tetrate

# Identity-Based Segmentation: An Emerging Standard for Zero Trust from NIST

## A dive into NIST SP 800-207A

Zack Butcher, Founding Engineer @ Tetrate

tetrate

published

# Identity-Based Segmentation: An ~~Emerging~~ Standard for Zero Trust from NIST

## A dive into NIST SP 800-207A

Zack Butcher, Founding Engineer @ Tetrate

tetrate

# Identity-Based Segmentation: ~~An Emerging Standard~~ for Zero Trust from NIST

published

guideline

## A dive into NIST SP 800-207A

Zack Butcher, Founding Engineer @ Tetrate

# Speaker Introduction

- **Early Engineer on Istio**
  - Part of the original team at Google
  - Sat 2 terms on project's Steering Committee
- **Founding Engineer at Tetrate**
  - Goal: power the world's application traffic
  - Previously worked on Istio at Google, and across GCP: resource hierarchy, service management, IAM, mesh
- **I write about Istio and collaborate with NIST on modern microservice security, zero trust, & access control**
  - *Istio: Up and Running*
  - NIST SP 800-204A: *Building Secure Microservices-based Applications Using Service-Mesh Architecture*
  - NIST SP 800-204B: *ABAC for Microservice-based Apps using a Service Mesh*
  - NIST SP 800-207A: *A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Cloud Environments*
  - Multiple patents around *Next Generation Access Control*


Zack Butcher

**Founding Engineer, Tetrate**

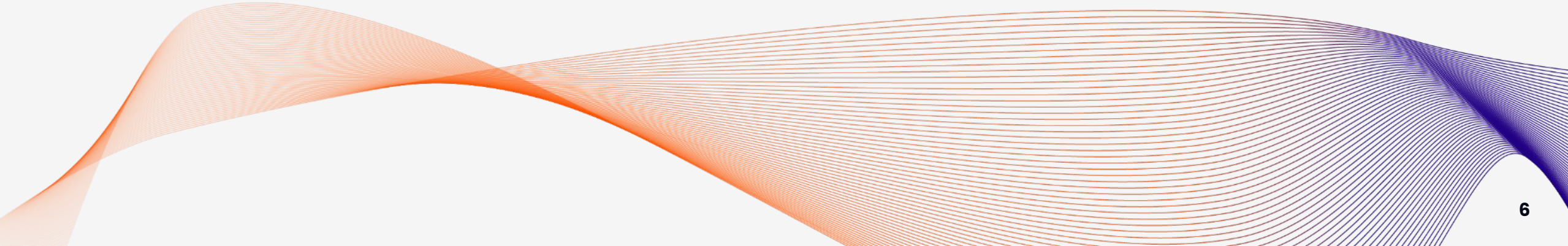
# Agenda

- Build a working definition of *Zero Trust*
- Introduce *Identity Based Segmentation*
- Understand how we can move incrementally from network-based policy to identity-based policy
- Discuss how a *Service Mesh* can be used to implement Identity Based Segmentation

This talk is summarizing NIST SP 800-207A, ***A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Cloud Environments***

tetrate

# Building a working definition of Zero Trust



# What does Zero Trust really mean?

The attacker is already in your network, how can you limit the damage they can do?

# Something we're familiar with: Users Access

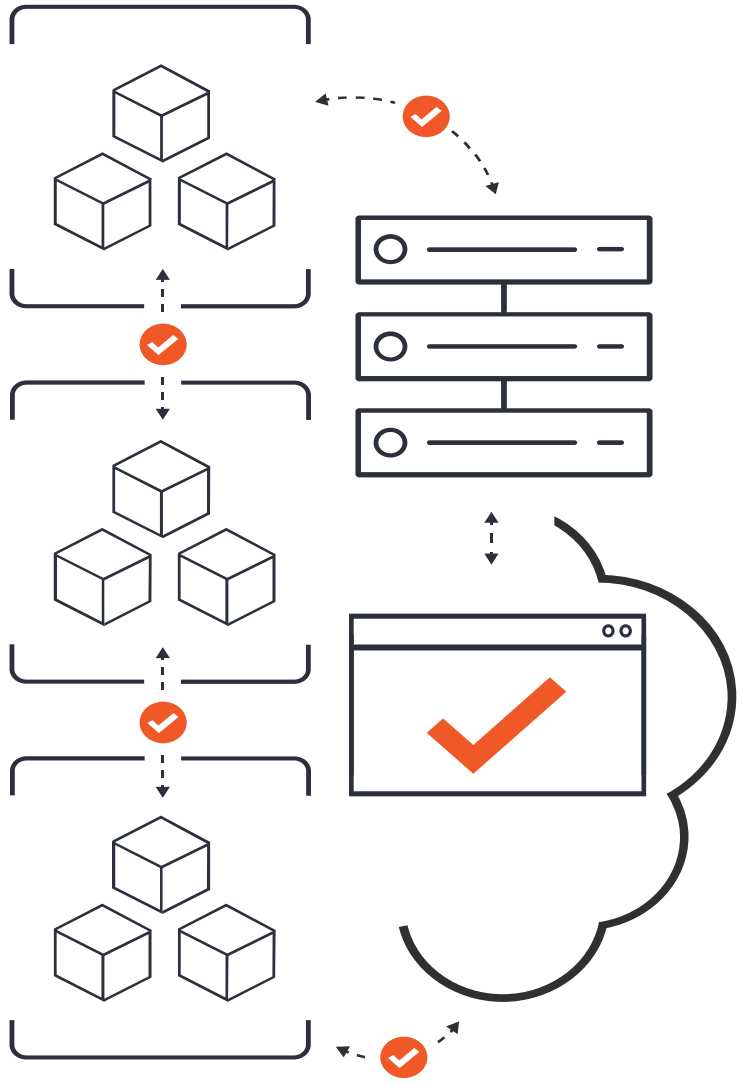
tetrate

1. Request comes in the front door
2. We validate the credential (Authentication)
3. We check the user allowed to call that method (Authorization)
4. We forward on to the app to do work

# Service-to-service communication needs similar checks

tetrate

- We need a runtime service identity (Authentication)
- That we can use to control which services are allowed to communicate (Authorization)
- Nice to have: if our identity can be a certificate, we can use it for encryption in transit too (thanks SPIFFE!)



# Final Piece of the Mental Model

**The attacker is already in the network.** How can I mitigate what an attacker that's already inside the perimeter can do?

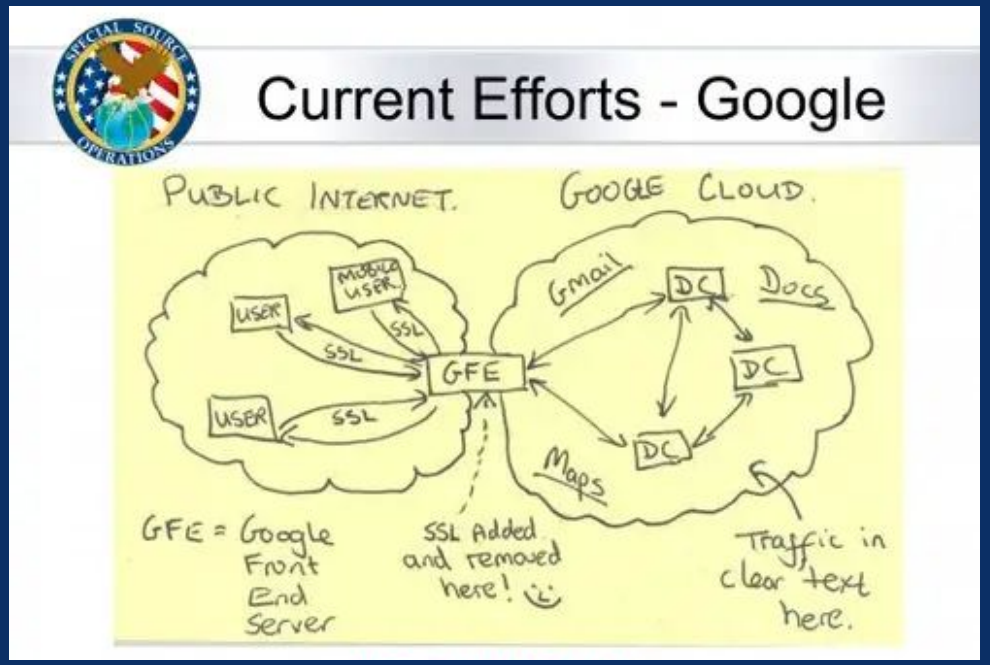



Image from the Washington Post

tetrate

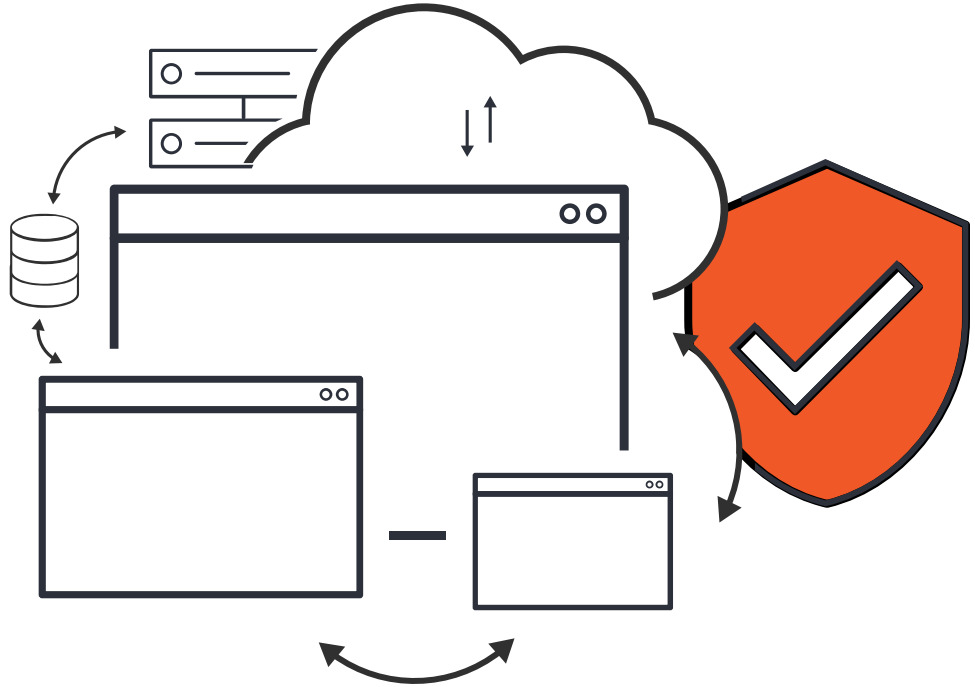
# We need to bound attacks in space and in time

Runtime encryption, authentication, and authorization reduce the attack surface exposed that's exposed by our applications.

# Zero Trust

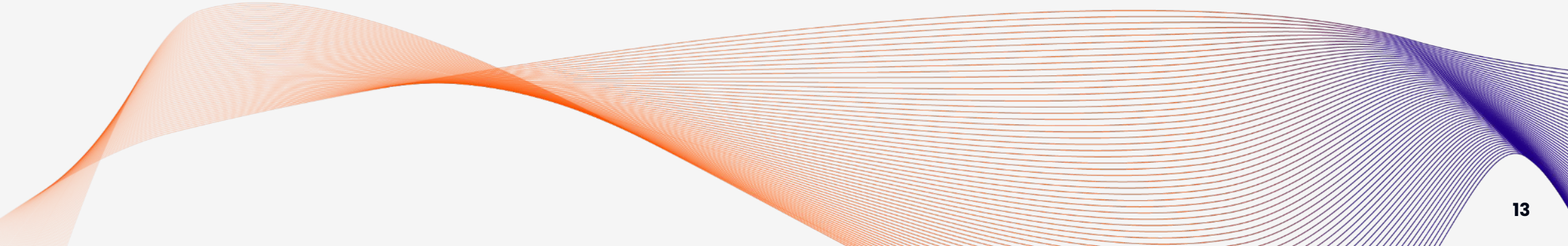
*Assume the attacker is in the network*

- Trust is no longer based on network perimeter, perimeters are assumed to be breachable.
- All access decisions must be based on least privilege, per-request, context-based and on identities such as users, services and devices.



tetrate

# Identity Based Segmentation

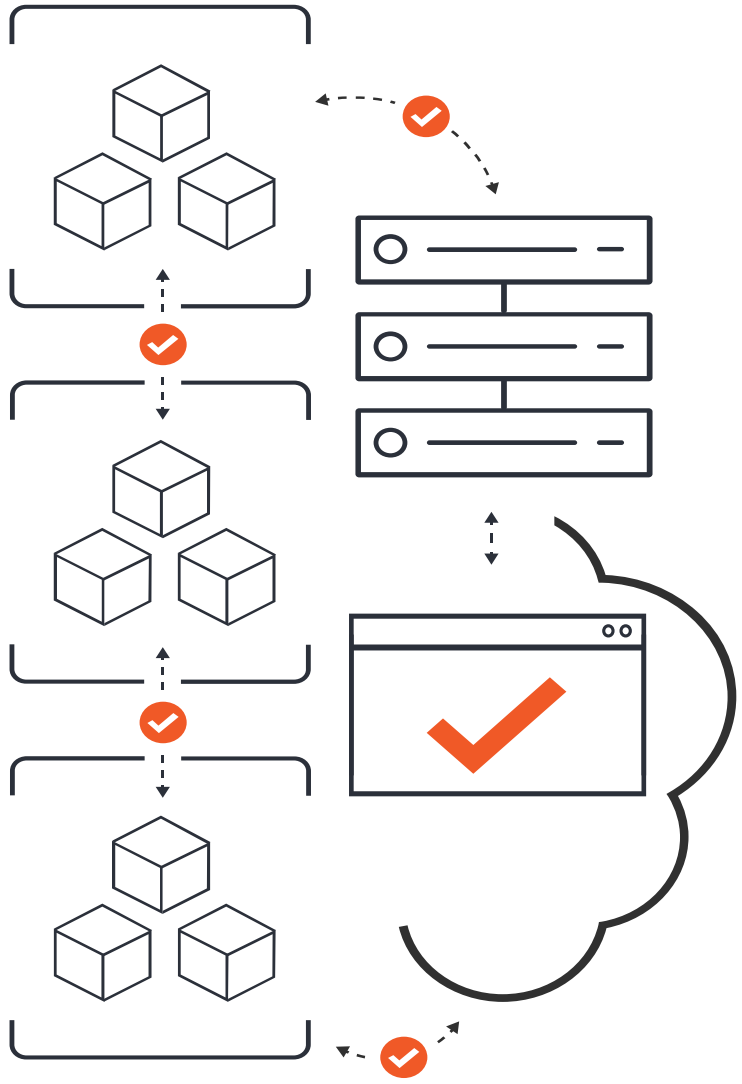


# Identity Based Segmentation

*aka Zero Trust Segmentation*

Isolation of a workload to access only those resources for which it is allowed, dictated by well-defined policies based on:

- Tamper-proof, cryptographically verifiable Identities
- Identities based on service, user, and device
- *not* on network parameters such as IP Address or Subnets



# Implementing Identity Based Segmentation at Runtime

tetrate

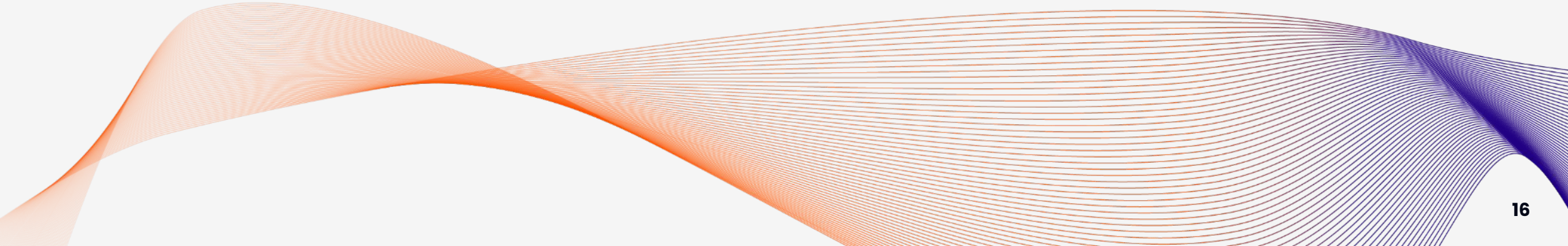
We need a minimum of 5 policy checks on each request in our infrastructure:

1. Encryption in Transit
2. Service Identity & Authentication
3. Service-to-Service Authorization
4. End User Identity & Authentication
5. End User-to-Resource Authorization

If we do these things, we realize a ZTA for our runtime systems

tetrate

# How we can move incrementally from network-based policy to identity-based policy



tetrate

# How we can move incrementally from ~~network-based policy~~ to identity-based policy *today's world*

# Policy at a Single Layer is a Problem

- Network level policy alone requires high maintenance
  - High rate of change due to dynamic cloud environment
- Service identity-based policies alone are difficult to administer
  - different identity domains make consistent policy hard across on-prem systems, cloud providers and different compute runtimes
- Network oriented policies cannot be completely eliminated given current compliance requirements

Therefore, Multi-tier Policies are required.

# Multi-tier Policies

- *At least* two layers:
  - Network-tier policies – e.g. Firewall rules
  - Identity-tier policies – e.g. App-to-App communication rules based on identities defined through your dedicated infrastructure layer
  - Plenty of other types/tiers of policy: they're good, this is a minimum!

# Multi-tier Policies

- *At least* two layers:
  - Network-tier policies & Identity-tier policies
- Multi-tier policies are realistic and non-disruptive to current compliance practices:
  - We can make network-tier policy relatively static
  - App-to-App communication rules based on service identities is dynamic
  - We can still force traffic through traditional inspection points as needed

# Multi-tier Policies

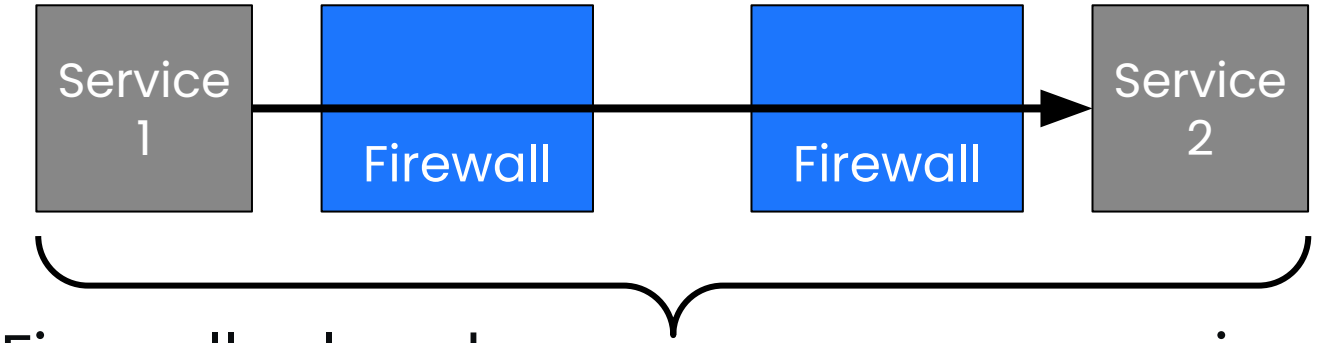
- *At least* two layers:
  - Network-tier policies & Identity-tier policies
- Multi-tier policies are realistic and non-disruptive to current compliance practices
  - Network policy become more static, dynamism shifts to identity policies

Outcome: we can relax network-tier policies that slow agility if we augment them with identity-tier policies

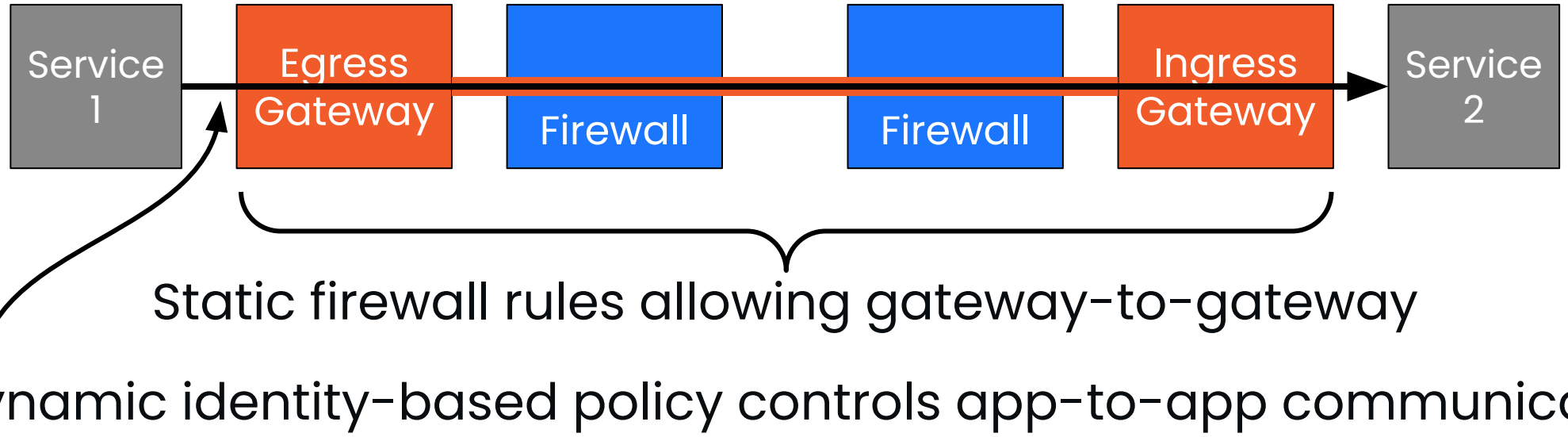
# Multi-tier Policies

tetrate

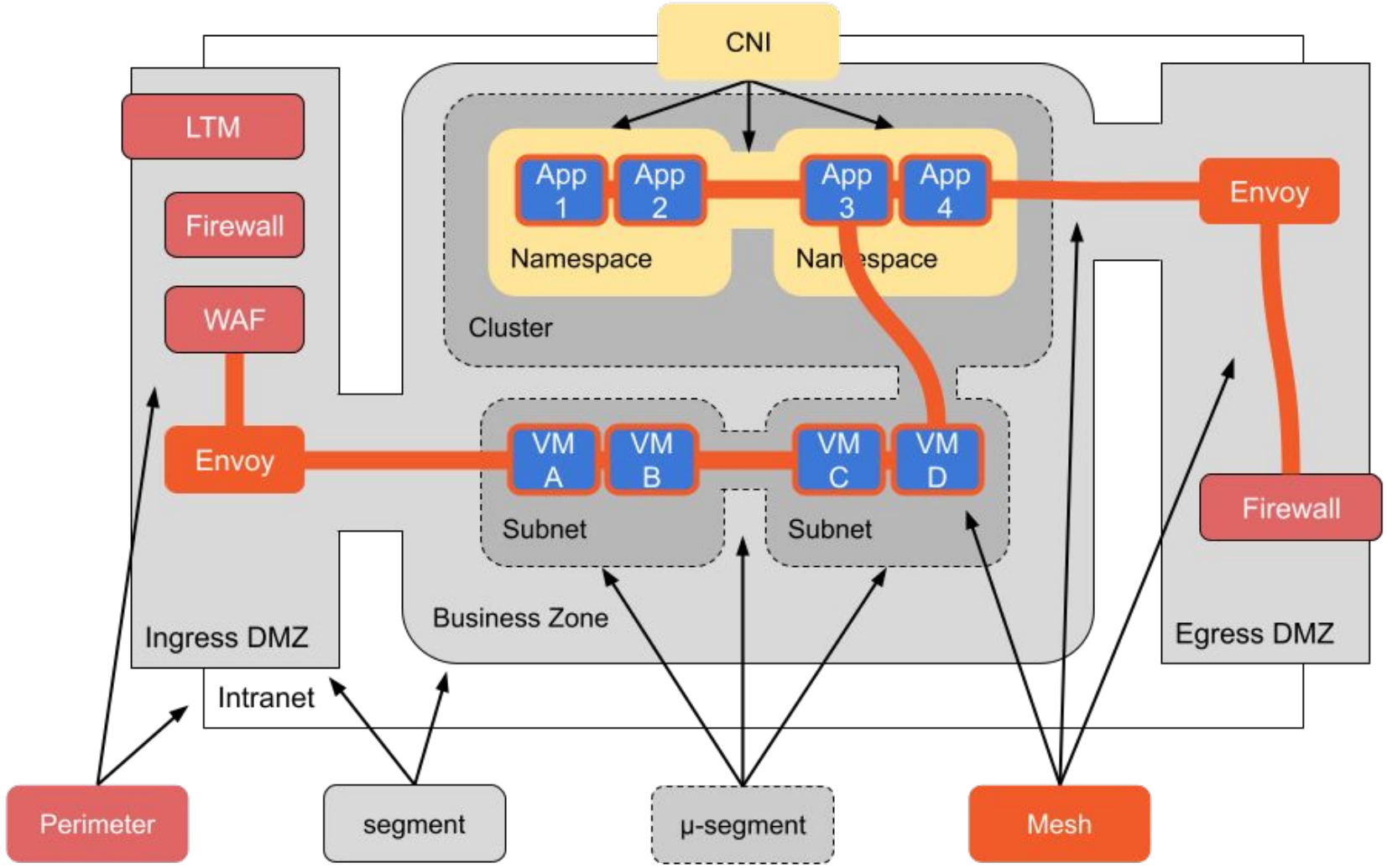
**Before**




**After**

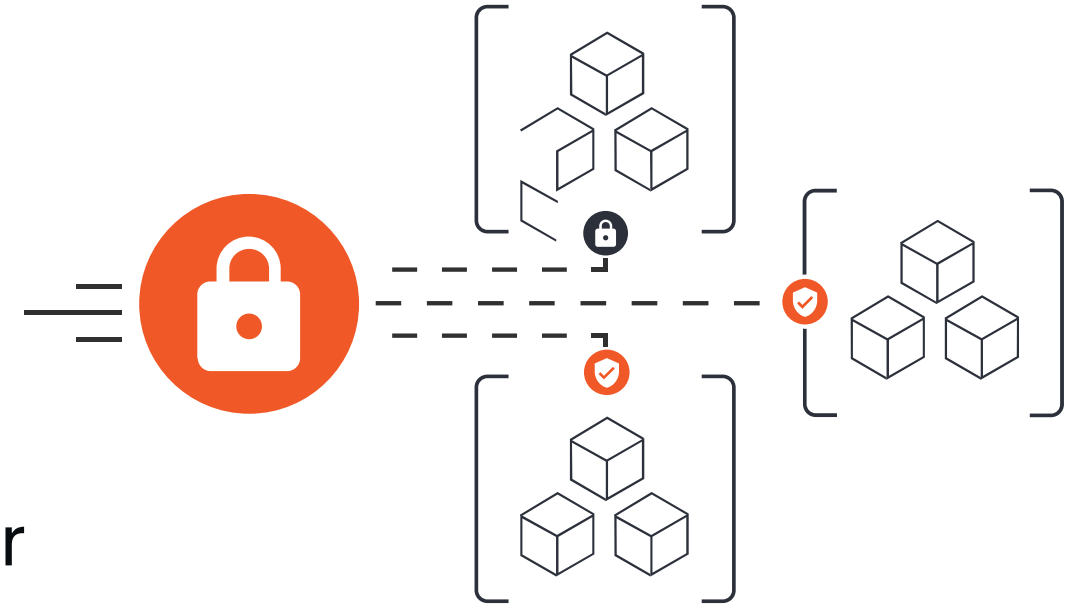


# Identity-tier Policies coexist with various Network Policies



# Advantages of Multi-tier Policies

- Multiple network-tier policies can continue to exist:
  - Perimeter controls like firewalls and WAFs
  - Virtual Network-based approaches such as NSX or VPCs
  - Cloud native controls like CNI
- Allows for relaxation of network-tier policies as any of the deficiencies there can be addressed through identity-tier policies.



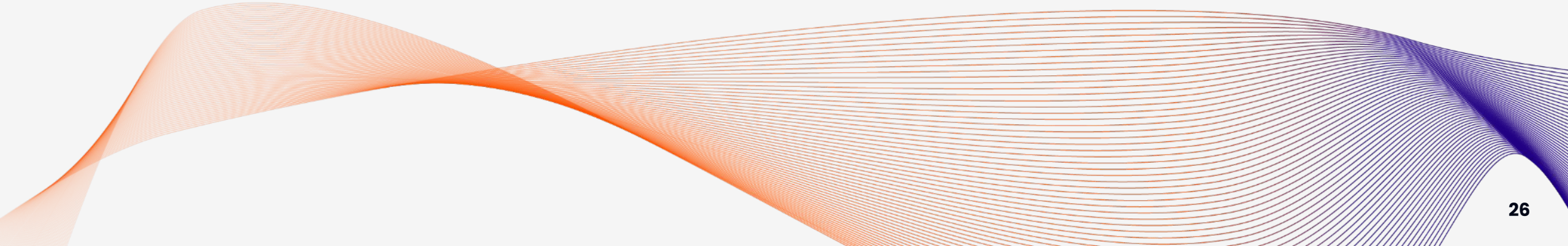
# Advantages of Multi-tier Policies

tetrate

- Identity-tier policies sit on top of your existing network policies
- Provides a defense in depth
- Service mesh components enforcing access policies - by configuration and functions -  play the role of security kernel
  - always invoked (non-bypassable)
  - Verifiable (independent of app services code)
  - See NIST SP 800-204B

tetrate

# How a Service Mesh can be used to implement Identity Based Segmentation



# Implementing Identity Based Segmentation at Runtime

tetrate

The Service Mesh is a dedicated infrastructure layer enabling you to monitor, secure, connect and manage services consistently.

It can be used to implement Identity Based Segmentation, among other use cases.

NIST SP 800-204A, NIST SP 800-204B

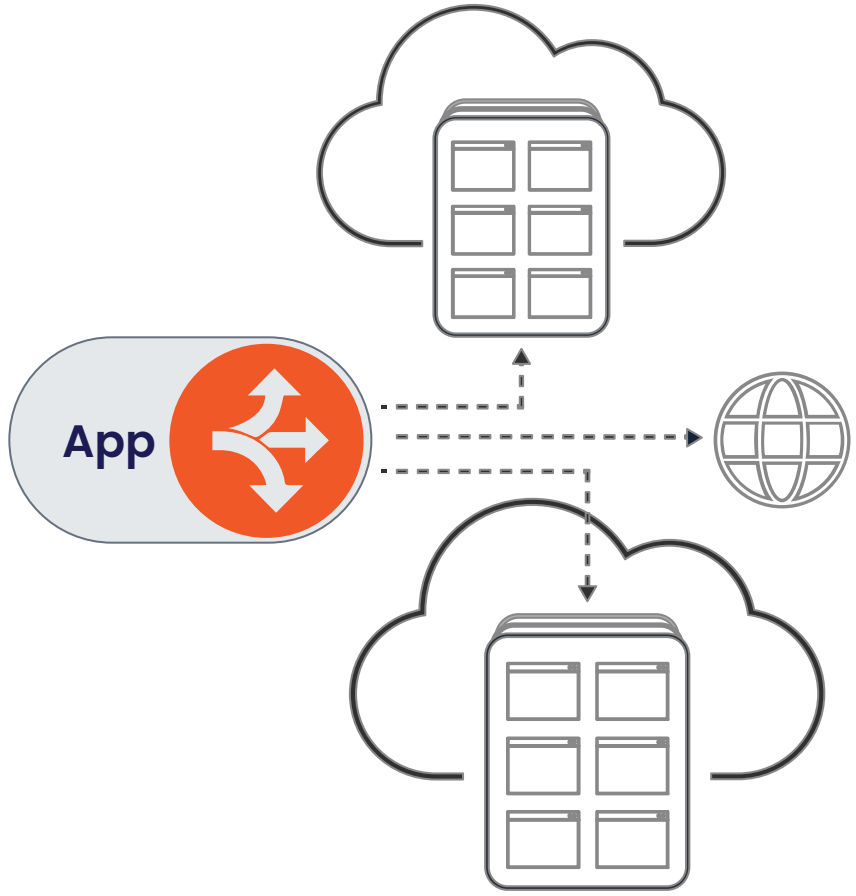
# What is a Service Mesh?

tetrate

Deploy a sidecar proxy next to every application instance, which intercepts all traffic in *and* out to achieve:

- L7 application identity & encryption in transit
- Per request policy and controls
- Service discovery, load balancing, and resiliency
- Operational telemetry: metrics, logs, and traces

And control them all centrally with declarative configuration.



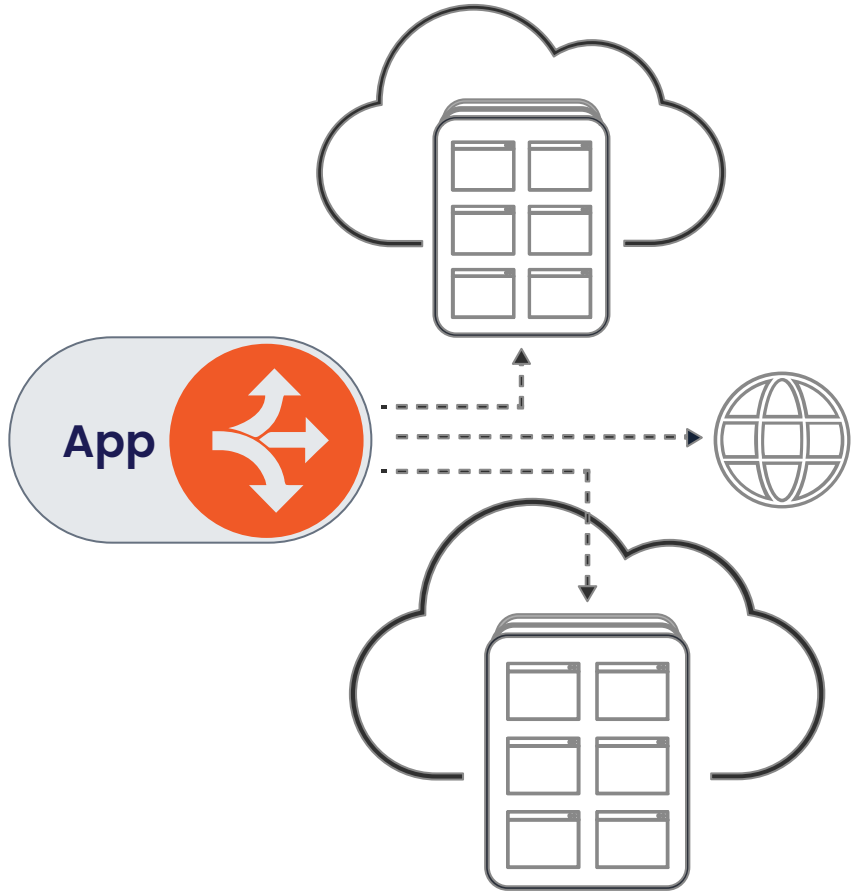
# What is a Service Mesh?

near

Deploy a ~~sidecar~~ proxy ~~next to~~ every application instance, which intercepts all traffic in *and* out to achieve:

- L7 application identity & encryption in transit
- Per request policy and controls
- Service discovery, load balancing, and resiliency
- Operational telemetry: metrics, logs, and traces

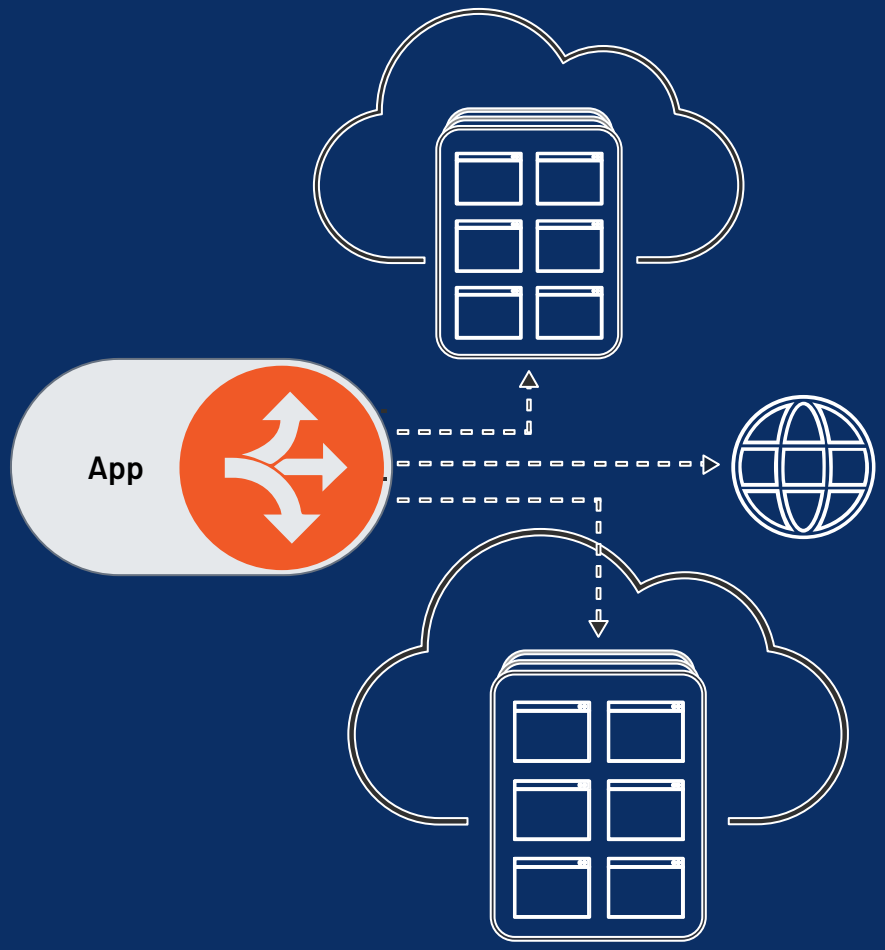
And control them all centrally with declarative configuration.



# The mesh's proxy is a universal Policy Enforcement Point

The proxy intercepts all traffic and can apply policies at the application layer. It is a *reference monitor.*

NIST SP 800-204B



# The Mesh forms a Security Kernel

With proxies as PEPs we can move security concerns out of the application and into the mesh.

NIST SP 800-204B

# A Service Mesh enables cross-cutting change

A mesh allows for centralized control with distributed enforcement. Focused teams can manage policy on behalf of the org.

NIST SP 800-204A

# The Service Mesh lets us bound attacks in space and in time

Runtime encryption, authentication, and authorization reduce the attack surface exposed that's exposed by our applications.

# Realizing Identity Based Segmentation with a Service Mesh

tetrate

1. Encryption in Transit
   - mTLS in the mesh, (m)TLS to external services
2. Service Identity & Authentication
   - SPIFFE identities for workloads in the mesh
3. Service-to-Service Authorization
   - Built-in policies good starting point, mature implementations should leverage dedicated authz infra for richer policy and decisions – e.g. Next Generation Access Control (NGAC)
4. End User Identity & Authentication
   - Defer to trusted identity provider or IDaaS
5. End User-to-Resource Authorization
   - Integrate with existing systems via OIDC or leverage dedicated authz infra - e.g. NGAC

tetrate

# Thank You

tetrate.io

Tetrate

@tetrateio

info@tetrate.io

**Session Feedback Appreciated!**