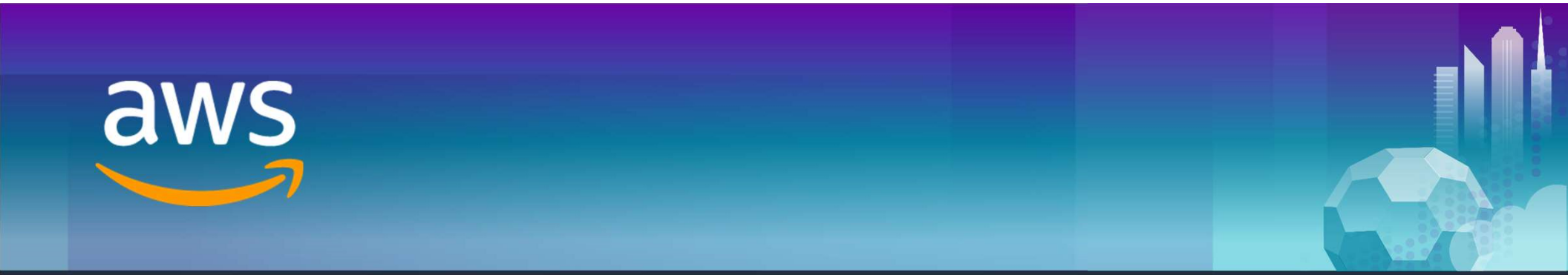
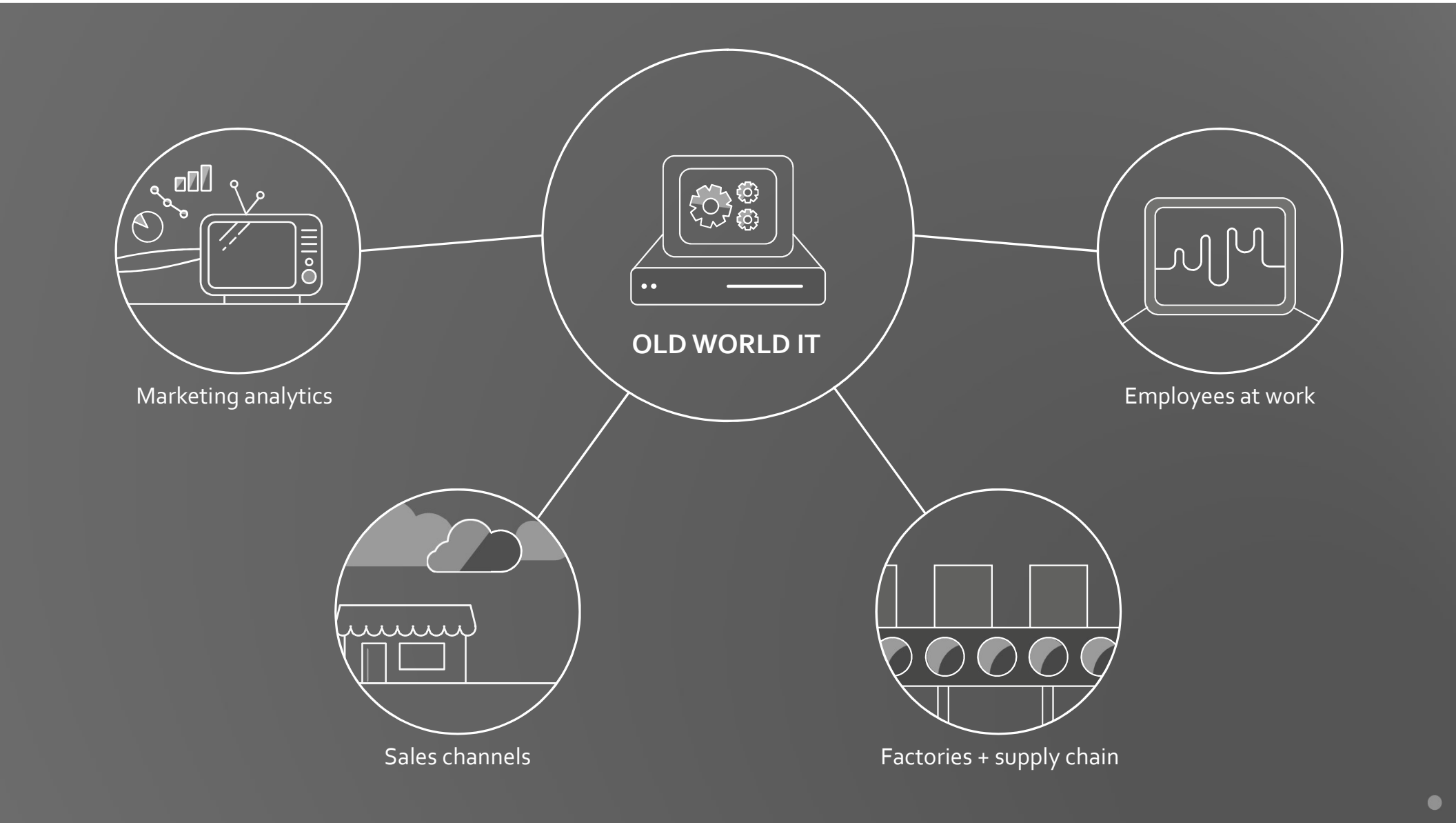
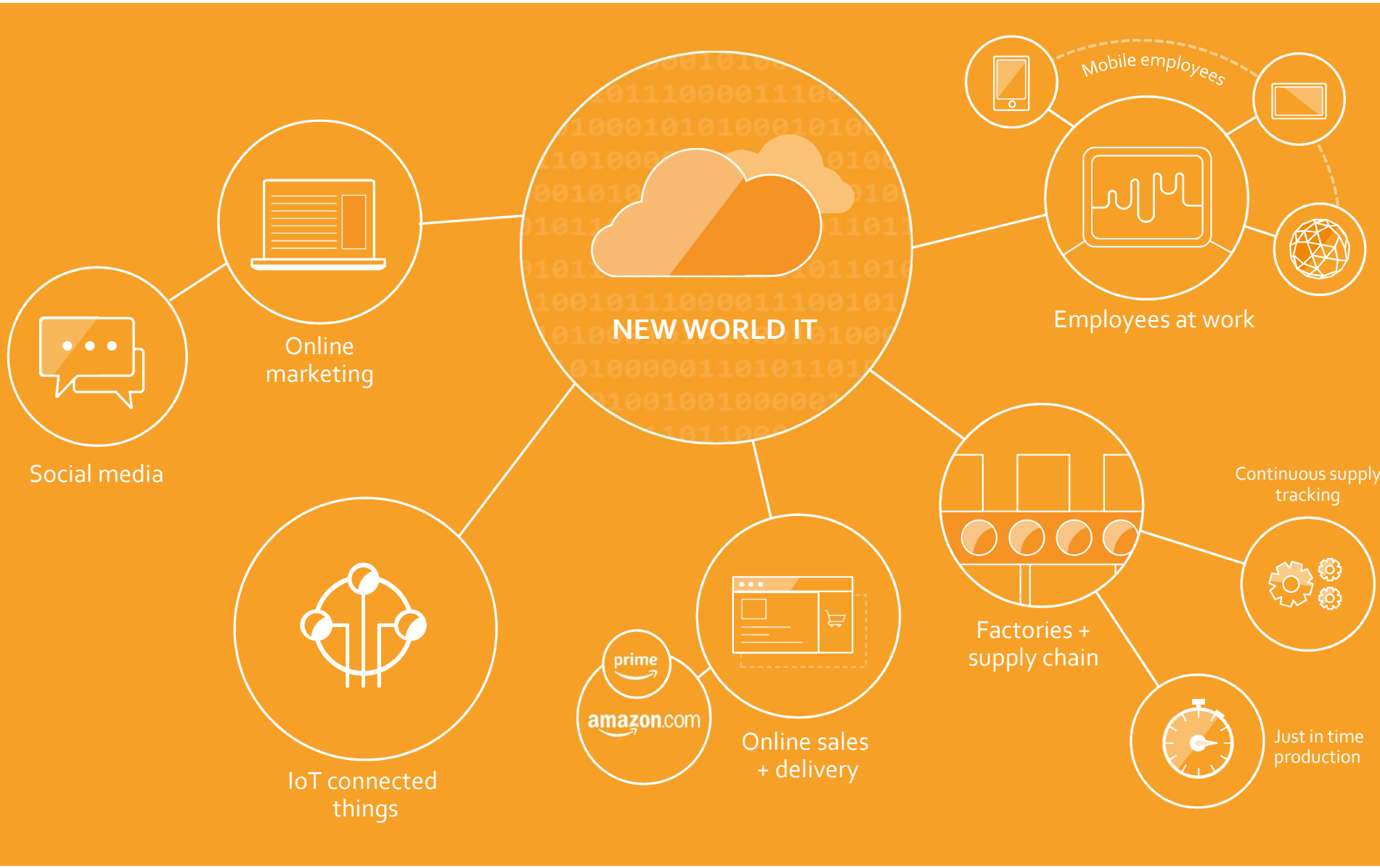


AWS: Unblocking Innovation for Digital Transformation

Nicolas Vautier

Head of Solutions Architecture, Taiwan


OLD WORLD IT
Marketing analytics
Employees at work
Sales channels
Factories + supply chain


NEW WORLD IT
Mobile employees
Employees at work
Online marketing
Social media
IoT connected things
prime
amazon.com
Online sales + delivery
Factories + supply chain
Continuous supply tracking
Just in time production

New Needs

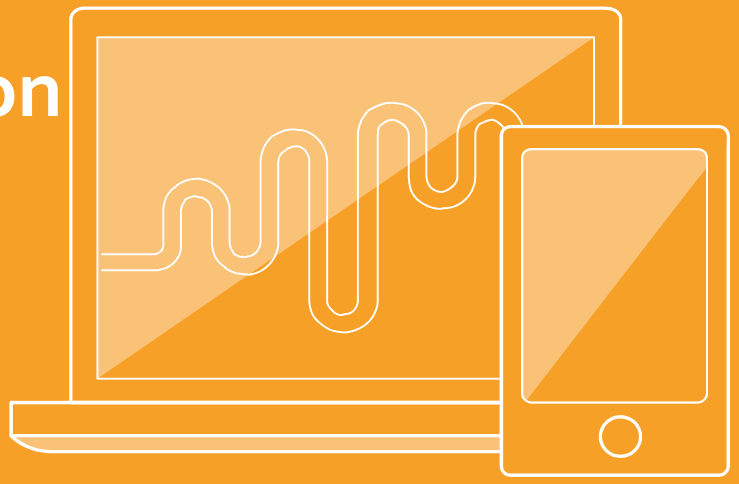
Personalization

Customer tracking

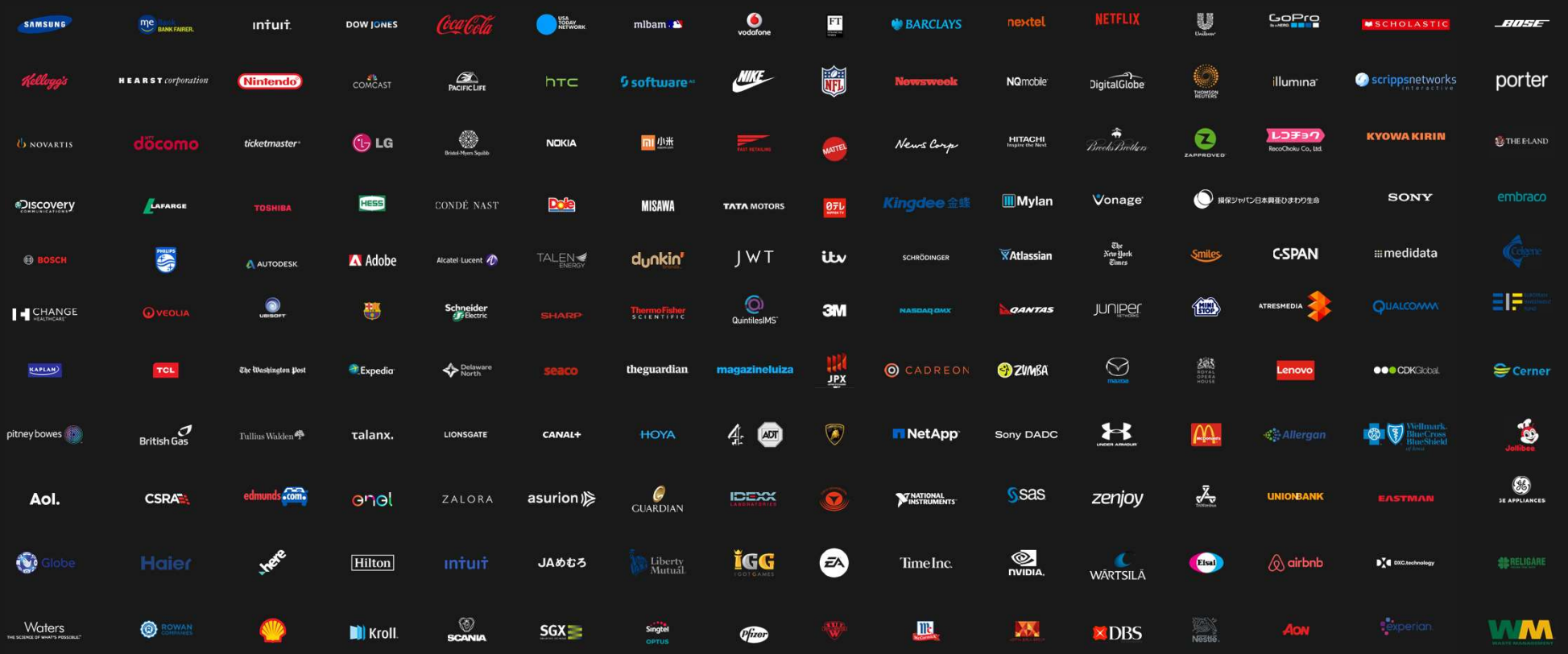
New channels direct to customer

More things, more scale, rapid change

AWS: Unblocking Innovation for Digital Transformation with Enterprise Customers

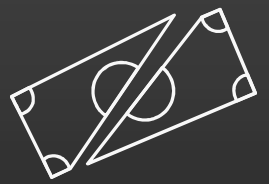


AWS Enterprise Customers



Digital Transformation Is Key To Survival For Enterprises

Blockers for Innovation

Culture	Skills	Organization	Finance
			
Leadership Systems and Feedback	Training and Compensation	Move from Projects to Product Teams	Capex Versus Opex

Leadership Systems and Feedback

Centralized decision making

Lack of trust

Inflexible policies and processes

Training and Compensation

Train existing staff on cloud tech

Fund pathfinder teams

Be prepared to shift pay structure around

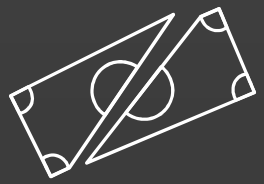
Move from Projects to Product Teams

Long term product ownership

Continuous delivery

DevOps and “run what you wrote”

Reduce tech-debt and lock-in



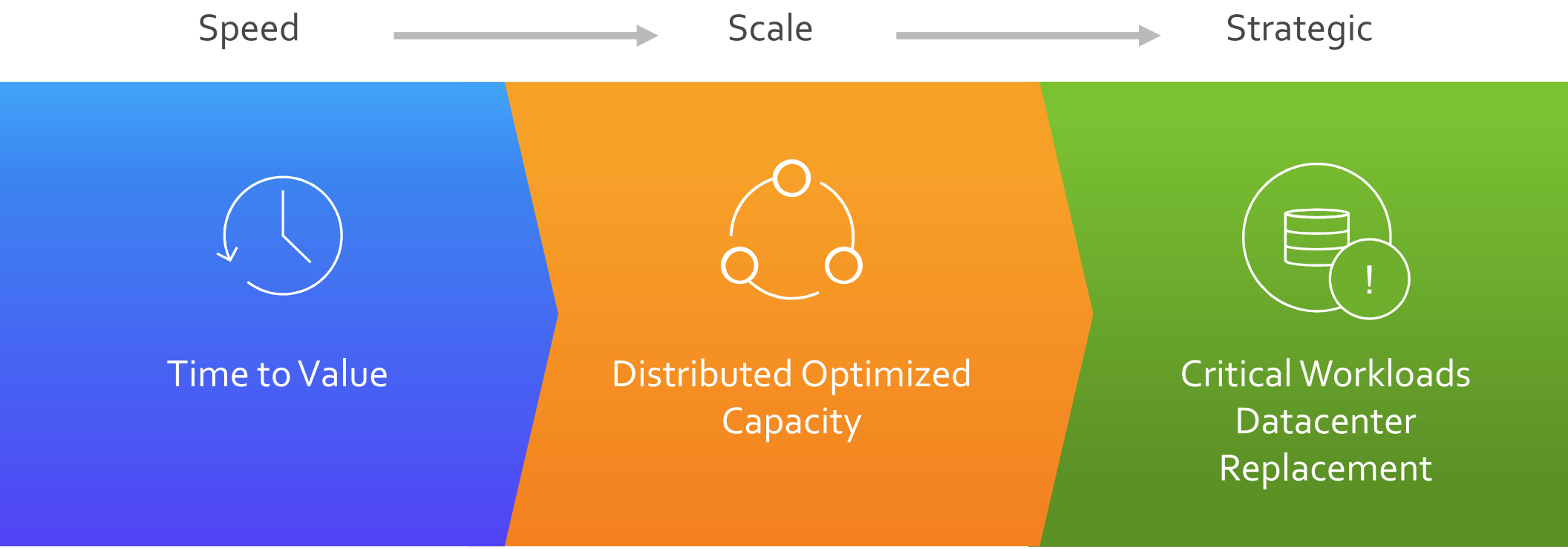
Capex Versus Opex

Datacenter to Cloud

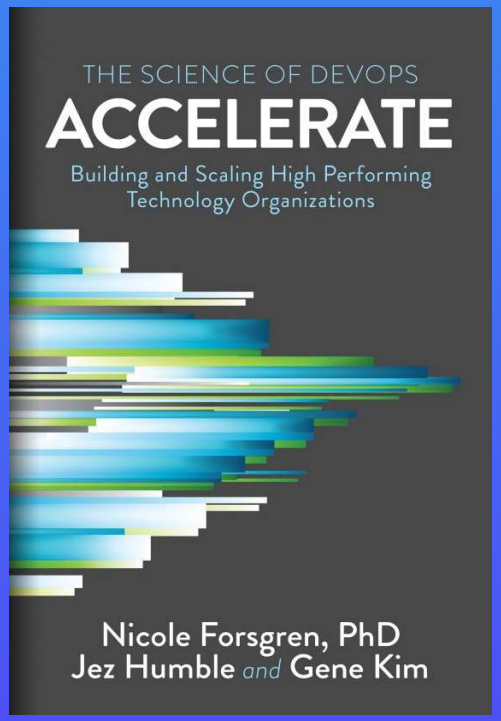
Understand the impact

Plan ahead, don’t surprise the CFO

Pathway for Digital Transformation



Time to Value



https://itrevolution.com/book/accelerate/

The fast companies are **440x** faster than the slow

We found that, compared to low performers, high performers have:

46 times more frequent code deployments

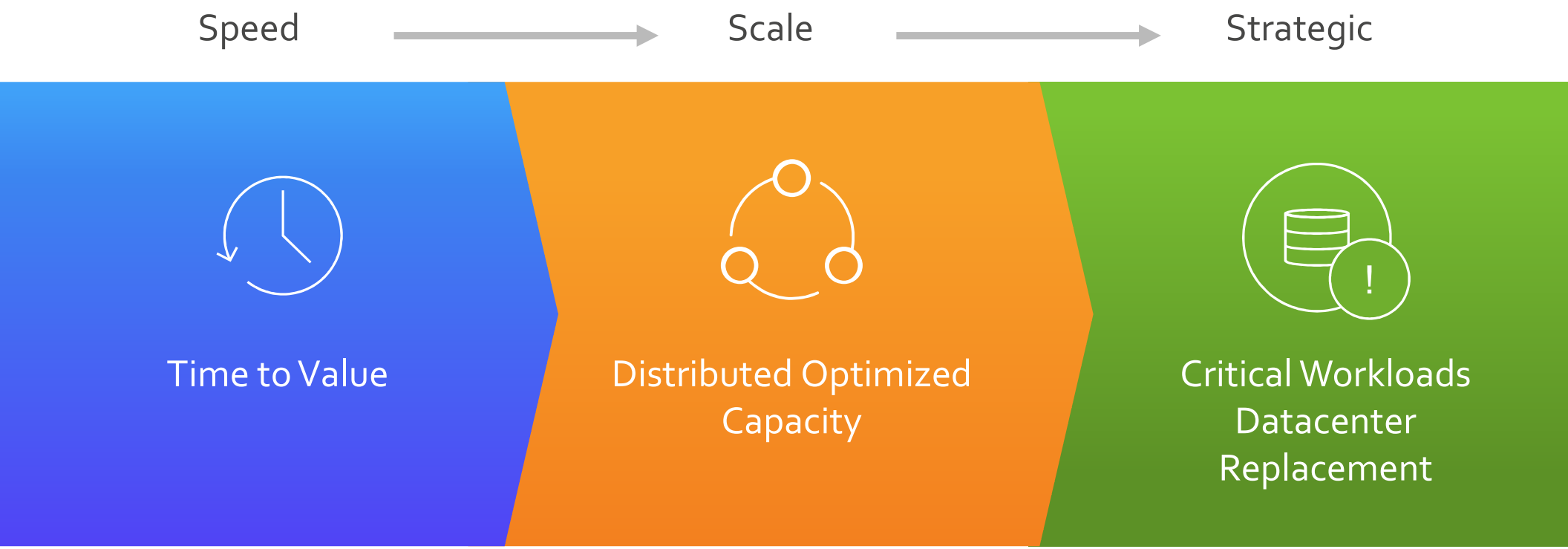
440 times faster lead time from commit to deploy

170 times faster mean time to recover from downtime

5.0 times lower change failure rate (1/5 as likely for a change to fail)

Months ⇨ Hours

Pathway for Digital Transformation



Distributed Optimized Capacity

Highly Scaled

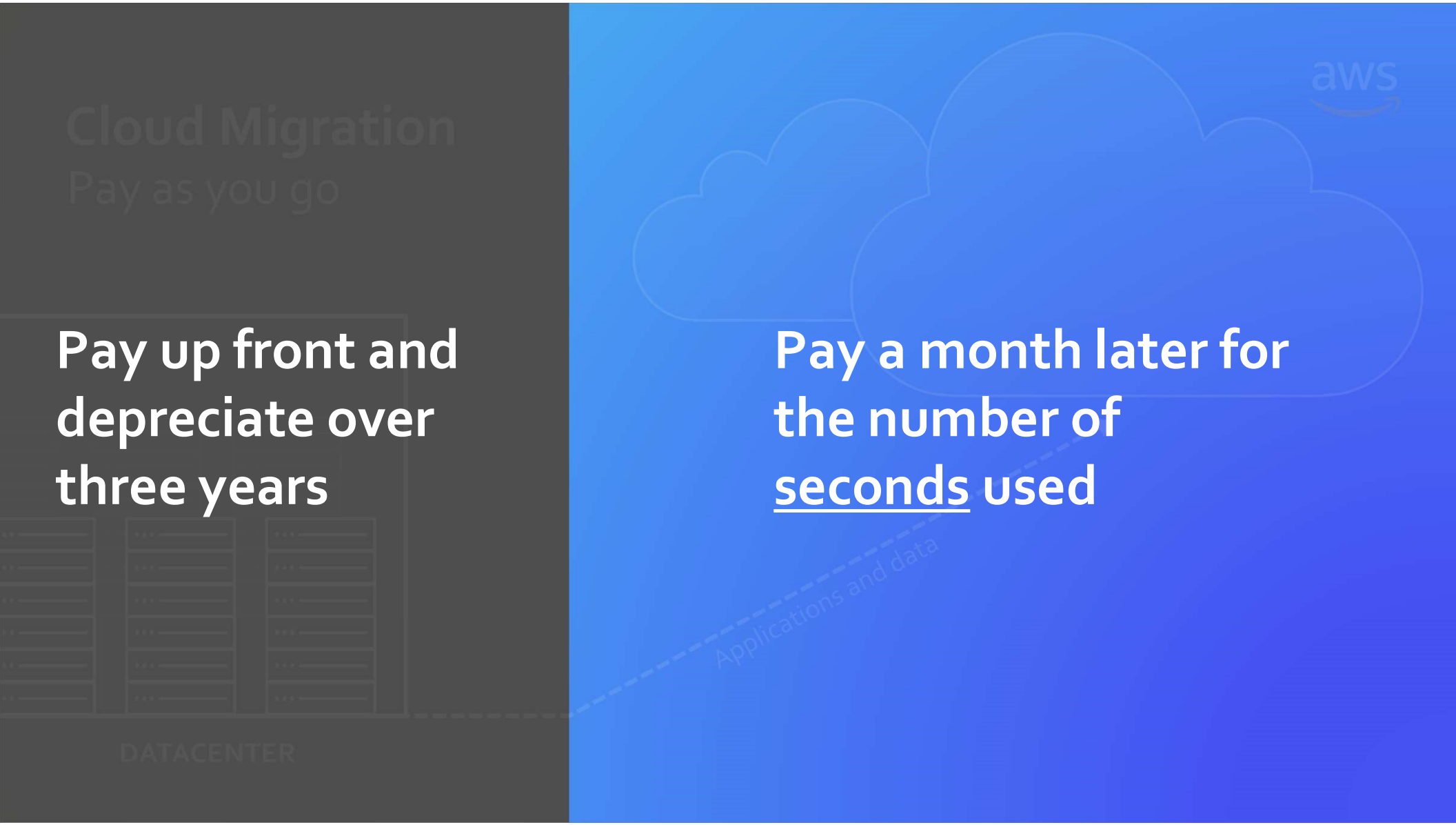
Distributed for Availability

Cost Optimized High Utilization

Cloud Native Architecture

Cloud Native Architecture

Principles and Practices

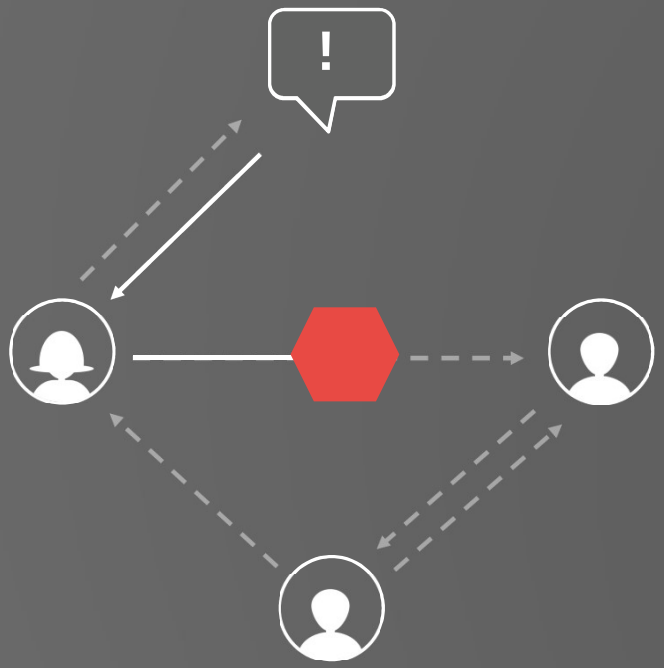

Cloud Migration
Pay as you go
Pay up front and depreciate over three years
DATACENTER
aws
Pay a month later for the number of seconds used
Applications and data

Cloud Native Principle

Pay for what you used last month

Not what you guess you will need next year

File tickets and wait for every step



VS

Self service, on-demand, no delays

File tickets and wait for every step

Deploy by filing a ticket and waiting weeks or months

vs

Self service, on-demand, no delays

Deploy by making an API call self service within minutes

Cloud Native Principle

Self service, API driven, automated

Move from request tickets at every step to a tracking ticket that records what happened

Cloud Native Principle

Instant globally distributed deployments and data by default

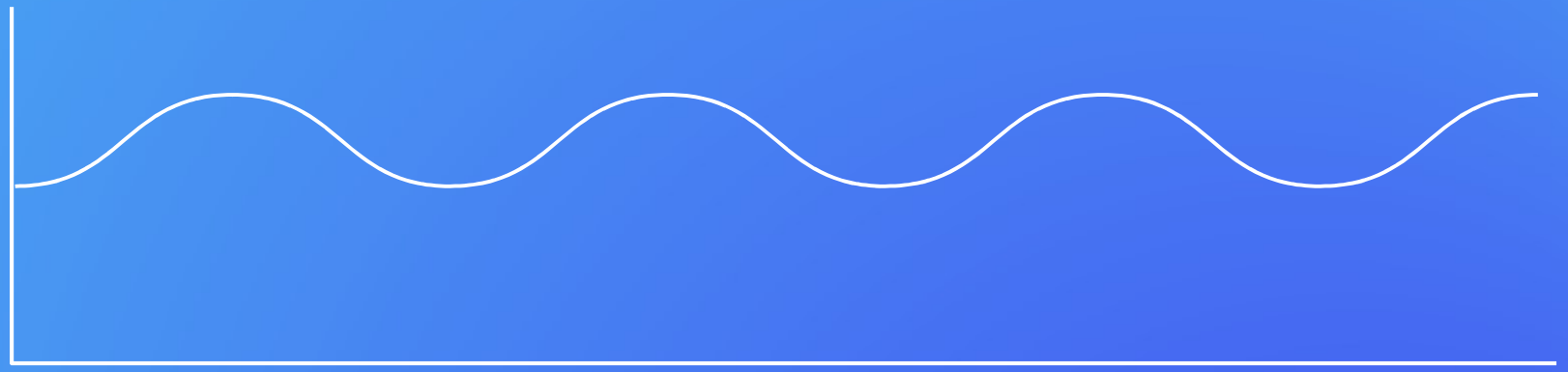
Elasticity

DATACENTER

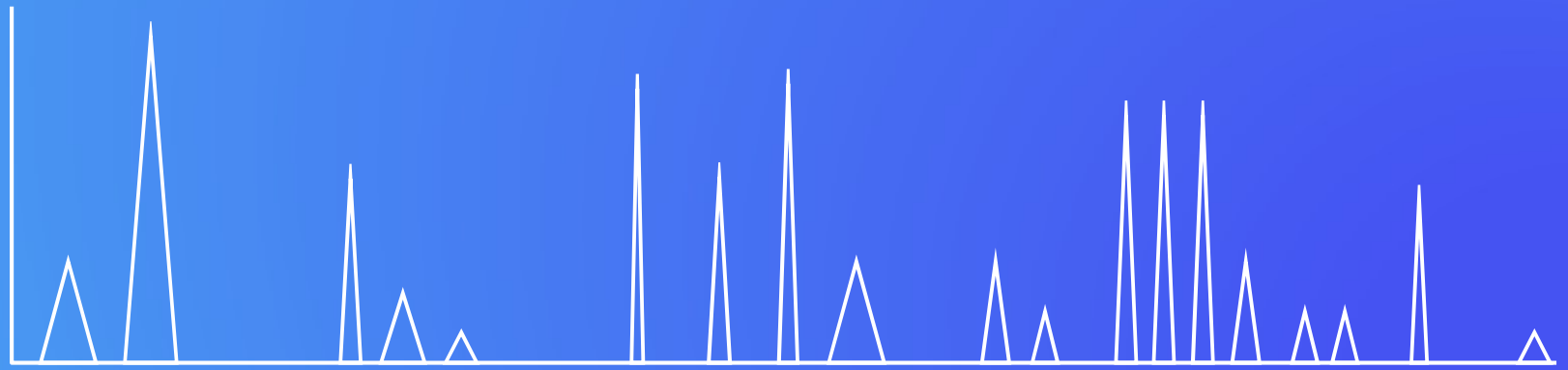
Hard to get over 10% utilization— need extra capacity in case of peak

CLOUD

Target over 40% utilization— no capacity overload issues



Autoscaling for predictable heavy workloads



Serverless for spiky workloads with idle periods

Cloud Native Principle

Turn it off when it’s idle

Many times higher utilization
Huge cost savings
Avoids capacity overloads

Cloud Native Principle

Modern DevOps

Automated builds
Ephemeral instances, containers, and functions
Blue–Green deployments

In Summary …

Cloud Native Principles

Pay as you go, afterwards

Self service—no waiting

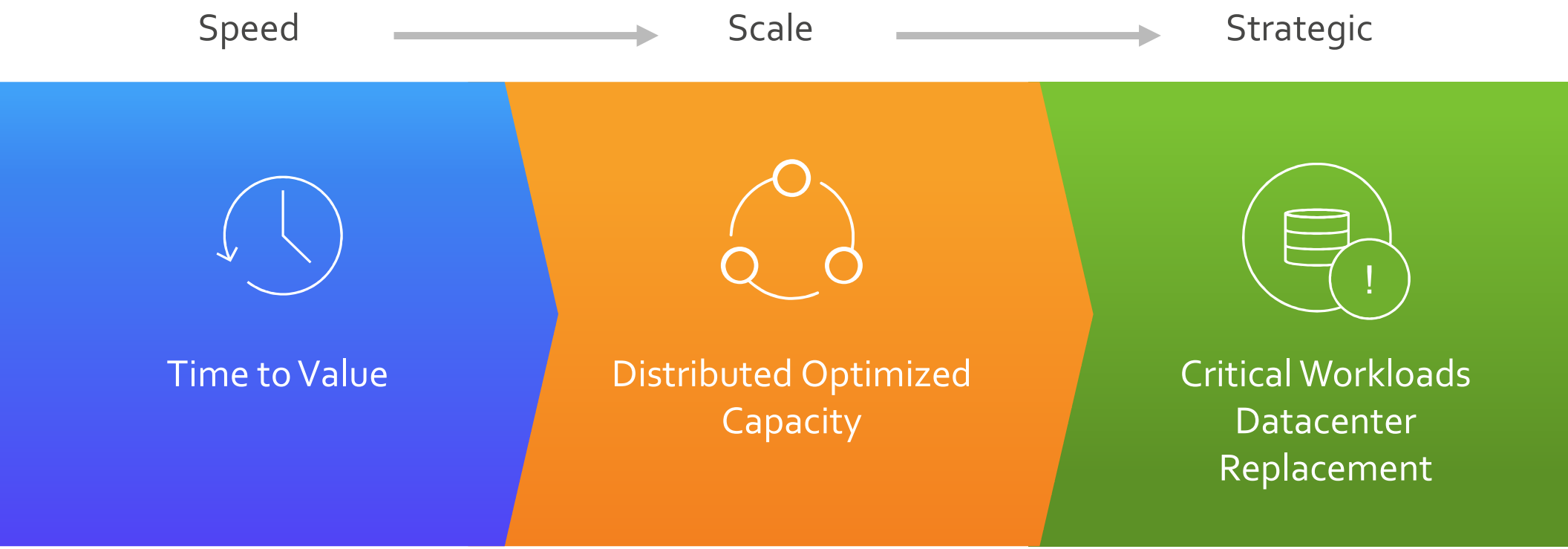
Globally distributed by default

Cross-zone/region availability models

High utilization—turn idle resources off

Immutable code deployments

Pathway for Digital Transformation



Critical Workloads Datacenter Replacement

Core Banking

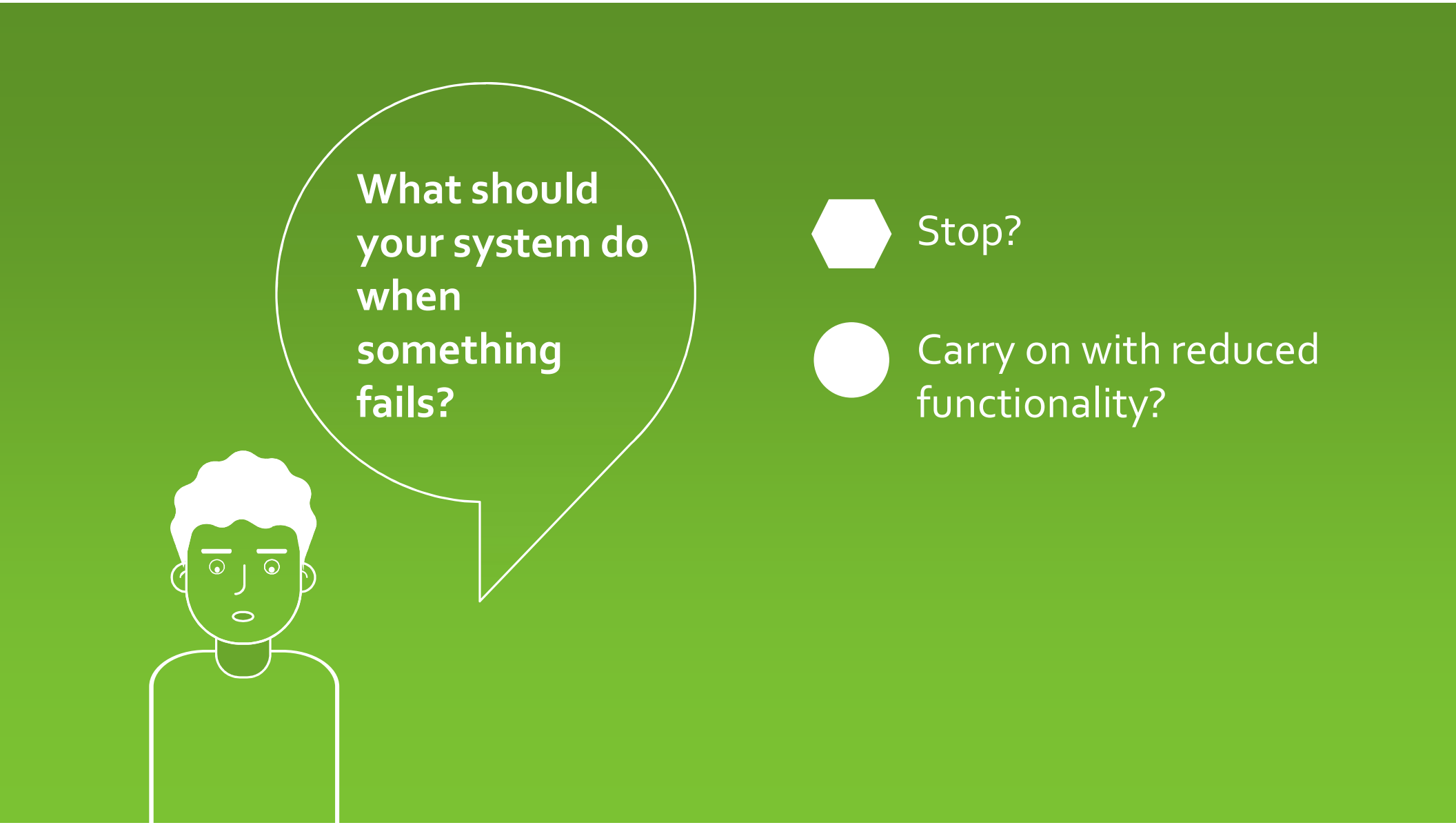
Industrial Control Systems

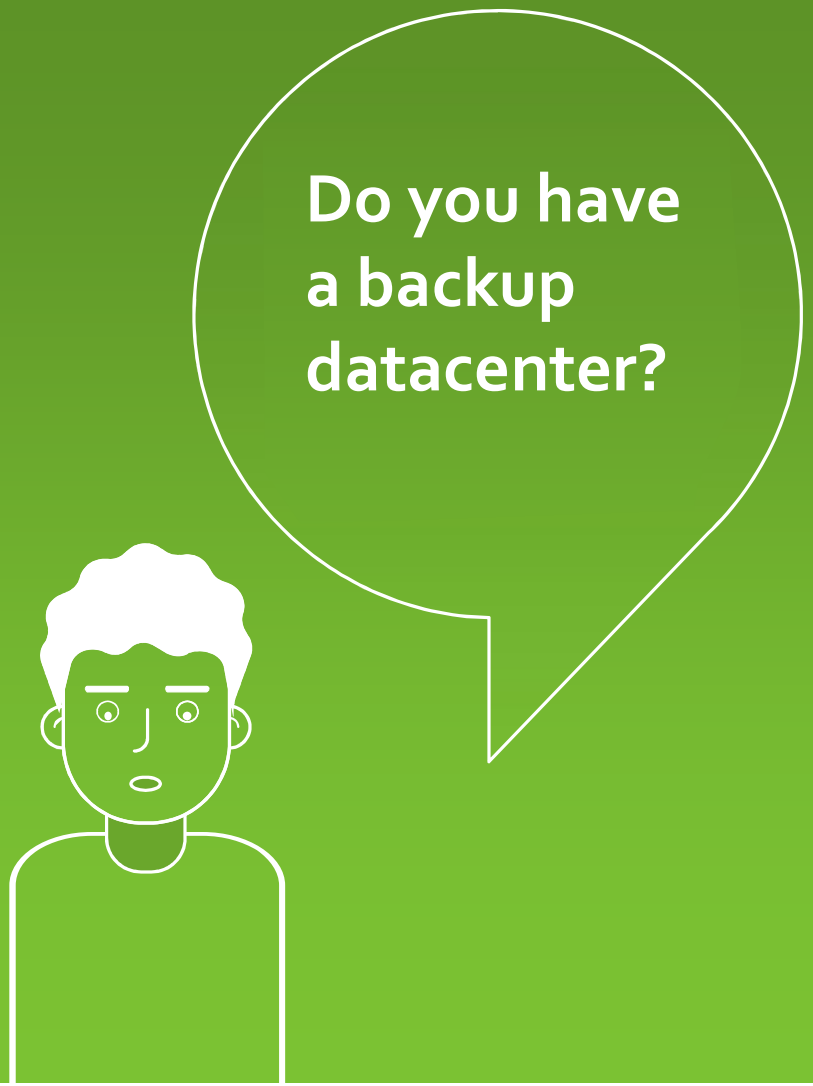
Transport

Healthcare

How do you know if you have a good architecture for critical systems?

Ask some awkward questions…


What should your system do when something fails?
Stop?
Carry on with reduced functionality?



How often do you failover apps to it?

How often do you failover the **whole datacenter** at once?

"Availability Theater"

A fairy tale…

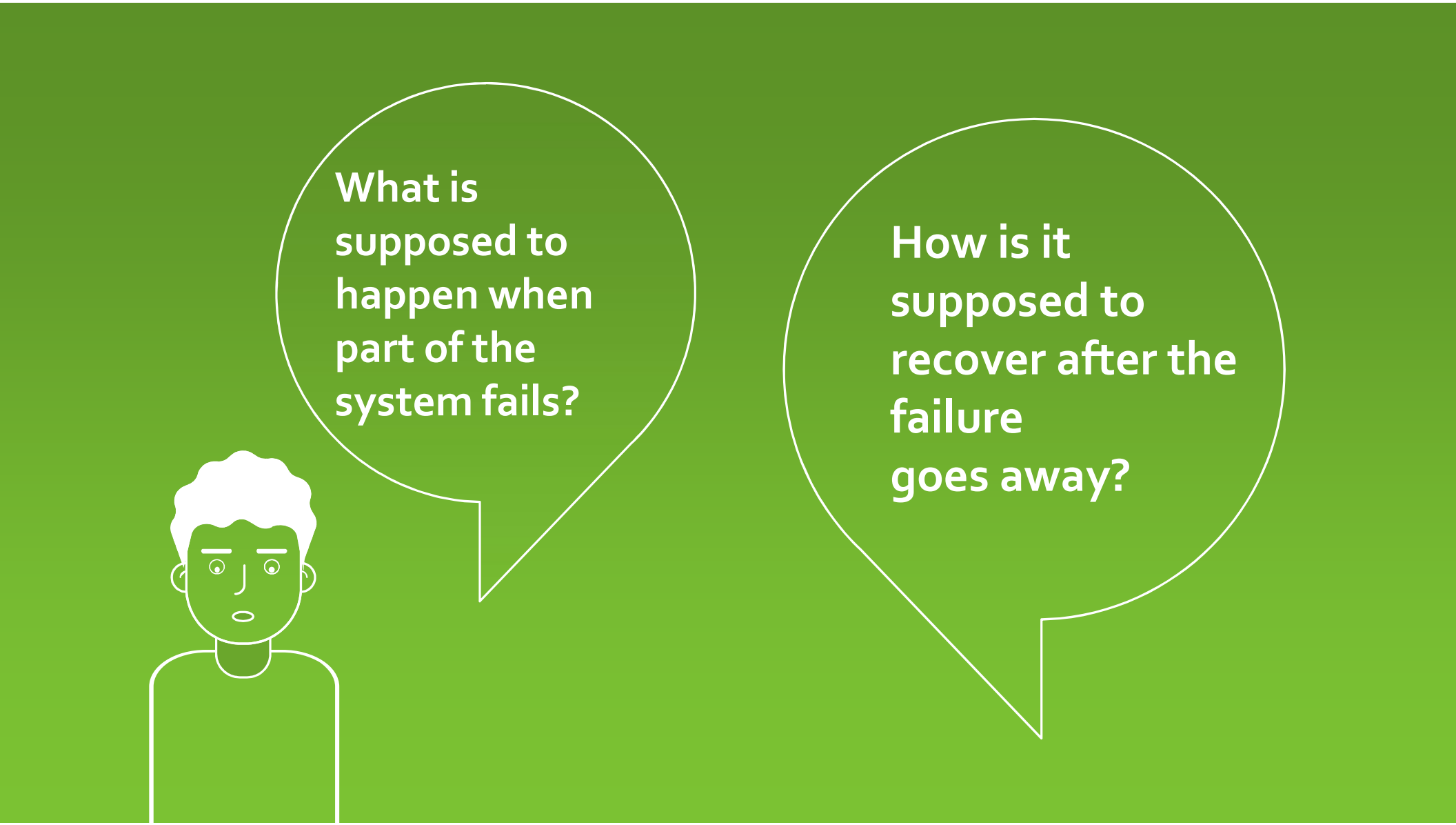
Once upon a time, in theory, if everything works perfectly, we have a plan to survive the disaster we thought of in advance

How did that work out?

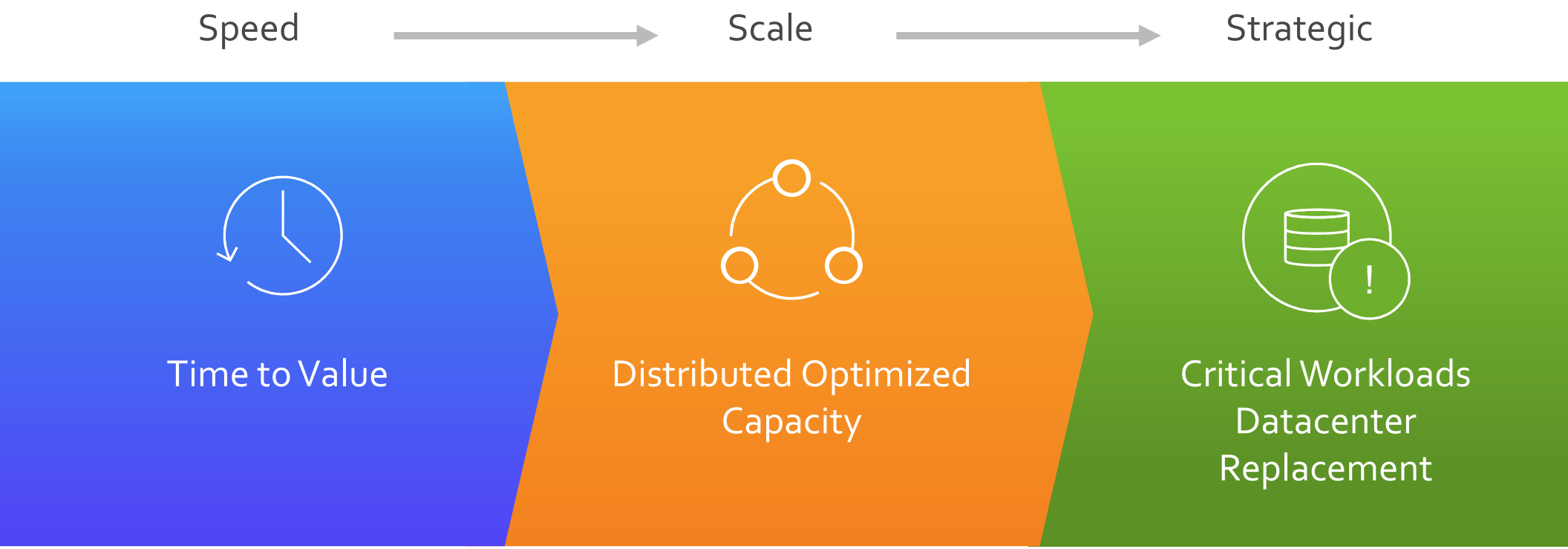
Forgot to renew domain name…	SaaS vendor
Didn’t update security certificate and it expired…	Entertainment site
Datacenter flooded in hurricane Sandy…	Finance company, Jersey City
Whoops!	**YOU, tomorrow**

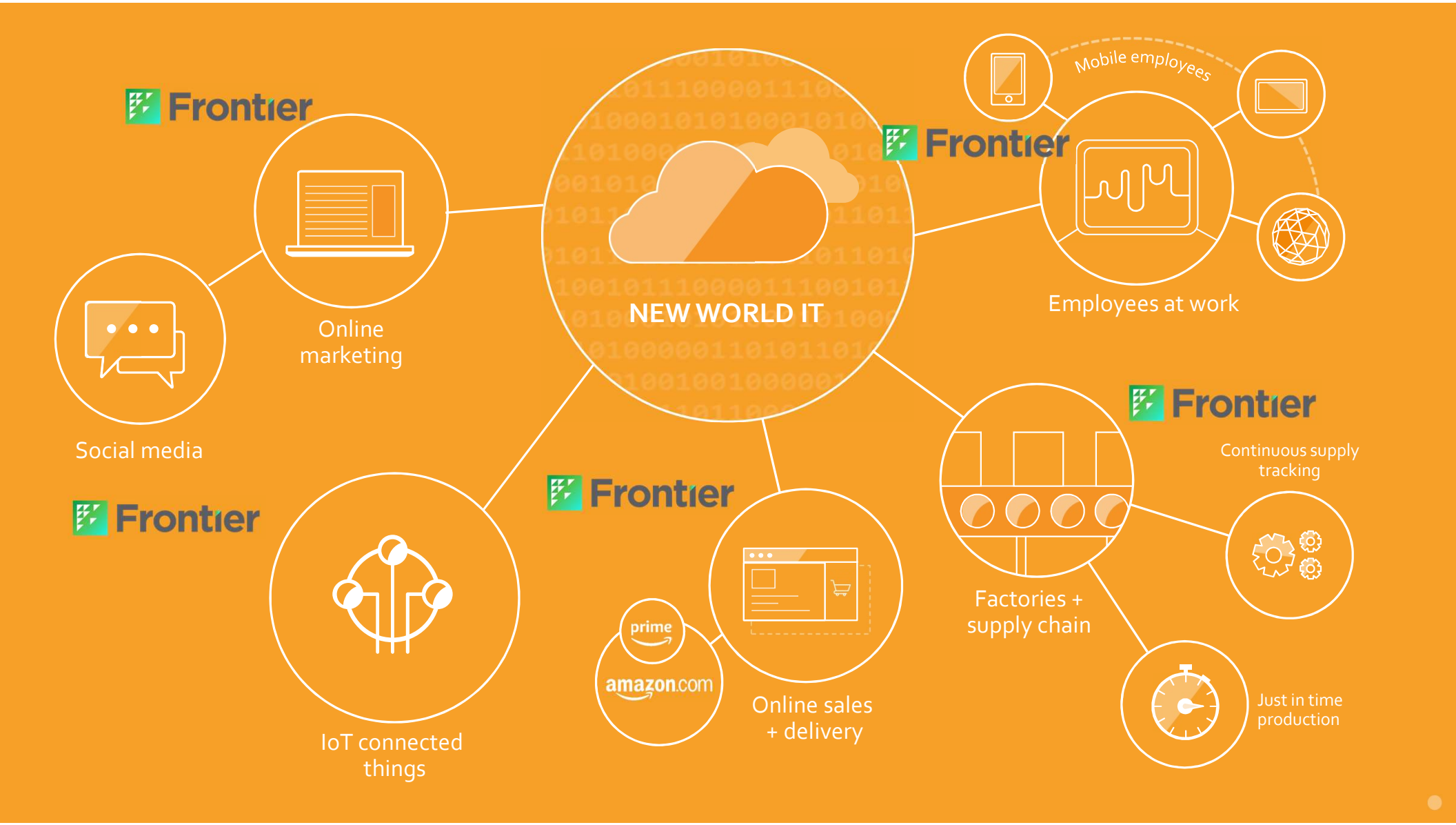
“You can’t legislate against failure, focus on fast detection and response.”

—Chris Pinkham


What is supposed to happen when part of the system fails?
How is it supposed to recover after the failure goes away?

Pathway for Digital Transformation




Frontier
Online marketing
Social media
Frontier
IoT connected things
NEW WORLD IT
Frontier
prime
amazon.com
Online sales + delivery
Frontier
Mobile employees
Employees at work
Frontier
Factories + supply chain
Continuous supply tracking
Just in time production

Thank you!

Nico Vautier
nvautier@amazon.com
https://www.linkedin.com/in/nicolasvautier/

Session Survey