Cloud-Native API Management
on Kubernetes

June 25, 2020

Hello!

Chathura Kulasinghe_

Lead Solutions Engineer

chathurak@wso2.com

Shehani Rathnayake_

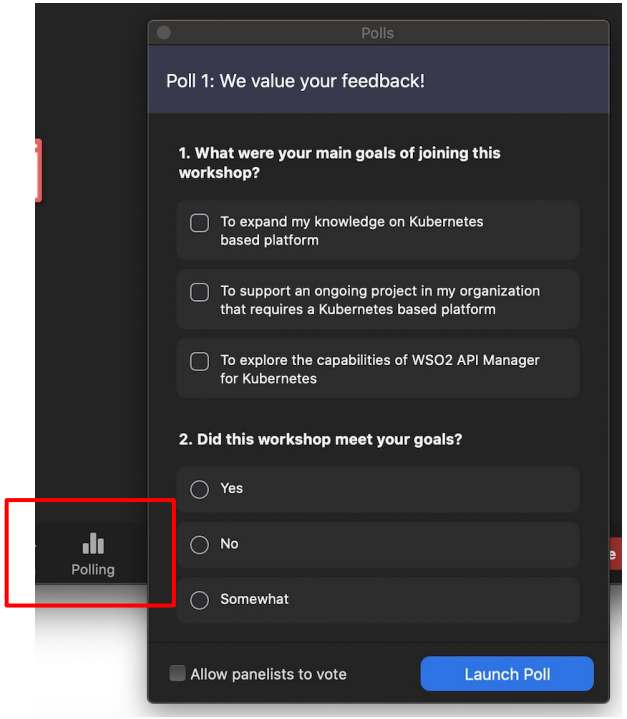
Software Engineer

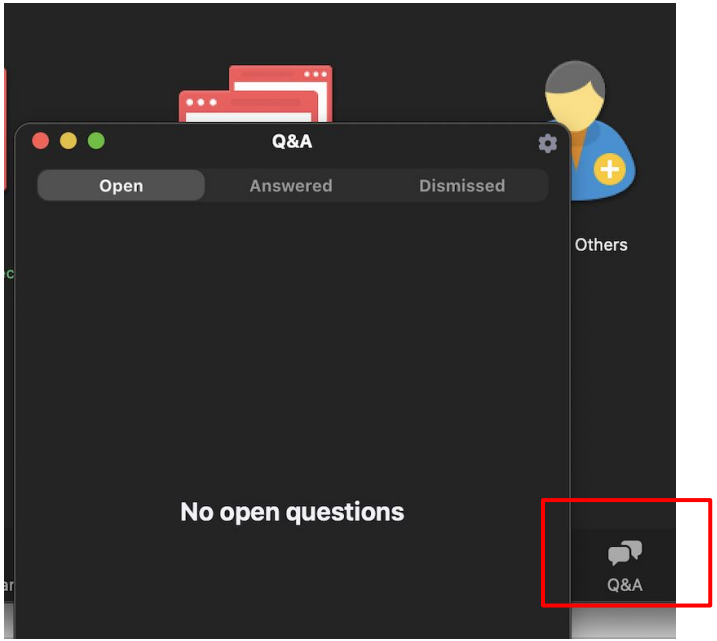
shehanir@wso2.com

Overview

- Development of Integration Services (Part 1)
  - Use WSO2 Integration Studio, Test, Debug
- Build Docker images of Integration Services (Part 2)
  - Use WSO2 Integration Studio to Build & Push images to Dockerhub
- Deployment of Integration Services on Kubernetes (Part 3)
  - Pull image from the Dockerhub and spin up a container
- Creating Managed APIs fronting backend services (Part 4)
  - Use `k8s-api-operator` and `apictl` to spin up micro-gateway containers on k8s
- CI/CID considerations (Part 5)
  - Register environments on `apictl` and automate import/export APIs

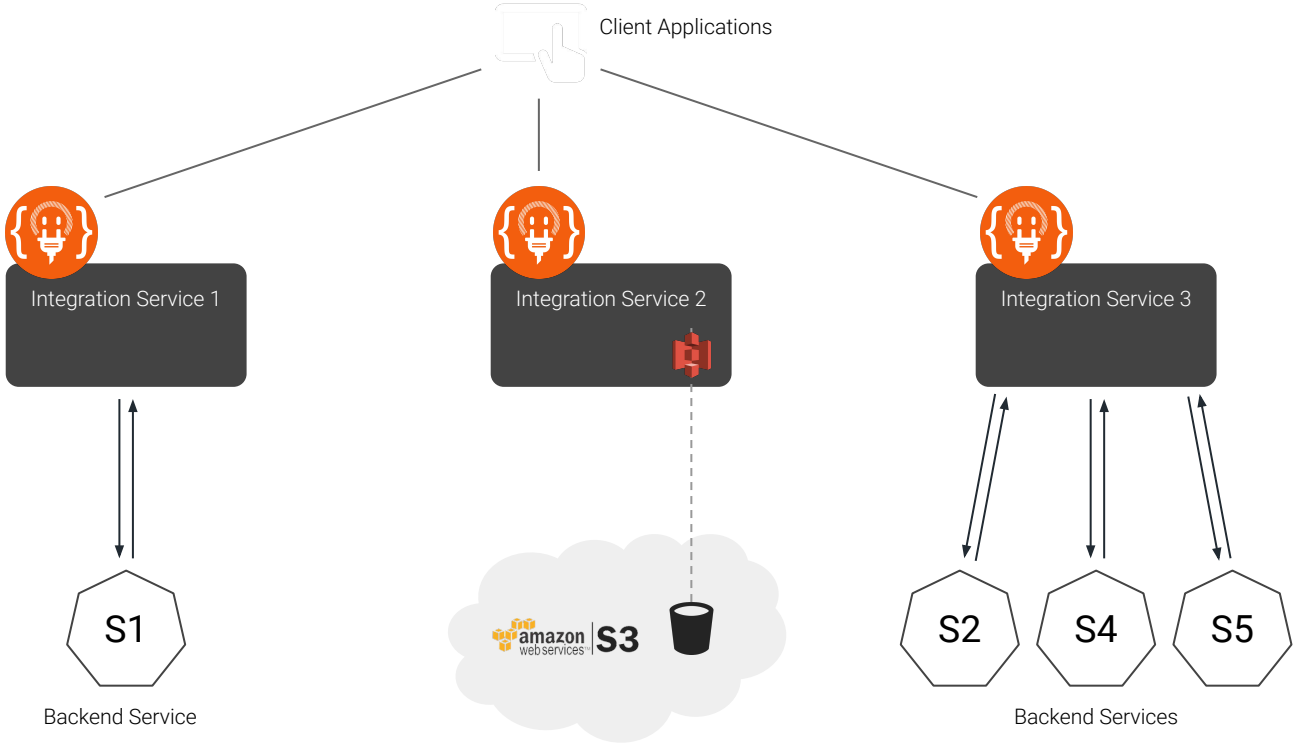
Feedback / Q&A


Polls
Poll 1: We value your feedback!
1. What were your main goals of joining this workshop?
To expand my knowledge on Kubernetes based platform
To support an ongoing project in my organization that requires a Kubernetes based platform
To explore the capabilities of WSO2 API Manager for Kubernetes
2. Did this workshop meet your goals?
Yes
No
Somewhat
Allow panelists to vote
Launch Poll
Polling

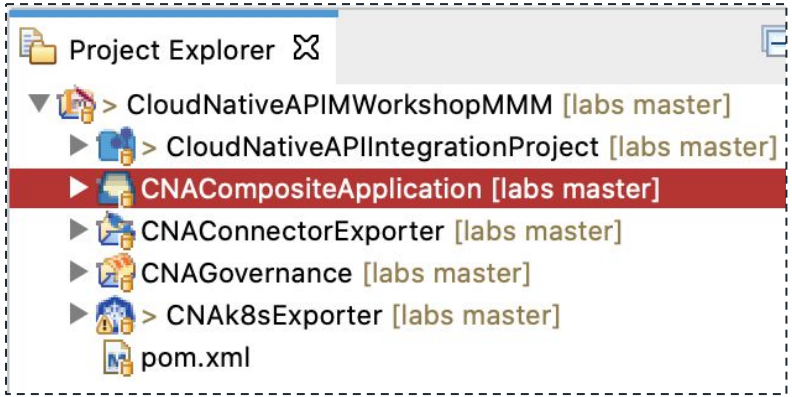

Q&A
Open
Answered
Dismissed
No open questions
Others
Q&A

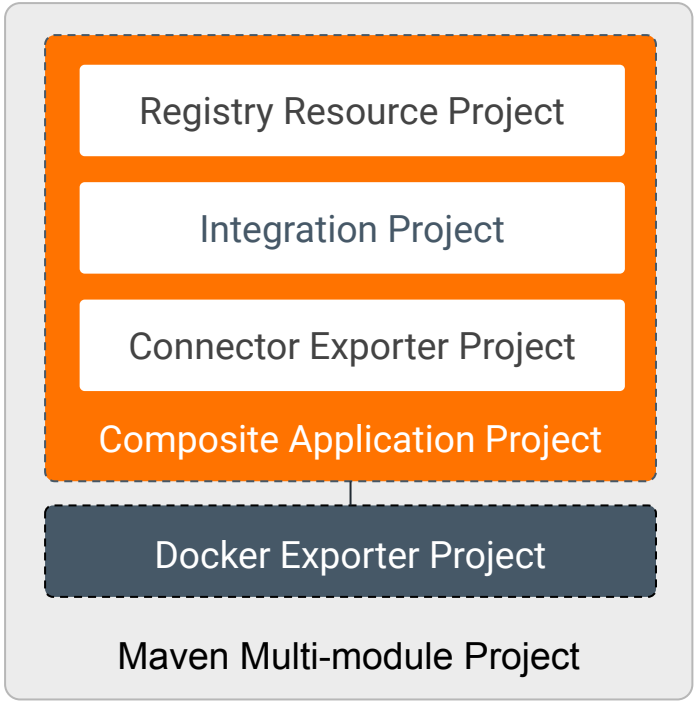
Part 1: Development of Integratrion Services

Demo Cases


Client Applications
Integration Service 1
Integration Service 2
Integration Service 3
S1
amazon web services S3
S2
S4
S5
Backend Service
Backend Services

WSO2 Workspace & Projects structure



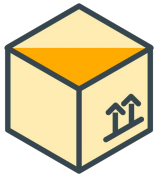


Part 2: Buld & Push a Docker Image of Integration Artefacts

The Development Process

Integration Studio IDE
model integration artefacts visually or with code editor

Composite Archives (*.car)



Synapse is an XML based declarative language that helps modeling Integration programme logic.

This is similar to how HTML tags and attributes are used to declare/define visual elements and the acitivies on web pages.

Composite Archive is something similar to a ZIP archive which packages WSO2 integration artefacts.

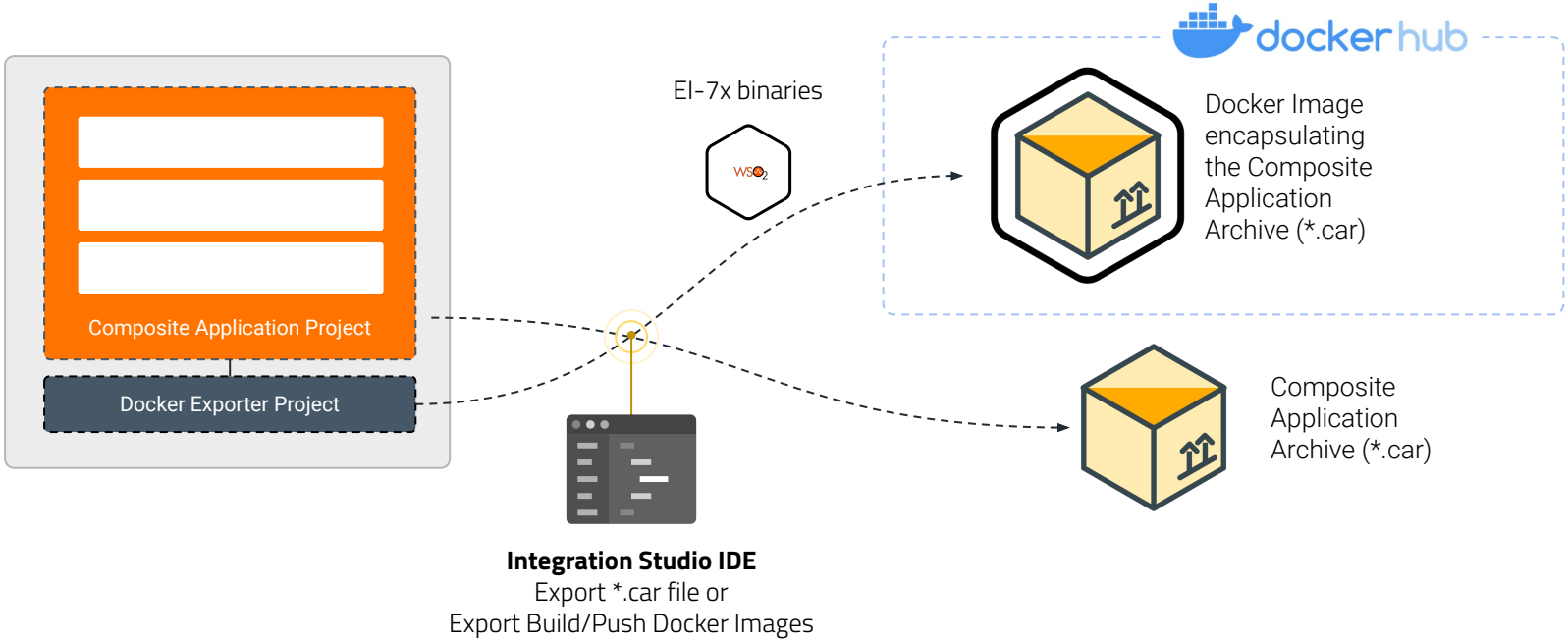
This can either be created by directly exporting from the IDE or by combining with typical build pipeline utilities/frameworks such as Jenkins, Maven and GitLab.

Project files
(xml, text)

</ Repository >

Maven, Jenkins

Export Docker Images

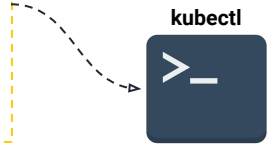


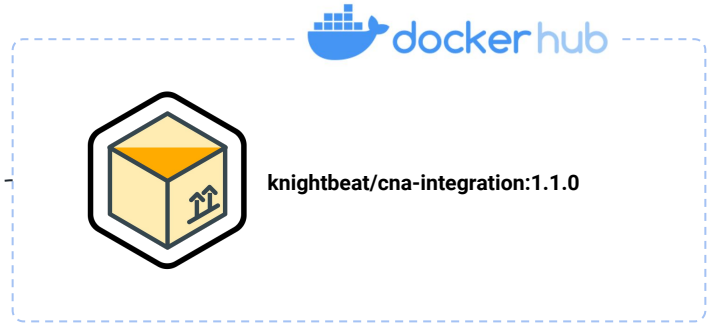
Part 3: Deploy the Integration services into a kubernetes cluster

Pull docker images and Create Deployment

integration_cr.yaml

```yaml
---
apiVersion: "integration.wso2.com/v1alpha1"
kind: "Integration"
metadata:
  name: "cna-integration"
spec:
  replicas: 3
  image: "knightbeat/cna-integration:1.1.0"
```





k8s-ei-operator

Client Applications

cna-integration-service

cna-integration-deployment

Part 4: Creating a managed API in kubernetes using the CLI

“Operators” in Kubernetes

Operators are:

- Software Extensions to Kubernetes
- that uses the Custom Resources
- for Packaging, Deploying and Managing applications.

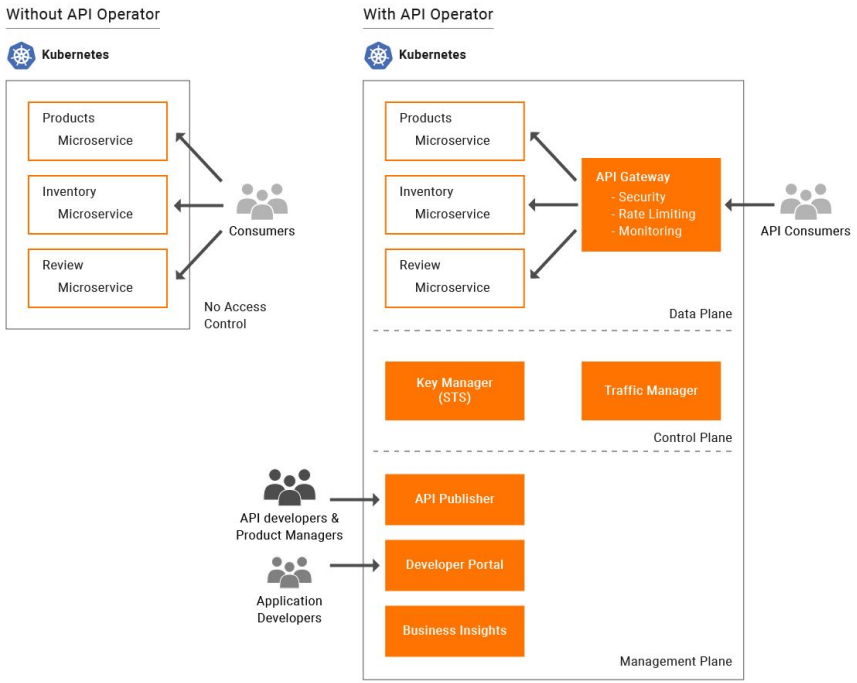
When they are used:

- You hide the Deployment complexities
- You don’t need domain-specific knowledge on application management

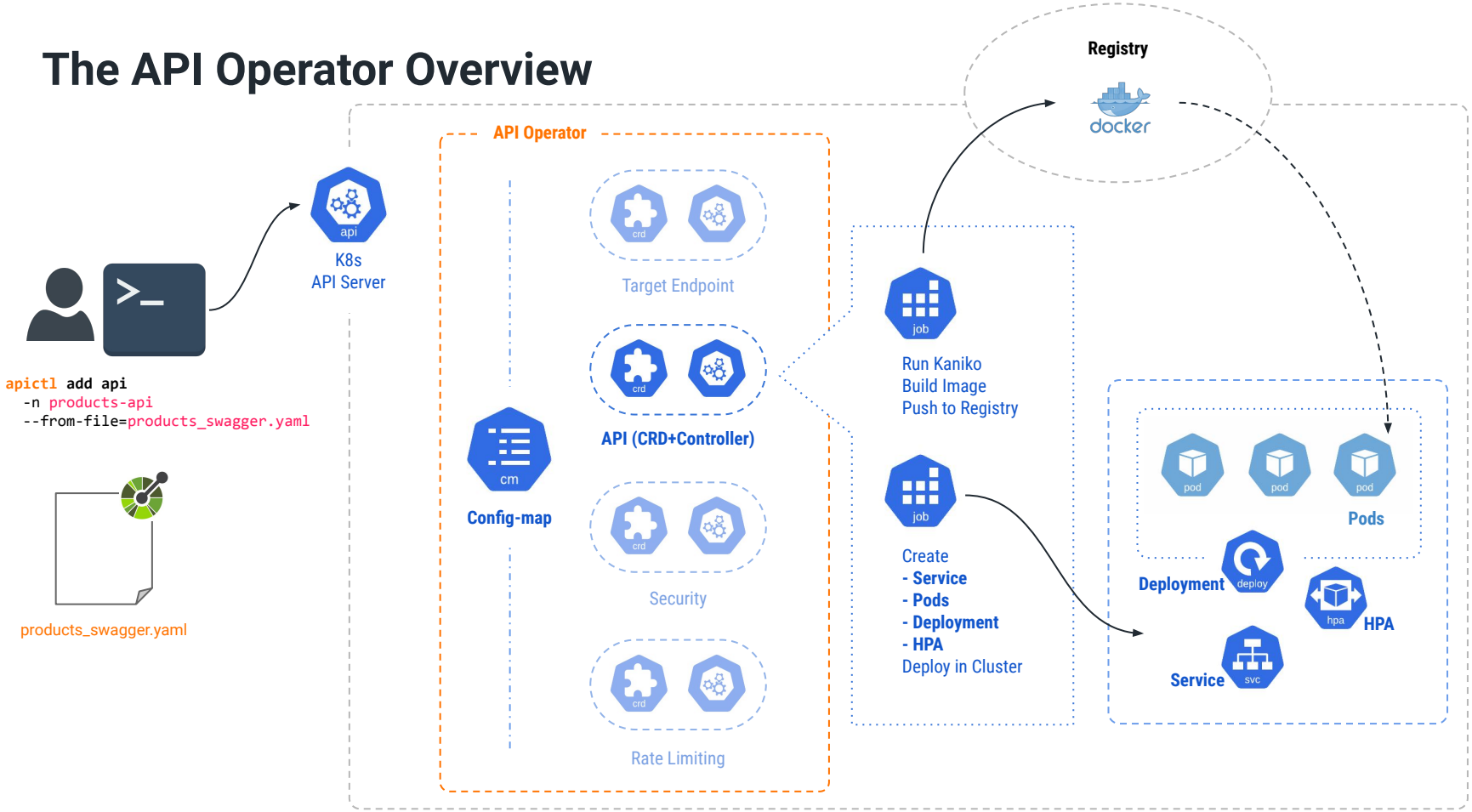
List of Kubernetes operators: https://operatorhub.io/

The “API Operator” for Kubernetes

1. Makes APIs a first-class citizen in the Kubernetes ecosystem
2. Provides fully automated experience for cloud-native API management
3. OpenAPI definition (Swagger) as one single source of truth



The API Operator Overview

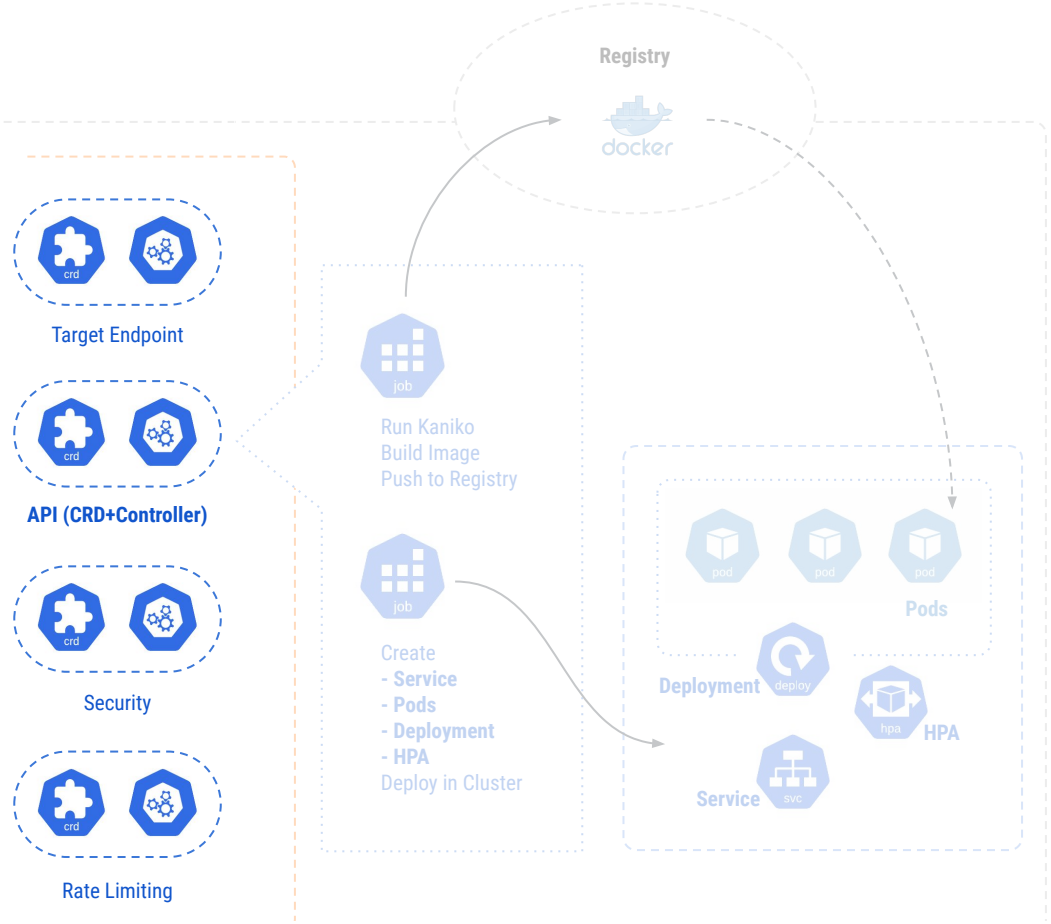



```
apictl add api
  -n products-api
  --from-file=products_swagger.yaml
```

Custom Resources

1. API
2. Security
3. Rate Limiting
4. Target Endpoint

These **C**ustom **R**esource **D**efinitions are created with its Controller by the API Operator



1. **API**

Custom Resource: API

3. Rate Limiting
4. Target Endpoint

```yaml
apiVersion: wso2.com/v1alpha1
kind: API
metadata:
 name: "product-api"
spec:
 definition:
   configmapName: "product-api-swagger"
   type: swagger
 replicas: 1
 mode: privateJet
```

1. API

Custom Resource: Security

3. Rate Limiting
4. Target Endpoint

```yaml
apiVersion: wso2.com/v1alpha1
kind: Security
metadata:
 name: petstorejwt
spec:
 # Security - JWT
 type: JWT
 issuer: https://wso2apim:32001/oauth2/token
 audience: http://org.wso2.apimgt/gateway
 # Create secret with certificate and add
secret name
 certificate: wso2am-secret
```

OPENAPI Defintion

```yaml
security:
    - petstorejwt: []
```

1. API

Custom Resource: Rate-Limiting

3. **Rate Limiting**

4. Target Endpoint

```yaml
apiVersion: wso2.com/v1alpha1
kind: RateLimiting
metadata:
 name: fourreqpolicy
 namespace: app1-ns
spec:
 type: application
 description: Allow 4 requests per minute
# optional
 timeUnit: min
 unitTime: 1
 requestCount:
   limit: 4
```

OPENAPI Defintion

```yaml
X-wso2-throttling-tier:
                fourreqpolicy
```

1. API
2. Security
3. Rate Limiting
4. **Target Endpoint**

Custom Resource: Target Endpoint

```yaml
apiVersion: wso2.com/v1alpha1
kind: TargetEndpoint
metadata:
 name: products-service
 labels:
   app: app2
spec:
 protocol: http
 port: 9090
 deploy:
   name: products-service
   dockerImage: pubudu/products:1.0.0
   count: 1
 mode : sidecar
```

OPENAPI Defintion

```yaml
x-wso2-production-endpoints:
            products-service
x-wso2-mode:
            sidecar
```

The Demo Setup

```
GET http://10.110.86.65:80/products
```

Microservice deployment (products)

Pod

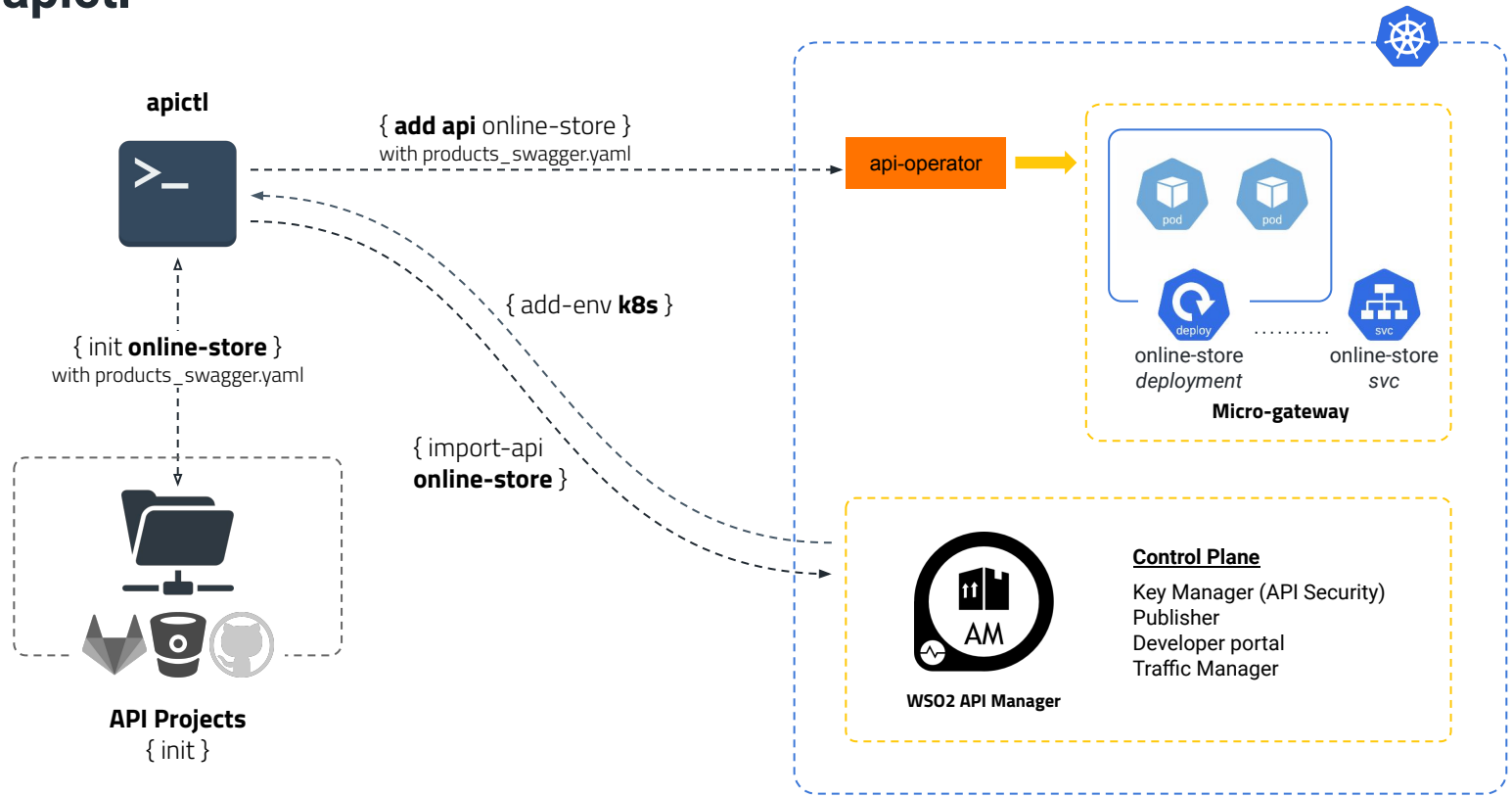
API Micro-Gateway deployment (online-store)

Pod

```
[CK» kubectl get pods
NAME                                    READY   STATUS              RESTARTS
online-store-88ffc7b56-zbfnk            0/1     ContainerCreating   0
products-deployment-649d9c8d67-qrzmj    1/1     Running             0
[CK» kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
online-store-88ffc7b56-zbfnk            1/1     Running   0          50s
products-deployment-649d9c8d67-qrzmj    1/1     Running   0          28m
[CK» kubectl get services
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)
kubernetes      ClusterIP      10.96.0.1        <none>           443/TCP
online-store    LoadBalancer   10.107.199.119   10.107.199.119   9095:31722/T
products        LoadBalancer   10.110.86.65     10.110.86.65     80:30766/TCP
```

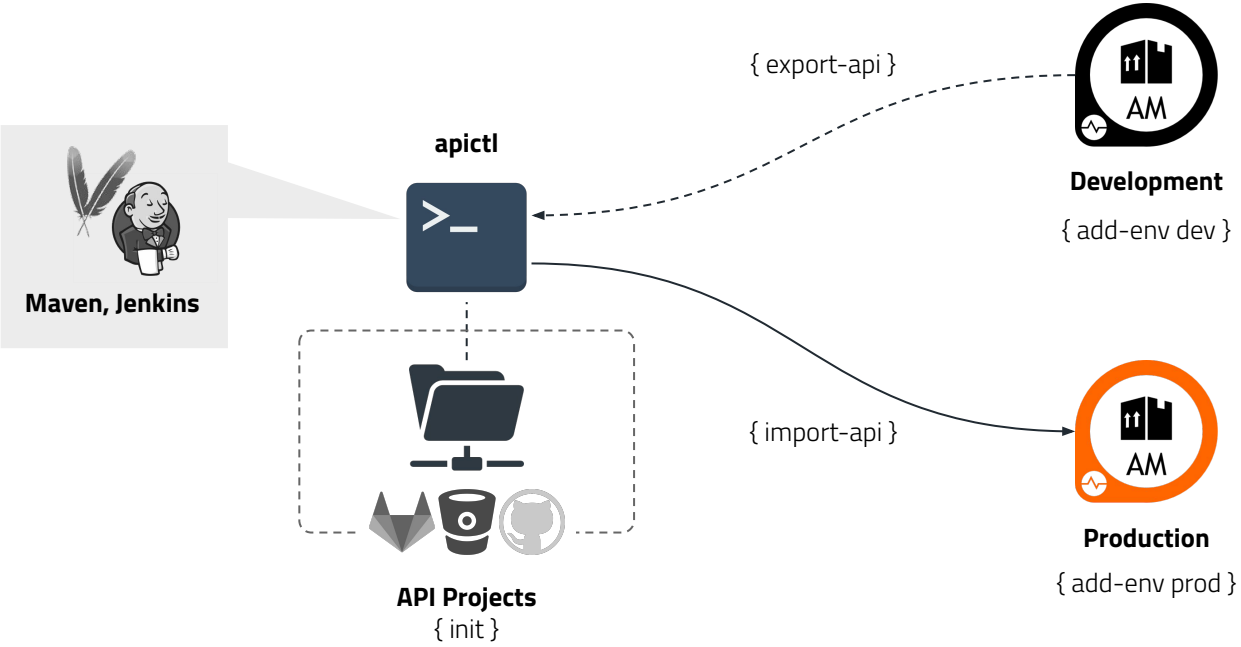
```
GET https://10.107.199.119:9095/store/v1.0.0/products
Authorization: Bearer {{TOKEN}}
```

`apictl`


apictl
{ add api online-store }
with products_swagger.yaml
api-operator
pod
pod
deploy
svc
online-store
deployment
online-store
svc
Micro-gateway
{ add-env k8s }
{ init online-store }
with products_swagger.yaml
{ import-api
online-store }
AM
WSO2 API Manager
Control Plane
Key Manager (API Security)
Publisher
Developer portal
Traffic Manager
API Projects
{ init }

Part 5: CI/CD pipeline considerations / planning

`apictl`


Maven, Jenkins
apictl
{ export-api }
AM
Development
{ add-env dev }
{ import-api }
AM
Production
{ add-env prod }
API Projects
{ init }

WSO2 Recommendations

- Use WSO2 official docker images (patched and production hardened) to build on top of it for packaging your binaries
- Maintain a private image registry
- Avoid NakedPods in production (not bound to a ReplicaSet or Deployment)
- Use officially released helm charts for production deployments
- Centralized log monitoring using a tool like ELK

Question Time!

GitHub Repository:

https://tinyurl.com/y83a72q8

WSO2 Slack Channels:

(EI) https://tinyurl.com/yclx5hnz

(API-M) https://tinyurl.com/y7x7zxu2

Thanks!

wso2.com