Building Multi-tenant Routing and Scaling with Envoy

KubeCon CloudNativeCon North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022

Yiming Peng
Senior SDE, TLM,
AWS
@pymhq

KubeCon CloudNativeCon North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022

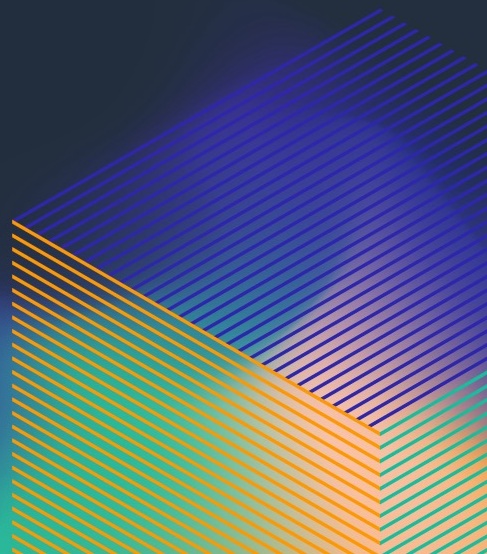
October 24-28, 2022

KUBECON + CLOUDNATIVECON NORTH AMERICA 2022

Building multi-tenant routing and scaling with Envoy

Yiming Peng

Senior Software Development Engineer
AWS

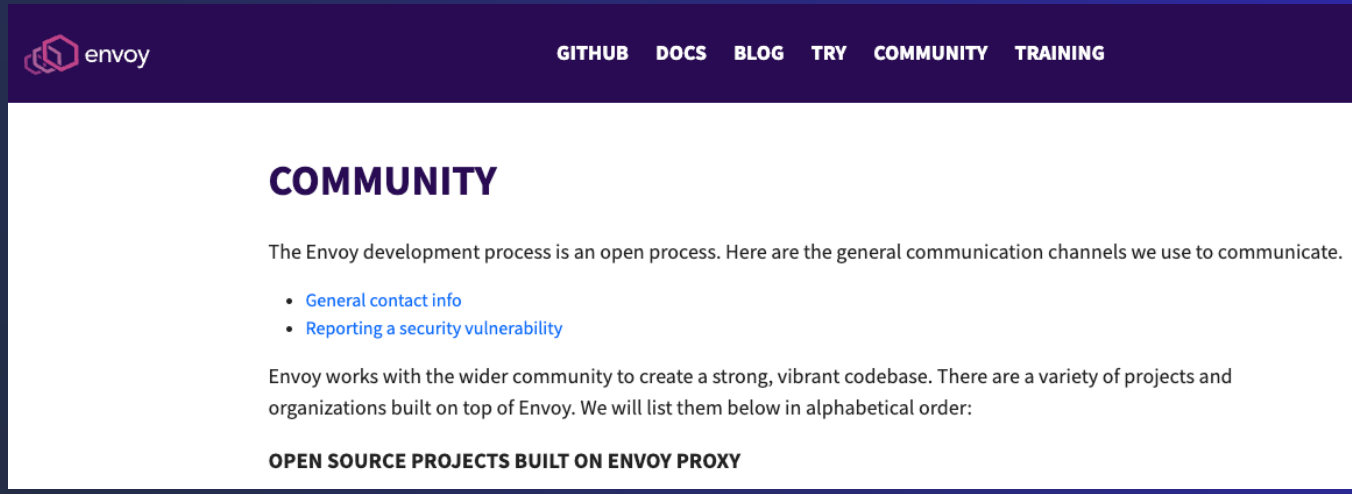


About me

- Senior Software Development Engineer at AWS based in Seattle, WA
- Founding engineer of AWS App Runner – built stealth-mode product from 0 to 1
- Products working on: AWS App Runner, Amazon ECS, AWS Fargate, AWS Elastic Beanstalk
- Expertise areas: Cloud native, containers, serverless, open source
- Open-source enthusiast: Founder and maintainer of **CloudNative & Serverless Meetup**: github.com/CloudNative-Serverless-Meetup
- Contacts: @pymhq (GitHub and Twitter)

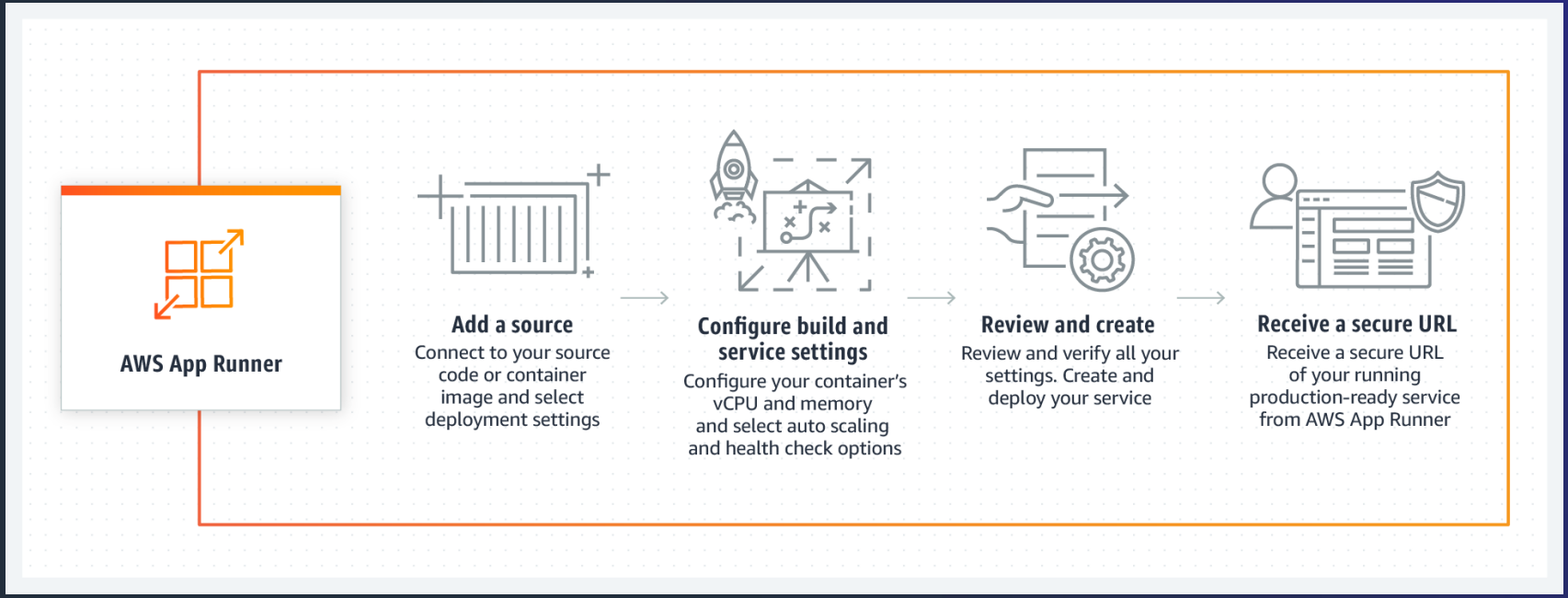
Motivation

- Share the journey from the end user point of view for Envoy
- Give back to community – share experience and lessons learned; hope brings value and helps community growth
- Appreciate support from Envoy community
- Platform for discussion and communication

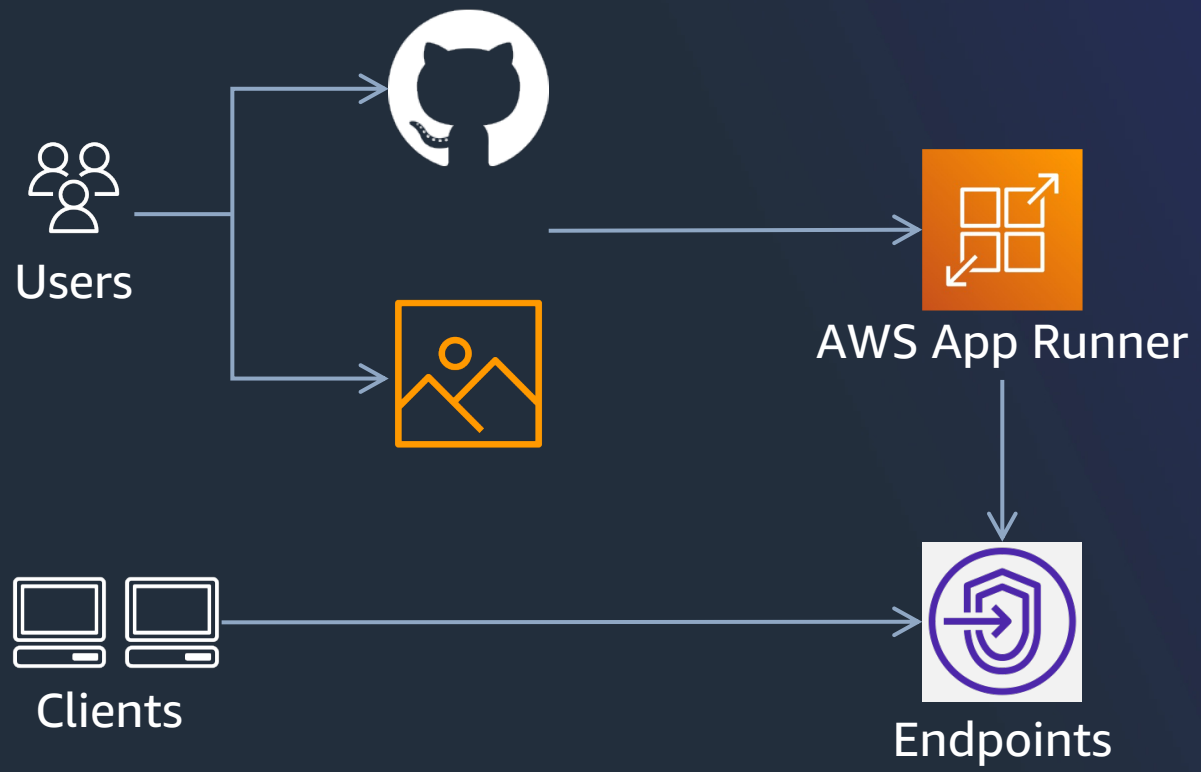


About AWS App Runner

- AWS App Runner (2021 GA) a fully managed service that makes it easy for developers to quickly deploy containerized web applications and APIs – at scale and with no prior infrastructure experience required
- Official website: https://aws.amazon.com/apprunner/



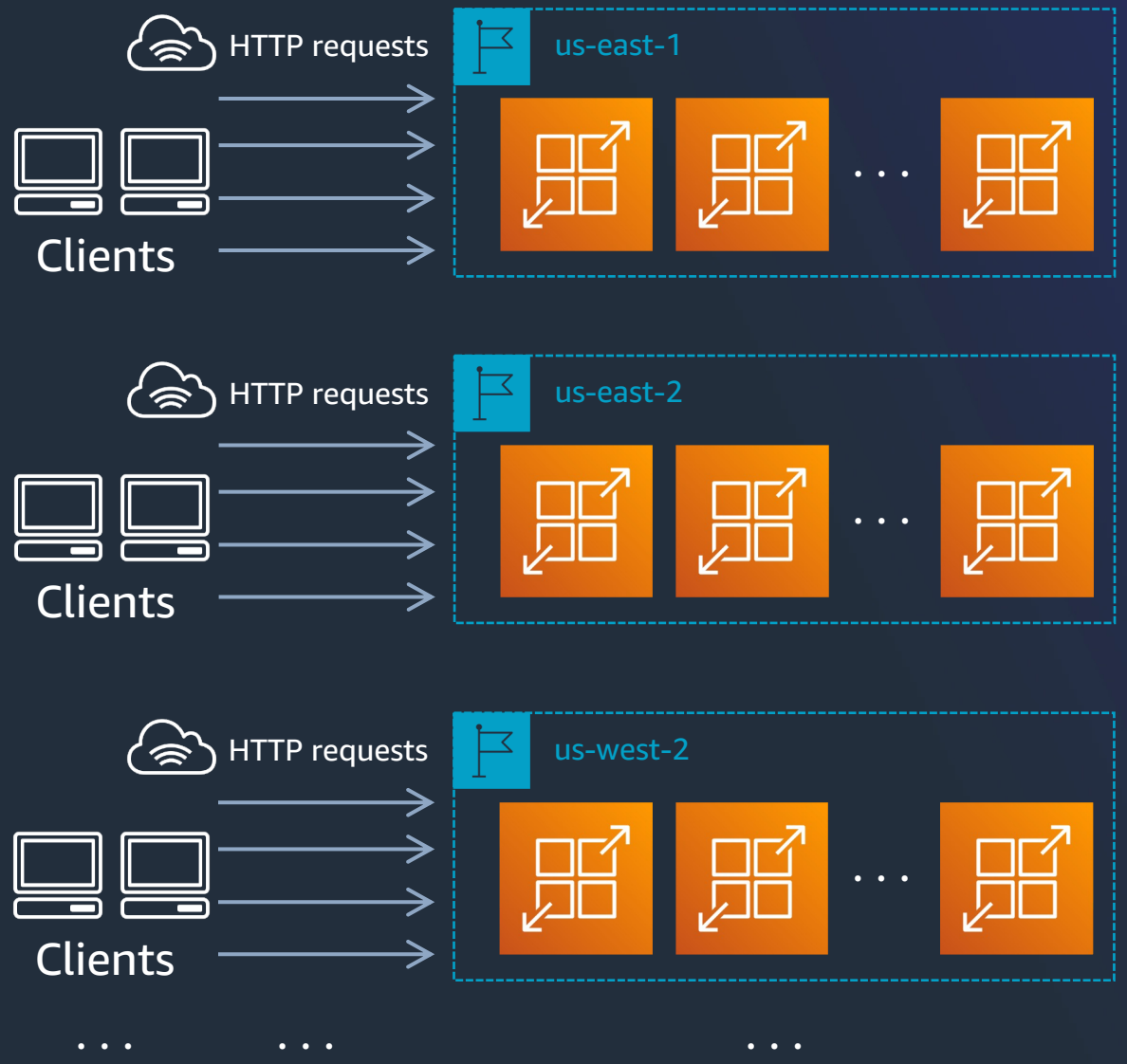
User point of view: App Runner



Managed experience

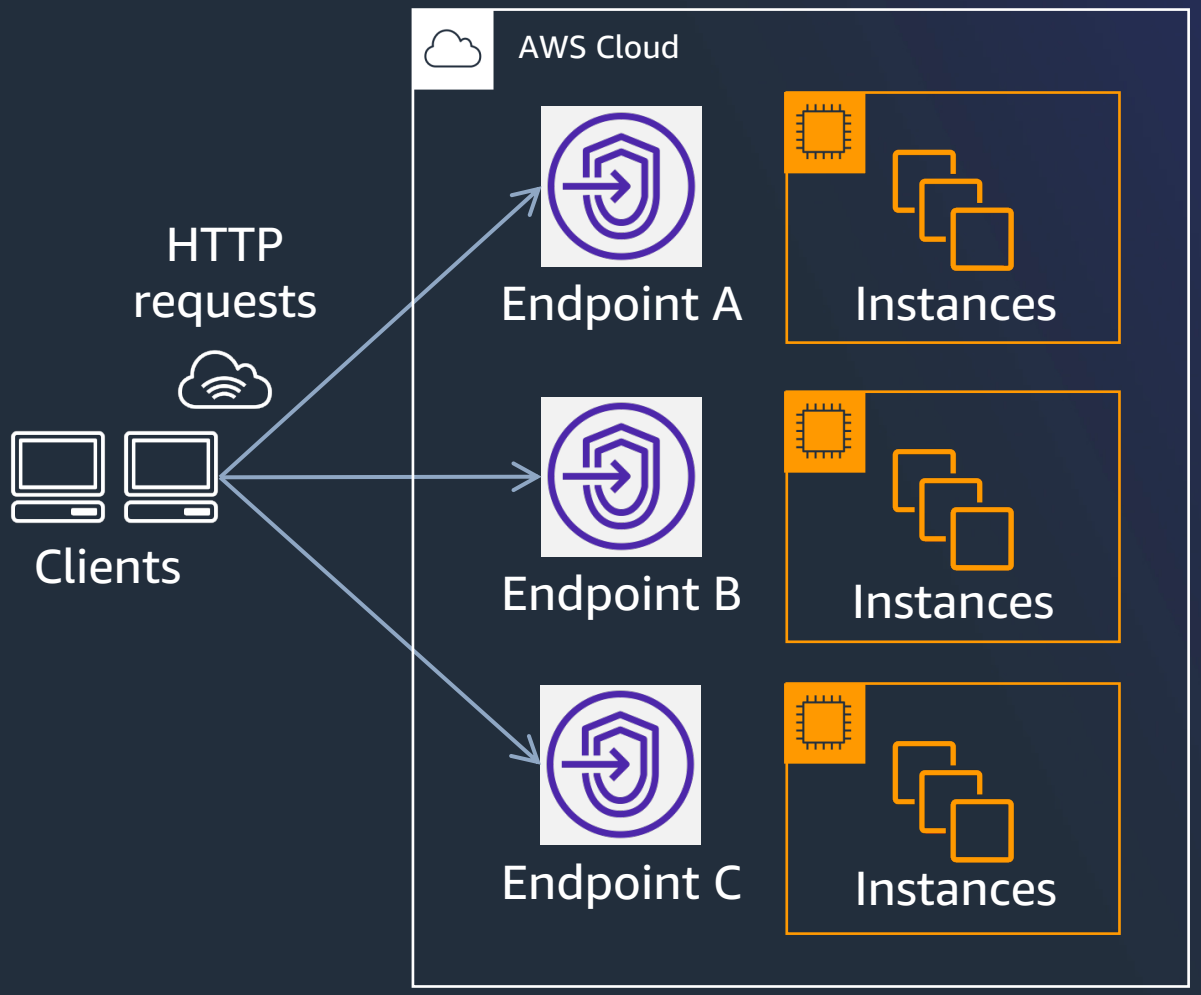
- App hosting
- Request routing
- Load balancing
- Automatic scaling
- Networking
- Observability
- CI/CD
- Safe deployment (blue-green)
- And more

User point of view: Workloads



- API/web application
- HTTP requests
- Concurrent requests
- Long-time running servers

Responsibility of request router



To support managed user experience and managed experience on request routing and load balancing, App Runner needs a request router on the backend to help underlying multi-tenant request routing, different AWS Regions, and customer traffic load balancing

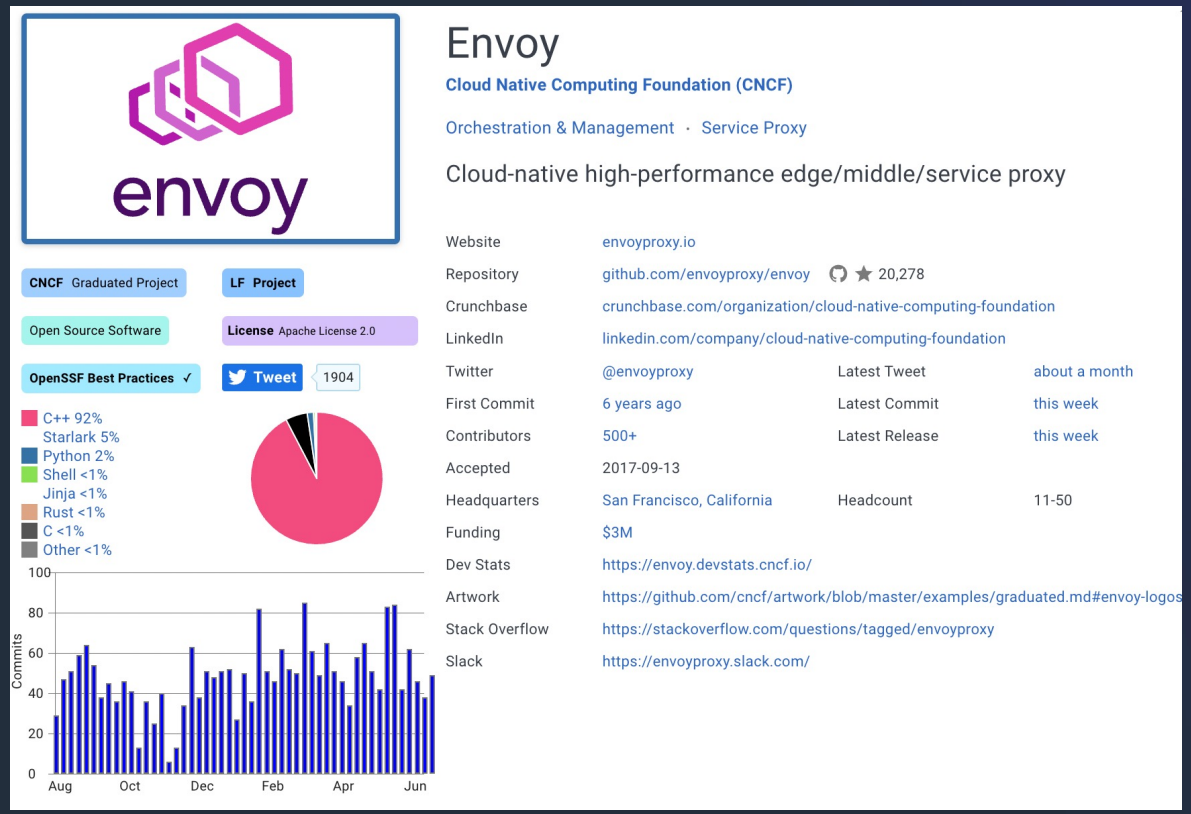
Product requirement

- Multi-Regions and Region-expansible
- Multi-tenant
- Automatic scaling
- Safe deployment
- Observable

Technical challenges

- High throughput
- High performance
- Availability
- Reliability
- Extensibility
- Security
- Large number of concurrent connections
- Observability

Envoy-based request router



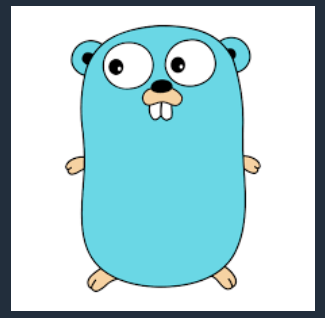
https://www.envoyproxy.io/

Envoy is an open-source edge and service proxy, designed for cloud-native applications

Features

- Out-of-process architecture (lightweight and portable)
- HTTP L7 routing support
- gRPC ready
- Best-in-class observability
- Microservice friendly
- Managed open-source version available (AWS AppMesh)

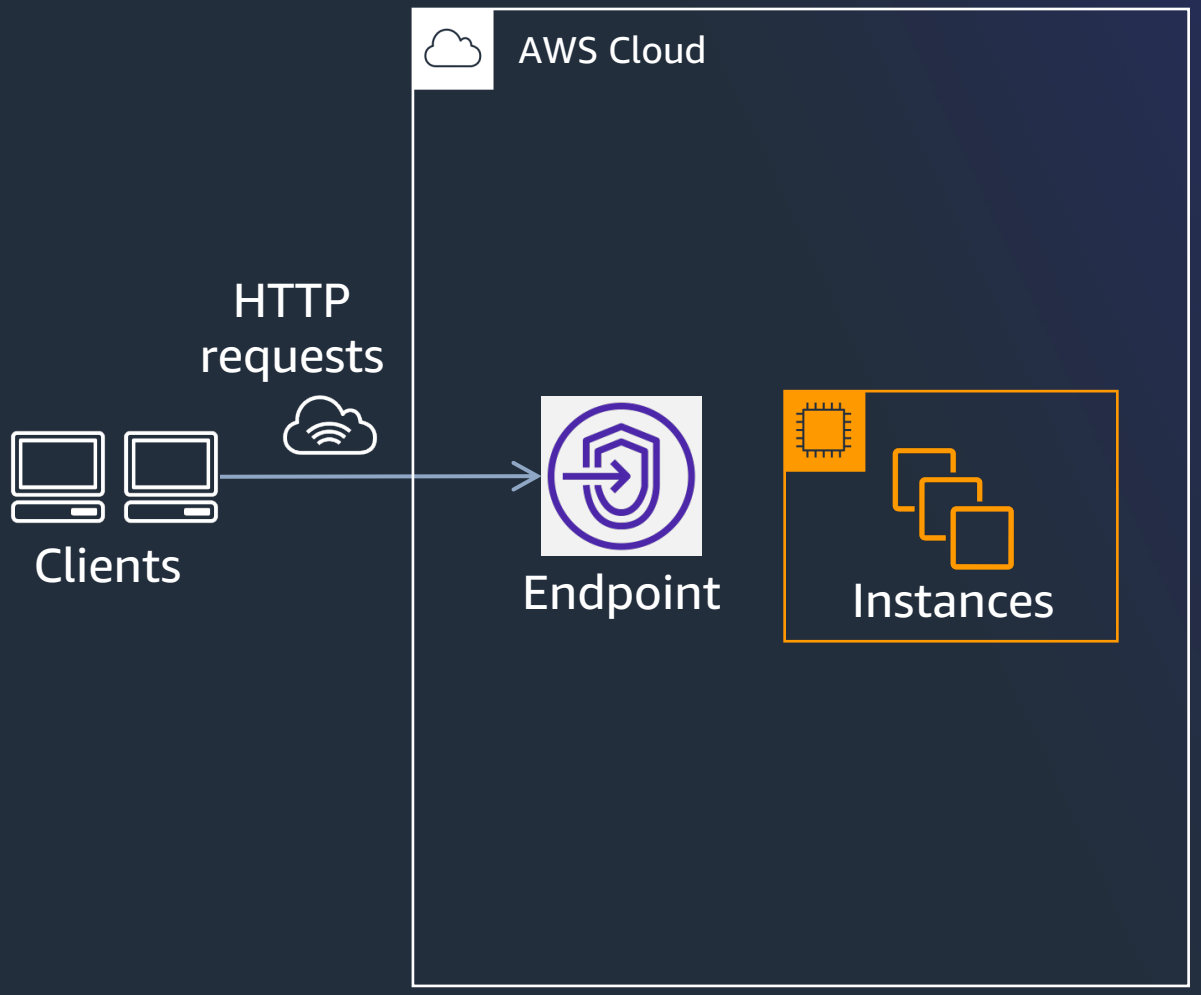
Envoy-based request router



Out-of-process architecture

- Envoy works with any programming language
  - Can write applications in Go, Java, C++, or any other language
  - Can bridge the gap between them – its behavior is identical, regardless of the application's programming language or the operating system they're running on
- The out-of-process architecture is beneficial, as it gives consistency across programming languages/applications stacks – you can get an independent lifecycle

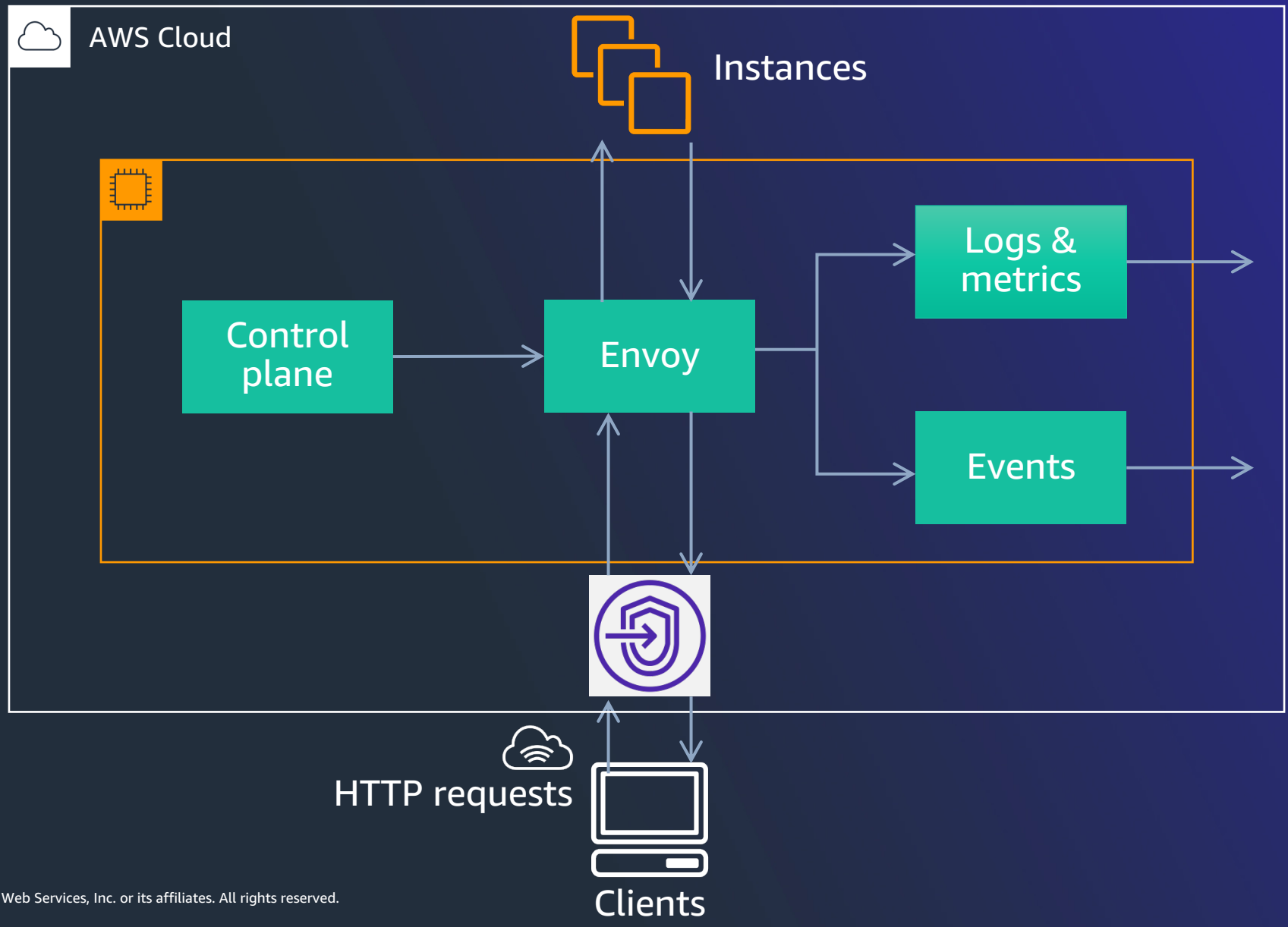
Building Envoy-based request router



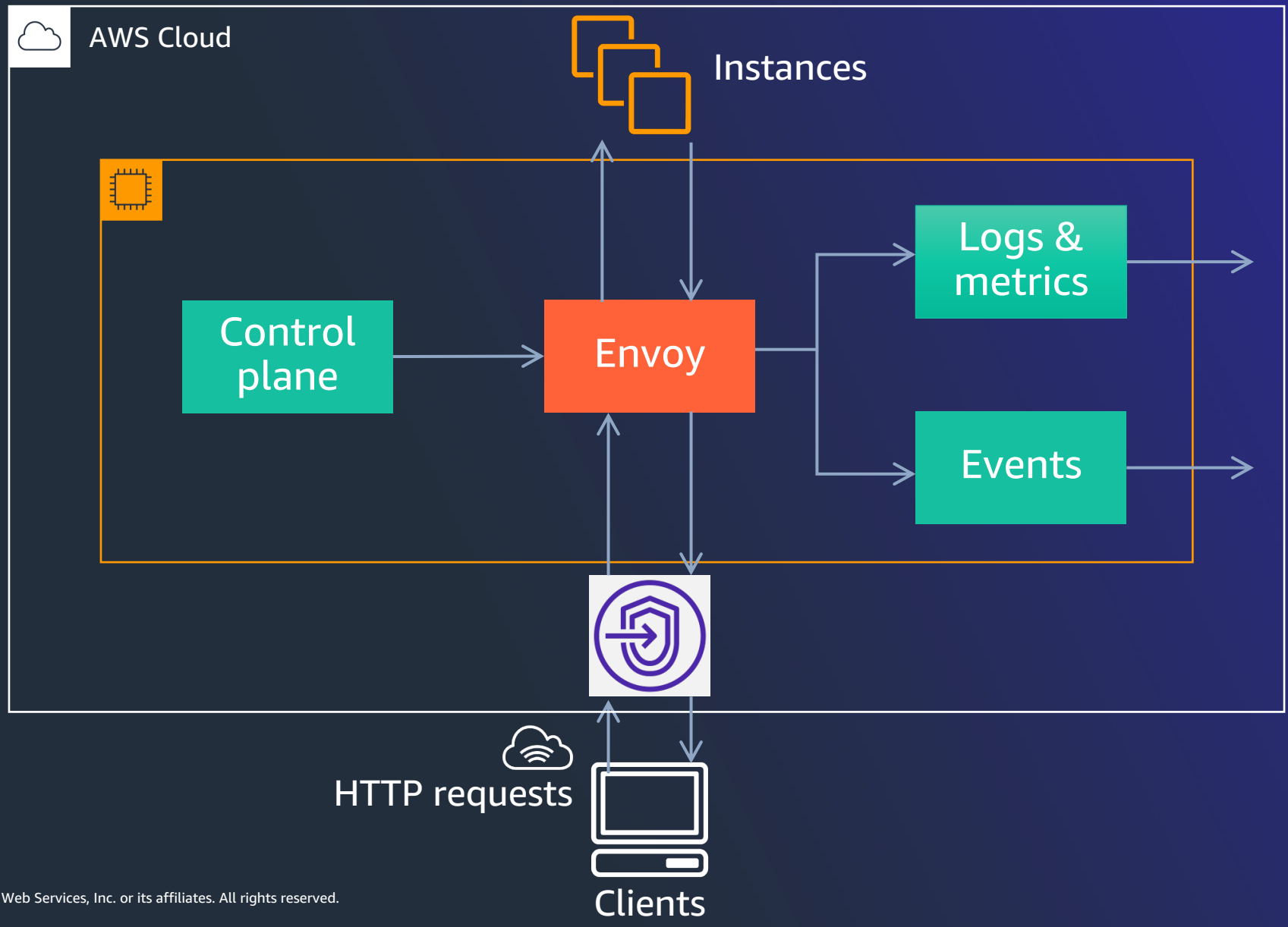
Building blocks

- Bootstrap configuration – dynamic resources → container Instances

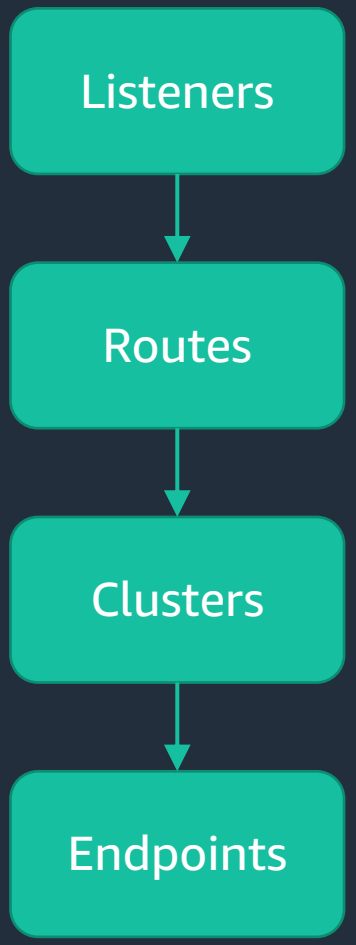
Architecture



Architecture



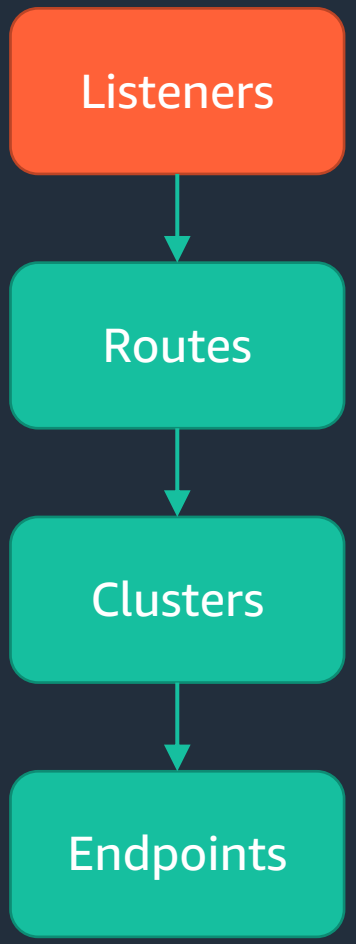
Building Envoy-based request router



Building blocks

- Listeners: Named network location (IP address + port); Envoy receives requests through it
- Routes: Route configuration in HCM filter; match the incoming requests (URI, headers, etc.) and define where traffic is sent
- Clusters: Group of upstream hosts that accept the traffic for a route; list of hosts or IP addresses on which the services are listening
- Endpoints: Service application instances

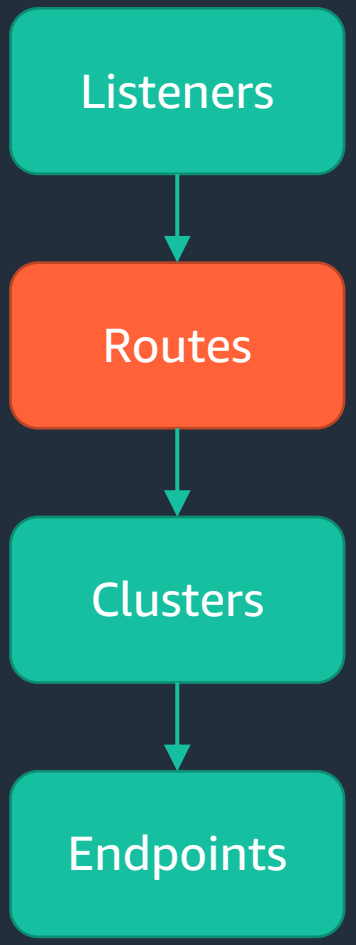
Building Envoy-based request router



Listeners

- Listening external HTTPS requests
- Operate on packet's payload

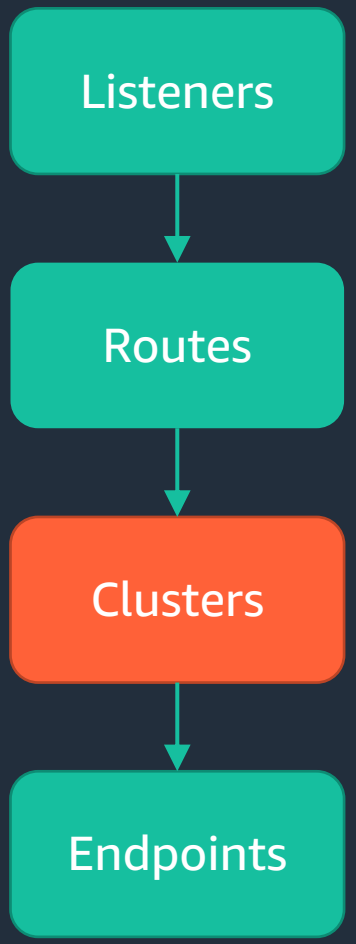
Building Envoy-based request router



Routes

- Envoy will be CRUX of LB and L7 request proxying layer
- Maintain mapping of service URL to application instance IPs; route incoming requests to an appropriate endpoint
- Domain map – for example https://helloworld.us-east-1.awsapprunner.com

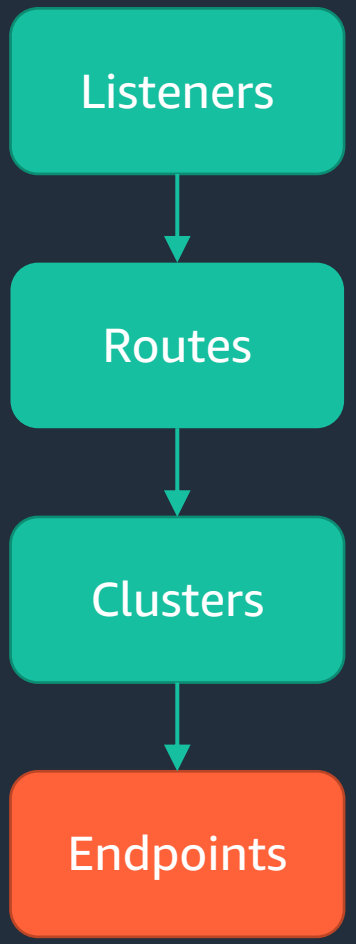
Building Envoy-based request router



Clusters

- Envoy cluster → AWS App Runner service
- Each service version is assigned an Envoy cluster
- Configuration updated via Envoy cluster discovery service (CDS)

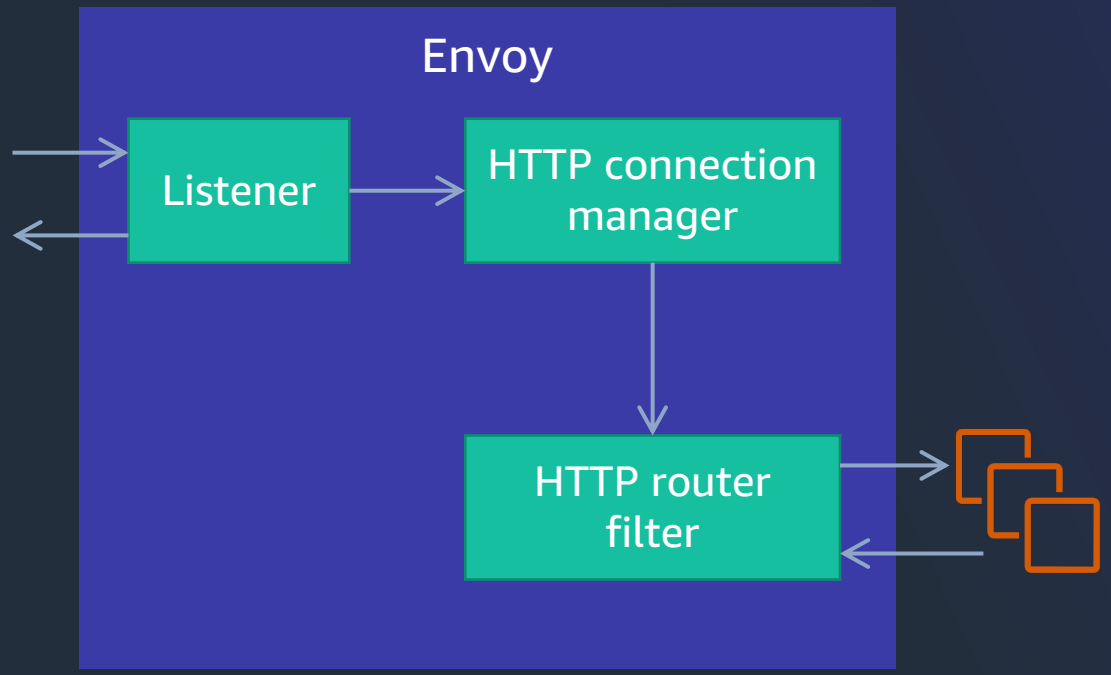
Building Envoy-based request router



Endpoints

- Upstream endpoints updated via Envoy endpoint discovery service (EDS)
- Dynamic and automatic scaling
- Unhealthy endpoints reaper

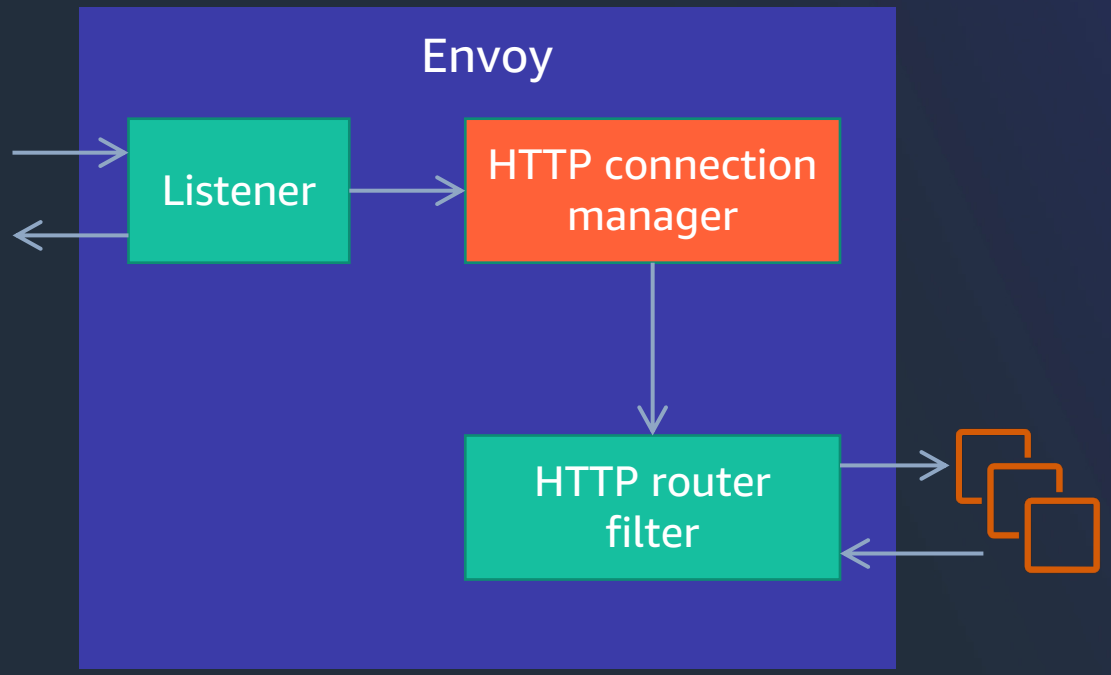
Building Envoy-based request router: HTTP connection manager (HCM)



HTTP connection manager (HCM)

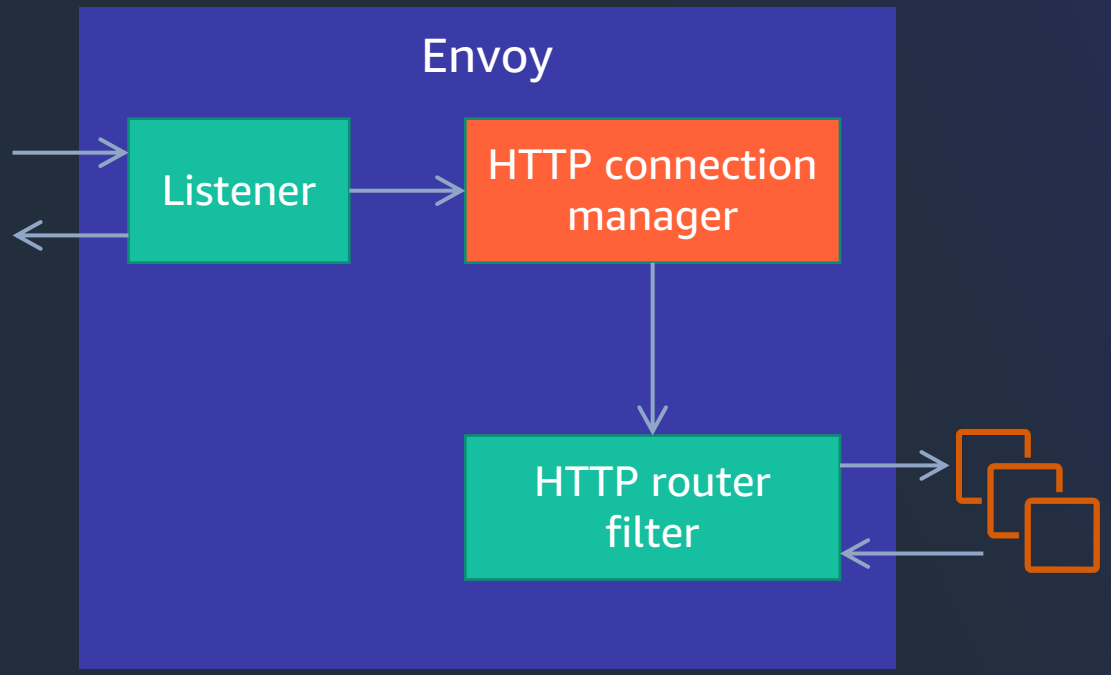
- Network-level filter: Translate raw bytes into HTTP
- Handles
  - Access logging
  - Request ID generation
  - Tracing
  - Header manipulation
  - Retry policy
  - Timeout
  - Traffic weights
  - Route matcher

Building Envoy-based request router: HTTP routing



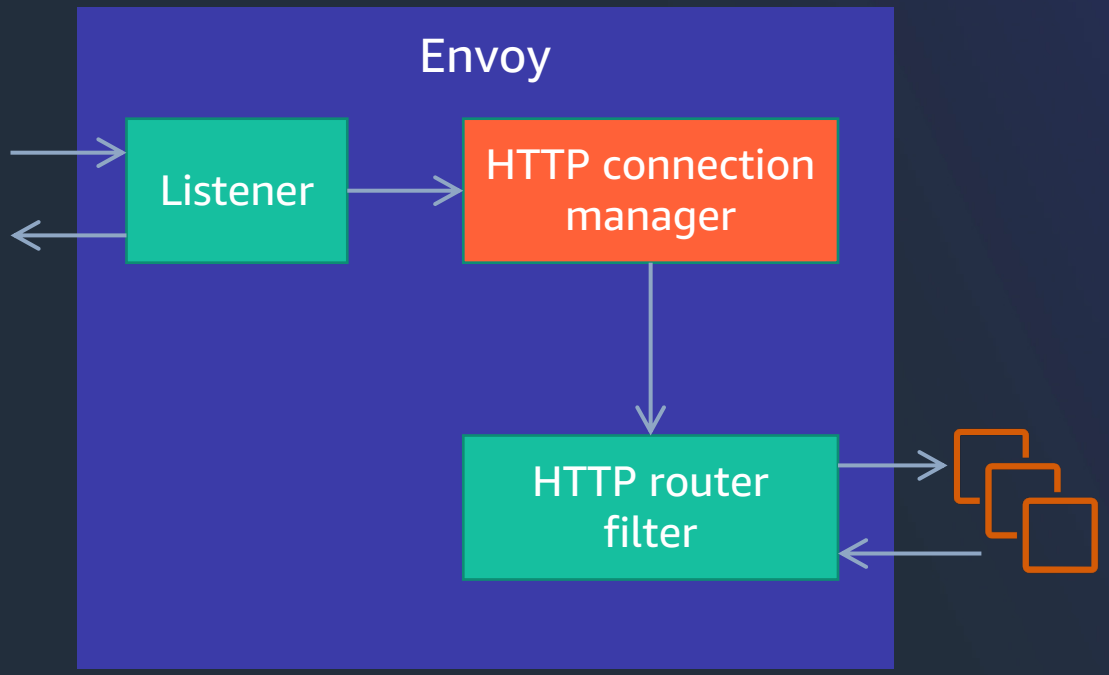
- The router uses the information from the incoming request (e.g., host or authority headers) and matches it to an upstream cluster through virtual hosts and routing rules
- HTTP filters use the route configuration (route_config) that contains the route table

Building Envoy-based request router: Request matching



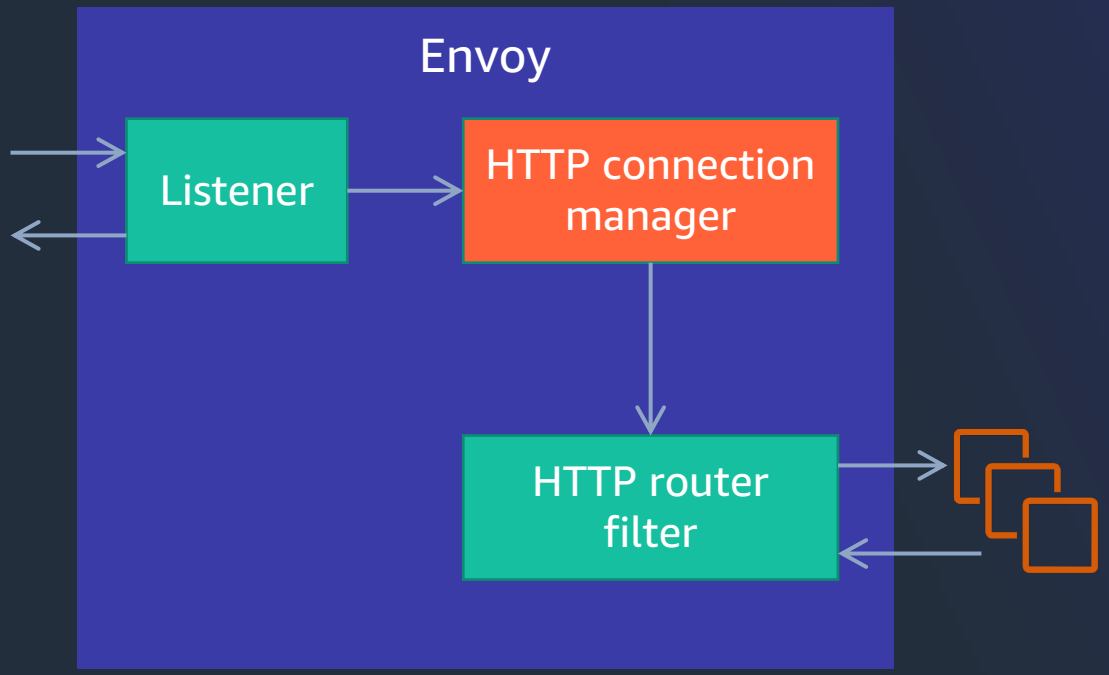
- Set of headers can be specified to match request – router checks request headers against all specified headers in the route config
- Some matches include range, present, string, invert match, etc.
- HCM filters also support query parameter matching, TLS context matching, and gRPC route matching for routing

Building Envoy-based request router: Traffic splitting



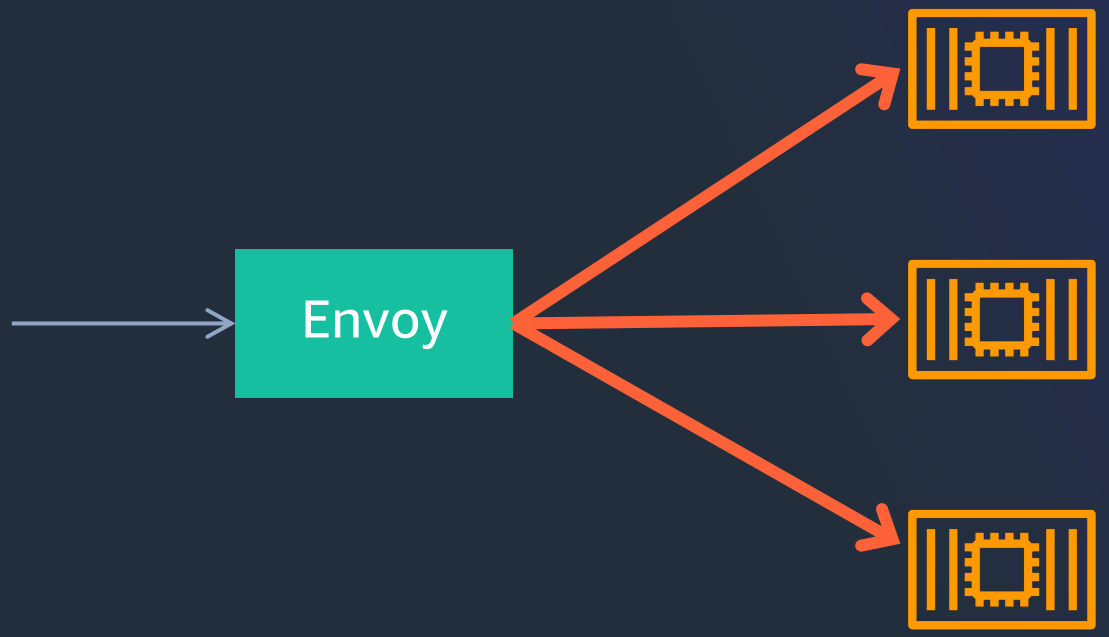
- Envoy supports traffic splitting to different routes within the same virtual host
- We can split traffic between two or more upstream clusters
- Traffic split can be on
  - Runtime percentage
  - Weighted clusters

Building Envoy-based request router: Health check



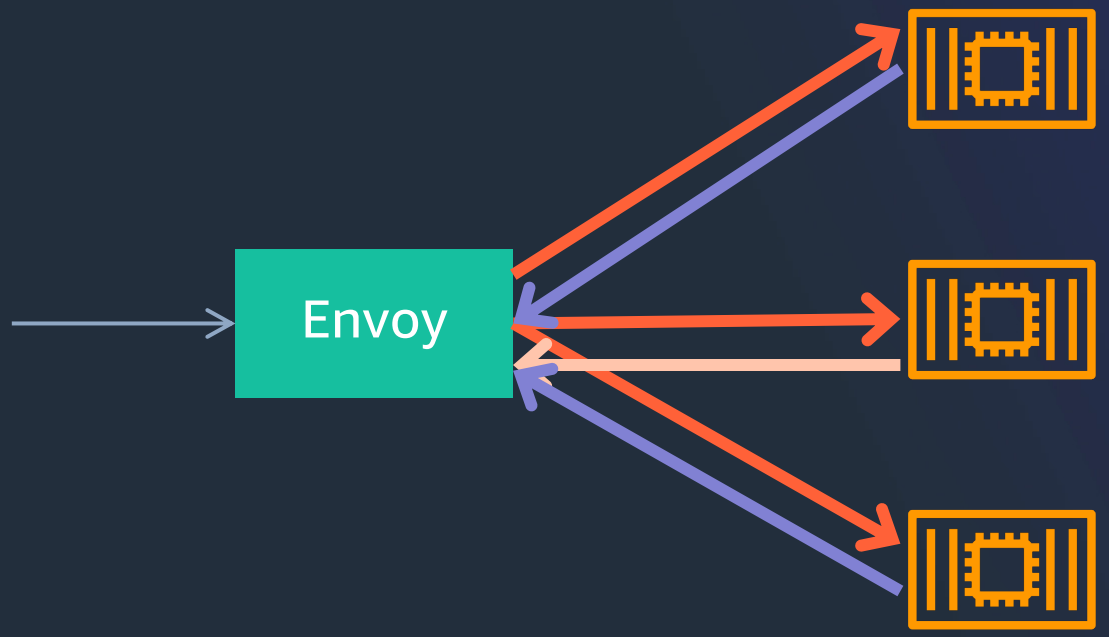
- Envoy supports two types of health checks: Active vs. passive
- With active health checking, Envoy periodically sends a request to the endpoints to check its status
- With passive health checking, Envoy monitors how the endpoints respond to connections

Building Envoy-based request router: Load balancing



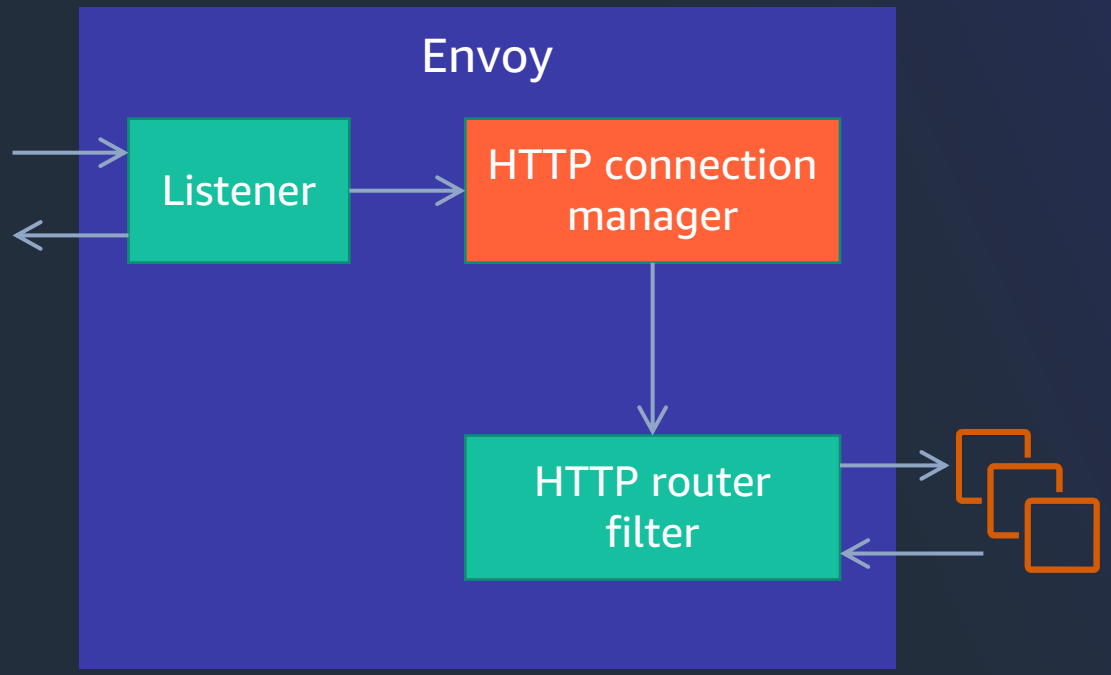
- Load balancing is a way of distributing traffic between multiple endpoints in a single upstream cluster
- The reason for distributing traffic across numerous endpoints is to make the best use of the available resources (cost effective)

Building Envoy-based request router: Retries



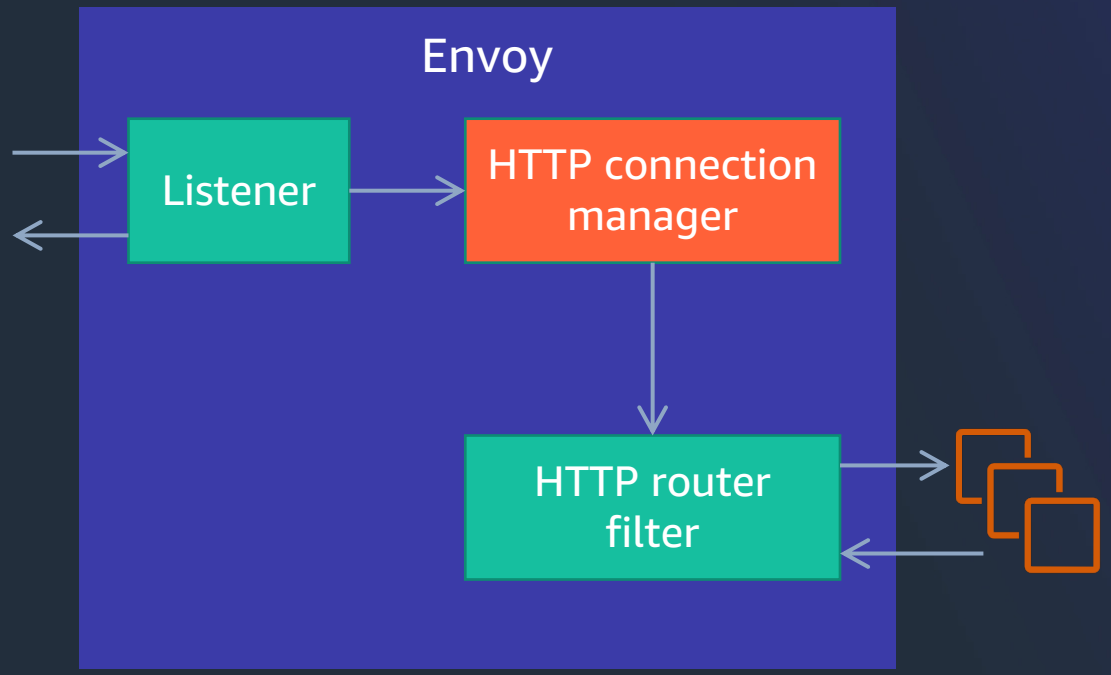
- Envoy allows setting retry policy at the virtual host and the route level based on conditions
- For example, if there is a 429 status code returned from upstream, then retry will happen seamlessly on upstream of other available instance

Building Envoy-based request router: Circuit breakers



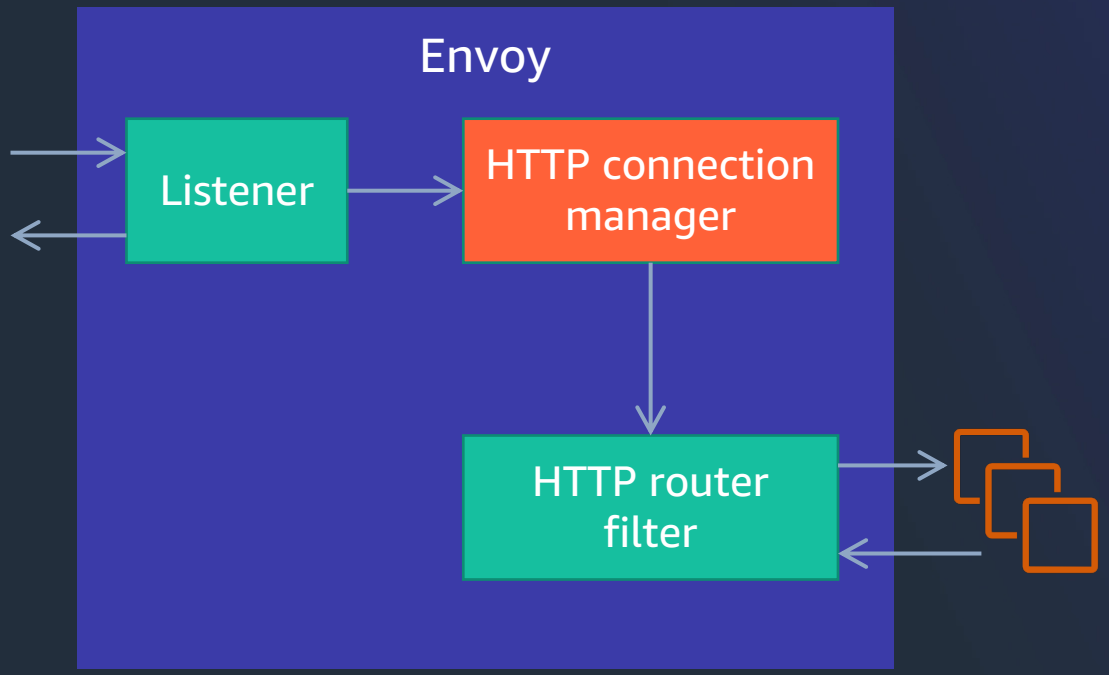
- Circuit breaker pattern prevents additional failures managing access to failing services
- It allows us to fail quickly and apply back pressure downstream as soon as possible
- We can configure the circuit breaker thresholds for each route priority separately and globally

Building Envoy-based request router: Timeout



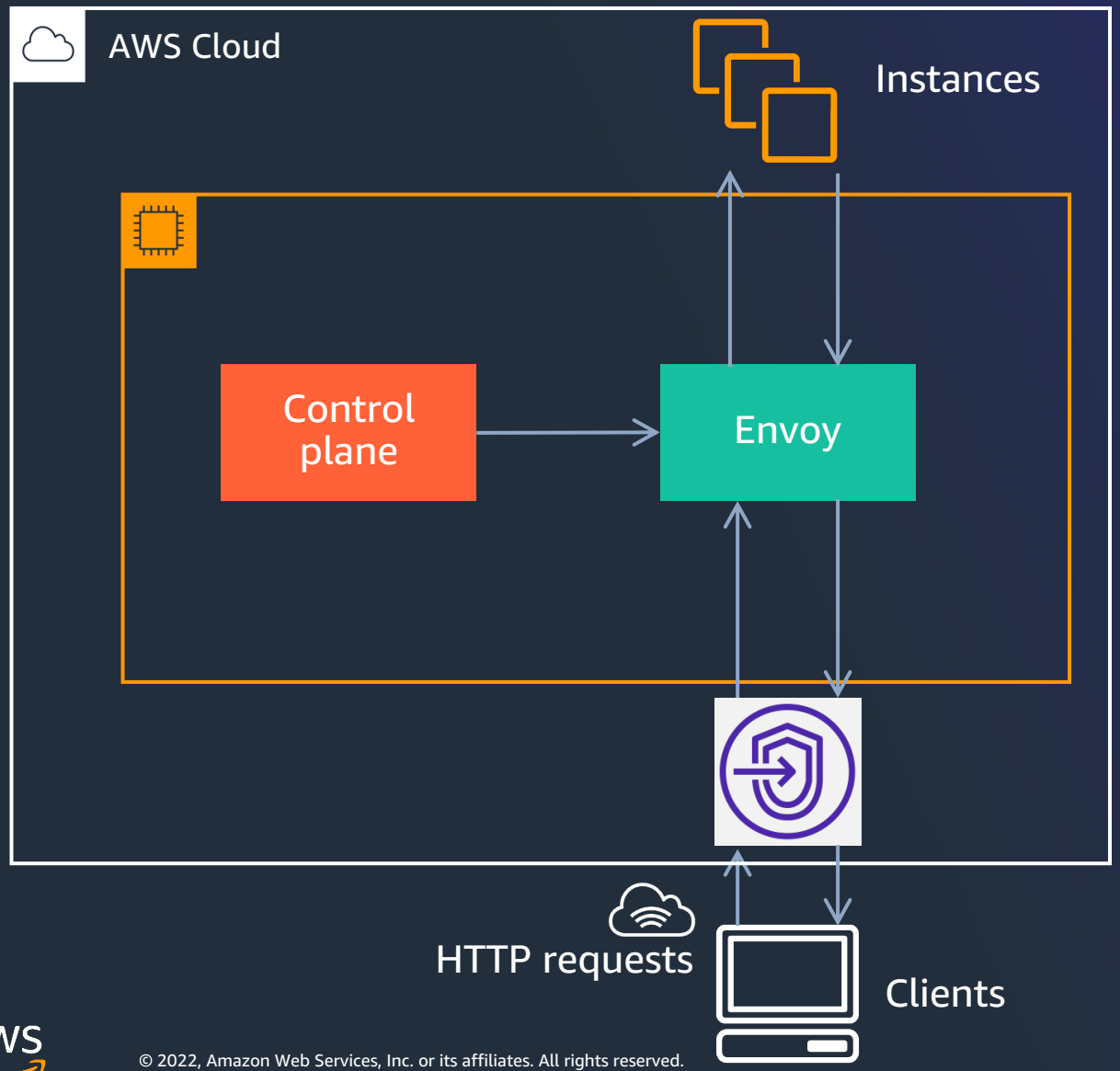
- Envoy supports numerous configurable timeouts that depend on the scenarios you're using the proxy for
- Example timeouts
  - `request_timeout` specifies the amount of time Envoy waits for the entire request to be received
  - `idle_timeout` represents when a downstream or upstream connection gets terminated if there are no active streams (default 1 hour)

Building Envoy-based request router: Rate limiter



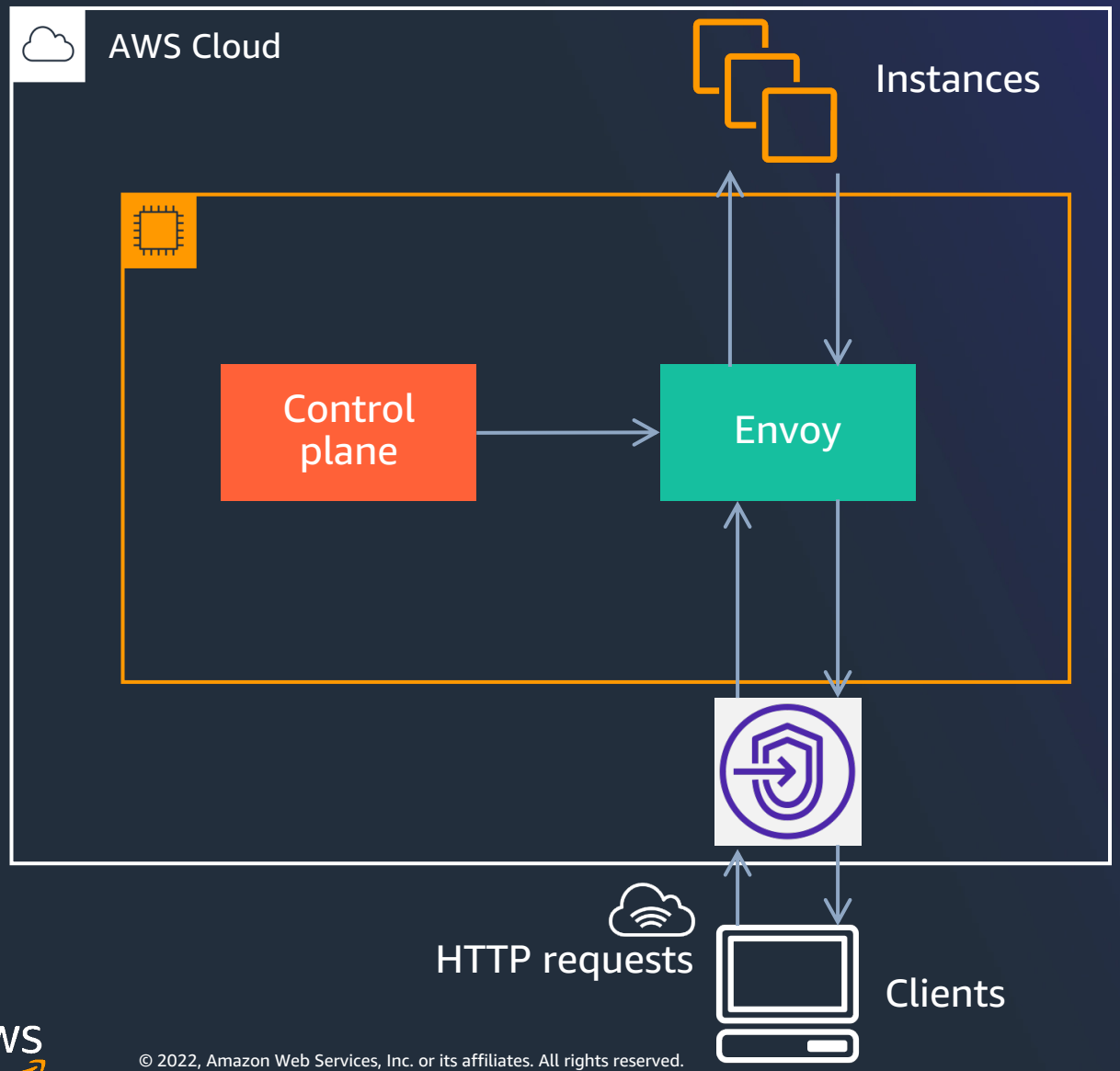
Envoy supports global and local rate limits to upstream endpoints

Building Envoy-based request router: Control plane



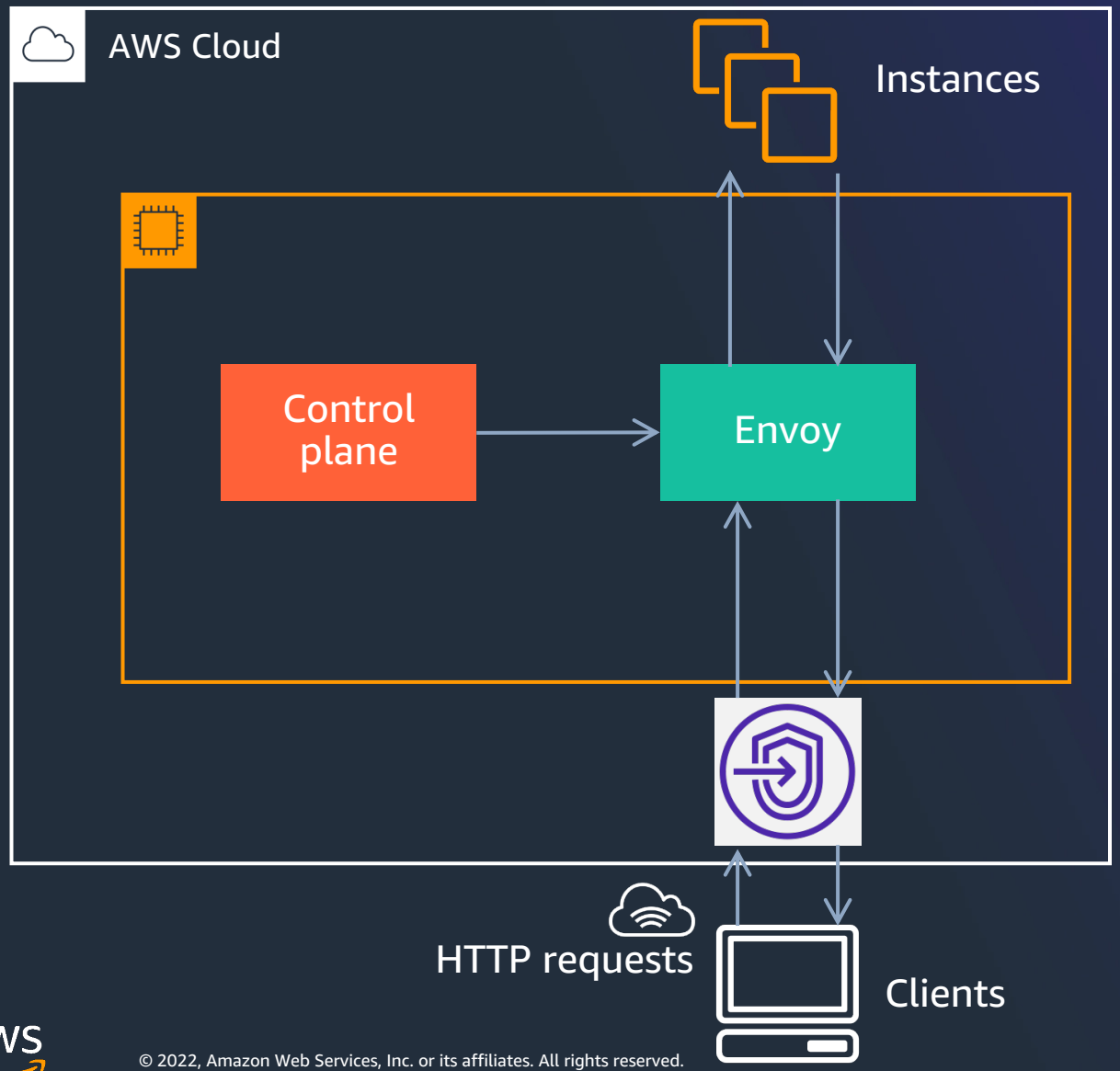
- Custom control plane for Envoy
- Responsible for
  - Service cluster discovery (CDS) and service endpoint discovery (EDS)
  - Route and other configuration
- gRPC server listening on defined port

Building Envoy-based request router: Envoy runtime



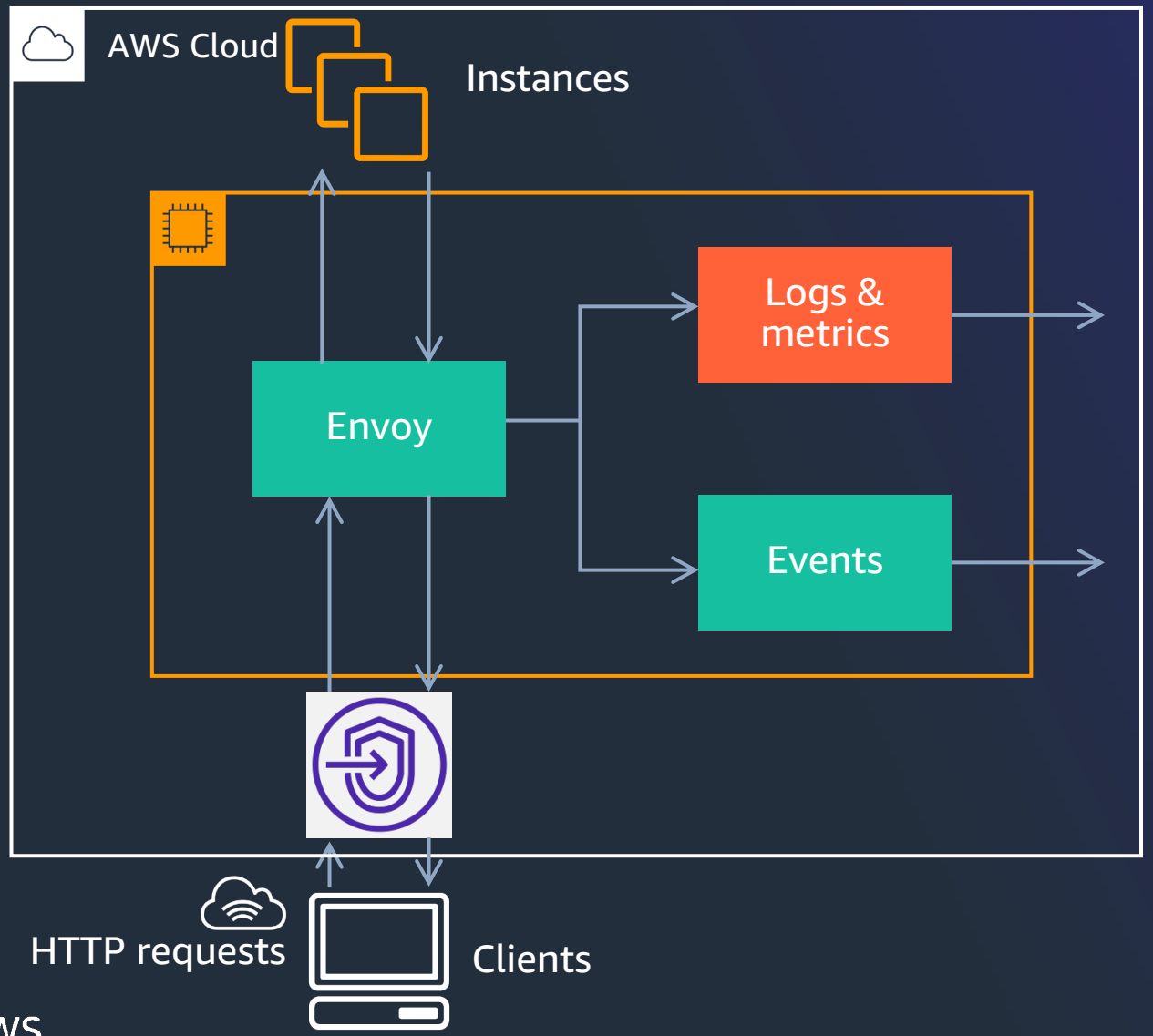
- Envoy needs to be updated with latest service information at runtime
- Envoy discovers dynamic resources by querying management servers using xDS (discovery service APIs)
- Dynamic resources:
  - Clusters
  - Endpoints
- Envoy requests resources via subscriptions by initiating gRPC streams from Envoy client to the management server

Building Envoy-based request router: Envoy runtime



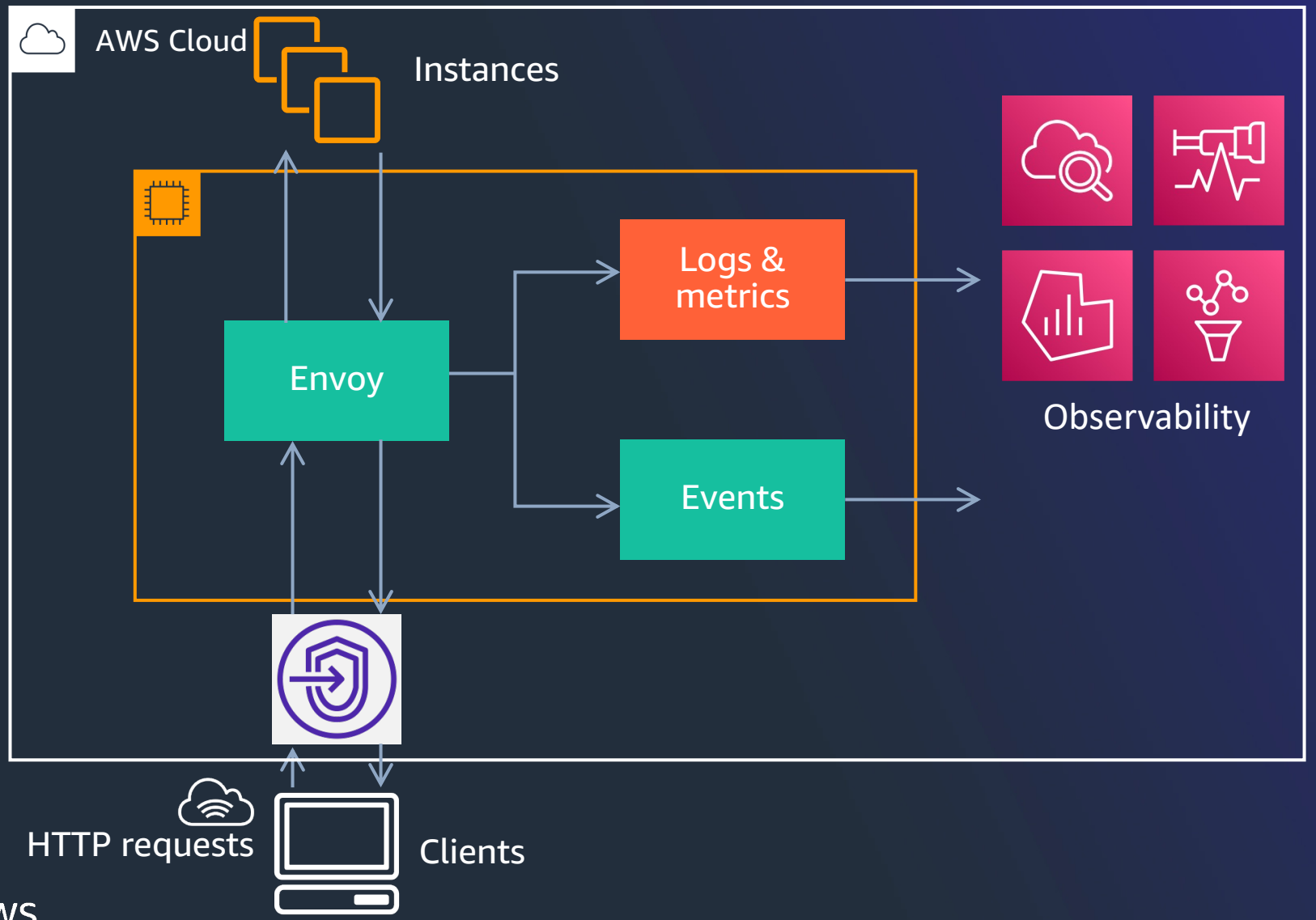
- Clusters configured on Envoy through **CDS xDS API –** bit.ly/3RZmMq6
- Endpoints are configured using **EDS xDS API –** bit.ly/3S0f8MR
- AWS App Runner uses CDS API to dynamically update clusters and EDS to update endpoints per cluster

Building Envoy-based request router: Logs and metrics



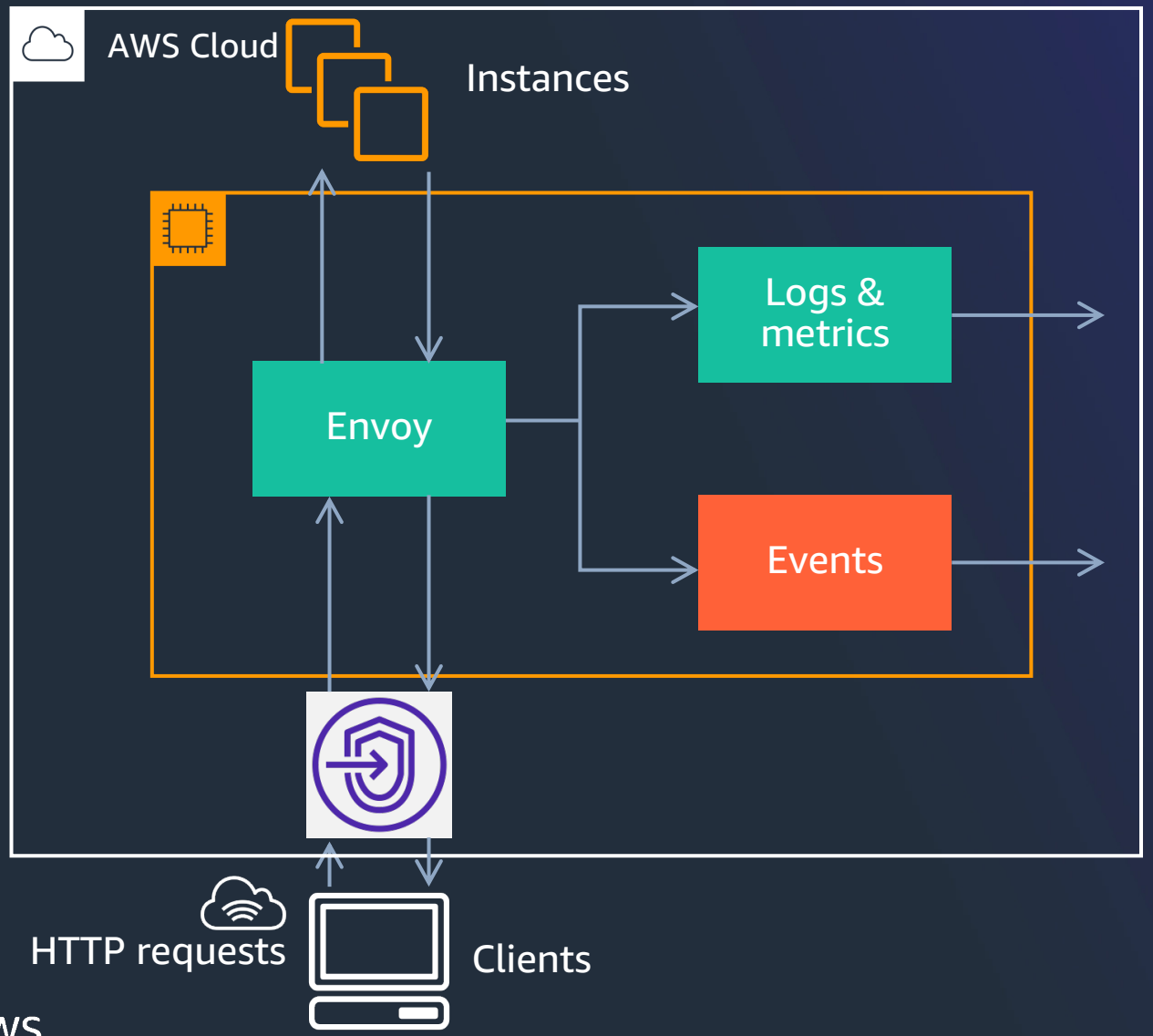
- Responsible for Envoy logs and metrics collection
- Envoy configures exposed metrics locally
- A stats_sink that receives metrics emitted from Envoy
- Metrics such as 2xx, 4xx, 5xx request count are eventually forwarded to logs and metrics service
- gRPC service listening on defined port

Building Envoy-based request router: Logs and metrics



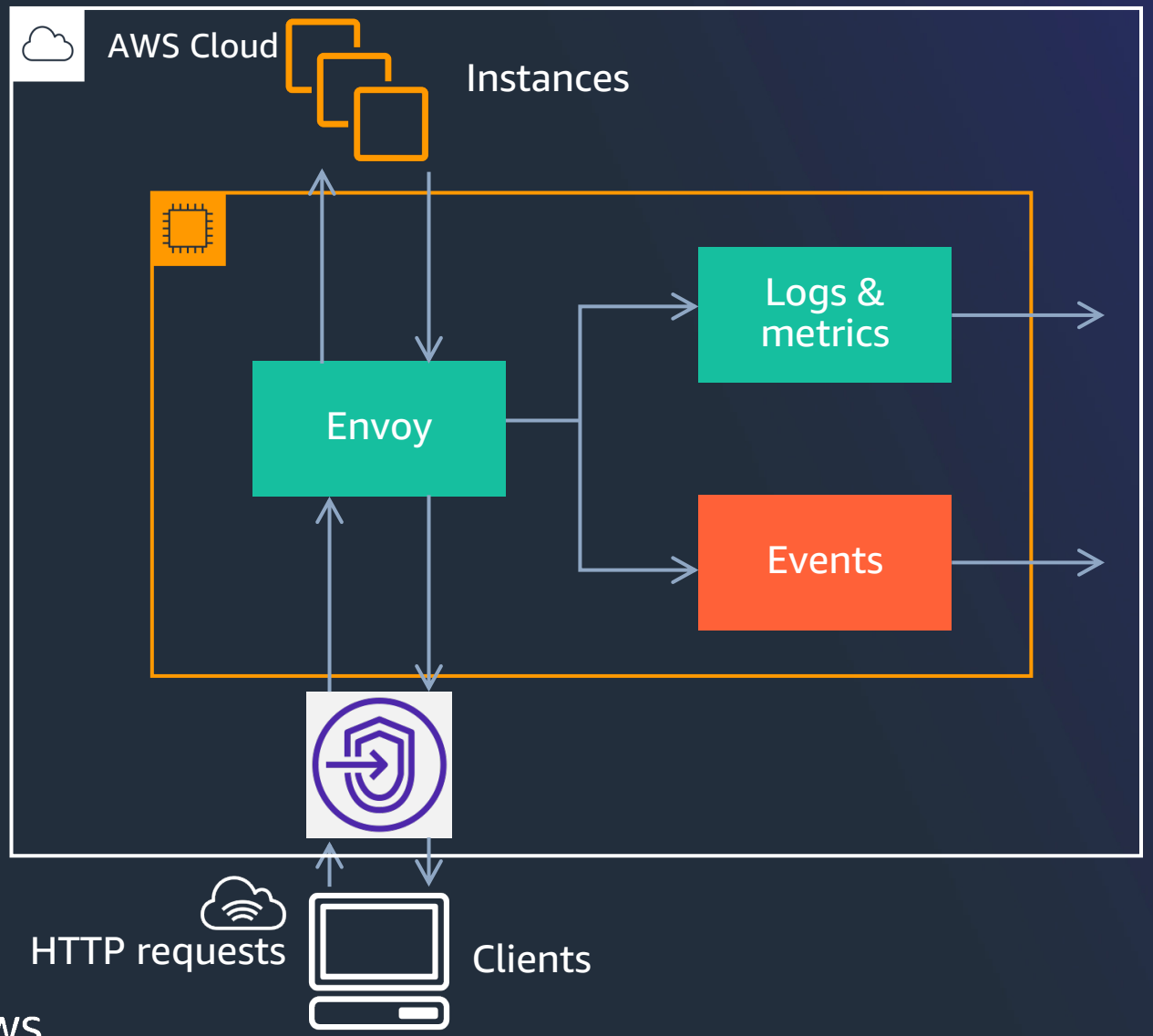
- Amazon CloudWatch
- Amazon Managed Service for Prometheus
- AWS Distro for OpenTelemetry
- Amazon Managed Grafana
- AWS X-Ray
- And more

Building Envoy-based request router: Event streamer



- Responsible for reporting service status and health check results
- gRPC service listening on defined port

Building Envoy-based request router: Event streamer



Used in supporting service

- Safe deploy (blue-green)
- Automatic scaling
- Load balancing
- Request routing
- . . .

Building Envoy-based request router: Production readiness

- Function readiness
- Capacity management – CPU/memory consumption
- Performance and scalability
- Security (e.g., DDoS)
- Other operational readiness – monitors, alarms, etc.

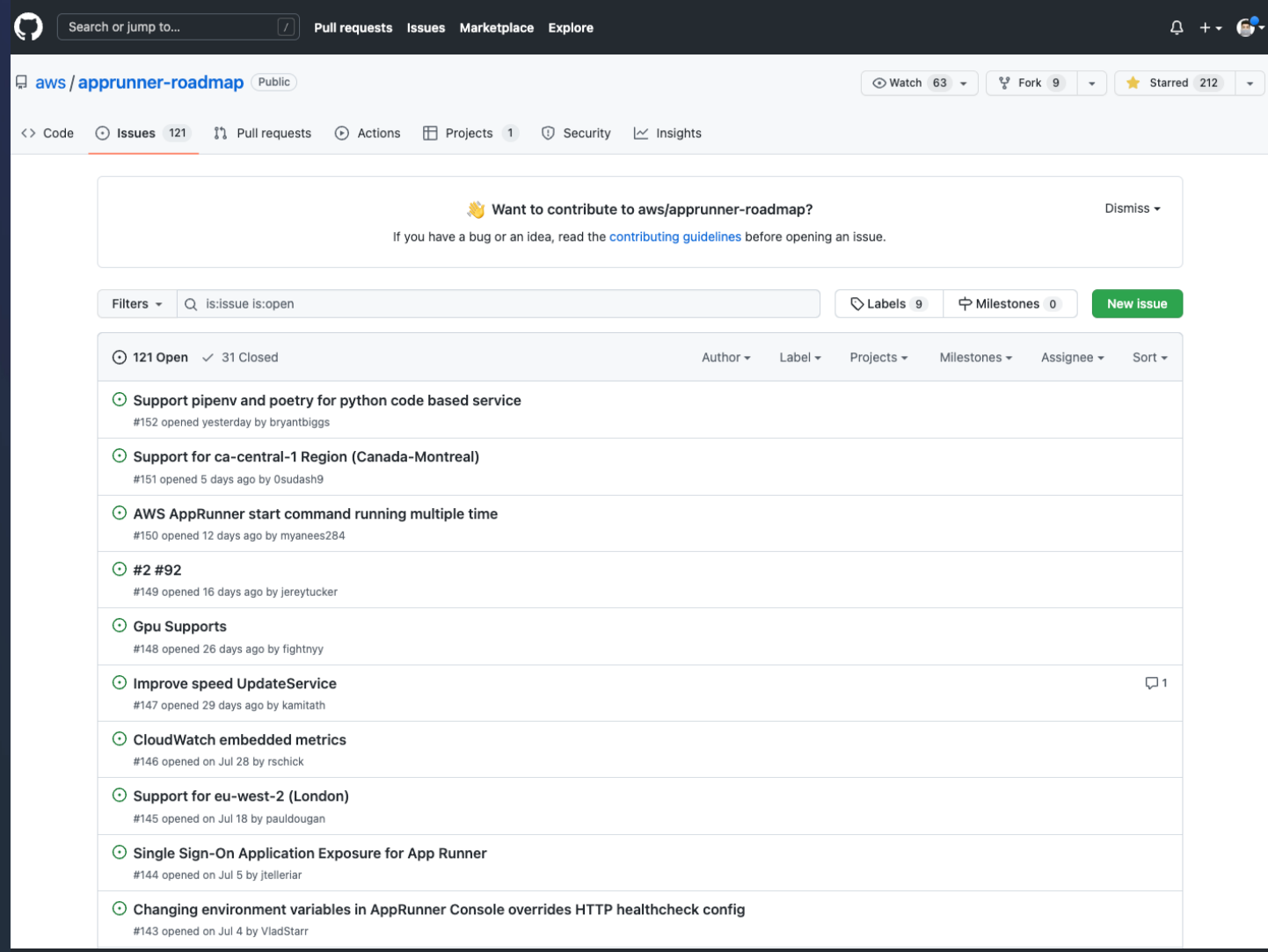
Wrap up

Envoy brings value in building request router with many features:

- High throughput and performance request routing and load balancing
- Portability and extensibility
- Reliability and availability
- Observability

More on roadmap

- GitHub issue 2: Allow private endpoints for App Runner services – github.com/aws/apprunner-roadmap/issues/2
- GitHub issue 52: Set additional X-Forwarded-Headers – github.com/aws/apprunner-roadmap/issues/52
- GitHub issue 58: AWS WAF support – github.com/aws/apprunner-roadmap/issues/58
- GitHub issue 104: Configurable timeout – github.com/aws/apprunner-roadmap/issues/104



Thank you!

Yiming Peng

 pengyiming

 @pymhq

Please complete the session survey