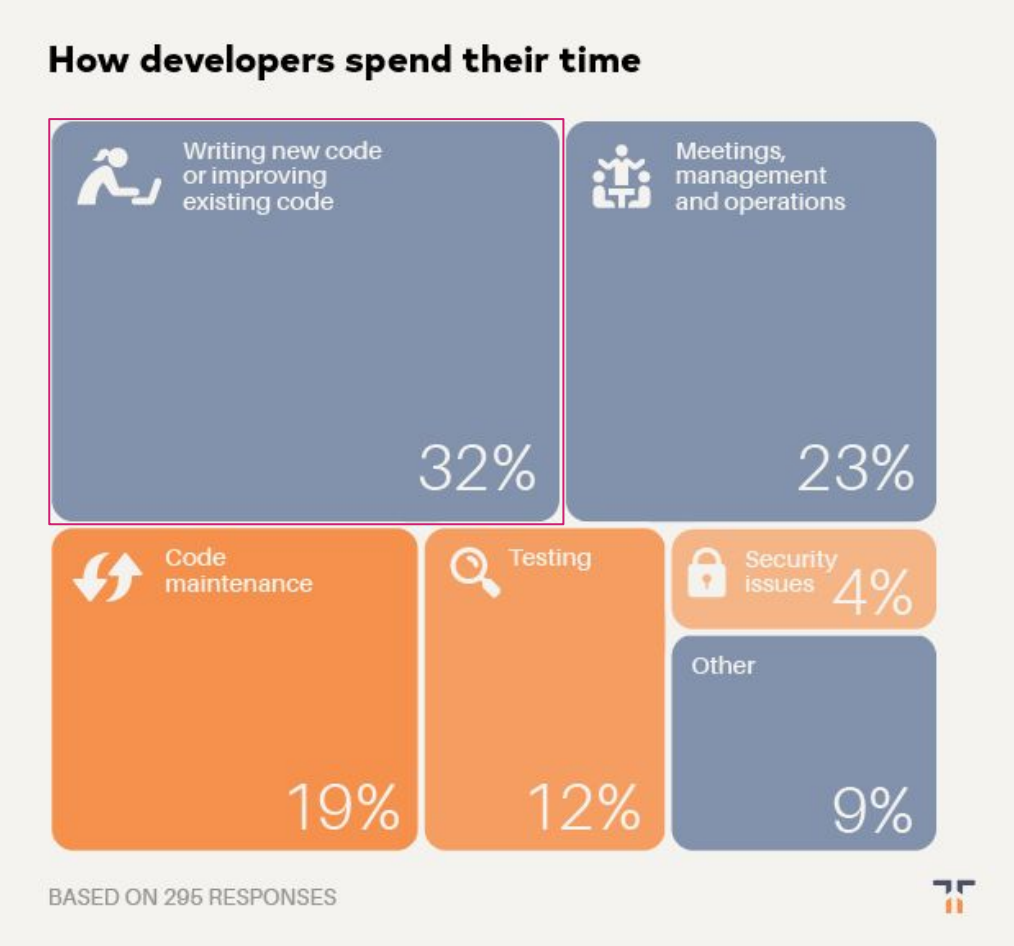
Consul

HashiCorp

Consul Education Series: Application Development

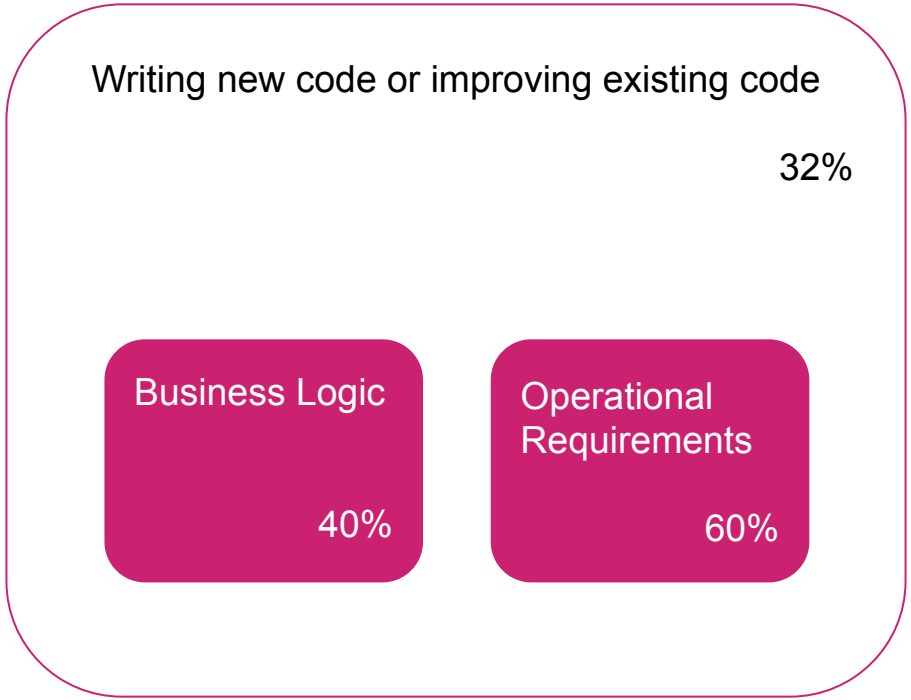
Poll:

Percent of time developer spends writing code?



https://thenewstack.io/how-much-time-do-developers-spend-actually-writing-code/

Reality*



* Estimated, not backed by a survey

Goal:

Maximize time spent writing business code.

Goal:

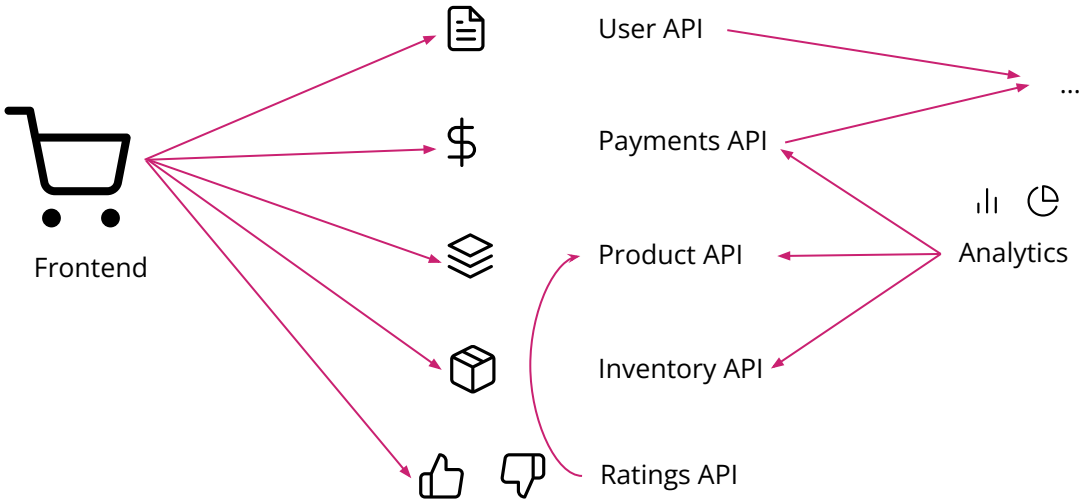
Minimize time spent on other work.

Change Software Architecture

Microservices

Apply domain-driven design to separate services

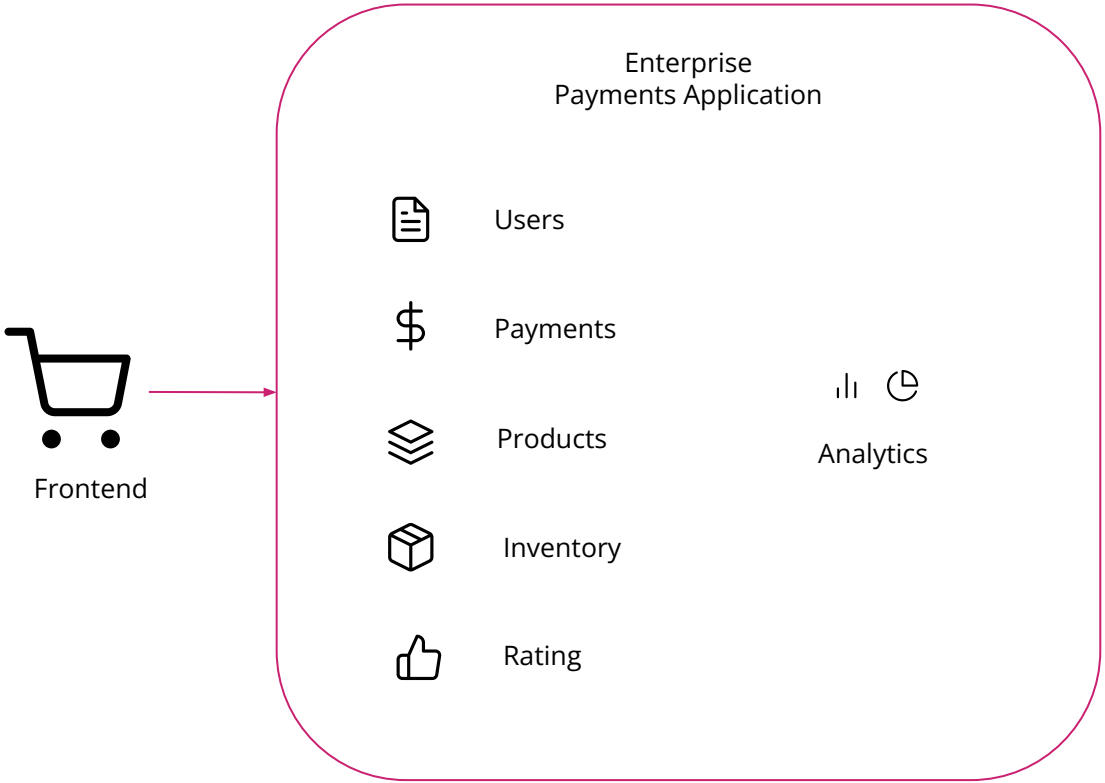
- Loosely coupled business logic
- Scale services more easily
- Deploy services independently
- Usually associated with “new” code



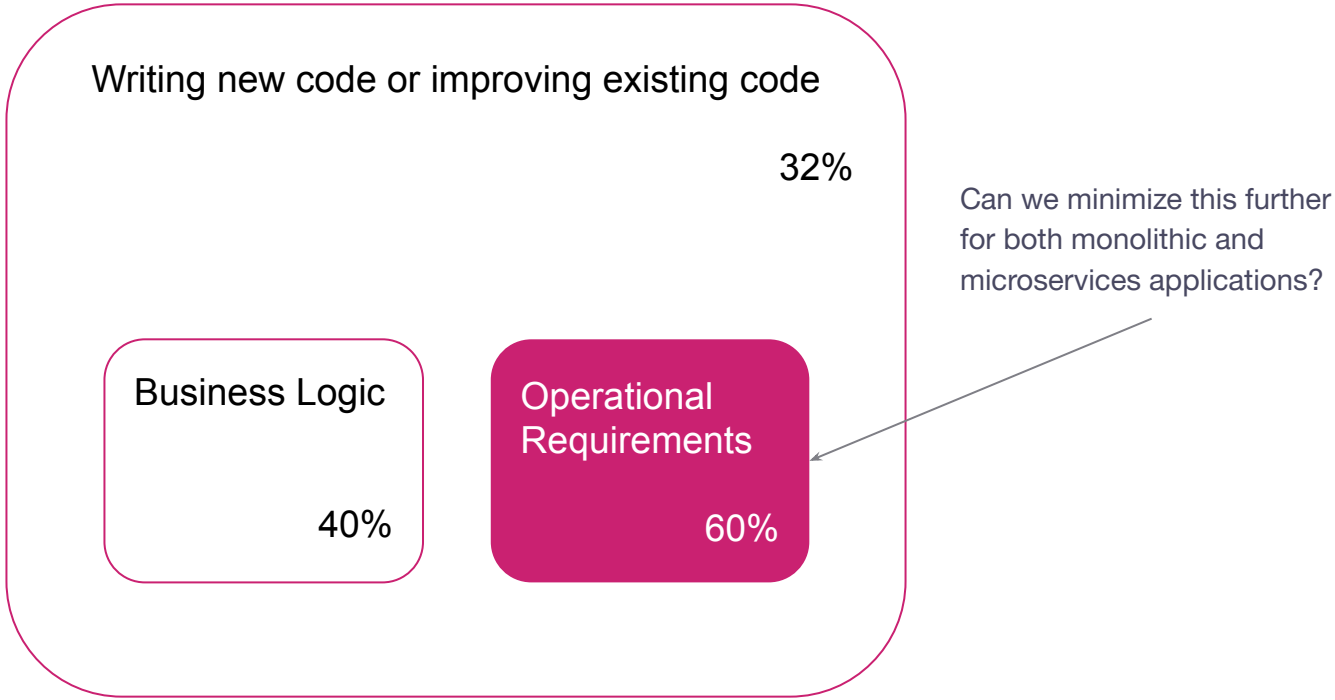
Monolith

A single code base

- Tightly coupled business logic
- Debug everything in one place
- Deploy everything once
- Usually associated with “legacy” code



Reality*

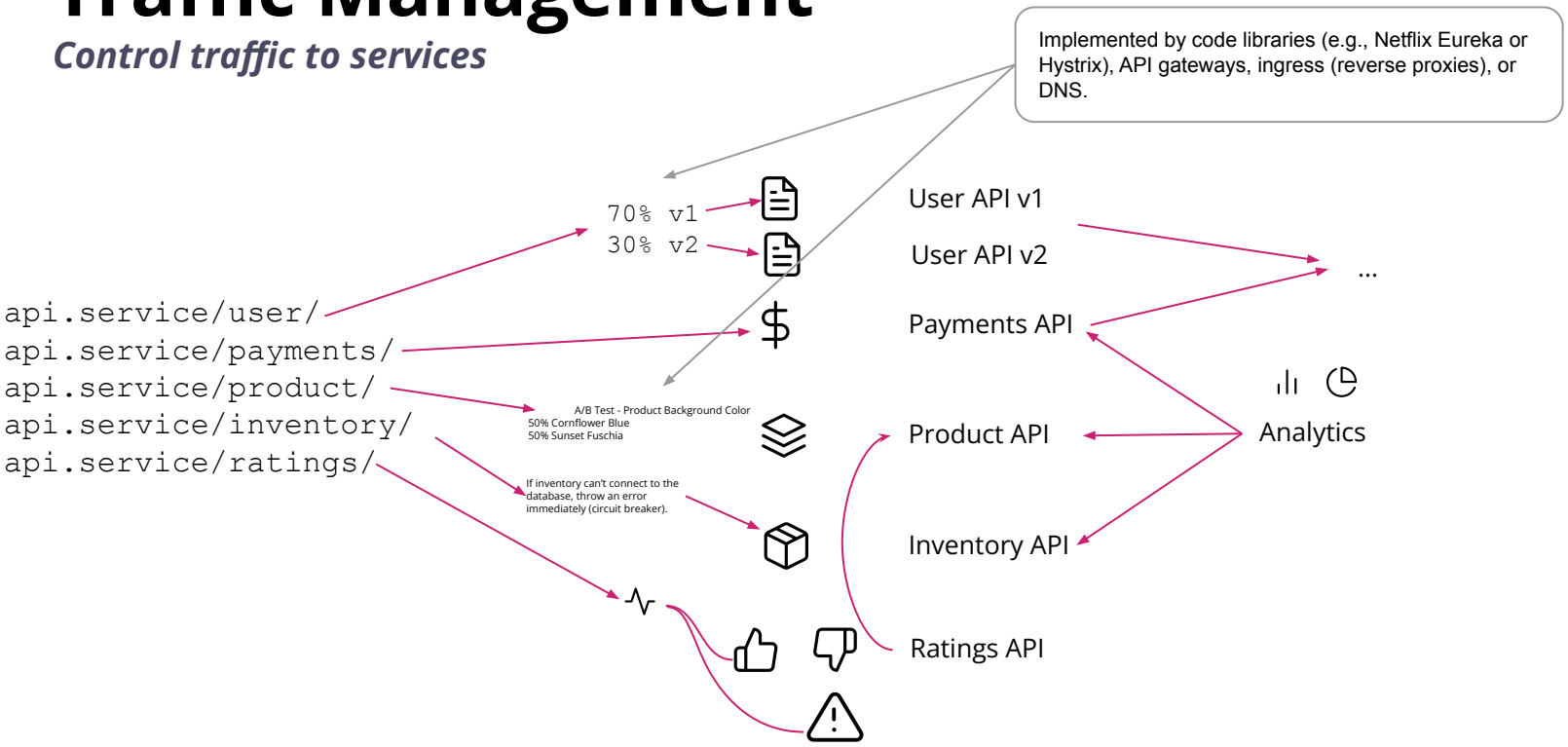


* Estimated, not backed by a survey

Challenge of Service Communication

Traffic Management

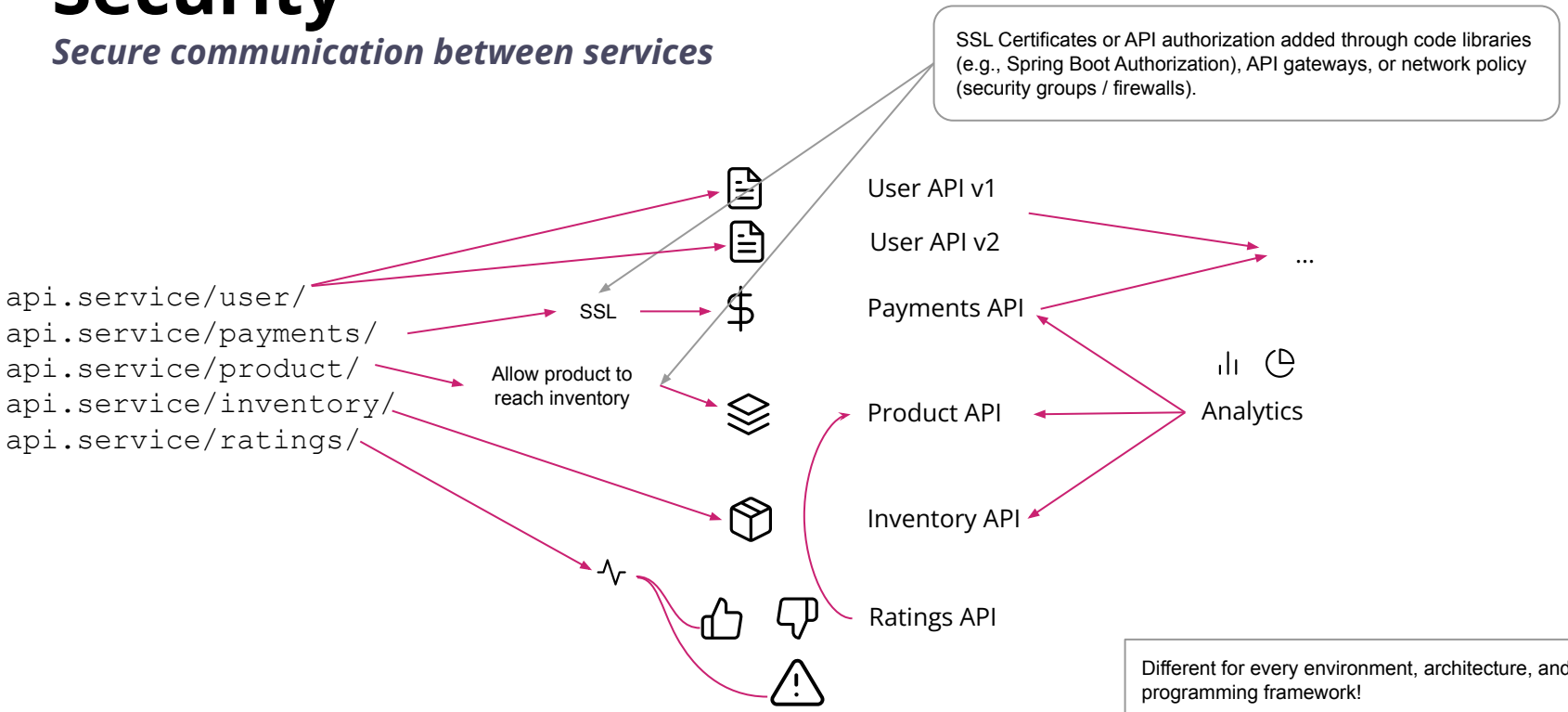
Control traffic to services



Different for every environment, architecture, and programming framework!

Security

Secure communication between services

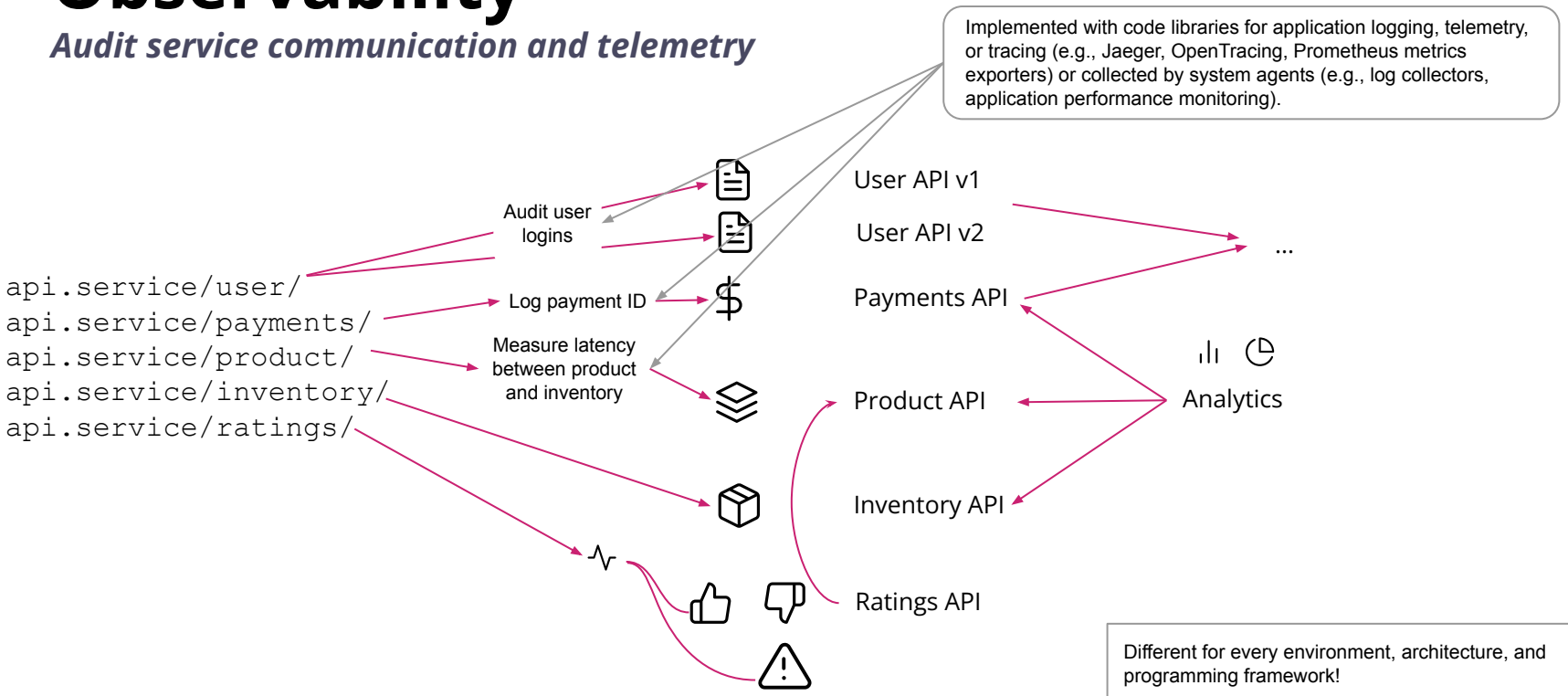


Different for every environment, architecture, and programming framework!

Difficult to develop and operate. What happens if the certificates change? What about a firewall rule that blocks traffic?

Observability

Audit service communication and telemetry



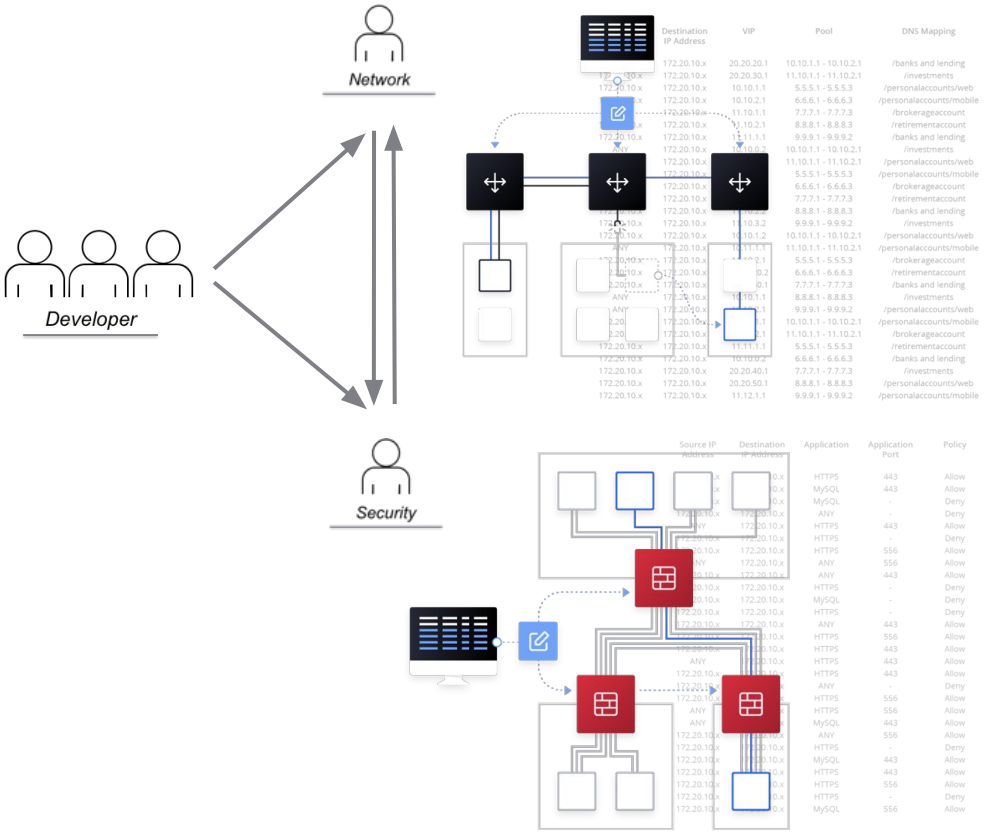
Different for every environment, architecture, and programming framework!

Difficult to standardize across the organization.

Developer Velocity versus Security & Observability

Currently

- Ticket-based system
- Manual approach
- Error-prone process
- Multiple handoffs between teams



Development

- Code abstractions for new libraries and formats
- Verify platform changes do not affect performance or functionality

- Time spent:
  - Reverse engineering libraries
  - Securing services
  - Adding logging, tracing, and telemetry

Security

- Secure application without breaking functionality
- Trust everything in the network

- Time spent:
  - Exploring exceptions to security requirements
  - Aggregating and standardizing logs and traces
  - Auditing network traffic

Operations

- Make platform updates with breaking functionality (rotating certificates)
- Toil across environments, platforms, and frameworks

- Time spent:
  - Checking network (firewall, load balancer, DNS) changes
  - Reverse engineering application
  - Troubleshooting logs and telemetry

Recall:

Minimize time spent on other work.

What if we had…

An *agent*
which can be completely automated…

What if we had…

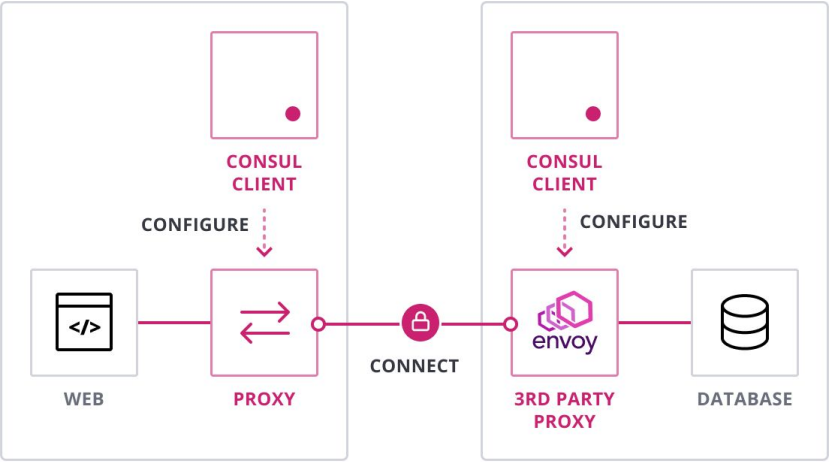
An *agent*
which can be completely automated…
&
A *service*
which manages and automates the agents

What if we had…

Service-to-service Communications
as a
Service?

Service Mesh

Leveraging the Sidecar Pattern



Using Service Mesh

Lab Exercises: Deploy Your App

Lab Exercise: Deploy Your App

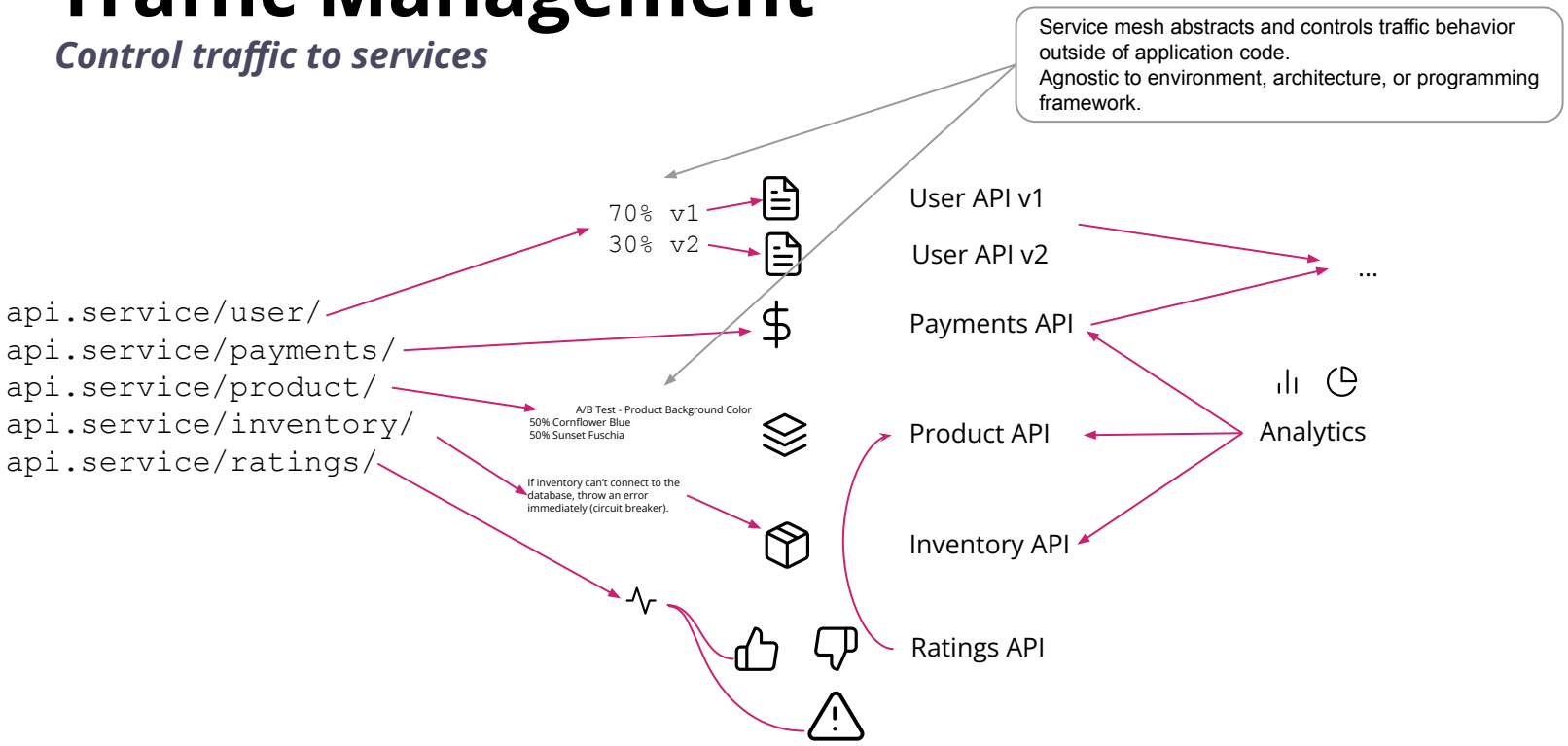
You will accomplish the following in this lab:

- Review your environment
- Connect your runtimes
  - K8s1 (stateless cluster)
  - K8s2 (stateful cluster)
- Deploy your application
- Test your application
- Service Mesh: Service Discovery

Your instructor will provide the URL for the lab environment.

Traffic Management

Control traffic to services



Lab Exercises:
Traffic Management

Lab Exercise: Traffic Management

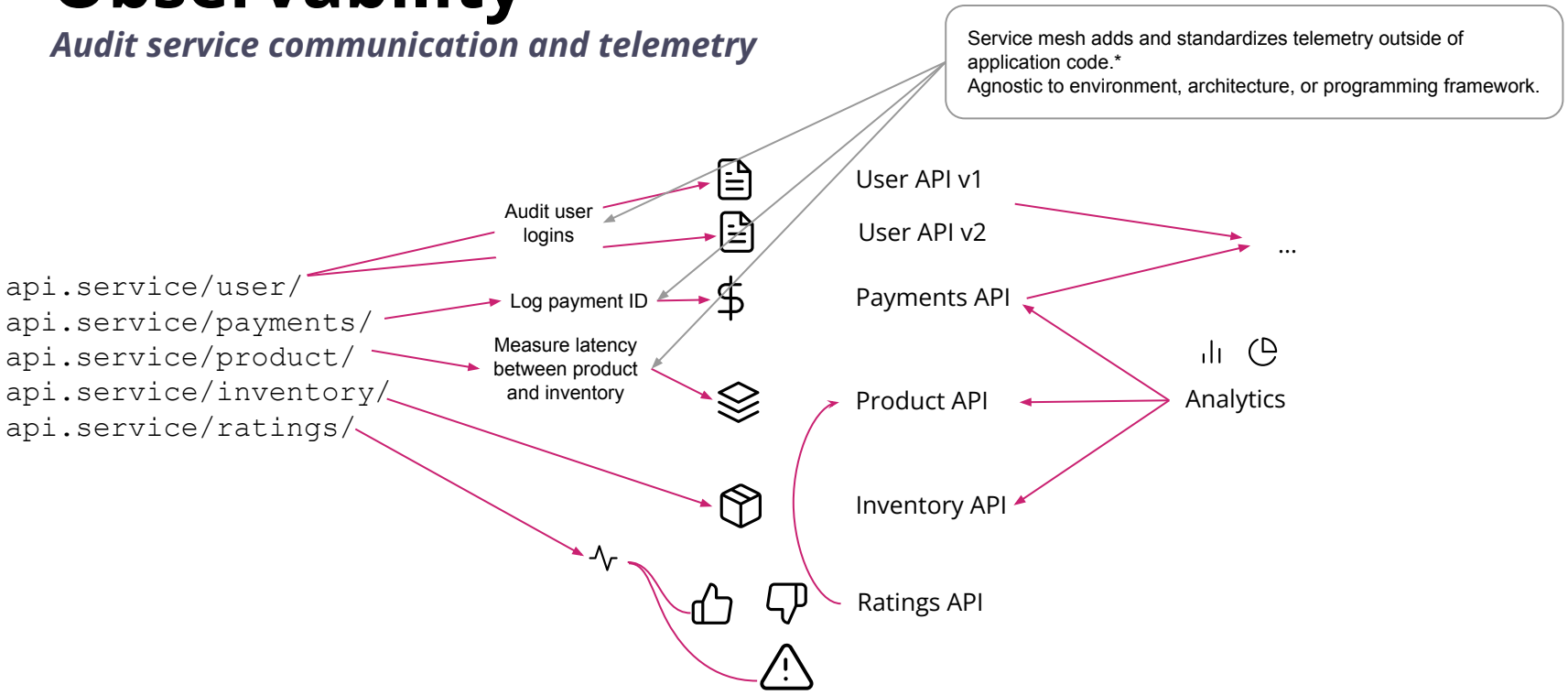
You will accomplish the following in this lab:

- Set up an Ingress Gateway
- Configure Request Routing for a Virtual Service
- Traffic Shape to a New, More Secure Version of your App

Your instructor will provide the URL for the lab environment.

Observability

Audit service communication and telemetry



* Requires organizational effort to standardize or refactor existing tools.

Lab Exercises: Observability

Lab Exercise: Observability

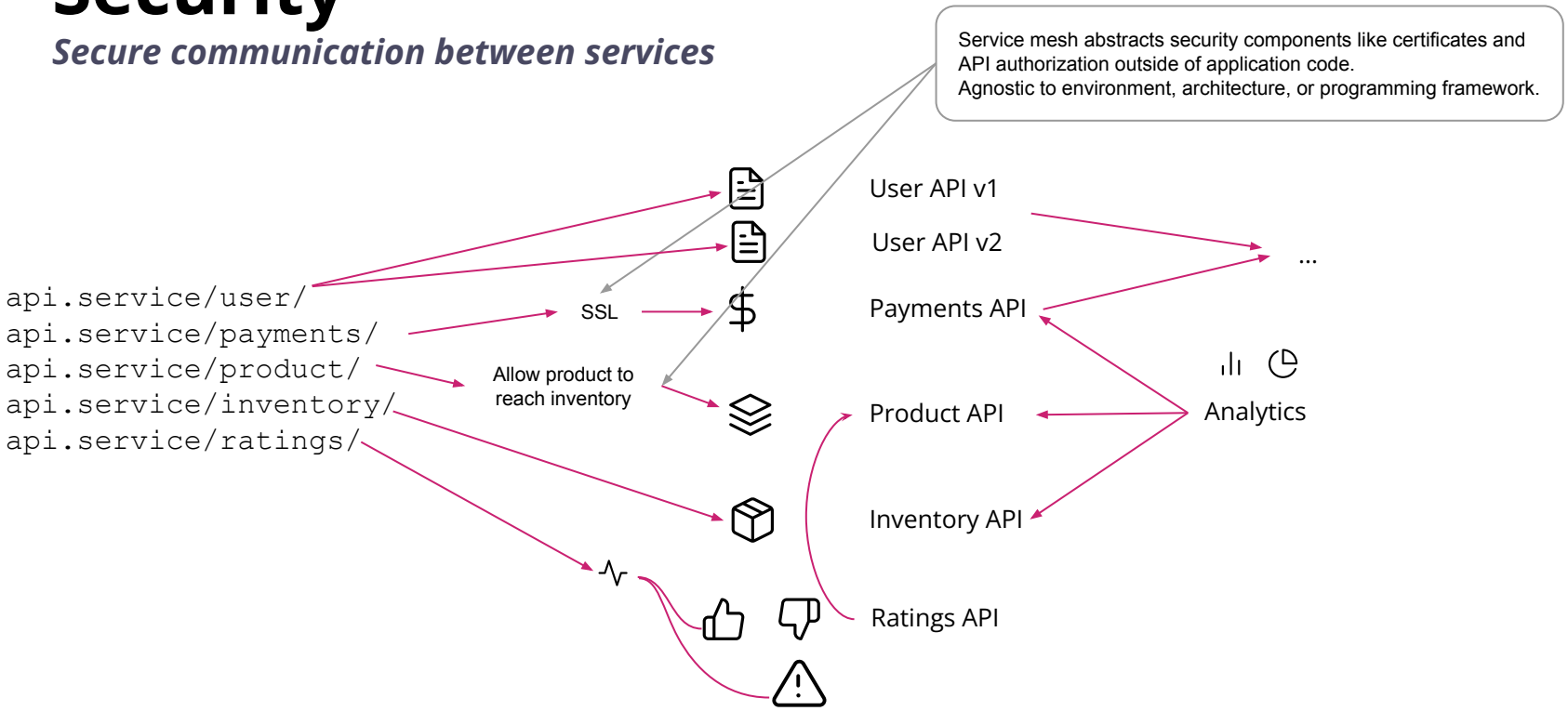
You will accomplish the following in this lab:

- Collect application metrics
- Collect application traces

Your instructor will provide the URL for the lab environment.

Security

Secure communication between services



Lab Exercises: Security

Lab Exercise: Security

You will accomplish the following in this lab:

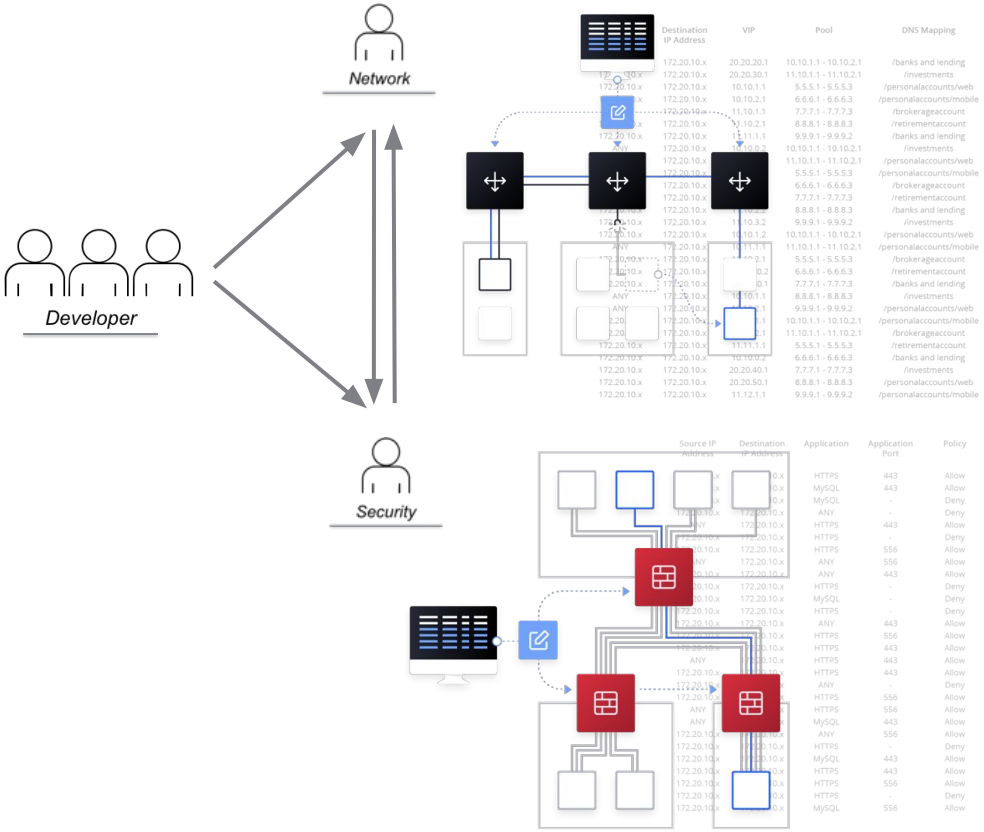
- Secure TCP Traffic
- Secure HTTP Traffic

Your instructor will provide the URL for the lab environment.

Developer Velocity through Security & Observability

Service Mesh

- Event-based system
- Automated
- Isolate service communication changes
- Standardization across teams



Recall:

Maximize time spent writing business code.

Development

- Focus on building business logic
- Abstract service-to-service communication

- Time spent:
  - Building business logic
  - Troubleshooting service mesh/application interactions

Security

- Standardize audit and encryption requirements for applications
- Zero Trust with less development friction

- Time spent:
  - Assessing application vulnerabilities
  - Automating audit of application logging and traces

Operations

- Make platform updates with less impact to applications (e.g., certificate rotations)
- Scale support across environments, platforms, and frameworks

- Time spent:
  - Supporting service mesh
  - Enabling other teams to standardize

Executive View

Service Mesh

Developer Velocity

- Maximize time to innovate
- Evolve application independent of infrastructure
- Add abstraction layer for business domains vs. technical interfaces

Auditability & Visibility

- Address production requirements for compliance and security
- Reduce tool sprawl for metrics systems
- Set foundation with open standards for metrics collection

Operational Evolvability

- Organizes one “view” agnostic of environment
- Accommodate more complex application behaviors
- Secure as close to zero trust as possible

Consul Enterprise Service Mesh

Thank You

thomas@hashicorp.com
www.hashicorp.com