Introduction to service mesh with Ιστίο (Istio) and Κιάλι (Kiali)

Alissa Bonas

mikeyteva
Who am I?

- Engineering manager in Kiali project at Red Hat
- B.Sc in Computer Science and Communication
- Developer and tech lead - Java, Ruby on Rails and more...
- Also worked at HP Software and Mercury Interactive
- Open source involvement
Evolution of application architecture

... How did we get to service mesh?
Monolith application

Single unit of executable
  =
Application
  =
Single process
Application modules

- Alerts
- Handle HTTP requests
- UI
- Data processing

Application
Microservices

Language agnostic

Scaled separately

Upgraded separately
A shift in Application Packaging and Runtime
Containerizing an app
Run multiple containers
Orchestrate containers

- Run many containers on multiple hosts
- Scale - manage several instances (replicas) of the same container
- Manage a container based environment
Container orchestration platforms

- Kubernetes
- OKD (Openshift)
Kubernetes building blocks (some...)

- **Pod** - a group of one or more containers, with shared storage/network
- **Deployment** - manages pod definition and defines replicas of pods
- **Service** - an abstraction, an access point to a set of Pods
  - Sometimes called a **microservice**
Microservices - the Kubernetes way

Access point = microservice

Service A

Container

Pod

Service B

Container

Pod

Instance 1

Instance 2

Code
High Complexity
Multiple points of failure
Challenges

- How requests are routed between services?
- How do I detect failures and downtime?
- How to upgrade and test new versions of a service?
- Securing the communication
Service mesh to the rescue
What is a service mesh

- Infrastructure/framework that handles communication between services
- Often implemented as network proxies deployed alongside the microservices
Istio - Ιστίο

Open source service mesh
Istio features

- Load balancing (HTTP, gRPC, TCP...)
- Traffic control (routing rules, retries, timeouts, fault injection, mirroring)
- Secure service-to-service communication
- Access controls (authorization)
- Metrics and traces for traffic
Important Terminology

- **Workload** - anything owning/controlling pods (like a Deployment) or the pods themselves
- **Service** - a microservice
- **Application** - *label* “app” on a pod/service
- **Version** - *label* “version” on a pod/service
Before Istio

POD A

- Container
  - Routing code
  - Circuit breaker code
  - Business logic code

POD B

- Container2
  - Routing code2
  - Circuit breaker code2
  - Business logic code2
Istio

POD A

Container

Routing code

Circuit breaker code

Business logic code

POD B

Container2

Routing code2

Circuit breaker code2

Business logic code2
Sidecar Proxy

- A proxy which is deployed next to each instance of your service (inside a pod)
- Envoy open source proxy is the current sidecar
- Proxy is transparent to application code
- Can be automatically injected to pod on creation
Sidecar Proxy in Istio and Kubernetes

Before Istio, no sidecar

With sidecar
With Istio - sidecar intercepts all traffic

Configuration is transparent to the services and not part of the code
Different routing scenarios

- A/B testing
- Traffic shifting
- Canary deployment (an example of traffic shifting)
- Mirroring traffic
Weighted Routing with Istio - A/B

Proportion of traffic routed to a version is independent of number of instances of that version
Weighted Routing - Canary

Proportion of traffic routed to a version is independent of number of instances of that version
Matching Routing with Istio

Service A

Pod

Service B

User Alissa

Pod Version 1

All other users

Pod Version 2
Mirroring traffic

Service A

Pod

Service B

Pod Version 1

Pod Version 2

Real traffic

Copy of traffic

Response disregarded
"Anything that can go wrong will go wrong"

(Murphy’s law)
Chaos engineering anyone?
Chaos engineering with Istio

- **Inject delays**
  - Simulate network latency
  - Simulate an overloaded service

- **Define aborts**
  - Simulate failure in a service (return a predefined HTTP Error)
  - A good alternative for a manual shutdown
Inject delay

Service A

Pod

Service B

Add 7 seconds delay

Pod
Version 1
Instance 1

Instance 2

Pod
Version 2
Inject Error

Service A

Pod

Service B

Pod

Version 1
Instance 1

Instance 2

Version 2

Work as usual for all the users

Return Error 500 for user Alissa

Service B

Pod

Version 2

Pod

Version 1
Instance 1
Circuit breaker

- Set a connection pool to limit connections and requests
- **Example:** “Set a connection pool of 100 connections with no more than 10 req/connection to service A”
Outlier detection

- Classify instances as healthy/unhealthy
- Eject unhealthy instances for a defined timeframe which can be increased over time
- Example: “Scan all pods every 5 mins, any instance that fails 7 consecutive times with 5XX error code will be ejected for 15 minutes.”
Security and RBAC

- In/out traffic of the mesh is disabled
  - Defining a Gateway ingress/egress to enable
- mTLS can be defined on multiple levels
  - All mesh, specific service, etc.
- Authorization and authentication
Configuration objects

- **VirtualService** ≠ Kubernetes service
  - Rules for how requests to a service are routed within service mesh
  - Routing logic, load weighting, chaos injection

- **DestinationRule**
  - Configures policies to be applied to a request **after** VirtualService routing has occurred
  - Load balancer, circuit breaker

- **MeshPolicy**, Gateway, ServiceEntry and more...
Configuration Yaml example

All Istio objects are CRD (CustomResource Definition)
New set of challenges

- How do I see what’s in the system?
- Is there any traffic now?
- Is routing configured for service A?
- Is my configuration even valid?
- Is security on?
- Is the app healthy?
Kiali - Κιάλι

Open source

Istio service mesh observability
Kiali Features

- Visualize mesh connections and traffic
- Service and application health
- Configure routing via UI
- Validate Istio configurations
- View metrics and traces
- Visualize security configuration
Demo based on Bookinfo example
(Bookinfo example)
Kiali Features
Overview page
Routing

Create Matching Routing

Matches:
- headers:
  - exact: Header value...

Route:
- Workload: reviews-v1

Advanced Options:
- LoadBalancer: ROUND_ROBIN
Runtime metric dashboards
Configuration validations
Visualizing security
Tracing (integration with Jaeger)
Connect with the community

Kiali.io
KialiProject
github.com/kiali

Istio.io
IstioMesh
github.com/istio
Icon credits

- Twitter by Lubos Volkov, the Noun Project
- Light Bulb by artworkbean, the Noun Project
- Magnifying Glass by Musket from the Noun Project
- Questions by Rediffusion from the Noun Project
- Mug by Alex Getty from the Noun Project
- Diamond by MarkieAnn Packer from the Noun Project
- Box by Cornelius Danger from the Noun Project
Thank you!

mikeyteva