Advanced Kubernetes - Service Mesh

Part II - Service Mesh

Marius Dieckmann[1,2,3]

Marius.Dieckmann@computational.bio.uni-giessen.de

EMBL-EBI Cloud Consultants[4]

cloud-consultants@ebi.ac.uk

[1] Justus-Liebig-Universität Gießen

[2]NFDI4Biodiversity

[3]de.NBI

[4]EMBL-EBI

February 9, 2021

What is a service mesh?

- A service mesh connects individual components of applications
- Usually implemented as a orchestrated network of proxies
- Allows advanced management of network traffic
  - Automated mTLS between components
  - Traffic introspection
  - Load-Balancing
  - Traffic splitting
  - (sometimes) Destination rules
- Various implementations
  - Linkerd
  - Istio
  - ...
- Normally handles HTTP and gRPC
- Multicluster gateway

mTLS and traffic introspection

- All services in a mesh use mTLS
- Certificates and rotation are managed by the service mesh
- Some meshes (Istio) allow destination rules based on mTLS
- Connections are automatically secured using the proxy
- The service-mesh can intercept and monitor data at its proxies
  - Display data routes
  - Requests statistics
  - Header inspection

Load-balancing and traffic splitting

- Load balancing based on traffic metrics
- Both for HTTP/1.1 & HTTP/2
  - Traffic routing on HTTP/2 based on requests
- Split traffic between services
  - Split traffic between canary and production release
  - Fault injection

Canary Deployments

- Named after canaries in coal mines to detect toxic gases (carbon monoxide)
- Introduced to test updates in production environment
- Old and new version run in parallel, traffic is gradually shifted to new version
- While shifting, metrics (reponse time, request-success-rate) are measured
- Shifting is based on these metrics and can be halted/rolled back automatically
- Can be configured:
  - Metrics and limit values
  - Timings
  - Additional HPA on canary releases

Blue/Green Deployments

- Old deployment strategy (Often used in mainframe systems)
- Two separate environment; one is production, the other staging
- Updates are deployed to staging
- Staging is permanently tested internally
- Requests are shifted from one deployment to the other
- The other deployment becomes the production environment
- The former production deployment is kept as backup and can be shifted back to
- After a while the former production deployment becomes the new staging deployment
- The cycle is permanently repeated

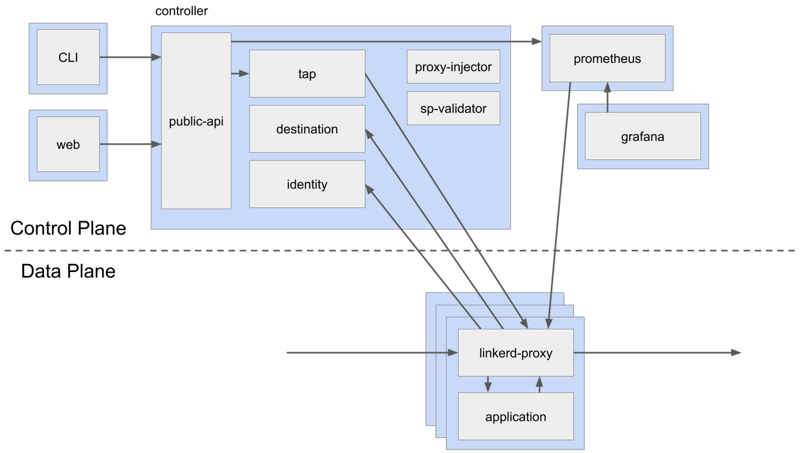
Automated deployment updates

- To fully automate a release cycle it is necessary to update the image of a deployment
- Various tools available: we use keel
- Deployments will be updated automatically based on SemVer rules
- Updates can be registered via webhook or polling
- Update notifications can be send to various services like rocketchat/slack/...

Linkerd

- Implementation for a service mesh
- Baseline functionality
- Simple graphical dashboard
- Comes with its own proxy implementation
- Features:
  - HTTP, HTTP/2 and gRPC proxying
  - Configurable timeout and retry handling
  - mTLS
  - Traffic observation
  - Load-Balancing
  - Automated proxy injection
  - Traffic-split
  - Canary/Blue-Green deployments with Flagger
  - ...

Linkerd implementation



Canary Deployments Linkerd I

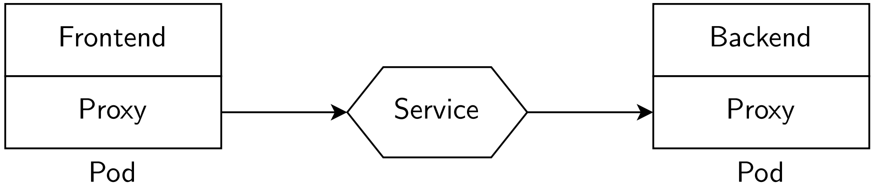


Figure: Basic application topology: frontend and backend components are connected via a service and the traffic is routed via local sidecar proxies

Canary Deployments Linkerd II

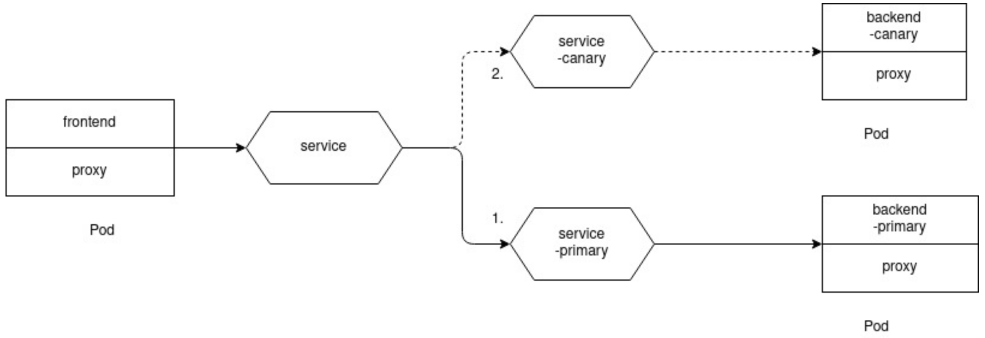


Figure: Canary application topology: frontend uses the same service as in the basic scenario, the backend is splitted into two components, a canary and a primary pod. Linkerd with flagger can route the traffic between them and update images as required.

Hands-on part 2