

0x22E

Oleg Sidorenkov (responsible for K8s infrastructure)

Maria Tsvetkova

Ksenia Poliakova

Anton Timonin

Egor Timonin

Product Description

Web editor for reproducible pipelines designed for creating, editing, validating specifications with a library of predefined and custom types that follow an ML experiment domain model defined as a semantic network represented in YAML. The editor could be used as an extension in VS Code or as Web App (WPA) served from website.

- **Store Specifications:** allows users save specification either by their own or via customizable autosave.
- **Edit Specifications:** built-in editor allows users to easily edit pipelines specifications without need to leave website or IDE.
- **Specifications Validation:** automatically checks job specifications for syntax errors, helps validate pipeline scripts, ensuring they are error-proof without need to execute.
- **Code Hints:** includes code highlighting, autocompletion and prompts (e. g. entity name automatic suggestions) and highly customizable linting.
- **Repository Integration:** all the changes can be applied to a GitLab/GitHub repository specification file.
- **Framework Integration:** specification state changes affect the MLOps platform via CI/CD framework integration.

Team: 0x22E

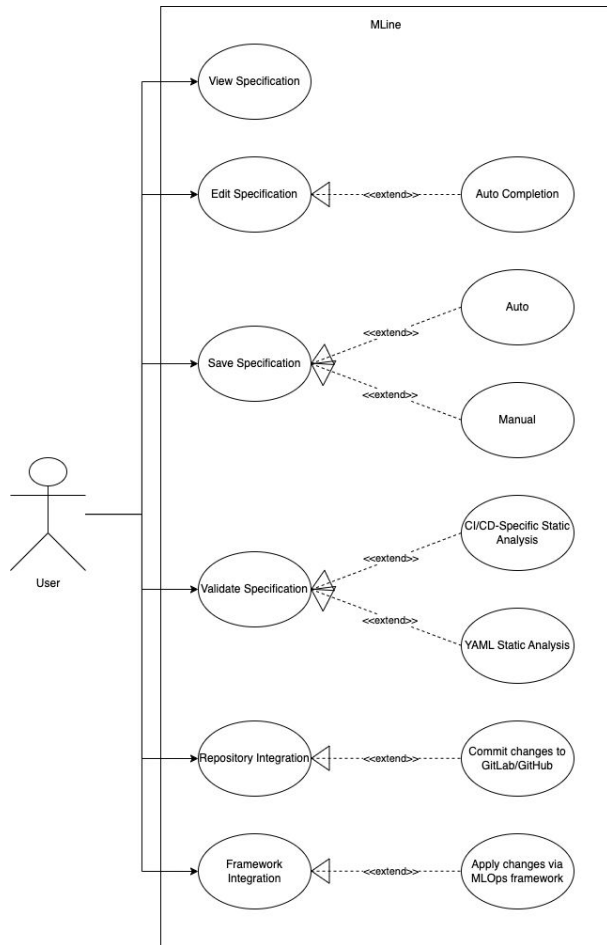
Project repo: <https://github.com/2Delight/mline>

Use case diagram

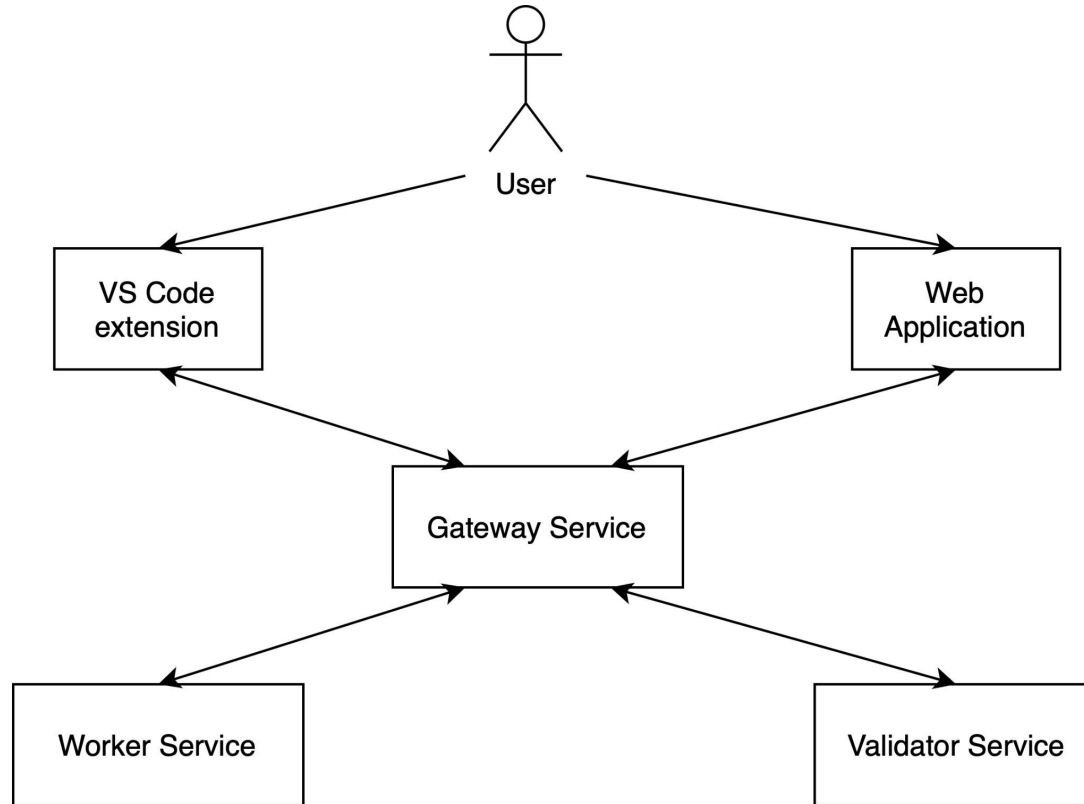
(take from the previous task)

Show main scenarios on a diagram

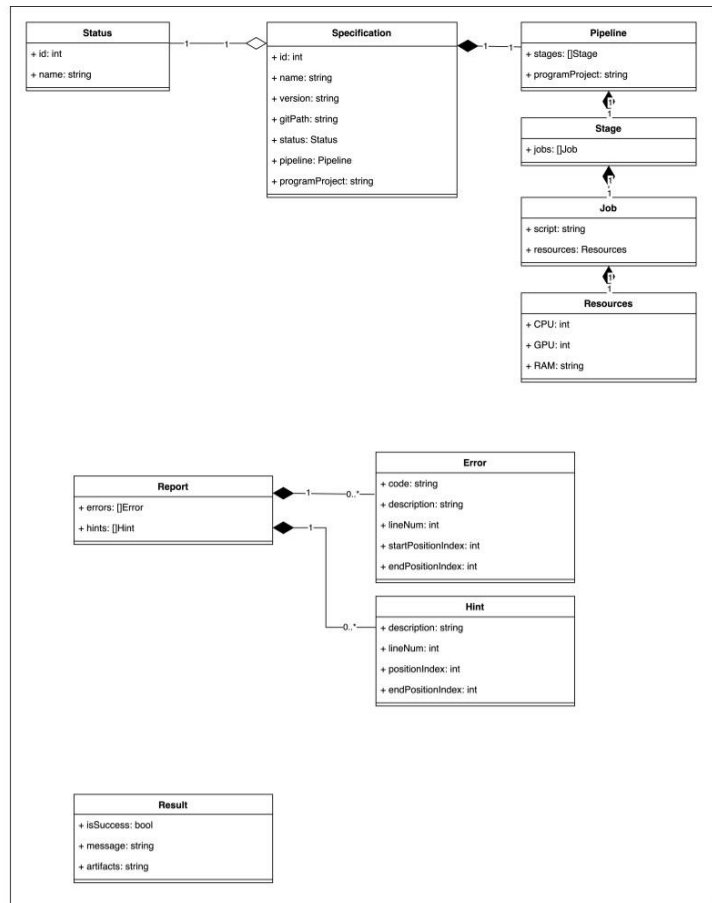
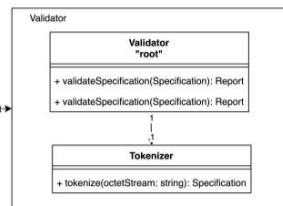
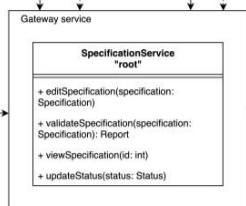
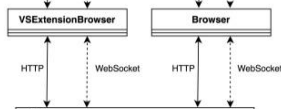
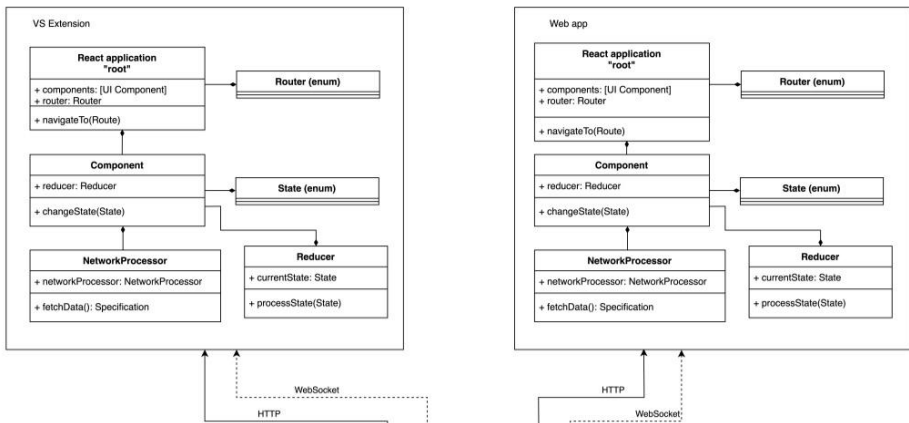
Provide a link to structured textual use case scenarios in your project repo.



DFD Diagram



Aggregate diagram



Open API (Gateway Service)

Gateway Методы для работы с Gateway Service.



GET	/gateway/specifications/{id}	Получить спецификацию	▼
PUT	/gateway/specifications/{id}	Обновить спецификацию	▼
GET	/gateway/specifications/{id}/status	Получить статус спецификации	▼
POST	/gateway/specifications/{id}/status	Обновить статус спецификации	▼
POST	/gateway/specifications/{id}/validate	Запросить валидацию спецификации	▼

Gateway service

SpecificationService "root"

```
+ editSpecification(specification: Specification)
+ validateSpecification(specification: Specification): Report
+ viewSpecification(id: int)
+ updateStatus(status: Status)
```

Use Cases:

View specification

Edit specification

Validate specification

Framework integration

Repository integration

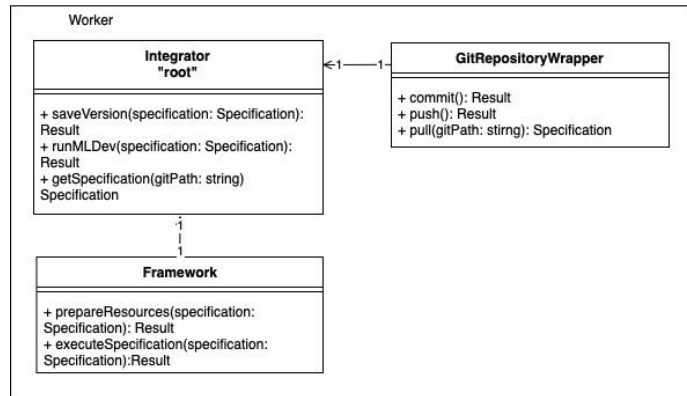
Open API (Worker service)

Use Cases:

Save specification

Framework integration

Repository integration



Worker Методы для работы с Worker Service.



POST `/worker/specifications/{id}/save` Сохранить спецификацию



POST `/worker/specifications/{id}/run-mldev` Запустить MLDev



GET `/worker/specifications/{id}/get` Получить спецификацию



Open API (Validator service)

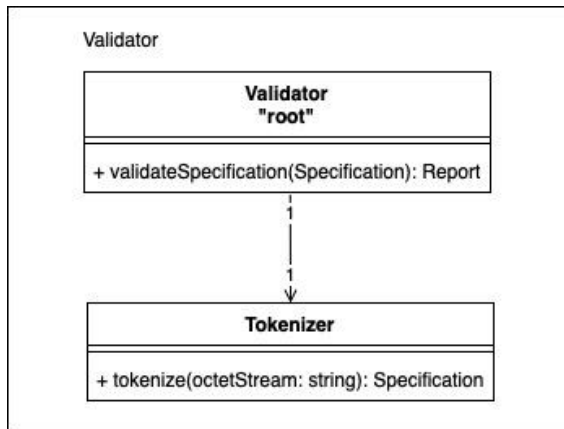
Use Case: Validate specification

Validator Методы для работы с Validator Service.



POST

/validator/specifications/validate Провалидировать спецификацию



Design complexity

Chidamber-Kemerer suite (except RFC, LCOM) for Gateway Service

Dependency metric / Class	SpecificationService	Status	Specification	Pipeline	Stage	Job	Resources	Report	Error	Hint
WMC (Weighted Methods Per Class)	4	0	0	0	0	0	0	0	0	0
DIT (Depth of Inheritance Tree)	0	0	0	0	0	0	0	0	0	0
NOC (Number of Children)	0	0	0	0	0	0	0	0	0	0
CBO (Coupling between Object Classes)	3	0	2	1	1	1	0	2	0	0

Service dependency metrics (SIY, AIS, ADS) for all microservices from this review

Dependency metric / Service	VS Extension	Web App	Worker Service	Gateway Service	Validator Service
SIY (Services Interdependence in the System)	0	0	1	1	0
AIS (Absolute Importance of the Service)	0	0	1	3	1
ADS (Absolute Dependence of the Service)	1	1	2	2	0

Solution stack

Implementation

- API definition: OpenAPI, gRPC
- Connection server for API: Golang net/http
- App framework: React TS, Go STL
- Serialization/state format: json, protobuf, raw bytes

Asynchronous interactions

- Messaging client library: WebSocket API

Testing tools: testing, testify, Jest, Go build-in testing

Operations

- Code build: makefile, npm builder
- CI/CD: GitHub Actions, ArgoCD
- Linux: Alpine
- K8s: Helm

Physical schema for Gateway Service (RDBMS)

```
CREATE TABLE specifications (  
  id SERIAL PRIMARY KEY,           -- Unique identifier  
  name VARCHAR(255) NOT NULL,      -- Specification name  
  git_path VARCHAR(512) NOT NULL,   -- Git repository path  
  status VARCHAR(50) NOT NULL,      -- Specification status (e.g., git_processing, git_failed, ...)  
  created_at TIMESTAMP DEFAULT NOW(), -- Creation timestamp  
  updated_at TIMESTAMP DEFAULT NOW() -- Last update timestamp  
);  
-- Indexes for faster querying  
CREATE INDEX specifications_git_path_ids ON specifications (git_path);
```

Table 'specification' is used to store statuses for specification. Also it includes information about name, git_path, created_at and updated_at for specifications.

Query flow:

1. Insert status specification with default status when it first time requested
2. Update status when specification changes status
3. Fetch status by git_path field

Specifications
+ id: SERIAL (PK)
+ name: VARCHAR(255)
+ git_path: VARCHAR(512)
+ status: VARCHAR(50)
+ created_at: TIMESTAMP
+ updated_at: TIMESTAMP

Physical schema for Gateway Service (NoSQL)

Key: specification_v1:9a0f5d8abf5d3e2dbb1ef1a84d9c77f5

Value:

```
{
  "specification_git_path": "https://github.com/MLDev/highextensive/blob/master/example.yaml",
  "value": "pipeline: !GenericPipeline\r\n # the pipeline sets the sequence of steps \r\n runs:\r\n  - !BasicStage\r\n    # each step is a state\r\n    # which takes inputs from \r\n    and\r\n    # puts outputs back to disk\r\n    name: hello_world\r\n    script:\r\n    # this is our bash script\r\n    - echo Hello World!\r\n    # note that we need not to escape\r\n    double quotes\r\n    # if we use text block like this \r\n    - >\r\n      echo \"Time is $(date)\""
```

Redis cache is used to exclude requests to external data source.

Query flow:

1. Insert after changing specification content.
2. Fetch specification content using hash from specification data.

For example, it could be SHA-256 from name + git_path + updated_at

Cache TTL would be about several minutes. This cache is not persistent.

Physical Schema for Validator

Redis: Hash<String, Hash<String, ...>>

Redis cache is used to prevent excessive computations

Query flow:

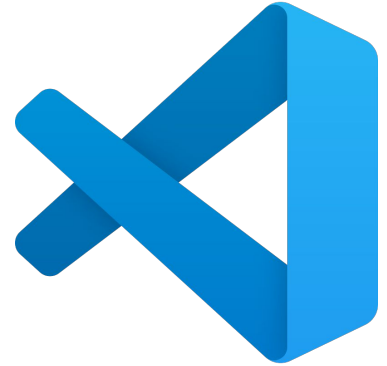
1. Insert after changing specification content
2. Try to fetch by hashing the file contents

Cache TTL would be not longer than 1 hour



Physical Schema for Visual Studio Code Extension

```
1 CREATE TABLE yaml_errors (  
2   id INT AUTO_INCREMENT PRIMARY KEY, /*Unique identifier for each error type.*/  
3   code VARCHAR(255) UNIQUE NOT NULL, /*A unique error code (e.g., YAML001, YAML002).*/  
4   message TEXT NOT NULL, /*A human-readable description of the error.*/  
5   severity VARCHAR(255) NOT NULL, /*Severity level.*/  
6   description TEXT, /*A more detailed explanation of the error and how to fix it.*/  
7   example TEXT, /*An example of YAML code that triggers this error.*/  
8   regex TEXT /*A regular expression that can be used to detect this error.*/  
9 );
```

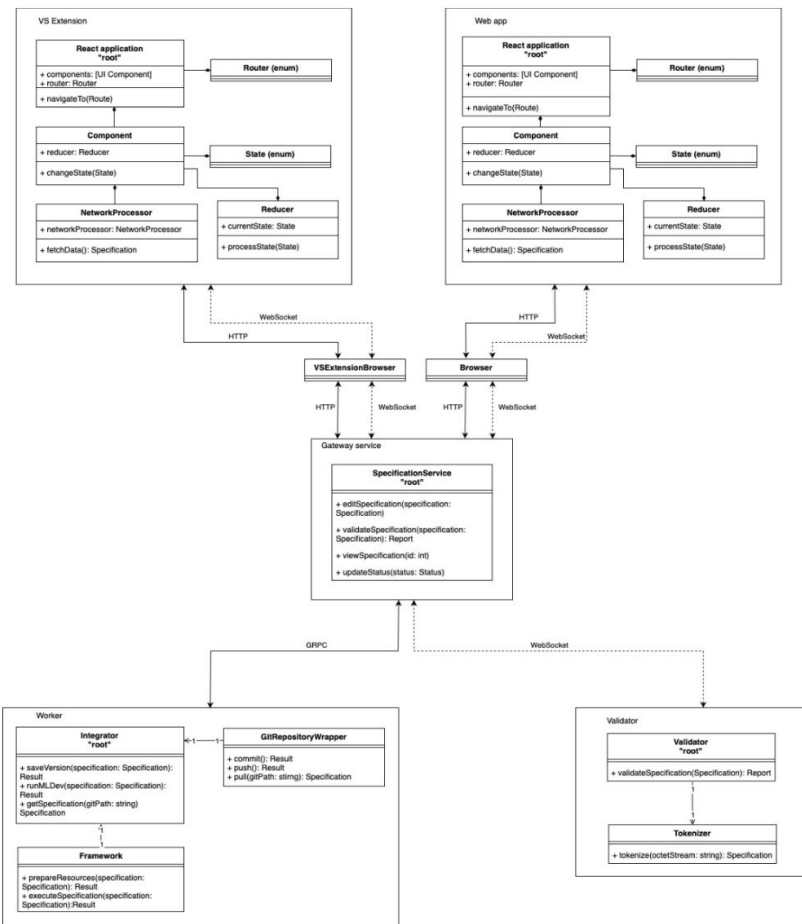
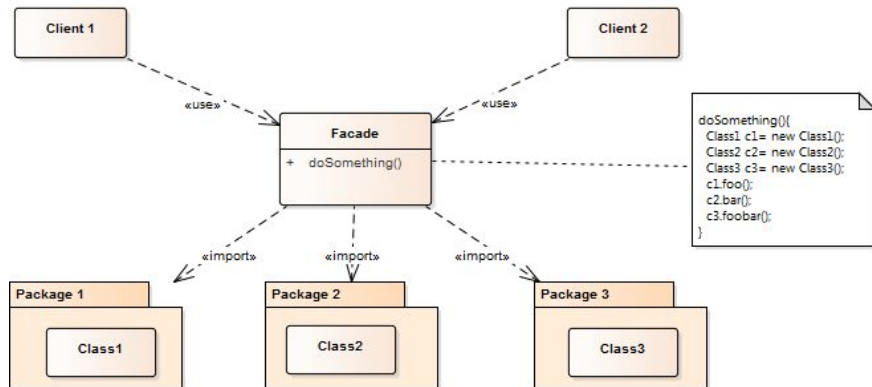


We would suggest to use SQLite as it's a lightweight, file-based database that's easy to embed in extension. It's suitable if we need to store and query structured data, such as a catalog of common YAML errors or a history of user-reported errors. Positive sides: Self-contained, no separate server required, relatively easy to use, good performance for moderate-sized datasets. Negative sides: Not as performant as larger server-based databases for massive datasets.

Design case #1 Facade

Problem: We have several frontends, so it is required to have one point which should provide communication with other backends

Solution: We created facade service

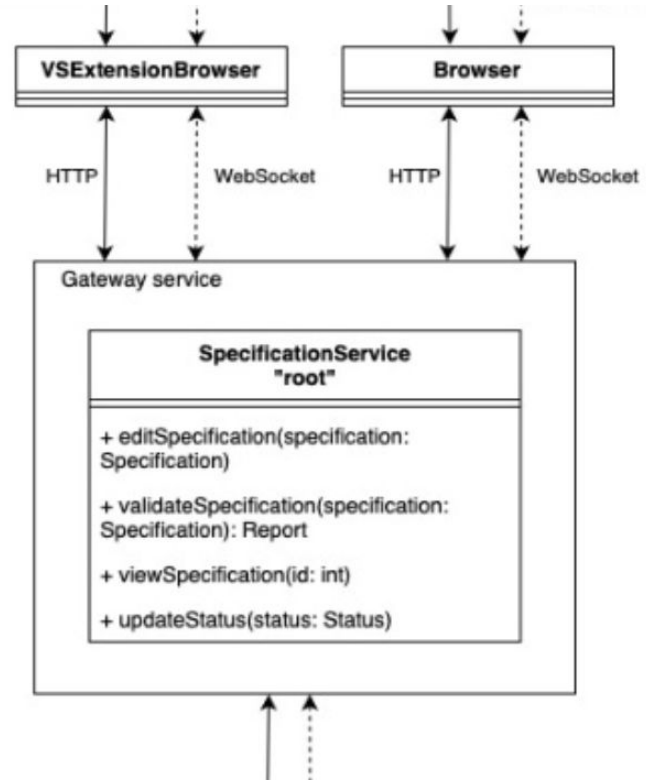


Design case #2

Proxy

Problem: Control and route traffic

Solution: Create a Gateway service that resolves the requests and pushes them further

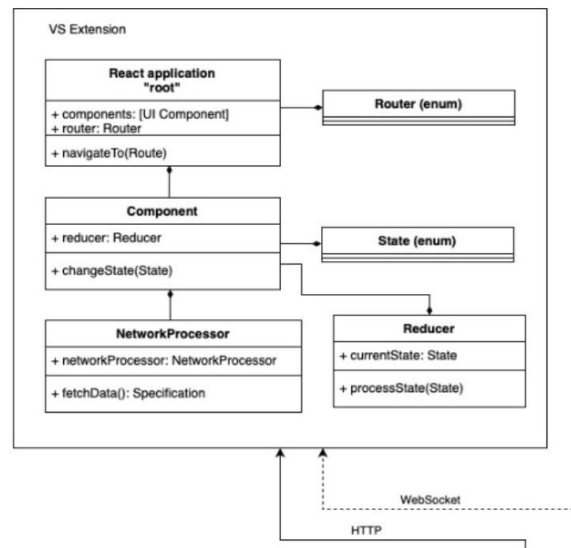


Design case #3

Singleton

Problem: no possibility to controlling access use resources that should be loaded once on user's application host

Solution: To load this resource and than use it
As singleton

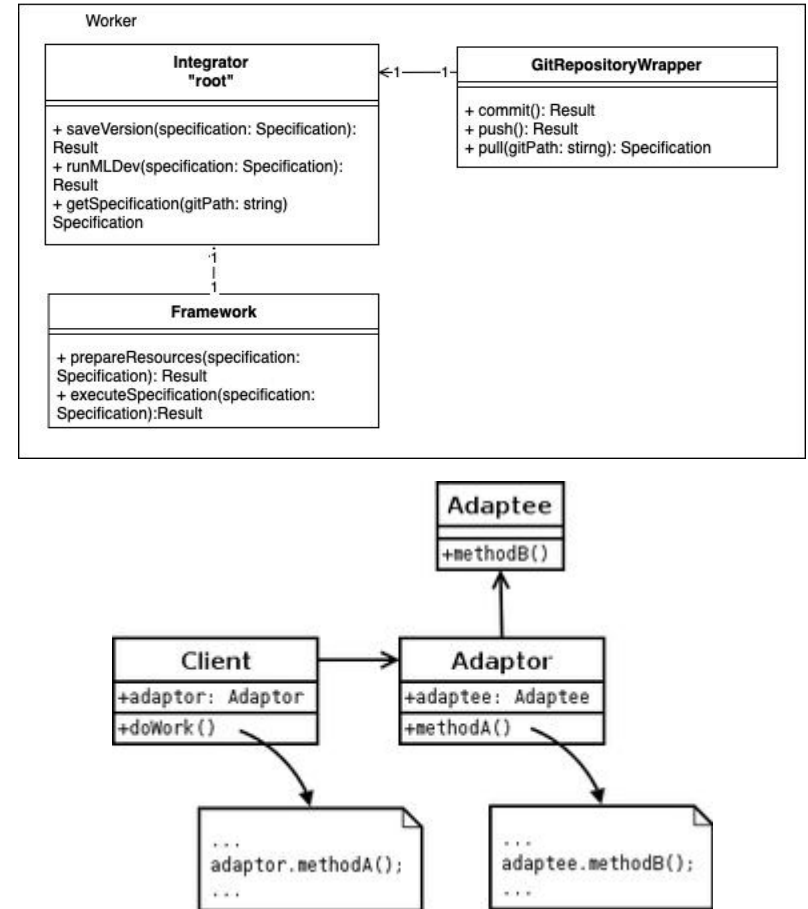


Design case #4

Adapter

Problem: We have several external services (like Git, MLDev), we should handle requests correctly, but response format from gateway don't fit for Git and MLDev

Solution: We created adapters for requests to external services service

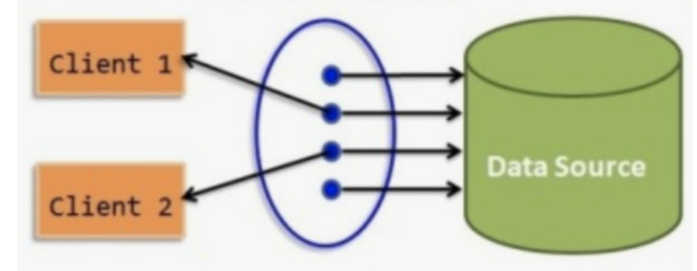


Design case #5

Connection Pool

Problem: Creating a connection on each request is pretty expensive

Solution: Pool with open connections



Prompts (ChatGPT o1)

1. Using the resource {RESOURCE_URL}, please select the two best microservices frameworks or service toolkits considering the following criteria: {CRITERIA}. Provide a brief description of each.
 2. From the two frameworks you selected ({FRAMEWORK_1} and {FRAMEWORK_2}), choose the best one and justify your selection based on {JUSTIFICATION_CRITERIA}.
 3. Using an LLM code assistant, generate "Hello World" examples for the two microservices frameworks ({FRAMEWORK_1} and {FRAMEWORK_2}).
 4. Evaluate the generated "Hello World" examples for the two frameworks ({FRAMEWORK_1} and {FRAMEWORK_2}), considering factors such as {EVALUATION_CRITERIA}
- {RESOURCE_URL}: <https://github.com/mfornos/awesome-microservices>
- {CRITERIA}: Popularity, community support, ease of use, language support, and performance.
- {JUSTIFICATION_CRITERIA}: Project requirements, team expertise, and performance considerations.
- {EVALUATION_CRITERIA}: Code readability, ease of setup, documentation quality, and suitability.

Answers

Top Two Microservices Frameworks

1. Spring Boot (Java)

- **Description:** Spring Boot is a highly popular Java-based framework that simplifies building production-grade microservices. It provides an opinionated approach to configurations, making it easier to get started, while still being customizable.
- **Why It Stands Out:**
 - **Popularity:** Extremely popular with strong enterprise adoption.
 - **Community Support:** Massive support from the Java community, extensive tutorials, and active forums.
 - **Ease of Use:** Pre-configured starter projects and embedded servers like Tomcat reduce setup complexity.
 - **Language Support:** Java, Kotlin, and Groovy.
 - **Performance:** Performs well for enterprise-level microservices and integrates seamlessly with tools like Spring Cloud.

2. Go-Micro (Golang)

- **Description:** Go-Micro is a lightweight, highly modular microservices toolkit for Go. It simplifies building distributed systems by providing abstractions for discovery, load balancing, messaging, and API development.
- **Why It Stands Out:**
 - **Popularity:** Popular within the Go developer community for building performant and distributed systems.
 - **Community Support:** Actively maintained, with Go's growing ecosystem.
 - **Ease of Use:** Simplified setup with clear APIs for microservices development.
 - **Language Support:** Golang.
 - **Performance:** Highly optimized for speed and lightweight memory usage.

2. Go-Micro (Golang)

Here's a basic "Hello World" service using Go-Micro:

Dependencies: Install Go-Micro using `go get github.com/asim/go-micro/v3`

```
go Copy

package main

import (
    "context"
    "fmt"

    "github.com/asim/go-micro/v3"
    "github.com/asim/go-micro/v3/server"
)

// HelloService defines the service interface
type HelloService struct{}

// SayHello is a handler function
func (h *HelloService) SayHello(ctx context.Context, req *Request, rsp *Response) error {
    rsp.Message = "Hello, World!"
    return nil
}

type Request struct{}
type Response struct {
    Message string
}

func main() {
    // Create a new service
    service := micro.NewService(
        micro.Name("hello"),
    )

    // Initialize the service
    service.Init()

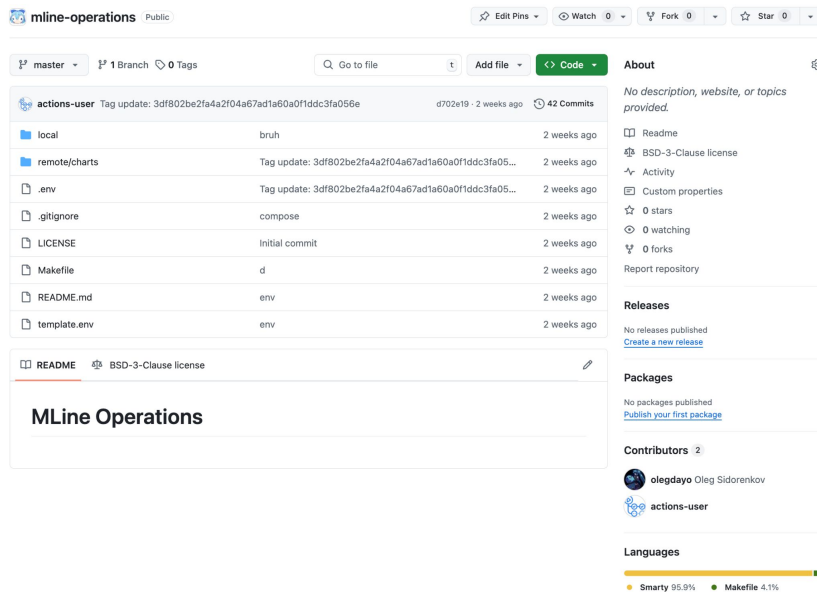
    // Register the handler
    micro.RegisterHandler(service.Server(), new(HelloService))

    // Run the service
    if err := service.Run(); err != nil {
        fmt.Println(err)
    }
}
```

Infrastructure. Repositories

MLine - code

MLine Operations - infrastructure state



mline-operations Public

1 Branch 0 Tags

Go to file

Add file Code

About

No description, website, or topics provided.

Readme

BSD-3-Clause license

Activity

Custom properties

0 stars

0 watching

0 forks

Report repository

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Contributors

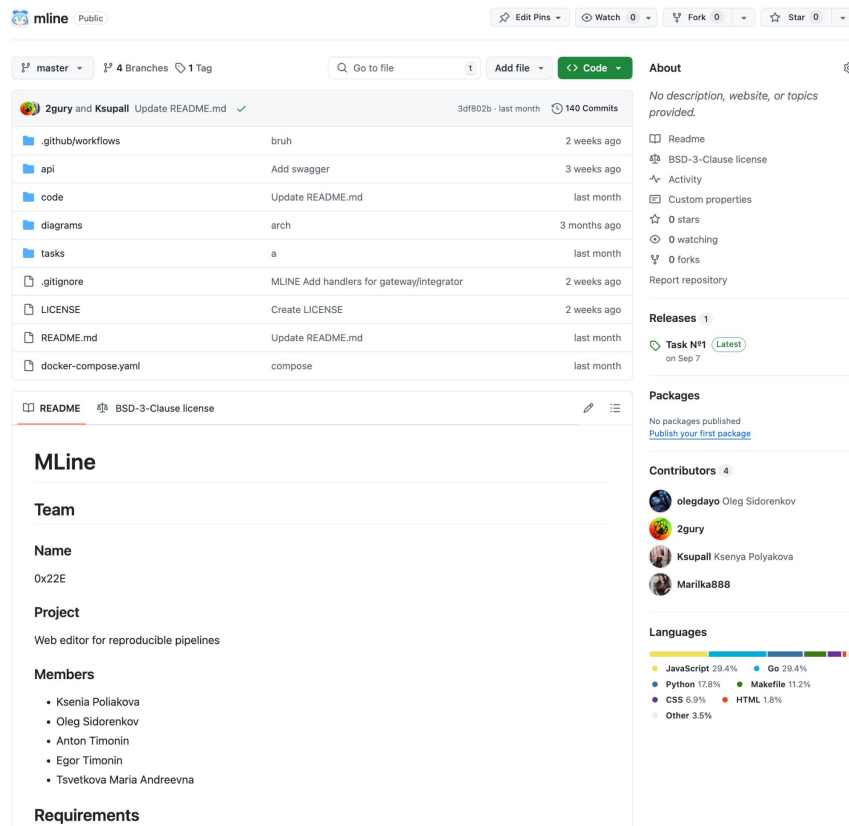
2

olegdayo Oleg Sidorenkov

actions-user

Languages

Smarty 95.9% Makefile 4.1%



mline Public

Edit Pins Watch 0 Fork 0 Star 0

master 4 Branches 1 Tag

Go to file

Add file Code

About

No description, website, or topics provided.

Readme

BSD-3-Clause license

Activity

Custom properties

0 stars

0 watching

0 forks

Report repository

Releases

1

Task N#1 Latest on Sep 7

Packages

No packages published

Publish your first package

Contributors

4

olegdayo Oleg Sidorenkov

2gury

Ksupall Ksenya Polyakova

Marika888

Languages

JavaScript 29.4% Python 17.8% CSS 6.9% HTML 1.8% Other 3.5%

MLine

Team

Name

0x22E

Project

Web editor for reproducible pipelines

Members

- Ksenia Poliakova
- Oleg Sidorenkov
- Anton Timonin
- Egor Timonin
- Tsvetkova Maria Andreevna

Requirements

Infrastructure. Docker

Multistage building

Running in Alpine

Pushed to dockerhub

```
1 FROM rust:latest AS builder
2 WORKDIR /app
3 COPY . .
4 RUN apt update && apt install musl-tools -y
5 RUN rustup target add x86_64-unknown-linux-musl
6 RUN cargo build --release --target x86_64-unknown-linux-musl
7
8 FROM --platform=linux/amd64 alpine:latest AS runner
9 WORKDIR /app
10 COPY --from=builder /app/target/x86_64-unknown-linux-musl/release/pingus .
11 CMD ["/pingus"]
```

olegdayo / [Repositories](#) / [gateway](#) / [General](#)

[General](#)

[Tags](#)

[Builds](#)

[Collaborators](#)

[Webhooks](#)

[Settings](#)

olegdayo/gateway 🤖











Last pushed 14 days ago

[Add a description](#)  [INCOMPLETE](#)

[Add a category](#)  [INCOMPLETE](#)

Tags

This repository contains 25+ tag(s).

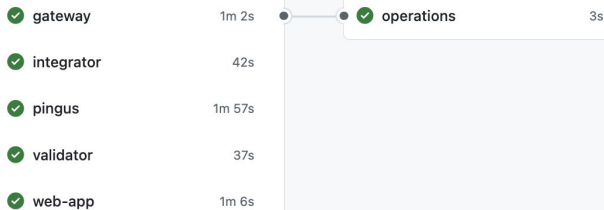
Tag	OS	Type	Pulled	Pushed
 3df802be2fa4a2f04a...		Image	---	14 days ago
 e983d5080c149b7c2...		Image	16 days ago	16 days ago
 b8ca884e8d09de2c5...		Image	16 days ago	16 days ago
 88c529d5e10b095ea...		Image	16 days ago	16 days ago
 0d5189cfe01eb6794...		Image	16 days ago	16 days ago

[See all](#)

Infrastructure. CI/CD

1. Push to dockerhub
2. Commit to operations repo
3. Sync in ArgoCD

publish.yaml
on: push



Commit d702e19

actions-user committed 2 weeks ago

Tag update: 3df802be2fa4a2f04a67ad1a60a0f1ddc3fa056e

master

Filter files...

- .env
- remote/charts
- gateway
 - values.yaml
- integrator
 - values.yaml
- pingus
 - values.yaml
- validator
 - values.yaml
- web-app
 - values.yaml

6 files changed +6 -6 lines changed

```
@@ -1,1 @@
1 - TAG=88c529d5e10b095eaff7eebab3c4d4423d2ad147
1 + TAG=3df802be2fa4a2f04a67ad1a60a0f1ddc3fa056e

remote/charts/gateway/values.yaml
@@ -3,7 +3,7 @@ replicaCount: 1
3 3 image:
4 4   repository: olegdayo/gateway
5 5   pullPolicy: IfNotPresent
6 - tag: 88c529d5e10b095eaff7eebab3c4d4423d2ad147
6 + tag: 3df802be2fa4a2f04a67ad1a60a0f1ddc3fa056e
7 7
8 8 imagePullSecrets: []
9 9
```


Infrastructure. Delivery

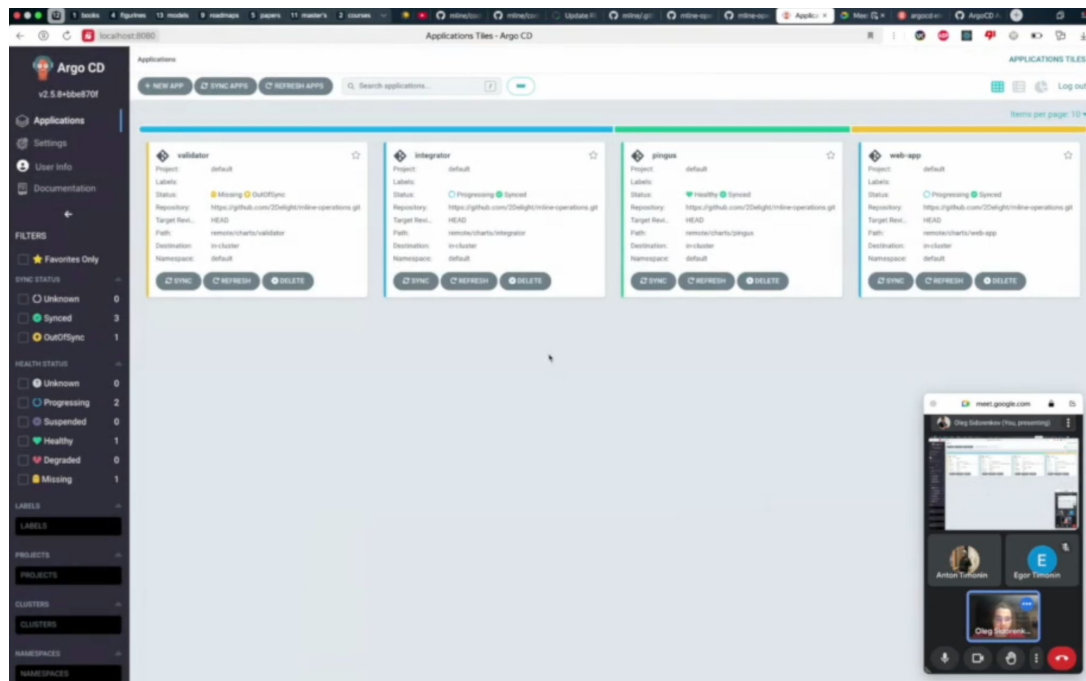
Using Helm + ArgoCD to deploy

[mline-operations](#) / [remote](#) / [charts](#) / [validator](#) / [values.yaml](#)

 actions-user Tag update: 3df802be2fa4a2f04a67ad1a60a0f1ddc3fa056e

Code Blame 69 lines (53 loc) · 956 Bytes

```
1 replicaCount: 1
2
3 image:
4   repository: olegdayo/validator
5   pullPolicy: IfNotPresent
6   tag: 3df802be2fa4a2f04a67ad1a60a0f1ddc3fa056e
```



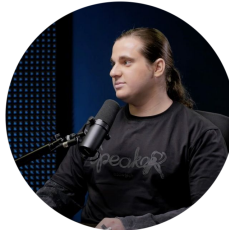
Team and Roles



Egor Timonin
Backend developer



Tsvetkova Maria
Backend developer



Oleg Sidorenkov
Infrastructure engineer
Teamlead
@olegdayo



Ksenia Poliakova
Project manager,
Frontend developer



Anton Timonin
Frontend developer

Thank you for your attention!

