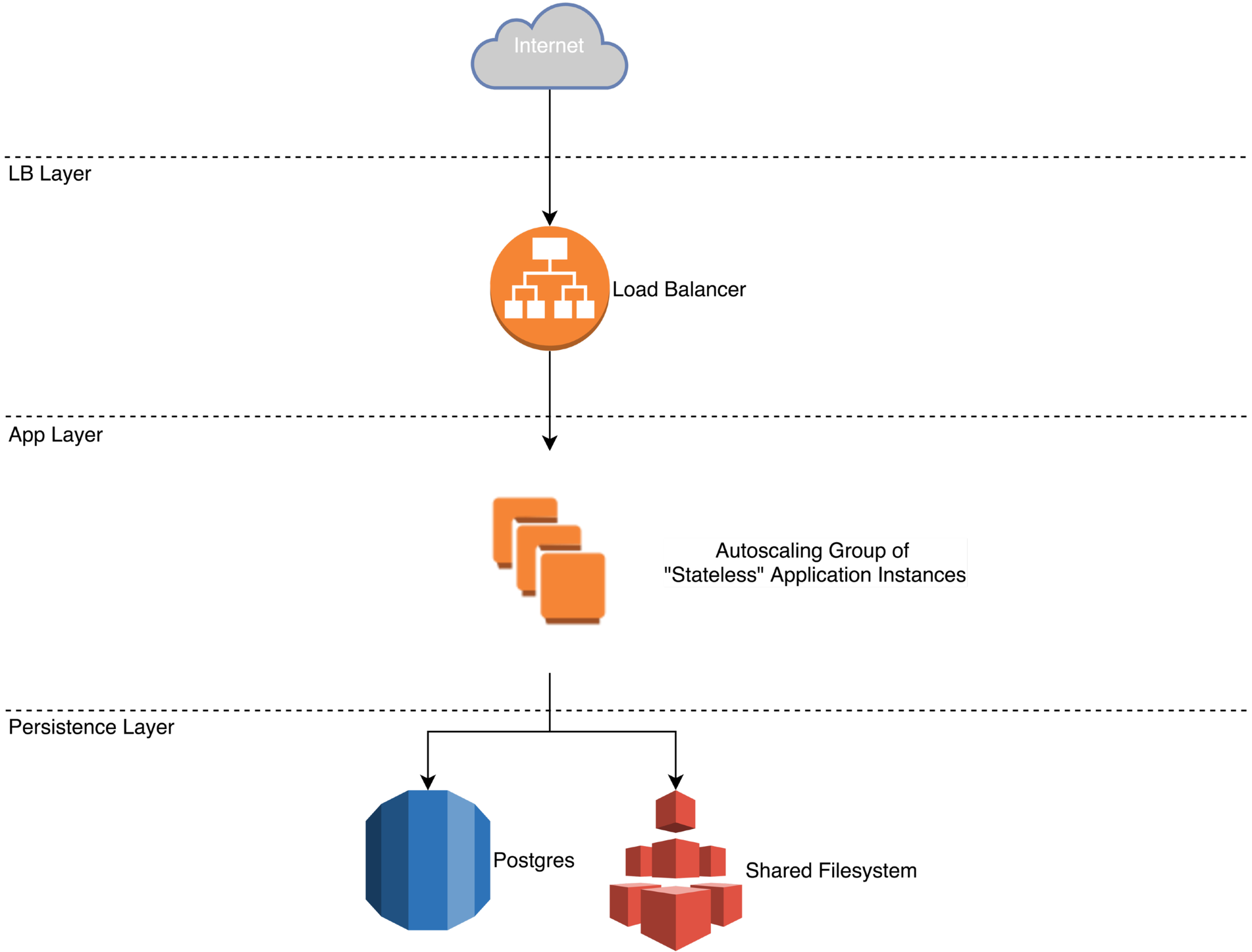
Terraform Structure

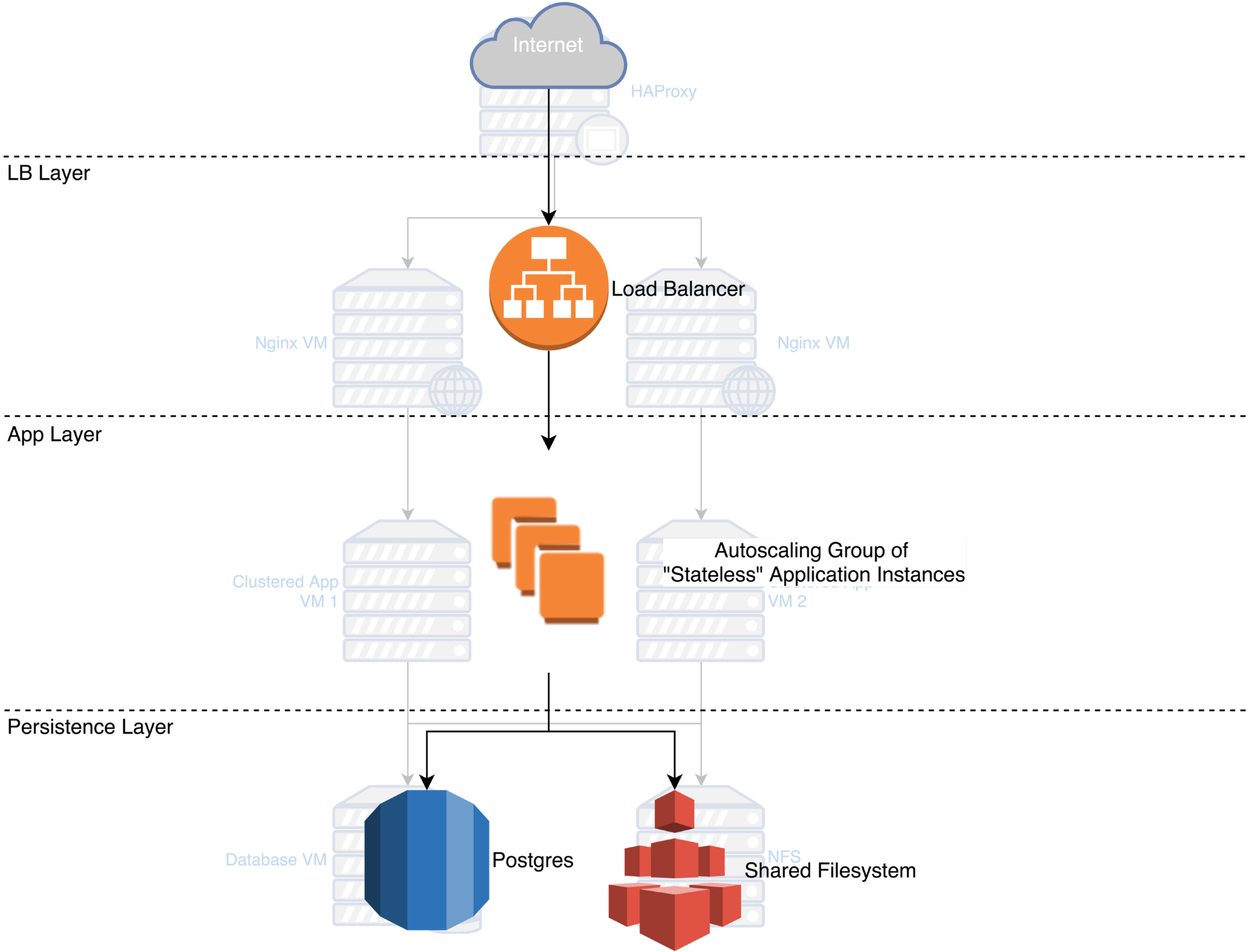
Overview

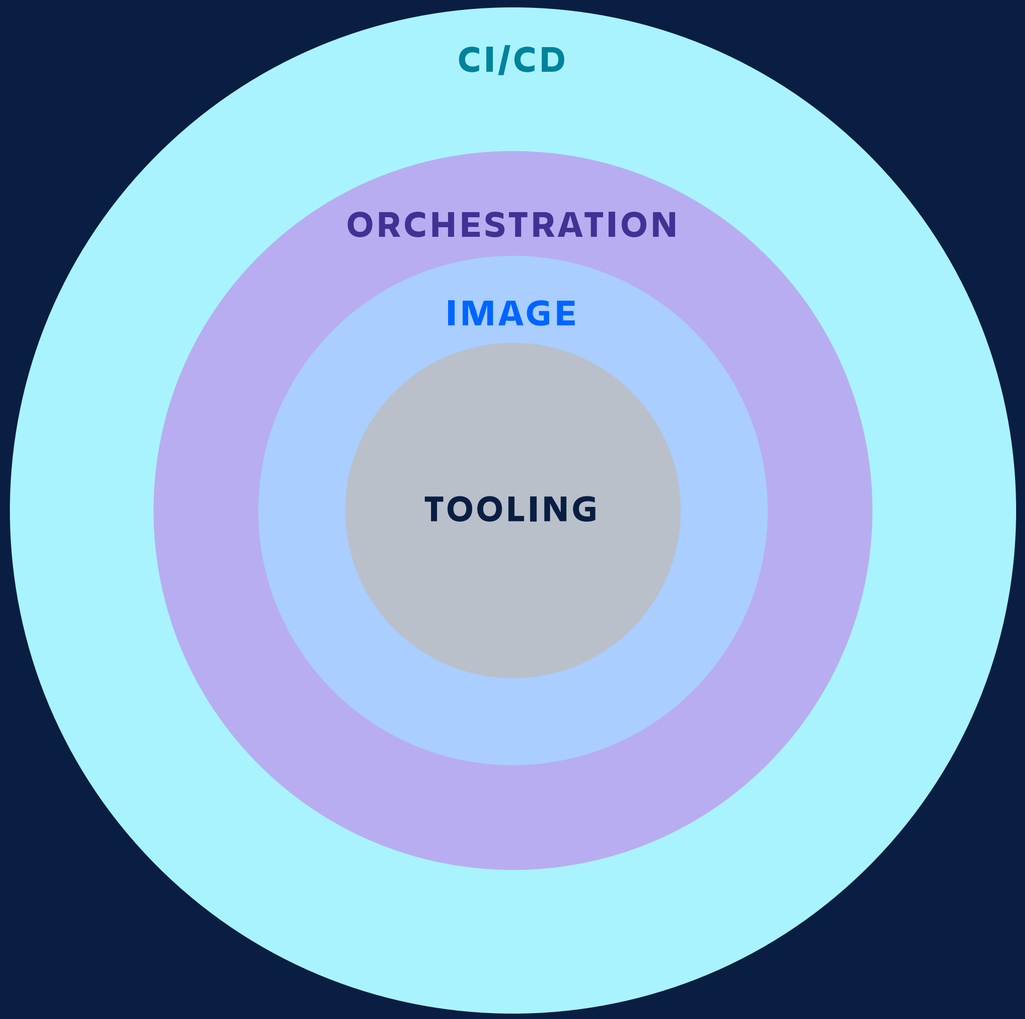
Module Layout

Environments



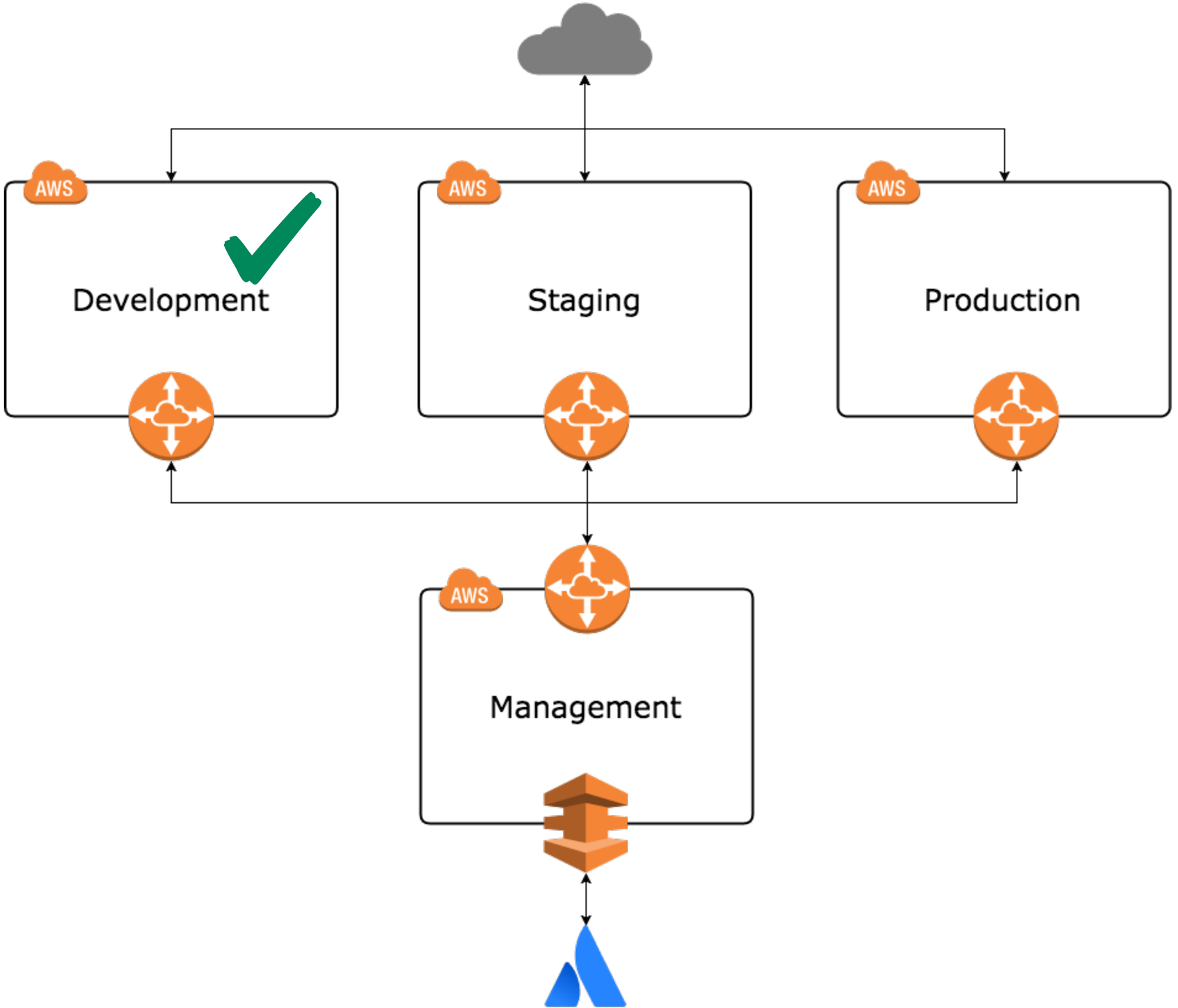

Internet
LB Layer
Load Balancer
App Layer
Autoscaling Group of
"Stateless" Application Instances
Persistence Layer
Postgres
Shared Filesystem

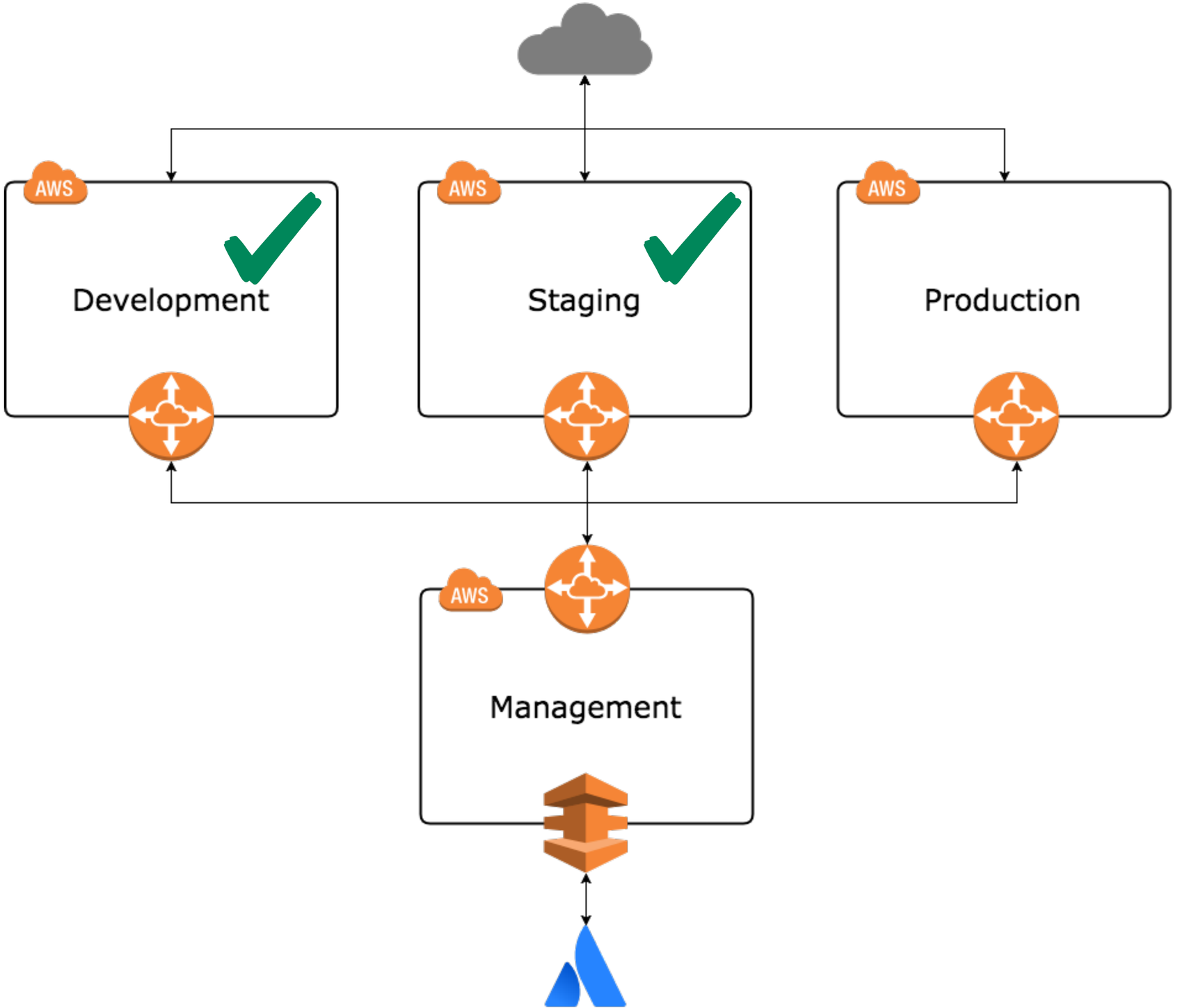

Internet
HAProxy
LB Layer
Load Balancer
Nginx VM
Nginx VM
App Layer
Clustered App
VM 1
Autoscaling Group of
"Stateless" Application Instances
VM 2
Persistence Layer
Database VM
Postgres
NFS
Shared Filesystem

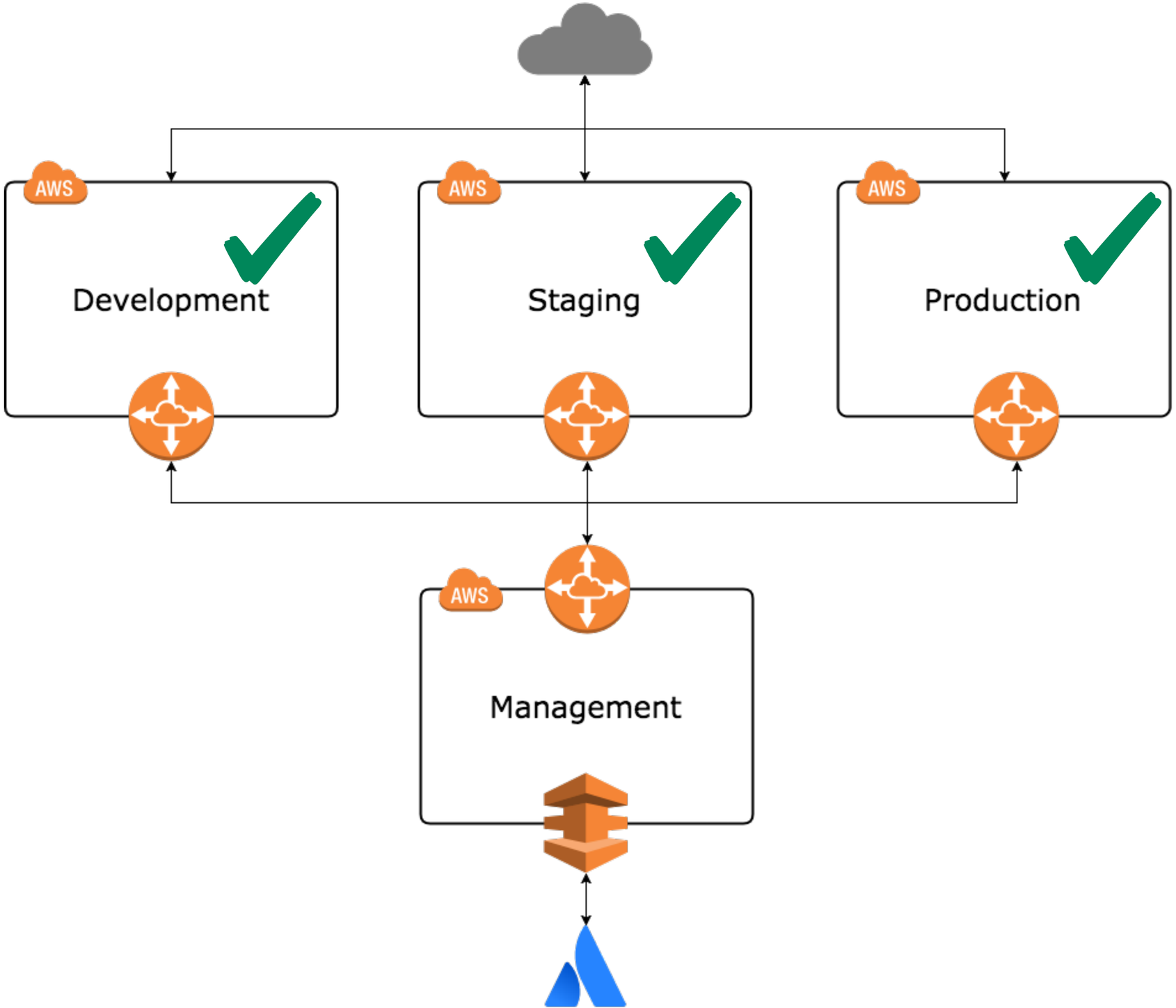

CI/CD
ORCHESTRATION
IMAGE
TOOLING

TERRAFORM CI/CD WITH BITBUCKET PIPELINES



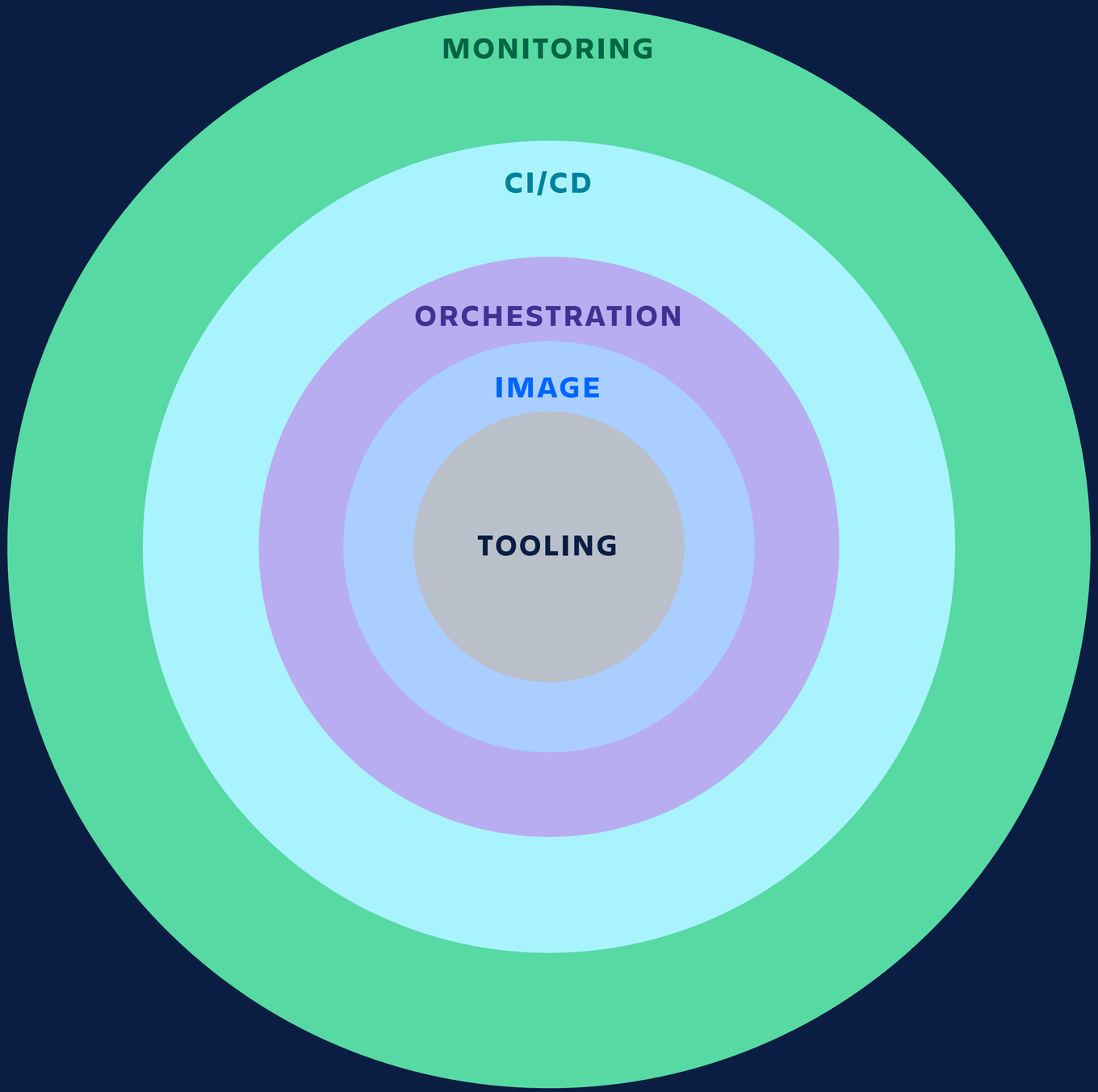

AWS
Development
AWS
Staging
AWS
Production
AWS
Management


AWS
Development
AWS
Staging
AWS
Production
AWS
Management


AWS
Development
AWS
Staging
AWS
Production
AWS
Management

PACKER CI/CD WITH BITBUCKET PIPELINES

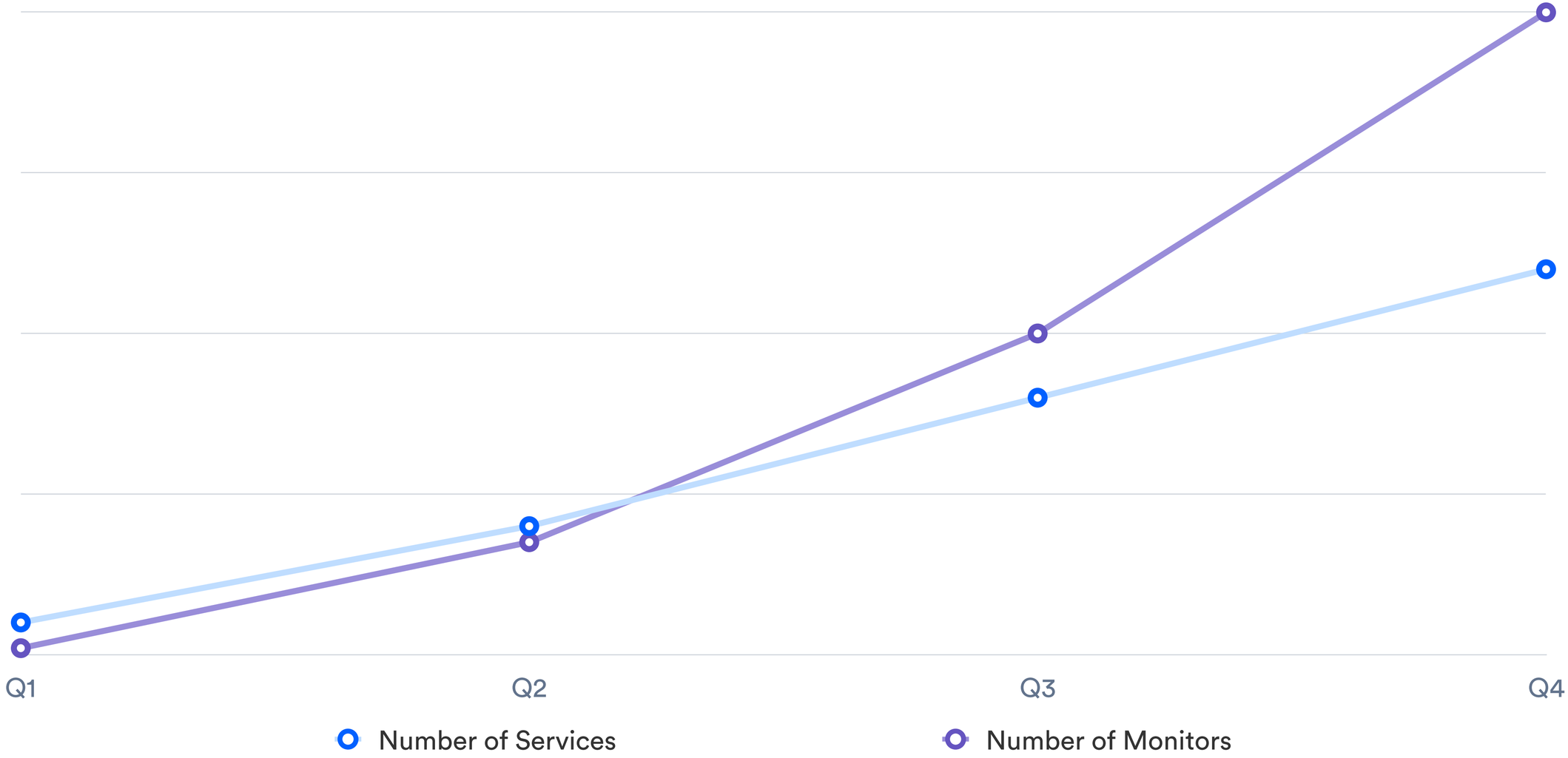



MONITORING
CI/CD
ORCHESTRATION
IMAGE
TOOLING

Monitoring may seem confusing



Realities of a growing business



Scaling up Monitoring

Amount of services ↑

\+

Amount of monitoring ↑

= Time spent on building monitoring?

Scaling up Monitoring

Amount of services ↑

\+

Amount of monitoring ↑

=

Time spent on building monitoring?

Probably ↑ but you want it ↓

You remember terraform? We can use that with Datadog too....

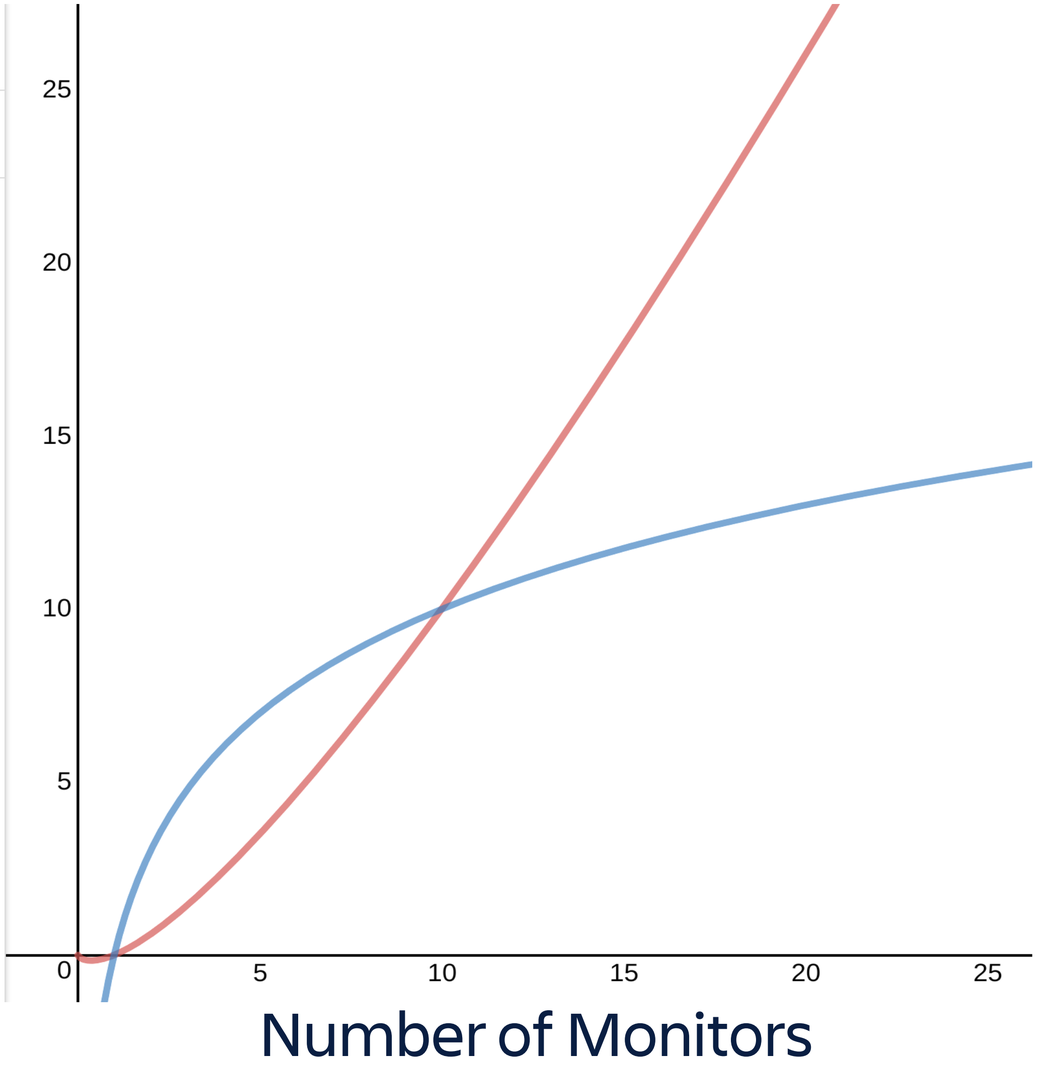
Manual

Terraform

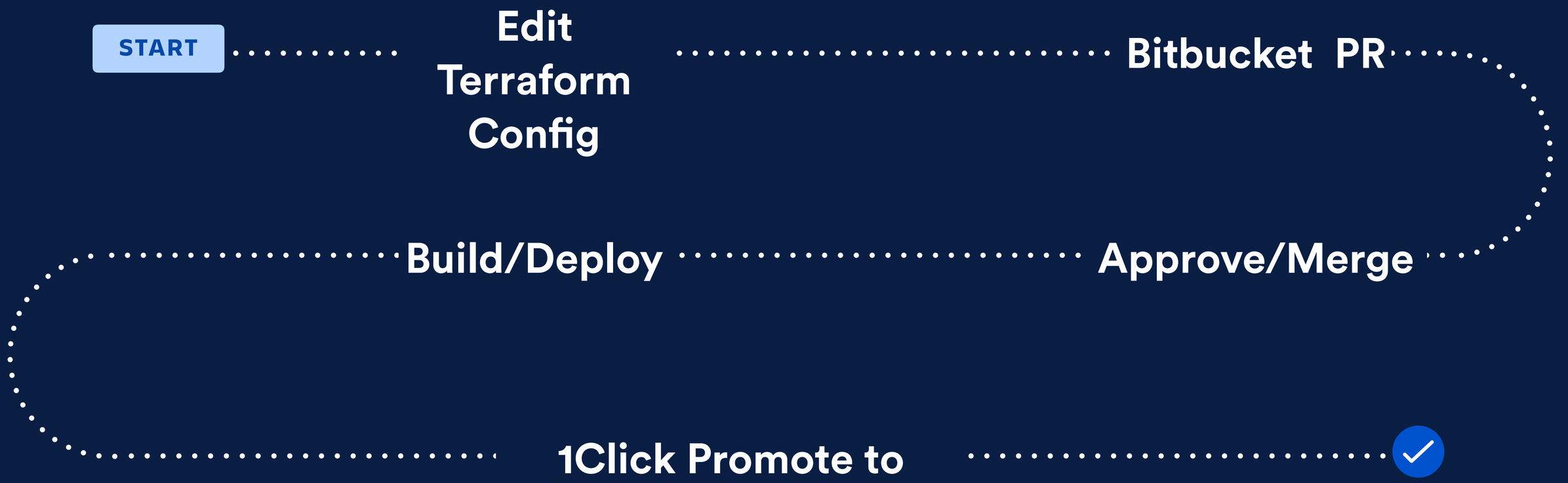
1 $y = n \cdot \log(n)$

2 $y = 10 \cdot \log(n)$

3



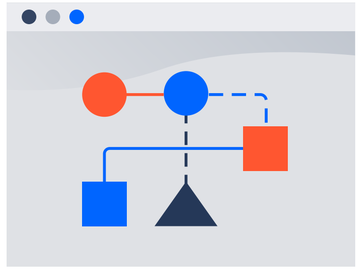
TERRAFORM CI/CD WITH BITBUCKET PIPELINES



Planning Migrations

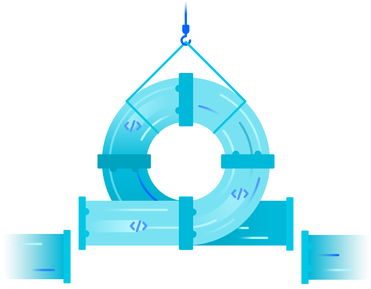


How 2 Migrate



Pre-Configure

New environment & application stack configured and ready to start



Migrate Data

Database, Shared Filesystem, Other Persistent Data



Cutover DNS, or Abort

Switch DNS to new stack, or abort migration

Migration Fails





You can spend an infinite time planning

Try stick to some kind of deadline, but be flexible you won't get to the **perfect** plan.



Involve stakeholders early

Will there be downtime? How long? Can we minimise it? Is it worth the effort to minimise it?



Document, document, document

Anyone on your **team** should be able to pick up your migration plan, understand what needs to be done and execute it



Pre-configure New Environment

Prepare anything that can be done before the migration starts



Secure Data

Ensure customer data will be protected during the migration





Automate Everything

If you feel like you need to ssh and manually run some command, script it instead



Don’t be afraid to abort

If something unexpected occurs you’re likely better off aborting and trying again another day



Have a test plan

Thoroughly test your migration plan in the pre-production environments

We did it!



Improvements



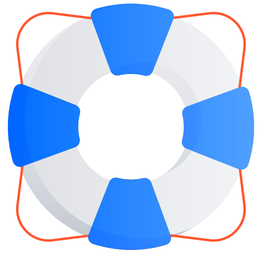
WE THOUGHT, GREAT!

WE THOUGHT, GREAT!
BUT HOW CAN WE MAKE THIS EVEN BETTER!?

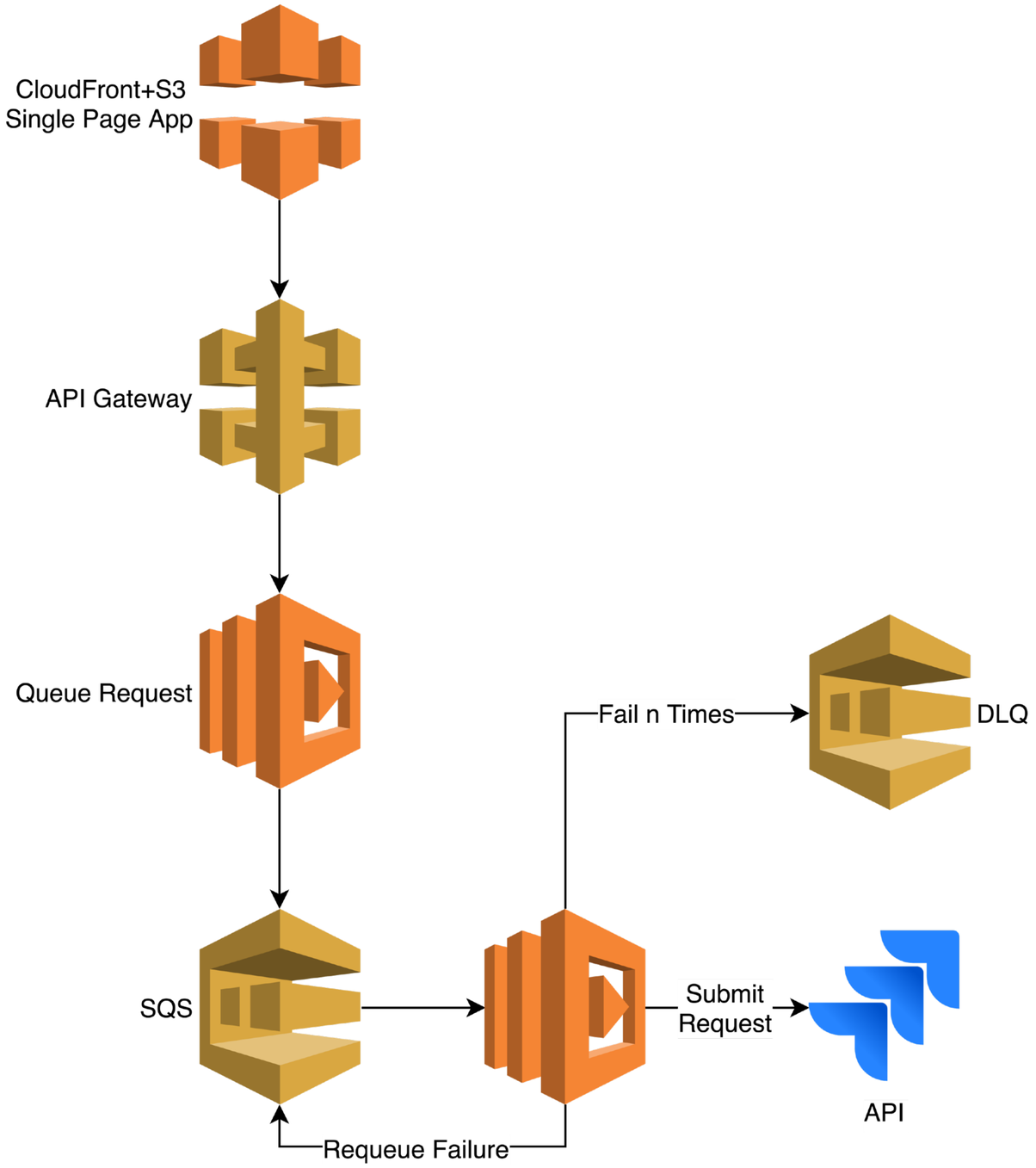
Core Capabilities



Ticket Creation



Updating Tickets

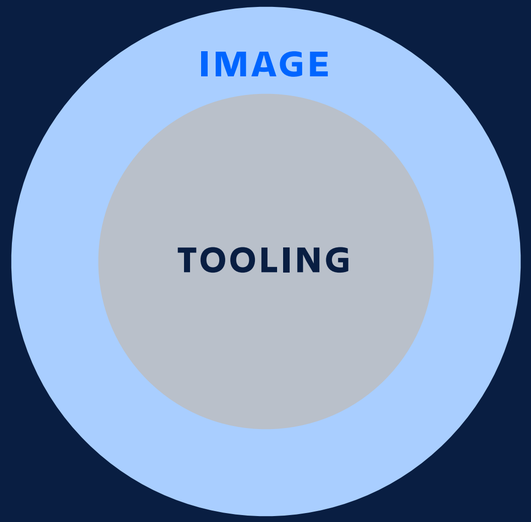


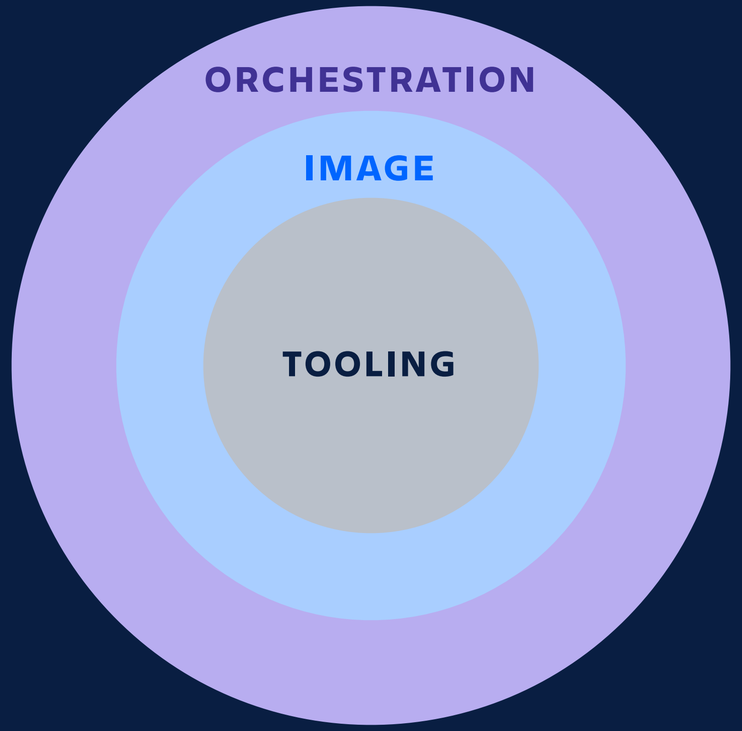
Wrapping Ticket Creation API

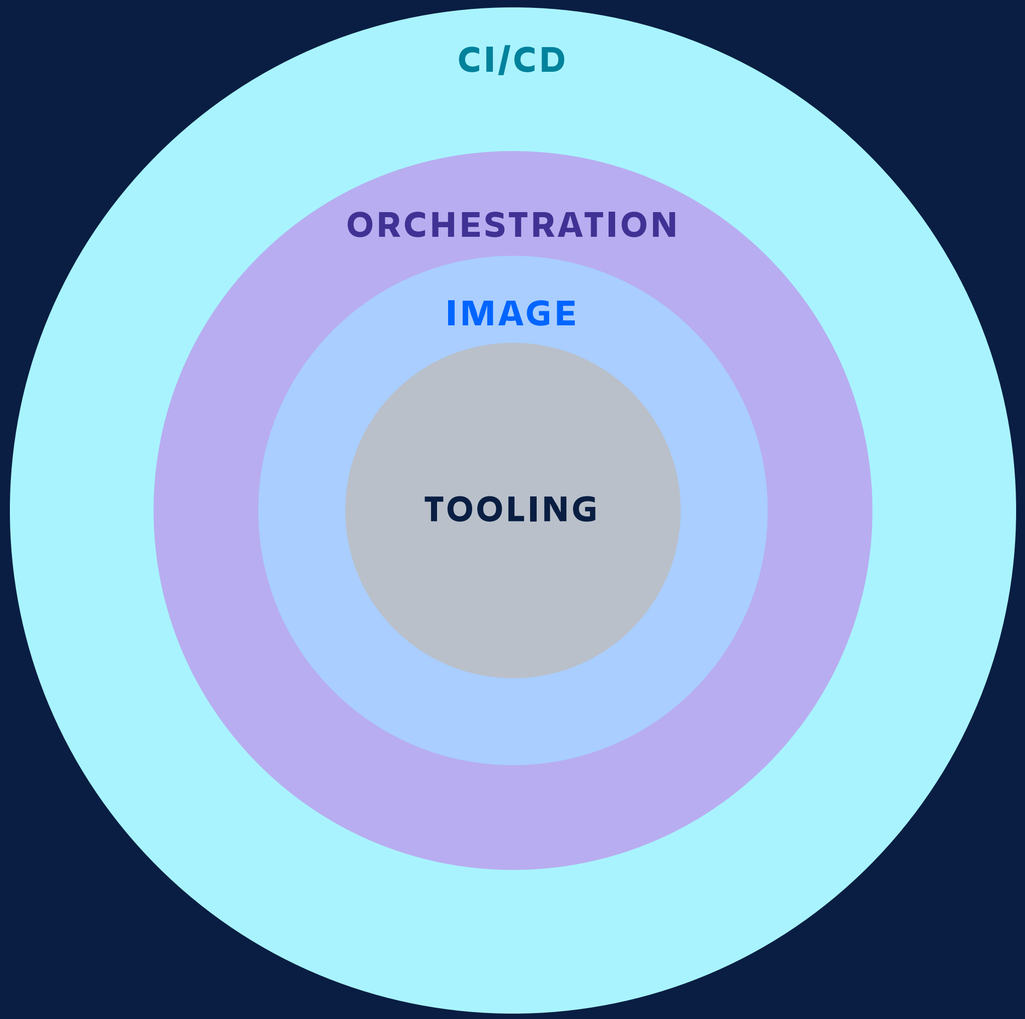
Quick Recap

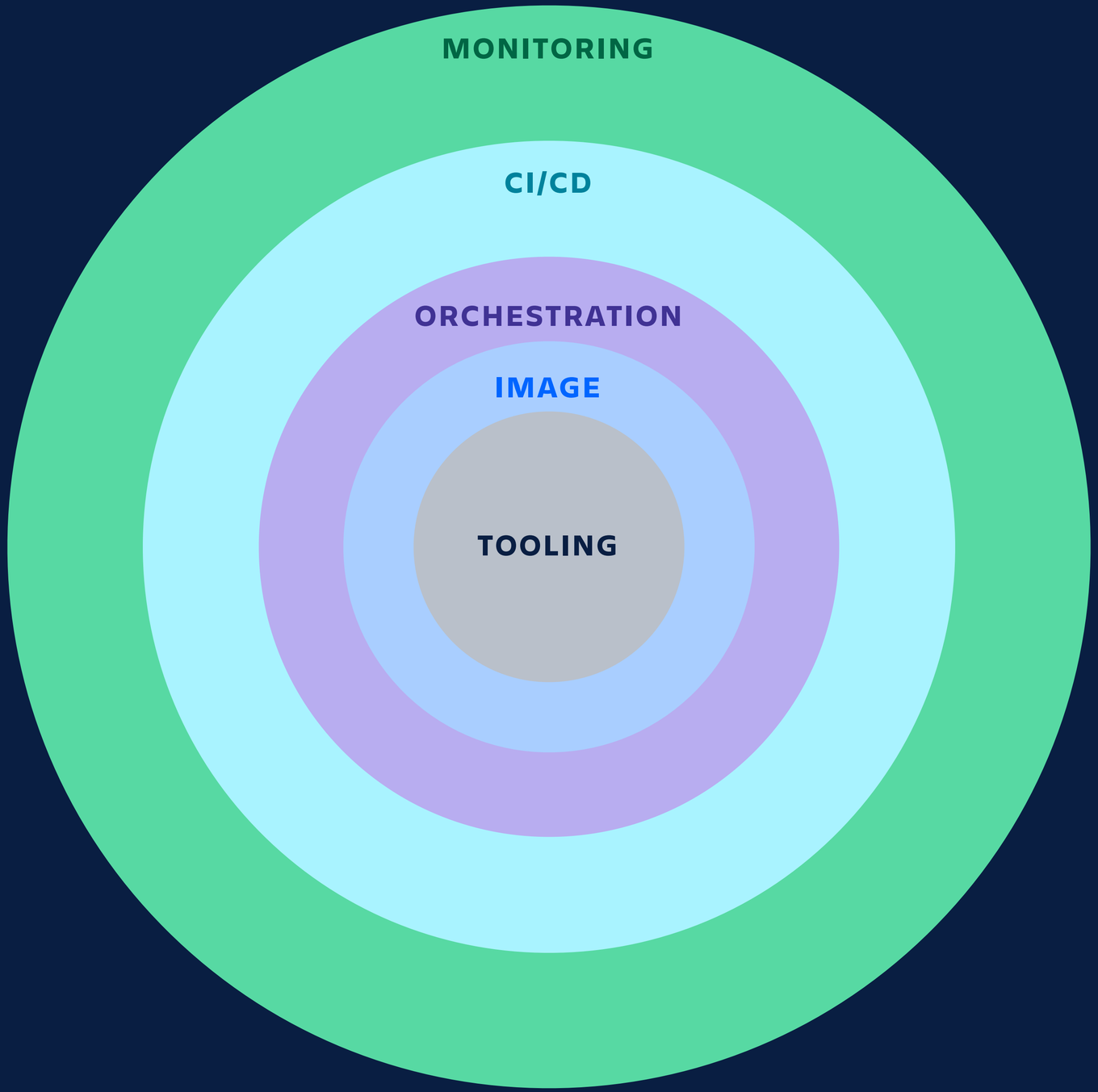


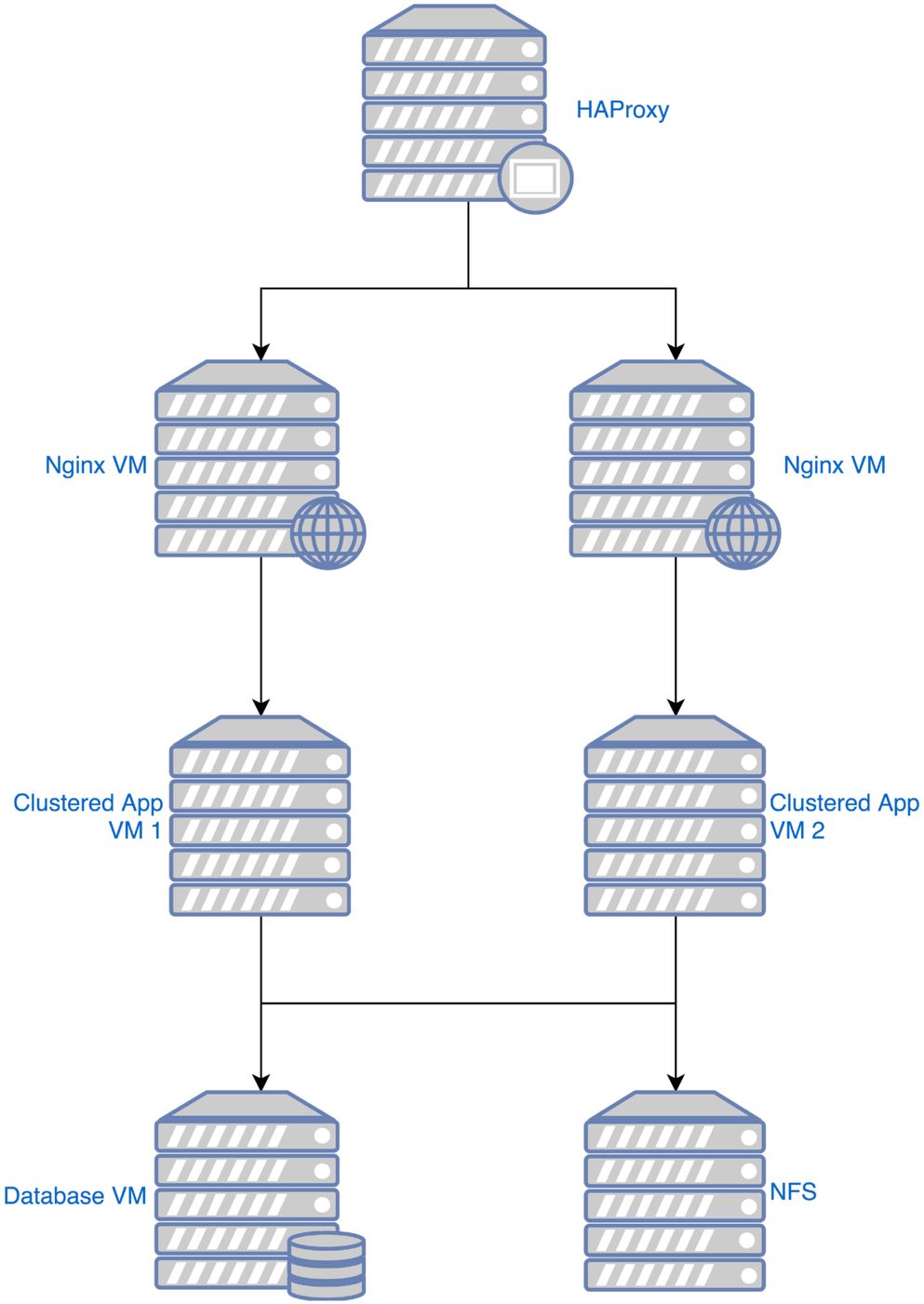
TOOLING

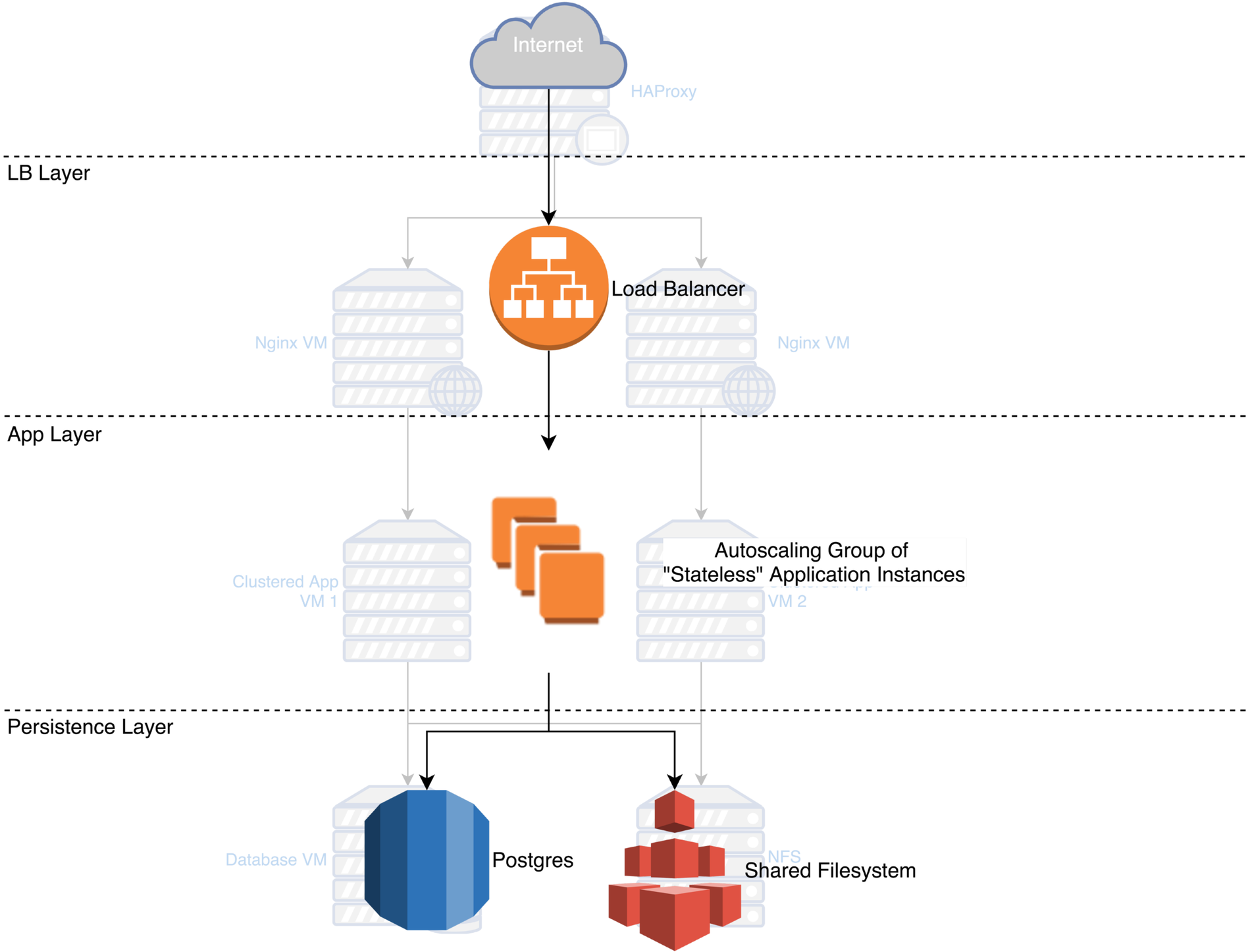

IMAGE
TOOLING


ORCHESTRATION
IMAGE
TOOLING


CI/CD
ORCHESTRATION
IMAGE
TOOLING


MONITORING
CI/CD
ORCHESTRATION
IMAGE
TOOLING


HAProxy
Nginx VM
Nginx VM
Clustered App
VM 1
Clustered App
VM 2
Database VM
NFS


Internet
HAProxy
LB Layer
Load Balancer
Nginx VM
Nginx VM
App Layer
Clustered App
VM 1
Autoscaling Group of
"Stateless" Application Instances
VM 2
Persistence Layer
Database VM
Postgres
NFS
Shared Filesystem

Key Takeaways



Cloud Services

are not for everyone, but they are for almost everyone



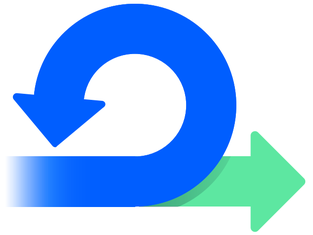
Use the right tooling

to make your life easier in the long run



Invest time in planning

before pushing on with migrations



Iterate and Improve

to reduce impact of failure & improve resiliency


Q
&
A


Thank you!
@CRAIGKNOTT92