

# KEA

Full design

# Product description

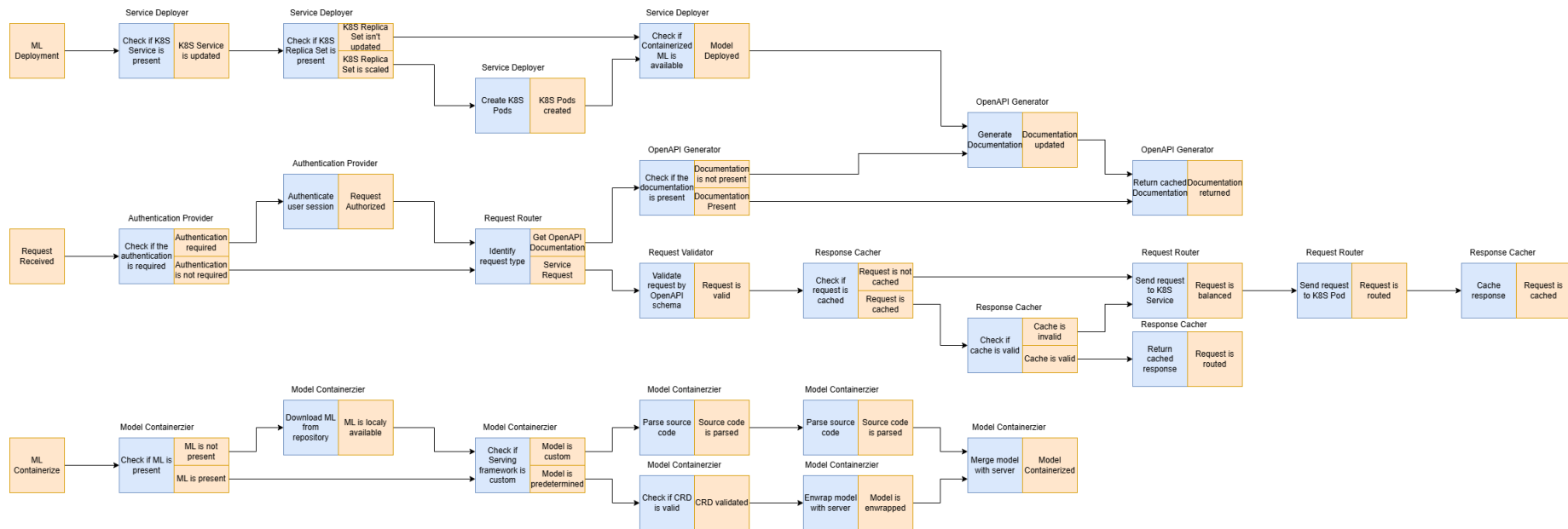
The product is a platform for deploying, managing, and scaling machine learning models in production. It offers a secure, flexible environment for automating ML tasks like model versioning, routing, and monitoring. With Kubernetes integration and containerization support, it's designed for developers,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

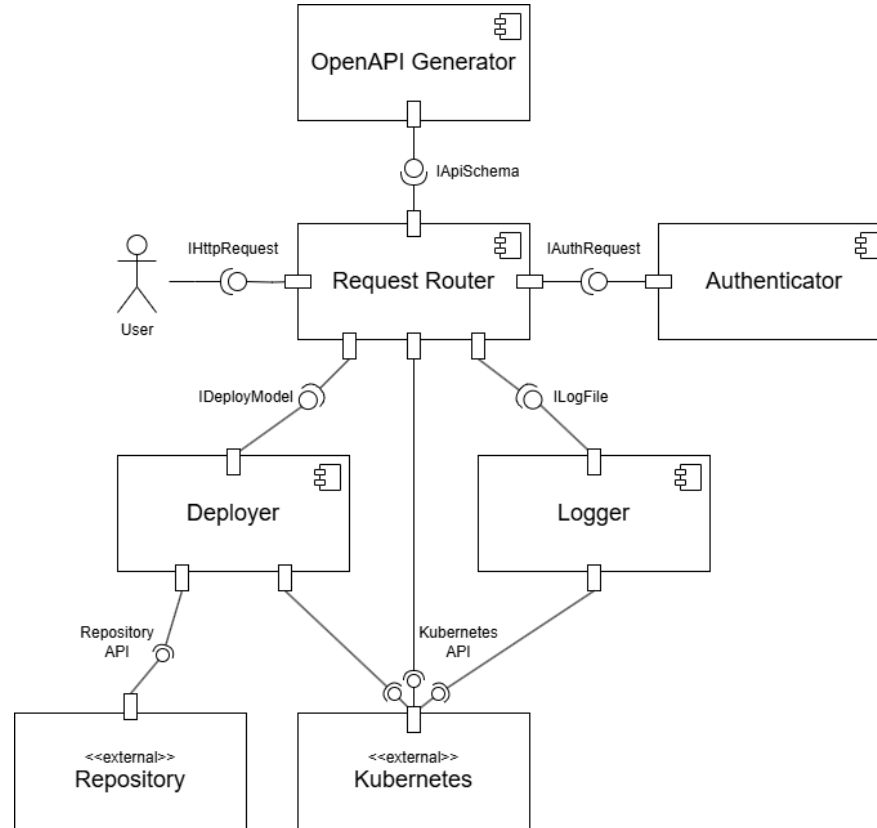
**This report:** [https://github.com/fanglores/Advanced-Software-Design  
/blob/master/Practice%20Tasks/Module2/FinalTask\\_2/FinalTask\\_2.pdf](https://github.com/fanglores/Advanced-Software-Design/blob/master/Practice%20Tasks/Module2/FinalTask_2/FinalTask_2.pdf)

## Event flow



# System architecture

BASE  
Microservices  
RESTful API



# Solution stack

## Implementation

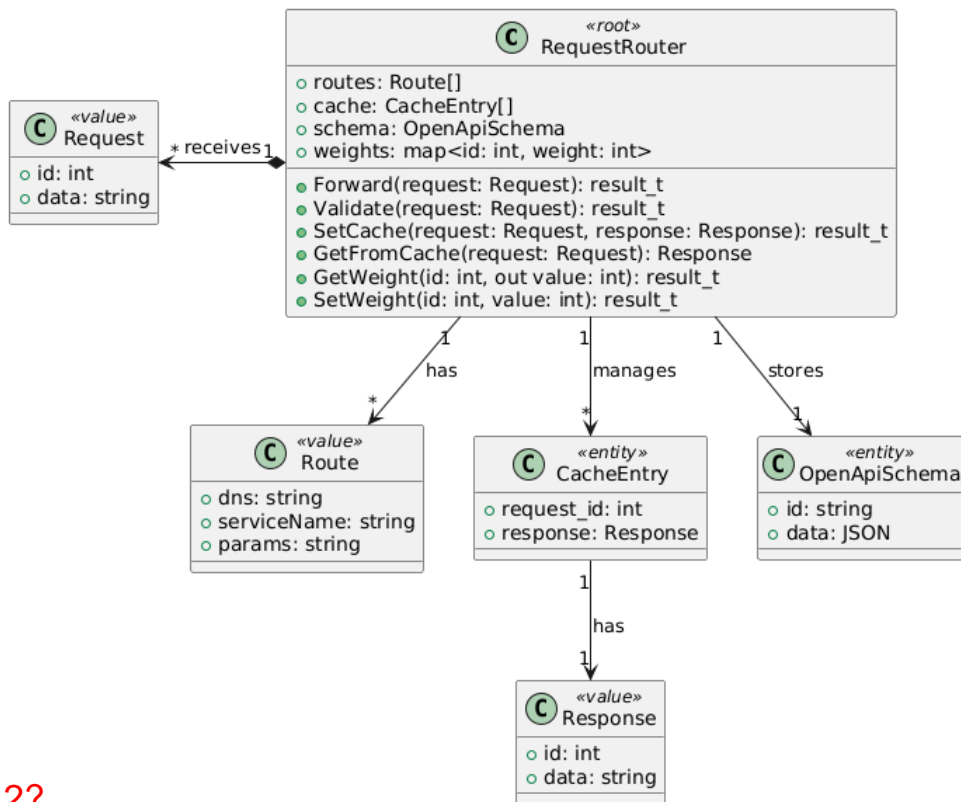
- API definition: OpenAPI
- Connection server for API: python gunicorn
- App framework: python FastAPI
- Serialization/state format: json

## Testing tools pytest

## Operations

- App initializer: cookiecutter
- Code build: makefile
- CI/CD pipeline: github ci/cd
- Delivery method: docker
- Logging & monitoring: ELK

# Logical data model RequestRouter



Should update to  
correspond task 12?

# API usage RequestRouter

**Use Case:** Forward Request

**Scenario:**

User sends request to a service

Request is being validated by OpenAPI schema

Request is being forwarded to a specific K8s service

**RequestRouter** Routes and validates API requests, with caching support. ^

POST

/router/loadbalancer Configure load balancer for a deployed ML service



GET

/router/loadbalancer List all load balancer configurations



GET

/router/loadbalancer/{serviceId}  
Get load balancer configuration for a specific ML service



DELETE

/router/loadbalancer/{serviceId} Remove load balancer configuration



POST

/router/validate Validate an API request against OpenAPI schema

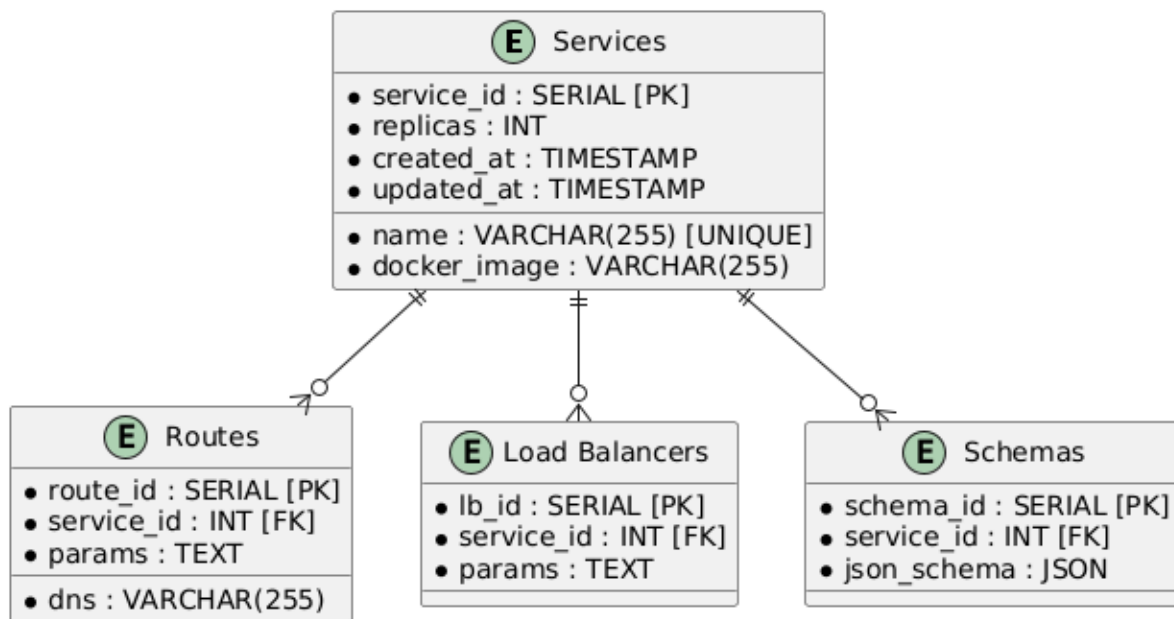


GET

/router/cache Fetch cached API response



# Physical schema RequestRouter

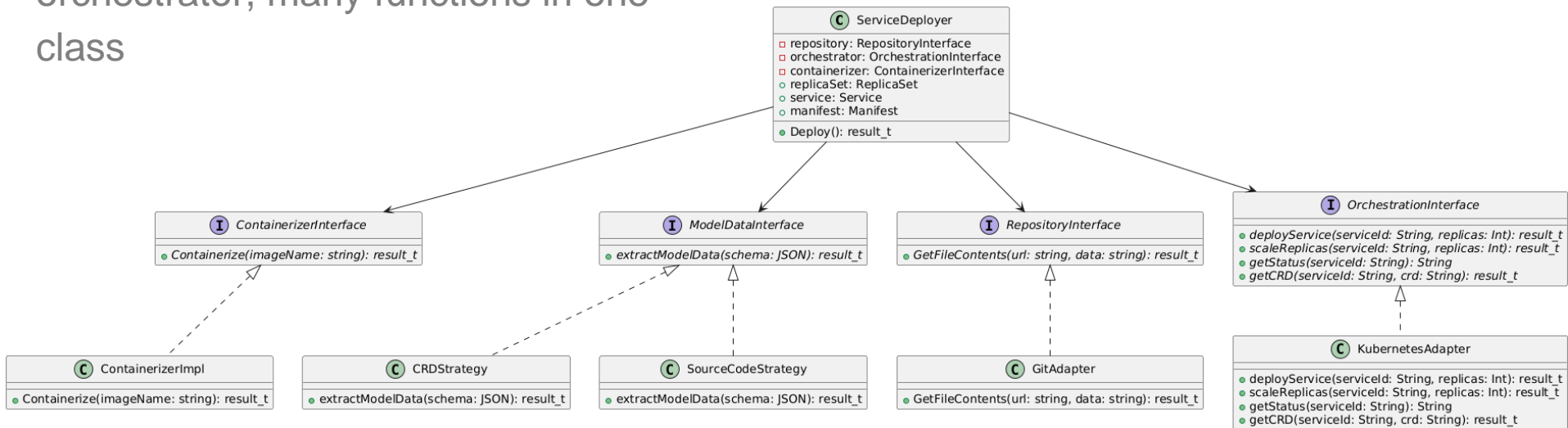




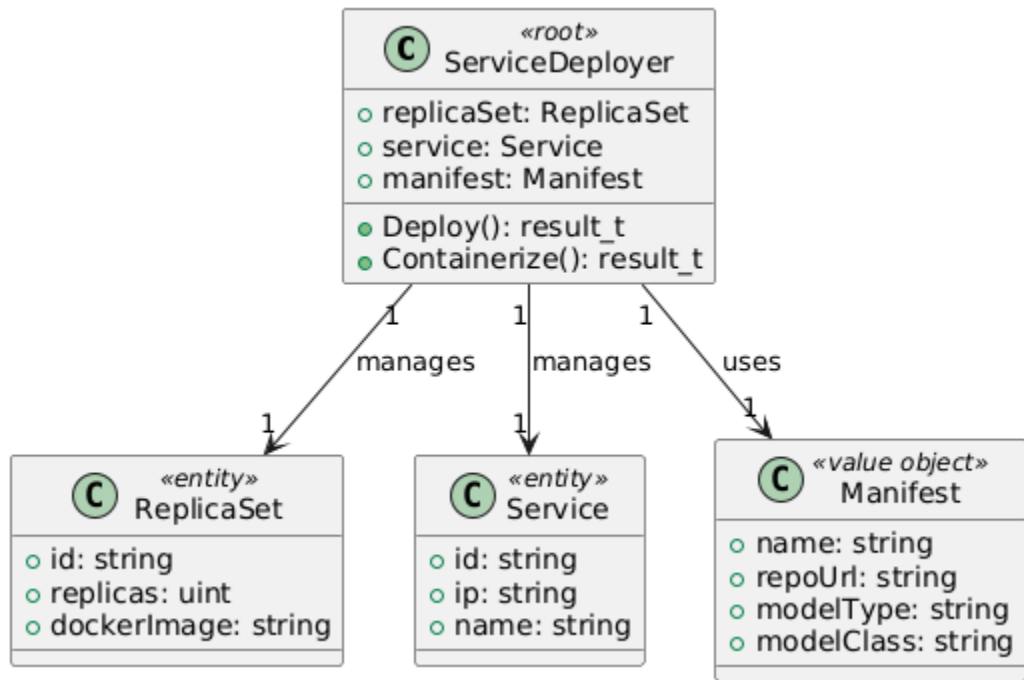
# Design case of Service Deployer

Problems: new deploy strategies  
require changes in ServiceDeployer;  
ServiceDeployer can work with  
different data, repository or  
orchestrator, many functions in one  
class

Solutions: use SOLID principles, Adapter  
and Strategy patterns



# Logical data model **ServiceDeployer**



Should update to  
correspond task 12?

# API usage **ServiceDeployer**

**Use Case:** Deploy Service

**Scenario:**

User sends request to  
deploy ML Model

ML Wrappers creates  
docker container

Docker container is  
deployed via service into  
K8s

**ServiceDeployer** Deploys and manages ML services. ^

POST

/services Deploy an ML service



GET

/services List all deployed services



GET

/services/{serviceId} Get service details



PUT

/services/{serviceId} Update an ML service



DELETE

/services/{serviceId} Delete an ML service

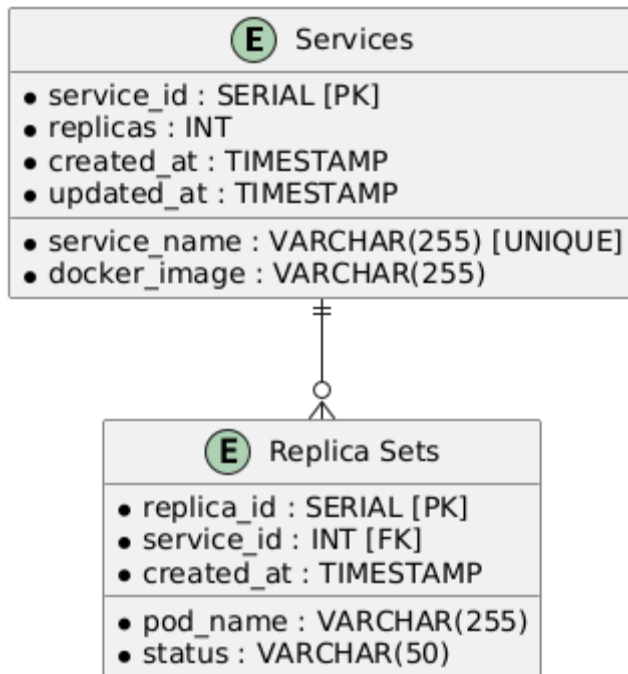


POST

/services/{serviceId}/predict Make a prediction



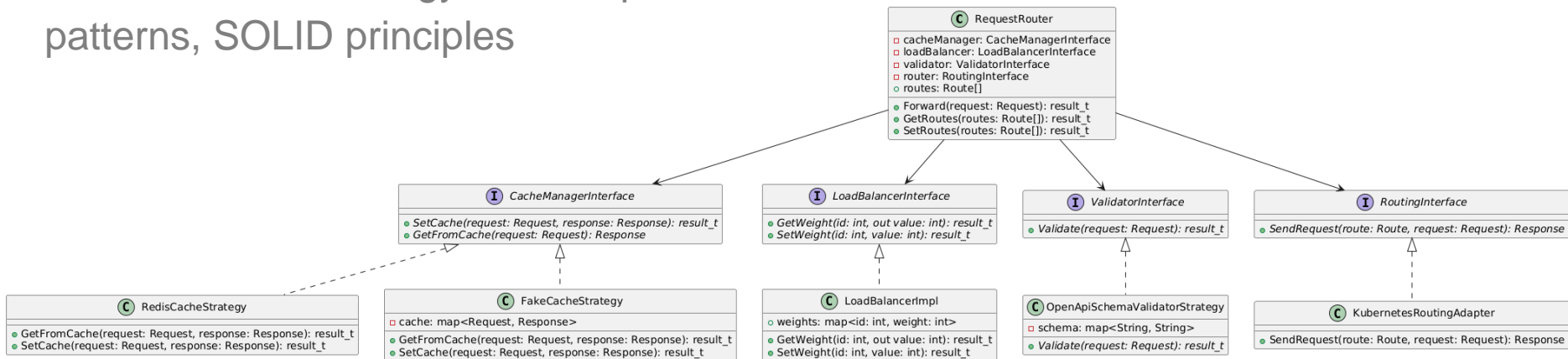
# Physical schema **ServiceDeployer**



# Design case for RequestRouter

Problems: strong dependency on  
Kubernetes, OpenAPI schemas,  
cache storage, many functions

Solutions: use Strategy and Adapter  
patterns, SOLID principles



# Design complexity

Calculate metrics:

- All from Chidamber-Kemerer suite (except RFC, LCOM) for all classes in the largest microservice
- Service dependency metrics (SIY, AIS, ADS) for all microservices from this review:

[Bogner, J., Wagner, S., & Zimmermann, A. \(2017, October\). Automatically measuring the maintainability of service-and microservice-based systems: a literature review. In Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement \(pp. 107-115\).](#)

# System demo

<Draw a diagram to present the structure of the system as deployed on k8s cluster.  
Choose appropriate notation>

<For each use case paste a link to a video with a demonstration how the product works at the k8s cluster. At least the main use case, recommended - one use case per team member>

Use case - <name of the UC/Story>

<url to video/screencast>

<Be prepared to repeat the demo at the exam>

# Repository structure

Repository structure

```
.
├── product_img.jpg
├── README.md
├── General/
│   ├── Domain_description_en.md
│   ├── Task_description_en.md
│   ├── DFD0/
│   ├── StoryMap/
│   └── UseCases/
└── Practice Tasks/module2
    ├── FinalTask_2/
    ├── Task_8/
    ├── Task_9/
    ├── Task_9.1/
    ├── Task_10/
    ├── Task_11/
    └── Task_12/
```

Tools Used:

- Github
- Drawio
- Planttext
- Swagger



# Team and roles



Class diagrams,  
design complexity

Use cases, design cases

Design cases/patterns,  
logical and physical  
schemas, components  
diagram

Event flow, API  
definition, K8s  
deployment,  
components diagram

Tsurkan Daniel  
Tg: @crazy\_deyzi

Dandamaev Gadji  
Tg: @dandamaev

Tsaturyan Konstantin  
Tg: @fanglores

Smolkin Mikhail  
Tg: @m0hnatik