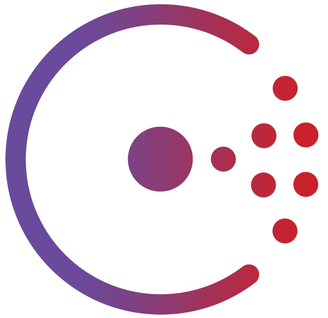


LOOKING FORWARD

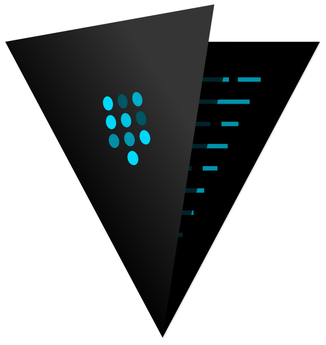
The Future of Tools and Techniques in Operations

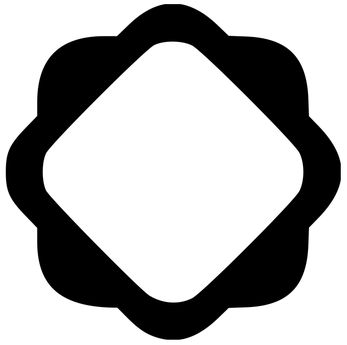
HashiCorp



HashiCorp





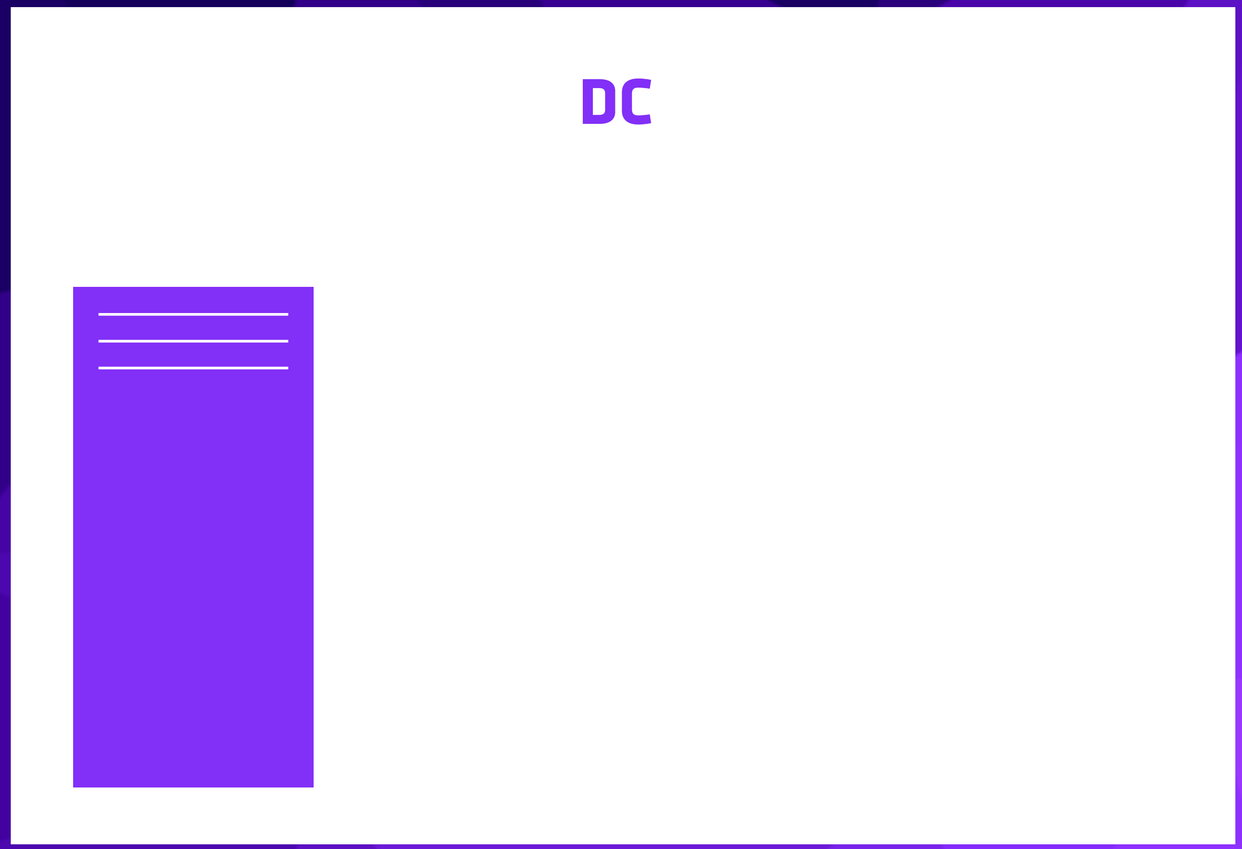
LOOKING FORWARD

The Future of Tools and Techniques in Operations

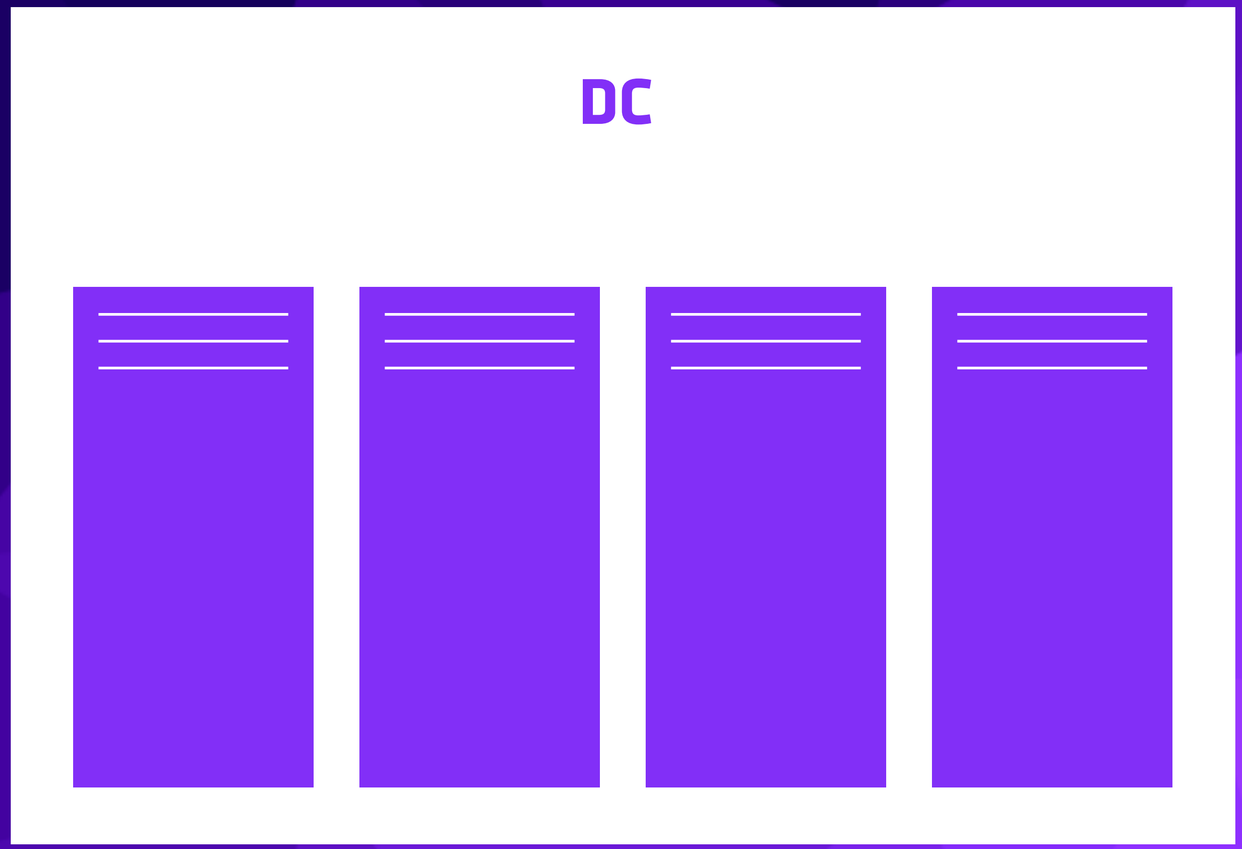
HashiCorp

The
MODERN DATACENTER

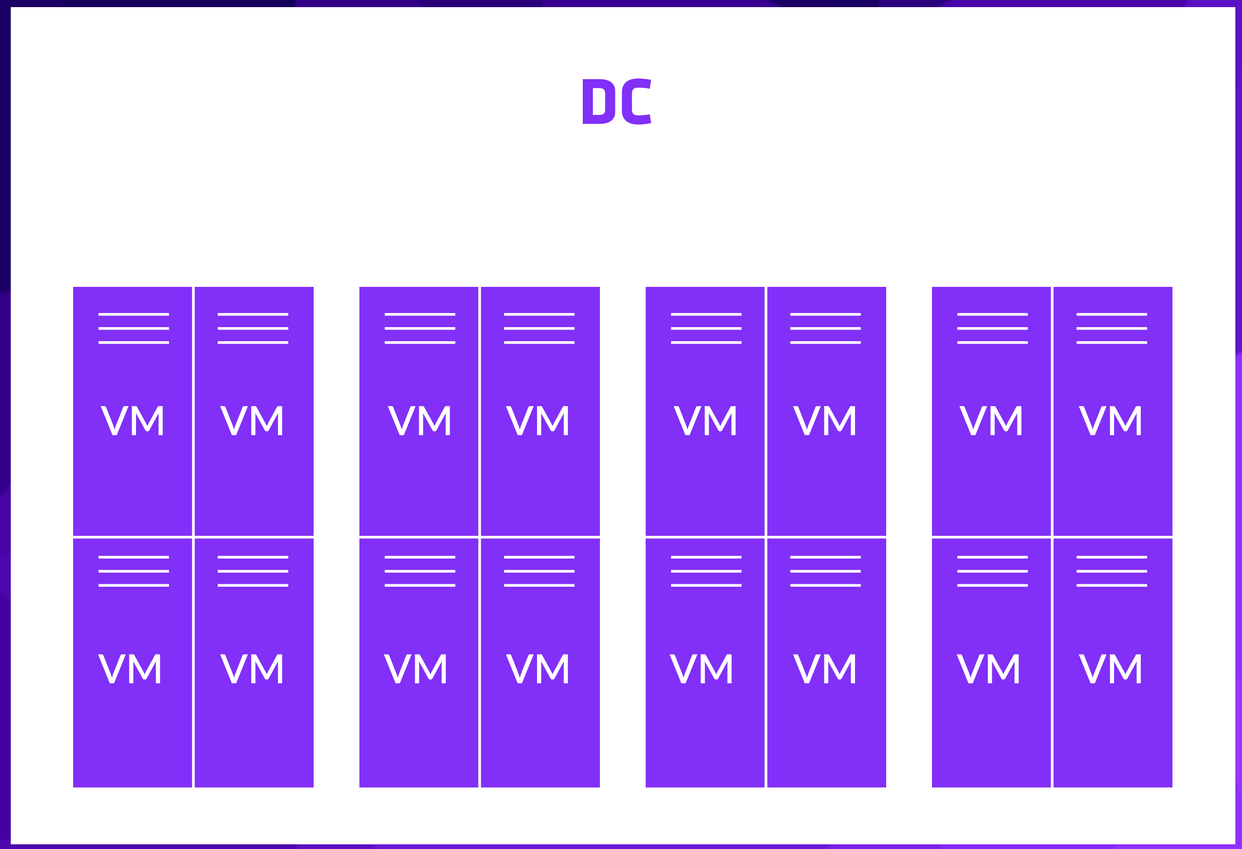
RISING DATACENTER COMPLEXITY



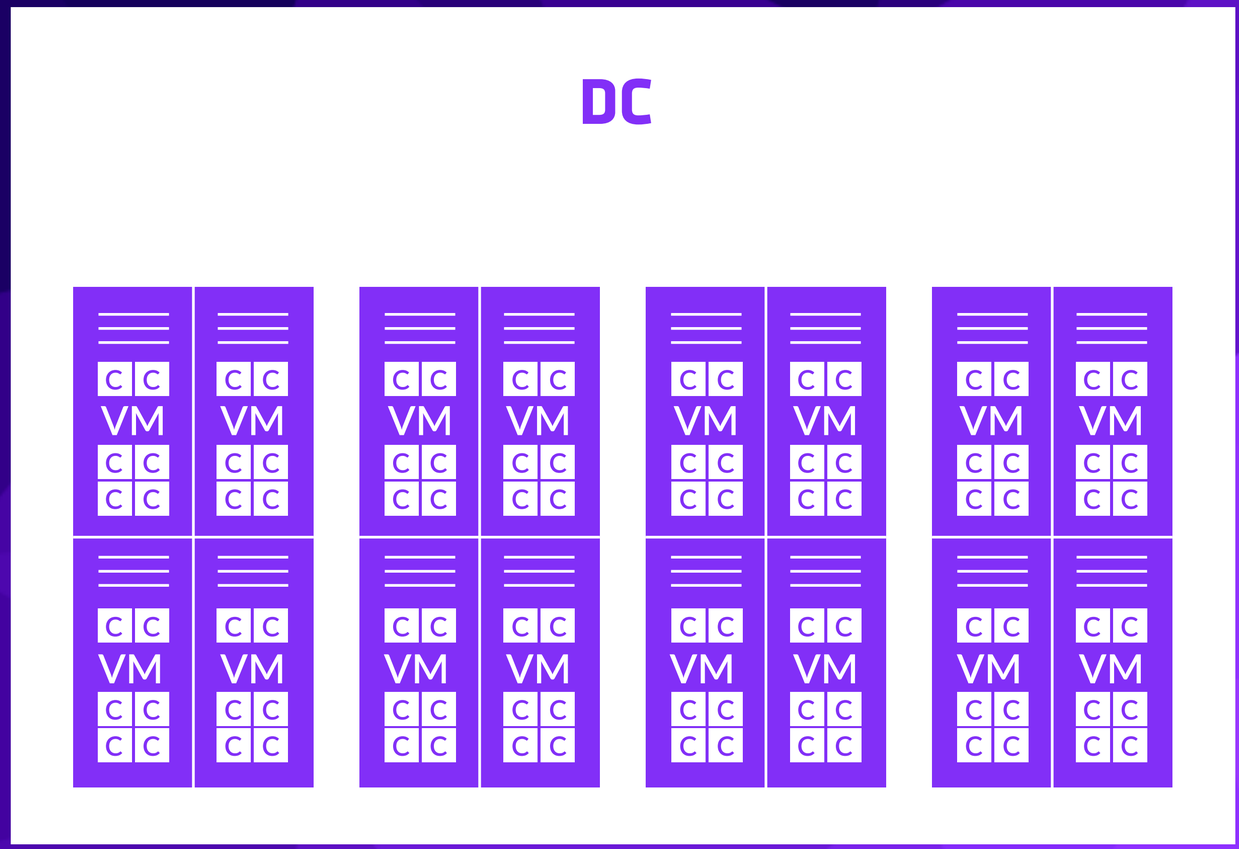
RISING DATACENTER COMPLEXITY



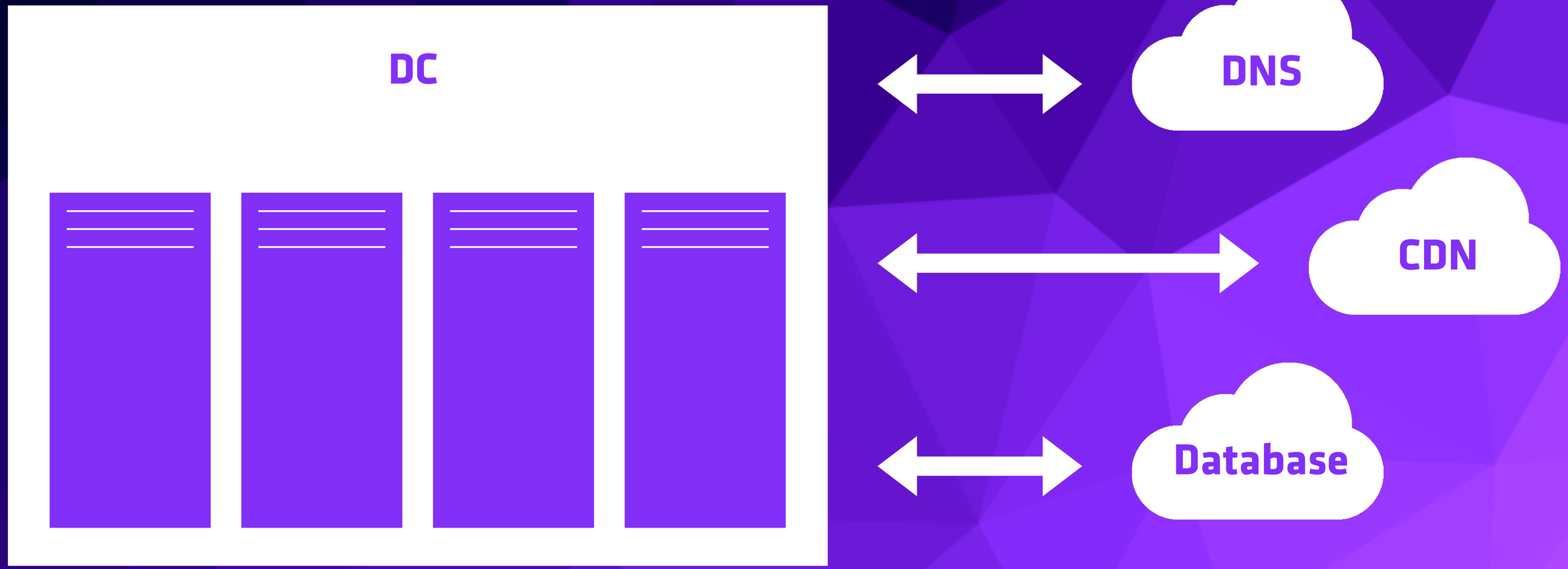
RISING DATACENTER COMPLEXITY



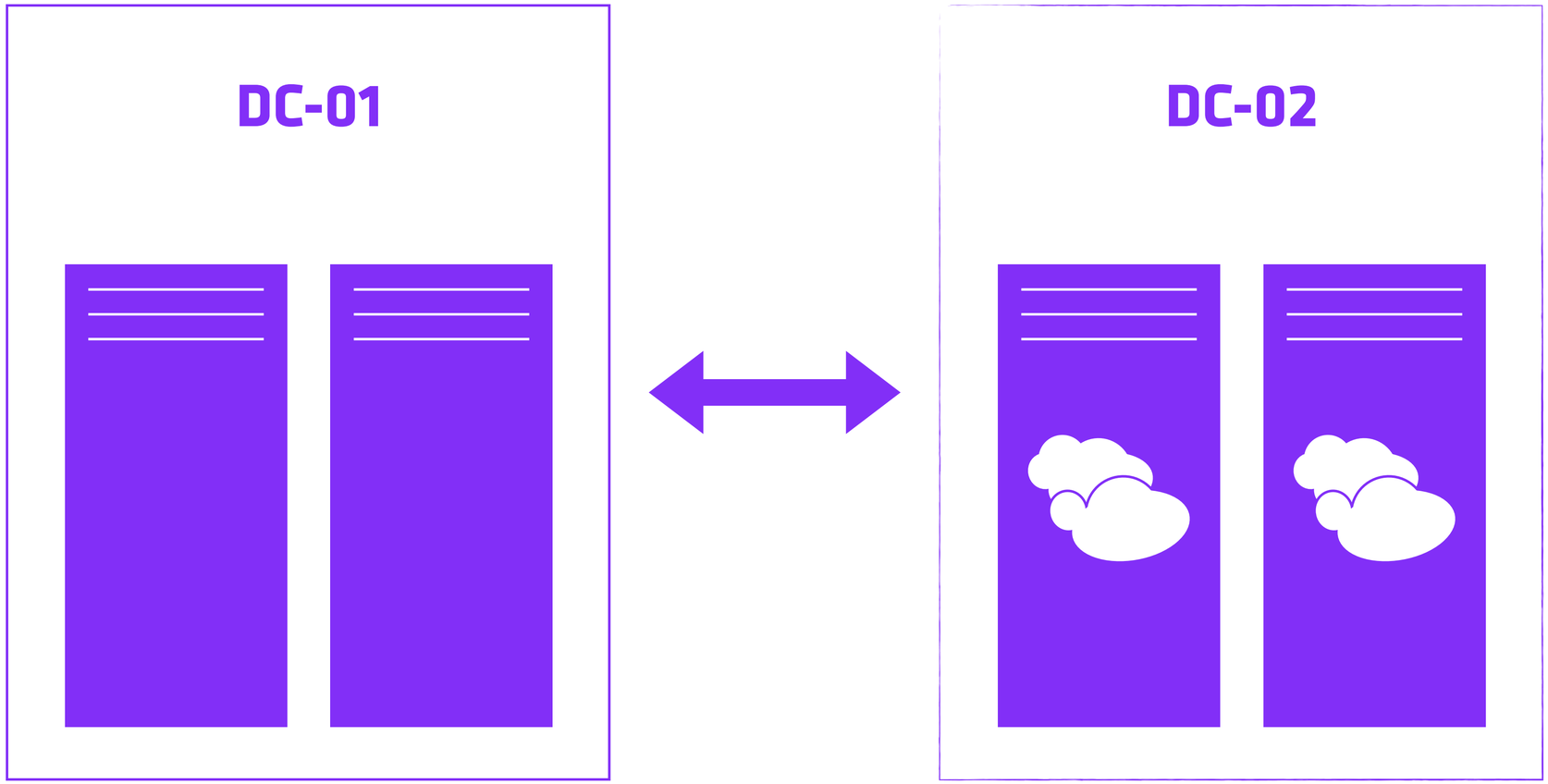
RISING DATACENTER COMPLEXITY



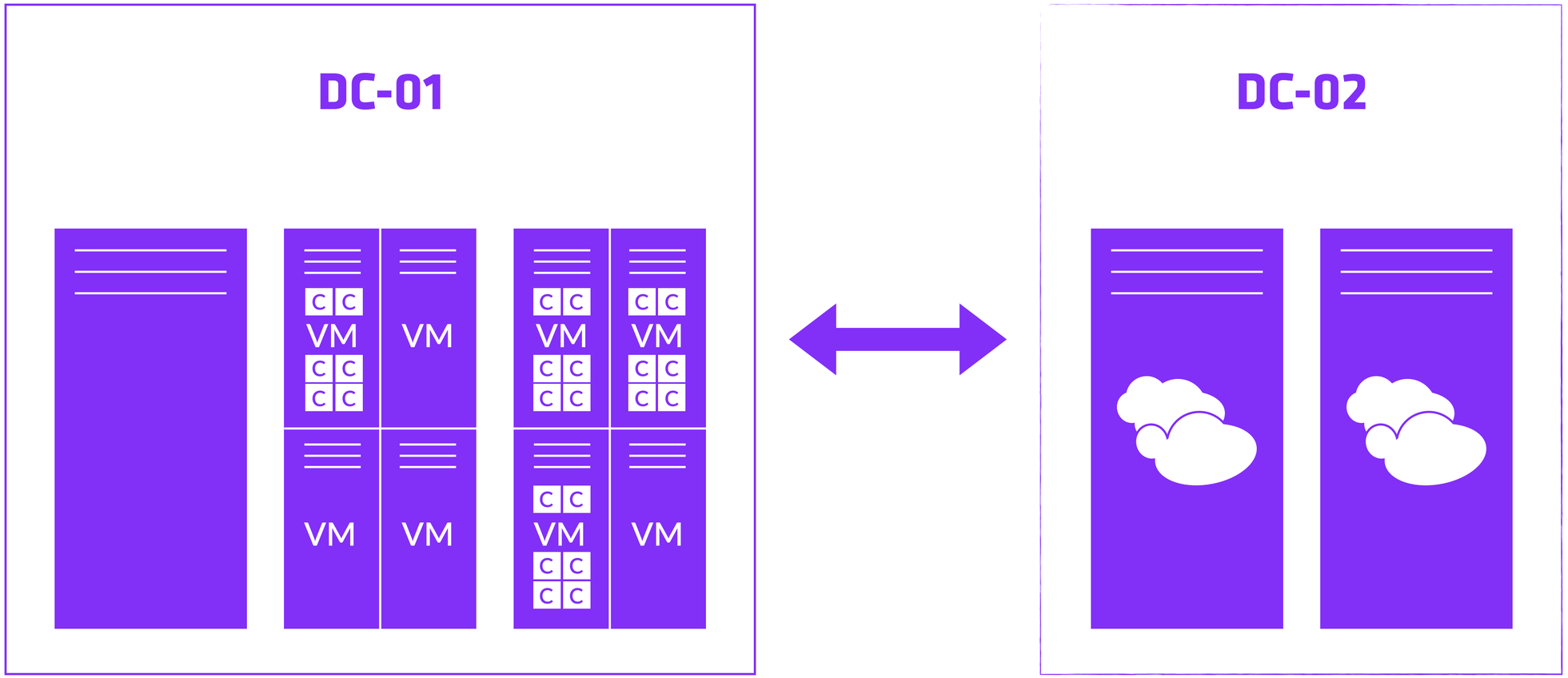
RISING DATACENTER COMPLEXITY



RISING DATACENTER COMPLEXITY



RISING DATACENTER COMPLEXITY

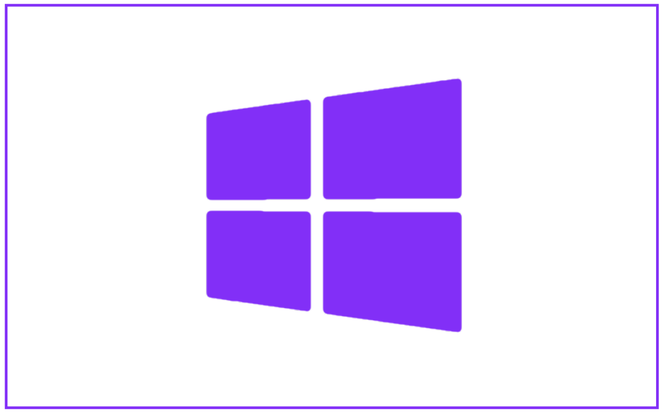


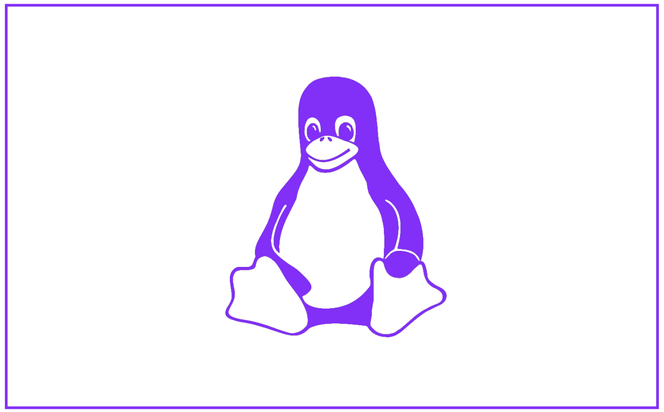
RISING DATACENTER COMPLEXITY



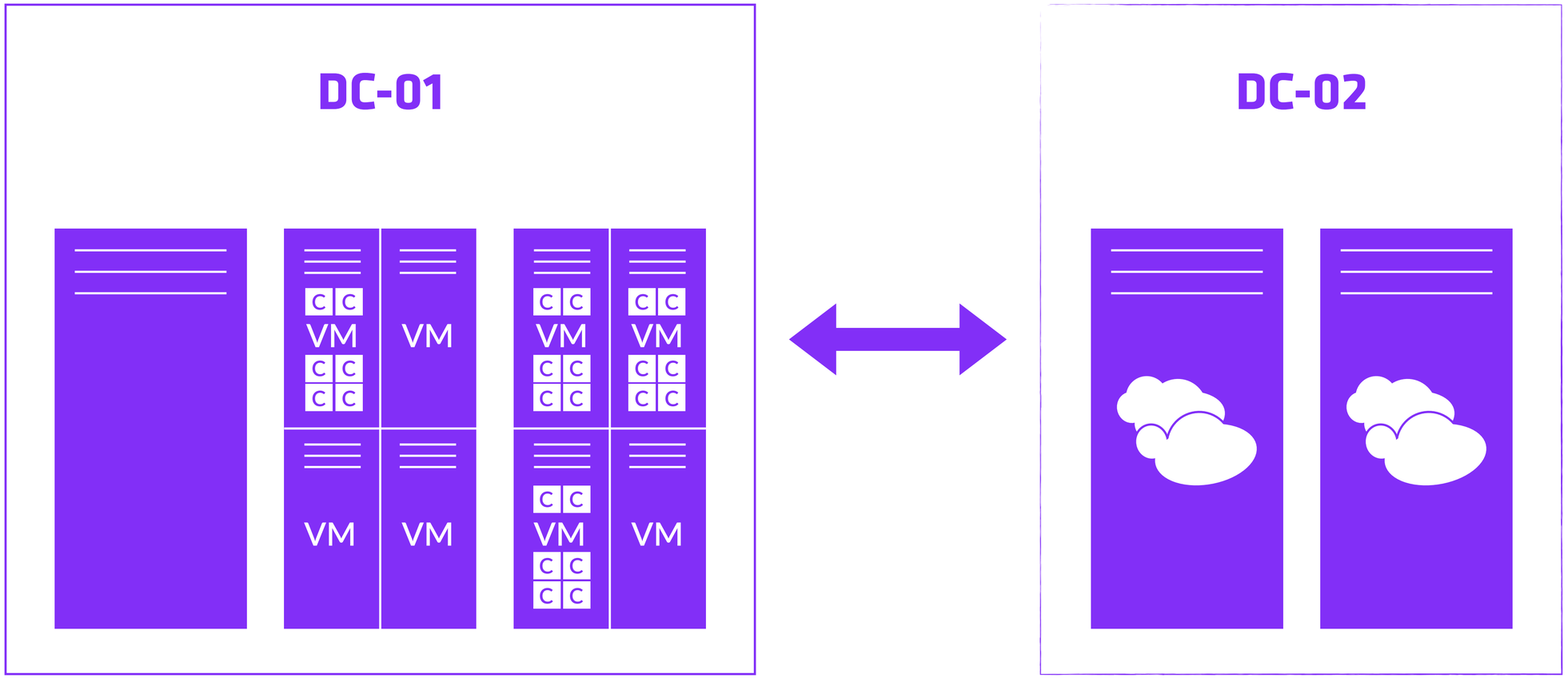
RISING DATACENTER COMPLEXITY







THE MODERN DATACENTER



Why?

Effectively **deliver and maintain** applications.

EMERGING
BEST PRACTICES

Heterogeneity

Idea: More technologies in use, not less.

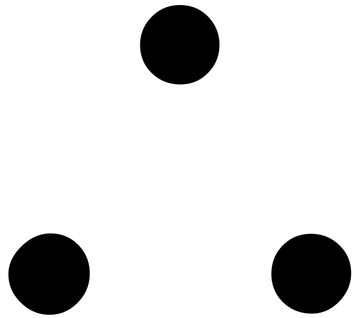
HOMOGENEITY

- A single paradigm: VMs, containers, etc.
- A single tech stack: Java, LAMP, etc.
- A single cloud provider: Amazon

Reality!

Things change for
the better.*

* Sometimes it takes a few changes



Homogeneity means one of:

Homogeneity means one of:

1.) Change won’t happen [for us]

Homogeneity means one of:

1.) Change won't happen [for us]

2.) Change is atomic (A → B)

Homogeneity means one of:

1.) Change won't happen [for us]

2.) Change is atomic (A → F)

3.) Change is gradual (A → A' → ... → F)

A → A’ → … → F

A → A’ → … → F

$A \rightarrow A' \rightarrow \ldots \rightarrow F_1$

$$A \rightarrow A' \rightarrow \ldots \rightarrow F_1$$

$$B \rightarrow \ldots \rightarrow F_2$$

$$A \rightarrow A' \rightarrow \ldots \rightarrow F_1$$

$$B \rightarrow \ldots \rightarrow F_2$$

$$C \rightarrow \ldots \rightarrow F_3$$

A → A’ → … → F_1

B → … → F_2

C → … → F_3

D

→ = Heterogeneity

Heterogeneity

Embrace things will be **different**.

Heterogeneity

- Embrace things will be different
- Choose tooling that aligns with that reality
- Hire people who align with that reality
- *Be mindful of change*

Multi-cloud

Multi-Cloud Motivation

- Regions

Multi-Cloud Motivation

- Regions
- Features

Multi-Cloud Motivation

- Regions
- Features
- Pricing

Multi-Cloud Motivation

- Regions
- Features
- Pricing
- Lock-in avoidance

Multi-tech

Multi-Tech Motivation

- Right tool for the job
- Job Market
- Forced requirements (SDK, walled garden, etc.)

Multi-compute

Multi-Compute Motivation

- Requirements

  (performance, legacy, security, etc.)
- Existing knowledge, experience

Challenges

- Management of resources and cost
- Speed of light

Solutions

Heterogeneity

Solutions

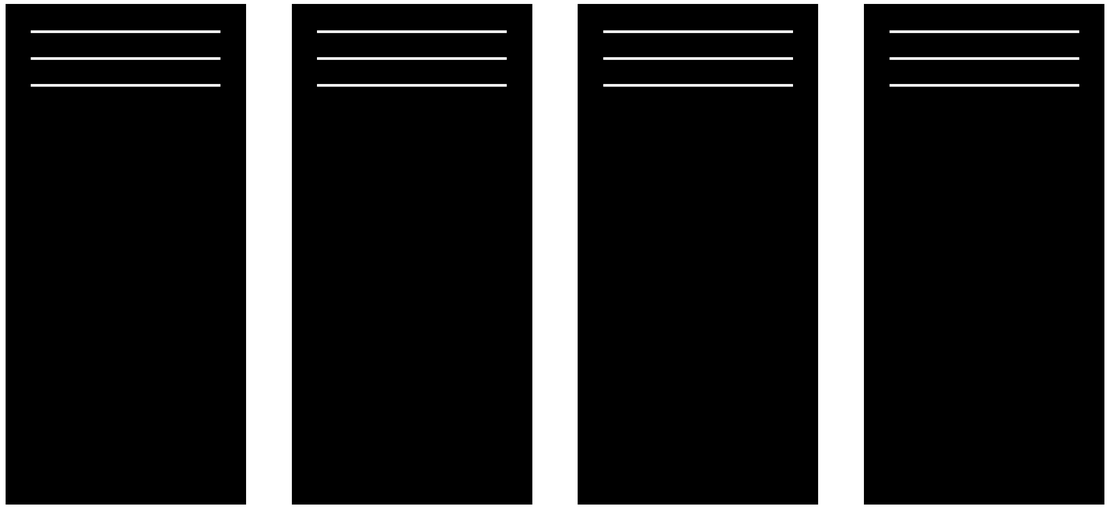
- Terraform, Kubernetes, etc...
- Unrelated workloads on different tech

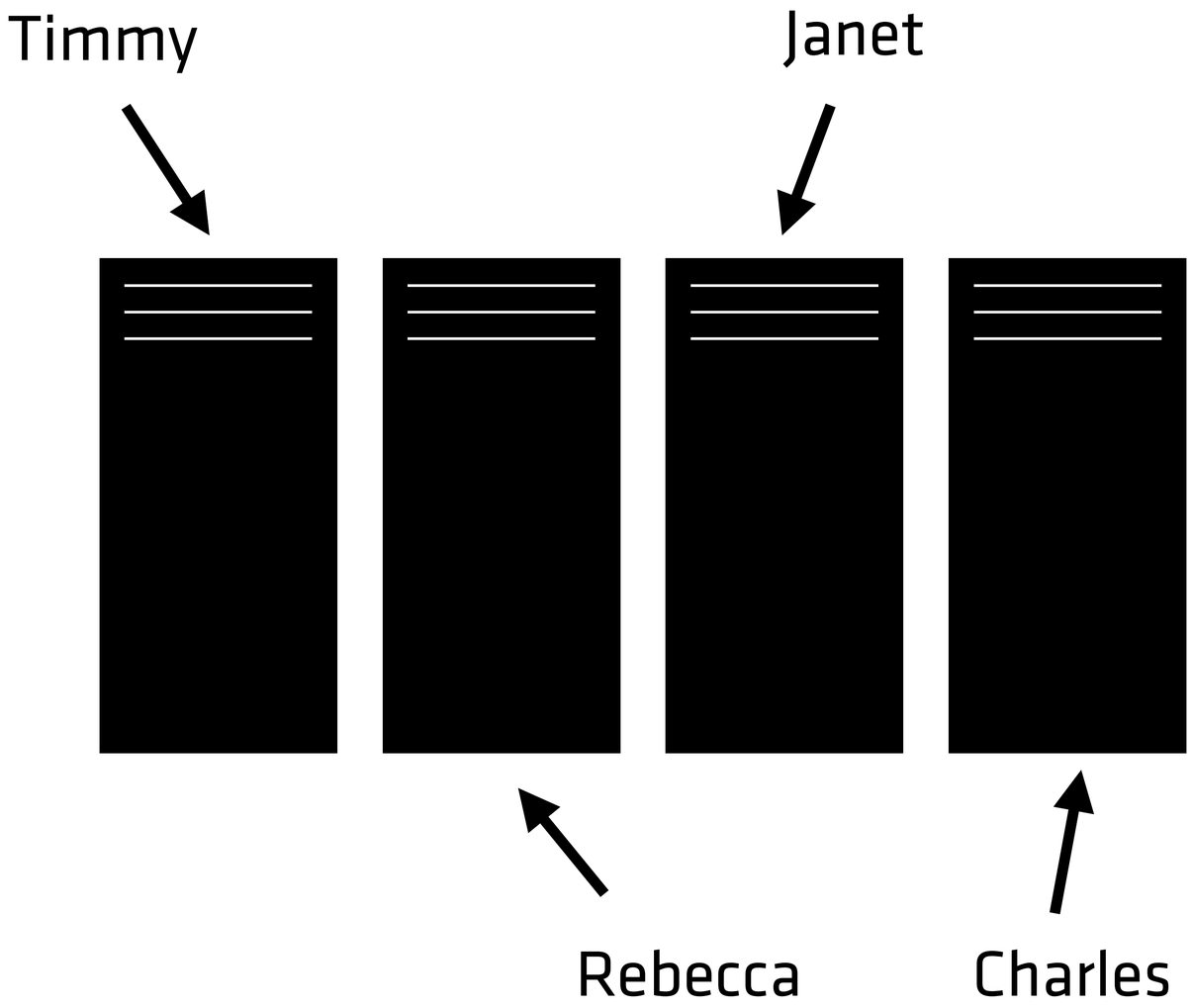
Datacenter as Computer

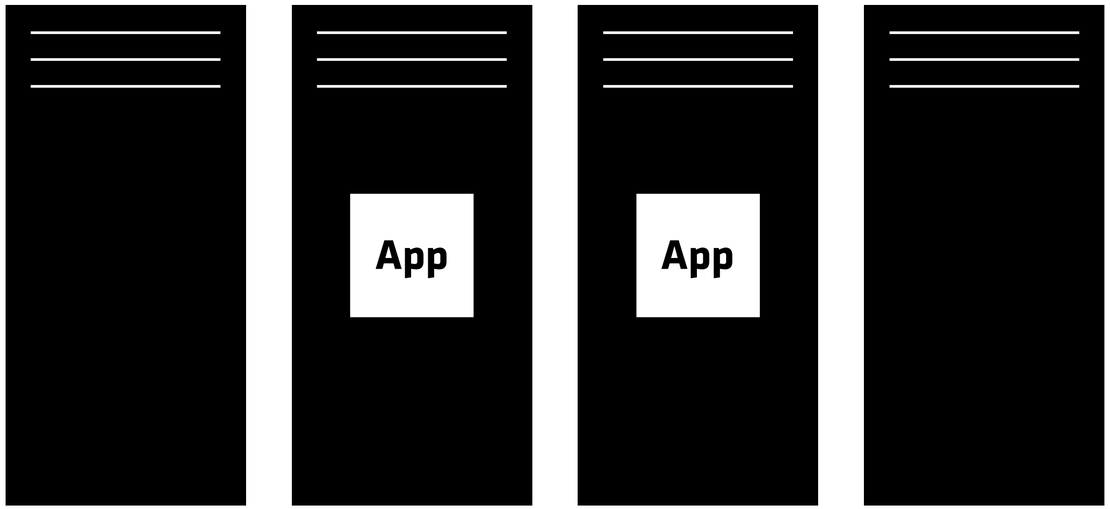
Idea: DC as single shared resource rather than a collection of devices.

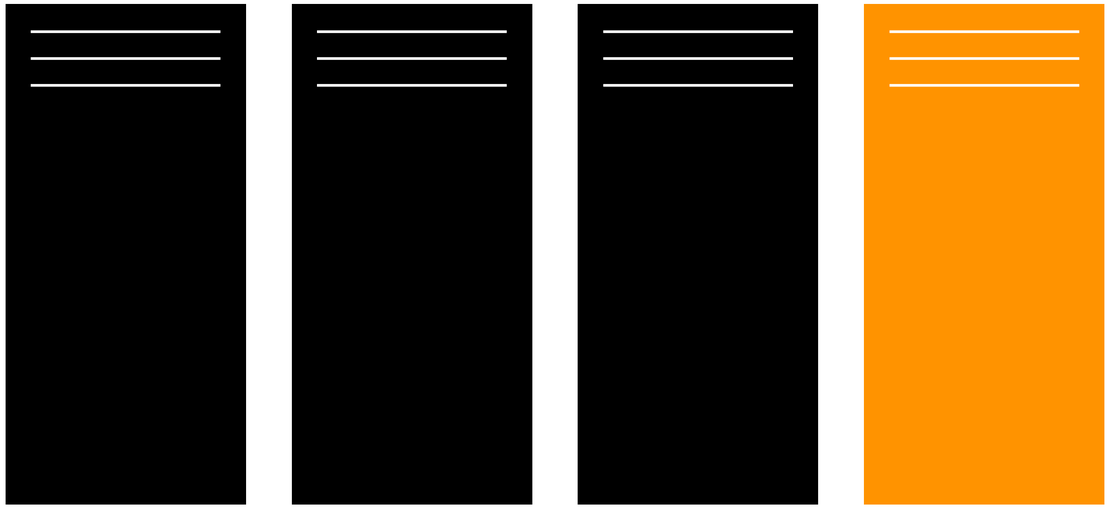
Today

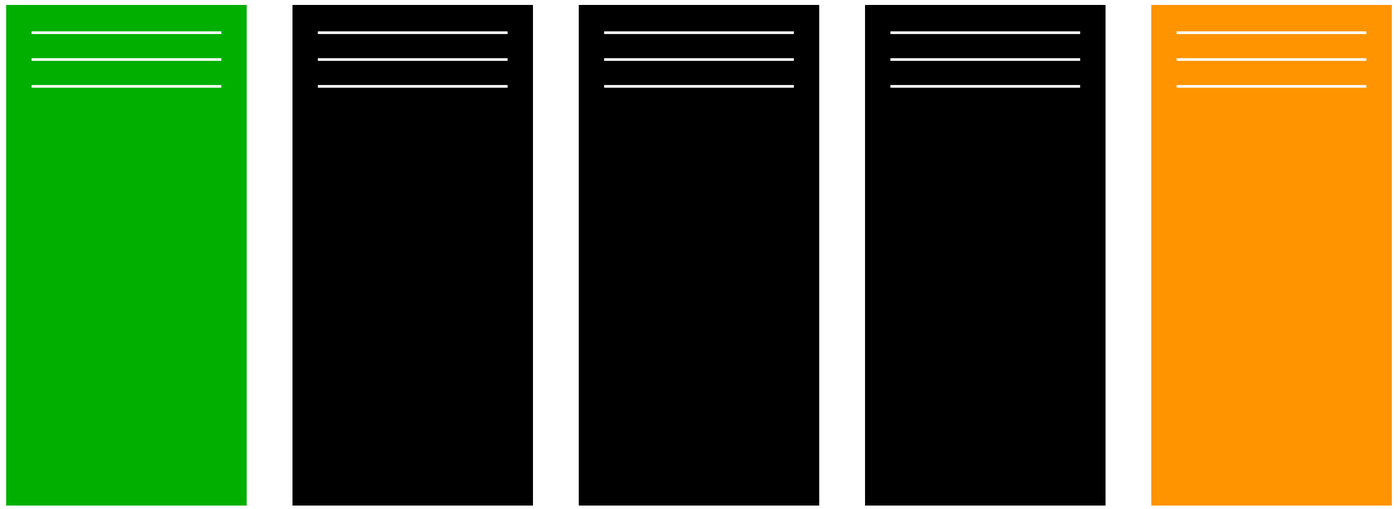





Timmy
Janet
Rebecca
Charles


App
App



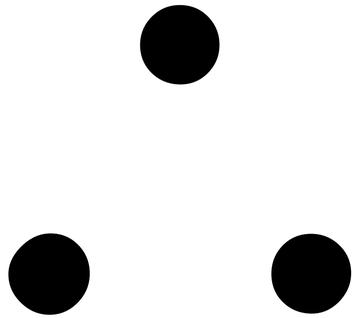




TEST: Can I unplug your servers? Is that okay?

Reality!

Number of servers, apps growing at a breakneck pace.



Choices:

Choices:

1.) Growth will slow down

Choices:

1.) Growth will slow down

2.) Train and hire

Choices:

1.) Growth will slow down

2.) Train and hire

3.) Lower cost/complexity

Lower cost/complexity

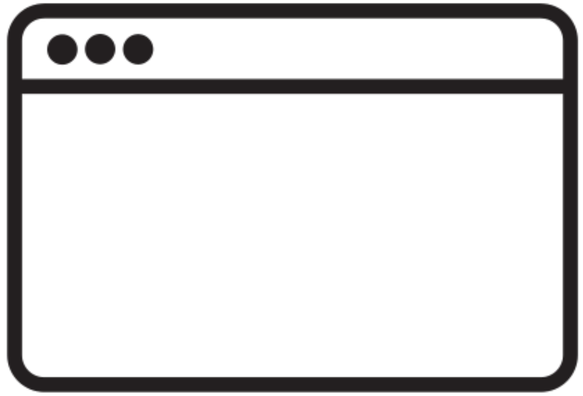
Lower cost/complexity

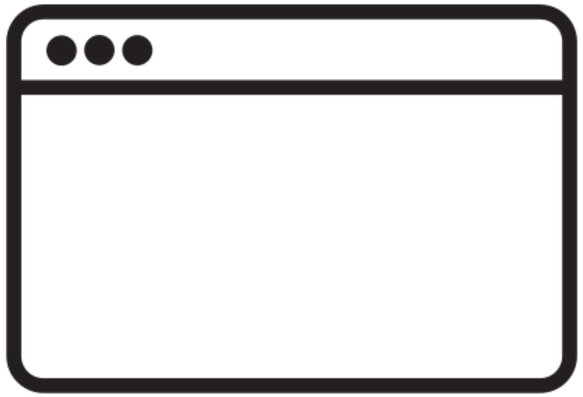
=

DC as Computer

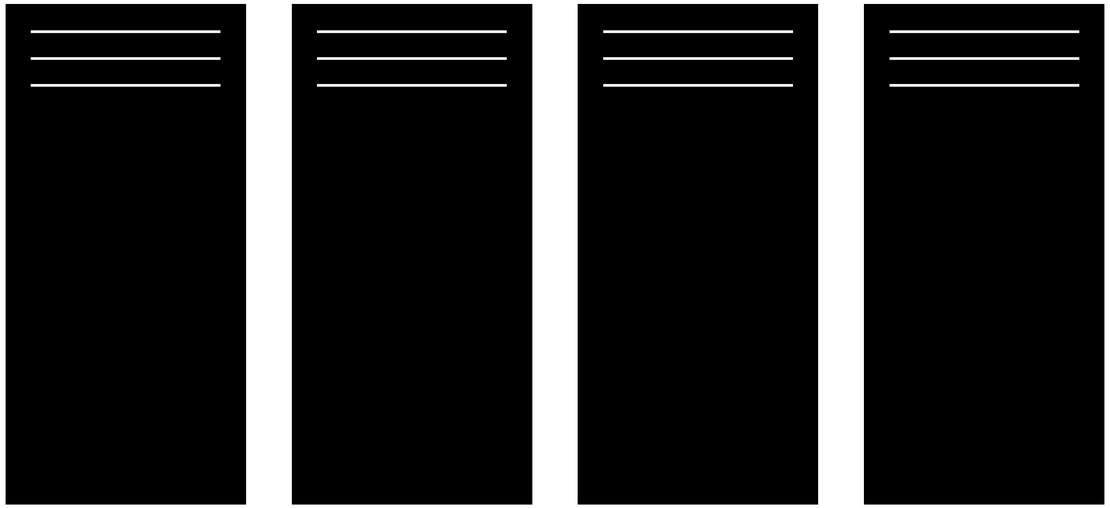
Datacenter as Computer

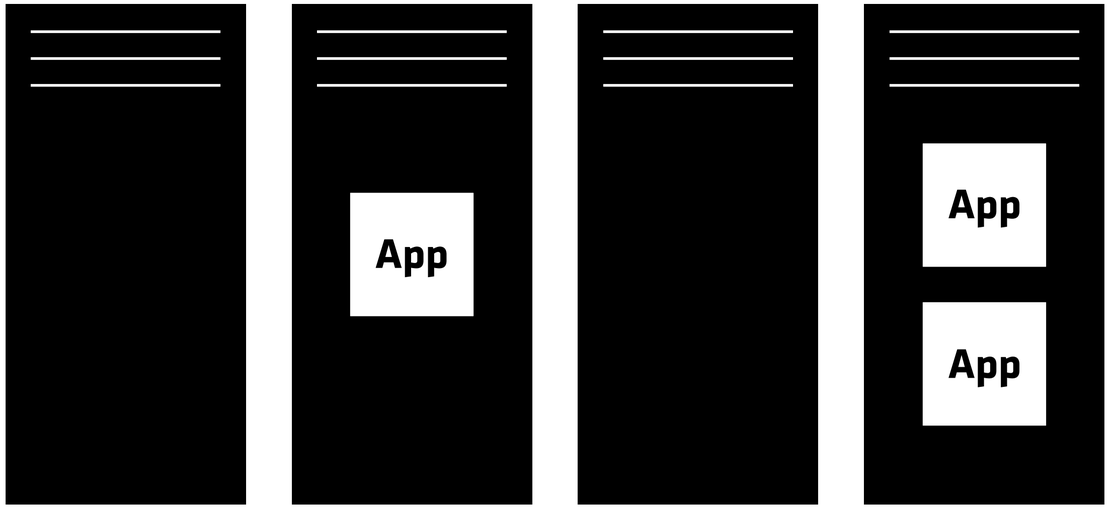
Separate application from the resources it runs on.

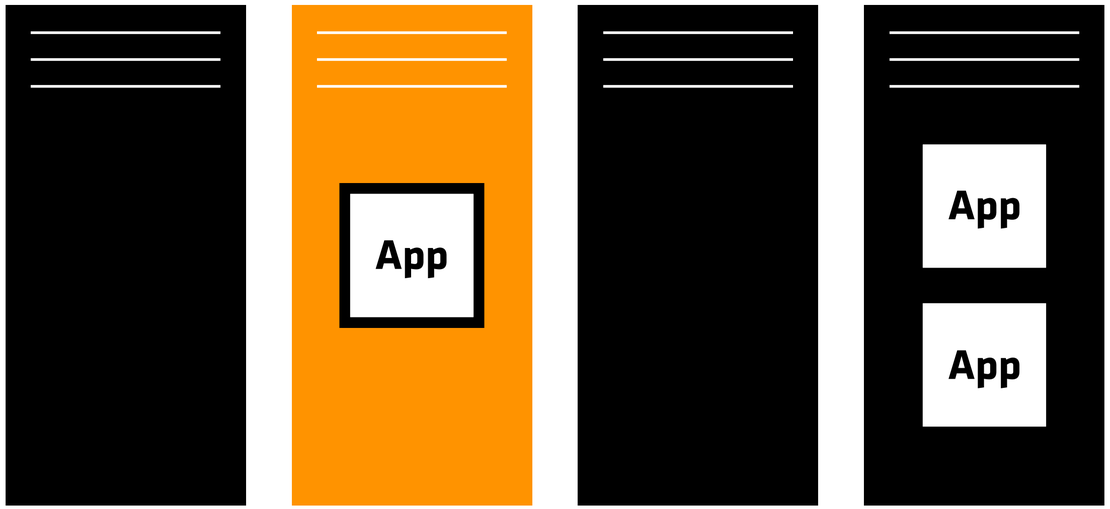


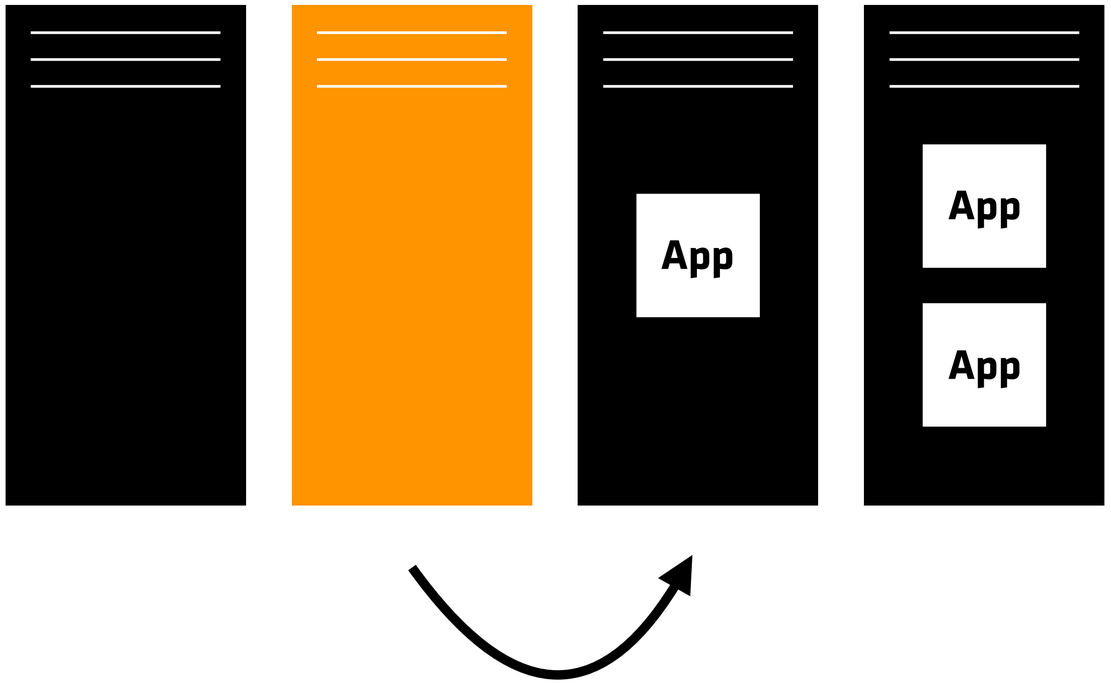


RAM: 500 MB
CPU: 200 MHz
DISK: 50 IOPS, SSD

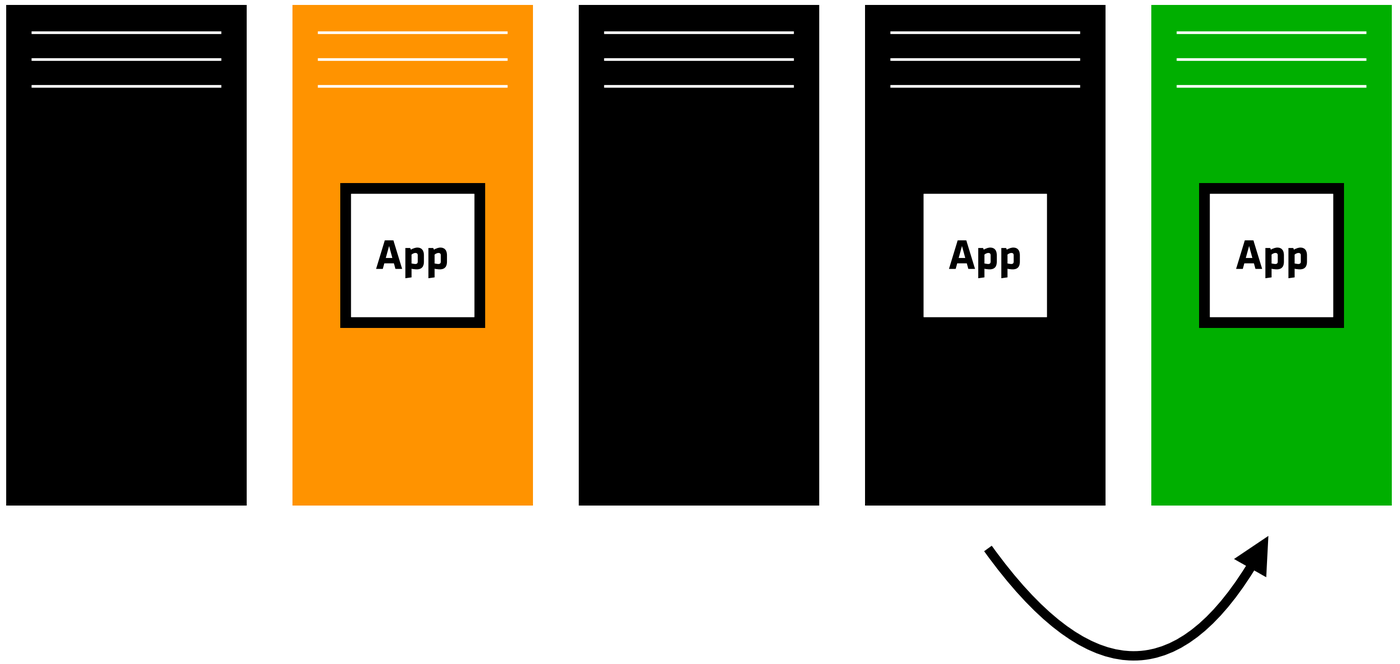



App
App
App


App
App
App


App
App
App


App
App
App


App
App
App

Allows for choice in lower level components.

Challenges

- Management is complex
- Not all apps fit this model easily
- Trust

Solutions

Datacenter as Computer

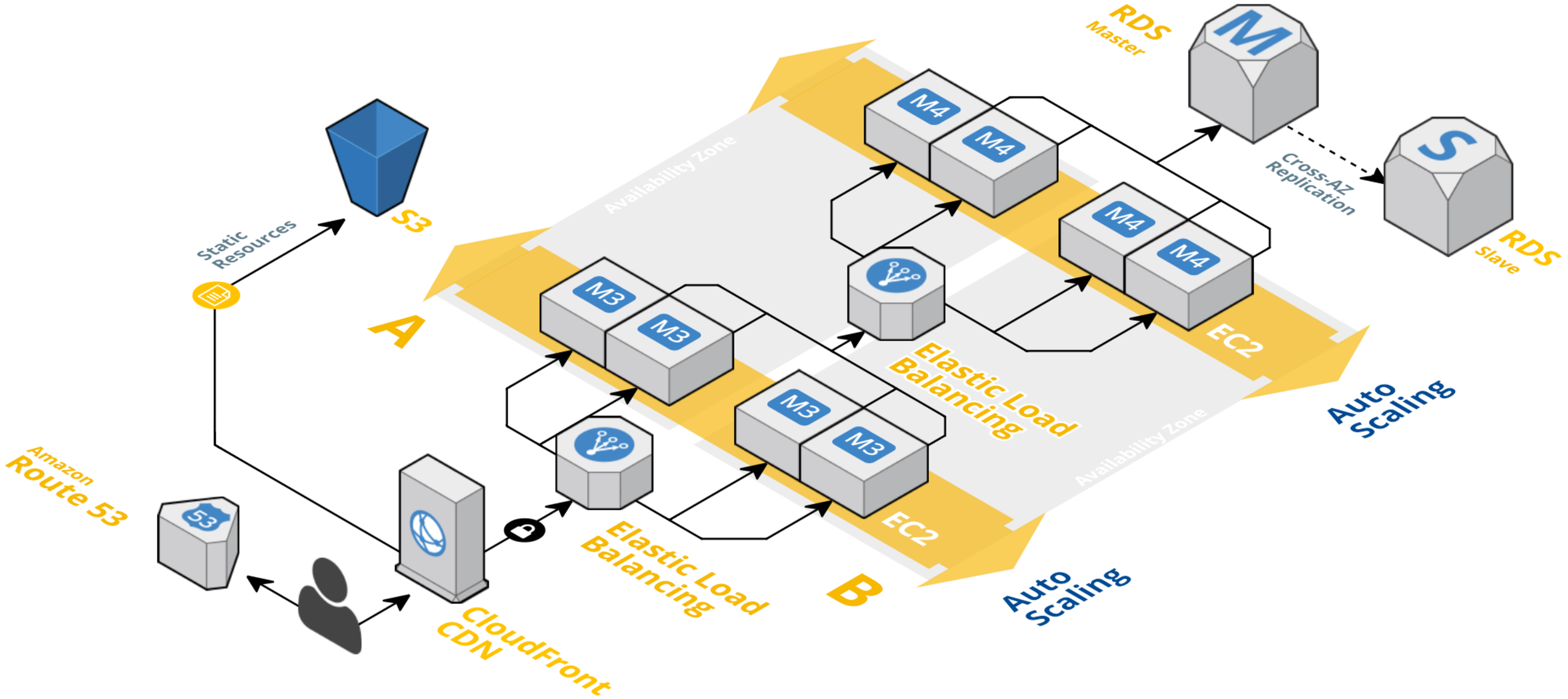
Solutions

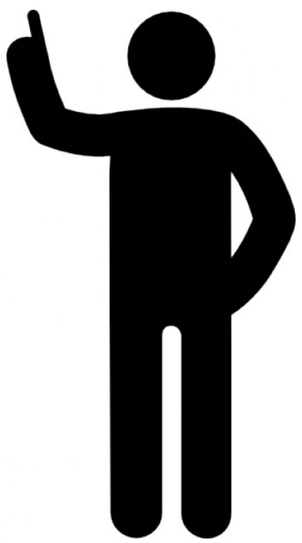
- Schedulers
  - Complex, years to mature
- State: Ephemeral vs. Sticky vs. Persistent
- Heterogeneity!

Declarative

Idea: Goal-driven tooling where we describe *what*, not *how*.

Today

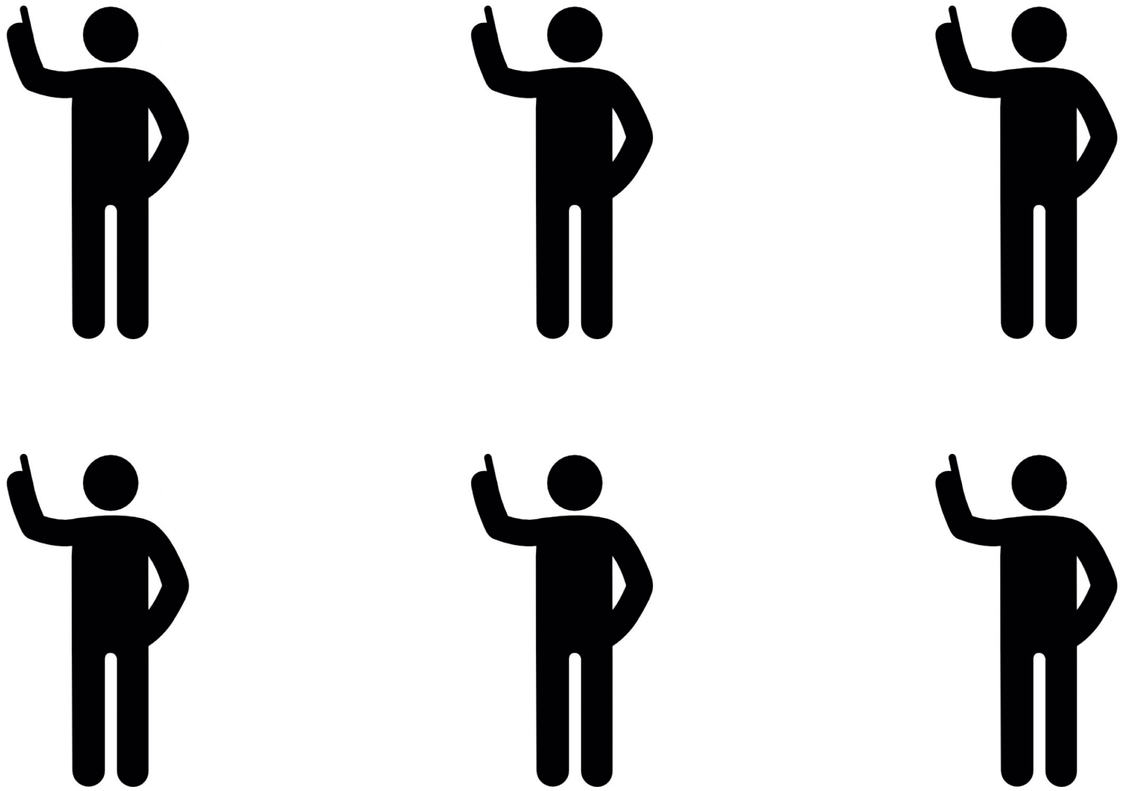

RDS
Master
M
Cross-AZ
Replication
S
RDS
Slave
S3
Static
Resources
Availability Zone
M4
M4
M4
M4
M3
M3
M3
M3
A
B
Elastic Load
Balancing
Elastic Load
Balancing
EC2
EC2
Auto
Scaling
Auto
Scaling
Availability Zone
Amazon
Route 53
53
CloudFront
CDN



I need the web server AMI updated to ami-abcd1234. I need to spin up 3 more servers. And I need to change the maintenance window for the database.

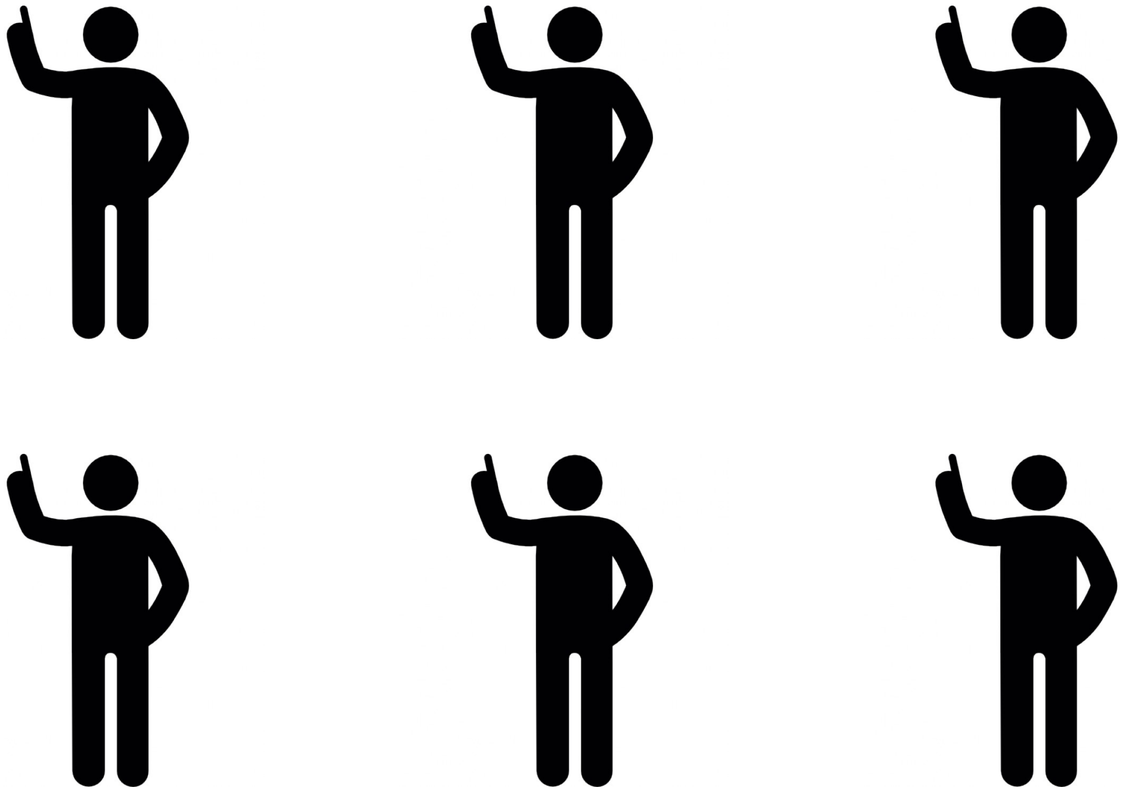
WHOA. Let’s break that down. Let’s start with the updated AMIs, write down a plan, test it out for a few weeks, then schedule a full day to roll it out.

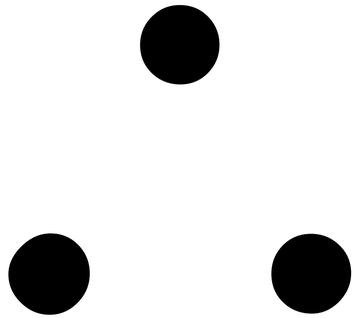




Reality!

Change requires a
burden of knowledge
too large.





Choices:

Choices:

1.) Hire or train more experts

Choices:

1.) Hire or train more experts

2.) Enable non-experts through higher level abstractions.

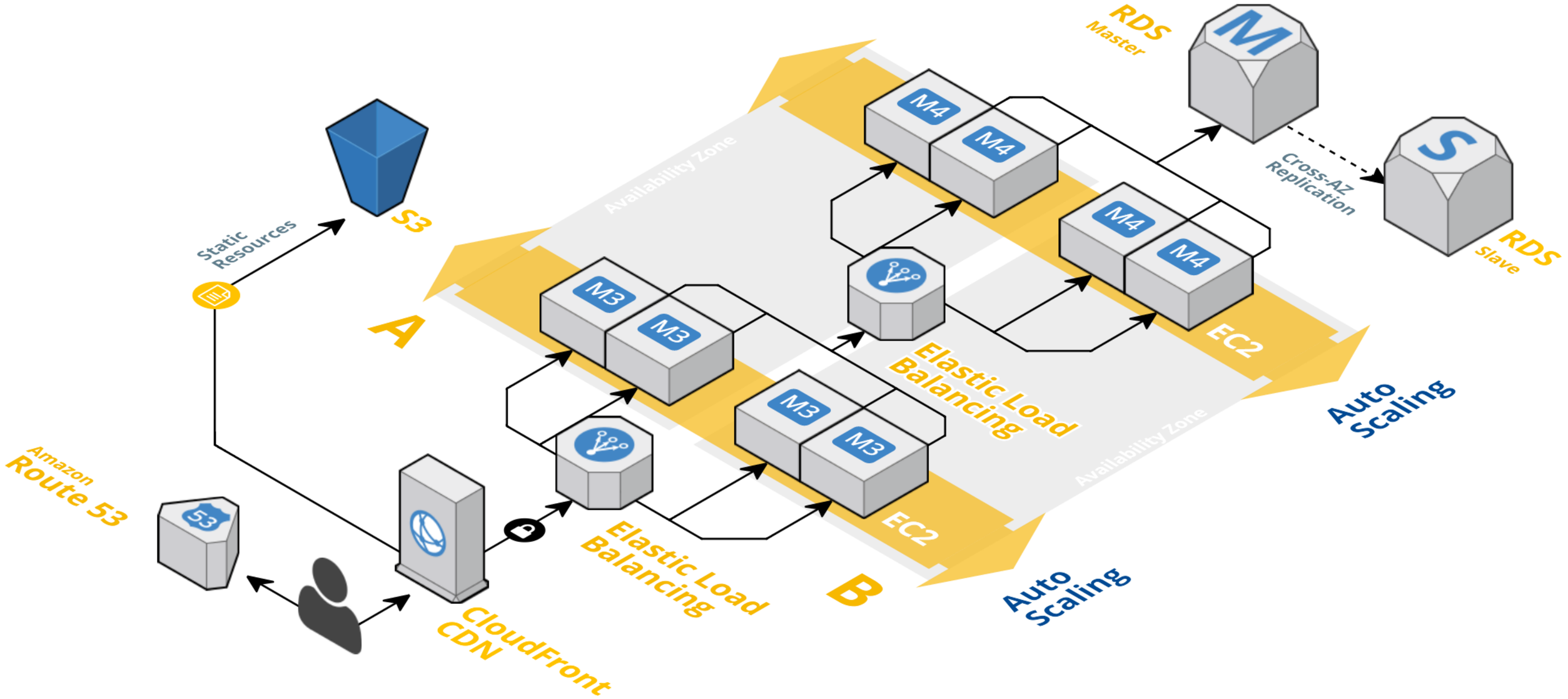
Higher Abstraction

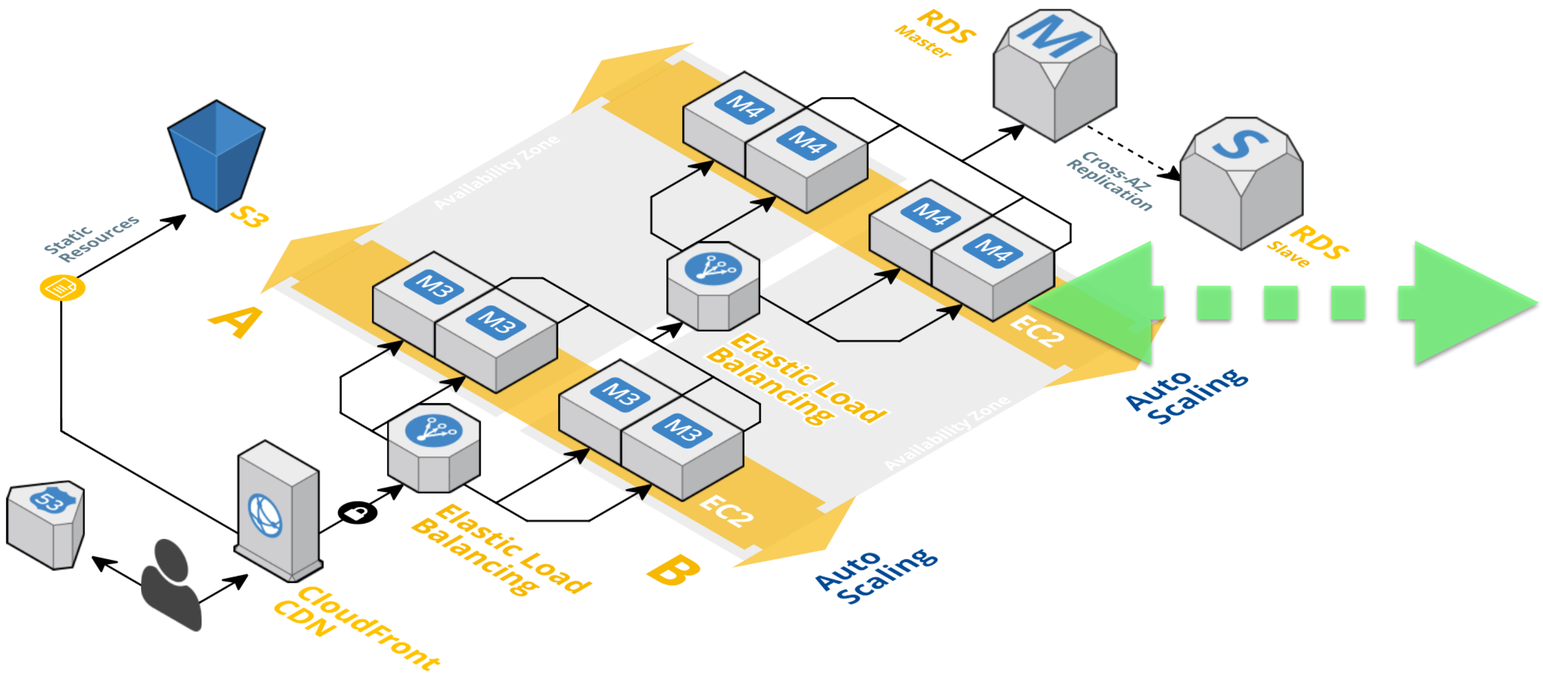
=

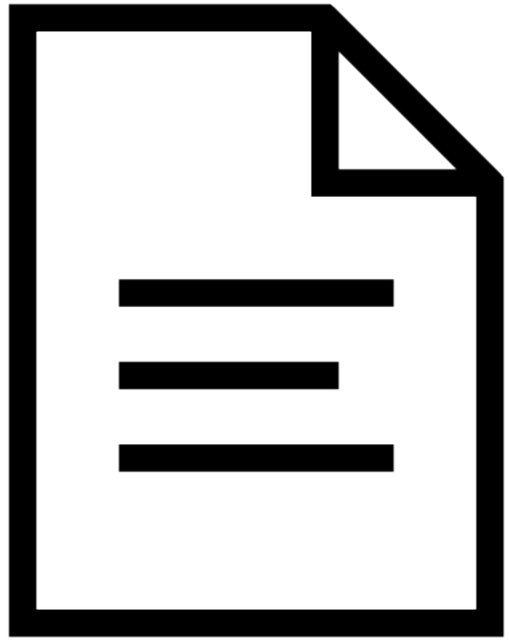
Declarative

Declarative

Describe your *goal*,
not the steps to get there.


RDS
Master
M
Cross-AZ
Replication
S
RDS
Slave
S3
Static
Resources
Availability Zone
M4
M4
M4
M4
M3
M3
M3
M3
A
B
EC2
EC2
Elastic Load
Balancing
Elastic Load
Balancing
Auto
Scaling
Auto
Scaling
Amazon
Route 53
53
CloudFront
CDN
Availability Zone


RDS
Master
M
Cross-AZ
Replication
S
RDS
Slave
S3
Static
Resources
Availability Zone
M4
M4
M4
M4
M3
M3
M3
M3
A
Elastic Load
Balancing
EC2
Auto
Scaling
Availability Zone
EC2
B
Auto
Scaling
Elastic Load
Balancing
CloudFront
CDN



Benefits

- Flexible choices for *how*
- Easier to educate, knowing *what* you want is usually easier than *how*.
- Complex tasks are easily modeled

Challenges

- Trust, trust, TRUST!
- Lack of control (trust)
- Difficult problem, large scope

Solutions

Declarative

Solutions

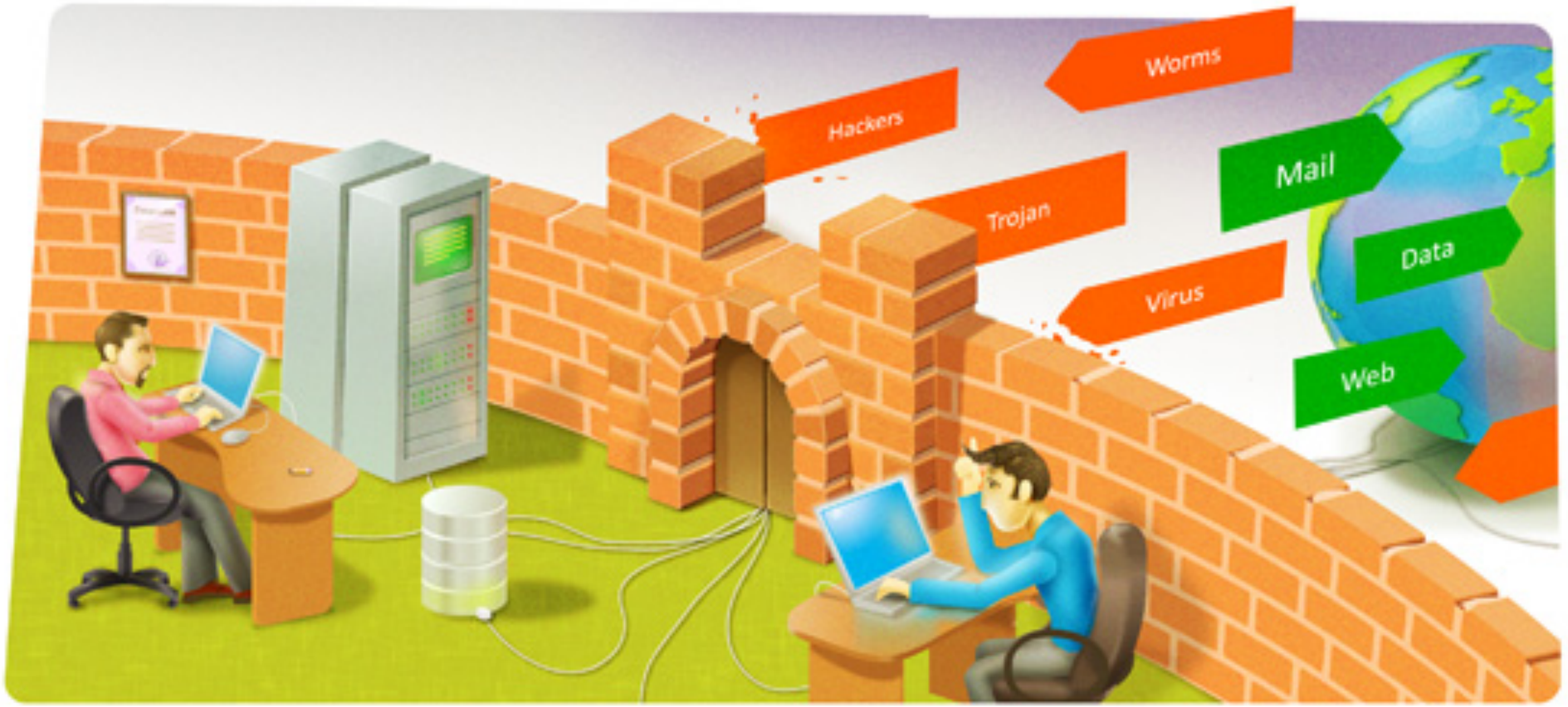
- Schedulers, Terraform, EC2
- “Plan” as a fundamental primitive

App-Level Security

Idea: App-level security in addition to network and server level.

Today

~~Today~~ Historically


Worms
Hackers
Mail
Trojan
Data
Virus
Web



“Us” vs. “Them”



Historically

- External vs. Internal
- Server to Server (iptables, route tables, etc.)
- Hand-curated set of rules
- Unencrypted internal communication

Reality!

Clouds blur the line.

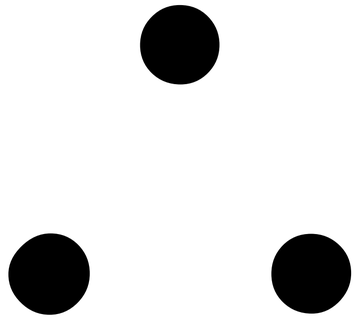
Clouds blur the line.

Threats come from within.

Clouds blur the line.

Threats come from within.

Orders of magnitude more apps.



Choices:

Choices:

1.) Manually curate all rules all the time

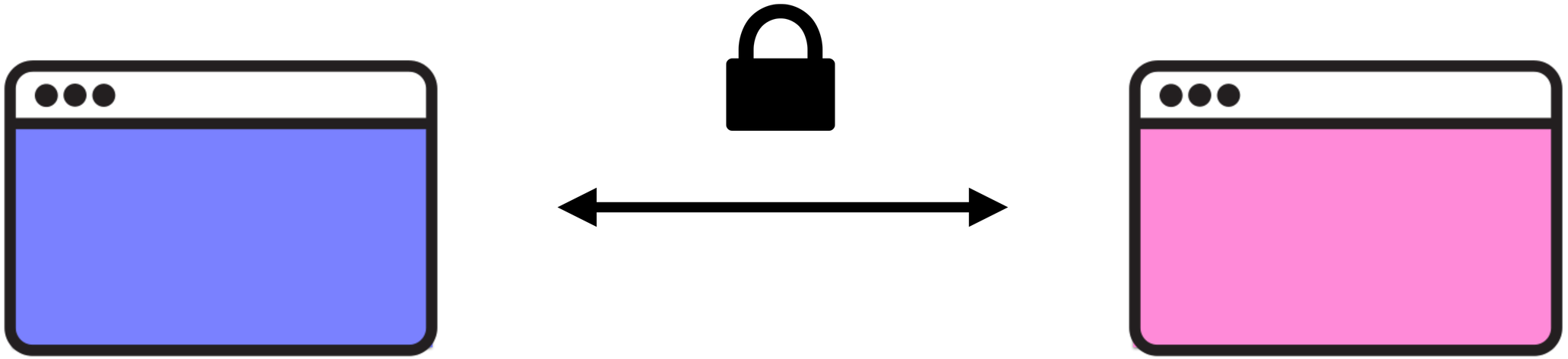
Choices:

1.) Hire or train more experts

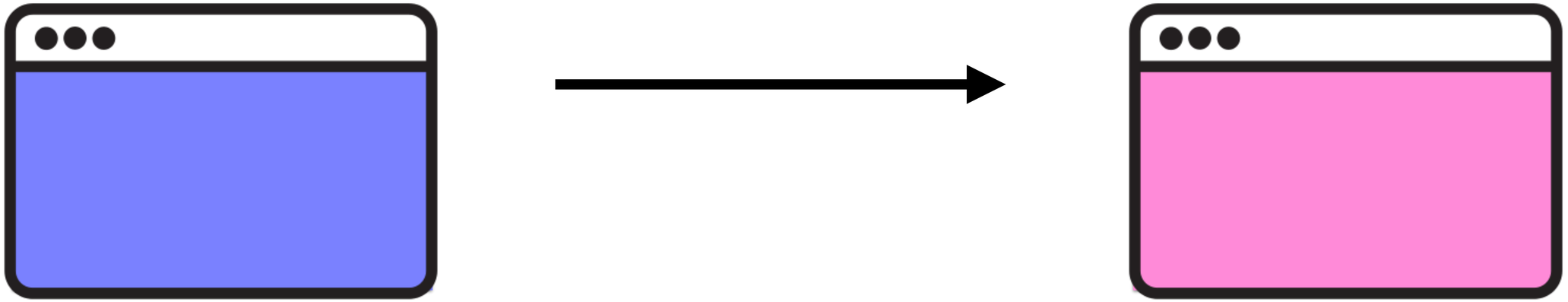
2.) Enforce security at the app-level

App-Level Security

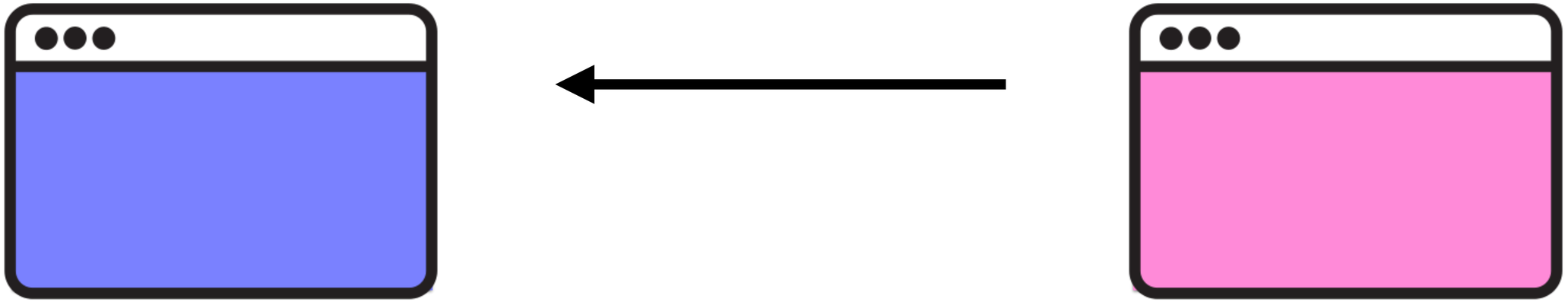
End-to-end encryption, identity for all connections, permission-checked APIs.



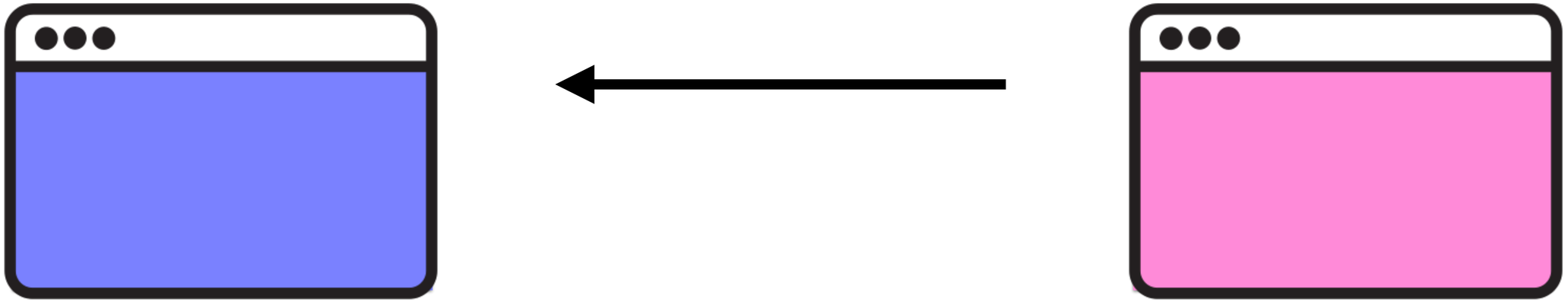
“Hi, I’m Alice!”



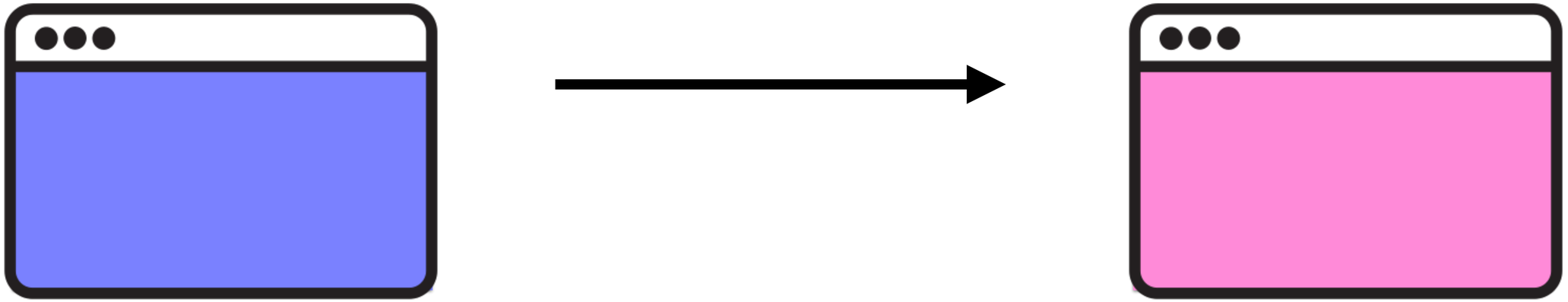
“Hey there, I’m Bob!”



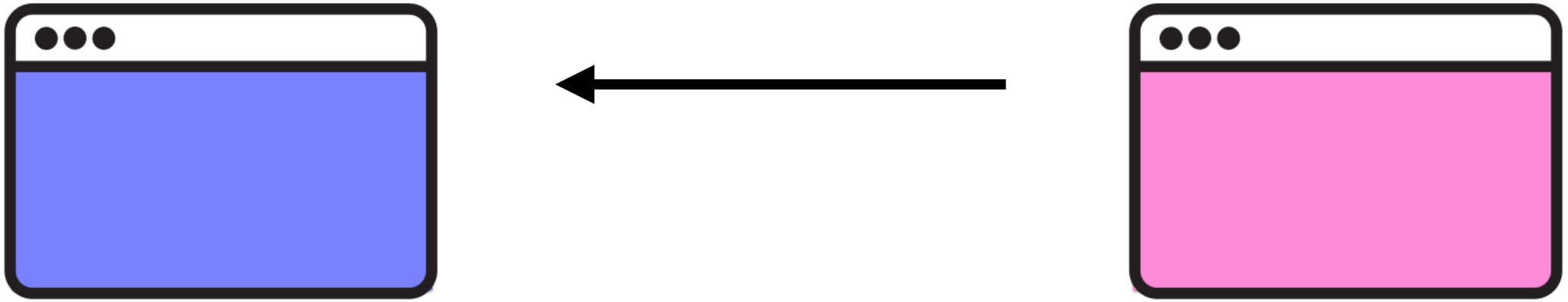
“Hey there, I’m Bob!”



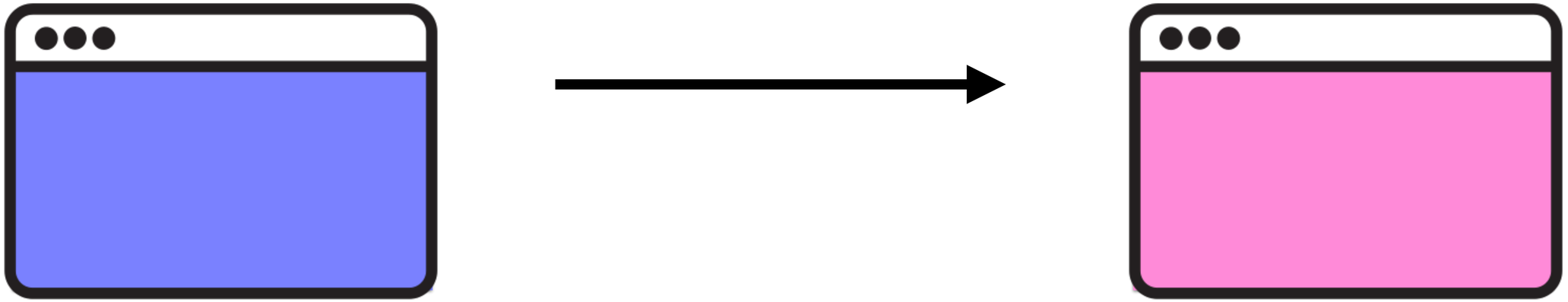
“I need to bill Charles $34.”



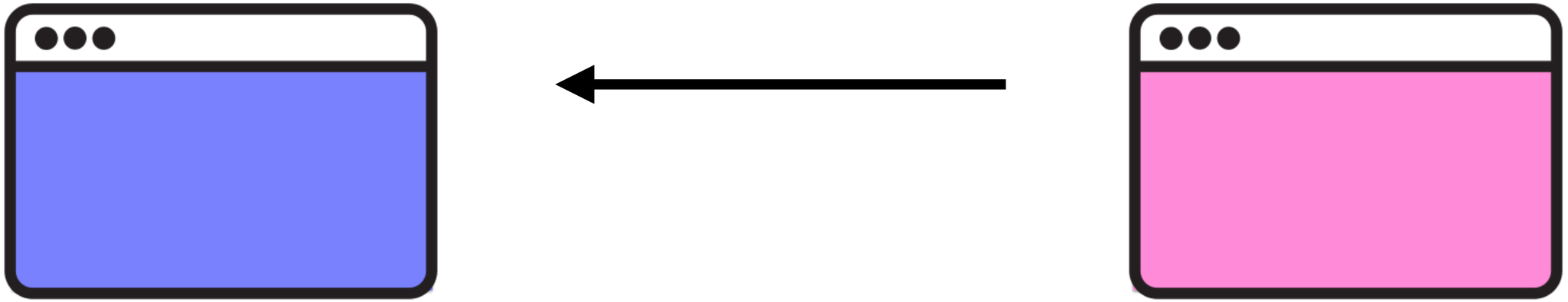
“Okay! Charles has been billed $34”



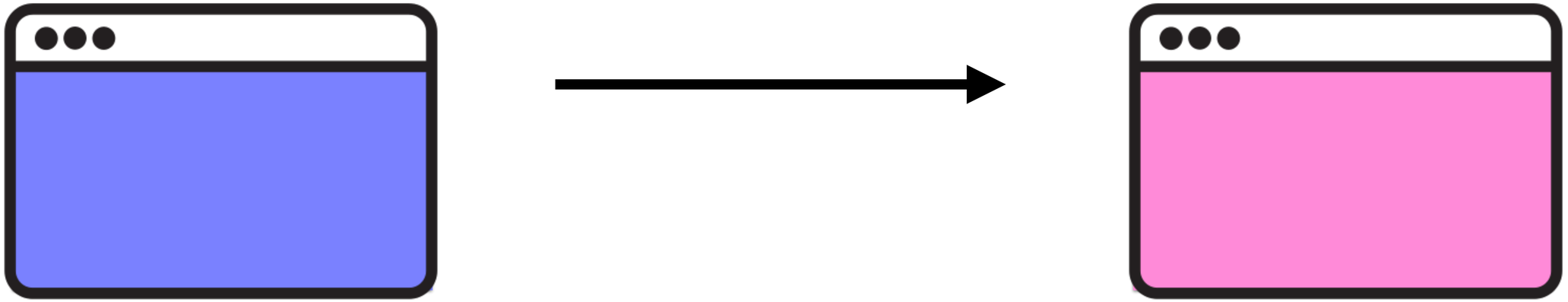
“Hi, I’m Alice!”



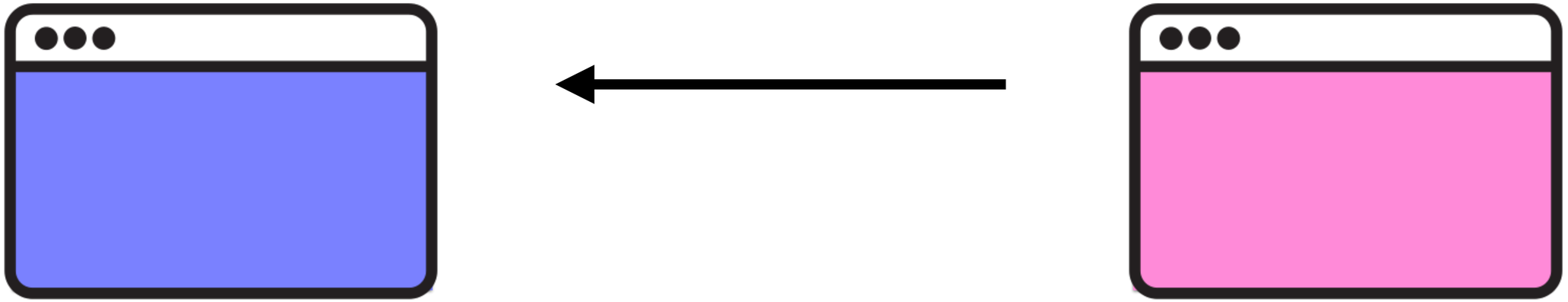
“No, I don’t think you are.”



“I need to bill Charles $34.”



“Sorry, Alice, you can’t do that.”



App-Level Security

- All connections have *identity*
- All connections are *encrypted*
- All requests are checked for *permission*

Enables *efficient* growth.

Done in addition to perimeter security.

Challenges

- TLS is hard
- Identity is hard
- Modifying apps is hard

Solutions

App-Level Security

Solutions

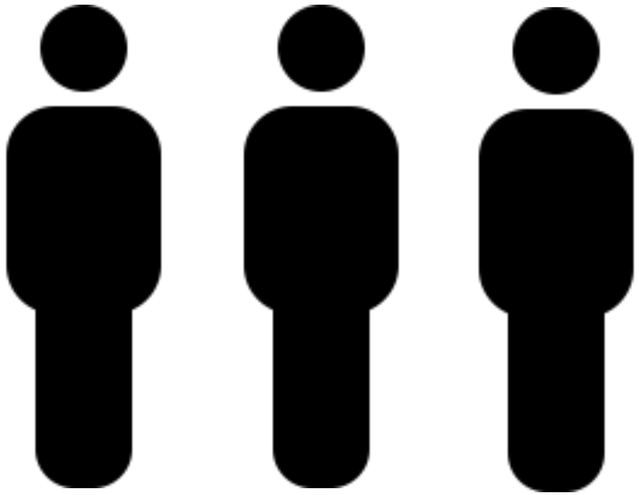
- “Security Server” (Vault)
- Scheduler can orchestrate setup
  - Secure introduction

DevOps

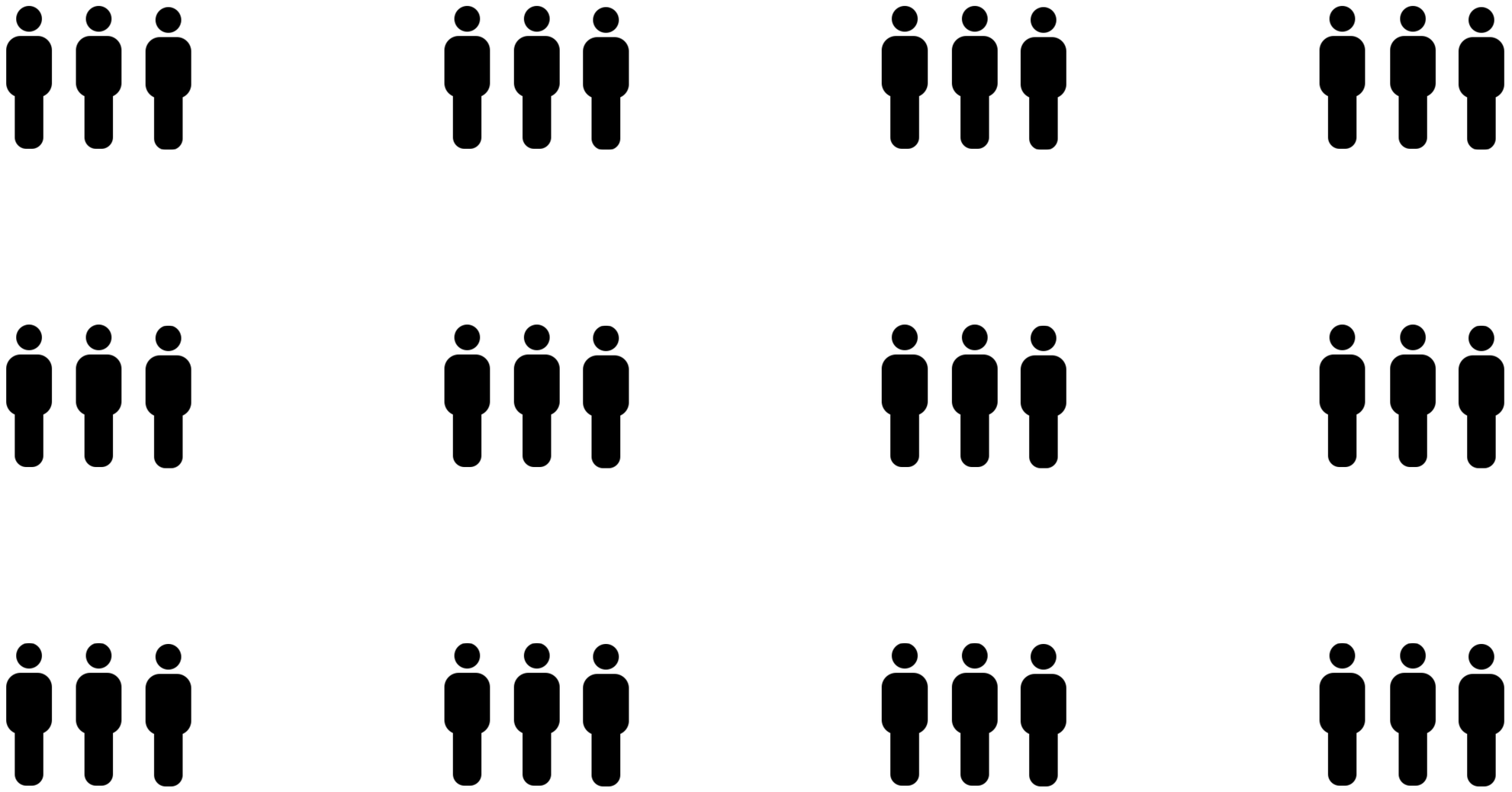
Idea: Dev, Ops, Sec must be able to work together to streamline deployment.

“DevOps”


Dev
Ops
Sec



Dev Ops Sec



DevOps

- Increased efficiency through parallelization of work.
- Requires both tooling and cultural change.

Our **Glorious Future**

Globally distributed set of available resources, available on demand.

(Cloud, Heterogeneity)

Utilizing this in most cases is as simple as “double click to run.”

(Schedulers, DC as Computer)

Build and deploy by describing what you want, not how to do it.

(Declarative)

App-to-app secured
communications with
a known identity.

(App-Level Security)

Deploying 10s, 100s, 1000s of apps quickly and efficiently.

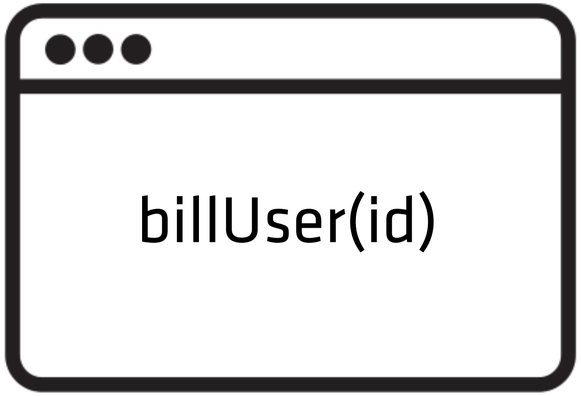
(DevOps)

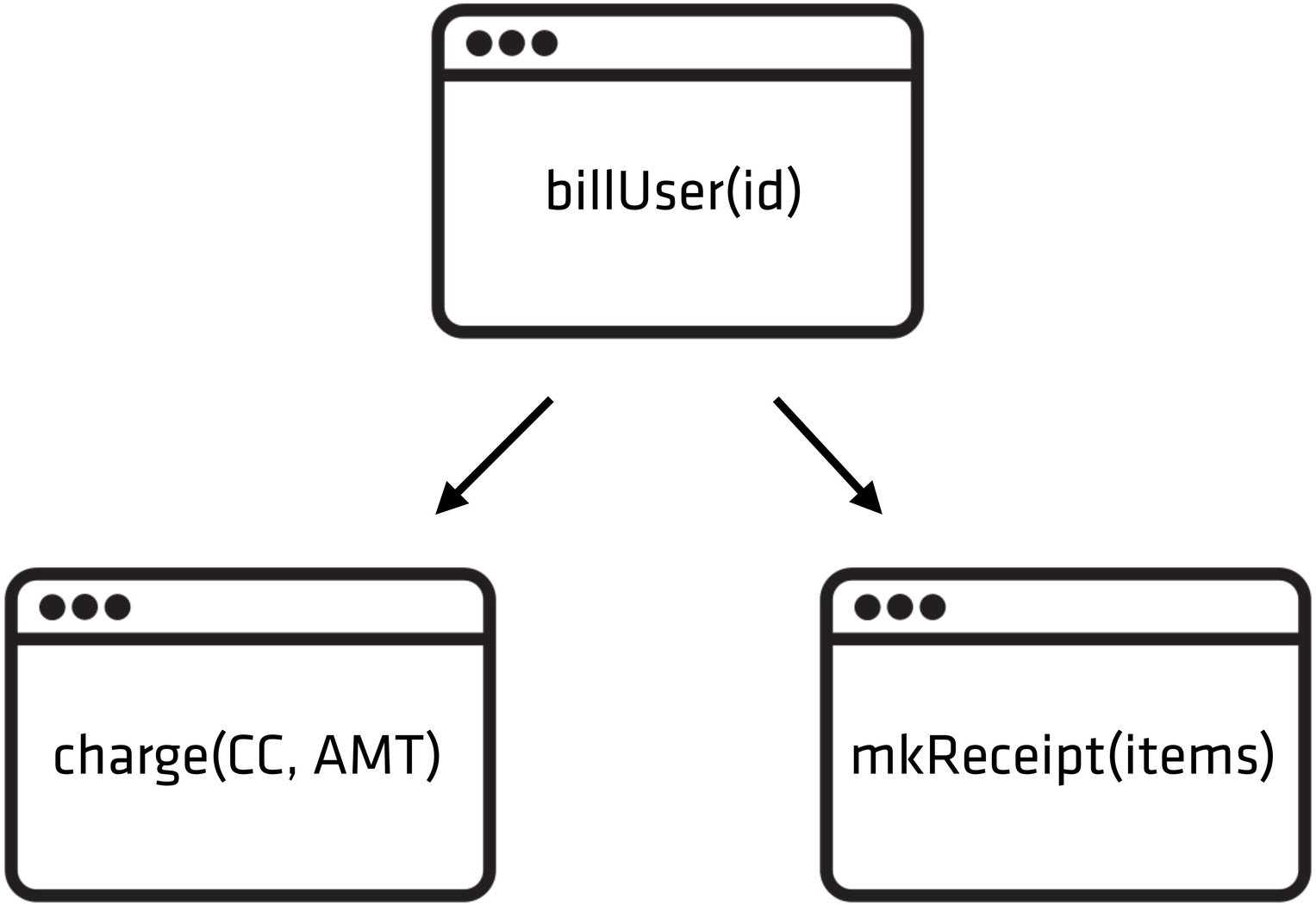
NOTICE: Workflows,
not technologies.

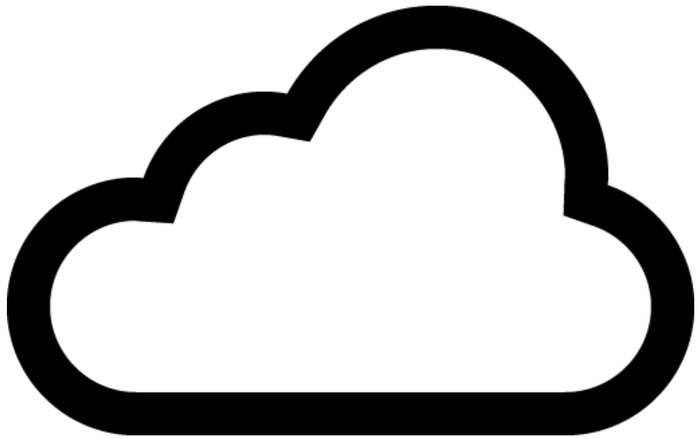
Future future…

Serverless

Idea: Upload small units that manifest the compute to realize the result upon invocation.


billUser(id)

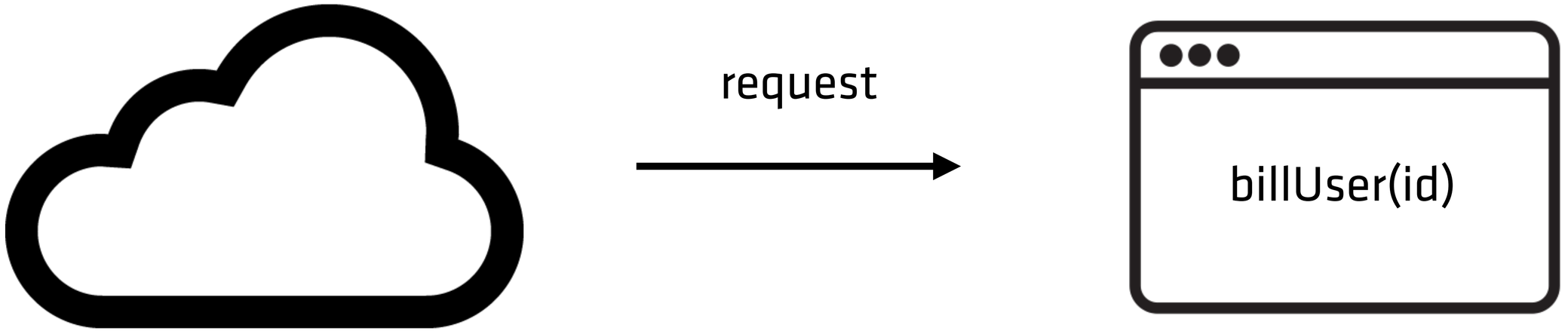

billUser(id)
charge(CC, AMT)
mkReceipt(items)

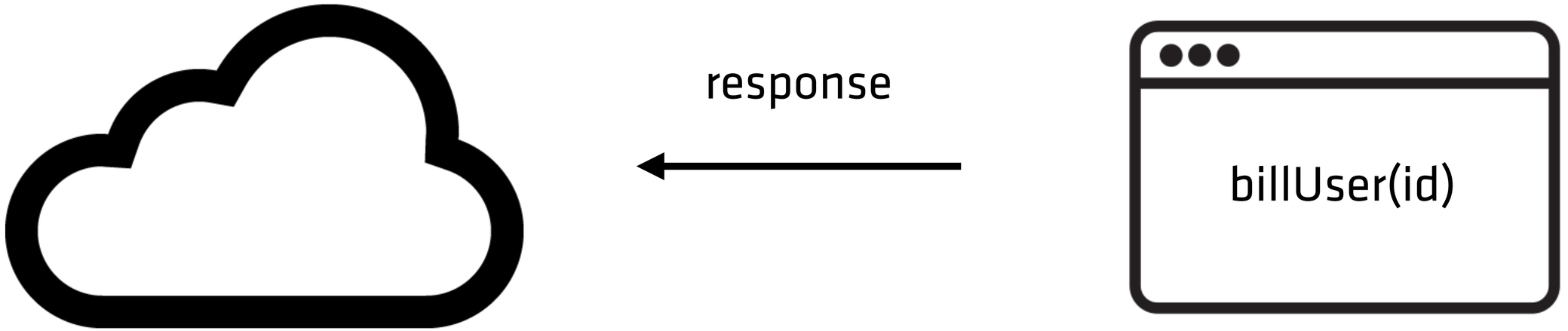


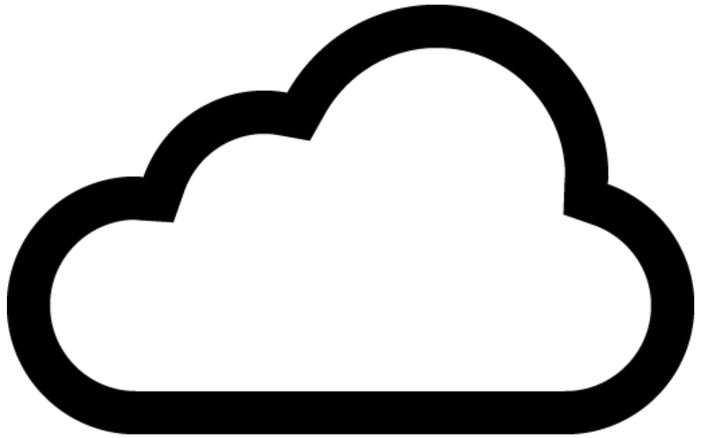
request



???


request
billUser(id)


response
billUser(id)



“Serverless”

“Lies, there are still servers.”

- Grumpy Fellow
(Also heard yelling
“get off my lawn” on the way out)

“Serverless”

- Of course there are servers!
- An unimaginably large number of servers.
- But… not **MY** servers.

“Serverless”

“It is just a PaaS.”

- Grumpy – but more reasonable – Fellow

“Serverless”

- Smaller unit vs. entire application
- Introduces new challenges and features that a platform *could maybe* do, but it makes this probably particularly interesting.

“But… why?”

Why Serverless?

- App dev is now more time consuming than deploying the application.

Why Serverless?

- App dev is now more time consuming than deploying the application.
- Microservices

Why Serverless?

- App dev is now more time consuming than deploying the application.
- Microservices
- Cost

Also, **why not?**

Potential Applications

- Microservices
- Business Intelligence
- Batch Jobs

What does it mean for ops?

- In the short term: mostly nothing
- Too many unknowns still
- Heterogeneity: everything else will still exist
- Servers still run underneath Serverless


THANK YOU!
@mitchellh
HashiCorp