Kong THE CLOUD NATIVE API PLATFORM

Kong Meetup

How Automation Changed the Game in API Management

Sven Walther
June 2023
sven@konghq.com
https://mastodon.cloud/@svenwal

The days before web-based APIs

In-app and in-memory based APIs, custom network protocols

Concepts

- Monolithic applications
- Function calls within the same application in memory or sockets on the same machine
- Exchange with external applications rare and in “crazy ways” (thing about floppy discs and FTP copied CSV files)

Developer

- 💚 Everything within the app
  -> no need to worry about extra security layers
- 🟥 Next to no agility
  -> slow and complex development

ESB and SOAP

Applications split into services / sub-components

Concepts

- Network used to have cross service communication
- SOAP as major communication protocol
- All security, routing and even transformation done by the ESB
- One big central carrier as single-point-of-failure

Developer

- 💛 Need to adopt SOAP and it’s complex security model
- 💚 Creating client software very easy thanks to WSDL
- 🟥 Services need to be registered in ESB
- 💛 ESB administration in charge of permissions and routing
- 🟥 Configuration changes by experts only - long processes and waiting time
- 🟥 A lot of business logic embedded

API Gateways

Offload the external communication to a more lightweight tool

Concepts

- API Gateway more lightweight and easier to use - in comparison to an ESB(!)
- User interface driven
- Automation capabilities for specific use cases - most notable propagation from one environment to another (dump & restore)

Developer

- 💚 Does not need to reinvent the wheel for authentication, logging, …
- 🟥 Services need to be registered in gateway
- 🟥 Gateway administration in charge of permissions and routing
- 🟥 Configuration changes by experts only - long processes and waiting time

API First / REST APIs

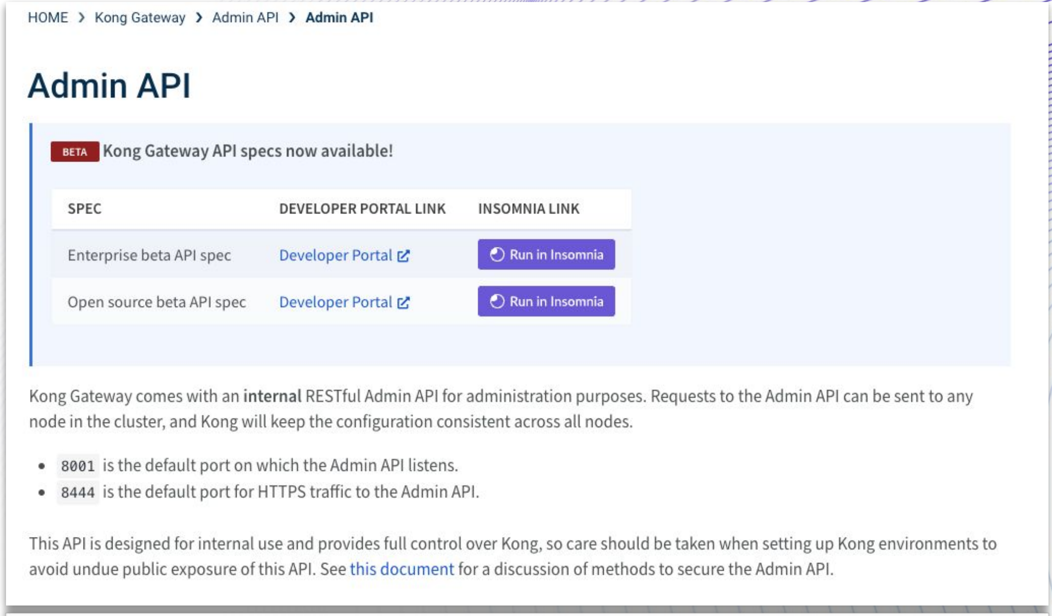
“Everything is an API” - the game changer for full automation

Concepts

- API Gateway very lightweight
- Every functionality automatable
- Designed to be automated

Developer

- 💚 Can automate whole configuration using scripts (curl, httpie, …)
- 💚 After pipeline is created no more extra work needed
- 💚 No need to go through a centralized API Gateway team
- 💚 **APIOps come true**
- 🟥 Needs to create / use shell scripts


HOME › Kong Gateway › Admin API › **Admin API**

Admin API

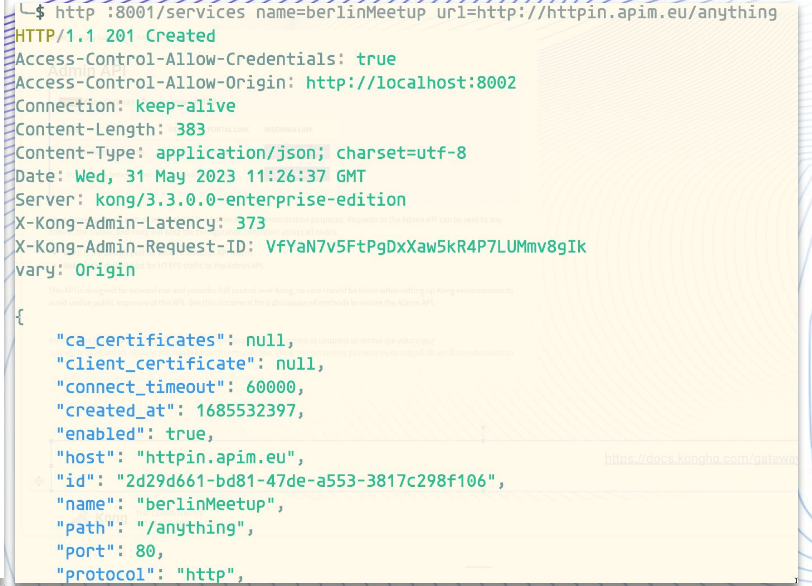
BETA Kong Gateway API specs now available!

SPEC	DEVELOPER PORTAL LINK	INSOMNIA LINK
Enterprise beta API spec	Developer Portal	Run in Insomnia
Open source beta API spec	Developer Portal	Run in Insomnia

Kong Gateway comes with an **internal** RESTful Admin API for administration purposes. Requests to the Admin API can be sent to any node in the cluster, and Kong will keep the configuration consistent across all nodes.

- `8001` is the default port on which the Admin API listens.
- `8444` is the default port for HTTPS traffic to the Admin API.

This API is designed for internal use and provides full control over Kong, so care should be taken when setting up Kong environments to avoid undue public exposure of this API. See this document for a discussion of methods to secure the Admin API.


```
└$ http :8001/services name=berlinMeetup url=http://httpin.apim.eu/anything
HTTP/1.1 201 Created
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://localhost:8002
Connection: keep-alive
Content-Length: 383
Content-Type: application/json; charset=utf-8
Date: Wed, 31 May 2023 11:26:37 GMT
Server: kong/3.3.0.0-enterprise-edition
X-Kong-Admin-Latency: 373
X-Kong-Admin-Request-ID: VfYaN7v5FtPgDxXaw5kR4P7LUMmv8gIk
vary: Origin

{
    "ca_certificates": null,
    "client_certificate": null,
    "connect_timeout": 60000,
    "created_at": 1685532397,
    "enabled": true,
    "host": "httpin.apim.eu",
    "id": "2d29d661-bd81-47de-a553-3817c298f106",
    "name": "berlinMeetup",
    "path": "/anything",
    "port": 80,
    "protocol": "http",
```

https://docs.konghq.com/gateway/latest/admin-api/

Declarative (decK)

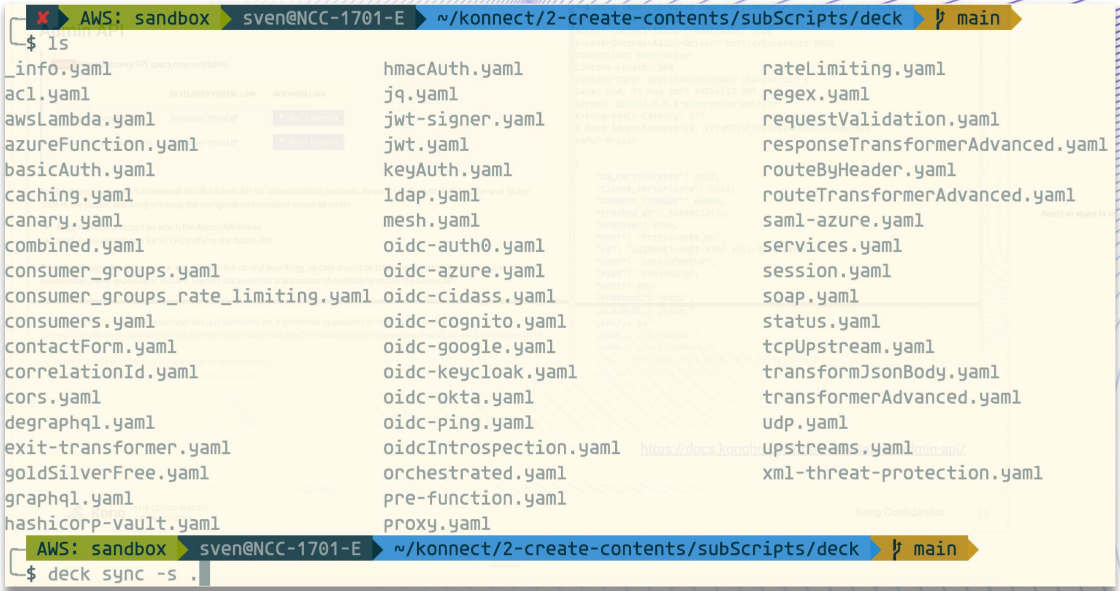
Becoming non-imperative removes complexity

Concepts

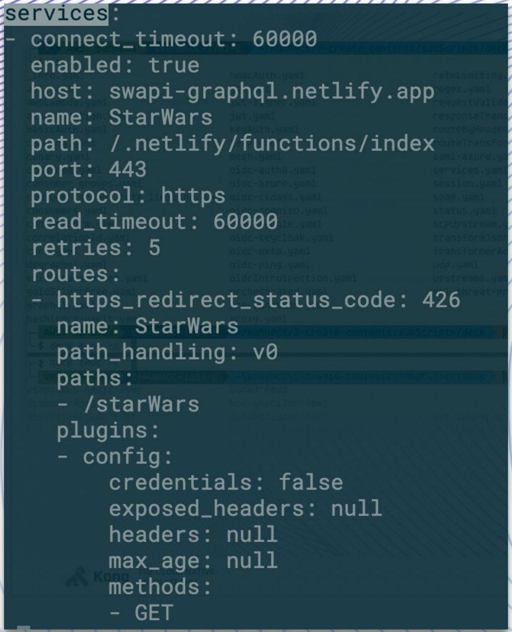
- Instead of telling the gateway exactly what you want in this second describe the desired state
- YAML files are very readable
- YAML files can easily be stored and diff’ed in a version control system

Developer

- 💚 Just needs to (auto-)generate a YAML file
- 💚 Publishing pipelines based on easy readable YAML files
- 💛 Needs to only know the structure and possible values in the YAML file



```
✘  AWS: sandbox  sven@NCC-1701-E  ~/konnect/2-create-contents/subScripts/deck  main
╰─$ ls
_info.yaml                        hmacAuth.yaml            rateLimiting.yaml
acl.yaml                          jq.yaml                  regex.yaml
awsLambda.yaml                    jwt-signer.yaml          requestValidation.yaml
azureFunction.yaml                jwt.yaml                 responseTransformerAdvanced.yaml
basicAuth.yaml                    keyAuth.yaml             routeByHeader.yaml
caching.yaml                      ldap.yaml                routeTransformerAdvanced.yaml
canary.yaml                       mesh.yaml                saml-azure.yaml
combined.yaml                     oidc-auth0.yaml          services.yaml
consumer_groups.yaml              oidc-azure.yaml          session.yaml
consumer_groups_rate_limiting.yaml oidc-cidass.yaml        soap.yaml
consumers.yaml                    oidc-cognito.yaml        status.yaml
contactForm.yaml                  oidc-google.yaml         tcpUpstream.yaml
correlationId.yaml                oidc-keycloak.yaml       transformJsonBody.yaml
cors.yaml                         oidc-okta.yaml           transformerAdvanced.yaml
degraphql.yaml                    oidc-ping.yaml           udp.yaml
exit-transformer.yaml             oidcIntrospection.yaml   upstreams.yaml
goldSilverFree.yaml               orchestrated.yaml        xml-threat-protection.yaml
graphql.yaml                      pre-function.yaml
hashicorp-vault.yaml              proxy.yaml
 AWS: sandbox  sven@NCC-1701-E  ~/konnect/2-create-contents/subScripts/deck  main
╰─$ deck sync -s .
```



```
services:
- connect_timeout: 60000
  enabled: true
  host: swapi-graphql.netlify.app
  name: StarWars
  path: /.netlify/functions/index
  port: 443
  protocol: https
  read_timeout: 60000
  retries: 5
  routes:
  - https_redirect_status_code: 426
    name: StarWars
    path_handling: v0
    paths:
    - /starWars
    plugins:
    - config:
        credentials: false
        exposed_headers: null
        headers: null
        max_age: null
        methods:
        - GET
```

https://docs.konghq.com/deck/latest/

Kubernetes Ingress Controller

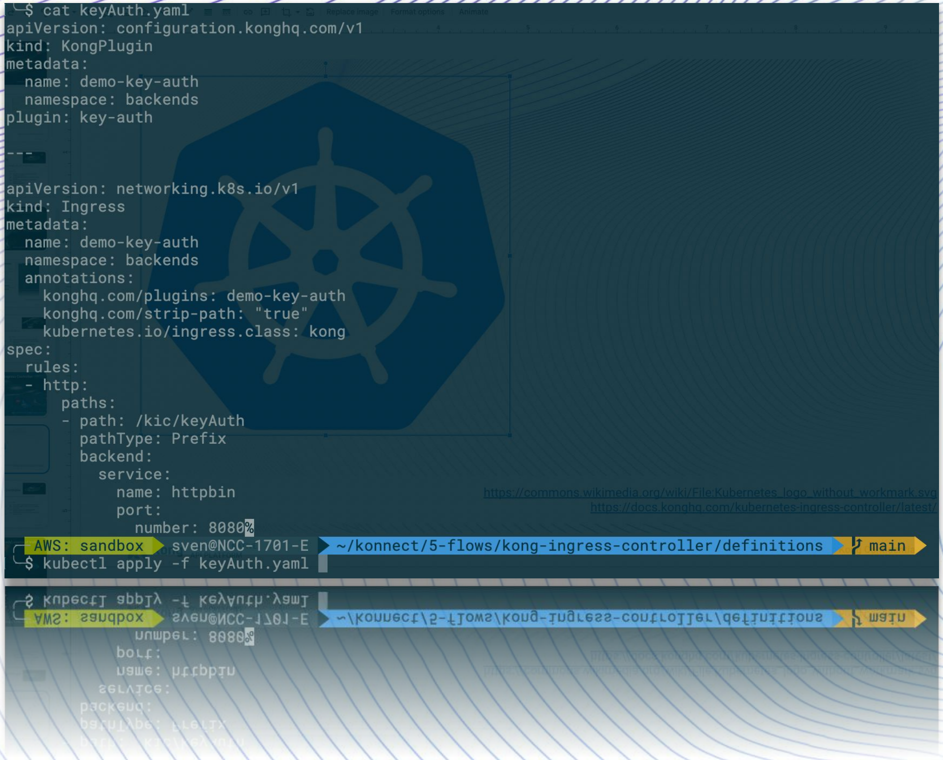
Follow the de-facto standard of modern software deployment

Concepts

- Instead of telling the gateway exactly what you want in this second describe the desired state
- YAML files are very readable
- YAML files can easily be stored and diff'ed in a version control system
- Ingress ressources are standardized and well known

Developer

- 💚 Just needs to (auto-)generate a YAML file
- 💚 Publishing pipelines based on easy readable YAML files
- 💚 Needs to only know ~~the structure and~~ possible values in the YAML file
- 💛 Only available when Kubernetes or Openshift are being used



```
└$ cat keyAuth.yaml
apiVersion: configuration.konghq.com/v1
kind: KongPlugin
metadata:
  name: demo-key-auth
  namespace: backends
plugin: key-auth

---

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: demo-key-auth
  namespace: backends
  annotations:
    konghq.com/plugins: demo-key-auth
    konghq.com/strip-path: "true"
    kubernetes.io/ingress.class: kong
spec:
  rules:
  - http:
      paths:
      - path: /kic/keyAuth
        pathType: Prefix
        backend:
          service:
            name: httpbin
            port:
              number: 8080%
┌ AWS: sandbox  sven@NCC-1701-E  ~/konnect/5-flows/kong-ingress-controller/definitions  main
└$ kubectl apply -f keyAuth.yaml
```

https://commons.wikimedia.org/wiki/File:Kubernetes_logo_without_workmark.svg
https://docs.konghq.com/kubernetes-ingress-controller/latest/

OpenAPI as source of truth

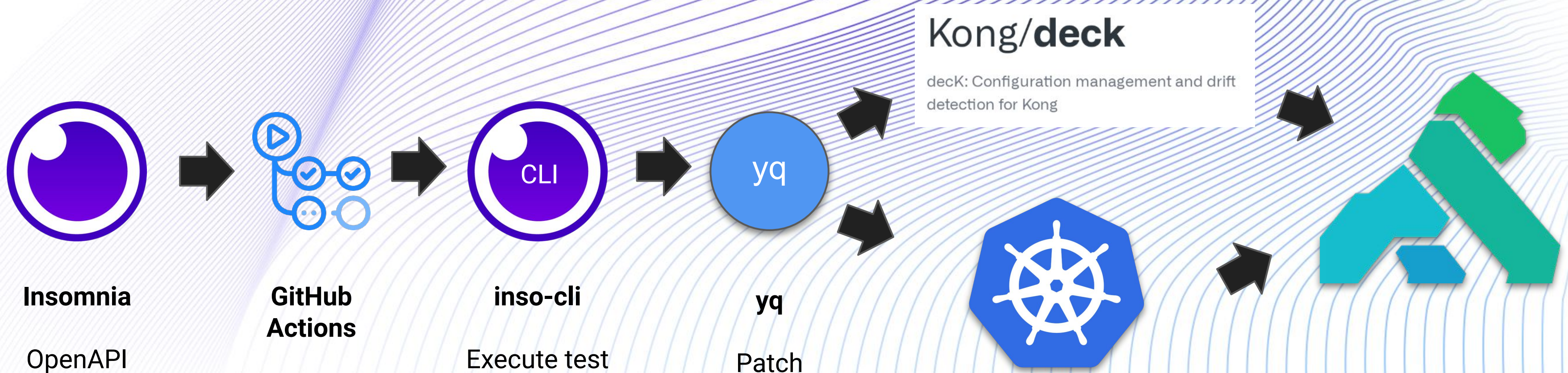
Re-use already being created documentation

Concepts

- OpenAPI (Swagger) has won the documentation wars and is the standard to describe (REST) APIs
- In a documentation first approach an API is only created when OpenAPI already has been designed
- Alternative are auto-generated OpenAPI specs from the code

Developer

- 💚 Can use a file format being used every day anyway
- 💚 In best case does not even need to know that an API Gateway even exists and what to do about it
- 💚 Automation based on existing assets
- 💚 Documentation and implementation are in sync by design
- 💛 A lot of moving parts / tools



Kong/**deck**

decK: Configuration management and drift detection for Kong

Insomnia

OpenAPI Design

Test Case generation

GitHub Actions

CI/CD Pipeline

inso-cli

Execute test cases

Generate declarative configuration

yq

Patch generated YAML

Update URLs

Include plugins

…

https://insomnia.rest/
https://github.com/features/actions
https://github.com/mikefarah/yq

OpenAPI - Titan level

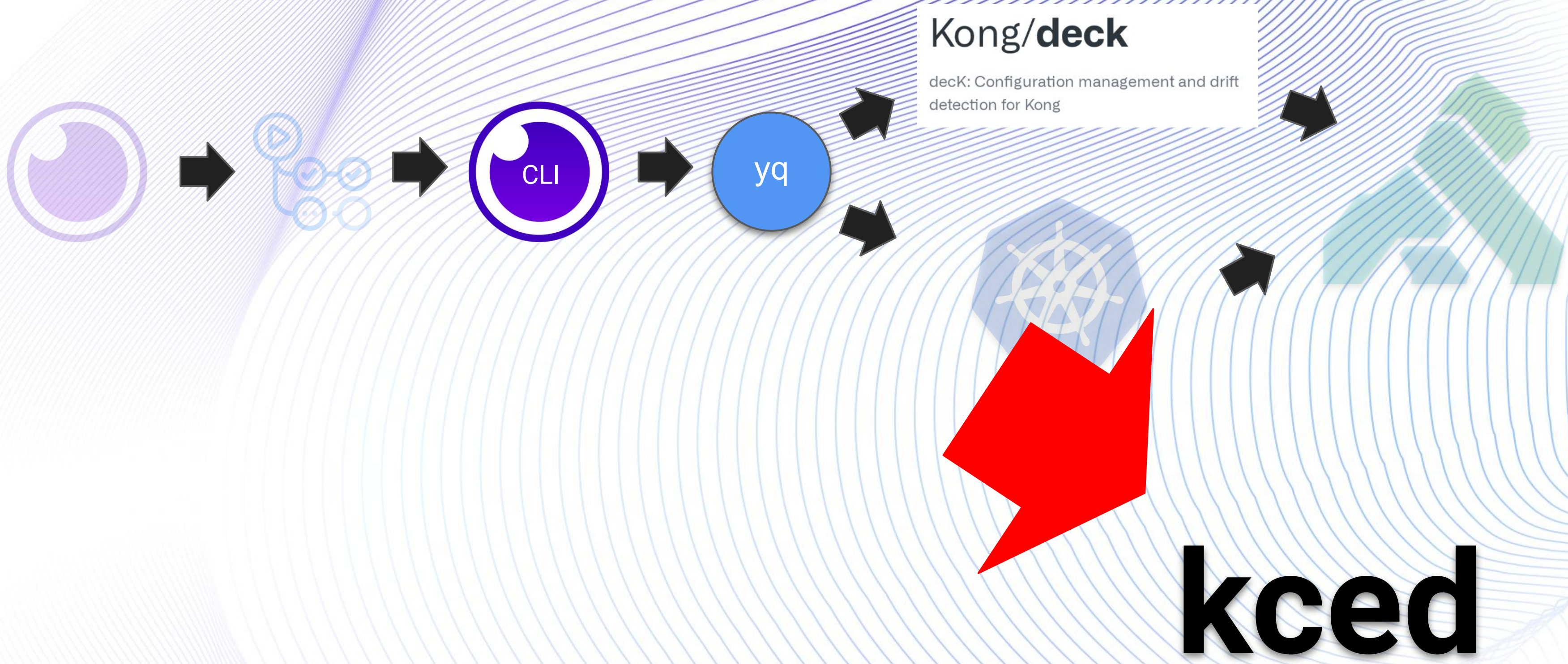
Heading off for the next stage

Concepts

- OpenAPI (Swagger) has won the documentation wars and is the standard to describe (REST) APIs
- In a documentation first approach an API is only created when OpenAPI already has been designed
- Alternative are auto-generated OpenAPI specs from the code

Developer

- 💚 Can use a file format being used every day anyway
- 💚 In best case does not even need to know that an API Gateway even exists and what to do about it
- 💚 Automation based existing assets
- 💚 Documentation and implementation are in sync by design
- 💚💚**Revamped focused tooling set**



https://github.com/Kong/go-apiops

Thank You

GitHub link to CI/CD examples:
https://github.com/svenwal/kong-meetup-berlin-june-2023