APISIX in Kubernetes environments

Łukasz Biegaj, 2022.12.21

Functional requirement: HTTP support

Almost every project requires an HTTP API interface, web frontend or both.

Existing support in multitude of dev frameworks: Spring, ExpressJS, Django, etc.

Load Balancers can be easily used for more complex deployments.

Typical solutions

- Serving static content
- Path-based routing
- Load balancing
  - Origin health checks
  - Service Discovery
- Response cache
- Content manipulation
  - i.e. custom HTTP response headers

- Apache HTTP Server
- NGINX
- Varnish
- HAProxy

Enter Kubernetes

- Significantly accelerated application development
- More control in the hands of developers.
- Many issues and project needs can be solved directly by dev teams.
- Rapid development cycle encourages experimentation with new technologies.

People like new technologies

- And sometimes they do not need them.

Hey, we need an API Gateway!

What for?

We need to route traffic to multiple applications based on URL path

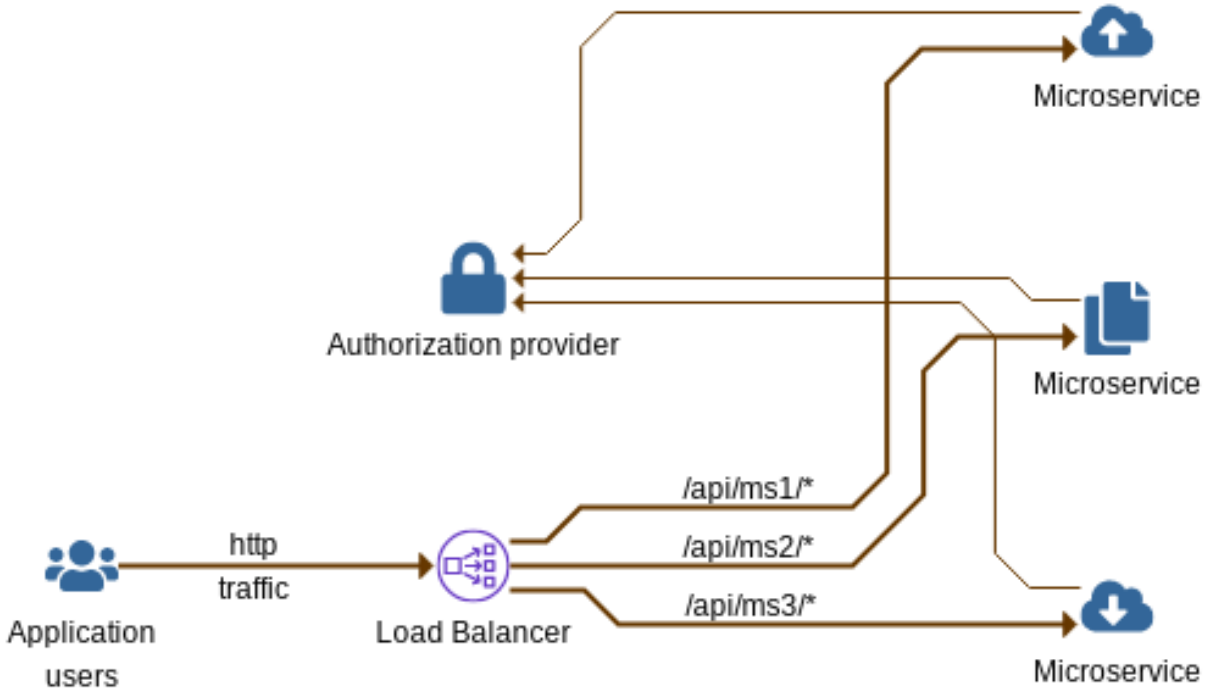
Your existing load balancer can do that just fine

Our use case for an API Gateway

- Multiple dev teams working in the same Kubernetes cluster
- Each dev team developing it's own microservice in different language
- A combination of Java, Nodejs, PHP, .net, etc.
- Common requirement: unified authentication and authorization against common identity provider.

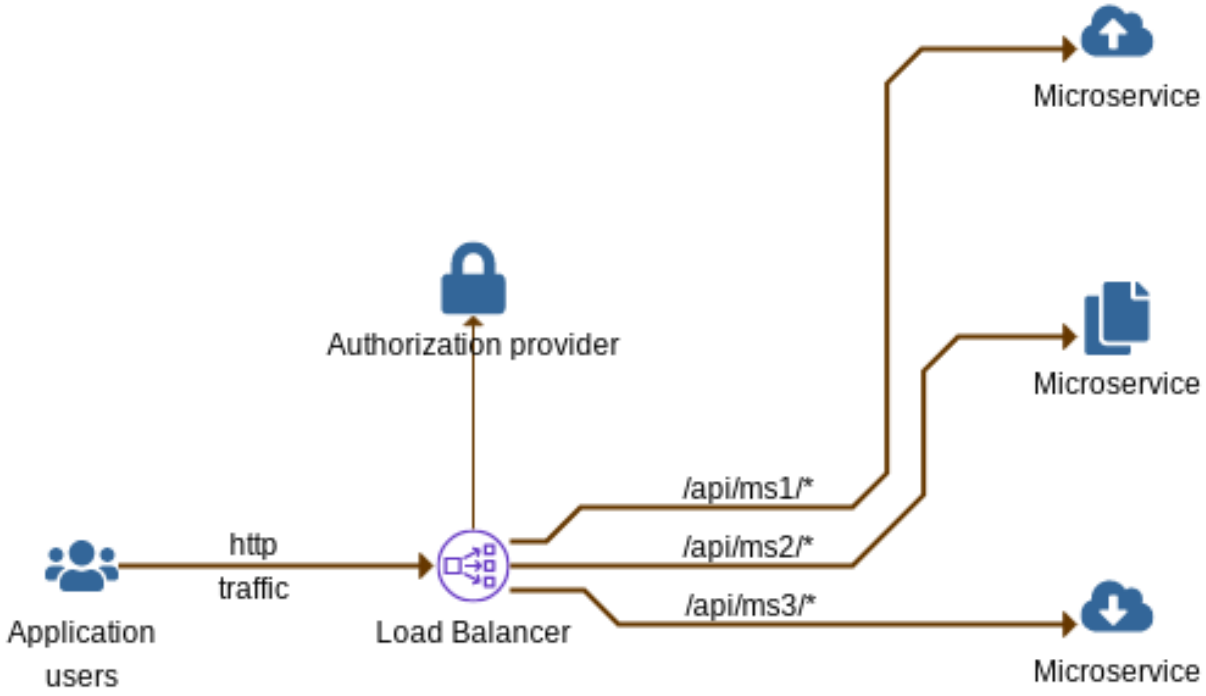
Authorization at microservices

- Each microservice is exposed at different URI path.
- Each microservice performs authorization on its own.



Authorization at ingress

- Each microservice is exposed at different URI path.
- The authorization is performed on ingress path and only there.
- Microservices can assume that all requests that they have received are already authorized.



Enter APISIX

- Support for multi-tenancy
- Cloud-Native
- Kubernetes integration
- An Apache project

Easy installation and configuration

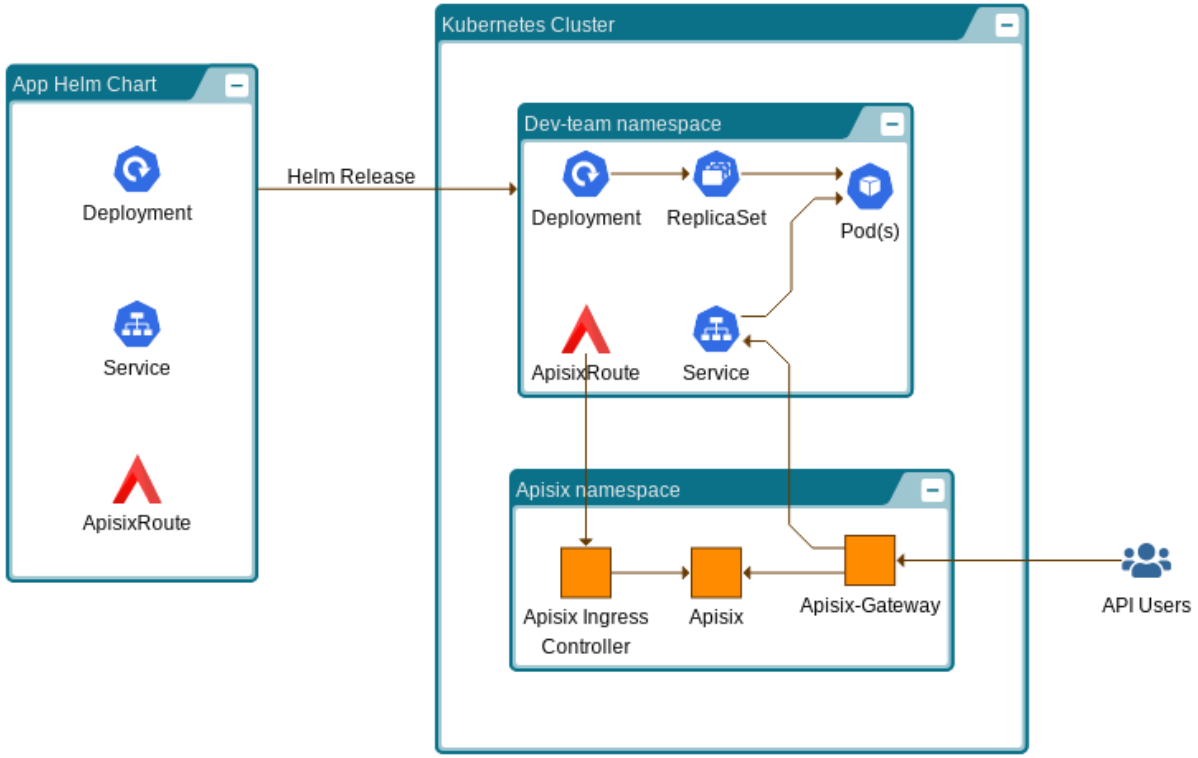
- Helm Chart allowed an easy installation.
- Dev teams were able to configure their endpoints using ApisixRoute CRDs.

Challenges

- Helm Charts.
- Etcd integration.
- Inter-team agreements on API use.

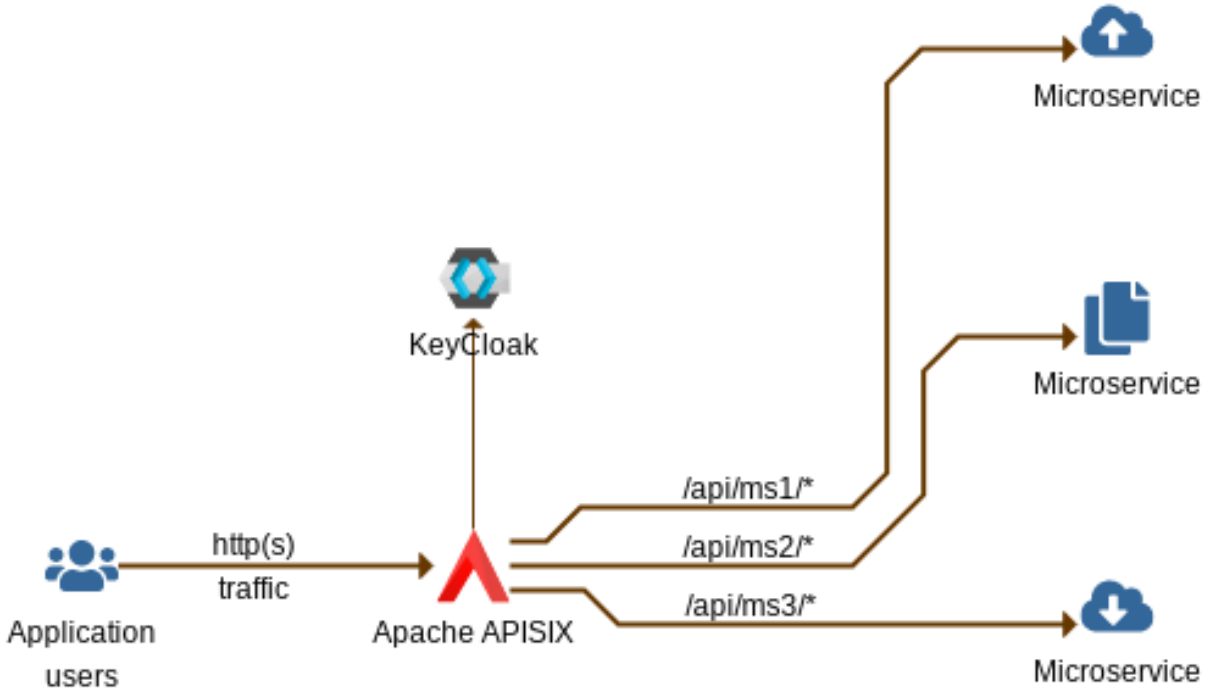
Deep dive

- Dev teams manage and use Helm Charts for their applications.
- Helm Charts include ApisixRoute object definitions.
- Apisix Route definitions are stored, versioned and deployed along with the application itself.
- Helm Release installs/updates objects in Kubernetes Cluster
- Apisix takes part of rest.



Outcome

- Each microservice is exposed at different URI path.
- The authorization is performed on ingress path and only there.
- Microservices can assume that all requests that they have received are already authorized.
- A multitude of other plugins already found a use for specific endpoints.



Deep dive cont.

E

example-helm-app

Project ID: 4518

Star 0 | Fork 0

1 Commit | 1 Branch | 0 Tags | 123 KB Project Storage

master | example-helm-app / | Find file | Web IDE | Clone

Example Helm app

Łukasz Biegaj authored just now

b0965003

Add README | Add LICENSE | Add CHANGELOG | Add CONTRIBUTING | Enable Auto DevOps

Add Kubernetes cluster | Set up CI/CD | Configure Integrations

Name	Last commit	Last update
.dockerignore	Example Helm app	just now
.gitignore	Example Helm app	just now
Dockerfile	Example Helm app	just now
package-lock.json	Example Helm app	just now
package.json	Example Helm app	just now
server.js	Example Helm app	just now
values.yaml	Example Helm app	just now

Deep dive cont.

Name	Last commit	Last update
.dockerignore	Example Helm app	just now
.gitignore	Example Helm app	just now
Dockerfile	Example Helm app	just now
package-lock.json	Example Helm app	just now
package.json	Example Helm app	just now
server.js	Example Helm app	just now
values.yaml	Example Helm app	just now

Dockerfile 188 bytes

Edit | Replace | Delete

```
FROM node:18

RUN mkdir /app && chown node:node /app
USER node
WORKDIR /app
EXPOSE 8080

COPY --chown=node:node package*.json /app/
RUN ls -la
RUN npm ci

COPY . /app/
CMD [ "server.js" ]
```

Deep dive cont.

Name	Last commit	Last update
.dockerignore	Example Helm app	just now
.gitignore	Example Helm app	just now
Dockerfile	Example Helm app	just now
package-lock.json	Example Helm app	just now
package.json	Example Helm app	just now
server.js	Example Helm app	just now
values.yaml	Example Helm app	just now

values.yaml 249 bytes

Edit Replace Delete

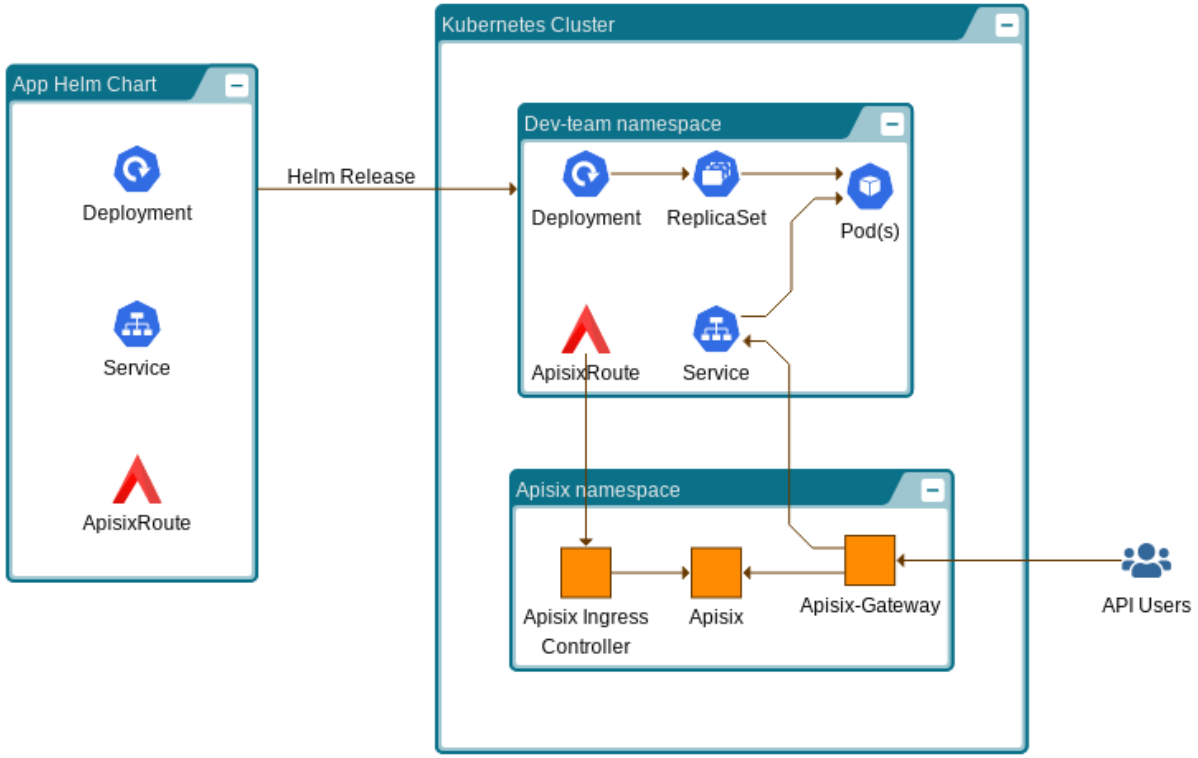
```yaml
replicaCount: 3

resources:
  limits:
    cpu: 1024m
    memory: 1024Mi
  requests:
    cpu: 512m
    memory: 1024Mi

apisixAuthorizedHttpRoutes:
  - name: ms1
    match:
      paths:
      - "/api/ms1/*"
      methods:
      - GET
      - DELETE
```

Deep dive

- Dev teams manage and use Helm Charts for their applications.
- Helm Charts include ApisixRoute object definitions.
- Apisix Route definitions are stored, versioned and deployed along with the application itself.
- Helm Release installs/updates objects in Kubernetes Cluster
- Apisix takes part of rest.



Key takeaways

- APISIX is easy to use
- APISIX greatly accelerates application development.
- We are already deploying APISIX in few more projects are considering to use it to fulfill every API Gateway need.

Questions?

Thank you!