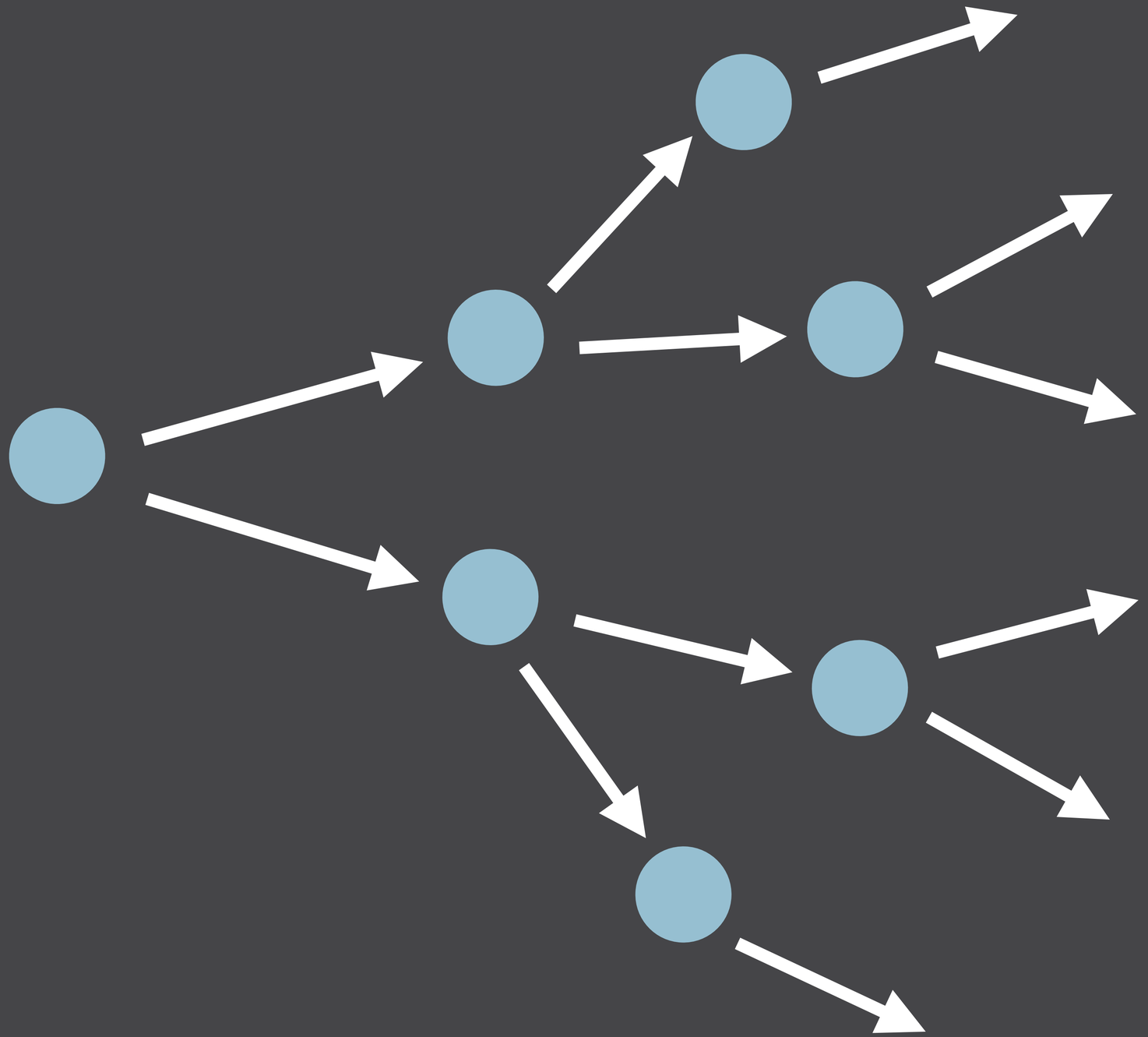
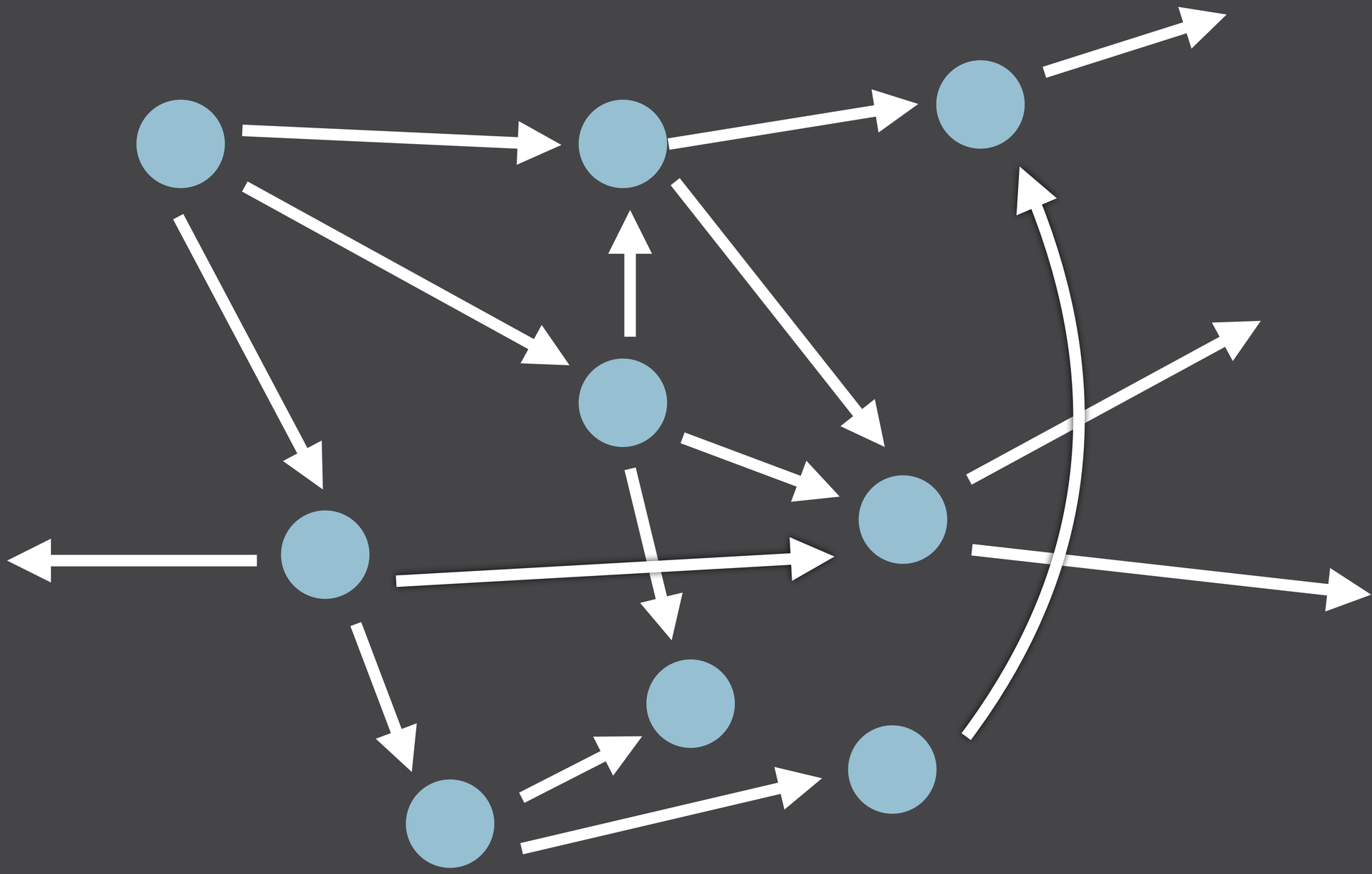
No Microservice Is An Island

Michele Titolo

Square

Microservices Are A Distributed System





Something

WILL PROBABLY

GO WRONG

How Do You Figure Out What's Wrong?

How Do You Deploy A Fix?

New Challenges

Microservices Are More Than Application Size

Failure Points++

Microservices Need An Ecosystem

Apps Must Contribute

Infrastructure Must Contribute

Otherwise You'll Play Catch Up

Three Biggest Challenges

Deployment

Scaling

Debugging

Create A Foundation

Deploying Microservices

Characteristics

Small Releases

Frequent Releases

Consistent Releases

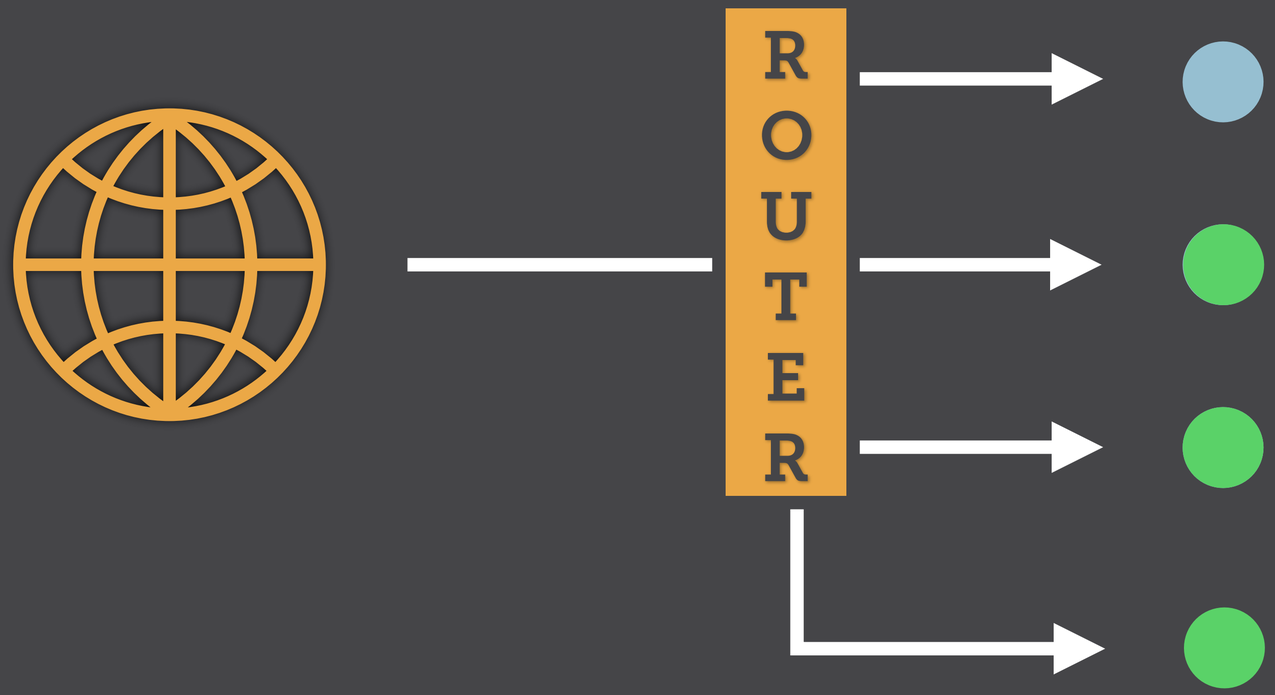
Limit Special Snowflakes

Top-Notch Tooling

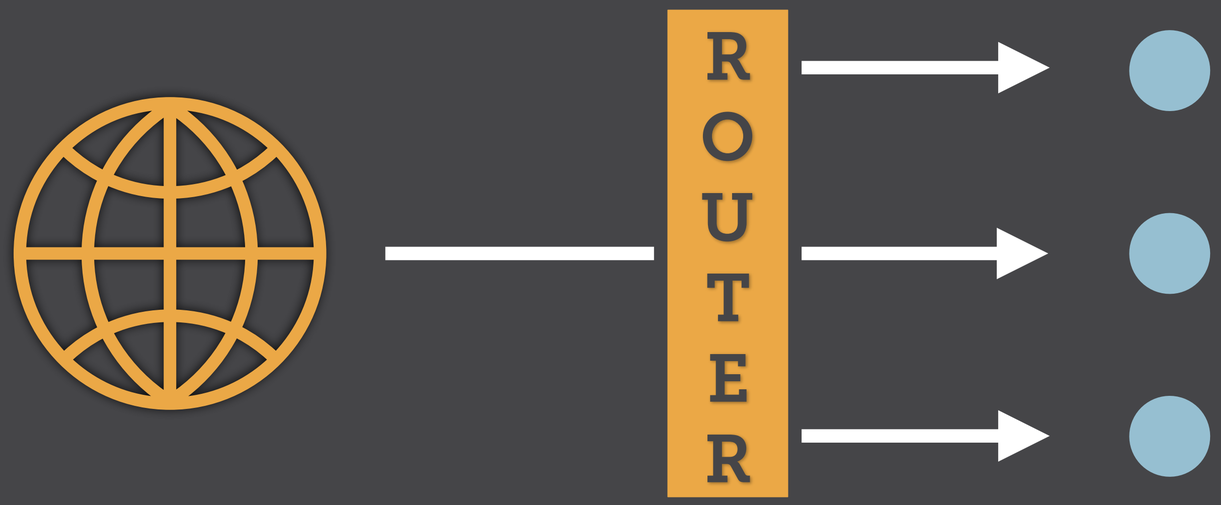
Automate Every Aspect

Staged Deployments

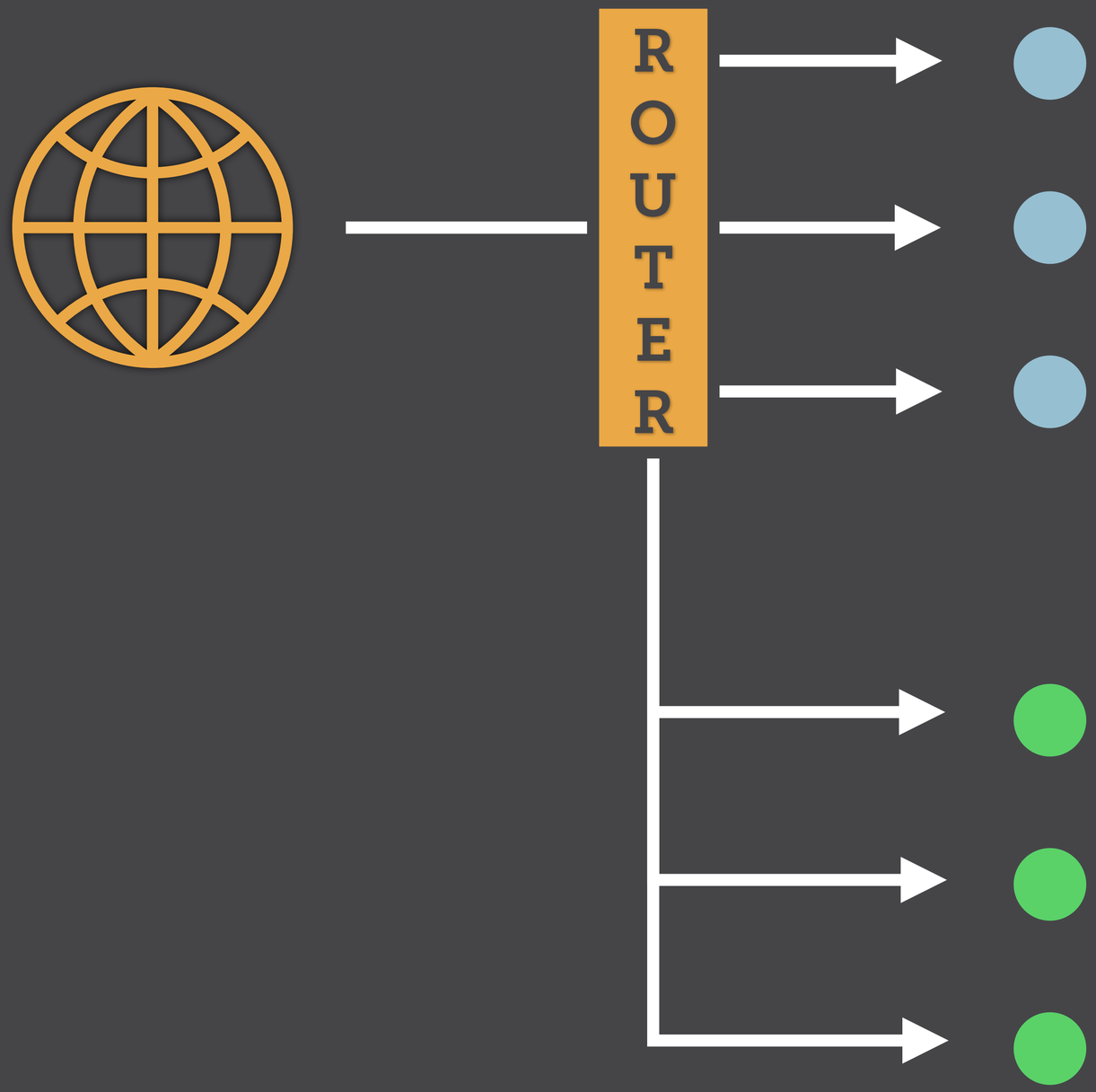
Canary


ROUTER

Blue/Green
Red/Black


ROUTER


ROUTER


ROUTER

Automatic Rollbacks

How Do You Know a Deployment Is Successful?

What Features or Tools Need To Exist?

App

Robust Tests

Unit And Integration

Frequent Deployments

Frequent Testing

Standardized Health Check

Consume Dependencies And Secrets

Dependencies i.e.
URL to database

Secrets i.e.
username/password to
DB

- Tests Pass
- Health Check
- Dependency/Secret Injection

System

Aggregate Health Checks

Show System State

Proactive

Status

Bots

Recap

Once a Deployment Is Finished, Everything's Done?

Distributed Systems Are Constantly Changing

Scaling Microservices

App Has More Or Less Load

Health Tracked

- RAM
- CPU
- Latency

Custom Metrics

How Do Microservices Scale?

Automation

Smart System

Trigger Scaling

Scale Up
Scale Down

Scale Up
Scale Down

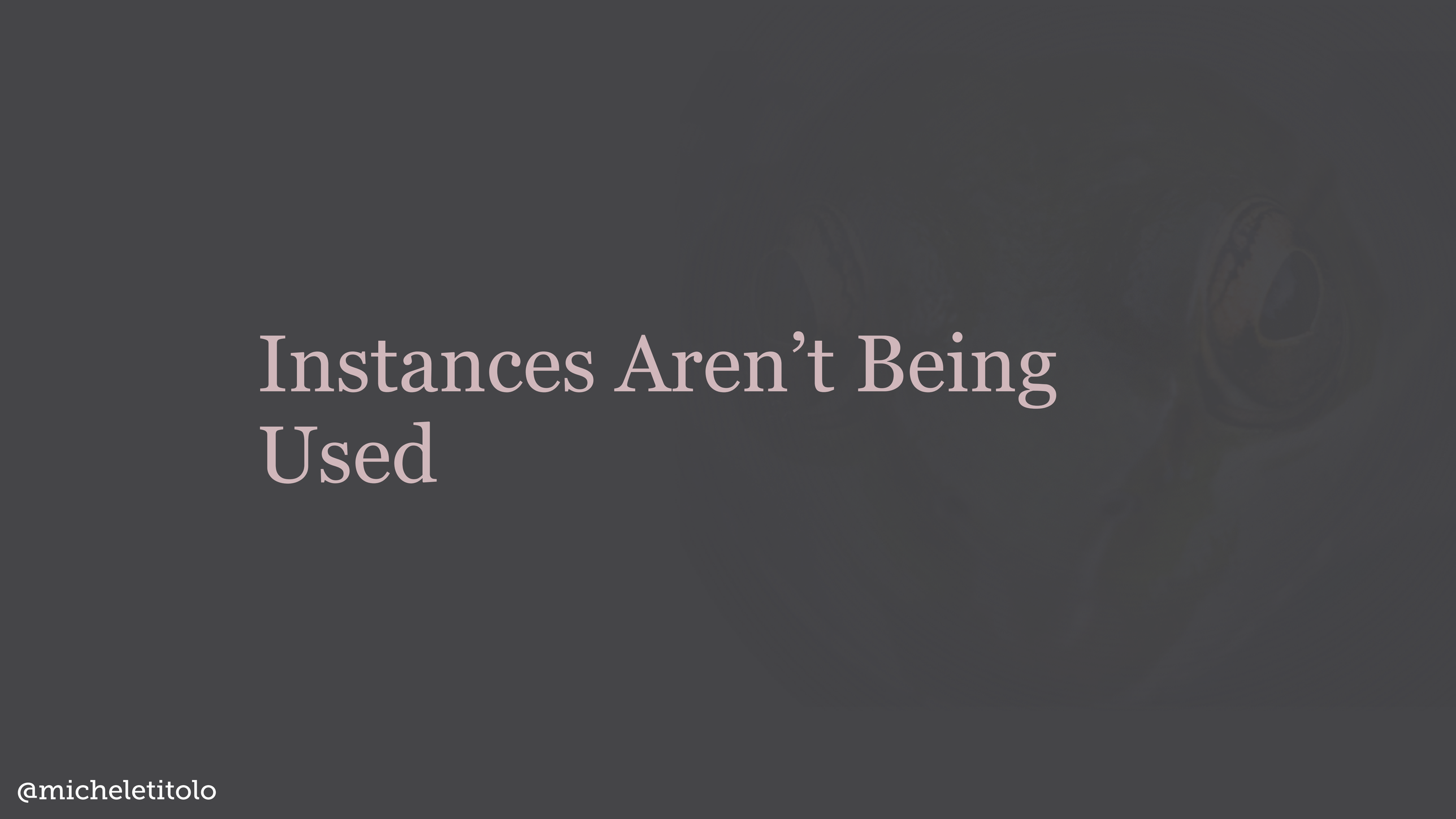
Deploy New Instances

Monitor Health Checks

Send Traffic To New Instances

Scale Up
Scale Down

Instances Aren’t Being Used



Stop Routing Traffic

Gracefully Shutdown

Report Progress

Ultimately Get Removed

Routing Traffic?

Infrastructure++

Load Balancer
Service Discovery

Load Balancer
Service Discovery

All Cloud Providers Have One Built-In

Attach To Scaling Group

New Instances Registered

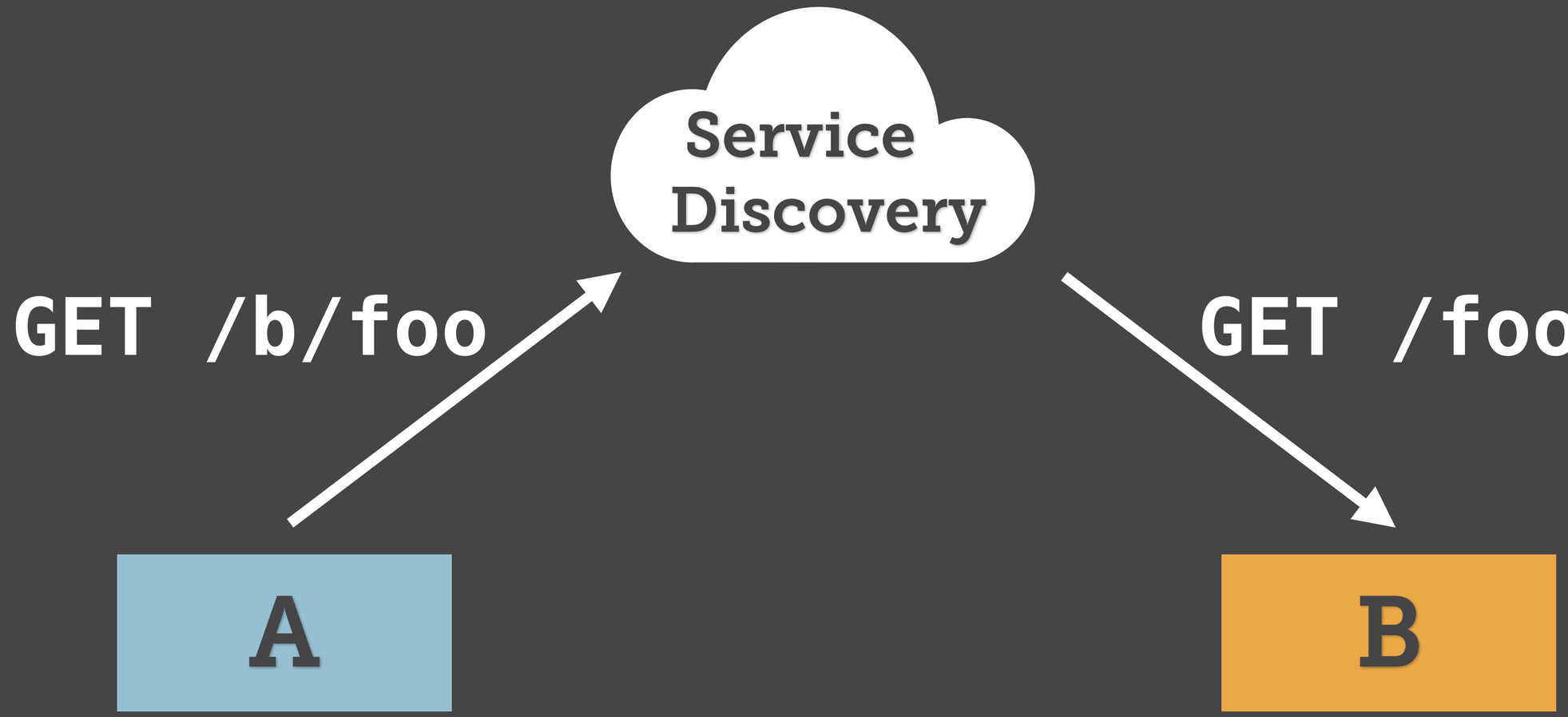
Conversely, Instances Going Away Are Removed

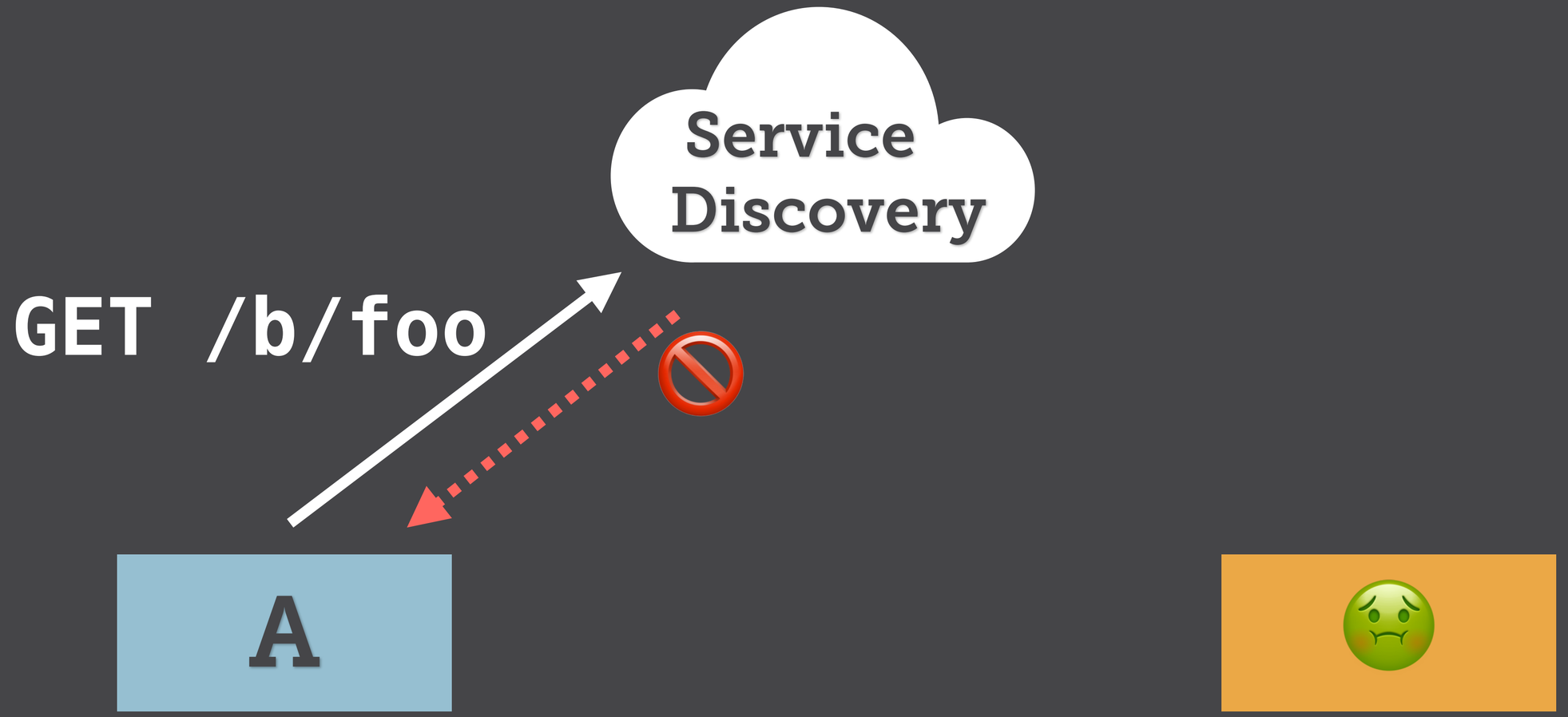
Load Balancer
Service Discovery

Routes To Healthy Instances

Apps Register

Route Via Convention


Service
Discovery
GET /b/foo
GET /foo
A
B


Service
Discovery
GET /b/foo
A

Knows Health, Can Trigger Scaling Automation

Scaling Only Solves So Many Problems

Debugging

First You Need To Know There's A Problem

How?

Alerts

What Qualifies As A Problem?

Varies Per App

RAM, CPU, Latency

Less Obvious, Too

Key Performance Indicator Or Service Level Agreement

Alerts Are For Known Issues

Dashboards

How Do You Debug An Issue?

First: Check Problematic App

Pretend Ssh Doesn’t Exist

Next Best Thing: Logs

Log Collection And Aggregation

Configurable Log Levels

Inspect Logs

Non-Trivial Issue

Escalate

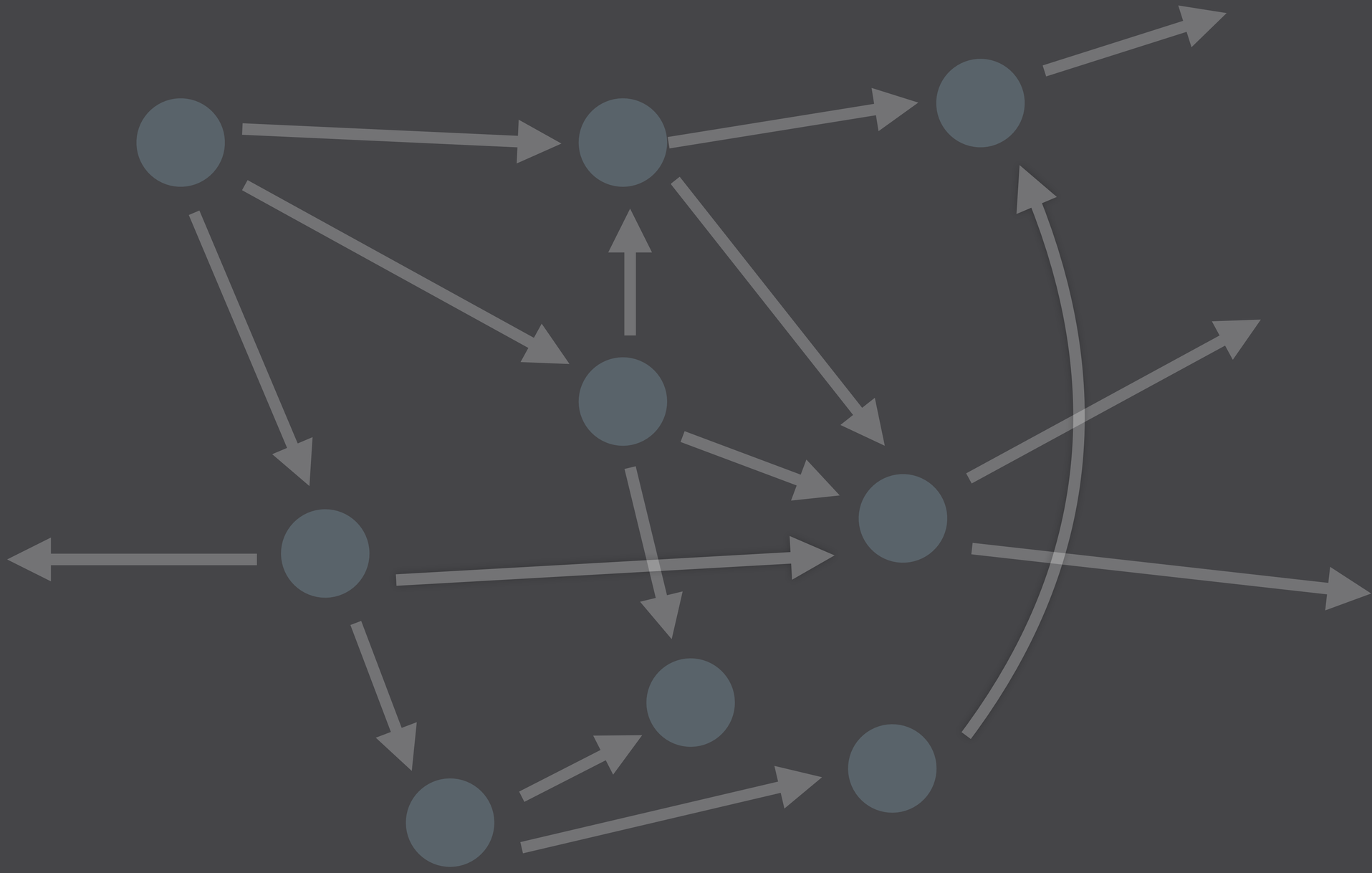
Avoid Cascading Failures

Identify
Isolate

Identify
Isolate

Q: What Parts Of The System Depend On Another?

Request Tracing



Unique Request Ids

Best As Overlay

Opentracing

Add Header Manually

Identify
Isolate

Circuit Breaking

Let App Recover

App Or System

Fail Gracefully

Deploy A Fix

Slowly Start Sending Traffic

Can Be Done Automatically

Built-In Load Balancers
Service Discovery

Otherwise Update Manually

Everything Returns To Normal

What If The Problem Isn't One Of Your Apps?

Do You Have An External Dependency?

No Access To Logs

No Access To Source

You Can’t Deploy A Fix

Fewer Tools

How To Limit Impact?

Status Page

Monitor The Status Page

If Status Page Shows No Issue, Raise It

Keeping Your System Healthy

What's Inside Your Control?

Can’t See Logs

Request Tracing

Circuit Breaking

Degrade Gracefully

Sometimes Infrastructure Fails

Ex: S3 Outage In 2017

Circuit Breaking

To Recap

Debugging Internal Apps

Log, Trace, Circuit Break

Debugging External Apps

Trace And Circuit Break

One Thing All Of These Have In Common...


Visibility

Gain Insights Into Distributed System

Not Free

Adding Visibility To Apps

- Standardized health checks
- Circuit breakers
- Logging
- Alerting
- Request Tracing
- Graceful shutdown

Adding Visibility To Infra

- Consume health checks
- Circuit breakers
- Service Discovery/Load Balancing
- Log Aggregation
- Automated Deployments and Rollbacks
- Dashboards

Gather Information

Create Confidence That System Is Healthy

Creating An Ecosystem For Microservices Is Doable

@micheletitolo

Requires More Metrics

Requires Smarter Infrastructure

Infrastructure won't evolve on its own; people need to change their behavior and fundamentally think of what it takes to run an application a different way.

Nova, Kris and Garrison, Justin (2017). *Cloud Native Infrastructure.* O'Reilly.

Build Software To Support A Healthy Ecosystem

Thank You!

@micheletitolo

Photo Credits

- https://unsplash.com/photos/Rfflri94rs8
- https://unsplash.com/photos/B87VRCWmpqU
- https://unsplash.com/photos/aLDuLRjZknE
- https://unsplash.com/photos/xal6tVZRCGc
- https://unsplash.com/photos/VRAxb_gNjQQ
- https://unsplash.com/photos/AaDKnhCKQ-Q