

# KEA

(Kubernetes Empowerer to API)

Requirements and analysis model

# Product description

The product is a platform for deploying, managing, and scaling machine learning models in production. It offers a secure, flexible environment for automating ML tasks like model versioning, routing, and monitoring. With Kubernetes integration and containerization support, it's designed for developers, ML engineers, and enterprises needing scalable, reliable ML infrastructure.

**Team K8C:** Tsurkan Daniel; Dandamaev Gadji; Tsaturyan Konstantin; Smolkin Mikhail

**Project repo:** <https://github.com/fanglores/Advanced-Software-Design>

**This report:** [https://github.com/fanglores/Advanced-Software-Design  
/blob/master/Practice%20Tasks/Final\\_Task/K8C\\_FinalTask1\\_\(Task8\).pdf](https://github.com/fanglores/Advanced-Software-Design/blob/master/Practice%20Tasks/Final_Task/K8C_FinalTask1_(Task8).pdf)

# Roles

## ML Engineer

**Description:** This role joins professionals involved in the development, deployment, and monitoring of ML models. They want to simplify the deployment process, automate API documentation, and ensure efficient request validation and caching, ultimately enhancing their workflow and model performance.

## API Consumer

**Description:** This role includes all users interacting with APIs to integrate ML models into their applications. They want to access reliable and well-documented APIs, enabling seamless integration of ML models into their business applications and ensuring optimal performance and usability.

# Personas

## ML Engineer (Maria, 32 years old)

### Goals:

- Deploy and version ML models in Kubernetes.
- Automatic deploy and further documentation.
- Flexibility for different ML frameworks.

### Pain points:

- Manual API documentation.
- Difficulties in monitoring model performance.

## Backend Developer (Alexander, 28 years old)

### Goals:

- Integration of the API gateway into existing infrastructure.
- Use automatic OpenAPI schema generation.
- Manage access rules.
- Scalable and secure deployment of ML models.

### Pain points:

- Incomplete or outdated API documentation.
- Challenges with integrating authorization and managing access control.
- Challenges with integration and corporate standards.

# Story map

**Security Specialist**  
Manage system's access parameters and perform audit

**DevOps Engineer**  
Efficient infrastructure management and traffic optimization

**Developer, ML Engineer**  
Automate API documentation updates and ensure API compliance

**API Consumer**  
Get access to ML-services API via ad-hoc and automated tools

Threat Response and Investigation

Access management

Maintain Network Operation

Reduce workload related to managing infrastructure

Reduce workload related to consumer support

Publish ML-model for using via API

API discovery and usage

Collect logs and events

Save logs and events for analysis

Implement Single Sign-On (SSO) for unified authentication

Traffic management

Automate infrastructure scaling and load balancing

Documentation updates

Enwrap model with web-app

Deploy model

Ensure API schemas are compliant and updated

Access service HTTP API

Granting role-based access

Ensuring high performance

Seamless model update

Audit (Traffic Logging)

Single Sign-On

Request Routing

Load Balancing

OpenAPI Scheme Generation

Modular Deployment of Models

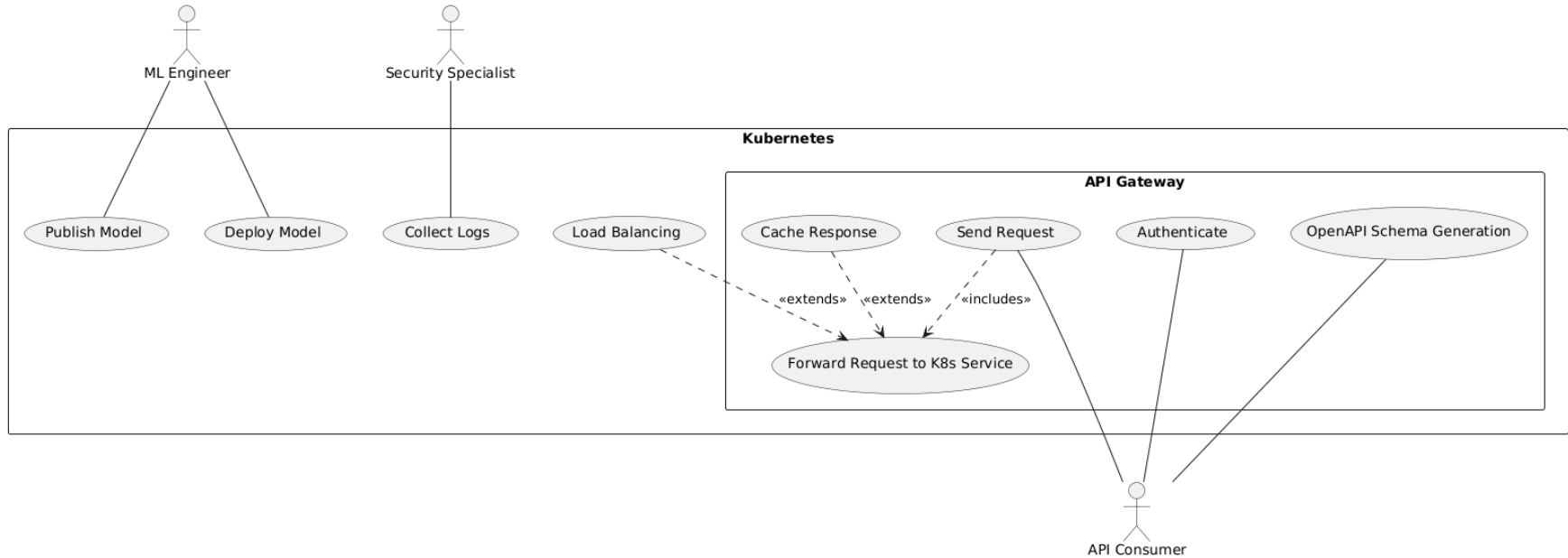
Service Deployment

Request Validation

Response Caching

Containerization

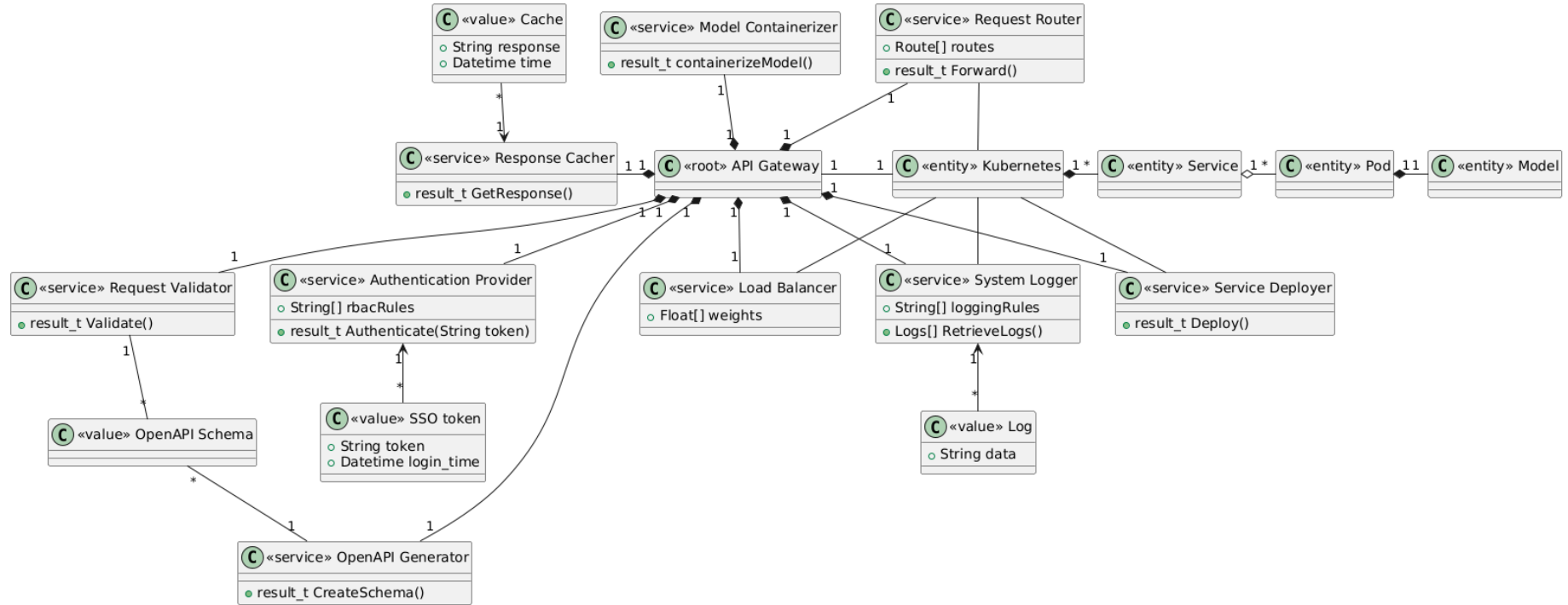
# Use case diagram



# Interaction analysis

| Use Case                  | Cooperation Name        | Used Roles                                | Candidate Classes                             |
|---------------------------|-------------------------|---|---|
| Forward Request           | Route Request           | Router, Request, Receiver                 | Request Router, Request, K8s                  |
| Load Balancing            | Distribute Load         | Load balancer, weights, ML services       | Load Balancer, K8s, Pod, ML Service           |
| Authenticate              | Validate Authentication | Request, Authenticator, SSO Keys          | Authentication Provider, SSO Key, Request     |
| Cache Response            | Cache Responses         | Request, Response, Cache, Cache validator | Response Cacher, Cache, Request, Response     |
| Collect Logs              | Log System Events       | Logger, Log, ML service, API Gateway      | System Logger, Log, ML Service, API Gateway   |
| Deploy Model              | Deploy Service          | Deployer, K8s, Pod, ML service            | Service Deployer, Pod, K8s, ML Service        |
| Publish Model             | Containerize Model      | Containerizer, Pod, K8s, ML model         | Model Containerizer, Pod, K8s, ML Service     |
| OpenAPI Schema Generation | Generate API Definition | ML service, OpenAPI schema, Generator     | OpenAPI Generator, OpenAPI Schema, ML Service |

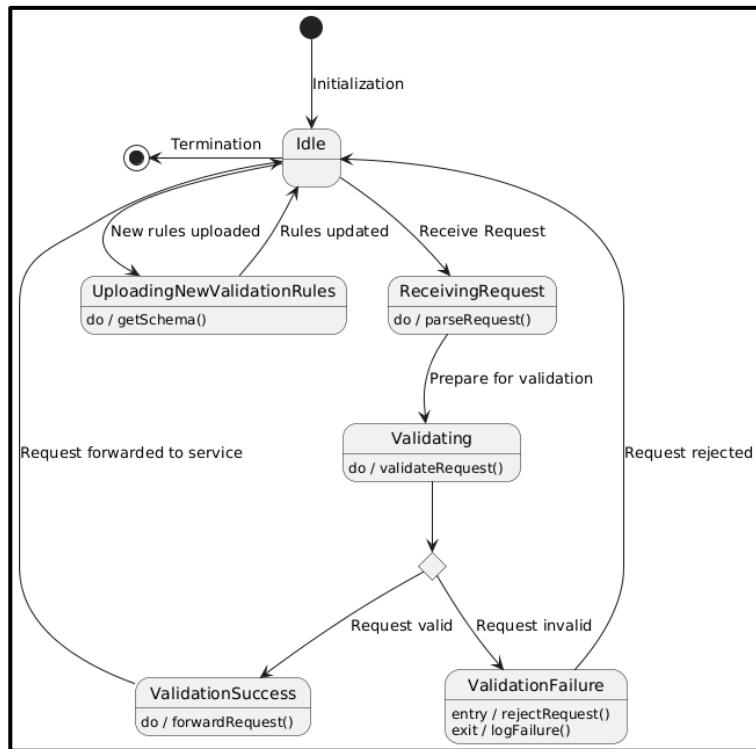
# Final class diagram



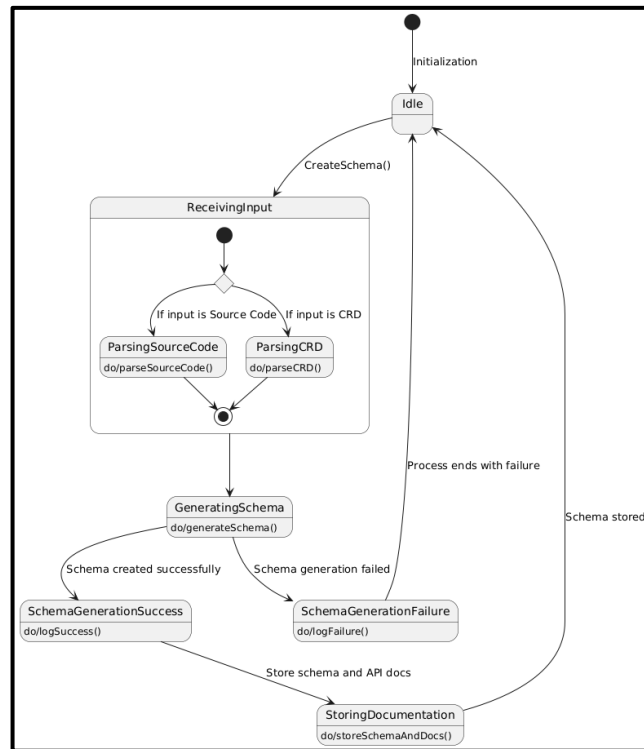


# Detailed behaviour

## Request Validator

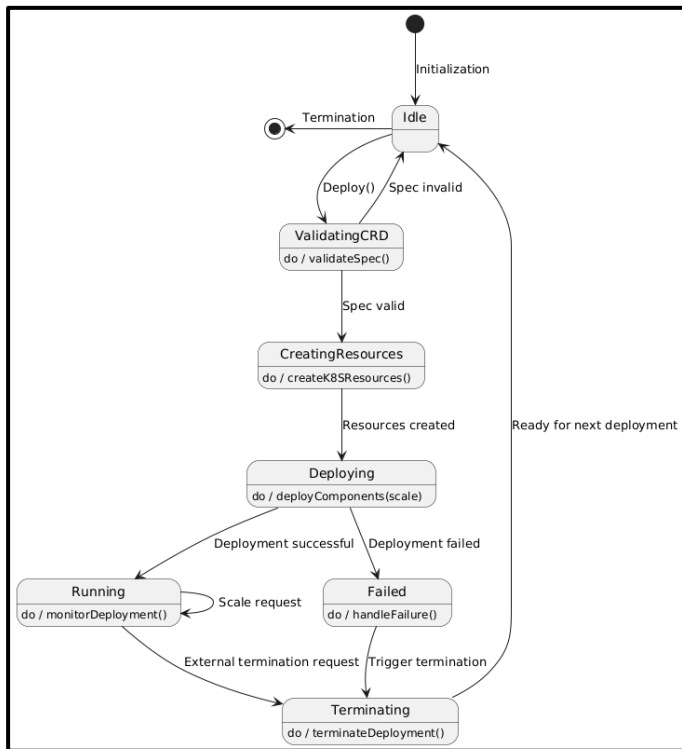


## OpenAPI Schema Generator

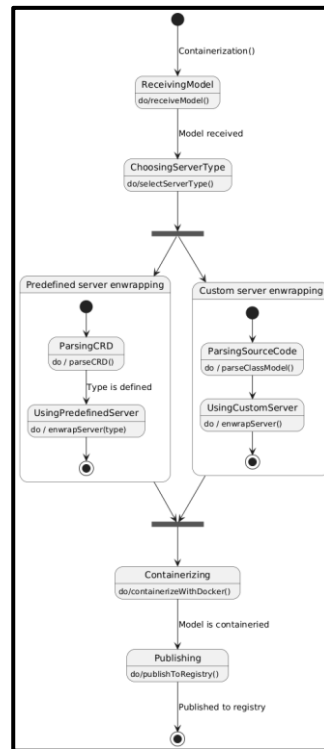


# Detailed behaviour

## Service Deployer



## Model Containerizer



# Repository structure

## Repository structure

```
.
├── product_img.jpg
├── README.md
├── General/
│   ├── Domain_description_en.md
│   ├── Task_description_en.md
│   ├── DFD0/
│   ├── StoryMap/
│   └── UseCases/
└── Practice Tasks/
    ├── Final_Task/
    ├── Task_1/
    ├── Task_2/
    ├── Task_3/
    ├── Task_4/
    ├── Task_5/
    ├── Task_6/
    └── Task_7/
```

## Tools Used:

- Github
- Drawio
- Planttext

# Team and roles



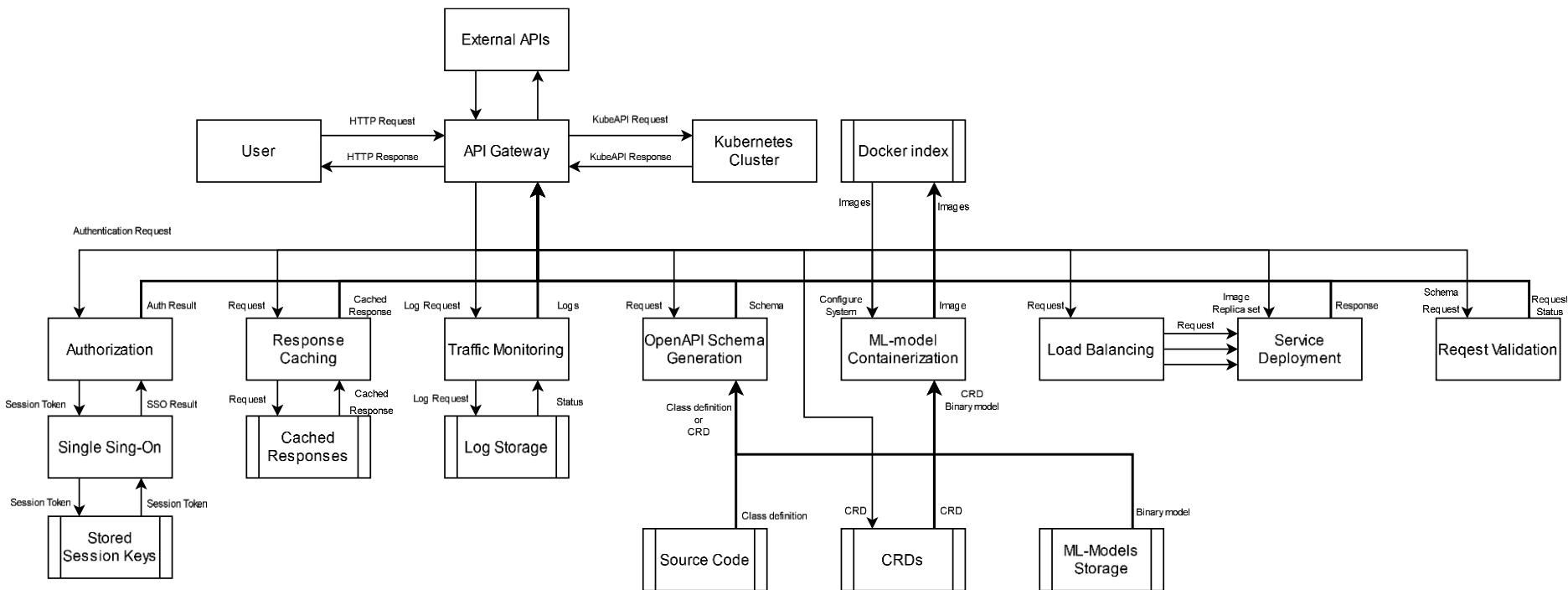
|   |                                     |  |  |
|---|-------------------------------------|--|--|
| Roles, Personas,<br>Story Map, CRC<br>Cards | Use cases, Interactions<br>Analysis | DFD, Classes Diagram,<br>Repository Management | Domain analysis,<br>Story Map,<br>Behavior model |
| Tsurkan Daniel<br>Tg: @crazy_deyzi          | Dandamaev Gadji<br>Tg: @dandamaev   | Tsaturyan Konstantin<br>Tg: @fanglores         | Smolkin Mikhail<br>Tg: @m0hnatik                 |

**Thanks for attention!**

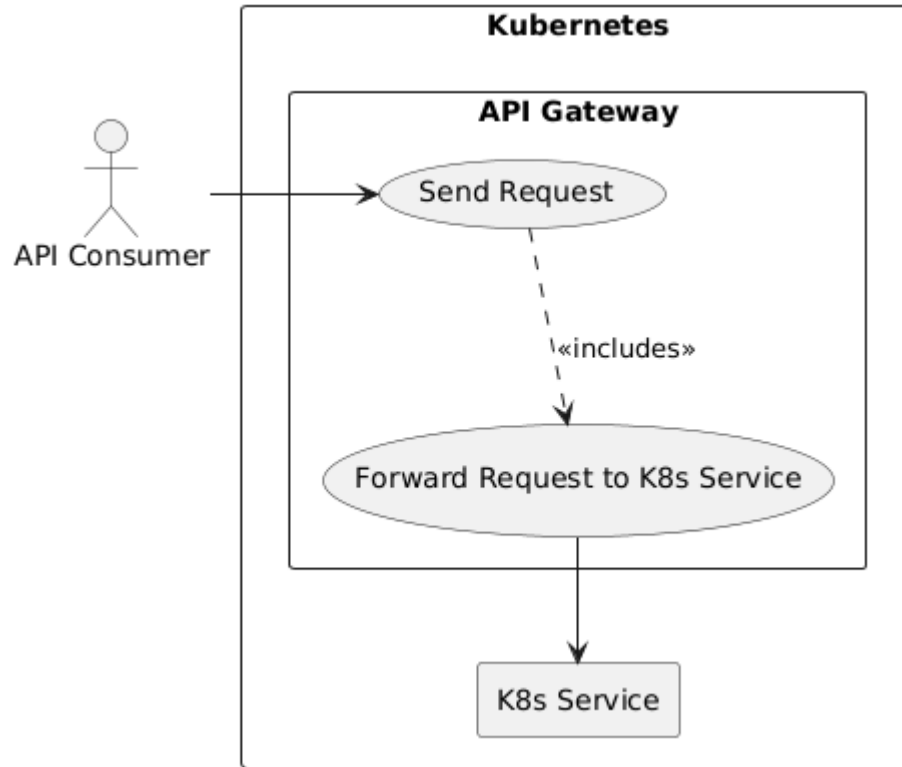
Now we are ready to answer your questions!

# **Extra slides**

# DFD (Level 0)

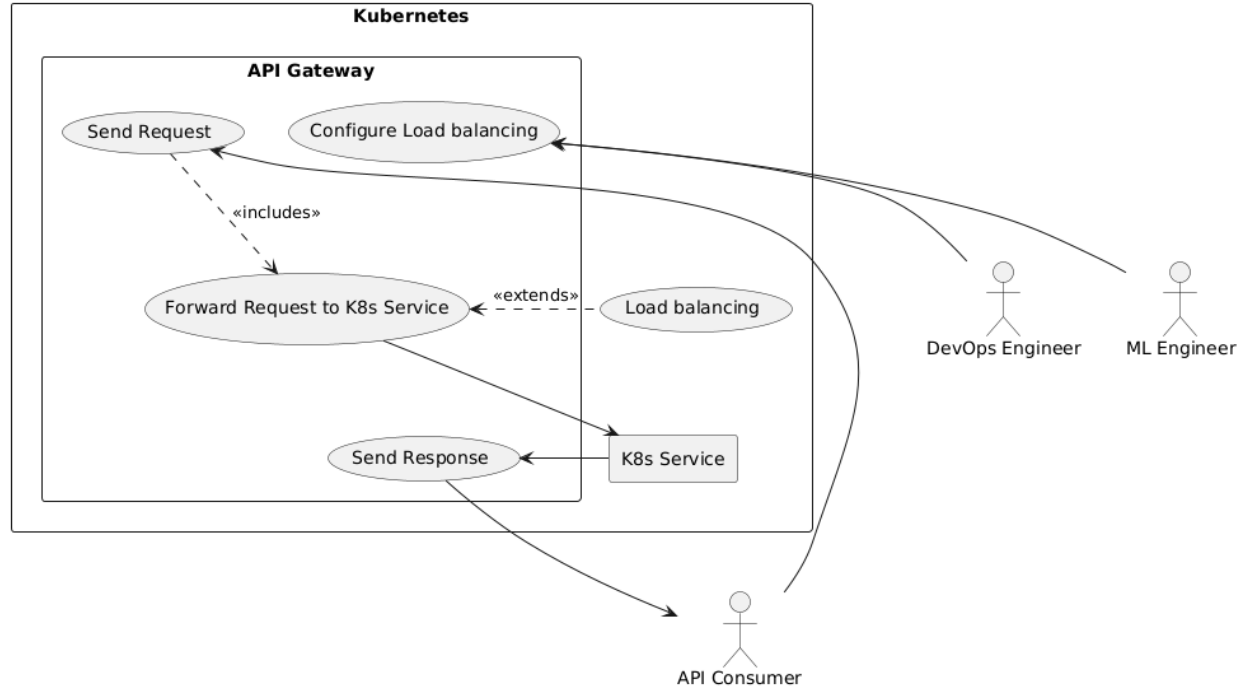


# Use Cases: Requests routing

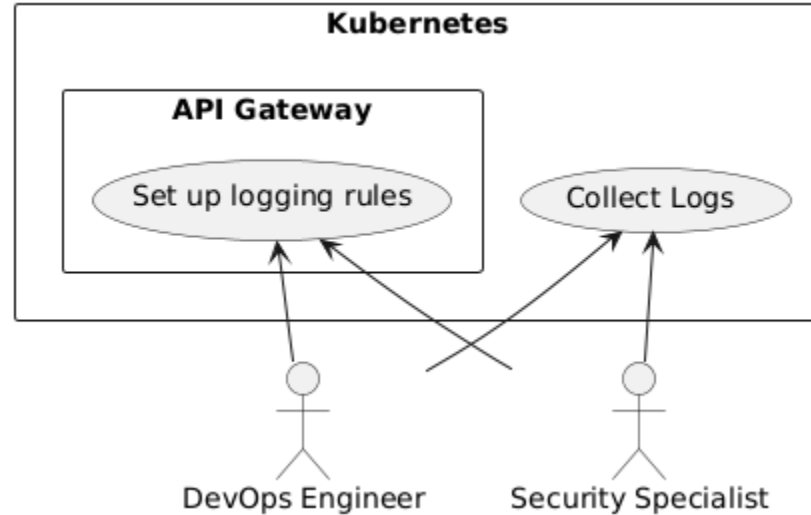




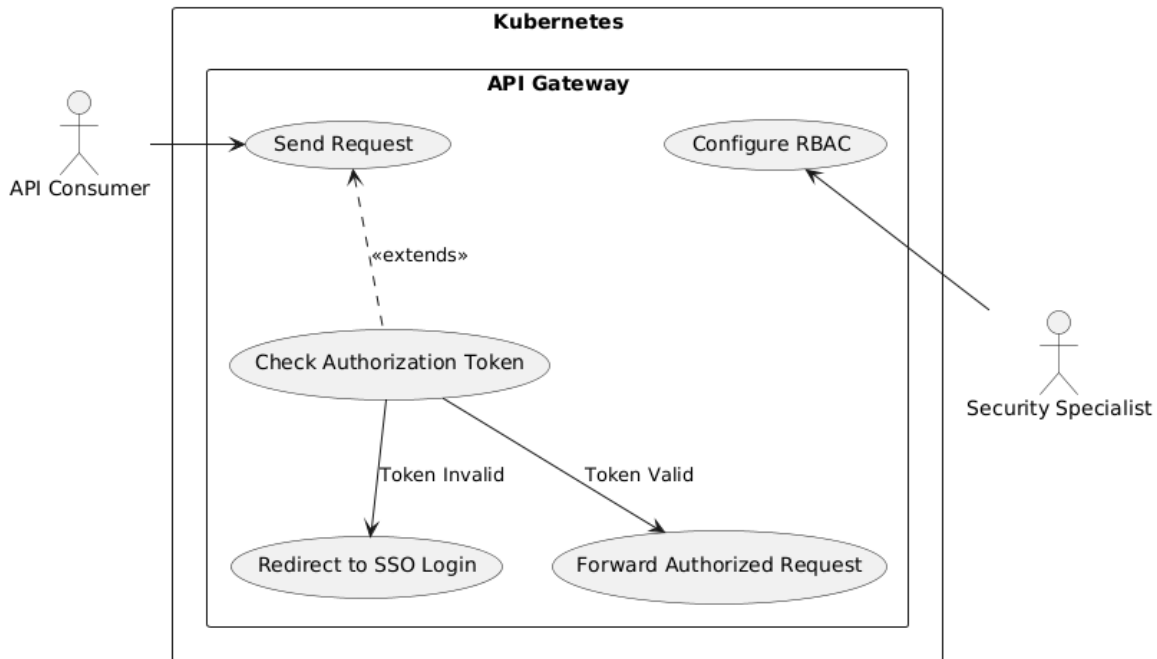
# Use Cases: Load balancing



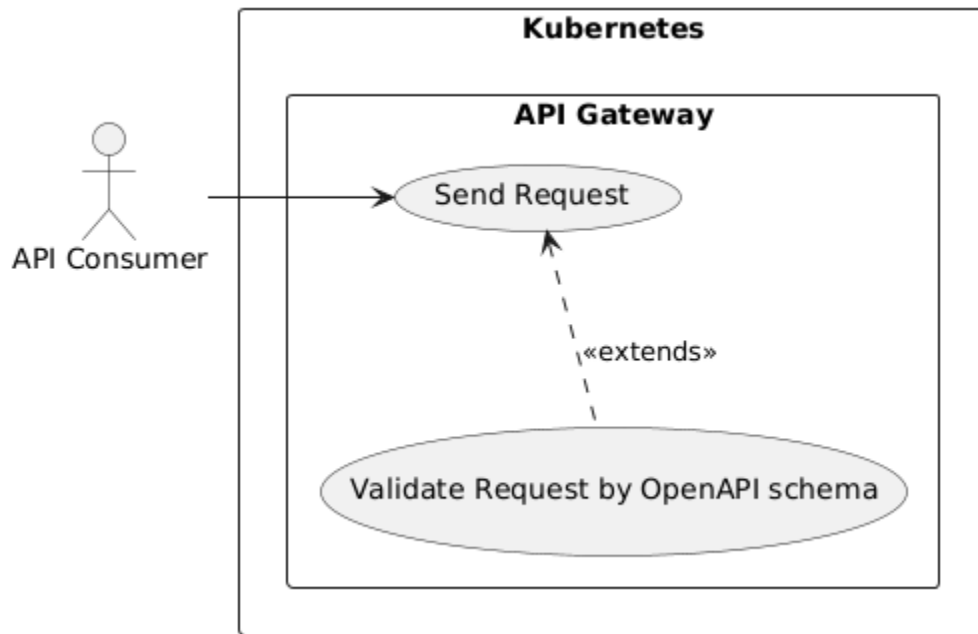
# Use Cases: Audit and Logging



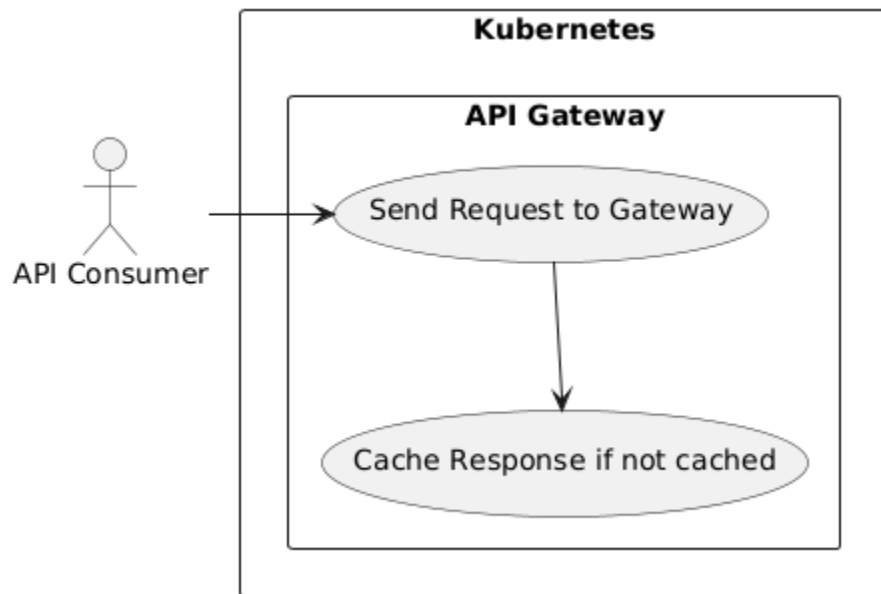
# Use Cases: Authorization (SSO)



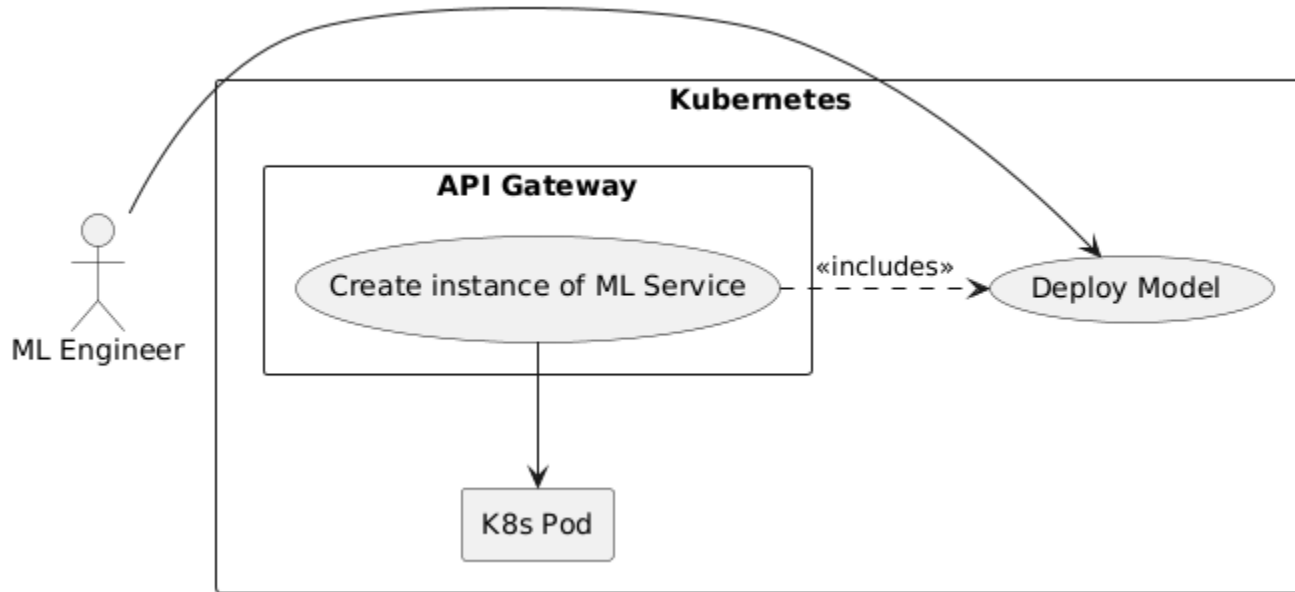
# Use Cases: Request Validation



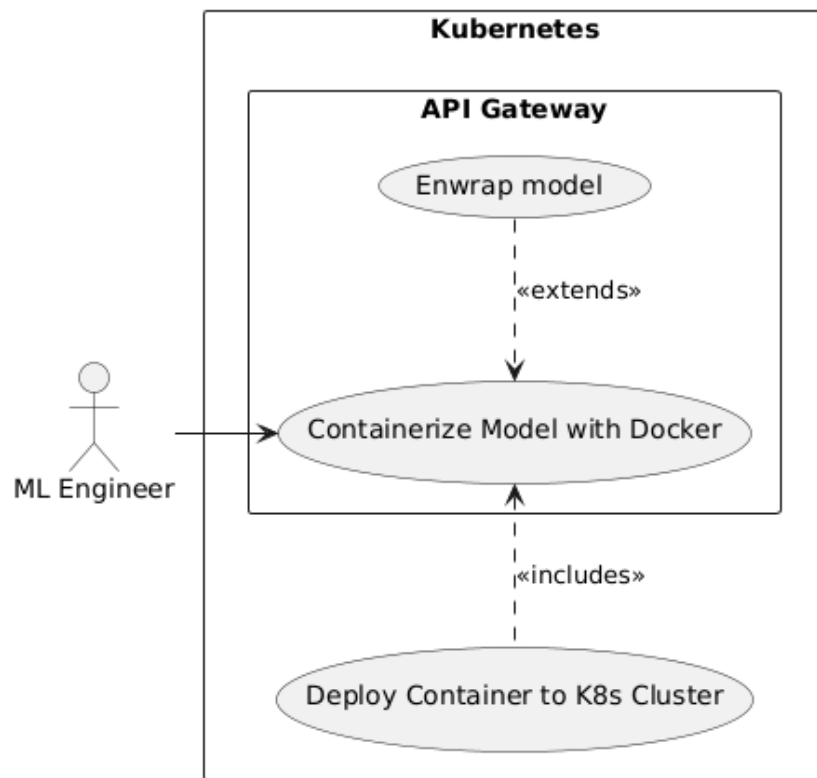
# Use Cases: Responses Caching



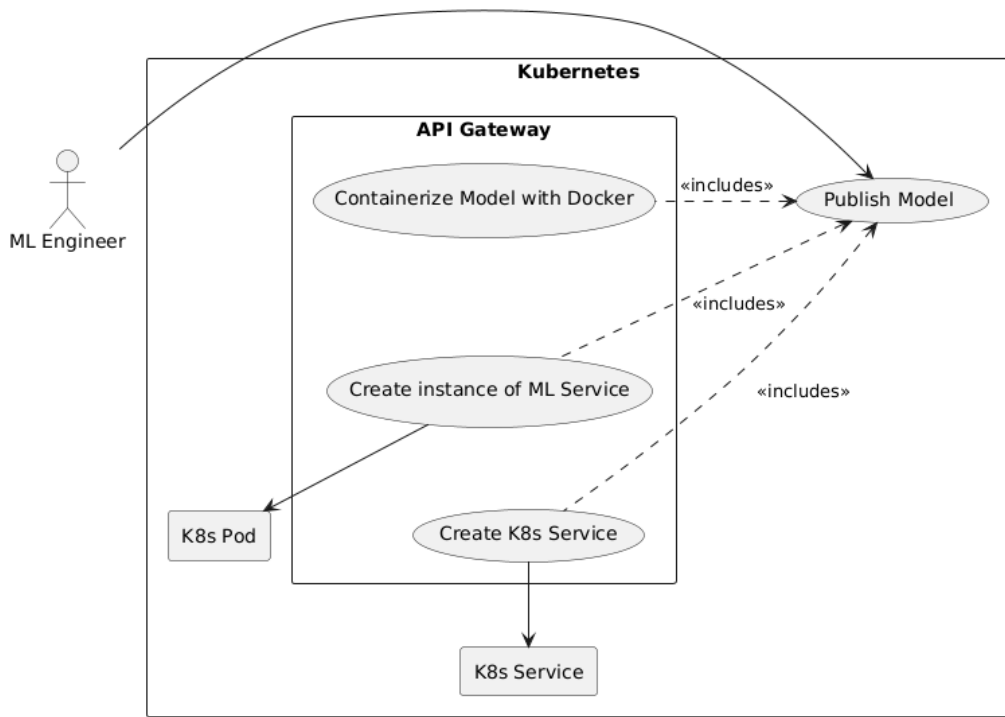
# Use Cases: Modular Deployment of Models



# Use Cases: Containerization



# Use Cases: Service Deployment





# Use Cases: Auto-Documentation

