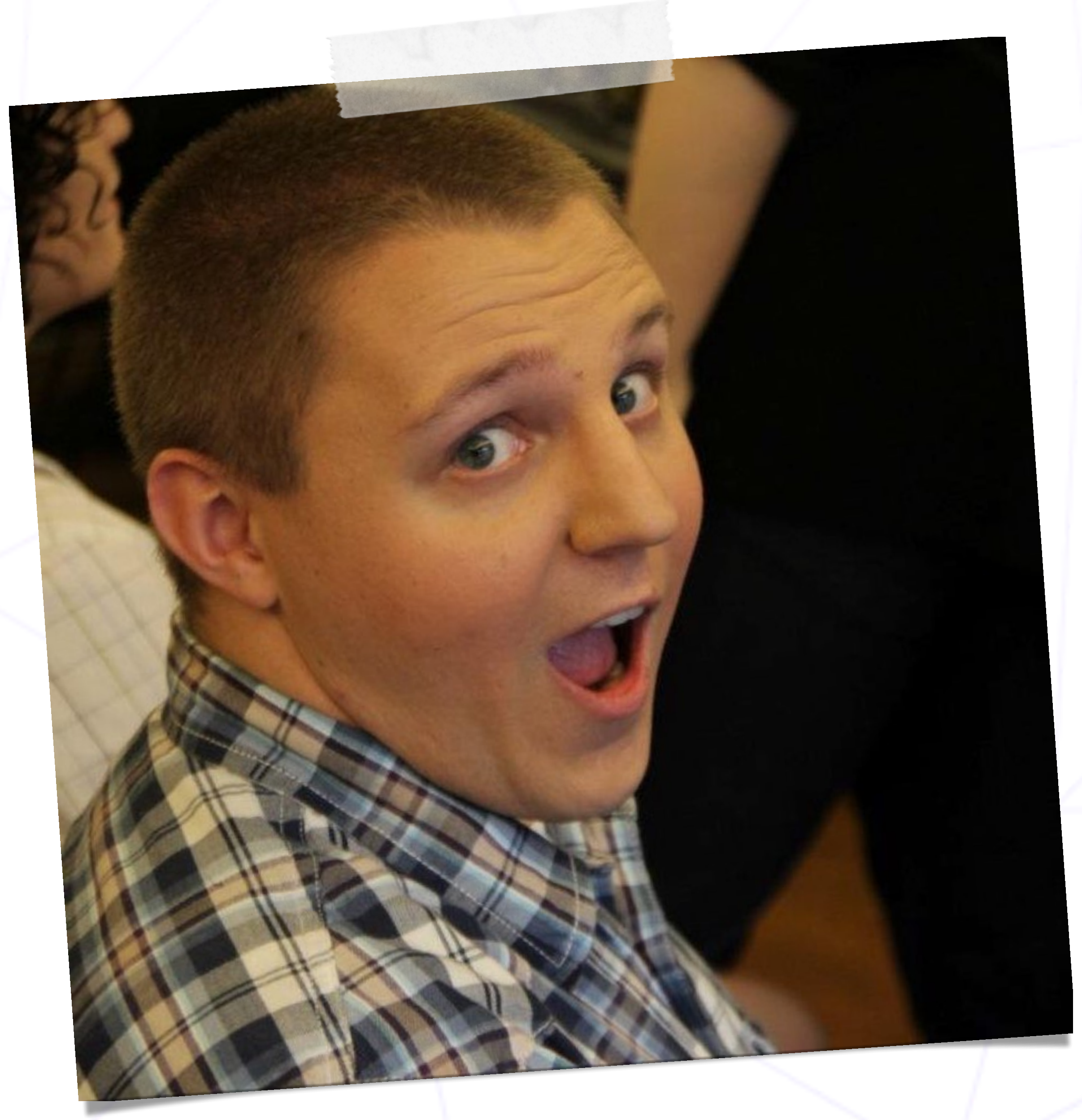
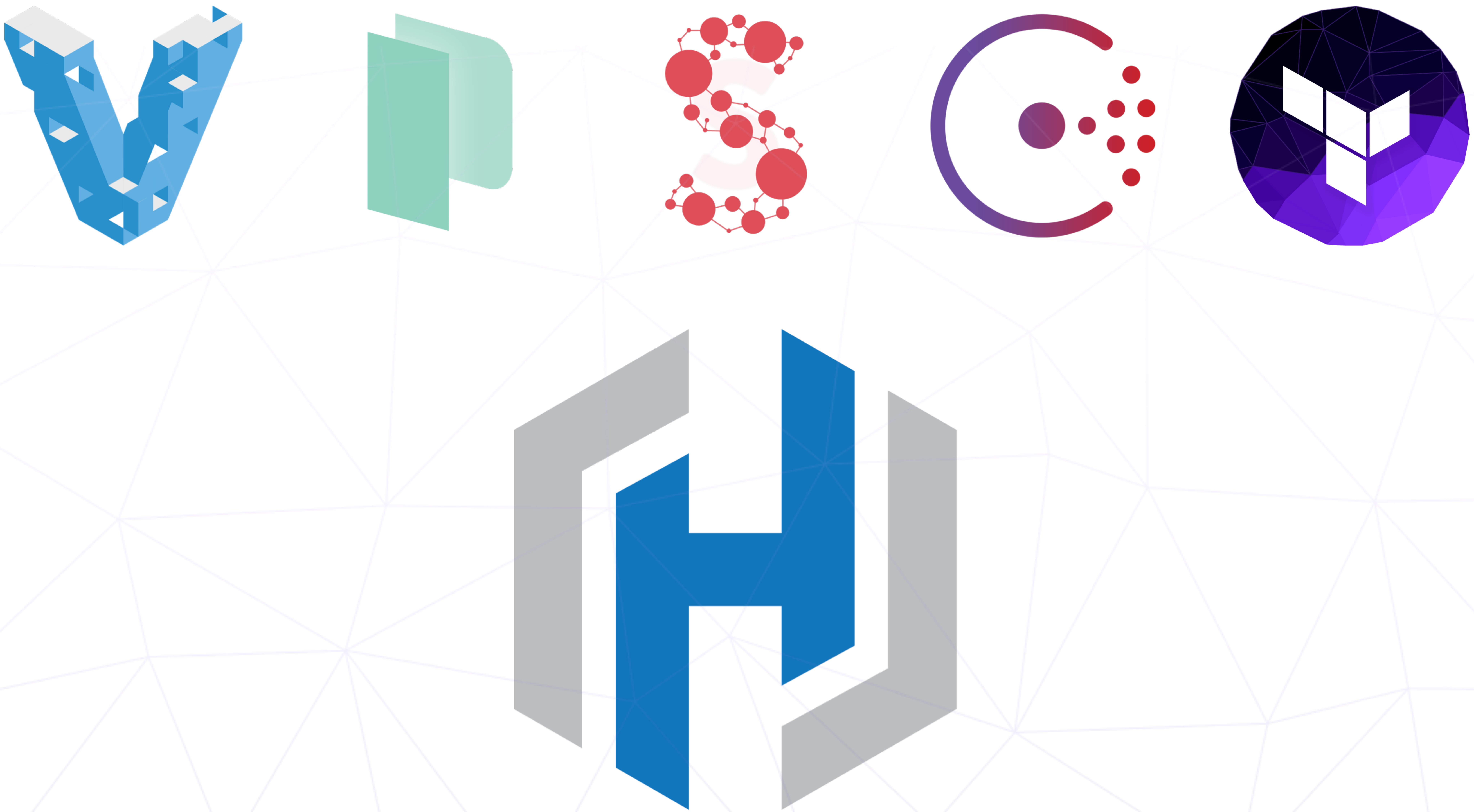
TAMING THE MODERN DATA CENTER

A Hybrid Talk for a Hybrid World



SETH VARGO

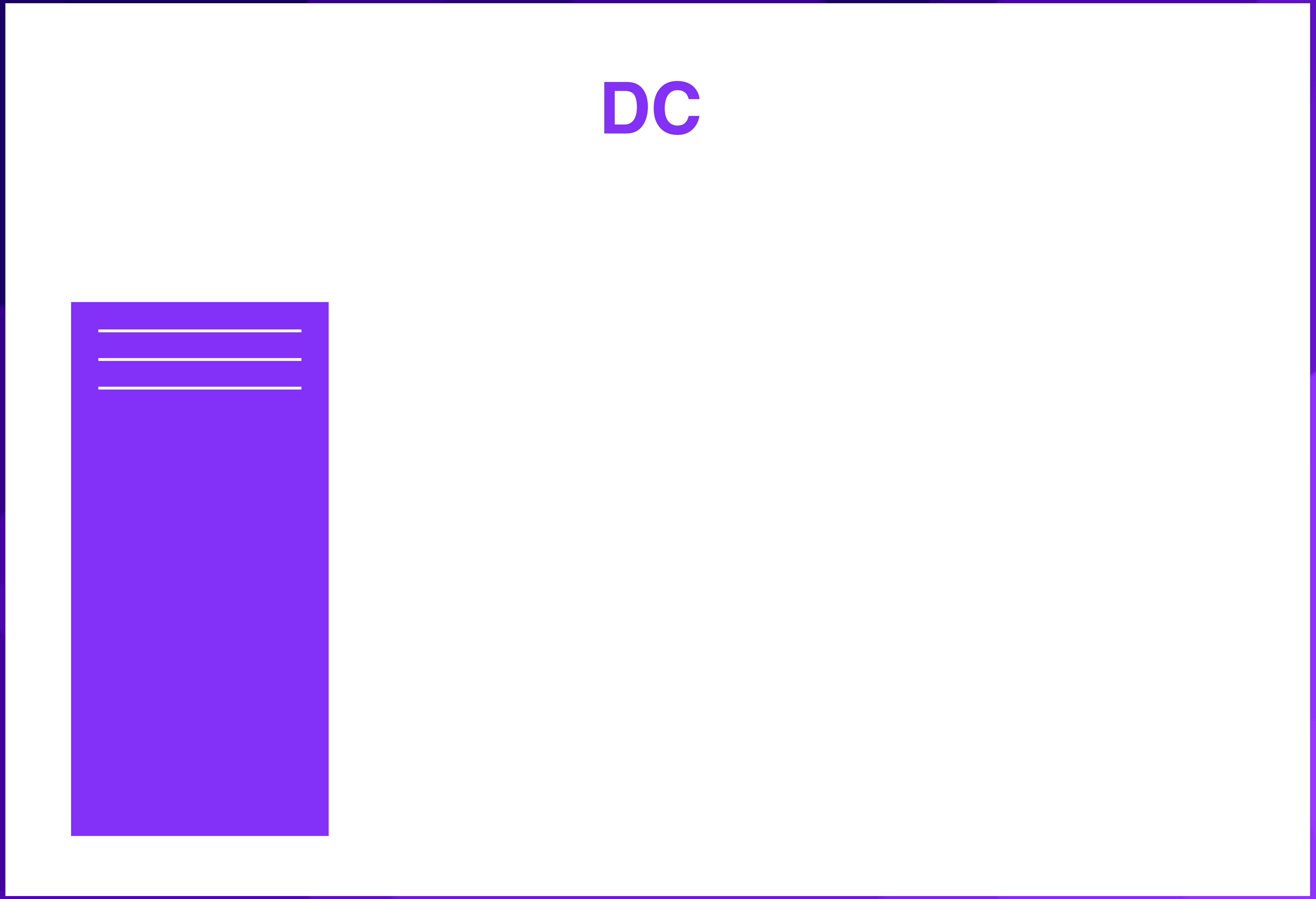
@sethvargo



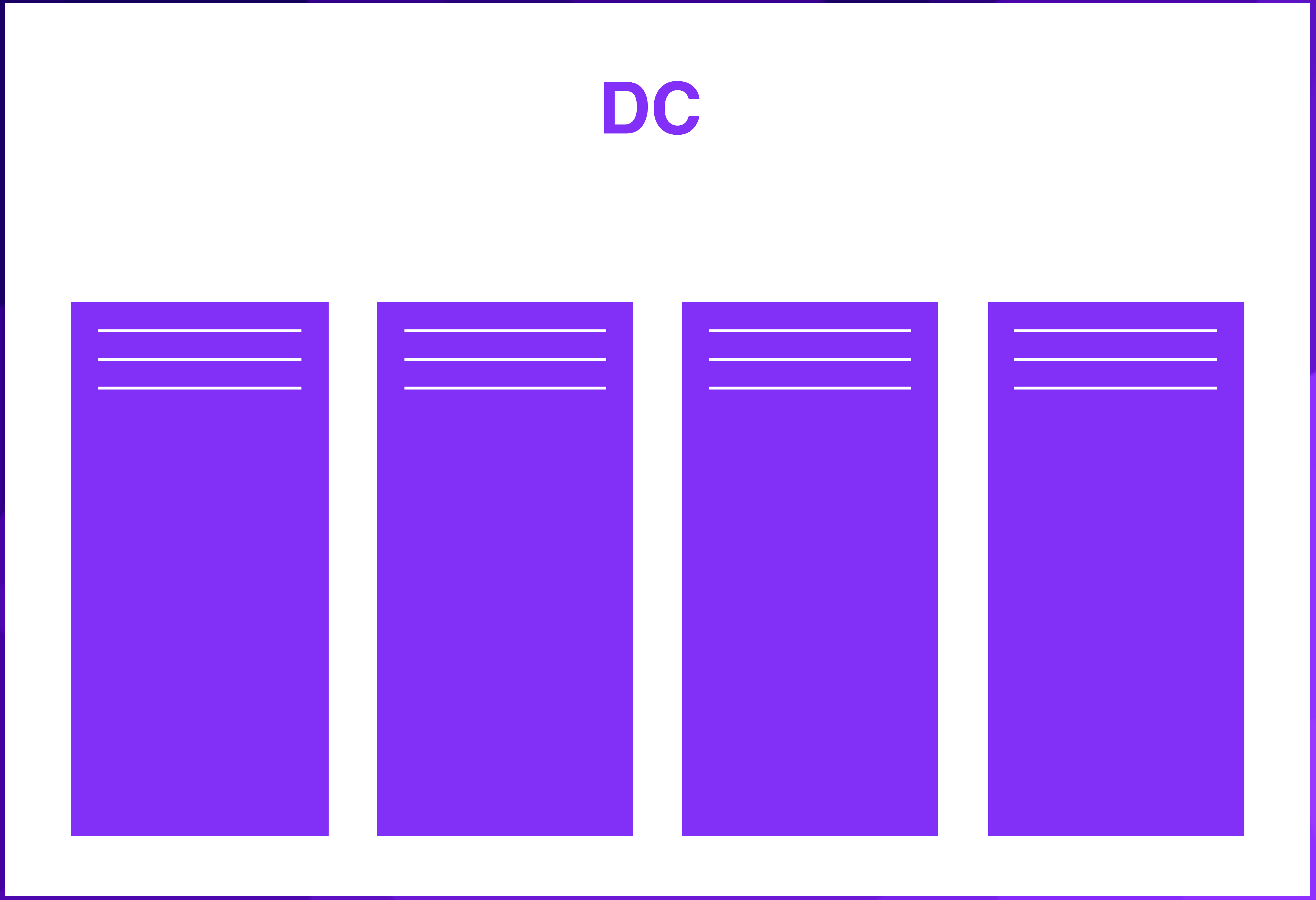
DC EVOLUTION

How did we get here?

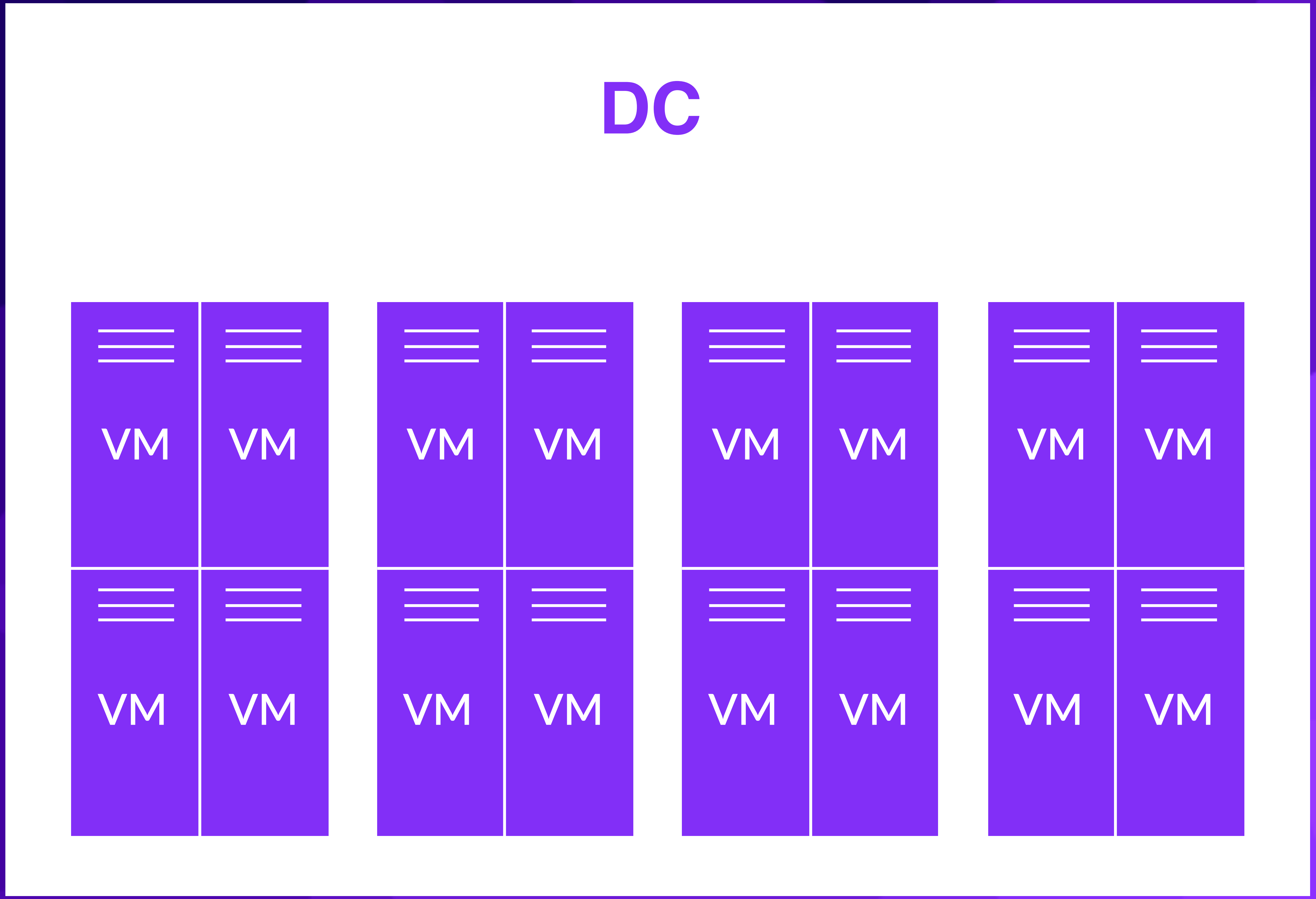
RISING DATACENTER COMPLEXITY



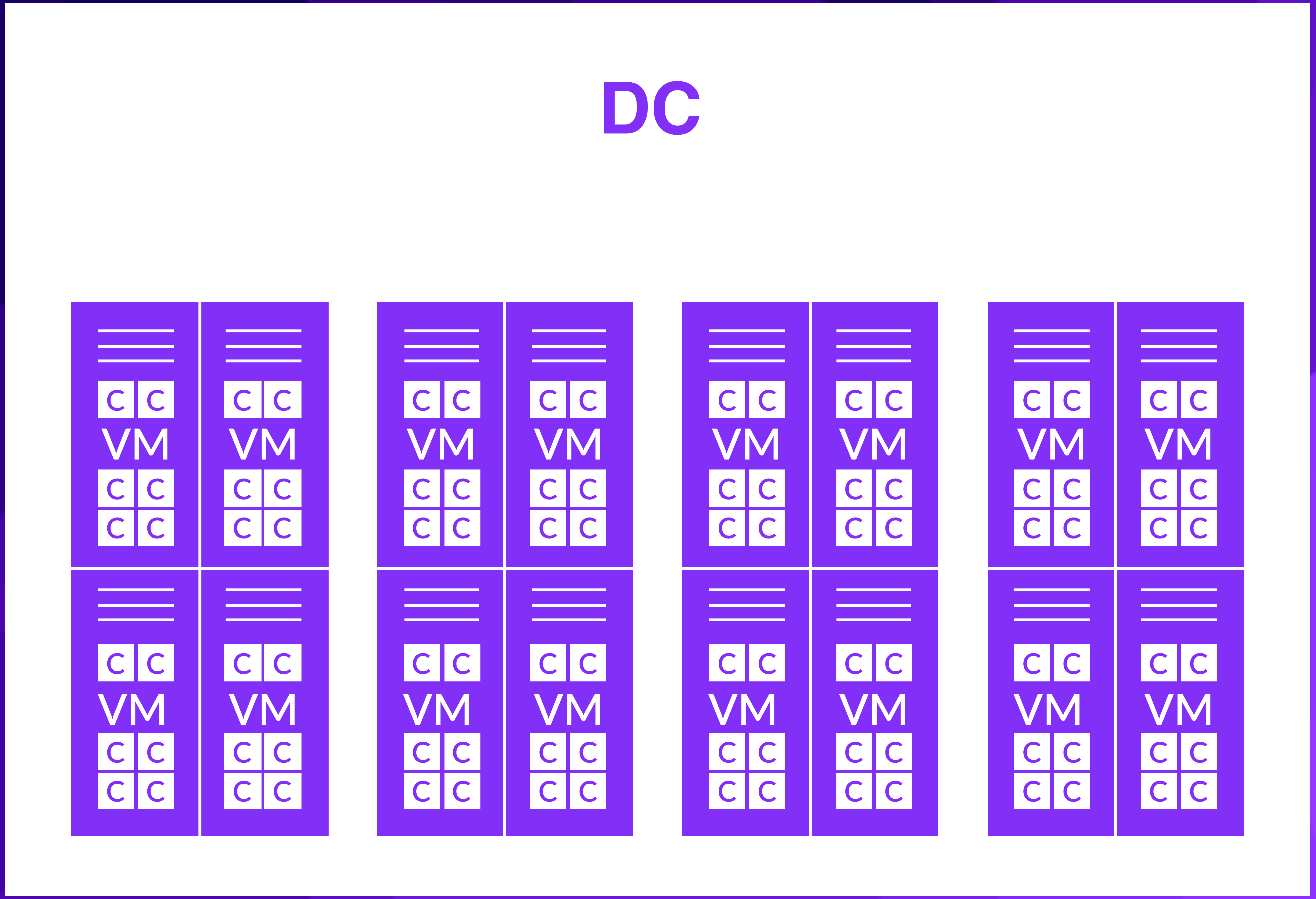
RISING DATACENTER COMPLEXITY



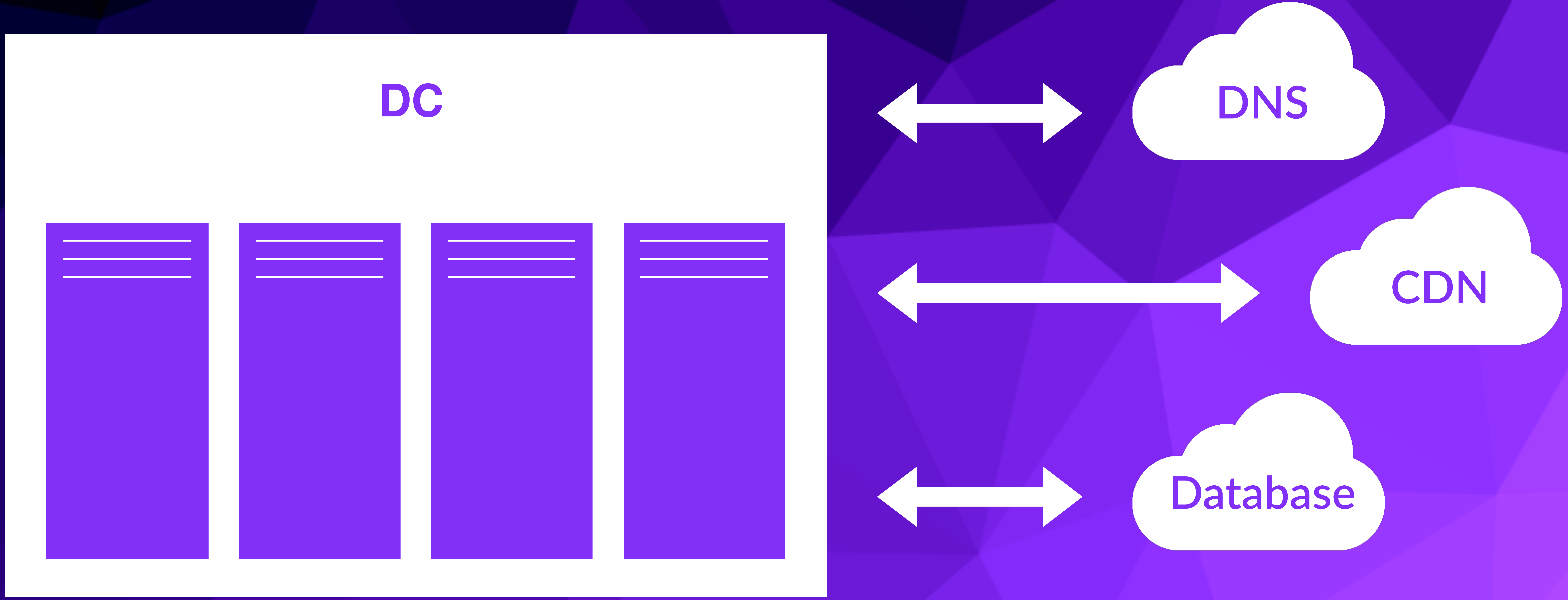
RISING DATACENTER COMPLEXITY



RISING DATACENTER COMPLEXITY



RISING DATACENTER COMPLEXITY



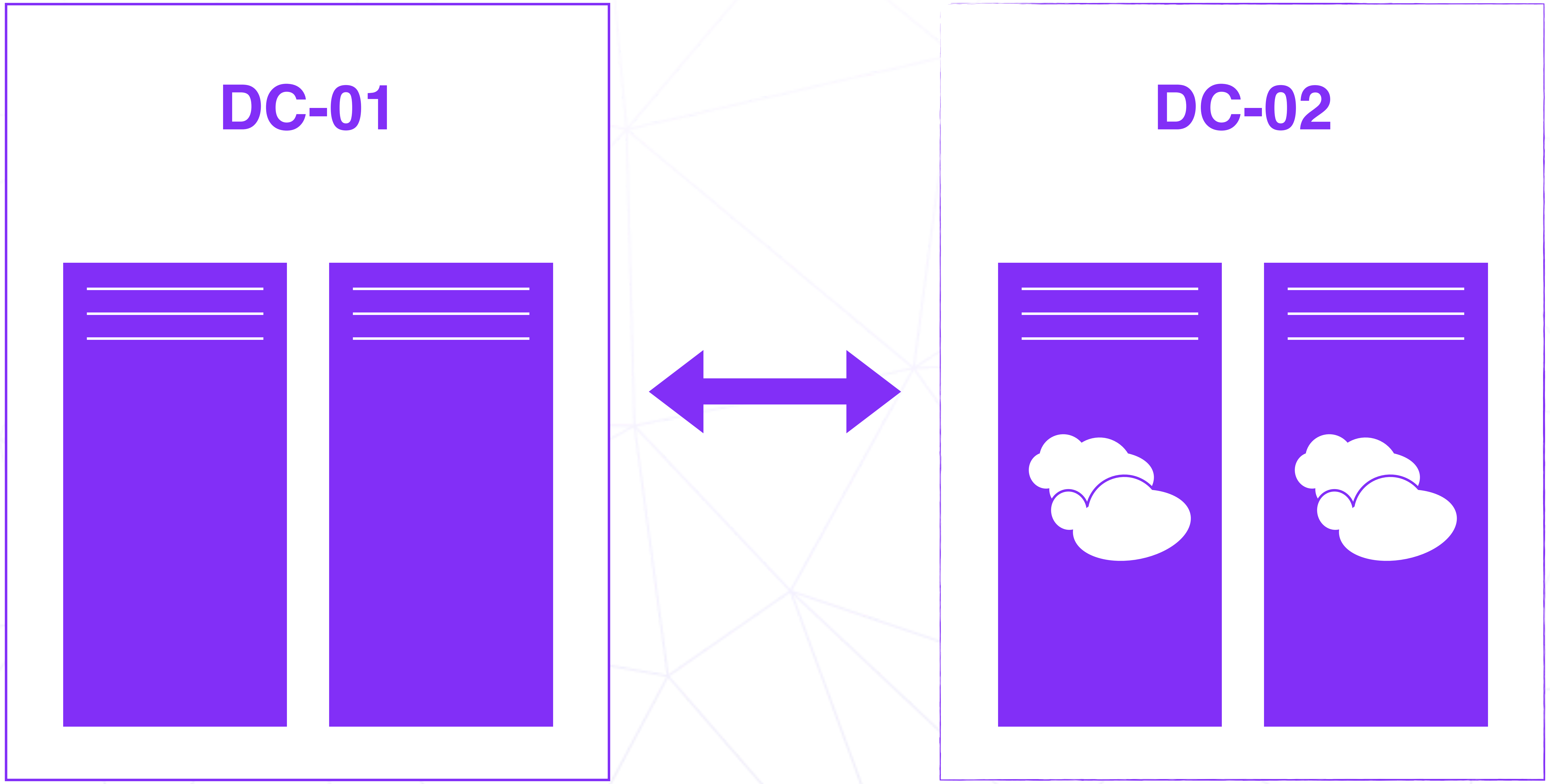
DC

DNS

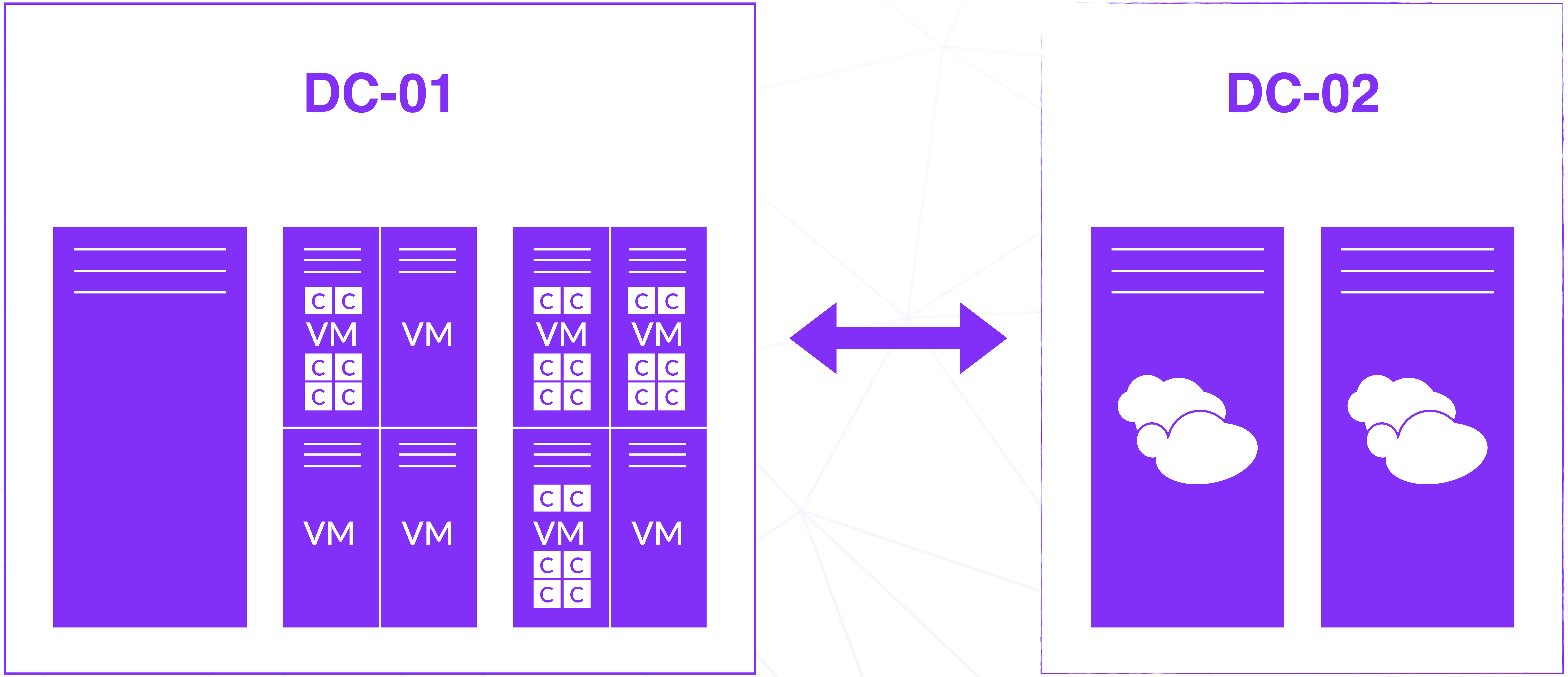
CDN

Database

RISING DATACENTER COMPLEXITY



RISING DATACENTER COMPLEXITY



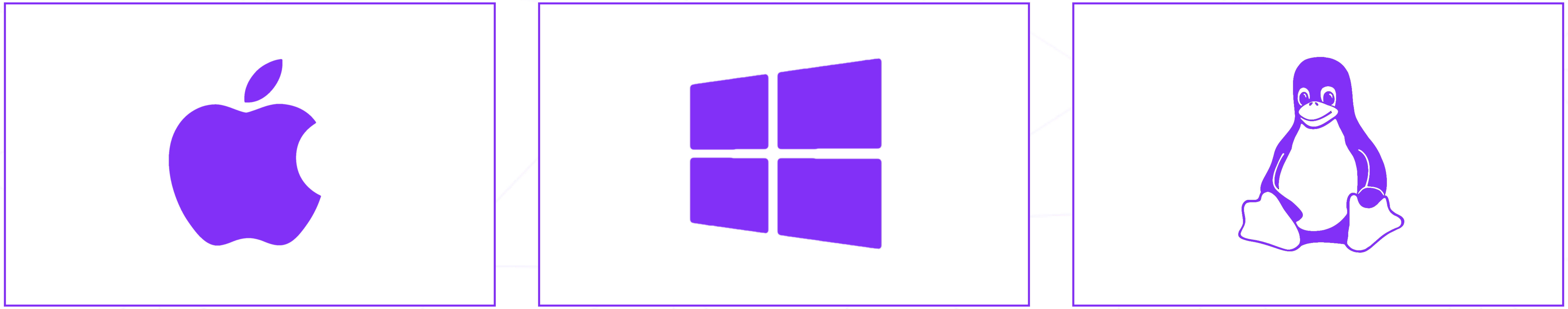
RISING DATACENTER COMPLEXITY

IaaS

PaaS

SaaS

RISING DATACENTER COMPLEXITY

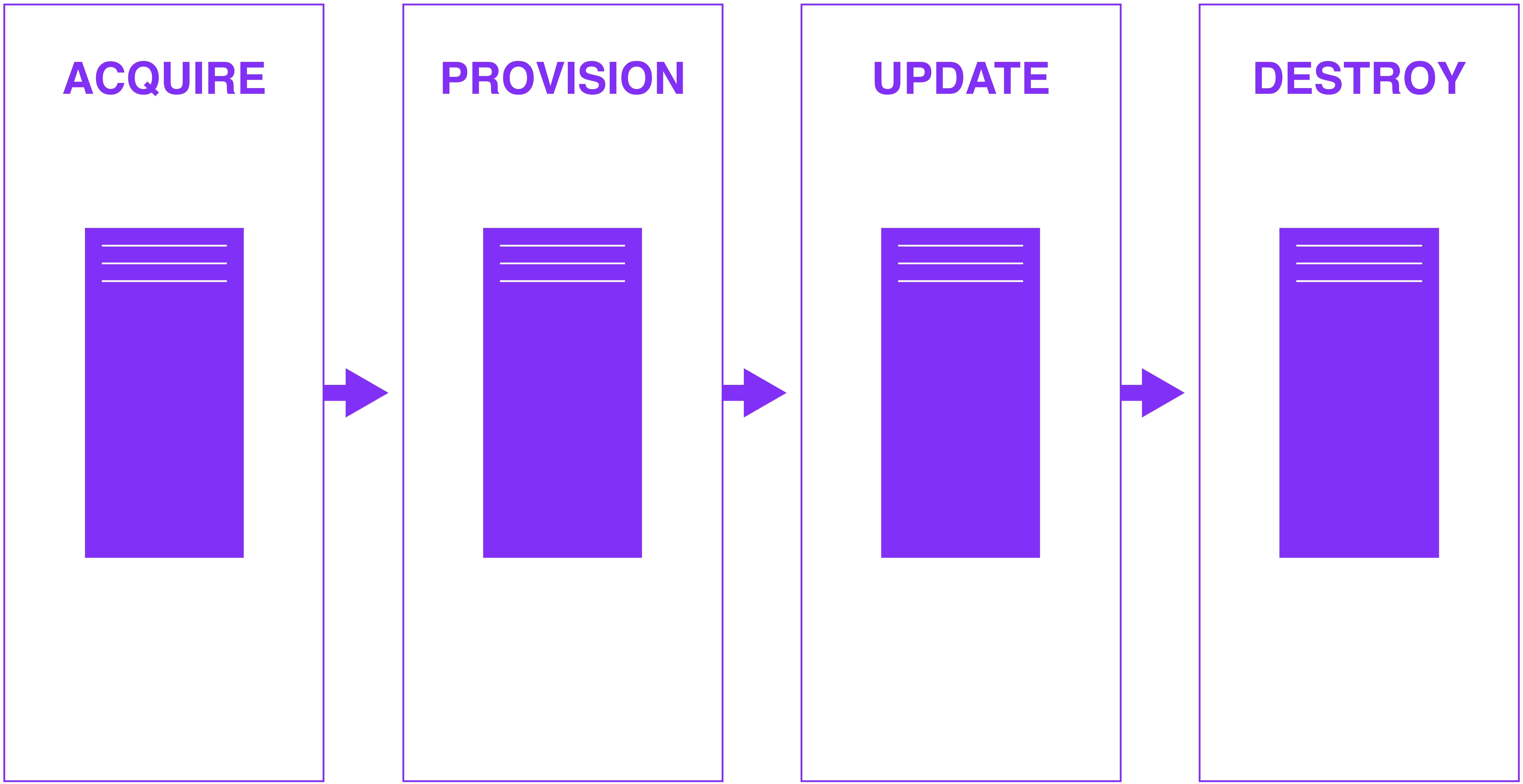


TAMING THE DC

Deployment + Maintenance

PREVIOUSLY

The APUD cycle

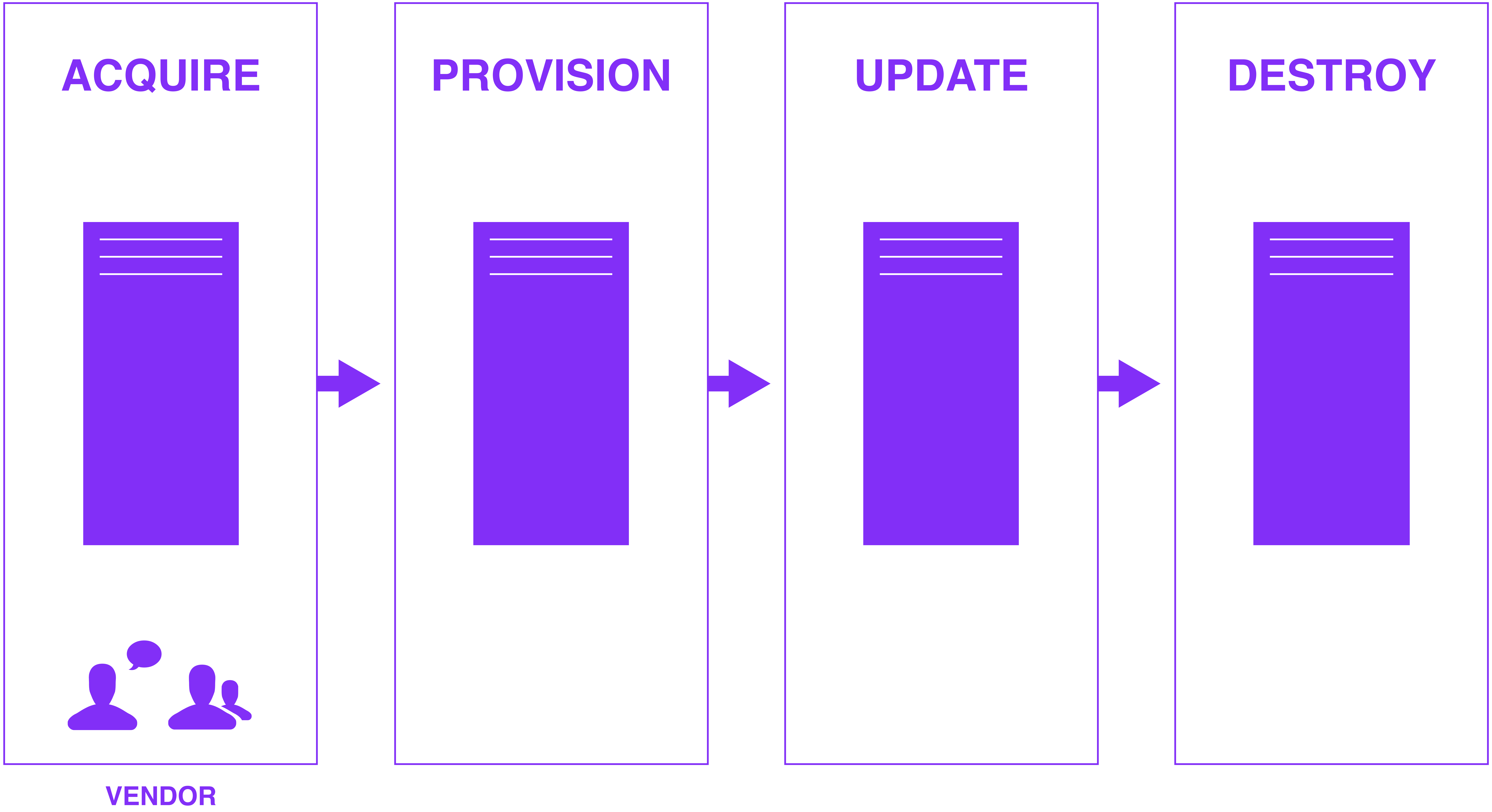


ACQUIRE

PROVISION

UPDATE

DESTROY



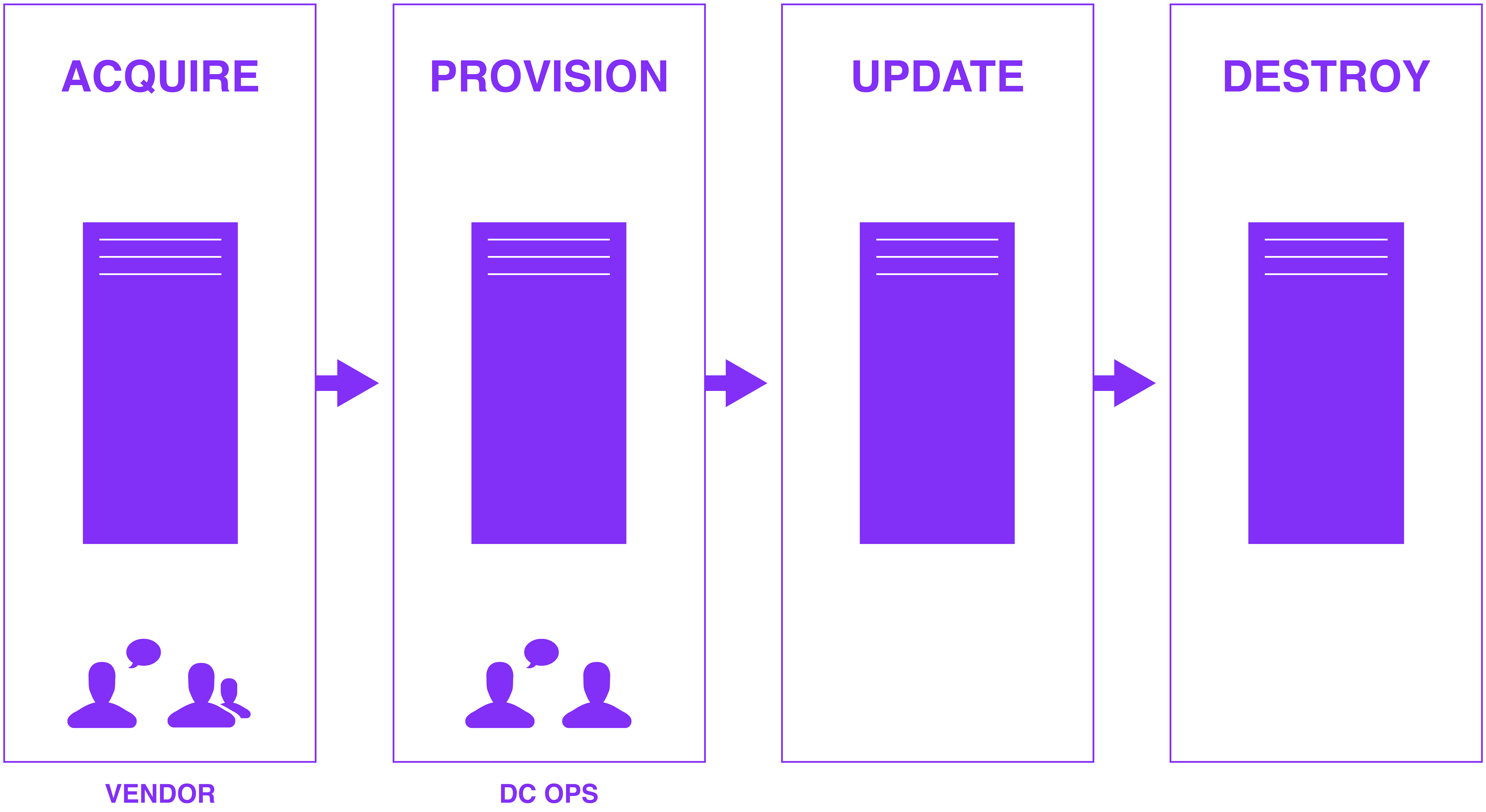
ACQUIRE

PROVISION

UPDATE

DESTROY

VENDOR



ACQUIRE

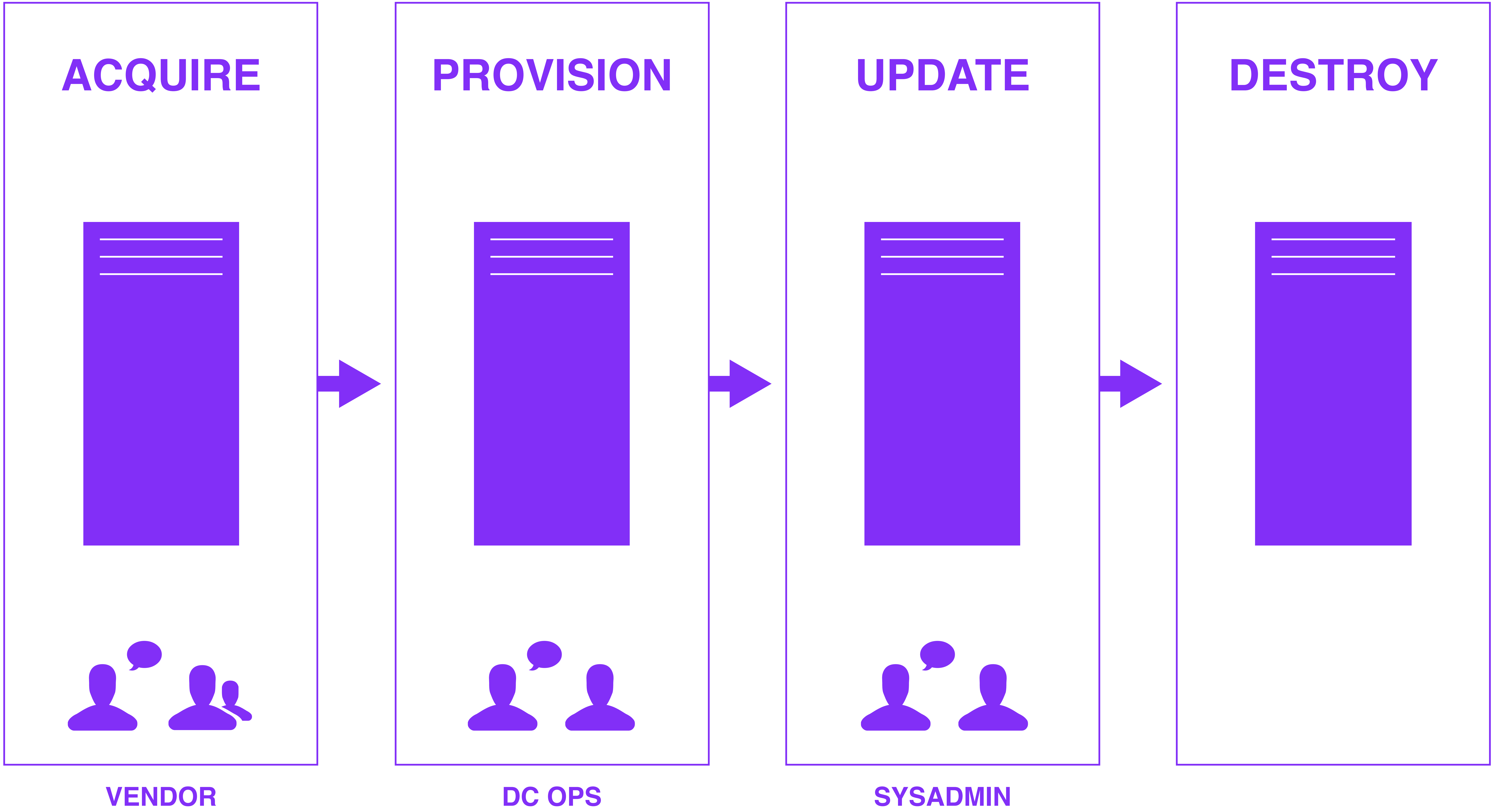
PROVISION

UPDATE

DESTROY

VENDOR

DC OPS



ACQUIRE

PROVISION

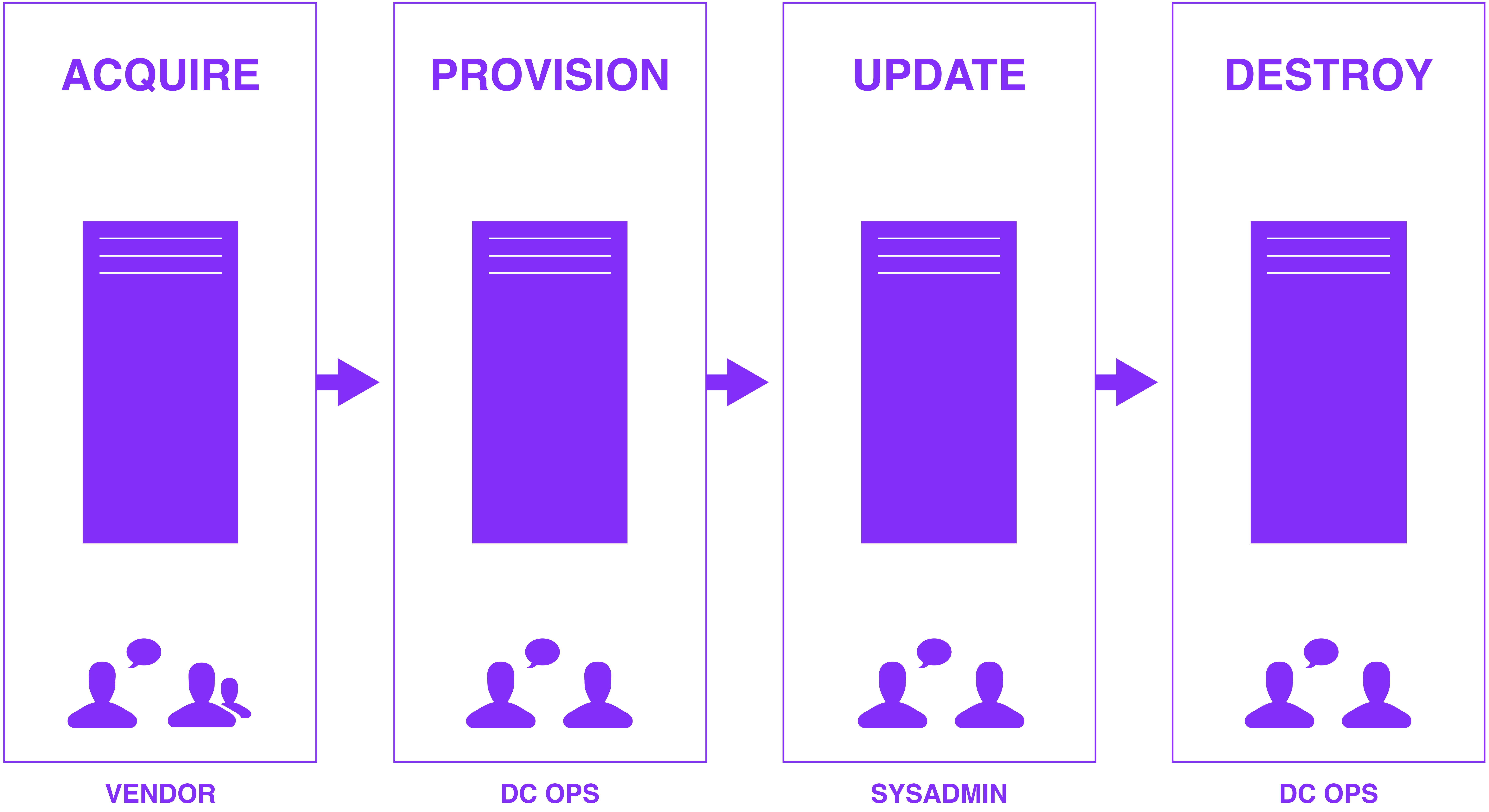
UPDATE

DESTROY

VENDOR

DC OPS

SYSADMIN



ACQUIRE

PROVISION

UPDATE

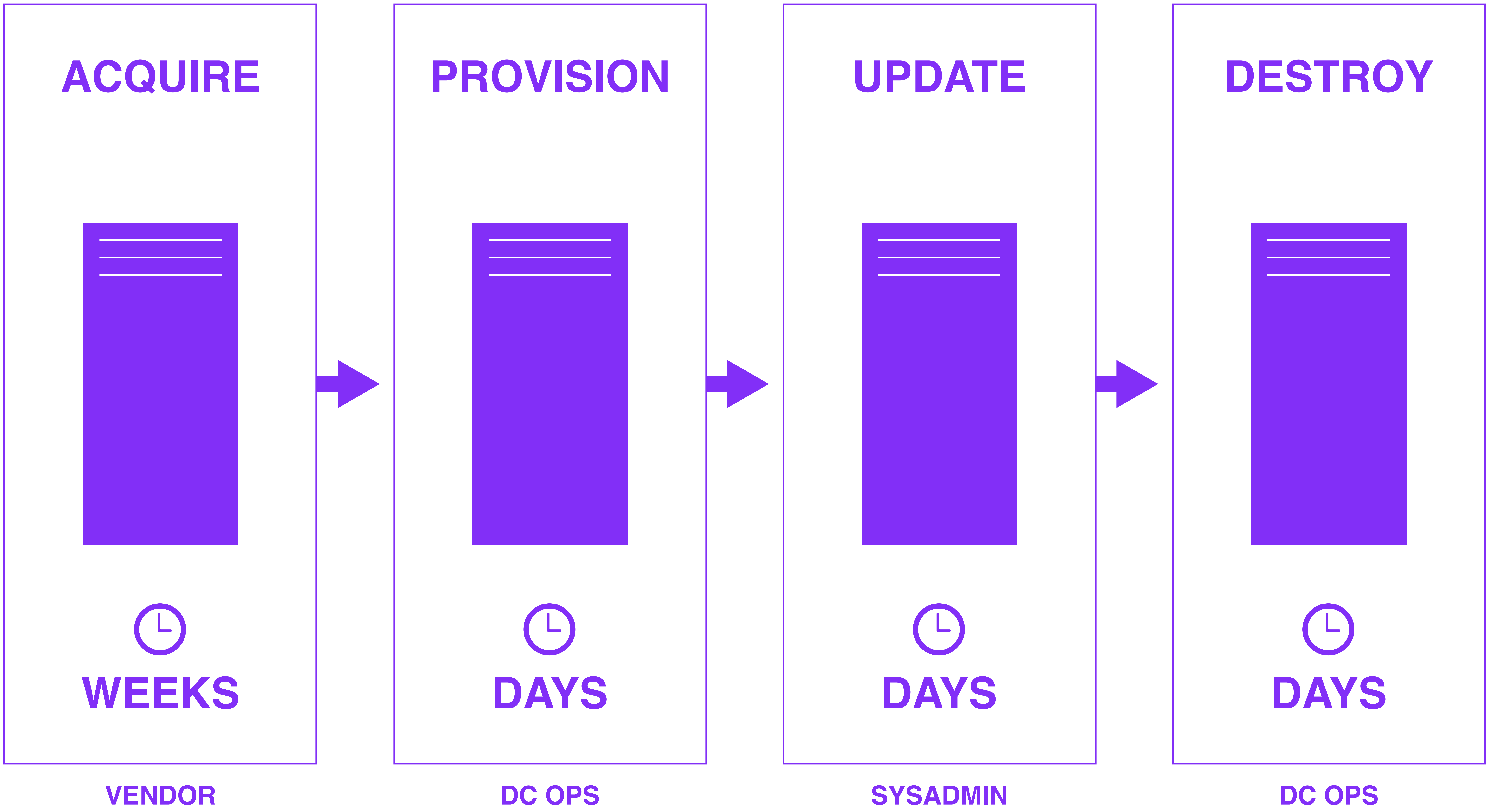
DESTROY

VENDOR

DC OPS

SYSADMIN

DC OPS



ACQUIRE

WEEKS

VENDOR

PROVISION

DAYS

DC OPS

UPDATE

DAYS

SYSADMIN

DESTROY

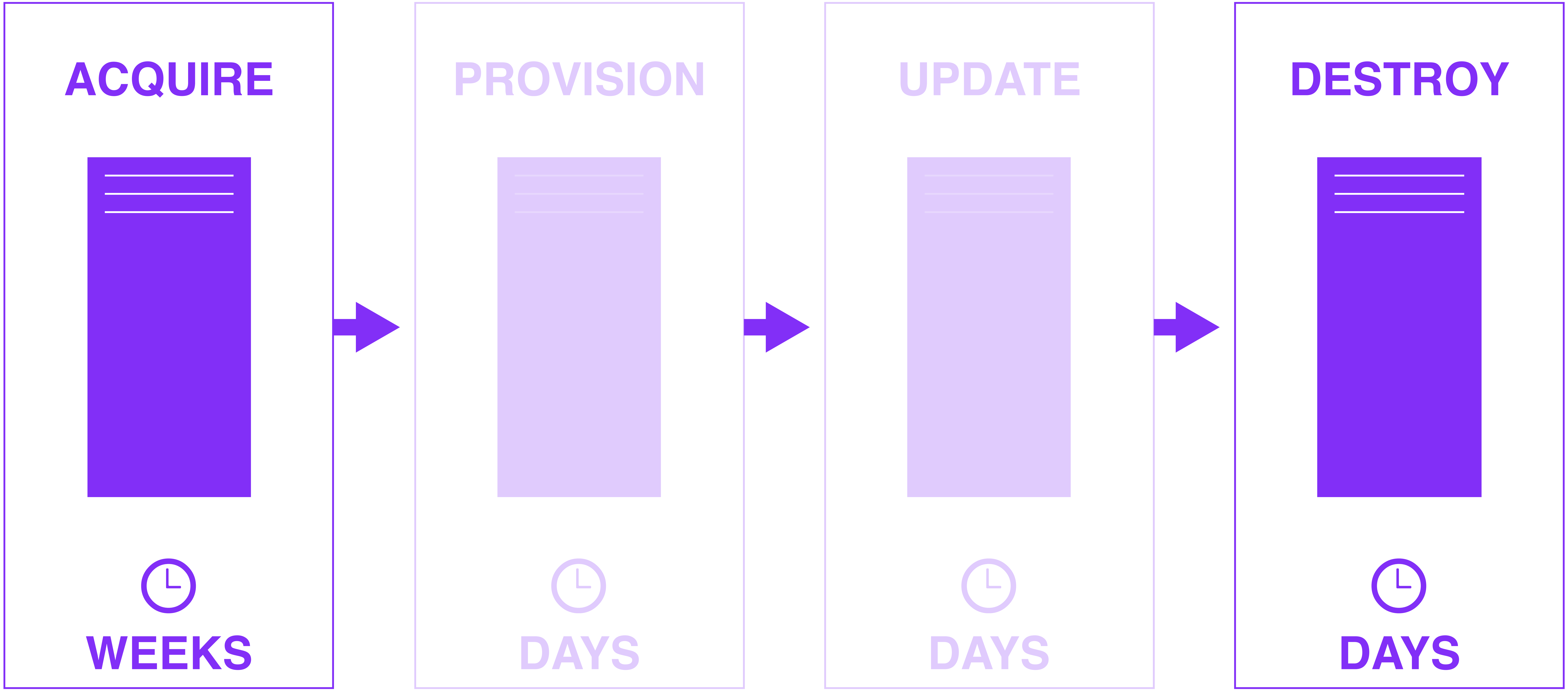
DAYS

DC OPS

PRESENTLY

The elastic compute and _aaS era

Elastic Compute



ACQUIRE

WEEKS

PROVISION

DAYS

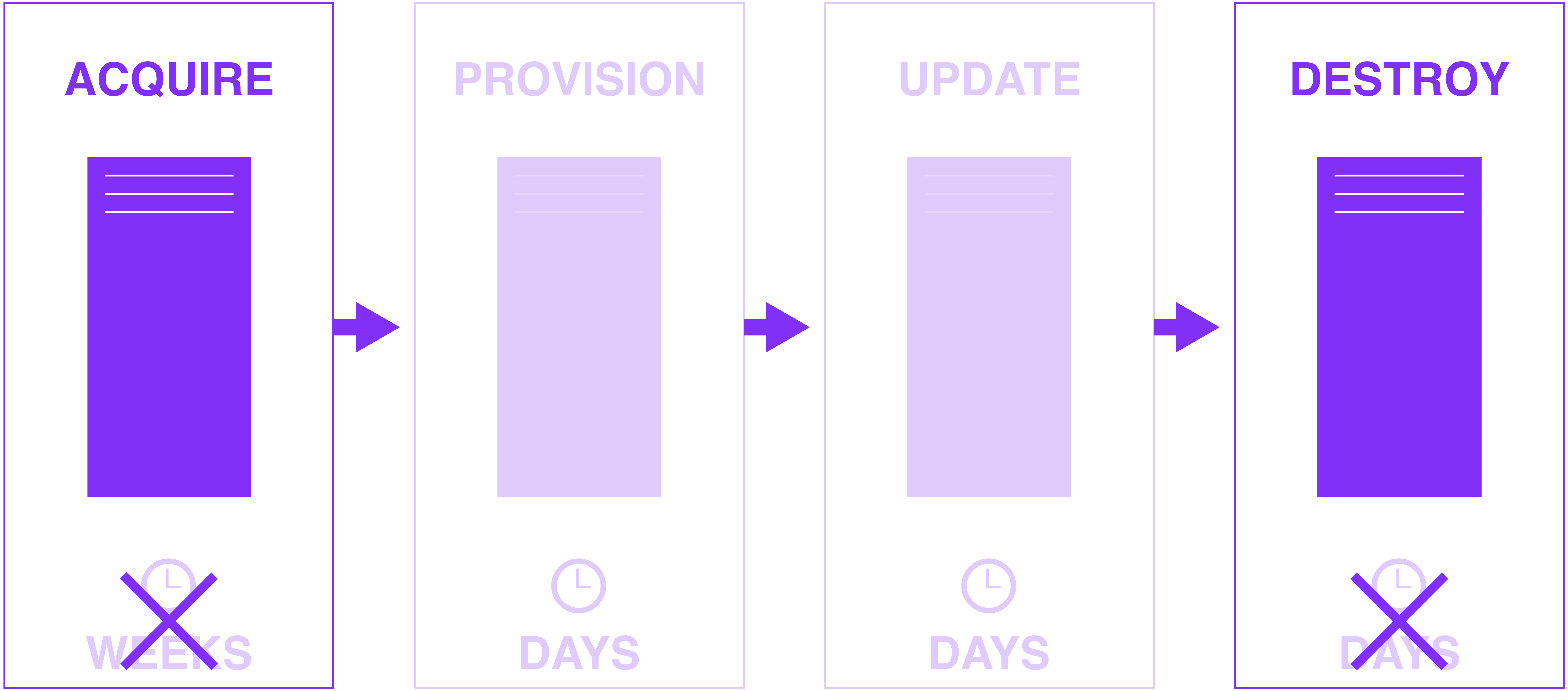
UPDATE

DAYS

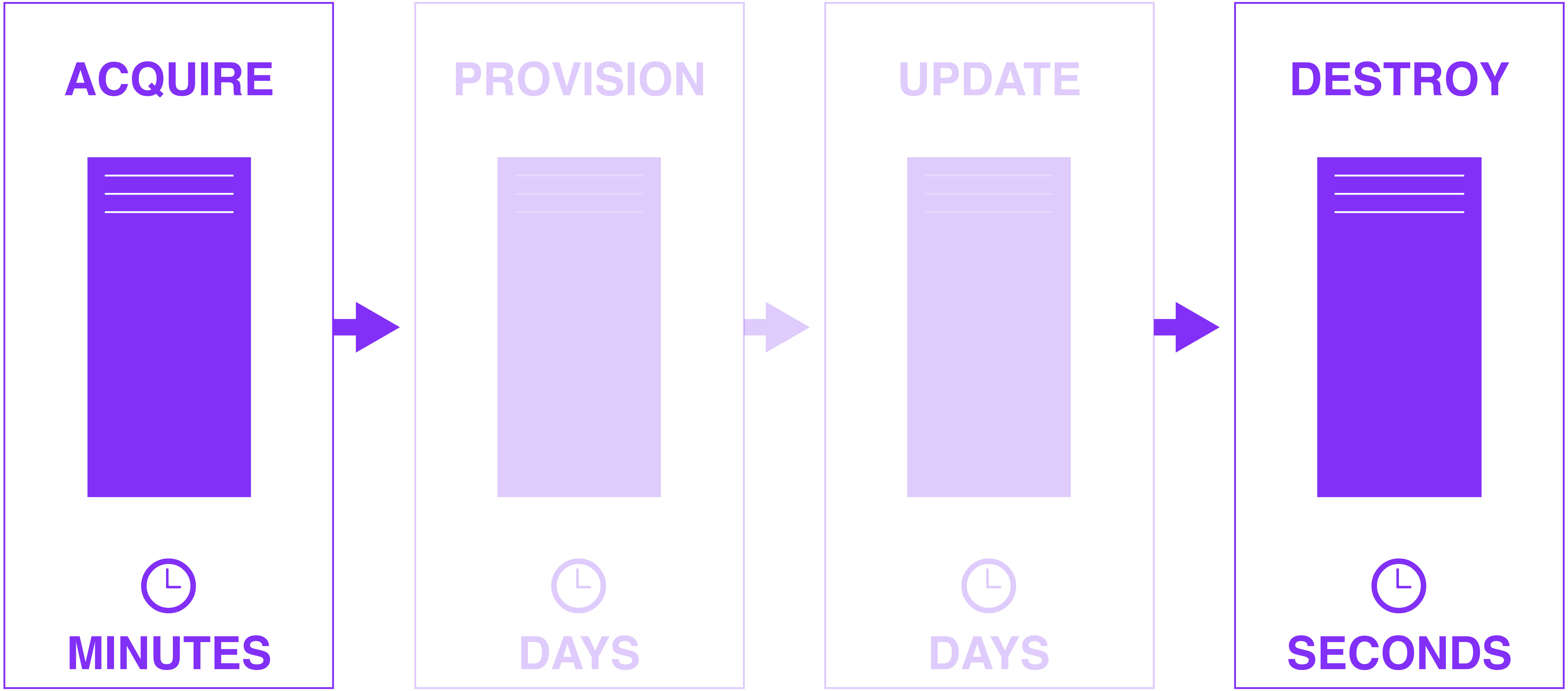
DESTROY

DAYS

Elastic Compute



Elastic Compute



ACQUIRE

MINUTES

PROVISION

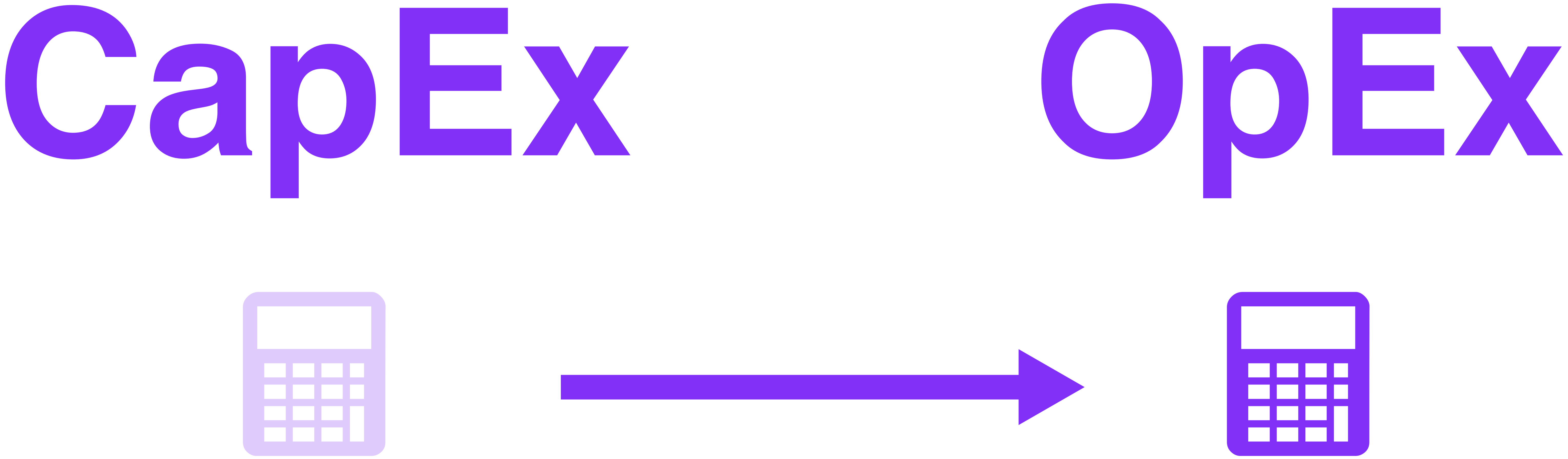
DAYS

UPDATE

DAYS

DESTROY

SECONDS

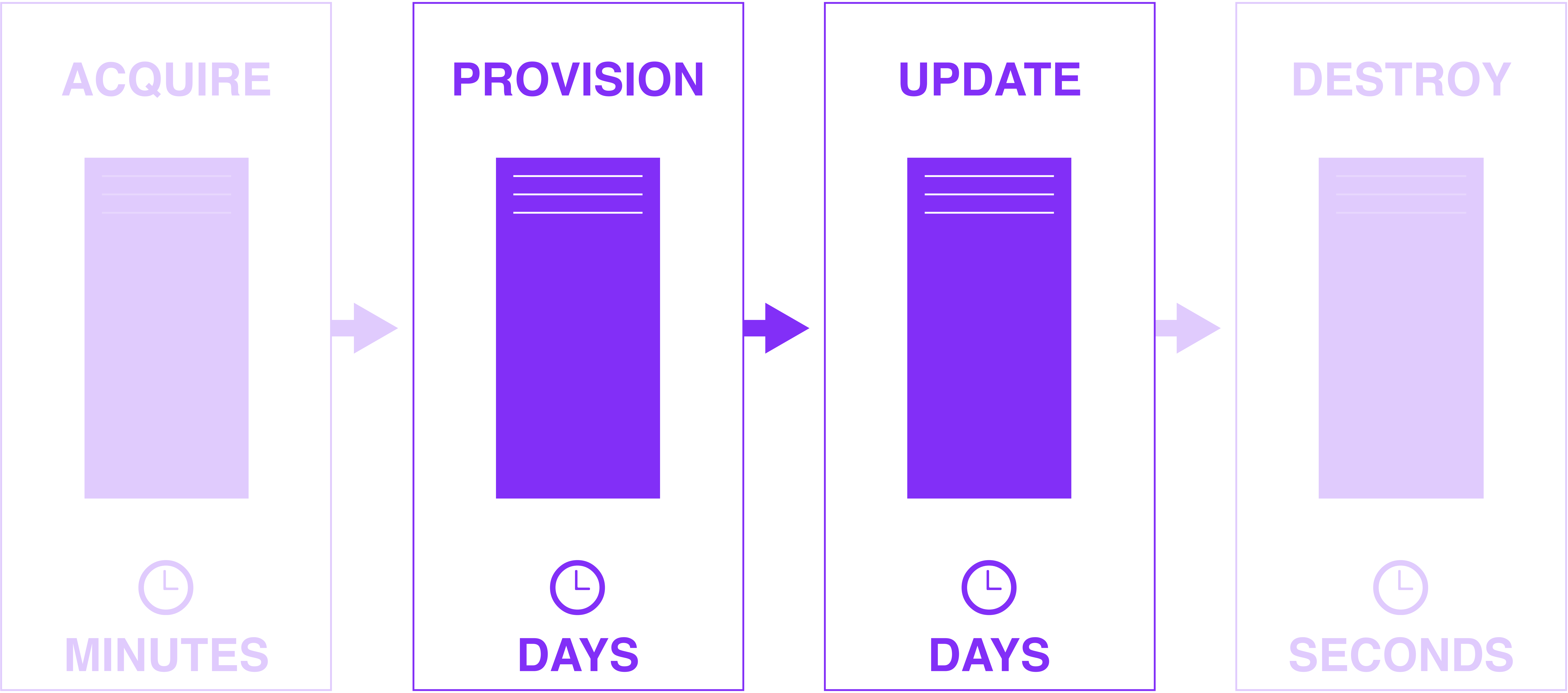


CapEx

OpEx

_aaS

Configuration Management


ACQUIRE

MINUTES

PROVISION

DAYS

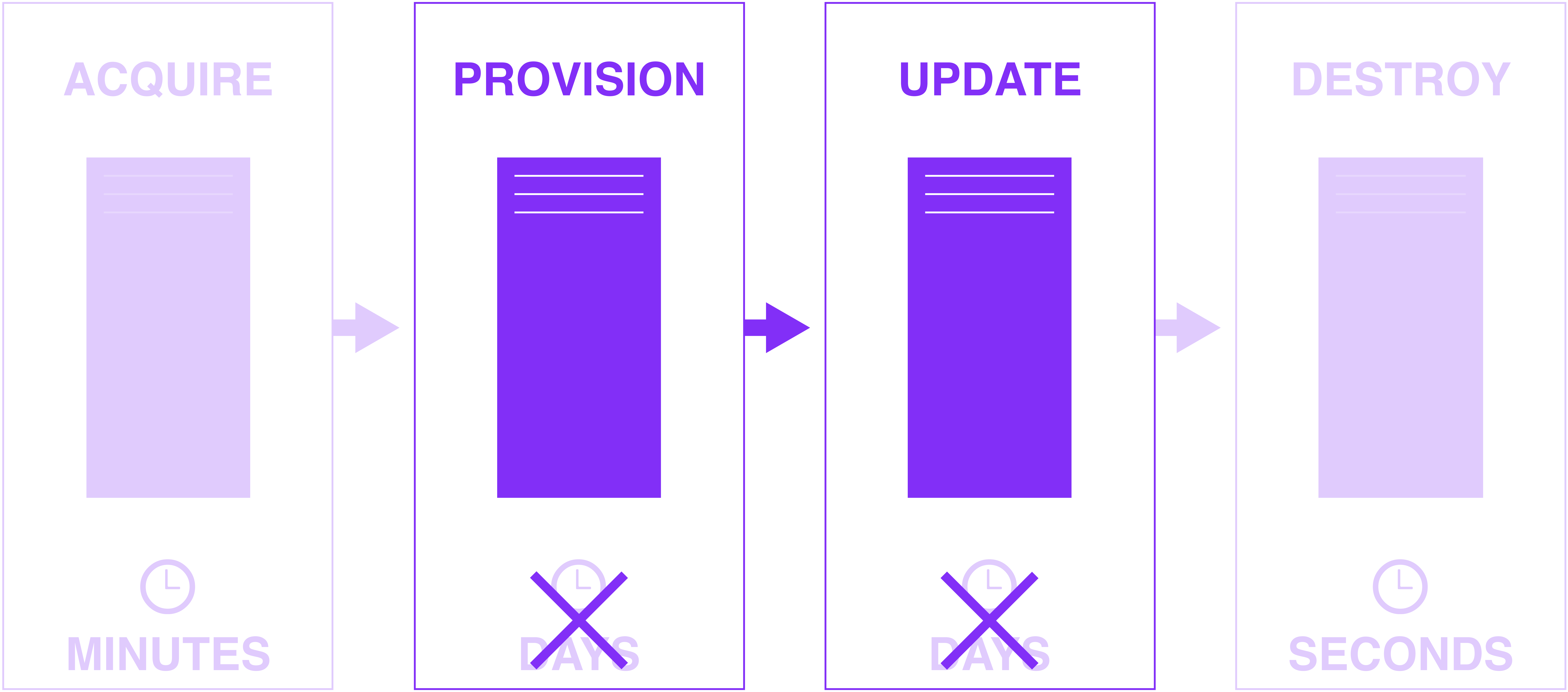
UPDATE

DAYS

DESTROY

SECONDS

Configuration Management



ACQUIRE

MINUTES

PROVISION

~~DAYS~~

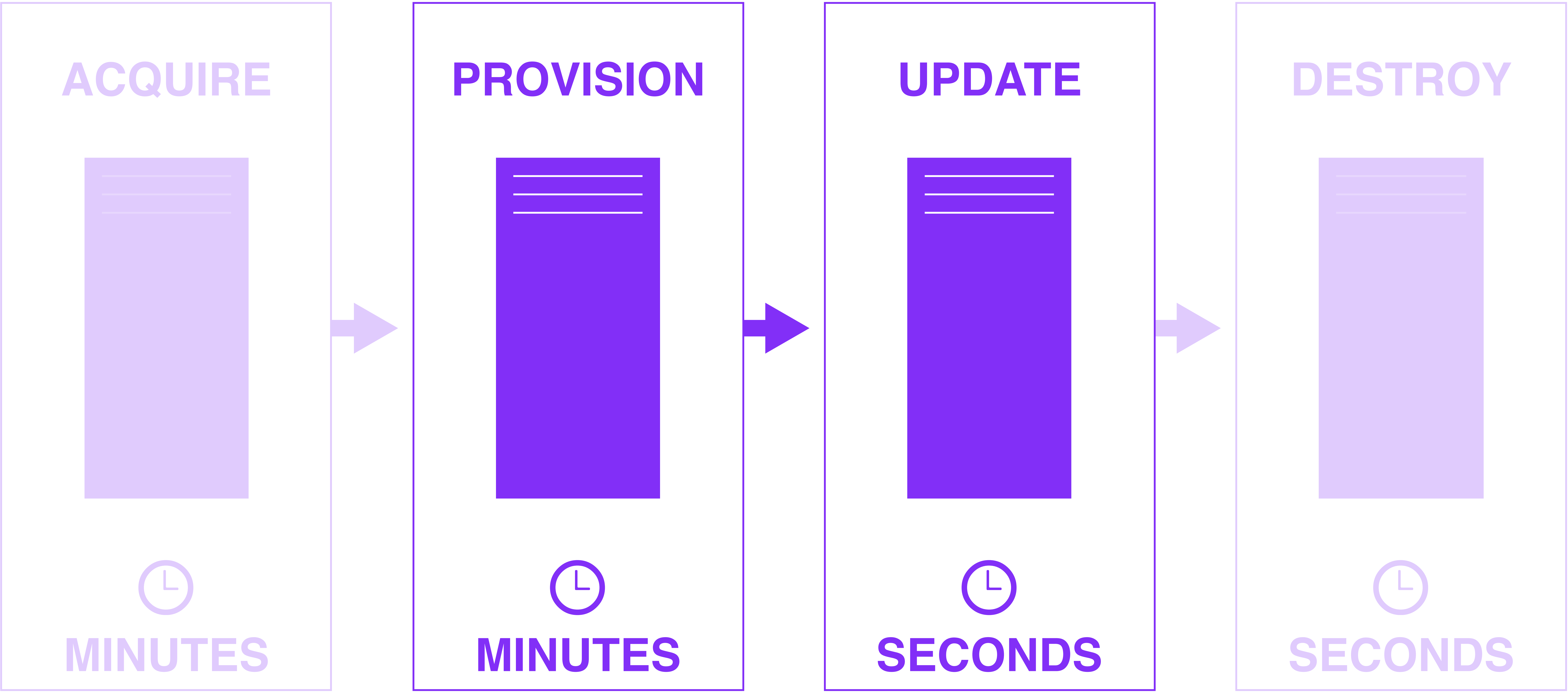
UPDATE

~~DAYS~~

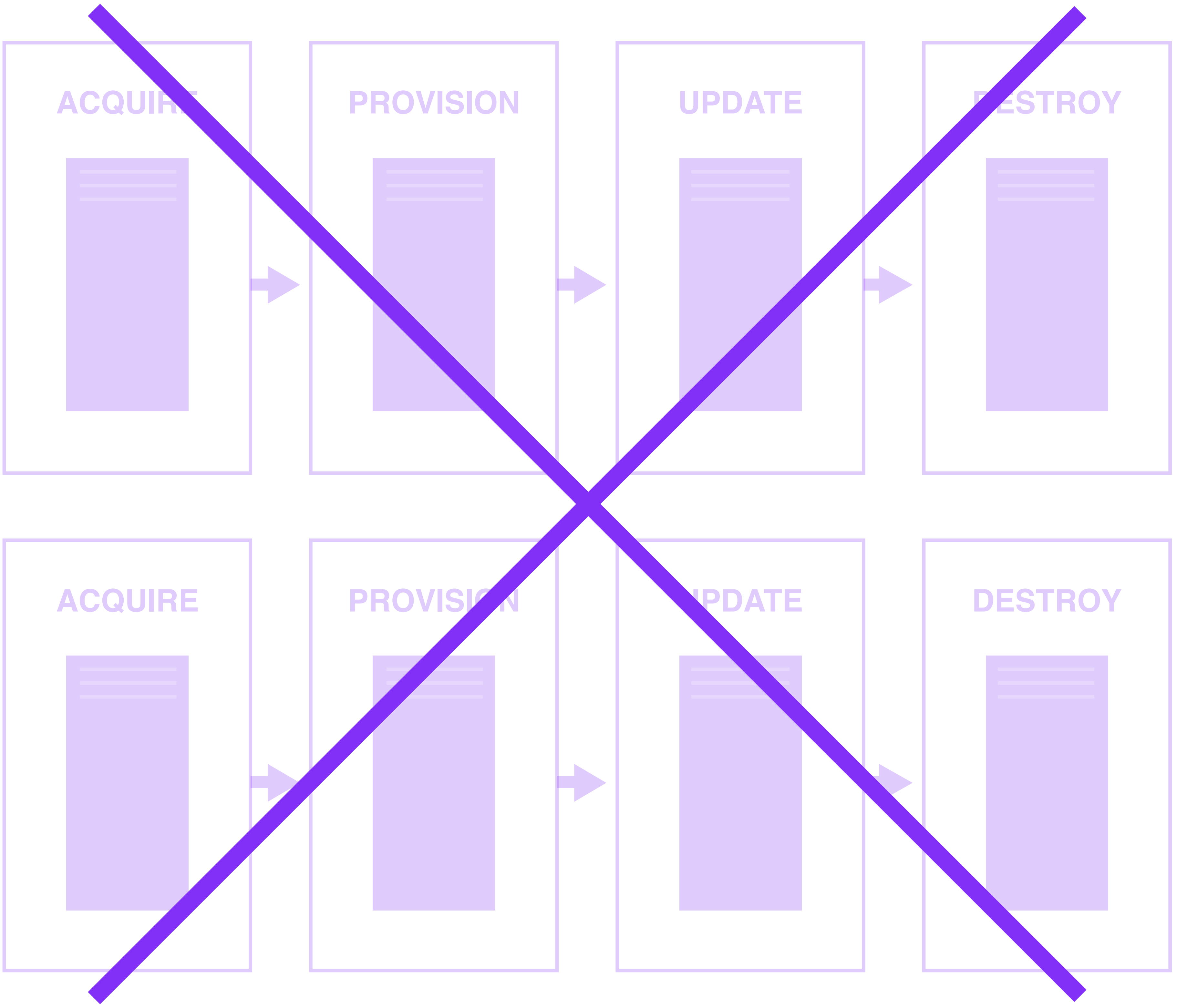
DESTROY

SECONDS

Configuration Management

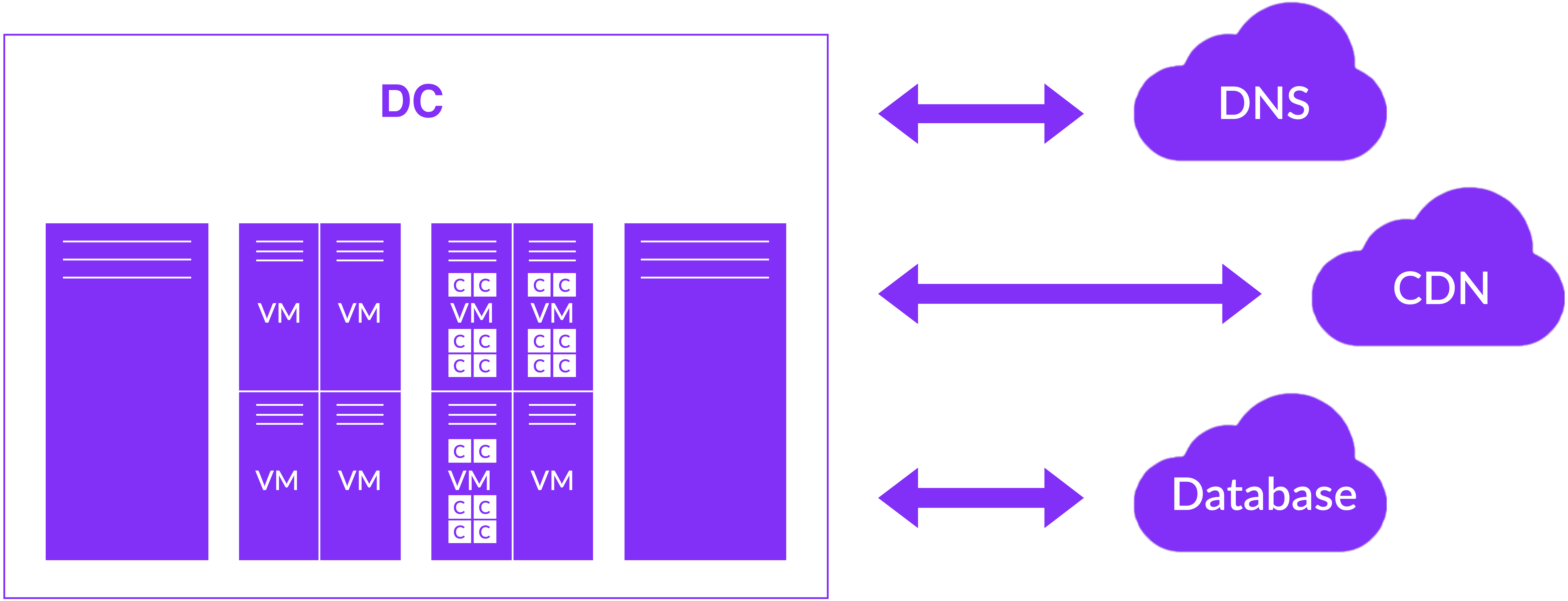


SaaS Proliferation



→ **https://specialized.com**

RISING DATACENTER COMPLEXITY

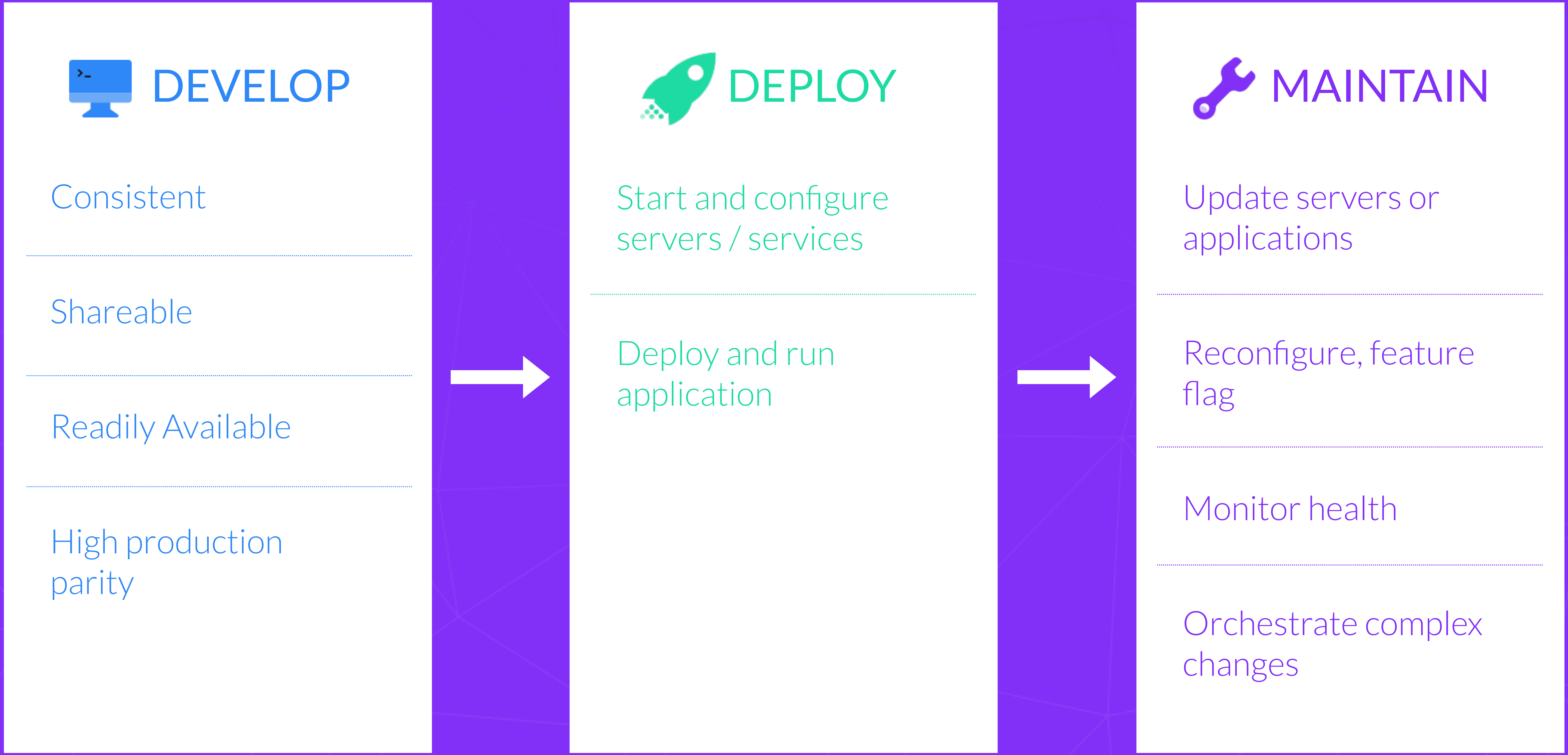


WHY?

What was our original goal?

EFFECTIVELY **DELIVER** AND **MAINTAIN** APPLICATIONS

MOVE **FAST** AND
DON'T BREAK THINGS



DEVELOP

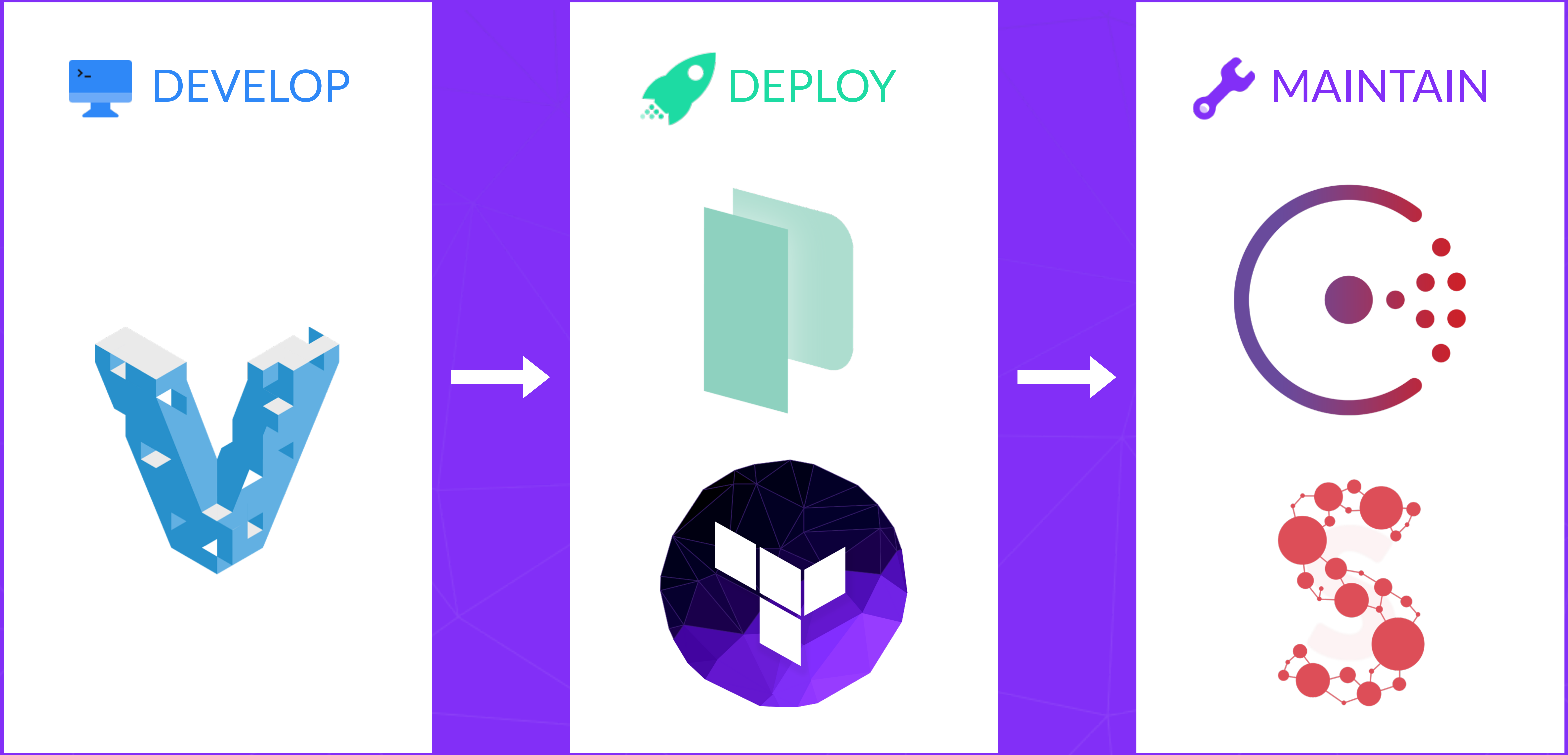
- Consistent
- Shareable
- Readily Available
- High production parity

DEPLOY

- Start and configure servers / services
- Deploy and run application

MAINTAIN

- Update servers or applications
- Reconfigure, feature flag
- Monitor health
- Orchestrate complex changes



DEVELOP

DEPLOY

MAINTAIN

TERRAFORM

Build, Combine, and Launch Infrastructure

MOTIVATION

Why Terraform?

How do I provision resources?

compute?

storage?

network?

How do I manage resource lifecycles?

How do I balance service providers providing core technology for my datacenter?

How do I enforce policy across all these resources?

How do I automate and share those configurations?

TERRAFORM'S **GOAL**

PROVIDE A SINGLE WORKFLOW

WITH A UNIFIED VIEW

USING INFRASTRUCTURE AS CODE

THAT CAN BE ITERATED AND CHANGED SAFELY

CAPABLE OF COMPLEX N-TIER APPLICATIONS

DIGITAL OCEAN DROPLET WITH DNS USING DNS SIMPLE

```
resource "digitalocean_droplet" "web" {
    name = "tf-web"
    size = "512mb"
    image = "centos-5-8-x32"
    region = "sfo1"
}

resource "dnsimple_record" "hello" {
    domain = "example.com"
    name = "test"
    value = "${digitalocean_droplet.web.ipv4_address}"
    type = "A"
}
```

DIGITAL OCEAN DROPLET WITH DNS USING DNS SIMPLE

```
resource "digitalocean_droplet" "web" {
    name = "tf-web"
    size = "512mb"
    image = "centos-5-8-x32"
    region = "sfo1"
}

resource "dnsimple_record" "hello" {
    domain = "example.com"
    name = "test"
    value = "${digitalocean_droplet.web.ipv4_address}"
    type = "A"
}
```

DIGITAL OCEAN DROPLET WITH DNS USING DNS SIMPLE

```
resource "digitalocean_droplet" "web" {
    name = "tf-web"
    size = "512mb"
    image = "centos-5-8-x32"
    region = "sfo1"
}

resource "dnsimple_record" "hello" {
    domain = "example.com"
    name = "test"
    value = "${digitalocean_droplet.web.ipv4_address}"
    type = "A"
}
```

DIGITAL OCEAN DROPLET WITH DNS USING DNS SIMPLE

```
resource "digitalocean_droplet" "web" {
    name = "tf-web"
    size = "512mb"
    image = "centos-5-8-x32"
    region = "sfo1"
}

resource "dnsimple_record" "hello" {
    domain = "example.com"
    name = "test"
    value = "${digitalocean_droplet.web.ipv4_address}"
    type = "A"
}
```

HUMAN-FRIENDLY CONFIG*

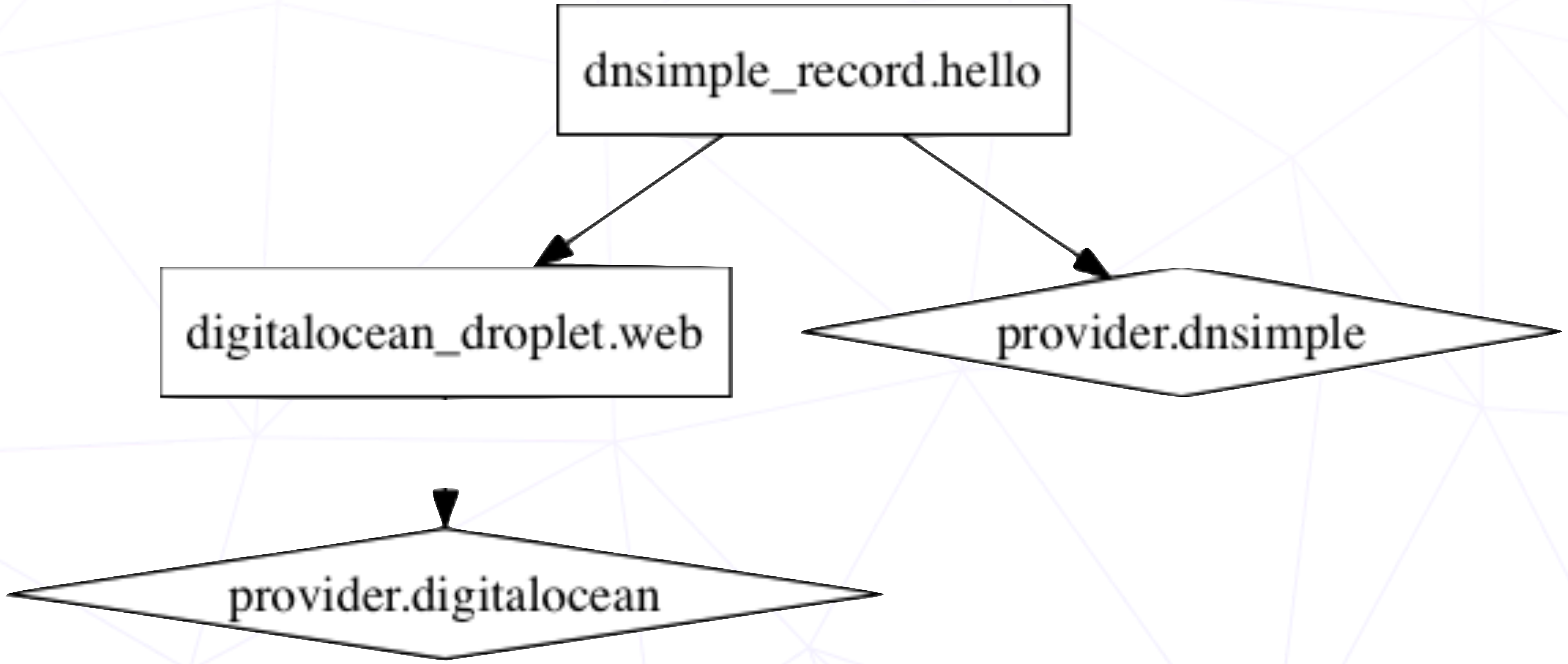
* JSON-COMPATIBLE FOR NON-HUMANS

VCS-FRIENDLY FORMAT

ENTIRE INFRASTRUCTURE…
IN A SINGLE TEXT FILE

TERRAFORM **GRAPH**

COMPOSE N-TIERS / PROVIDERS



TERRAFORM **PROVIDERS**

SINGLE INTEGRATION POINT

EXPOSE ("PROVIDE") A RESOURCE

EXPOSE ("PROVIDE") A RESOURCE

`aws_instance`

`dnssimple_record`

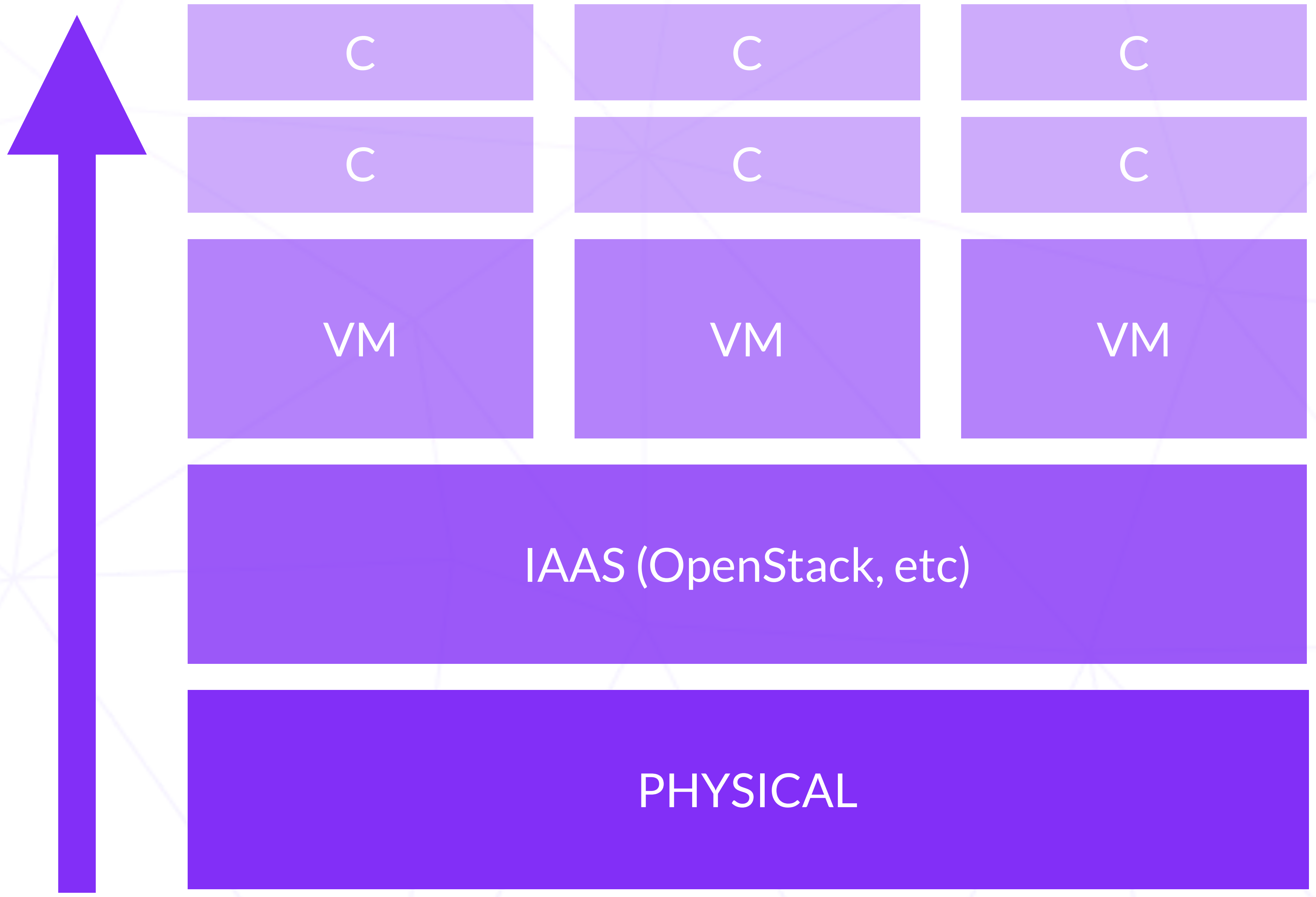
CRUD API

PLUGGABLE FOR INTEGRATIONS

"LAYER CAKE"

PROVIDER PER LAYER

UNIFIED CONFIGURATION



```
$ terraform apply
```

All logos and trademarks are copyright their respective owners

TERRAFORM **PLAN**

$ terraform plan

```
+ digitalocean_droplet.web
    backups:                "" => "<computed>"
    image:                  "" => "centos-5-8-x32"
    ipv4_address:           "" => "<computed>"
    ipv4_address_private:   "" => "<computed>"
    name:                   "" => "tf-web"
    private_networking:     "" => "<computed>"
    region:                 "" => "sfo1"
    size:                   "" => "512mb"
    status:                 "" => "<computed>"

+ dnsimple_record.hello
    domain:     "" => "example.com"
    domain_id:  "" => "<computed>"
    hostname:   "" => "<computed>"
    name:       "" => "test"
    priority:   "" => "<computed>"
```

$ terraform plan

```
+ digitalocean_droplet.web
    backups:                ""  =>  "<computed>"
    image:                  ""  =>  "centos-5-8-x32"
    ipv4_address:           ""  =>  "<computed>"
    ipv4_address_private:   ""  =>  "<computed>"
    name:                   ""  =>  "tf-web"
    private_networking:     ""  =>  "<computed>"
    region:                 ""  =>  "sfo1"
    size:                   ""  =>  "512mb"
    status:                 ""  =>  "<computed>"

+ dnsimple_record.hello
    domain:     ""  =>  "example.com"
    domain_id:  ""  =>  "<computed>"
    hostname:   ""  =>  "<computed>"
    name:       ""  =>  "test"
    priority:   ""  =>  "<computed>"
```

```
$ terraform plan
```

```
+ digitalocean_droplet.web
    backups:              "" => "<computed>"
    image:                "" => "centos-5-8-x32"
    ipv4_address:         "" => "<computed>"
    ipv4_address_private: "" => "<computed>"
    name:                 "" => "tf-web"
    private_networking:   "" => "<computed>"
    region:               "" => "sfo1"
    size:                 "" => "512mb"
    status:               "" => "<computed>"

+ dnsimple_record.hello
    domain:    "" => "example.com"
    domain_id: "" => "<computed>"
    hostname:  "" => "<computed>"
    name:      "" => "test"
    priority:  "" => "<computed>"
```

$ terraform plan

```
+ digitalocean_droplet.web
    backups:               "" => "<computed>"
    image:                 "" => "centos-5-8-x32"
    ipv4_address:          "" => "<computed>"
    ipv4_address_private:  "" => "<computed>"
    name:                  "" => "tf-web"
    private_networking:    "" => "<computed>"
    region:                "" => "sfo1"
    size:                  "" => "512mb"
    status:                "" => "<computed>"

+ dnsimple_record.hello
    domain:    "" => "example.com"
    domain_id: "" => "<computed>"
    hostname:  "" => "<computed>"
    name:      "" => "test"
    priority:  "" => "<computed>"
    backups:              "" => "<computed>"
    image:                "" => "centos-5-8-x32"
    ipv4_address:         "" => "<computed>"
    ipv4_address_private: "" => "<computed>"
    name:                 "" => "tf-web"
    private_networking:   "" => "<computed>"
    region:               "" => "sfo1"
    size:                 "" => "512mb"
    status:               "" => "<computed>"

+ dnsimple_record.hello
    domain:    "" => "example.com"
    domain_id: "" => "<computed>"
    hostname:  "" => "<computed>"
    name:      "" => "test"
    priority:  "" => "<computed>"
    ttl:       "" => "<computed>"
    type:      "" => "A"
    value:     "" => "${digitalocean_droplet.web.ipv4_address}"
    backups:               "" => "<computed>"
    image:                 "" => "centos-5-8-x32"
    ipv4_address:          "" => "<computed>"
    ipv4_address_private:  "" => "<computed>"
    name:                  "" => "tf-web"
    private_networking:    "" => "<computed>"
    region:                "" => "sfo1"
    size:                  "" => "512mb"
    status:                "" => "<computed>"

+ dnsimple_record.hello
    domain:     "" => "example.com"
    domain_id:  "" => "<computed>"
    hostname:   "" => "<computed>"
    name:       "" => "test"
    priority:   "" => "<computed>"
    ttl:        "" => "<computed>"
    type:       "" => "A"
    value:      "" => "${digitalocean_droplet.web.ipv4_address}"
```

SHOWS YOU WHAT WILL HAPPEN

EXPLAINS CERTAIN ACTIONS

PREVIOUSLY?

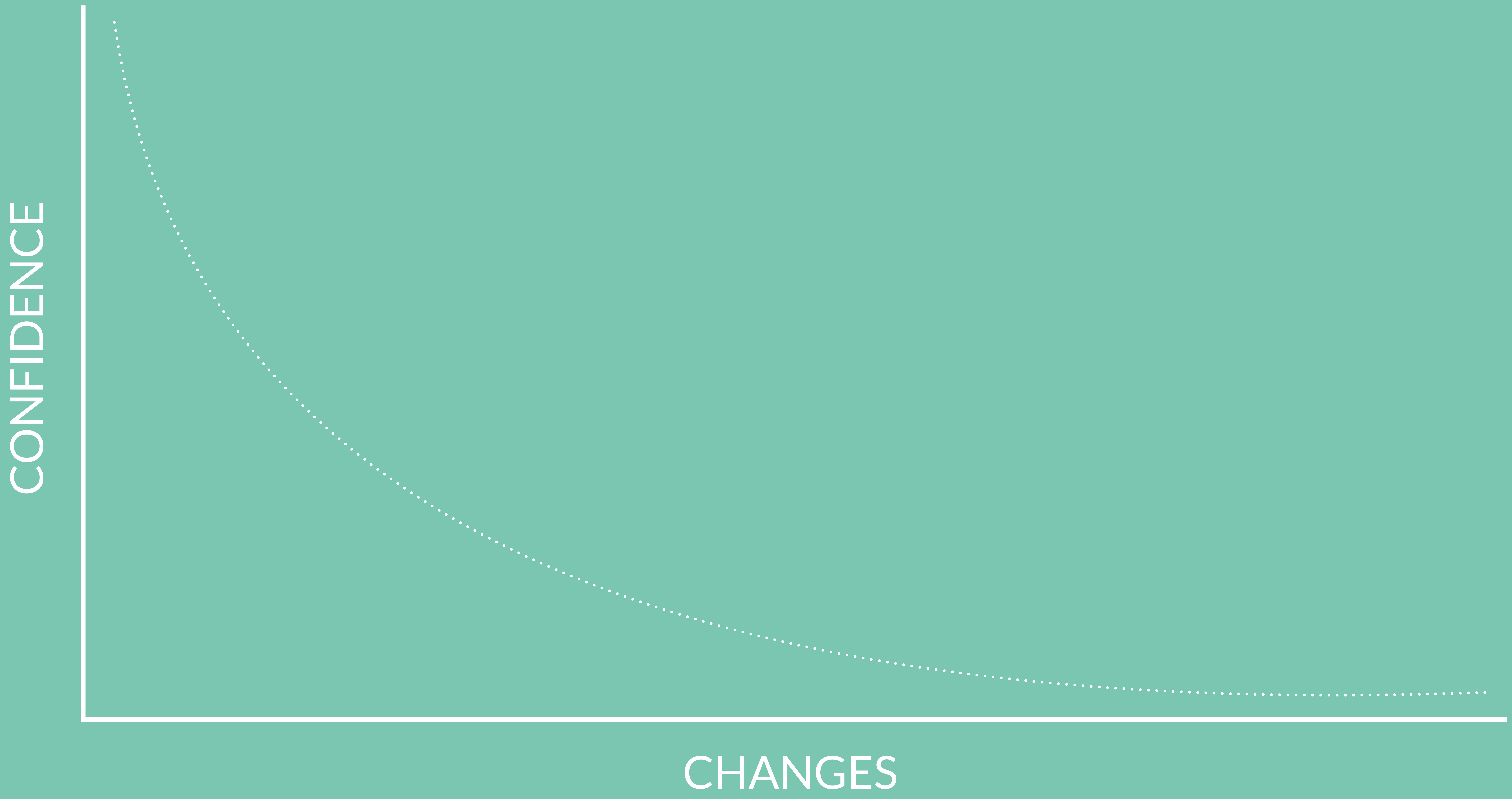
STILL UNCERTAINTY…

FUTURE OPS

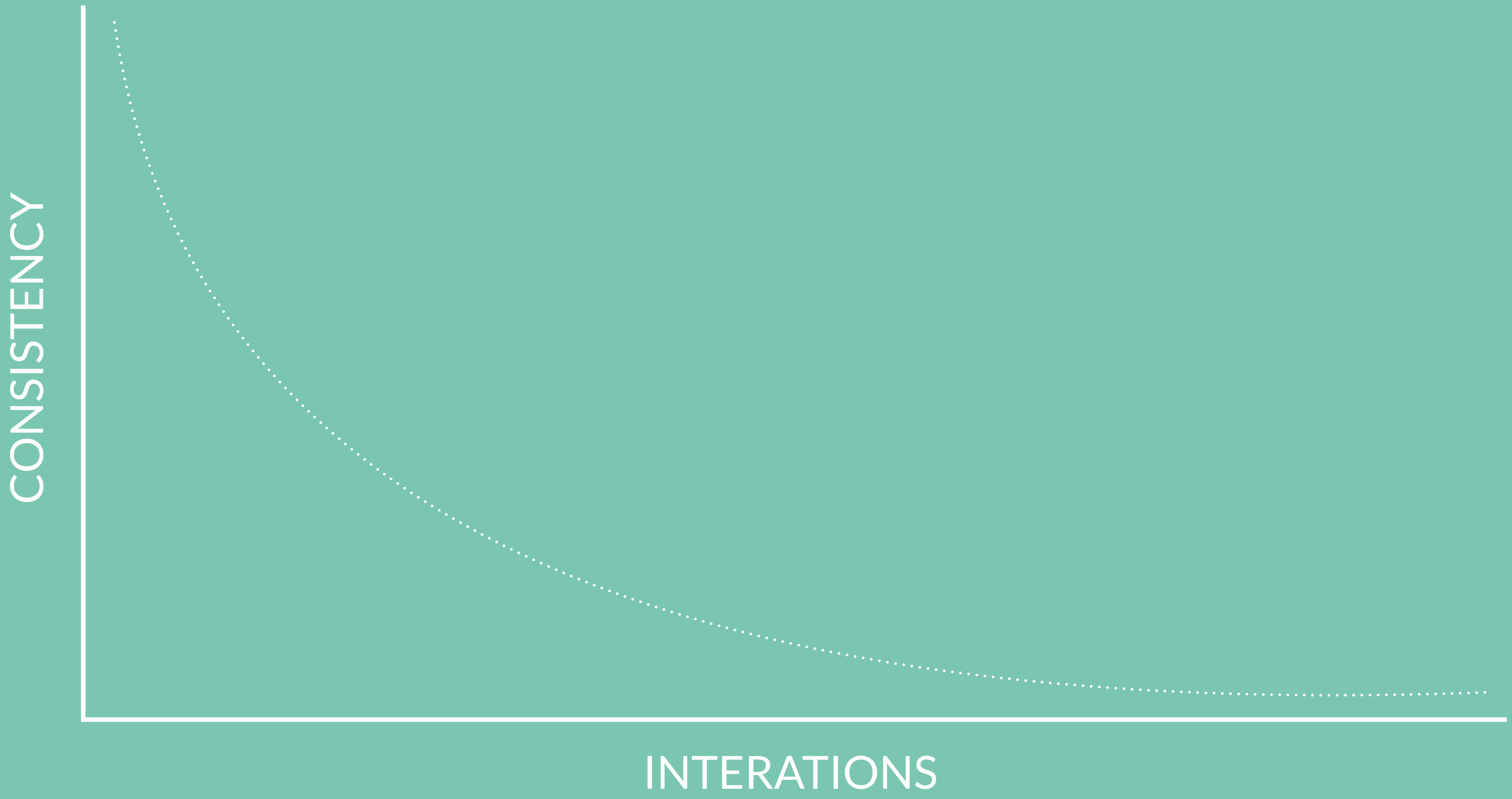
Managing tomorrow’s infrastructure

DEPLOY **IMMUTABLE** INFRASTRUCTURE

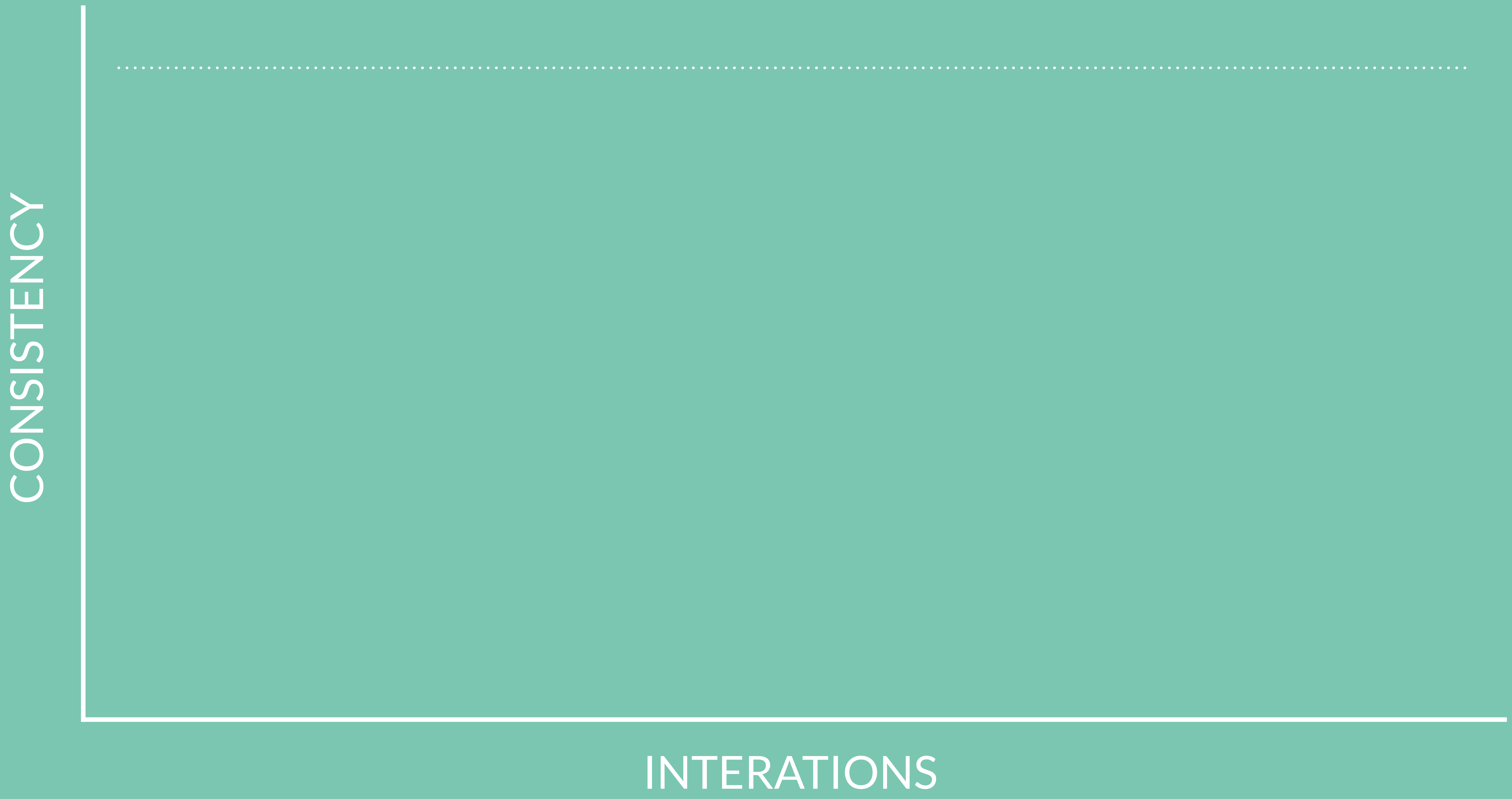
Mutable Infrastructure



Mutable Infrastructure



Immutable Infrastructure



IMMUTABLE INFRASTRUCTURE IS FASTER

IMMUTABLE
INFRASTRUCTURE
ALLOWS FOR
GREATER PARITY

IMMUTABLE INFRASTRUCTURE NEEDS **AUTOMATION**

PACKER

Automated Golden Images (and more)

MACHINE IMAGES

YUCK… IMAGES?

WHY HAVE WE BEEN **GENERALLY AGAINST** MACHINE IMAGES?

GOLDEN IMAGES USED TO BE THE WAY

QUARTERLY, UNCHANGED, AND **BLESSED** IMAGE

CHANGES WERE

SLOW AND FRUSTRATING

TOOLING WAS

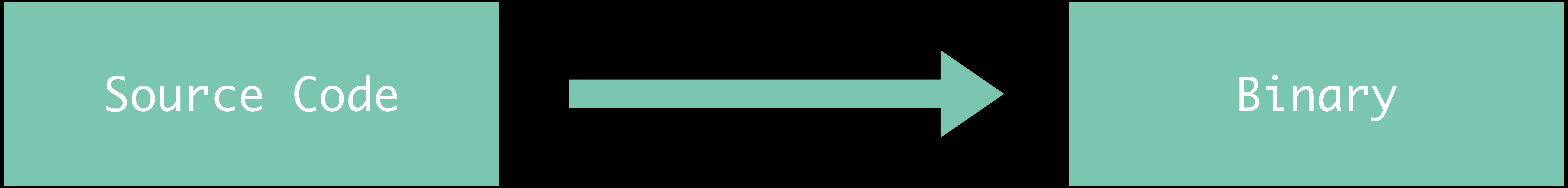
NOT MATURE

COMPARED TO TODAY

MODERN **CONFIG MANAGEMENT** CHANGED THAT

OPS WITHOUT **MACHINE IMAGES** IS LIKE APPLICATIONS WITHOUT **BINARIES**

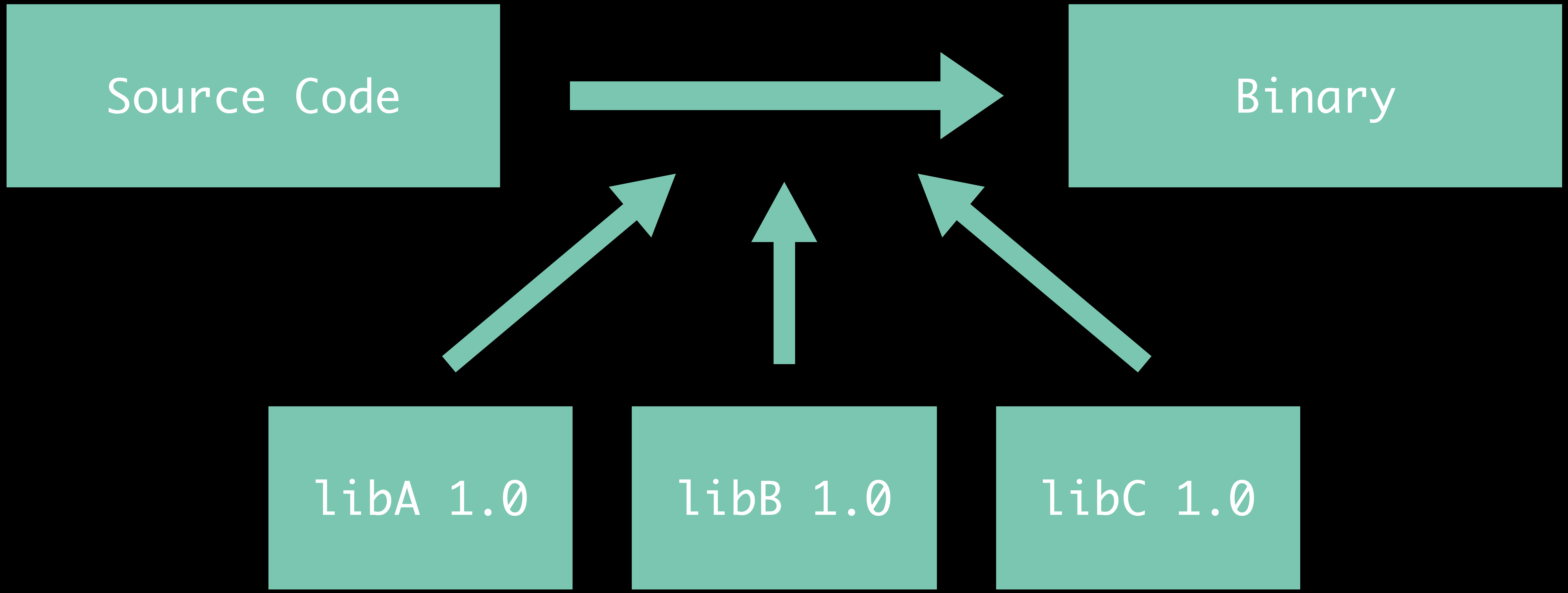
Application Lifecycle



Source Code

Binary

Application Lifecycle


Source Code

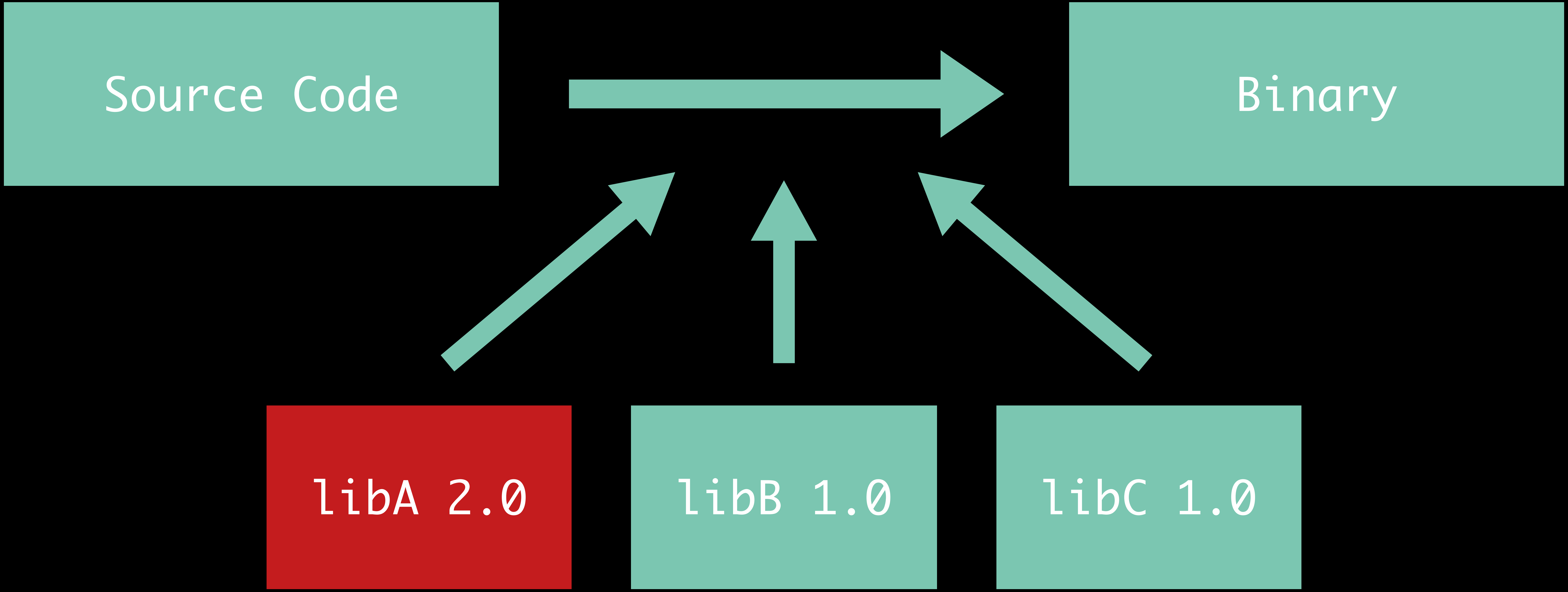
Binary

libA 1.0

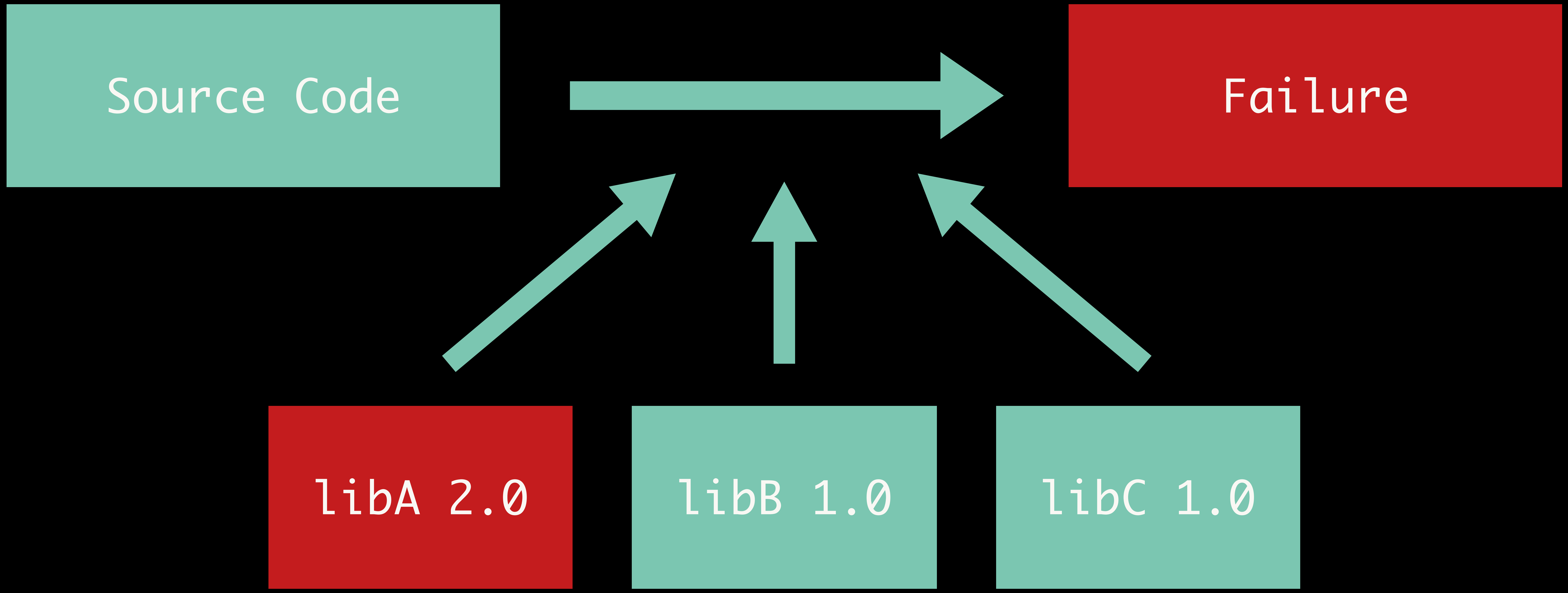
libB 1.0

libC 1.0

Application Lifecycle

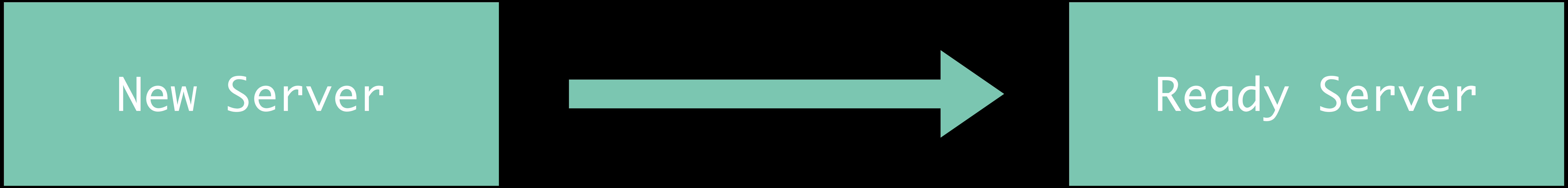


Application Lifecycle



Application Lifecycle

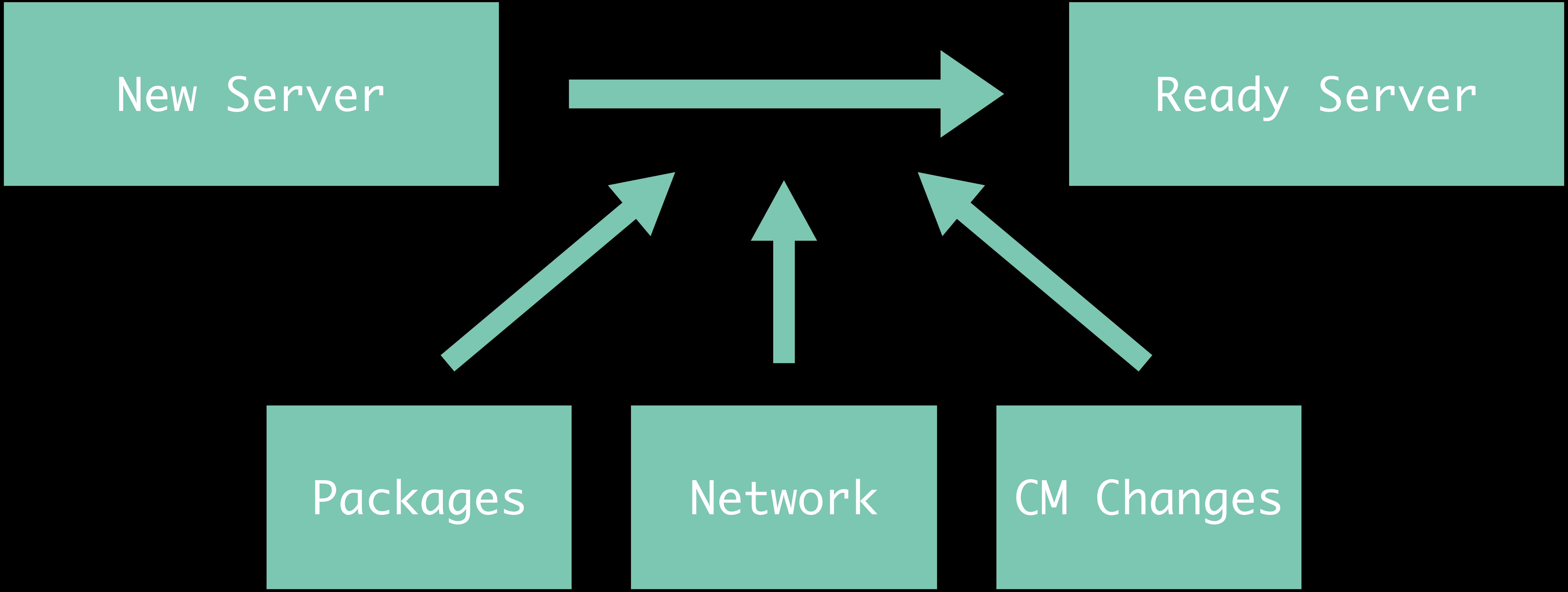
Mutable Server Lifecycle



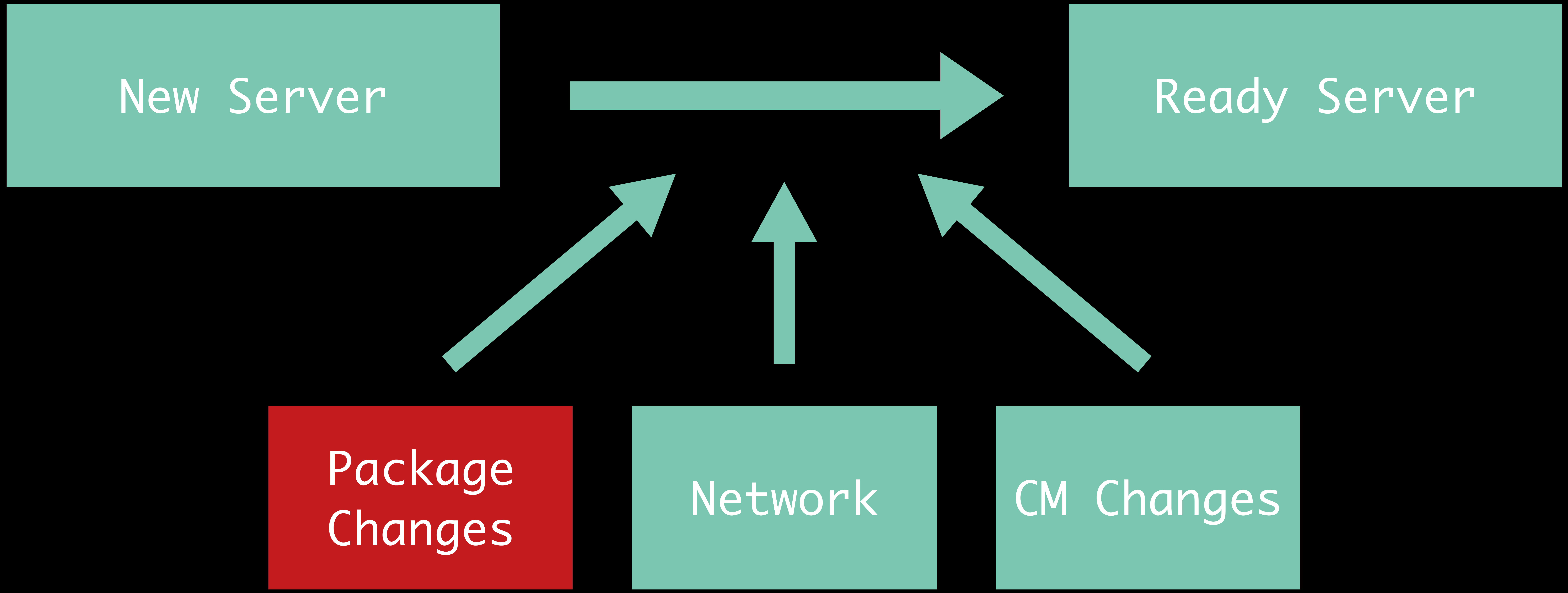
New Server

Ready Server

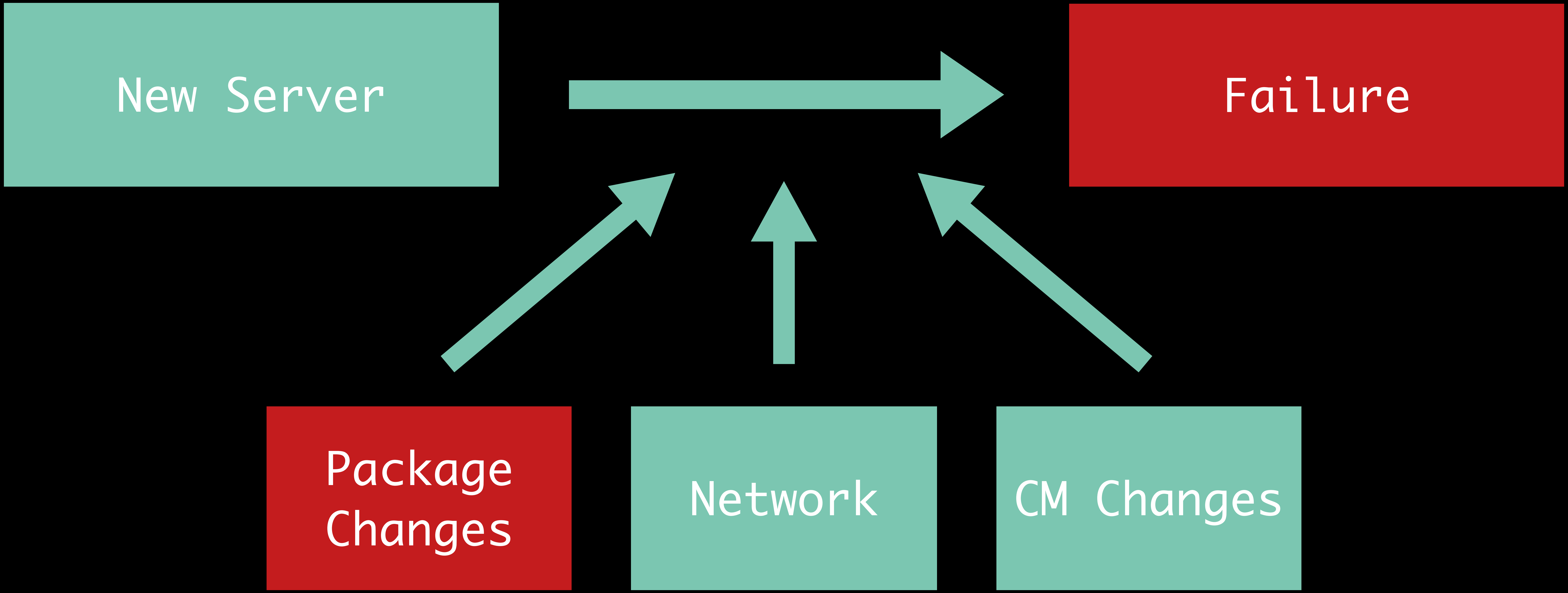
Mutable Server Lifecycle



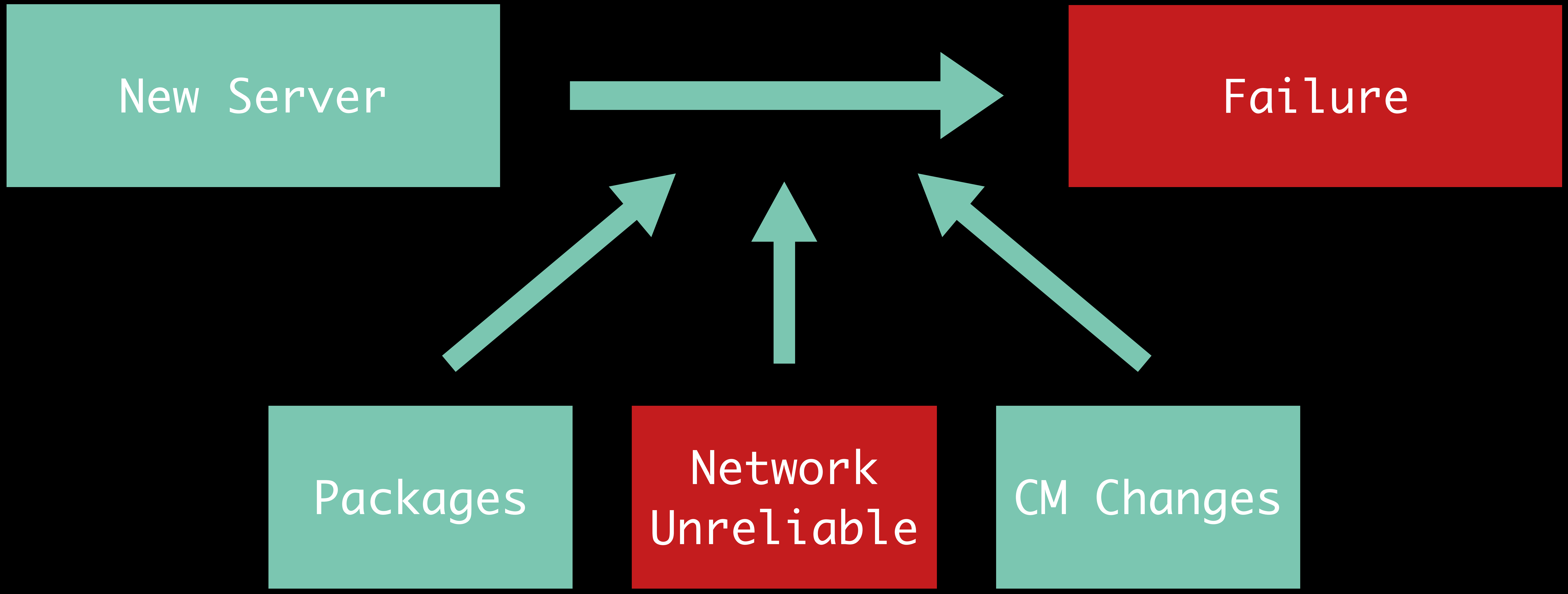
Mutable Server Lifecycle



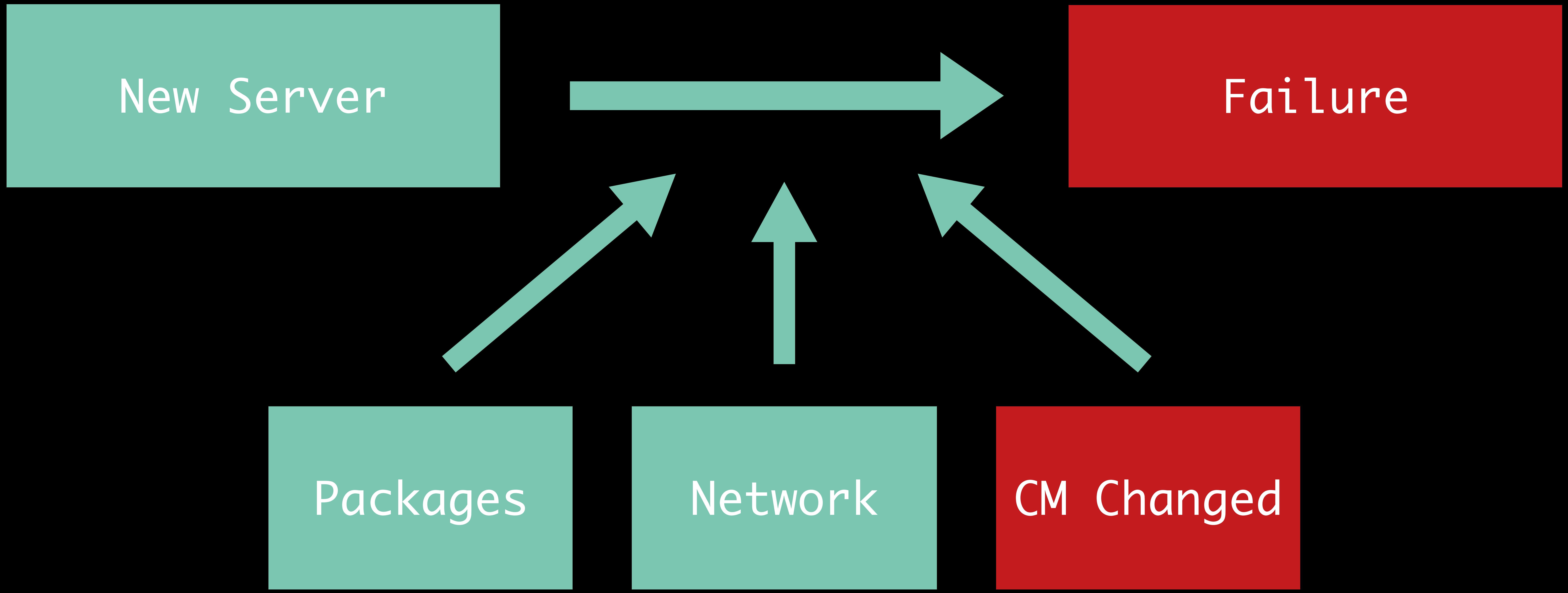
Mutable Server Lifecycle



Mutable Server Lifecycle



Mutable Server Lifecycle

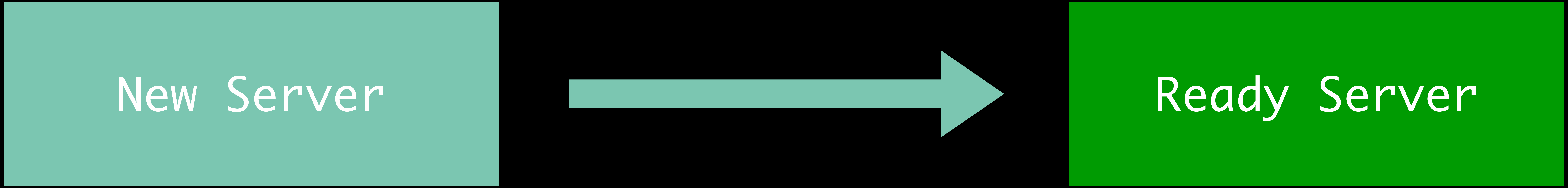


Mutable Server Lifecycle

Machine Image Lifecycle

New Server
Ready Server

Machine Image Lifecycle



Machine Image Lifecycle

PACKER **BUILD**

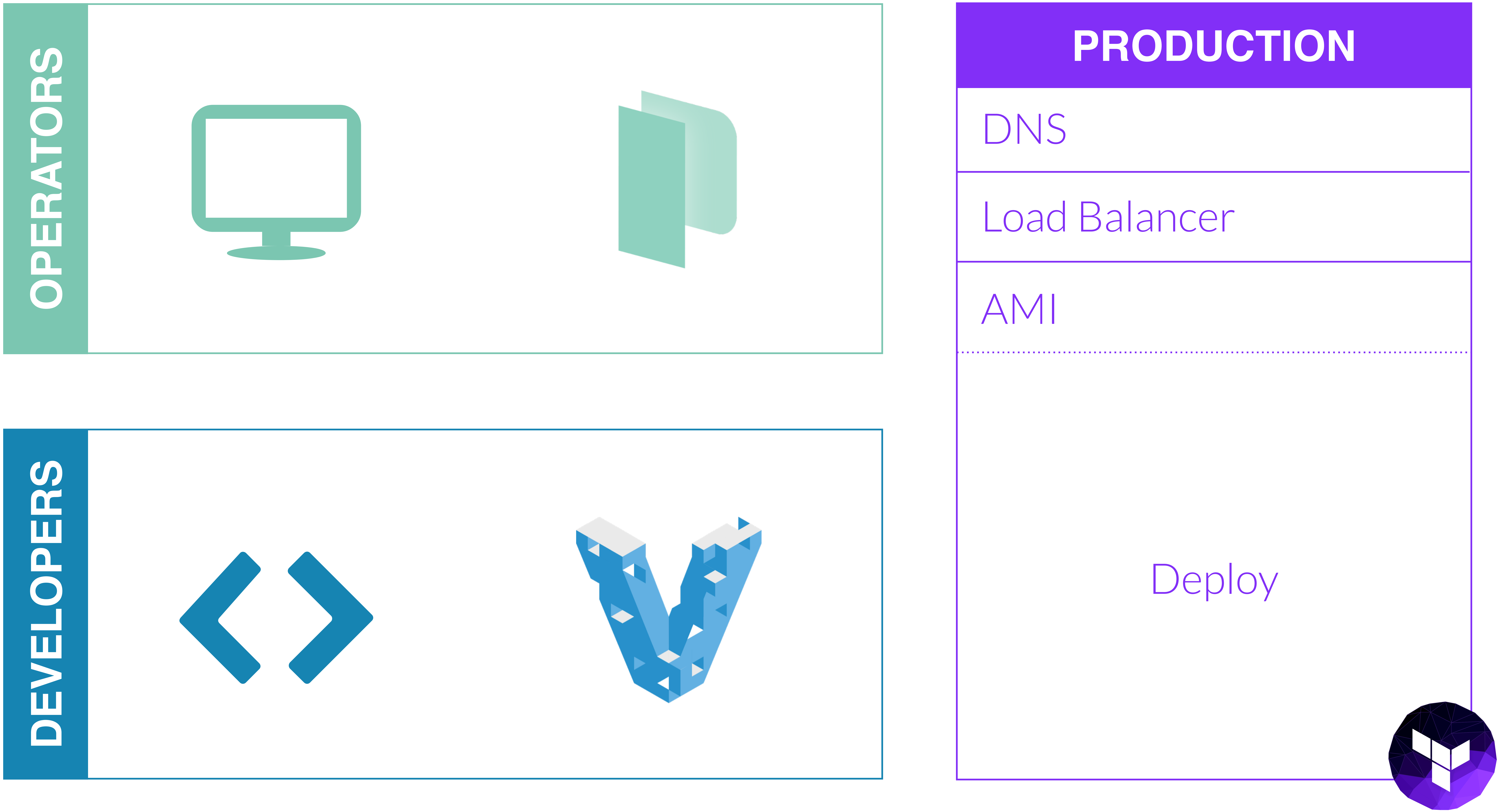
EMBRACES CONFIG MANAGEMENT

TRANSITIONS FAILURES FROM RUN-TIME TO BUILD-TIME

ENFORCES PARITY WITH STAGING

... AND EVEN DEVELOPMENT

WHAT DOES **THE FUTURE** LOOK LIKE?



OPERATORS

DEVELOPERS

PRODUCTION

DNS

Load Balancer

AMI

Deploy

OPERATORS

DEVELOPERS

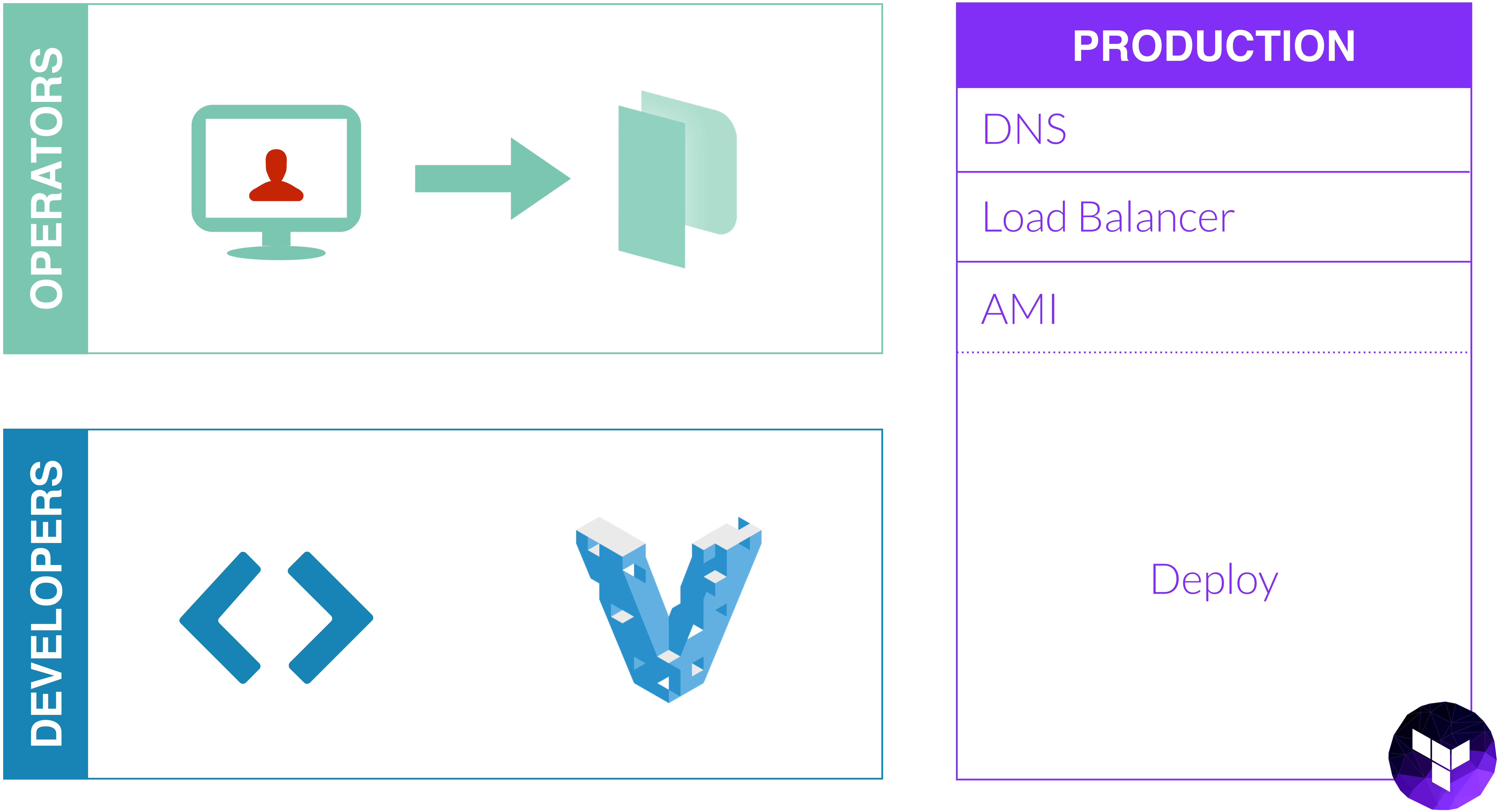
PRODUCTION

DNS

Load Balancer

AMI

Deploy



OPERATORS

DEVELOPERS

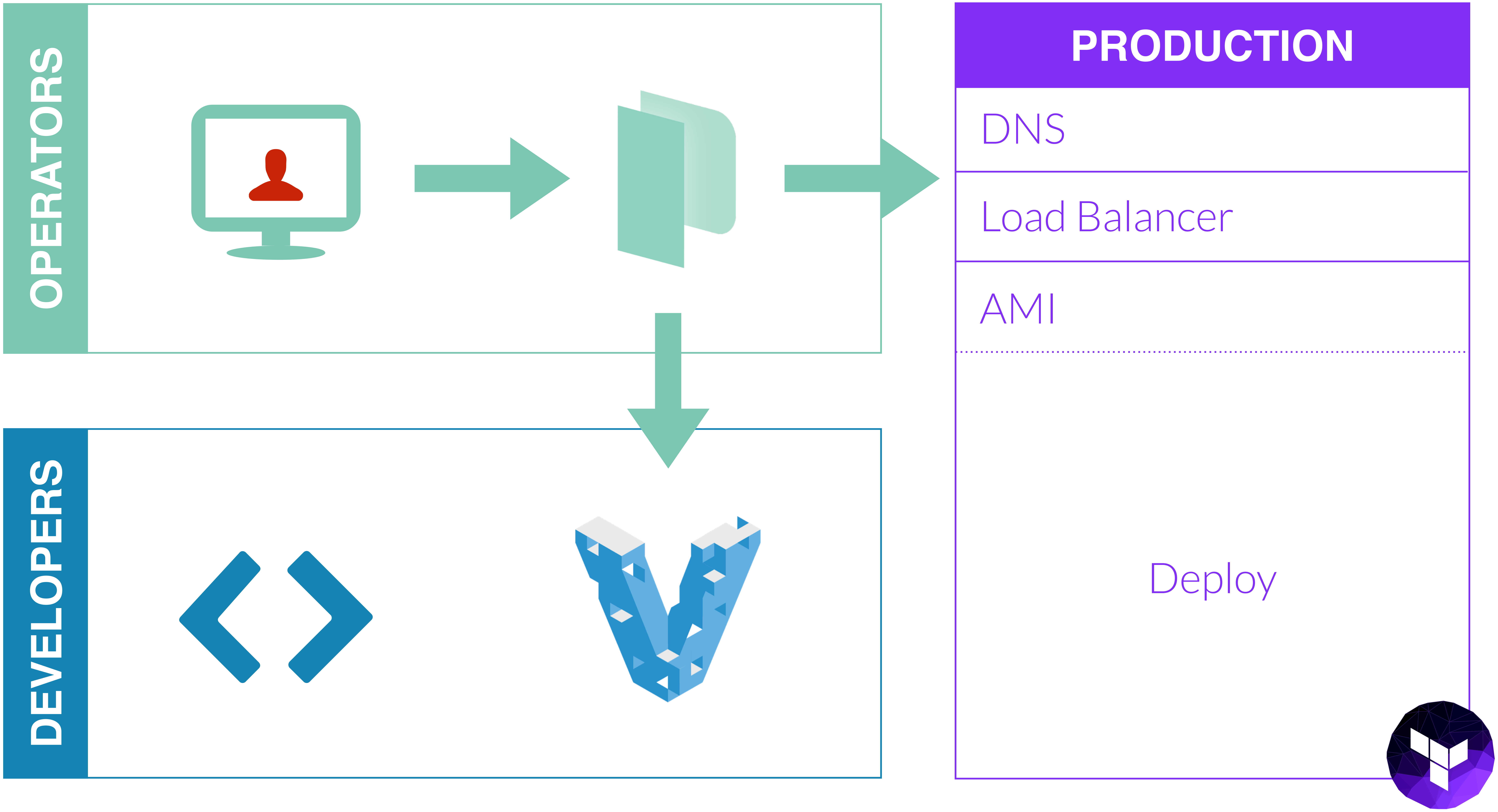
PRODUCTION

DNS

Load Balancer

AMI

Deploy



OPERATORS

DEVELOPERS

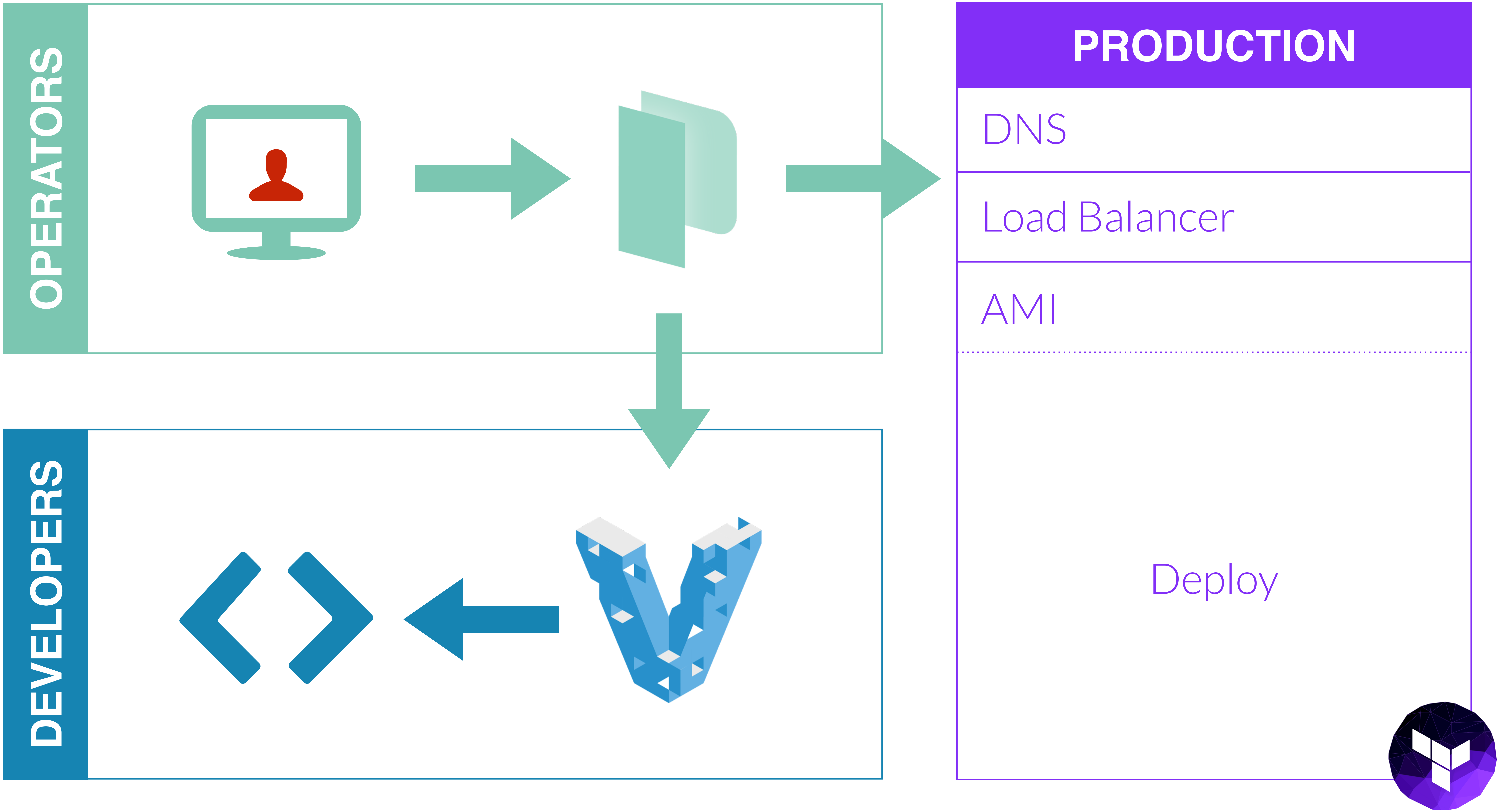
PRODUCTION

DNS

Load Balancer

AMI

Deploy



OPERATORS

DEVELOPERS

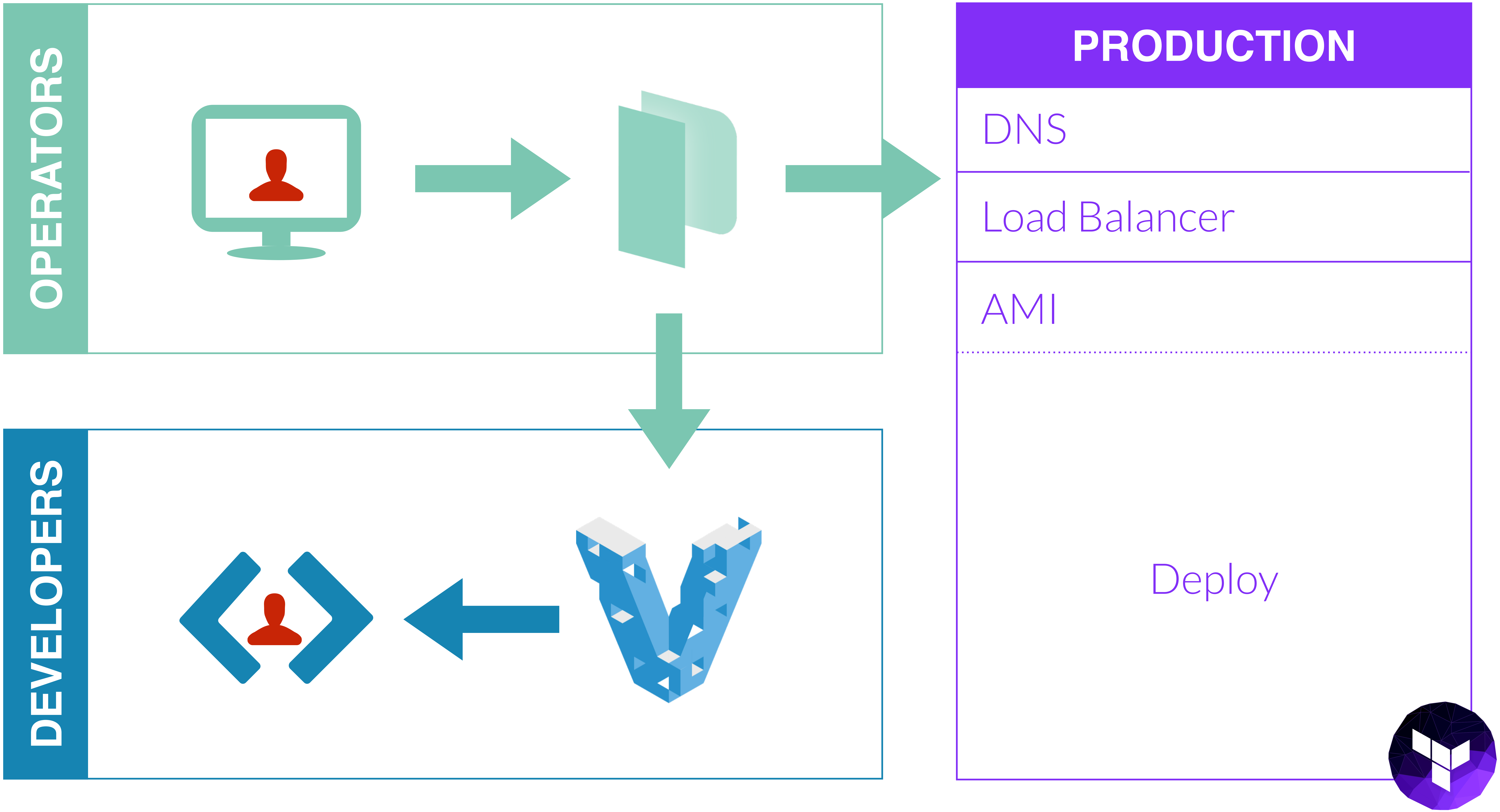
PRODUCTION

DNS

Load Balancer

AMI

Deploy



OPERATORS

DEVELOPERS

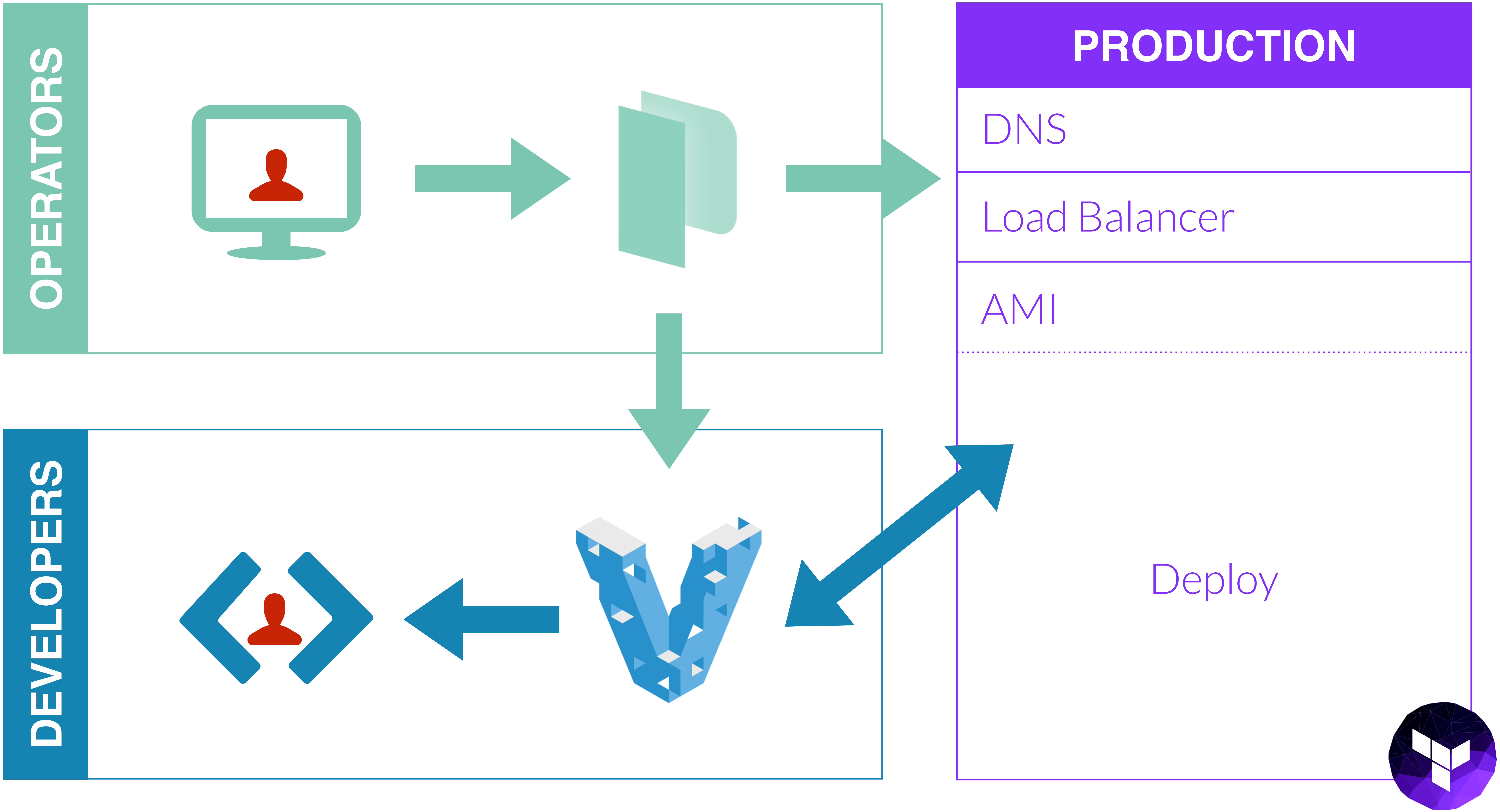
PRODUCTION

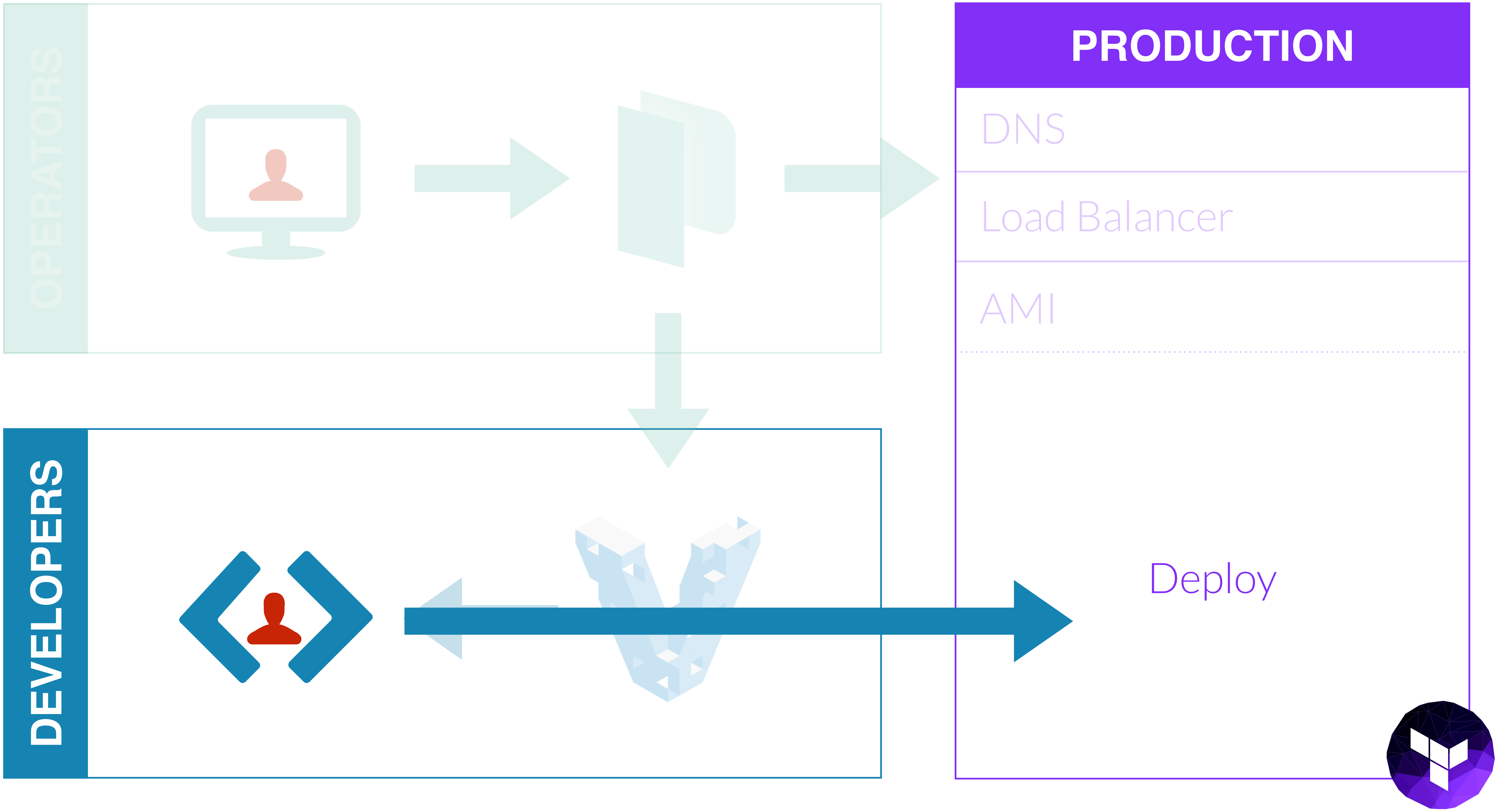
DNS

Load Balancer

AMI

Deploy


OPERATORS
DEVELOPERS
PRODUCTION
DNS
Load Balancer
AMI
Deploy



OPERATORS

DEVELOPERS

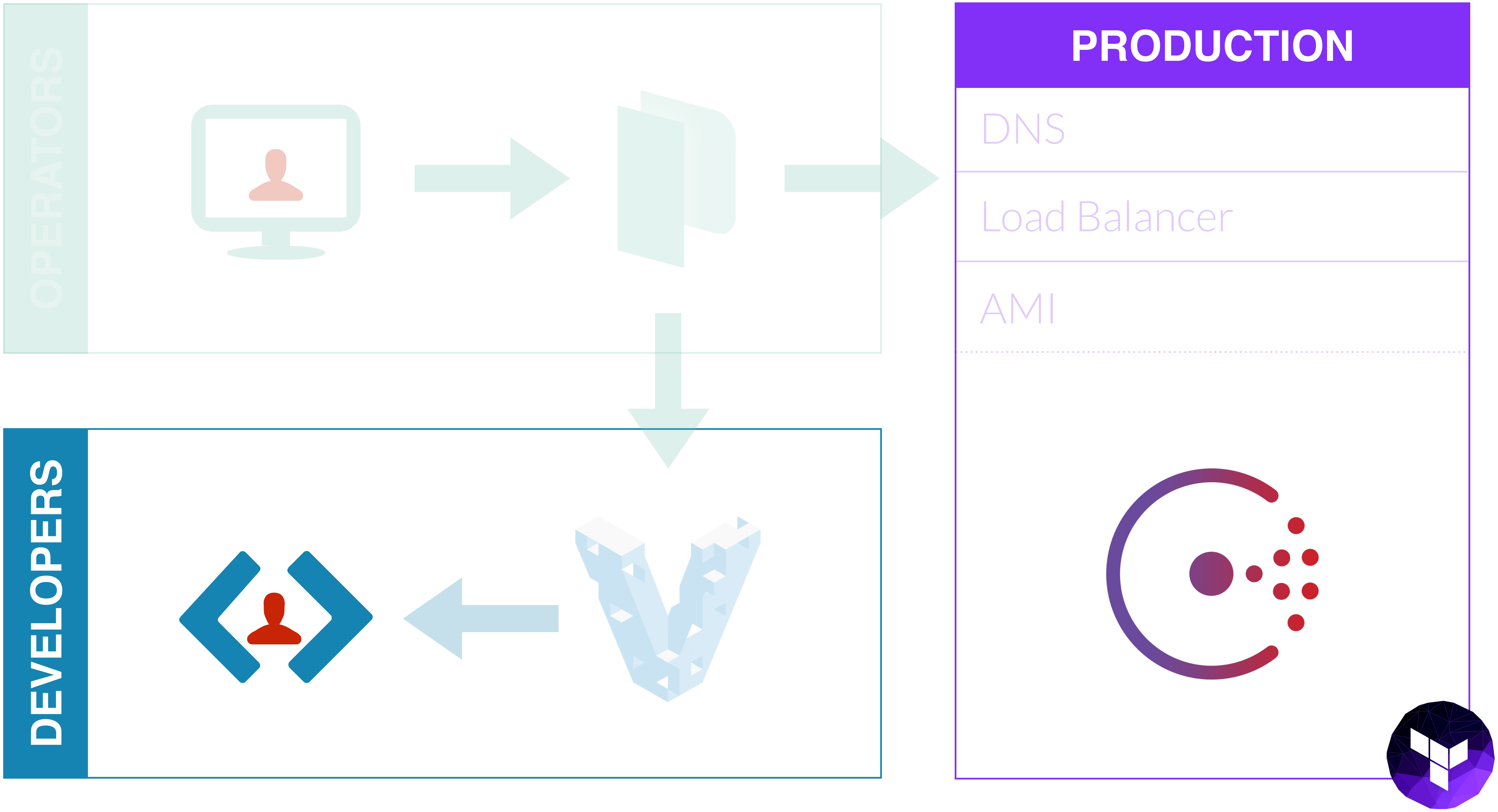
PRODUCTION

DNS

Load Balancer

AMI

Deploy



OPERATORS

DEVELOPERS

PRODUCTION

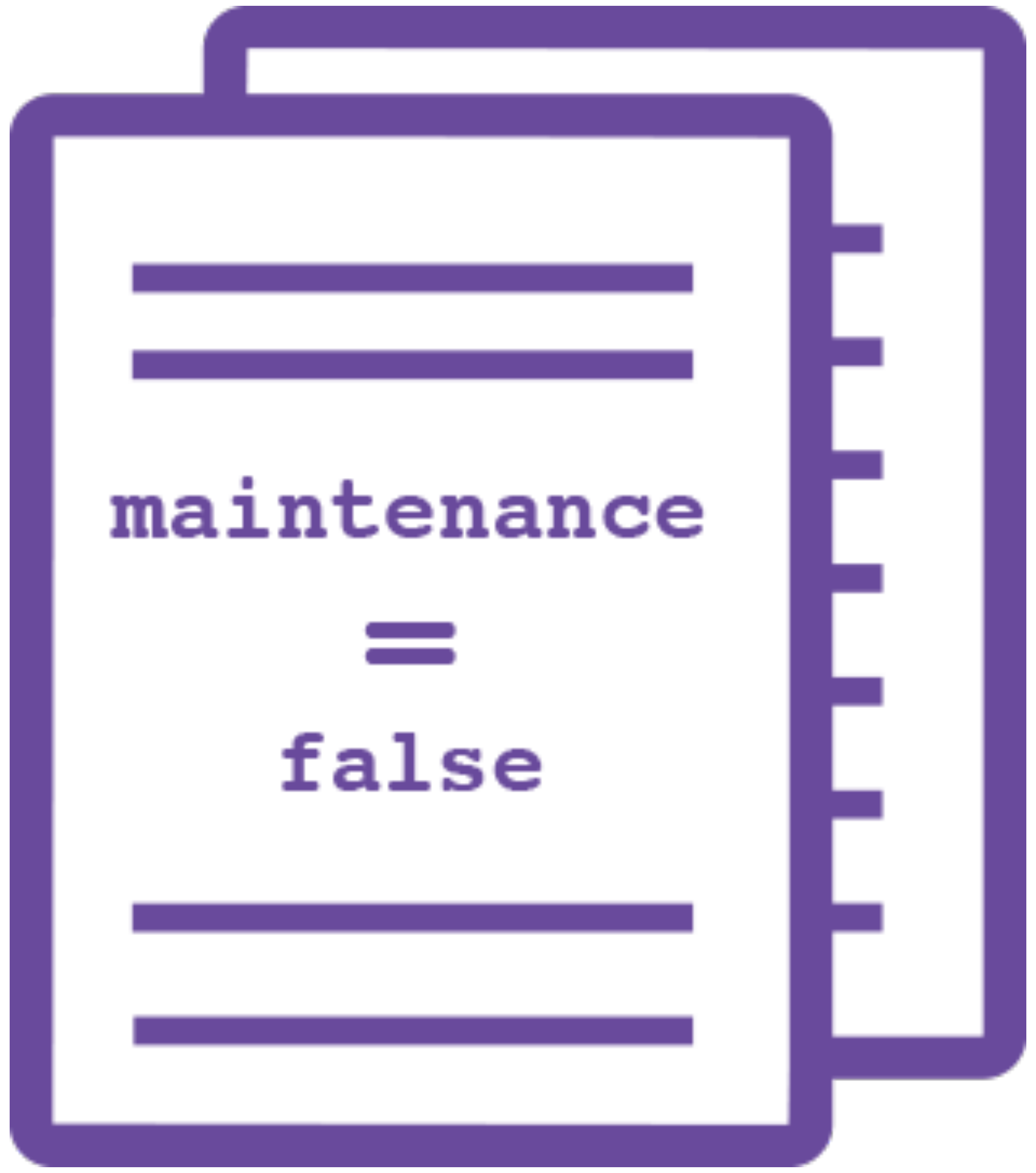
DNS

Load Balancer

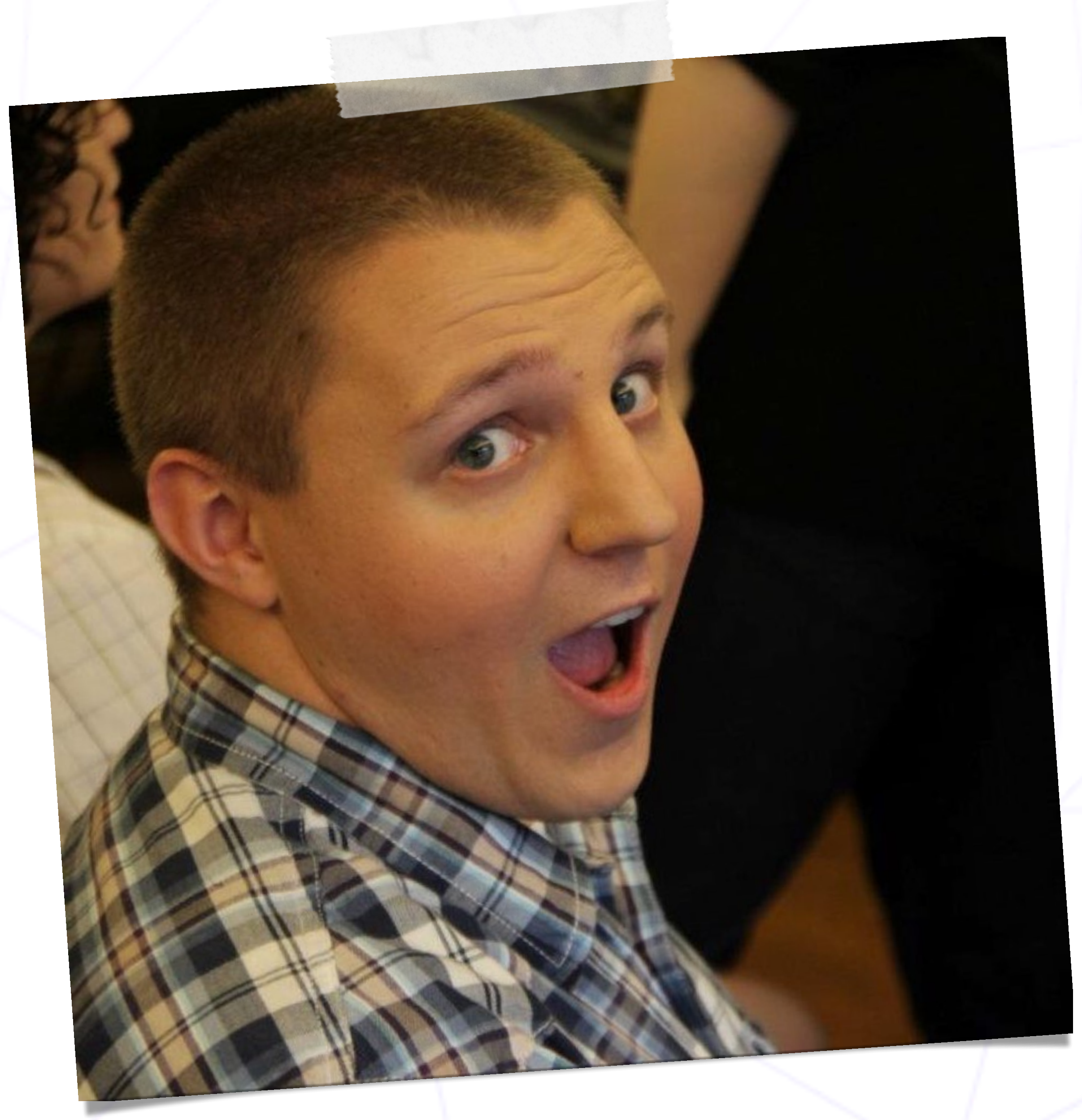
AMI

Service Discovery
HTTP + DNS

Host & Service
Level Health Checks



K/V Store
HTTP API



SETH VARGO

@sethvargo

QUESTIONS?