ТИНЬКОФФ

Фича-флаги. Практический пример реализации

Руковишников Михаил

Август 2023

ТИНЬКОФФ

Обо мне

Университет ИТМО

.NET Backend

План доклада

1. Наш проект
2. Проблематика
3. Фича-флаги
4. Исходная реализация
5. Критерии хорошего решения
6. Способы реализации
7. Финальная реализация
8. Демо
9. Итоги
10. Good Practice

Наш проект

- ASP API
- Исходный код в GitLab
- Деплой через Kubernetes
- Scrum
- Trunk Based Development
- Большое количество данных
- Множественные интеграции

Трудности

Неготовый функционал в мастере

В нашем флоу каждая целая фича представлена одной User Story, состоящей из нескольких небольших подзадач.

Мы используем TBD. На каждую подзадачу создаётся отдельная короткоживущая ветка, которая как можно быстрее сливается с мастером.

Высокая сложность тестирования

Бизнес-требования заставляют поддерживать данные всё время в строго консистентном виде.

Обилие интеграций создаёт внешние зависимости также накладывающие определённые ограничения на данные и увеличивающие сложность тестирования.

Feature Flags

Feature Flags

Фича флаги, Feature Flags или Feature Toggles – это концепция переключателей, позволяющих включать или выключать тот или иной функционал приложения.

Feature Flags

Применяются для следующих целей:

- Разработка через TBD
- Канареечный релиз
- Проведение A/B тестирования
- Включение и выключение функционала по решению бизнеса
- Полное отключение функционала на конкретном контуре

Feature Flags

```
public void DoSomething()
{
    if (flag)
    {
        // Выполняем функционал, закрытый фича-флагом
    }

    // Основной функционал функции
}
```

Главные вопросы

Где хранить?

Как проверять?

Как изменять?

Исходная реализация

Флаги через конфигурацию

```
"ExampleClassSettings": {
  "SomeFeature": true
}
```

```
public class ExampleClassSettings
{
    public bool SomeFeature { get; set; }
}
```

```
builder.Services.Configure<ExampleClassSettings>(
    builder.Configuration.GetSection(key:"ExampleClassSettings"));
```

Флаги через конфигурацию

```
public class ExampleClass
{
    private readonly ExampleClassSettings _settings;

    public ExampleClass(IOptions<ExampleClassSettings> options)
    {
        _settings = options.Value;
    }

    public void DoSomething()
    {
        if (_settings.SomeFeature)
        {
            // Выполняем функционал, закрытый фича флагом
        }

        // Основной функционал функции
    }
}
```

Проблемы решения

Высокая сложность изменения конфигурации

Требуется создать MR, дождаться завершения CI/CD, дождаться завершения деплоя

Необходимость участия разработчика

Вышеописанный процесс сложен для не разработчиков

Сложность отката изменений

Откат несёт всё те же проблемы

Критерии хорошего решения

Отсутствие необходимости в редеплое

Хотим включать/выключать фичи прямо в рантайме

Возможность использования не разработчиком

Хотим, чтобы QA, системная аналитика и бизнес могли включать/выключать фичи самостоятельно

Простота отката изменений

Хотим иметь возможность откатить изменения также прямо в рантайме

Возможные решения

IOptionsMonitor

```
public class ExampleClassWithMonitor
{
    private readonly ExampleClassSettings _settings;

    public ExampleClassWithMonitor(IOptionsMonitor<ExampleClassSettings> options)
    {
        _settings = options.CurrentValue;
    }

    public void DoSomething()
    {
        if (_settings.SomeFeature)
        {
            // Выполняем функционал, закрытый фича флагом
        }

        // Основной функционал функции
    }
}
```

IOptionsMonitor

Плюсы:

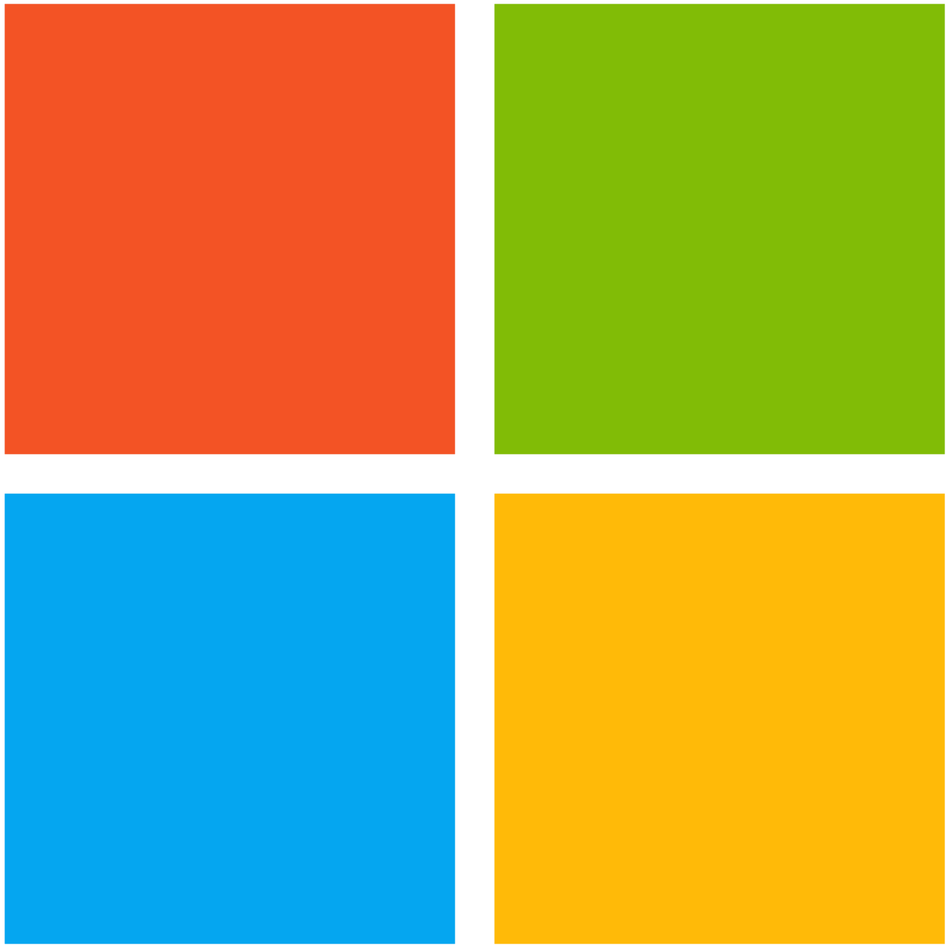
- Простота

Минусы:

- Малая возможность конфигурации
- Непонятно, как переключать флаги

FeatureManagement

Библиотека для ASP, поставляющая методы и интерфейсы для настройки фича-флагов в приложении.



Возможности FeatureManagement

- Возможность кеширования состояния флага в течение запроса
- Встроенные FeatureFilter и возможность написать свои
- FeatureGate и активация MVC сущностей по флагам
- Возможность написать кастомный FeatureProvider

FeatureManagement

Плюсы:

- Множество возможностей
- Подробная документация

Минусы:

- Непонятно, как переключать флаги

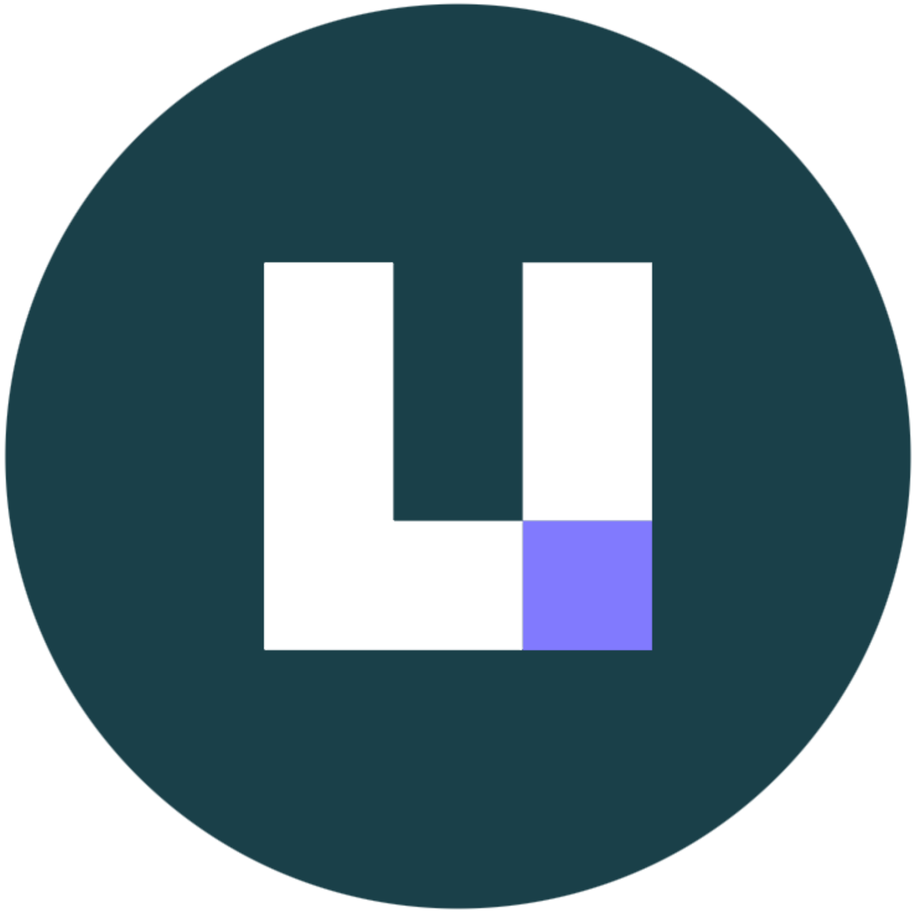
ConfigMap

ConfigMap - это сущность Kubernetes, используемый для хранения данных в виде пар ключ-значение.

Конфигурация может быть передана из ConfigMap как переменные среды, аргументов командной строки или как файлы.

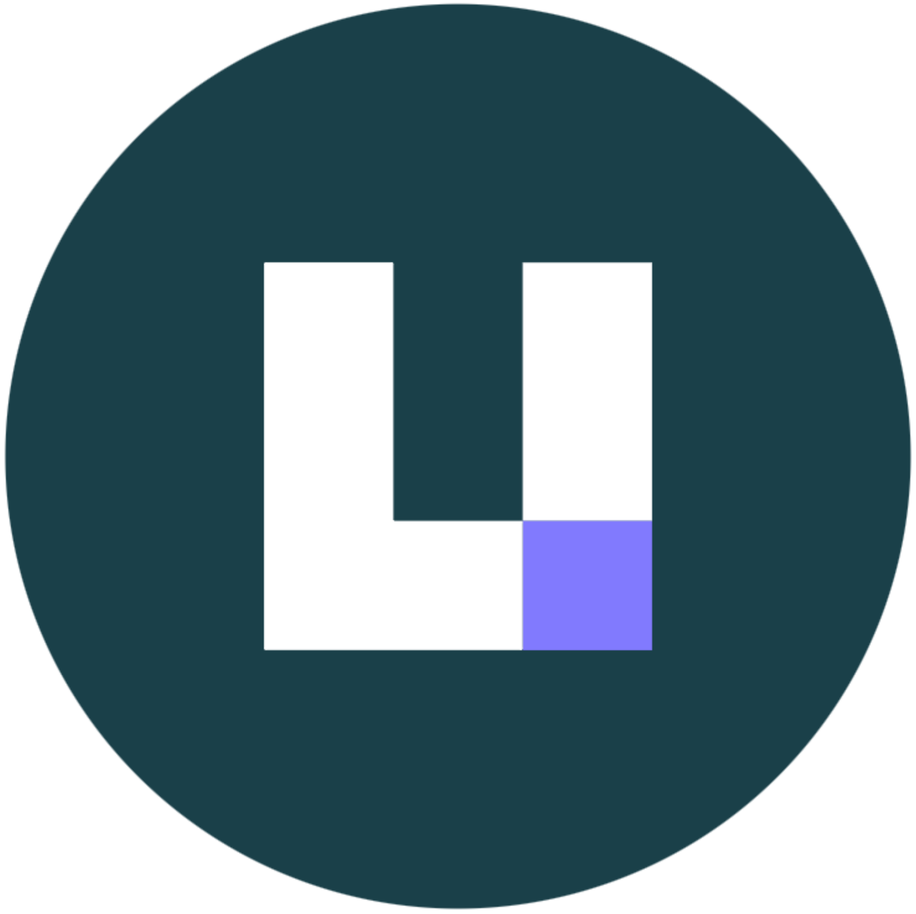
Unleash

Проект, поставляющий готовый образ бэкенд-сервиса для управления фича-флагами, а также клиенты для него на разные платформы



Возможности Unleash

- Множество встроенных стратегий активации
- Возможность написать кастомную стратегию
- Редактирование флагов через REST API



Unleash

Плюсы:

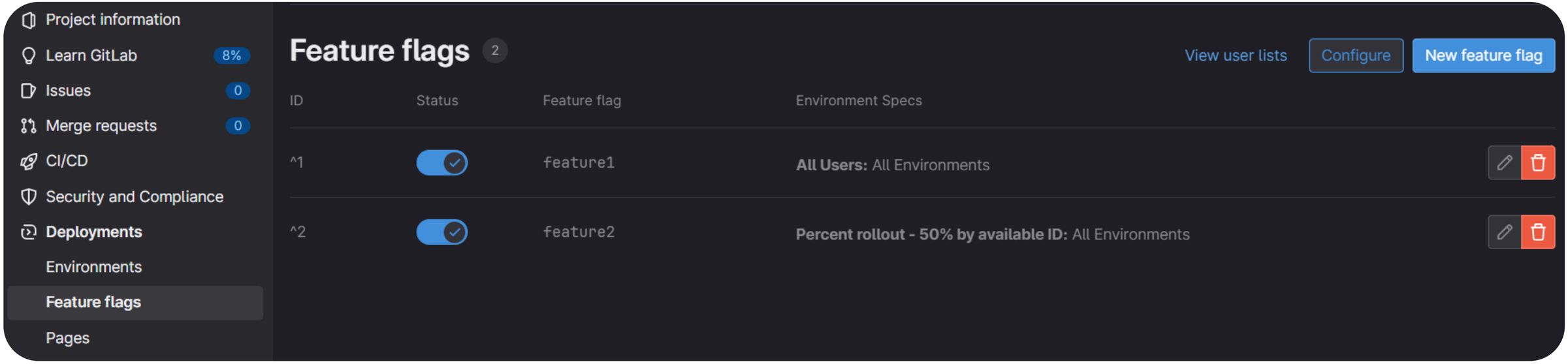
- Много возможностей для конфигурации
- Мультиплатформенность

Минусы:

- Необходимость деплоить новый сервис

Unleash + GitLab

GitLab предоставляет UI для управления флагами с бэкендом поддерживающим Unleash API.



Unleash + GitLab

Плюсы:

- Много возможностей для конфигурации
- Мультиплатформенность
- Удобный интерфейс

Минусы:

- Внешняя зависимость

FeatureManagement + ConfigMap

IFeatureManager

```
builder.Services.AddFeatureManagement();
```

```
IL code
public interface IFeatureManager
{
    IL code
    IAsyncEnumerable<string> GetFeatureNamesAsync();

    IL code
    Task<bool> IsEnabledAsync(string feature);

    IL code
    Task<bool> IsEnabledAsync<TContext>(string feature, TContext context);
}
```

IFeatureManager

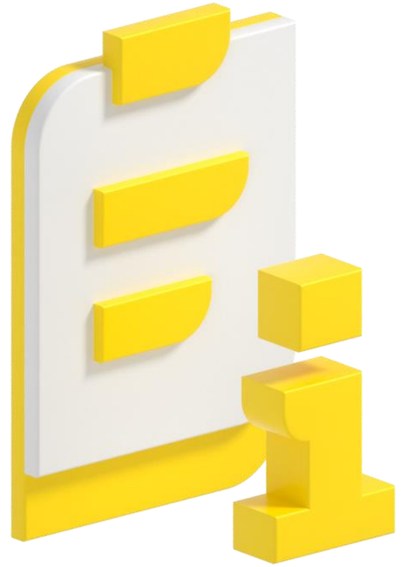
```
public async Task DoSomething()
{
    if (await _featureManager.IsEnabledAsync(FeatureFlags.SomeFeature))
    {
        // Выполняем функционал, закрытый фича флагом
    }

    // Основной функционал функции
}
```

Справка по Kubernetes

Pod – абстракция Kubernetes, в которой крутится инстанс приложения

Volume – абстракция Kubernetes, позволяющая сохранять файловое состояние подов.

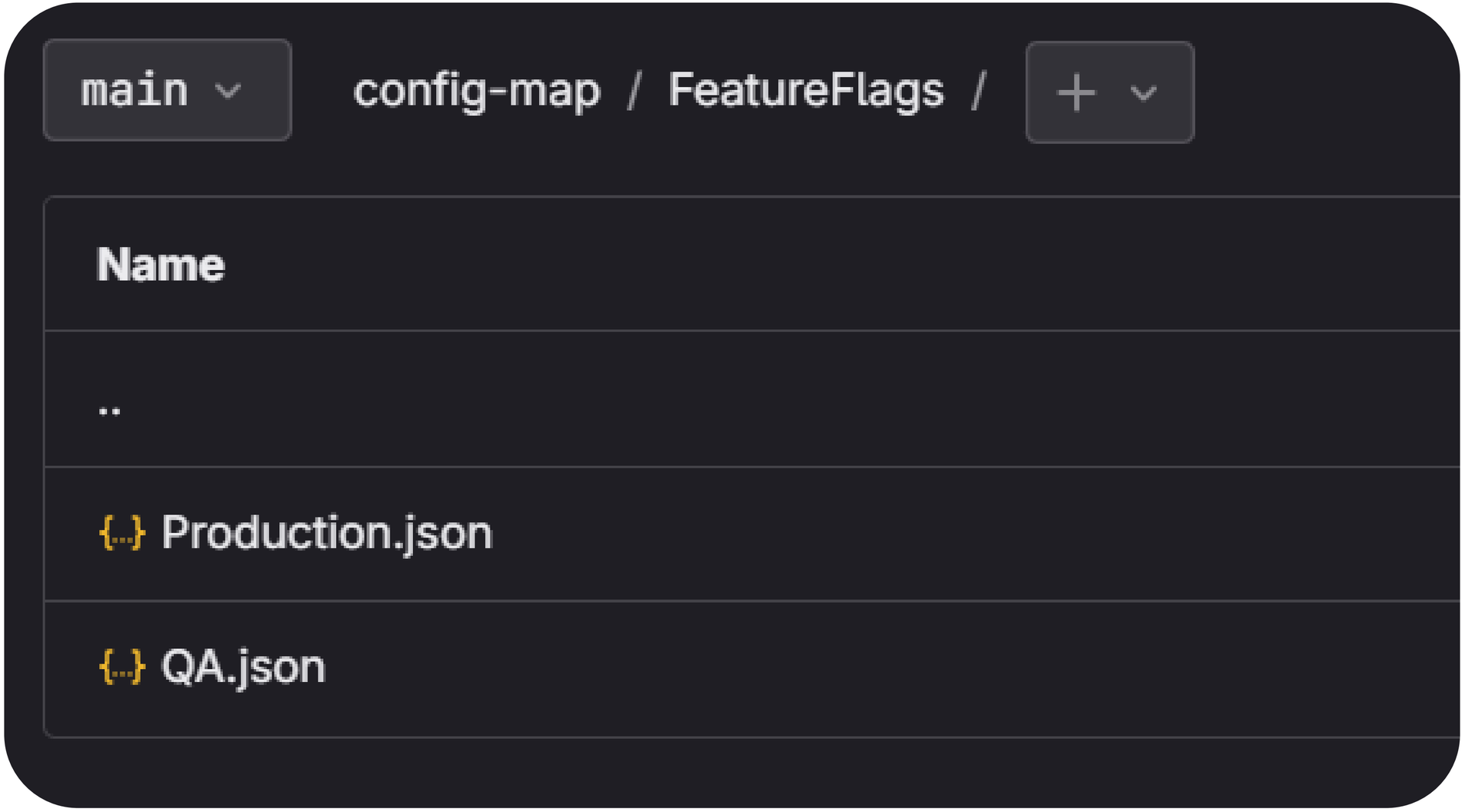
ConfigMap – разновидность Volume, позволяющая сохранять конфигурацию в формате key-pair и инъектировать её в под



ConfigMap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: feature-flags
data:
  feature-flags.json: |
    {
      "FeatureManagement": {
        "CheckPassword": false
      }
    }
```

Хранение ConfigMap



Размещение ConfigMap

```
1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    name: feature-flags
5  data:
6    feature-flags.json: |-
7    {{ .Files.Get "features.json" | indent 4 }}
```

Mount to ConfigMap

Указываем ConfigMap, к которому нужно подключиться и путь в нашем приложении, куда сохранить файл конфигурации.

ConfigMap и Pod должны находится в одном namespace.

Возможен mount к файлу или в переменные среды

```yaml
spec:
  containers:
    - name: api
      image: kawwik/example-api:latest
      ports:
        - containerPort: 4000
      volumeMounts:
        - name: config
          mountPath: "/api/config"
          readOnly: true
  volumes:
    - name: config
      configMap:
        name: feature-flags
```

Конфигурация ASP

Подключаем FeatureManagement и добавляем файл с фича-флагами.

```
builder.Services.AddFeatureManagement();
builder.Configuration.AddJsonFile(
    path: "config/feature-flags.json",
    optional: false,
    reloadOnChange: true);
```

FileProvider

FileProvider – абстракция для отслеживания изменений в файле.

Если FileProvider не передан, ASP смотрит на дату последнего изменения файла.

Файл конфигурации подключается к поду как символическая ссылка. При изменении конфига, дата изменения ссылки не меняется.

FileProvider

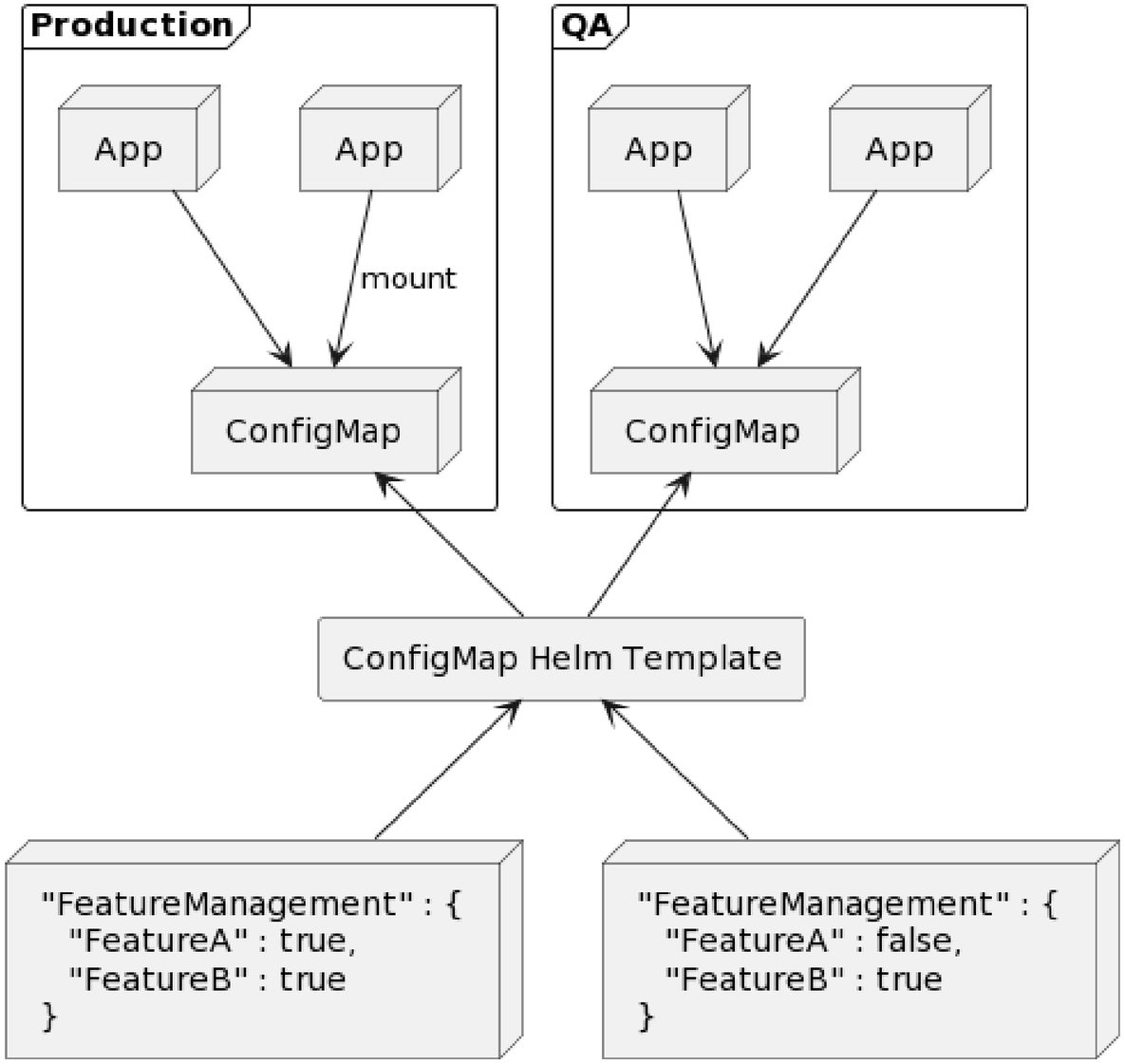
```
1 usage
public static void ConfigureFeatureFlags(this IServiceCollection services, ConfigurationManager configuration)
{
    services.AddFeatureManagement();

    var configFolderPath :string = GetAbsolutePath(relativePath: "configs");
    var fileProvider = new PhysicalFileProvider(configFolderPath)
    {
        UsePollingFileWatcher = true,
        UseActivePolling = true
    };

    configuration.AddJsonFile(
        fileProvider,
        path: "feature-flags.json",
        optional: false,
        reloadOnChange: true);
}
```

FileProvider

В старых версиях .NET нет реализации из коробки

```
builder.Configuration.AddJsonFile(
    new ConfigMapFileProvider(rootPath:"config"),
    path:"feature-flags.json",
    optional: false,
    reloadOnChange: true);
```

Резюме по реализации



Демо

Итог

- Имеем возможность включать и выключать определённый функционал без изменения кода приложения прямо в рантайме.
- Тестировщик сам по необходимости включает/выключает определённые функции.
- По запросу бизнеса можем включать определённый функционал на проде без релиза.

Good practice

- При заведении задач на новую User Story заранее закладывать под неё фича-флаг
- При добавлении временного флага сразу заводить задачу на его удаление
- Вести документацию и хранить описание каждого фича-флага

Полезные ресурсы

https://kubernetes.io/docs/concepts/configuration/configmap/ - подробнее о ConfigMap

https://github.com/microsoft/FeatureManagement-Dotnet - документация по пакету FeatureManagement для ASP

https://github.com/fbeltrao/ConfigMapFileProvider - репозиторий с реализацией FileProvider’a

https://github.com/Unleash/unleash - репозиторий проекта Unleash

https://docs.gitlab.com/ee/operations/feature_flags - про фича-флаги в GitLab

https://github.com/kawwik/Feature-Flags-Example - пример из демо (скоро)