

KEA

API Design

Product description

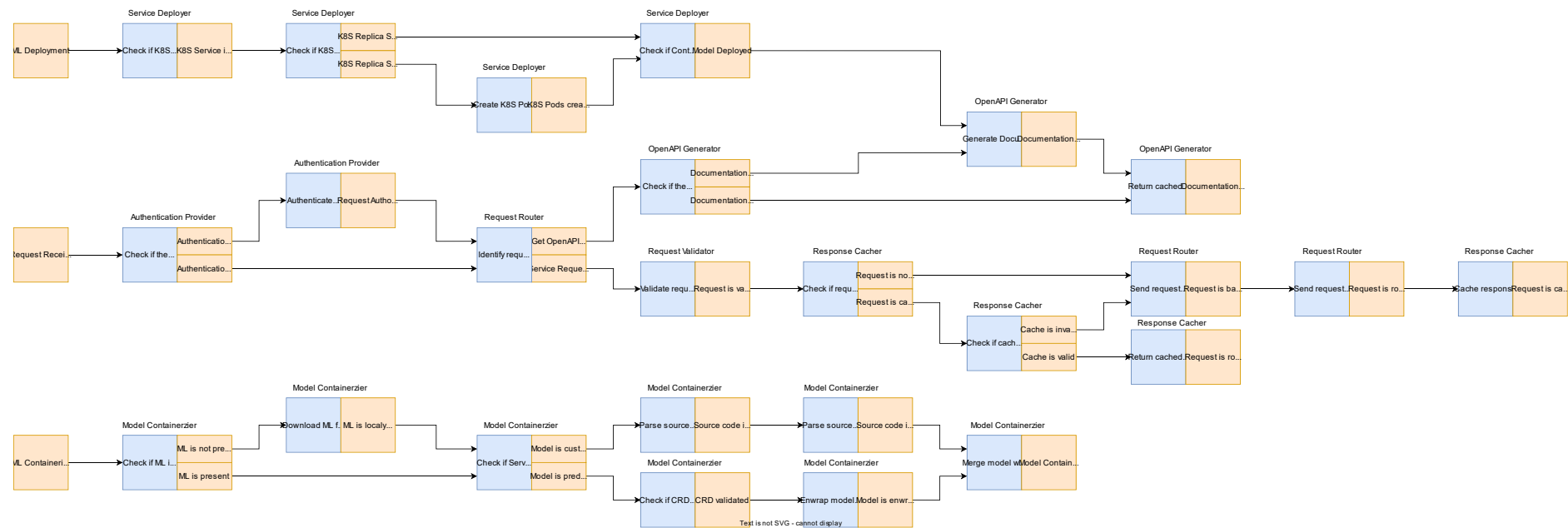
The product is a platform for deploying, managing, and scaling machine learning models in production. It offers a secure, flexible environment for automating ML tasks like model versioning, routing, and monitoring. With Kubernetes integration and containerization support, it's designed for developers,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

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

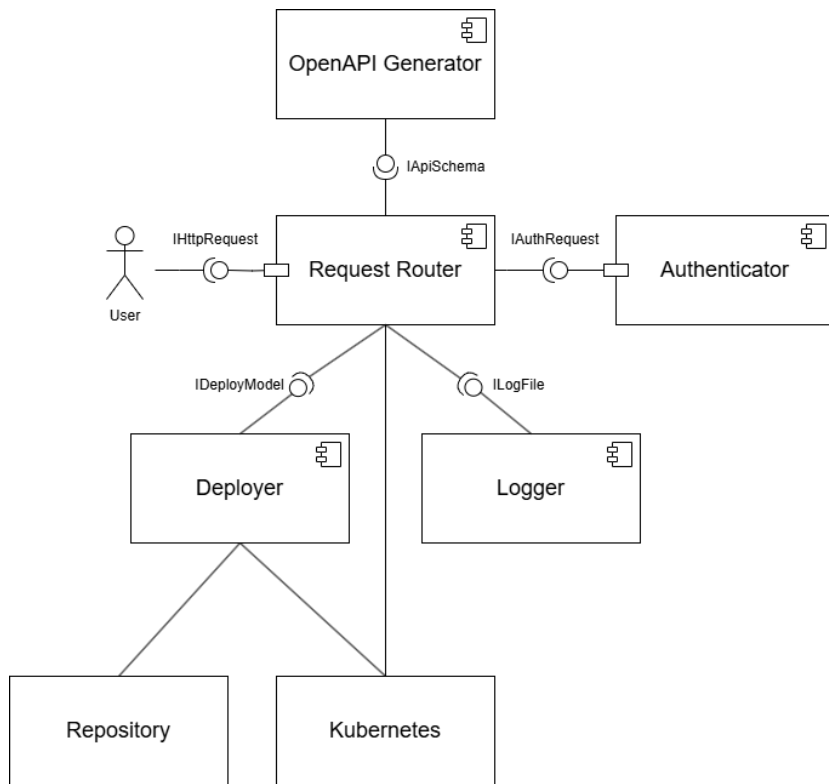
Project repo: <https://github.com/fanglores/Advanced-Software-Design>

This report: [https://github.com/fanglores/Advanced-Software-Design
/blob/master/Practice%20Tasks/Module2/Task_10/Task_10.pdf](https://github.com/fanglores/Advanced-Software-Design/blob/master/Practice%20Tasks/Module2/Task_10/Task_10.pdf)

Event flow



Service diagram



Open API

Using an Open API editor develop RESTful API for all microservices in the project, including data transfer objects. Store the API schema in your repository

(Note that the default example is excessive in detail)

<https://editor.swagger.io>

Verify step-by-step that your API supports scenarios or event flow

For each of the microservices show a scenario fragment/flow fragment where its API is invoked

<At least one microservice per team member>

API usage <Service-1>

Fragment of a scenario -
step before, step when
invoked, a step after

Or user story


Or event flow fragment

Show all (most) places
where the service is used

pet Everything about your Pets Find out more: <http://swagger.io> ^


POST

/pet Add a new pet to the store

✓ 

GET

/pet
/findByStatus Finds Pets by status

✓ 

Solution stack (prepare)

Find an example implementation of a microservices application in the programming language chosen.
Specify one value for each option below

Implementation

- API definition (e.g. OpenAPI, jax-rs, gRPC, pact?)
- Connection server for API (e.g. python gunicorn, java servlets, spring boot/cloud, glassfish...)
- App framework (e.g. python flask/django, node.js, spring framework, jakarta?...)
- Serialization/state format (json, xml, protobuf...)

Asynchronous interactions (optional)

- Message queue (e.g. rabbitmq, kafka, redis streams...)
- Messaging client library (e.g. celery, spring stream ...)

Testing tools (e.g. pytest, junit, mockito, arquillian ...)

Operations

- App initializer (e.g. Spring Initializr)
- Code build (e.g. maven, makefile)
- CI/CD pipeline (e.g. jenkins, gitlab..)
- Delivery method (e.g. docker, war/ear, helm)
- Logging & monitoring (logrotate, prometheus, ELK, grafana, ...) (optional)

Some references

<https://github.com/mfornos/awesome-microservices>

<https://awesomeopensource.com/projects/microservices-architecture>

<https://www.redhat.com/en/blog/comparing-openapi-grpc>

<https://cloud.google.com/apis/design/resources>