Внедрение и эффективное использование фича-флагов в ASP.NET-приложении

Руковишников Михаил

Обо мне

Санкт-Петербург

Университет ИТМО

.NET Backend

Михаил Руковишников

План доклада

1. Контекст и проблематика
2. Фича-флаги
3. Простейшие паттерны
4. Проверка фича-флагов
5. Канареечный релиз и A/B-тесты
6. Переключение фича-флагов
7. Финальная реализация
8. Возможные проблемы и решения
9. Заключение

Контекст проекта

- Scrum
- Trunk Based Development
- Исходный код в GitLab
- Деплой в Kubernetes
- Сложные бизнес-правила

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

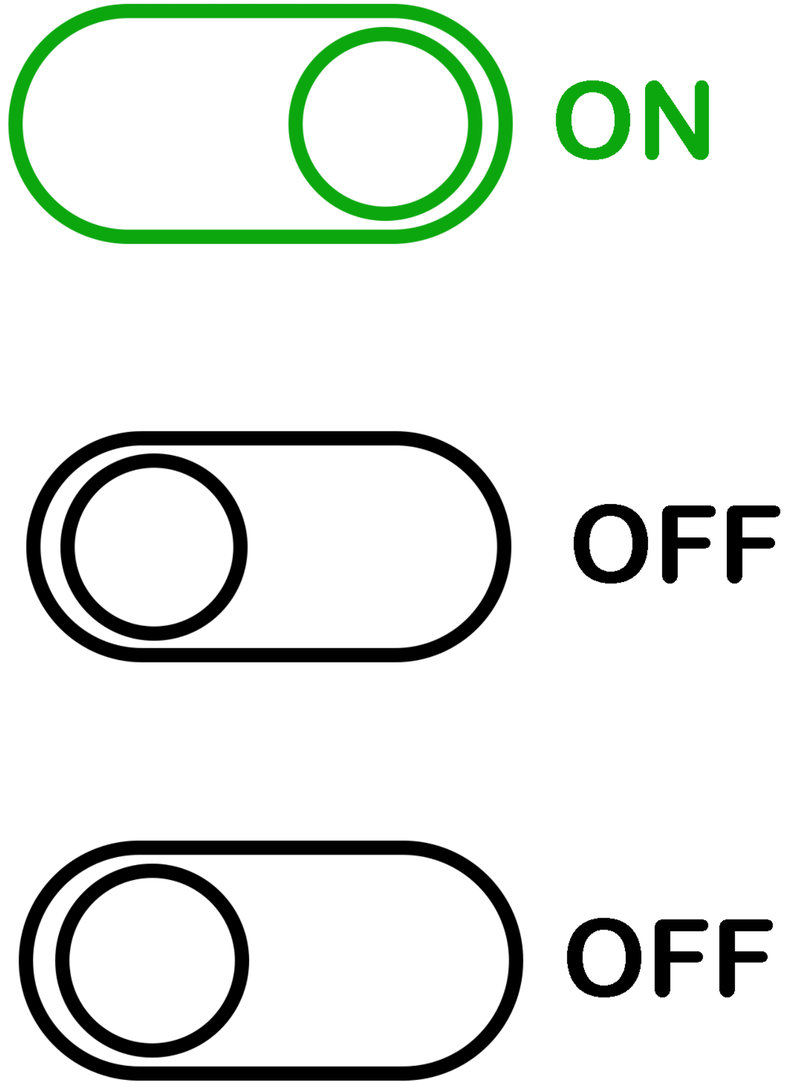
Бизнес-требования заставляют поддерживать данные в строго консистентном виде.

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:

Feature Flags

Применяются для следующих целей:

- Облегчение тестирования

Feature Flags

Применяются для следующих целей:

- Облегчение тестирования
- Разработка через TBD

Feature Flags

Применяются для следующих целей:

- Облегчение тестирования
- Разработка через TBD
- Канареечный релиз

Feature Flags

Применяются для следующих целей:

- Облегчение тестирования
- Разработка через TBD
- Канареечный релиз
- Проведение A/B тестирования

Feature Flags

Применяются для следующих целей:

- Облегчение тестирования
- Разработка через TBD
- Канареечный релиз
- Проведение A/B тестирования
- Включение и выключение функционала по решению бизнеса

Простейшие паттерны

Feature Flags

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

    // Основной функционал метода
}
```

Feature Flags

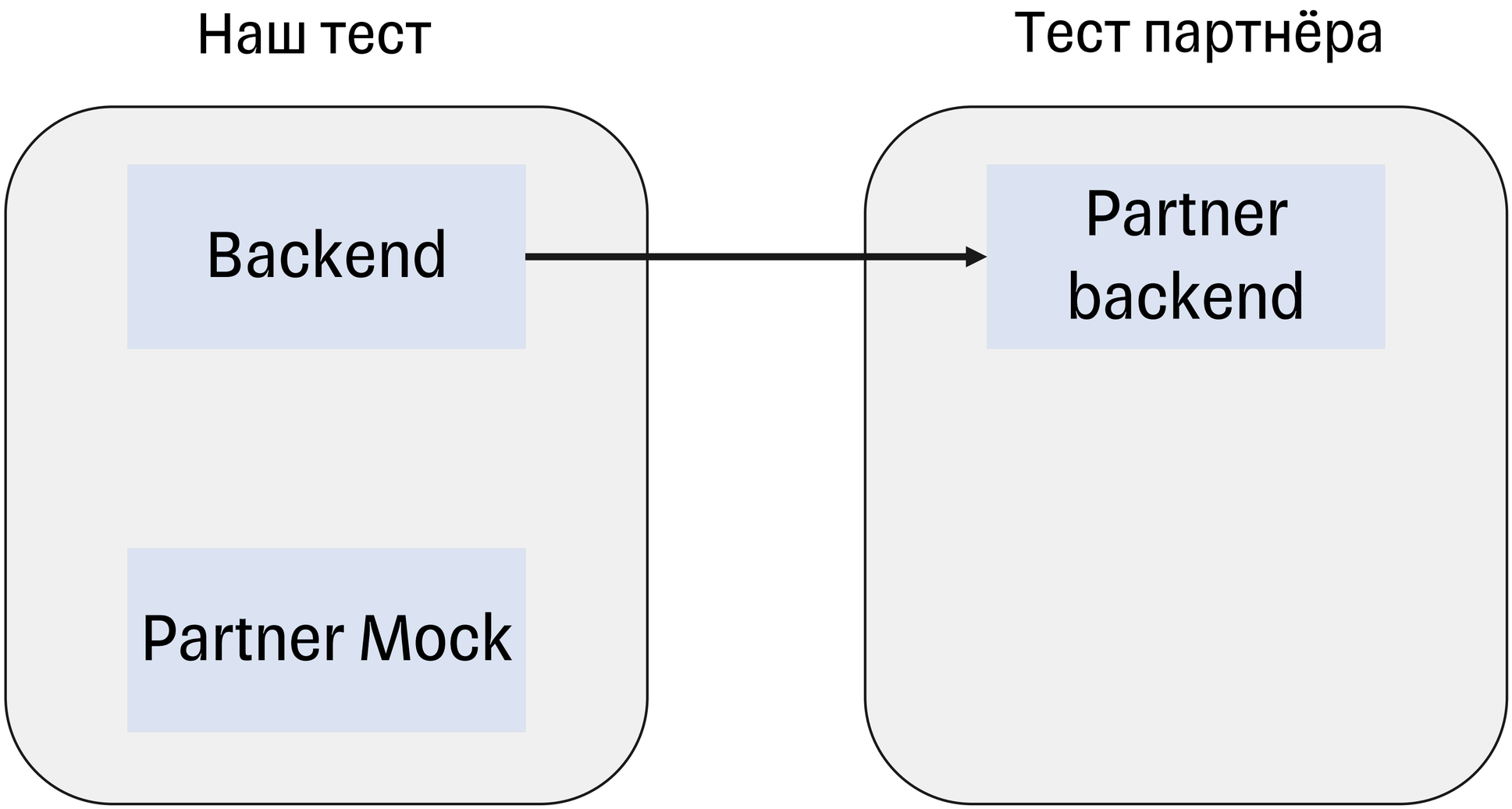
```
public void DoSomething()
{
    if (flag)
        DoSomethingNew();
    else
        DoSomethingOld();
}

public void DoSomethingOld()
{
    // Выполняем старую версию функции
}

public void DoSomethingNew()
{
    // Выполняем новую версию функции
}
```

Feature Flags Via DI

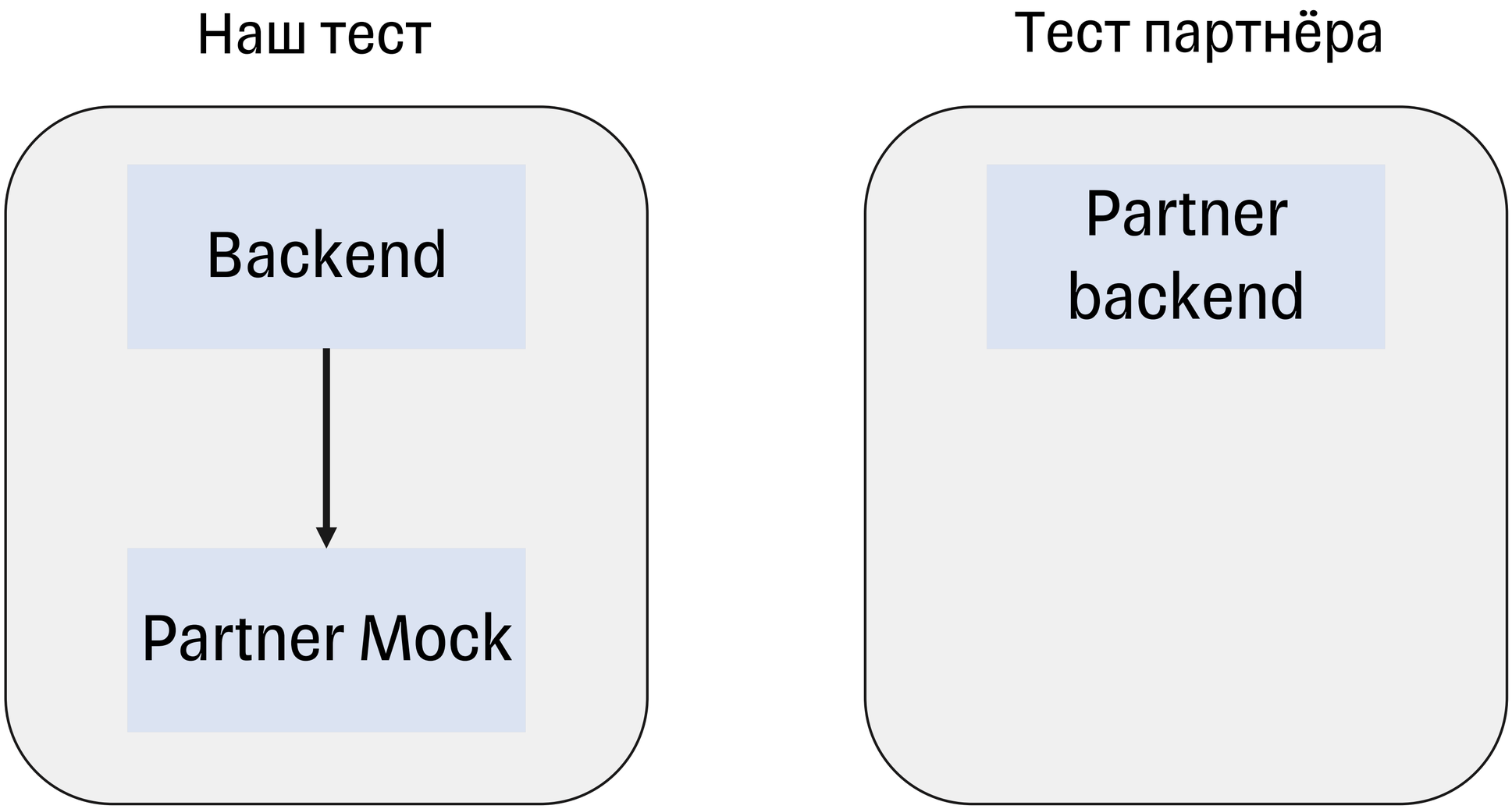
```
public interface IService
{
    void DoSomething();
}

public class Service : IService
{
    public void DoSomething()
    {
    }
}

public class NewService : IService
{
    public void DoSomething()
    {
    }
}
```

```
public class ServiceProxy : IService
{
    private readonly bool useNewService = true;

    private readonly Service _service;
    private readonly NewService _newService;

    public ServiceProxy(Service service, NewService newService)
    {
        _service = service;
        _newService = newService;
    }

    public void DoSomething()
    {
        GetServiceImplementation().DoSomething();
    }

    private IService GetServiceImplementation()
    {
        return useNewService ? _newService : _service;

    }
}
```

Feature Flags Proxy



Feature Flags Proxy



Feature Flags Proxy

```
public interface IProxiedClientSettings
{
    public string BaseUri { get; set; }
    public string FakeBaseUri { get; set; }
}
```

Feature Flags Proxy

```
protected override async Task<HttpResponseMessage> SendAsync(
    HttpRequestMessage request,
    CancellationToken cancellationToken)
{
    if (flag)
    {
        var uri = request.RequestUri!.AbsoluteUri;
        var fakeUri = uri.Replace(
            _proxiedClientSettings.BaseUri,
            _proxiedClientSettings.FakeBaseUri);

        request.RequestUri = new Uri(fakeUri);
    }

    return await base.SendAsync(request, cancellationToken);
}
```

Реализация

Реализация фича-флагов

Как проверять?

Где хранить?

Как изменять?

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(IOptionsMonitor<ExampleClassSettings> settings)
    {
        _settings = settings.CurrentValue;
    }

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

        // Основной функционал метода
    }
}
```

Стратегии активации (FeatureFilter)

Логика, определяющая, должен ли флаг быть активирован в текущем контексте.

Стратегии активации (FeatureFilter)

Логика, определяющая, должен ли флаг быть активирован в текущем контексте.

- Percentage Rollout

Стратегии активации (FeatureFilter)

Логика, определяющая, должен ли флаг быть активирован в текущем контексте.

- Percentage Rollout
- Time window

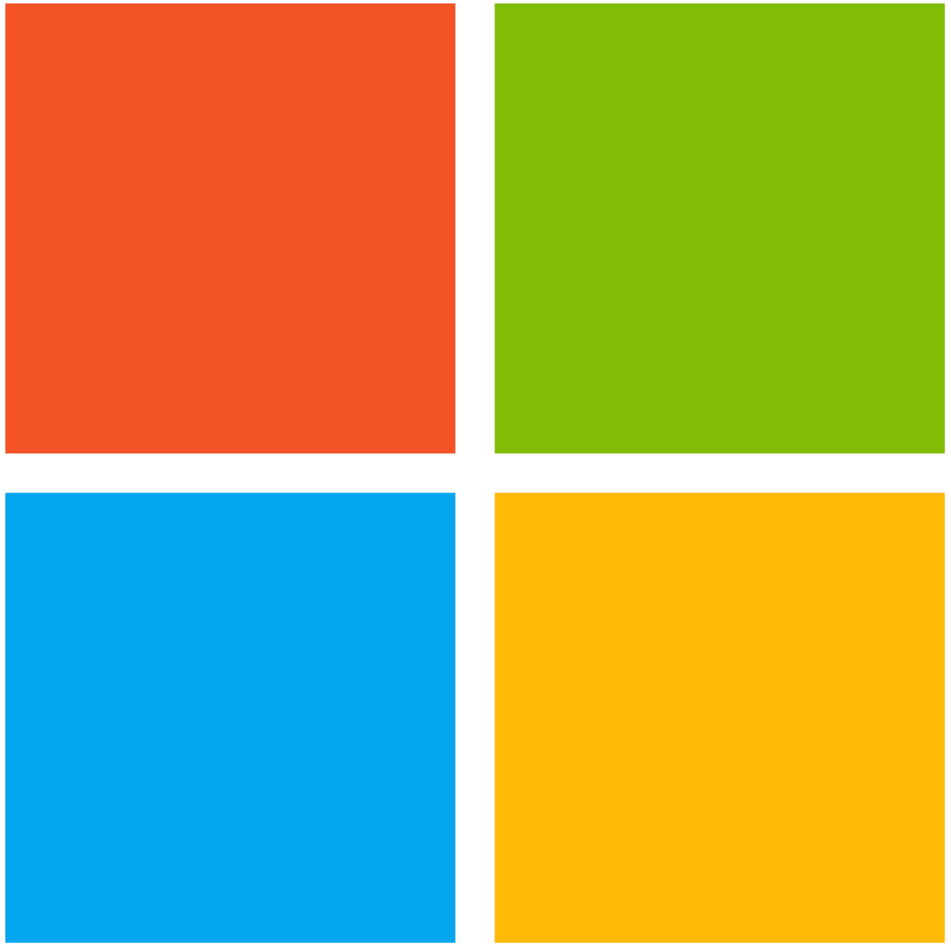
Стратегии активации (FeatureFilter)

Логика, определяющая, должен ли флаг быть активирован в текущем контексте.

- Percentage Rollout
- Time window
- Конкретный пользователь

FeatureManagement

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



Возможности библиотеки

- Возможность использовать различные стратегии активации
- Возможность реализовать источник флагов самостоятельно
- Возможность кеширования состояния флага в рамках одного и нескольких запросов

FeatureManagement

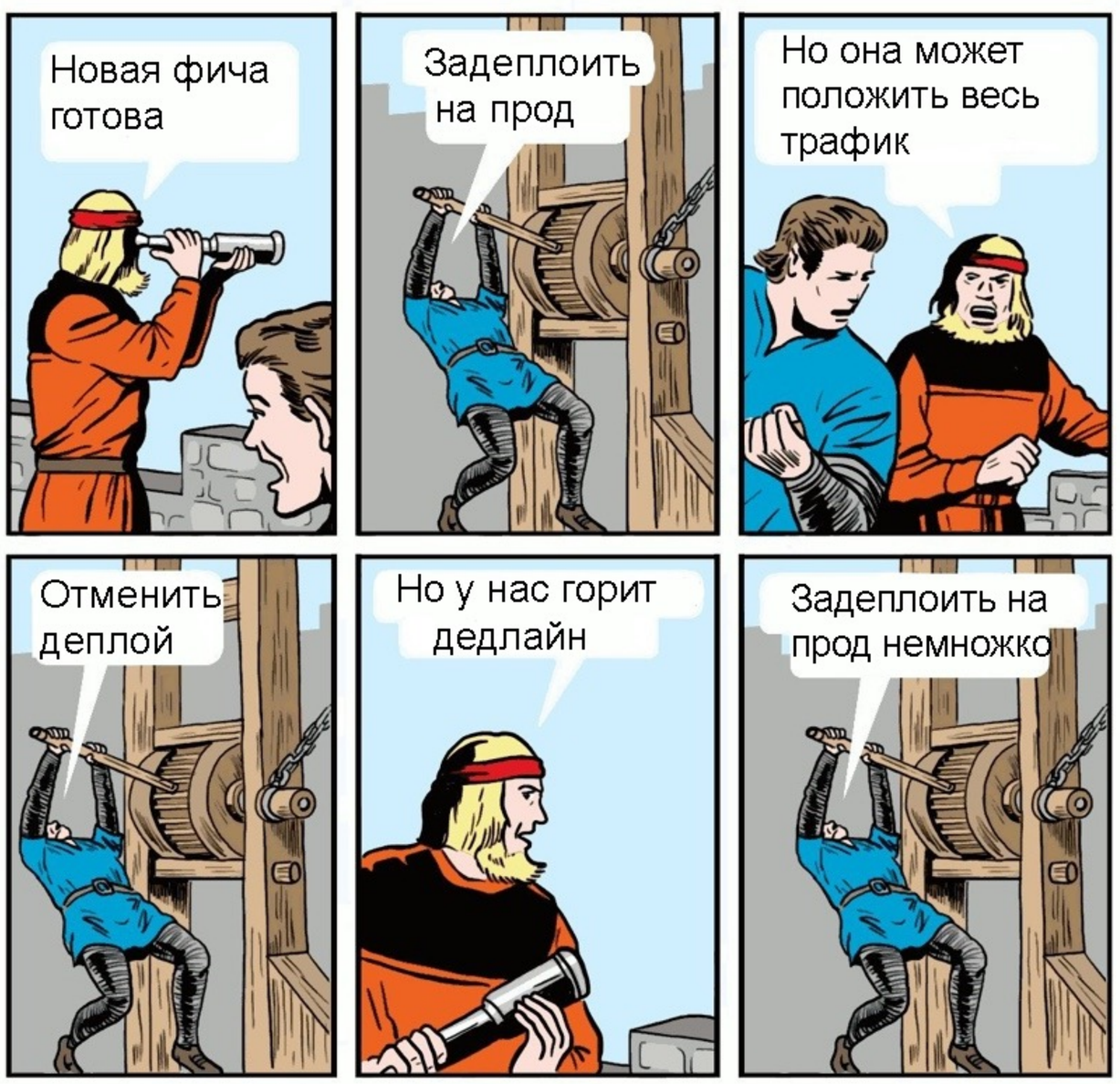
```
builder.Services.AddFeatureManagement();
```

```
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);
}
```

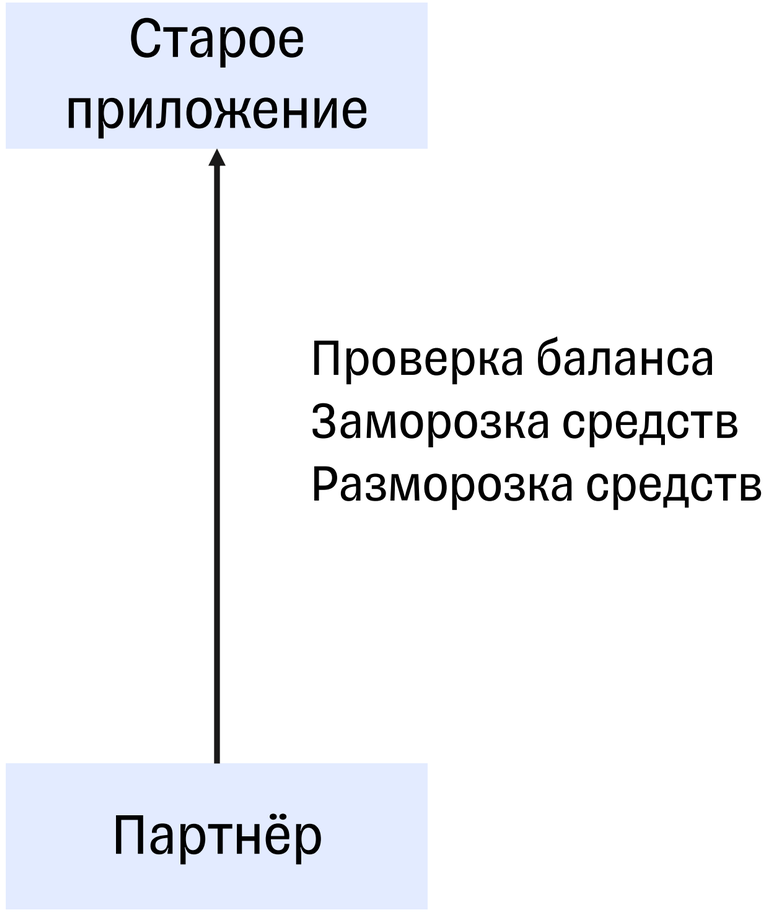
FeatureManagement

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

    // Основной функционал метода
}
```

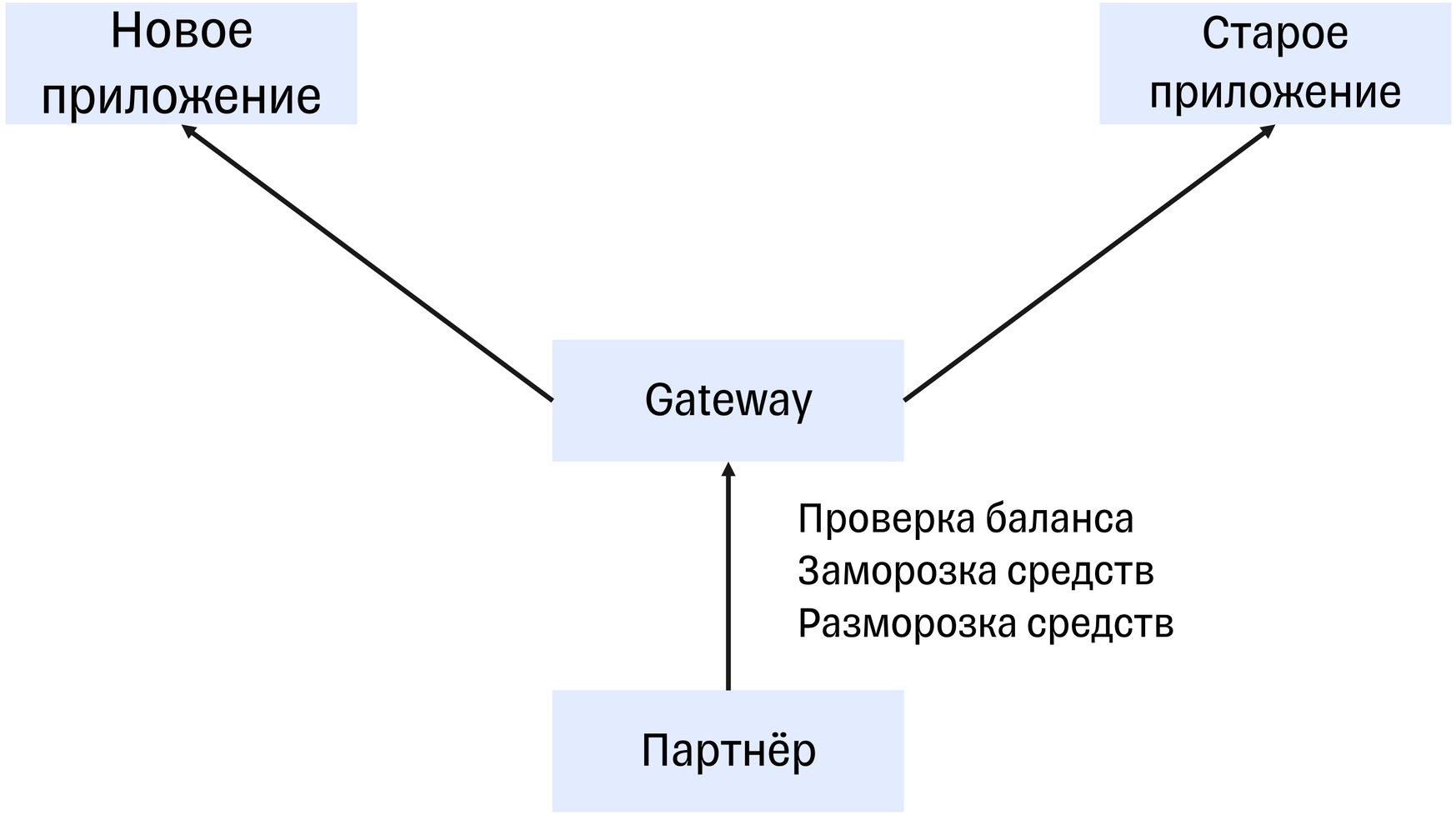
Канареечный релиз



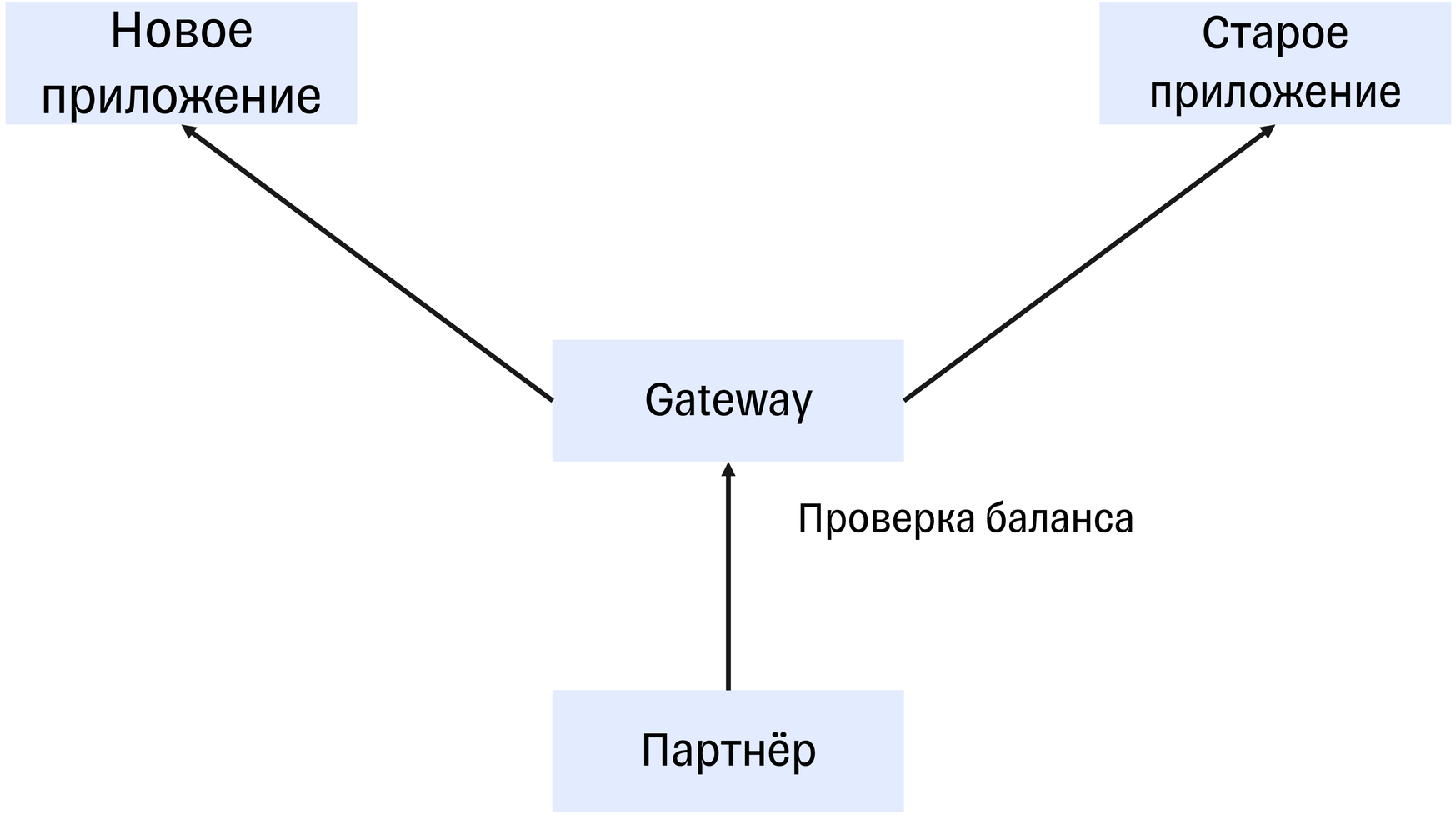
Канареечный релиз



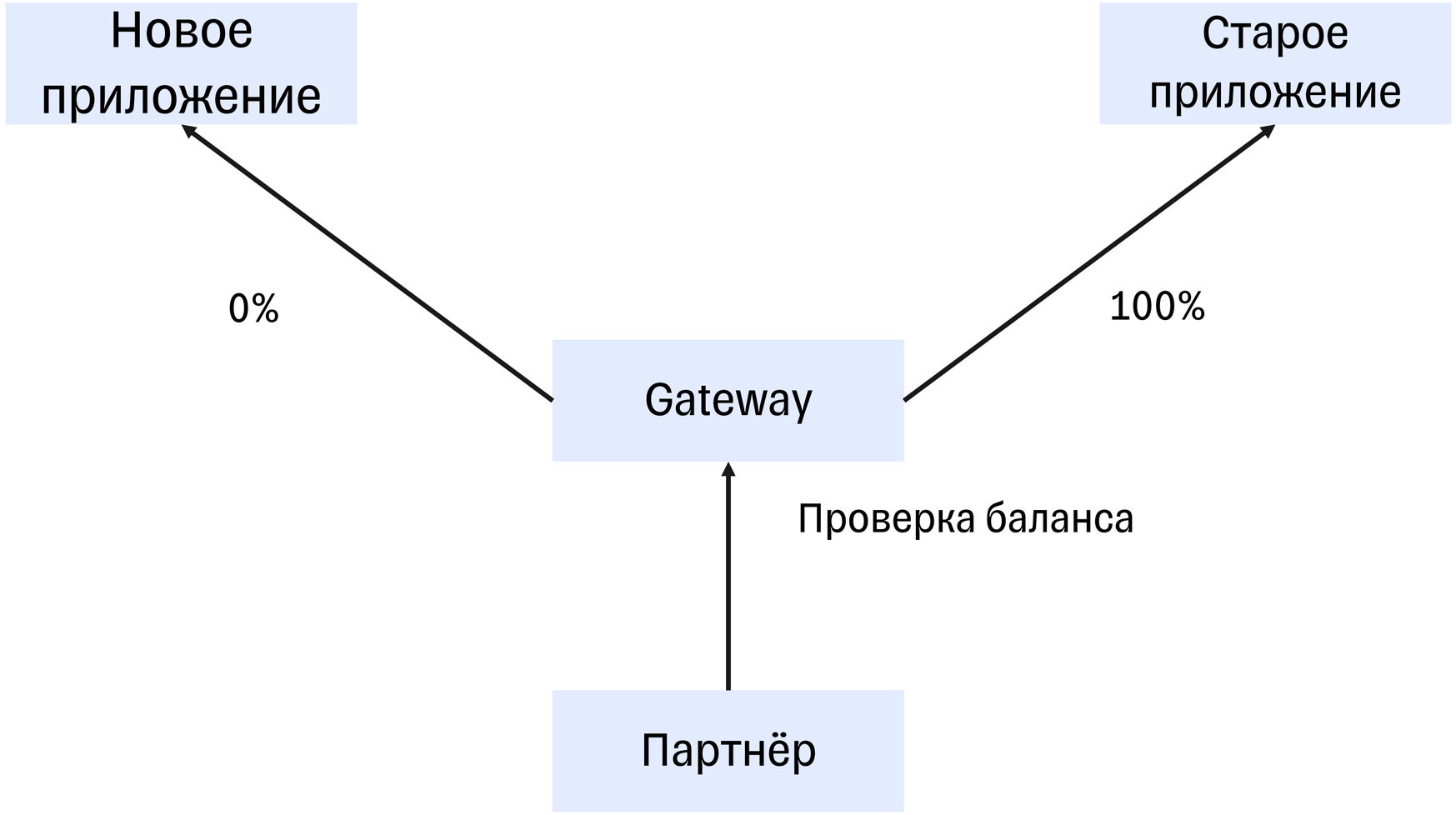
Канареечный релиз



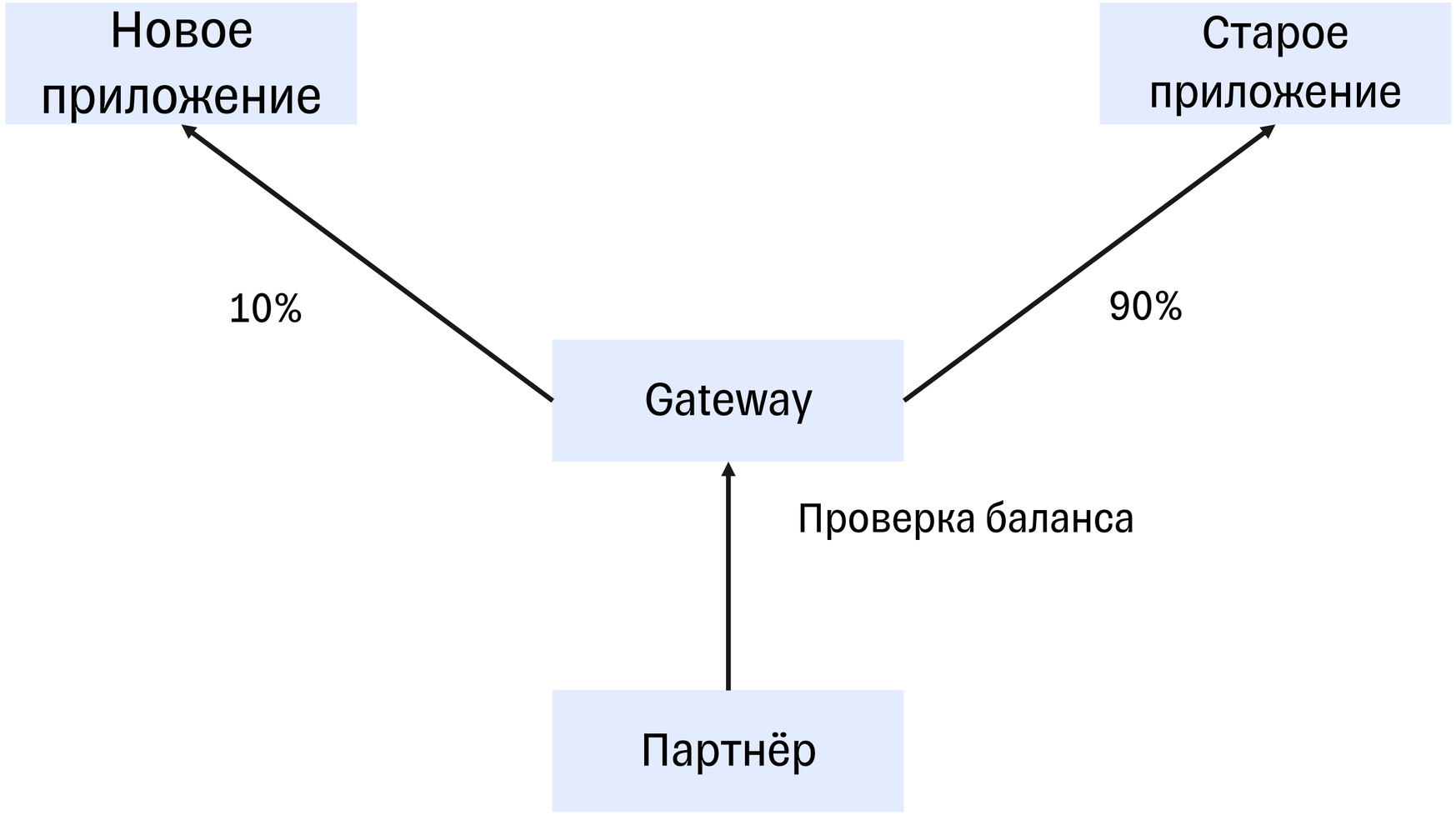
Канареечный релиз



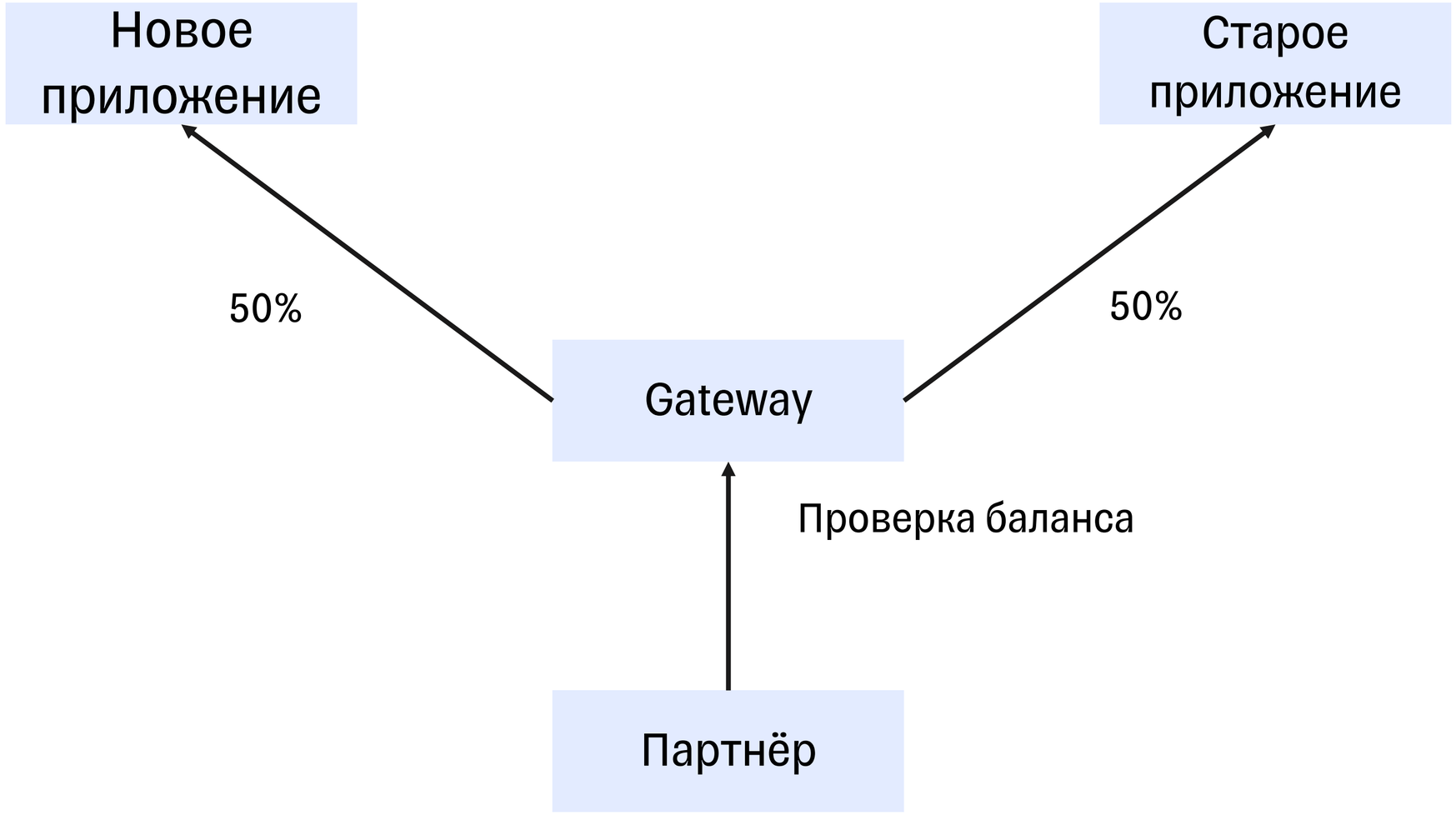
Канареечный релиз



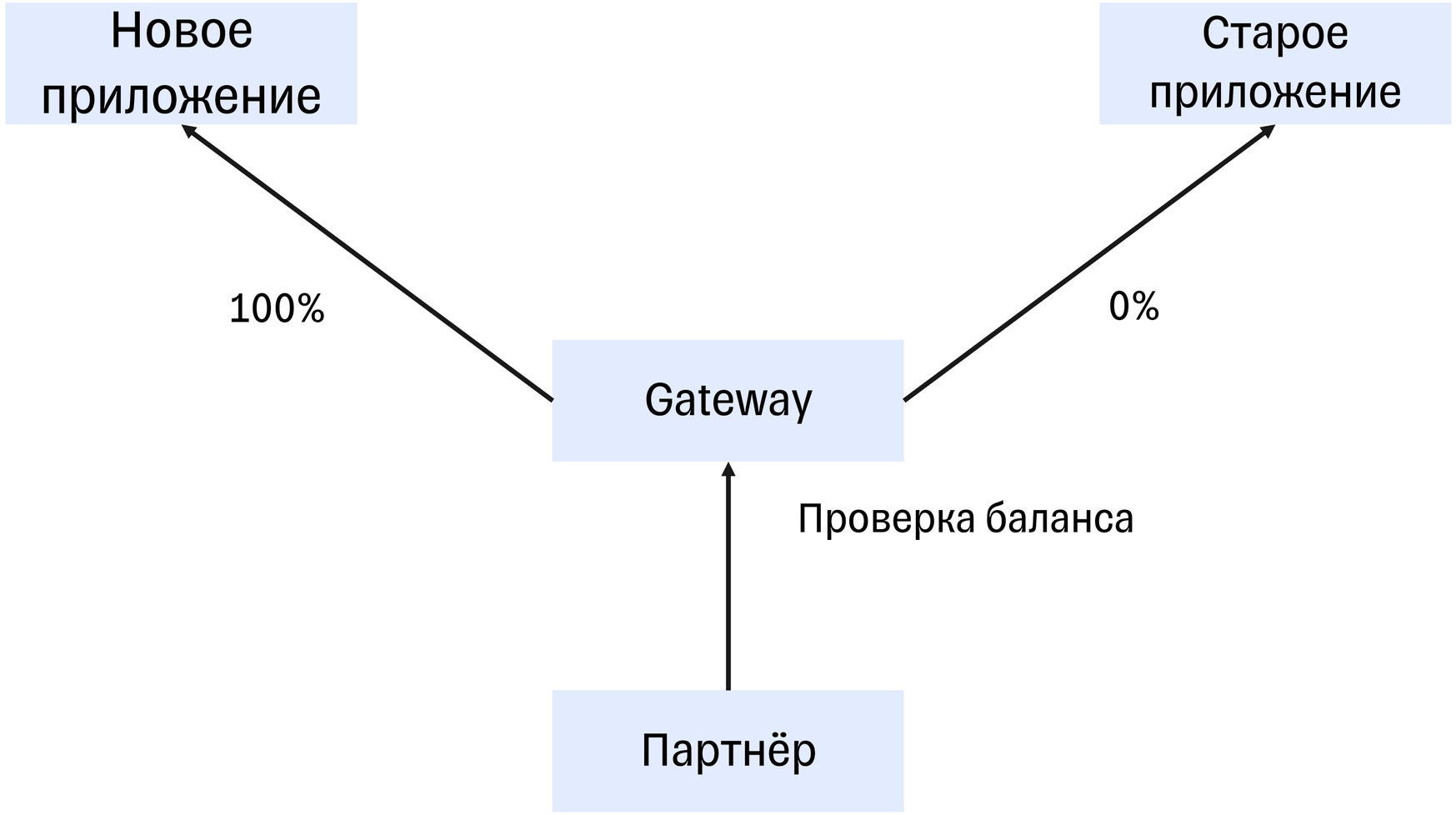
Канареечный релиз



Канареечный релиз



Канареечный релиз

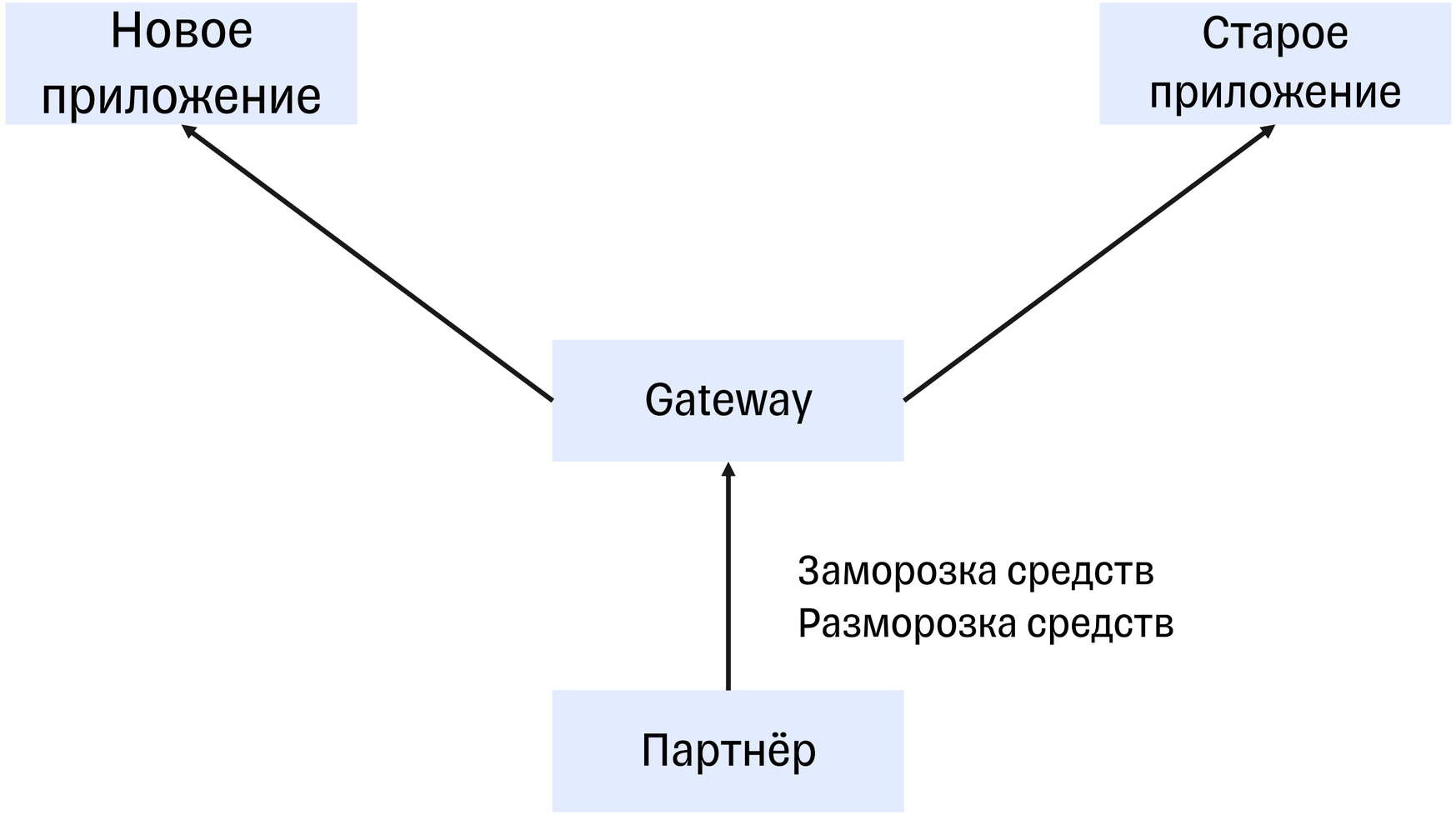


FeatureManagement

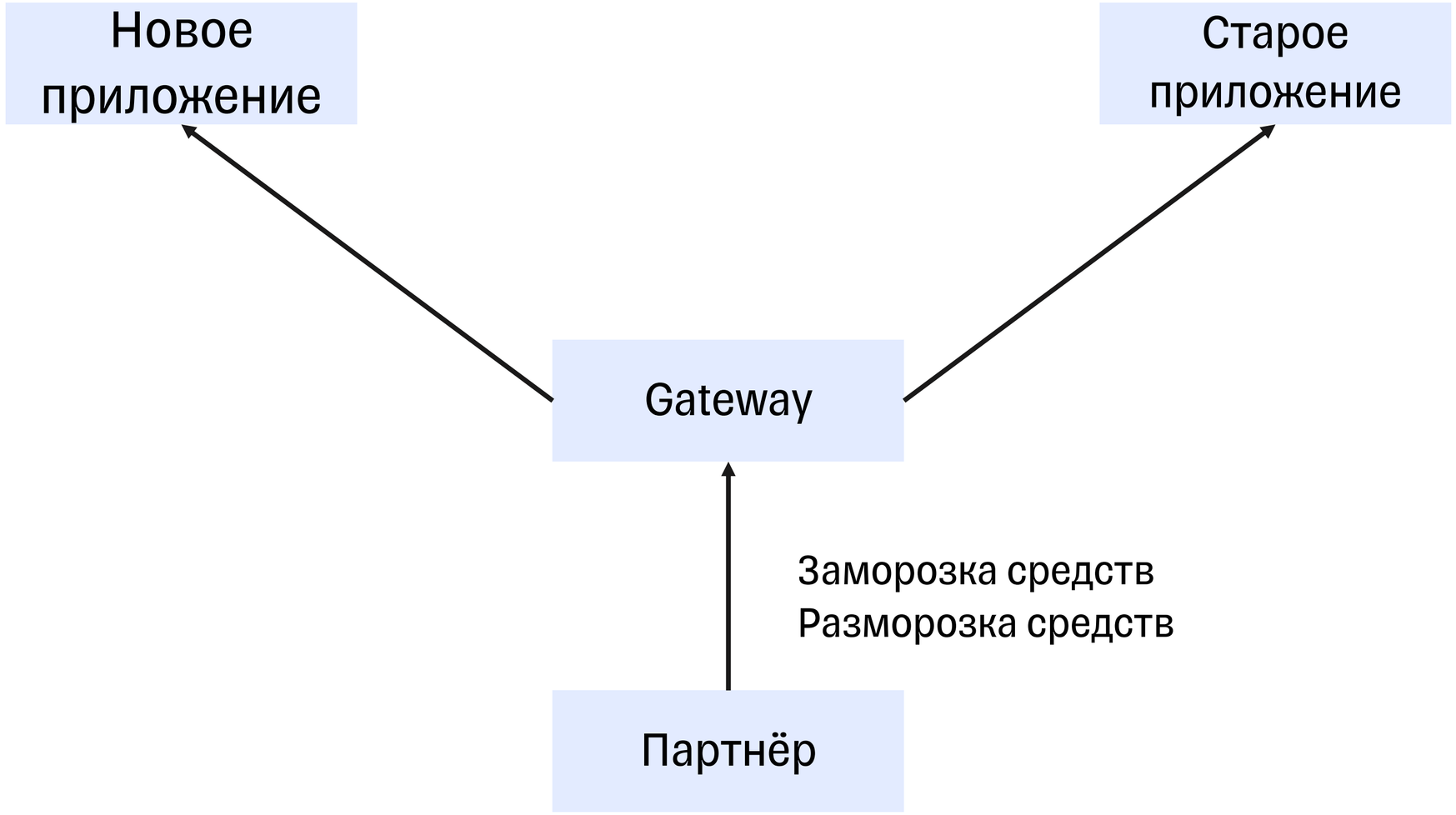
```
builder.Services// IServiceCollection
    .AddFeatureManagement()
    .AddFeatureFilter<PercentageFilter>();
```

```
"FeatureManagement": {
  "FeatureA": true,
  "FeatureB": {
    "EnabledFor": [
      {
        "Name": "Percentage",
        "Parameters": {
          "Value": 50
        }
      }
    ]
  }
}
```

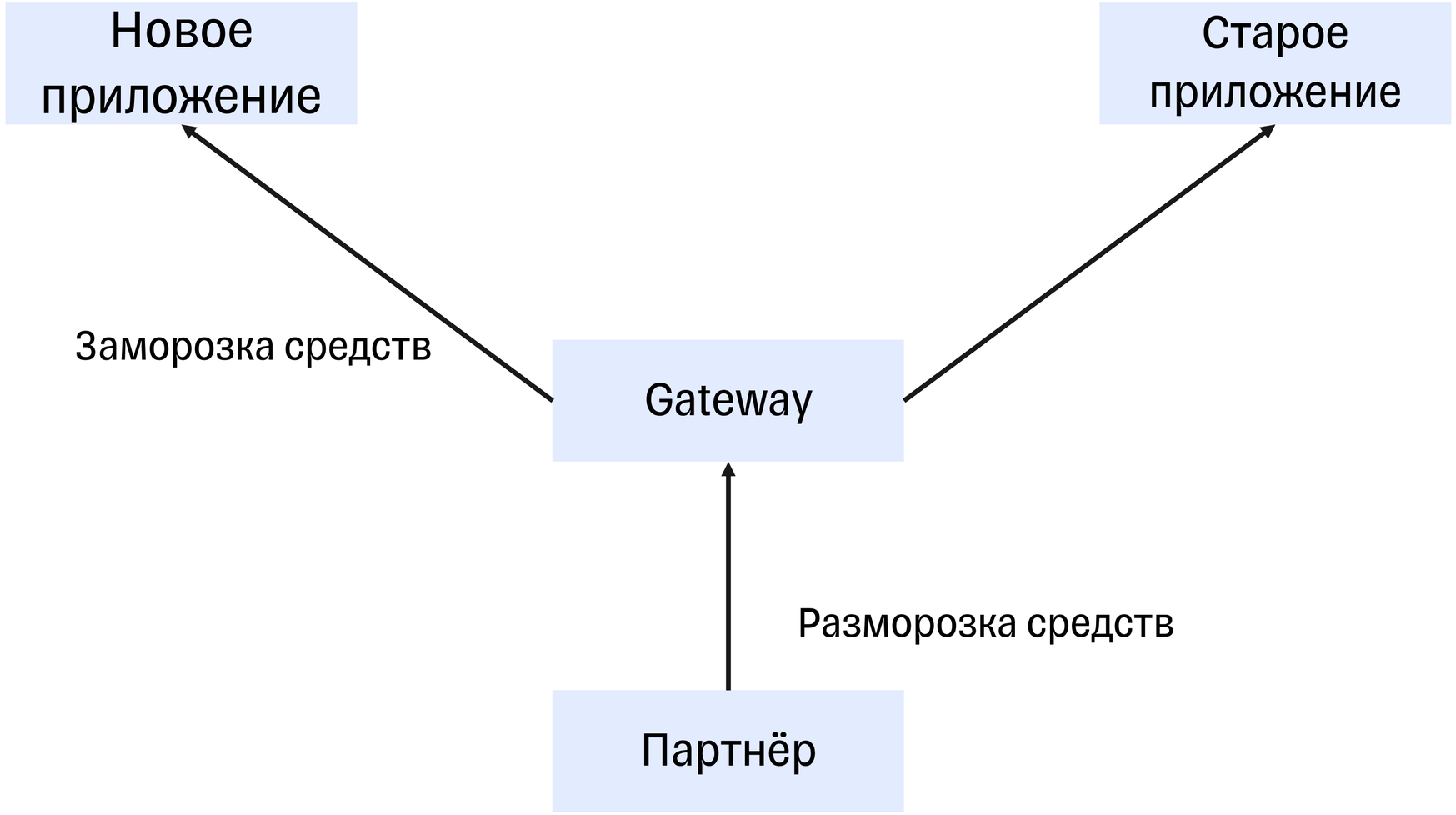
Канареечный релиз



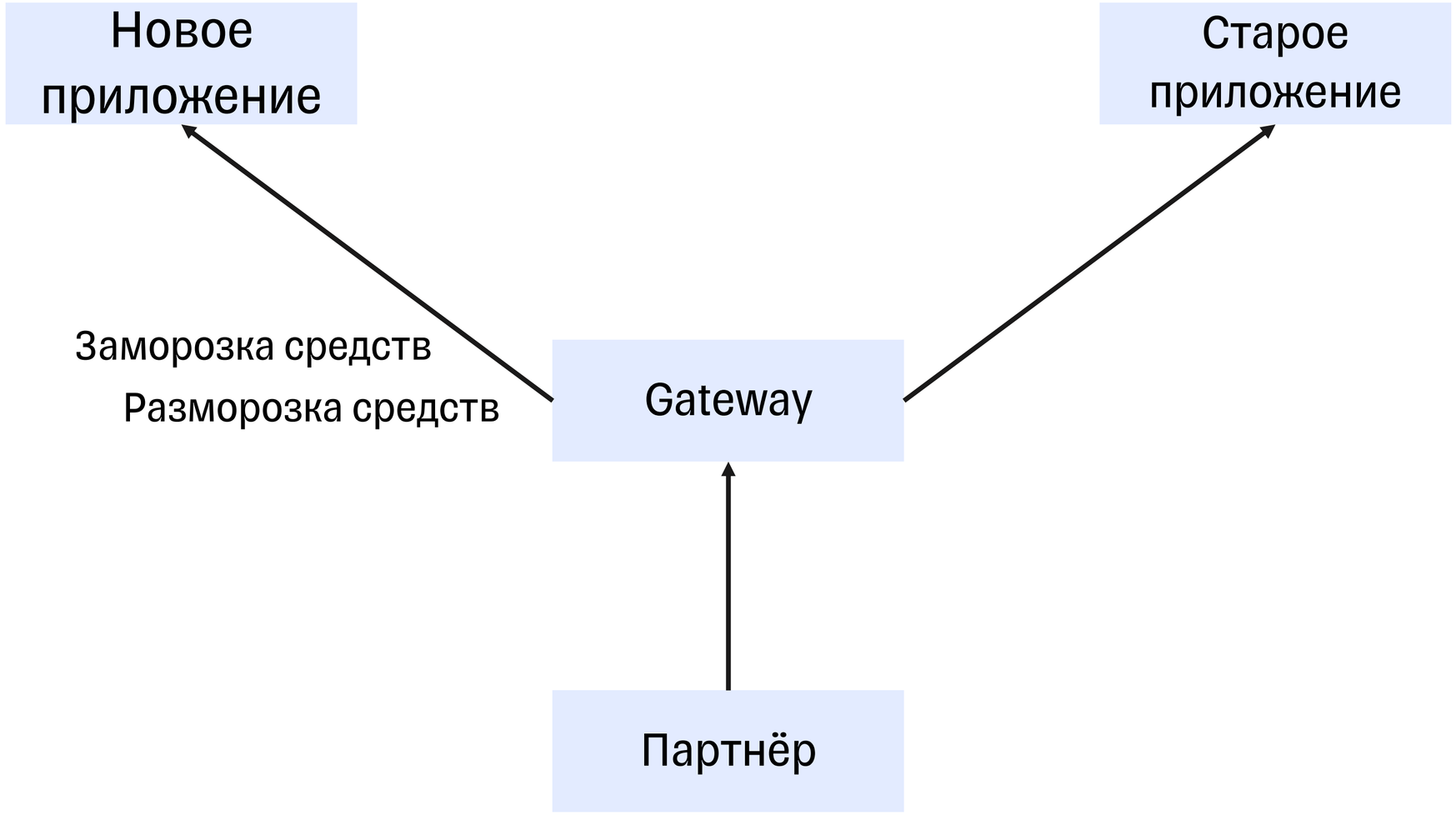
Канареечный релиз



Канареечный релиз



Канареечный релиз



Канареечный релиз

768155475

Канареечный релиз

768155475

Канареечный релиз

```
public record InnContext(string Inn);

public class InnFeatureFilterParameters
{
    public int LastDigitLessOrEqual { get; set; }
}

[FilterAlias("Inn")]
public class InnFeatureFilter : IContextualFeatureFilter<InnContext>
{
    public Task<bool> EvaluateAsync(
        FeatureFilterEvaluationContext featureFilterContext,
        InnContext appContext)
    {
        var parameters = featureFilterContext.Parameters.Get<InnFeatureFilterParameters>()
                         ?? throw new ArgumentException();

        var result = (appContext.Inn.Last() - '0') % 10 <= parameters.LastDigitLessOrEqual;

        return Task.FromResult(result);
    }
}
```

Канареечный релиз

```
"FeatureManagement": {
  "FundsHold": {
    "EnabledFor": [
      {
        "Name": "Inn",
        "Parameters": {
          "LastDigitLessOrEqual": 4
        }
      }
    ]
  }
}
```

A/B тесты

Метод сравнения двух вариантов реализации той или иной фичи, при котором часть клиентов получает один вариант, а часть – другой, после чего сравниваются их бизнес-метрики.

ISessionManager

```
public interface ISessionManager
{
  Task SetAsync(string featureName, bool enabled);

  Task<bool?> GetAsync(string featureName);
}
```

ISessionManager

```
public Task SetAsync(string featureName, bool enabled)
{
    if (featureName is not FeatureFlags.FundsHold)
        return Task.CompletedTask;

    var httpContext = httpContextAccessor.HttpContext;

    if (httpContext is null)
        throw new InvalidOperationException();

    httpContext.Response.Cookies.Append($"feature_{featureName.ToLower()}", enabled.ToString());

    return Task.CompletedTask;
}
```

ISessionManager

```
public Task<bool?> GetAsync(string featureName)
{
    var httpContext = httpContextAccessor.HttpContext;

    if (httpContext is null)
        throw new InvalidOperationException();

    if (!httpContext.Request.Cookies.TryGetValue($"feature_{featureName.ToLower()}", out var enabledString))
        return Task.FromResult<bool?>(null);

    var result = bool.TryParse(enabledString, out var enabled) ? enabled : null as bool?;

    return Task.FromResult(result);
}
```

Переключение фича-флагов

Как хотелось бы?

Как хотелось бы?

Быстро

Как хотелось бы?

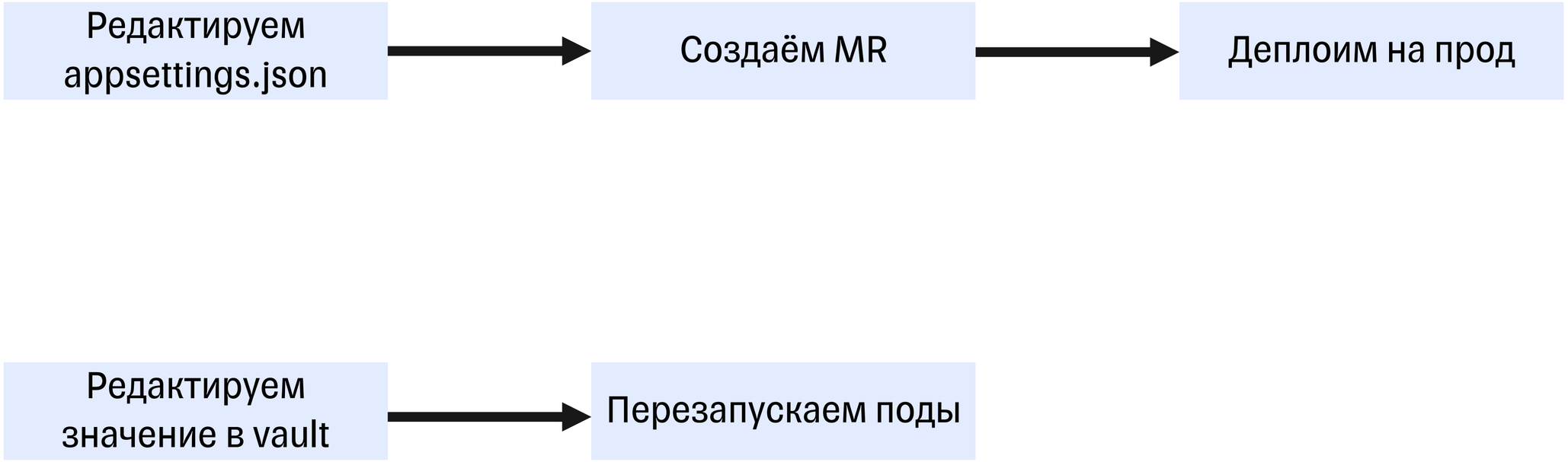
Быстро

Просто

Редеплой или перезапуск приложения

Редеплой или перезапуск приложения

Редеплой или перезапуск приложения



ConfigMap

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

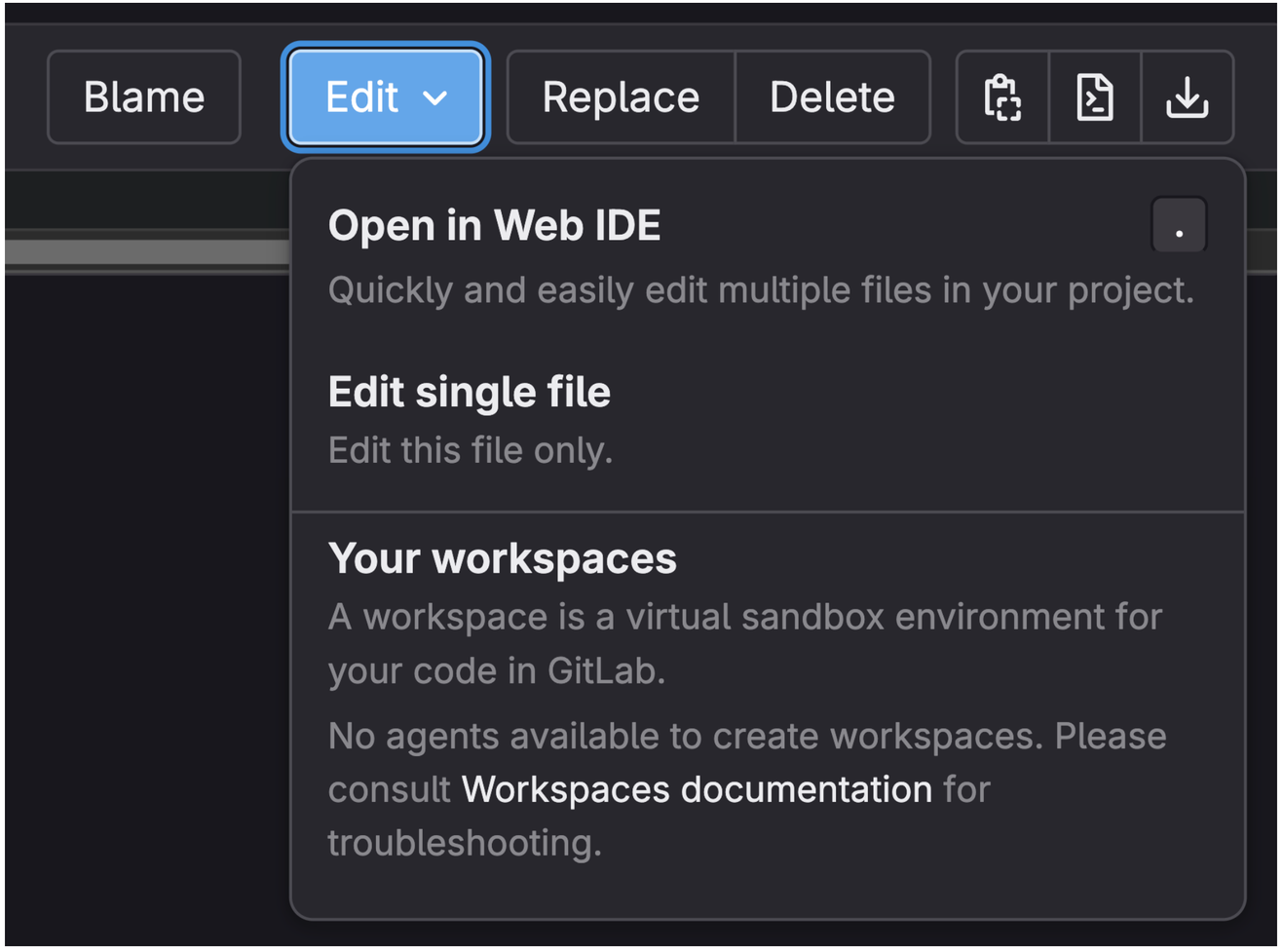
Конфигурация может быть передана из ConfigMap как переменные среды, аргументов командной строки или как файлы.

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.

ConfigMap


Blame
Edit
Replace
Delete
Open in Web IDE
.
Quickly and easily edit multiple files in your project.
Edit single file
Edit this file only.
Your workspaces
A workspace is a virtual sandbox environment for your code in GitLab.
No agents available to create workspaces. Please consult Workspaces documentation for troubleshooting.

Хранение ConfigMap в GitLab

- Историчность
- Аудит
- Контроль доступа
- Удобство редактирования

ConfigMap

Плюсы:

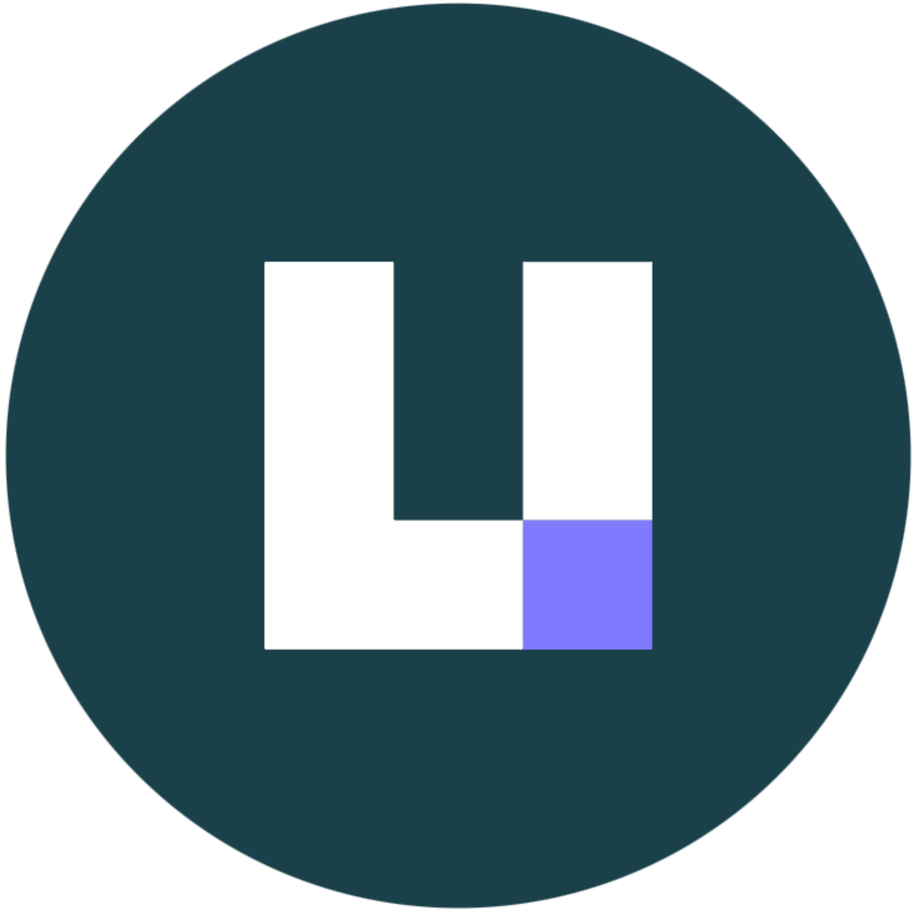
- Лёгкость редактирования
- Аудит и контроль доступа из коробки

Минусы:

- Требует значительной начальной конфигурации

Unleash

Проект, поставляющий готовый образ бэкенд-сервиса для управления фича-флагами через REST API, а также клиент для него на разные платформы



Unleash

Плюсы:

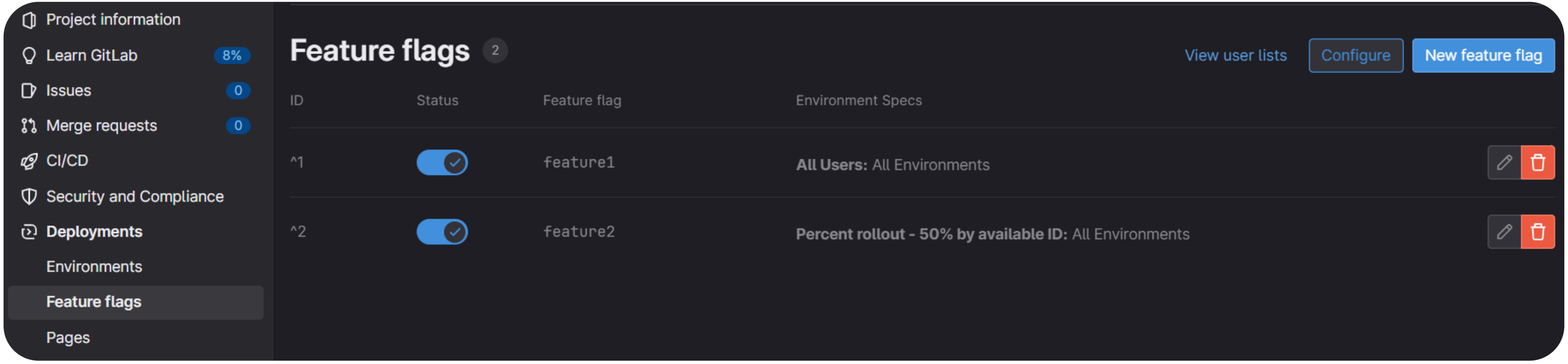
- Мультиплатформенность

Минусы:

- Необходимость деплоить новый сервис
- Управление флагами через HTTP-вызовы

Unleash + GitLab

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



Unleash + GitLab

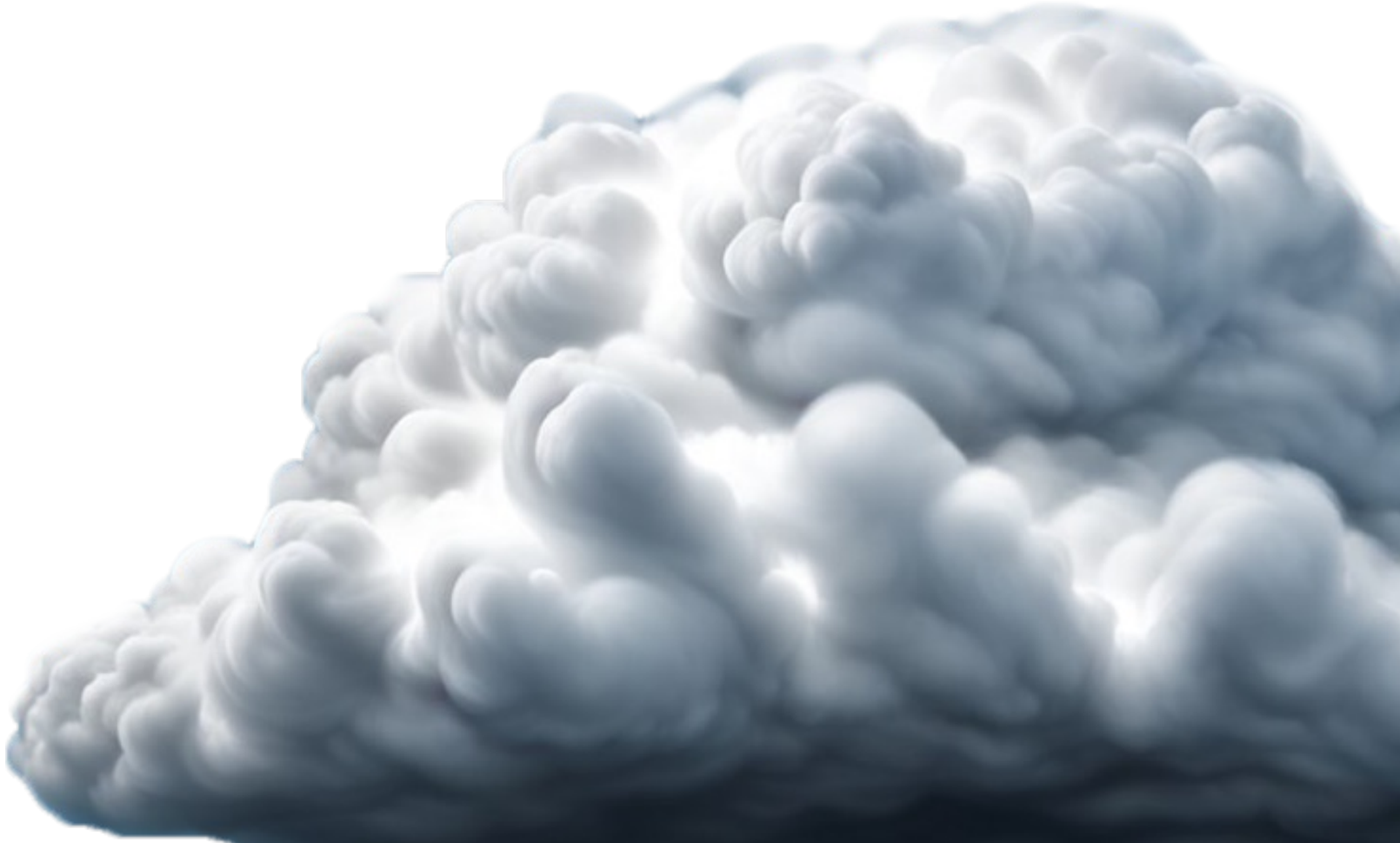
Плюсы:

- Мультиплатформенность
- Удобный интерфейс

Минусы:

- Внешняя зависимость

Облачные сервисы



DIY

FeatureManagement + ConfigMap

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

Mount – операция подключения вольюма к поду

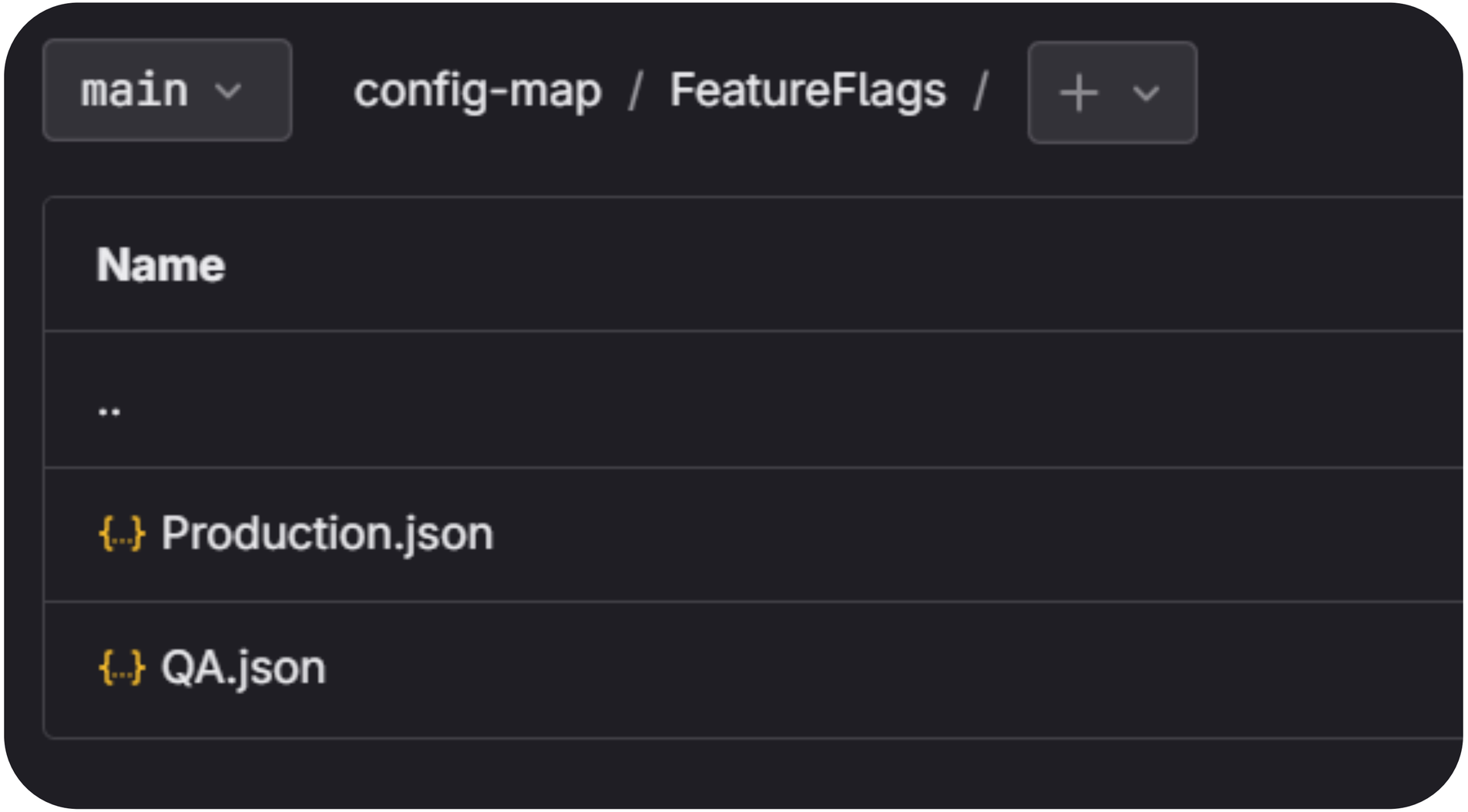
ConfigMap

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

Размещение ConfigMap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: feature-flags
data:
  feature-flags.json: |-
  {{ .Files.Get "features.json" | indent 4 }}
```

Хранение ConfigMap



Mount to ConfigMap

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

```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 приложения

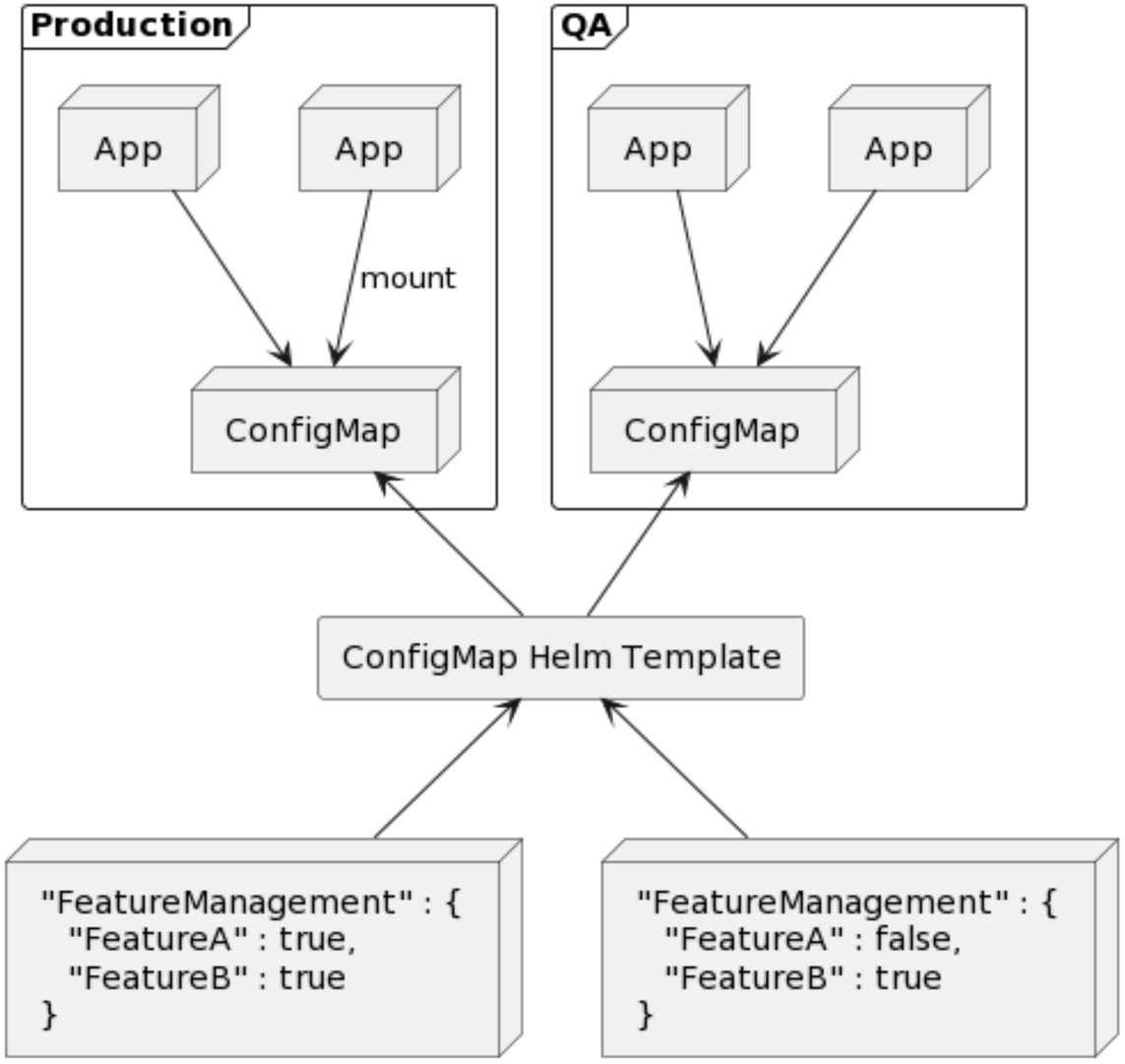
Подключаем FeatureManagement и добавляем файл с фича-флагами.

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

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);
}
```

Резюме по реализации

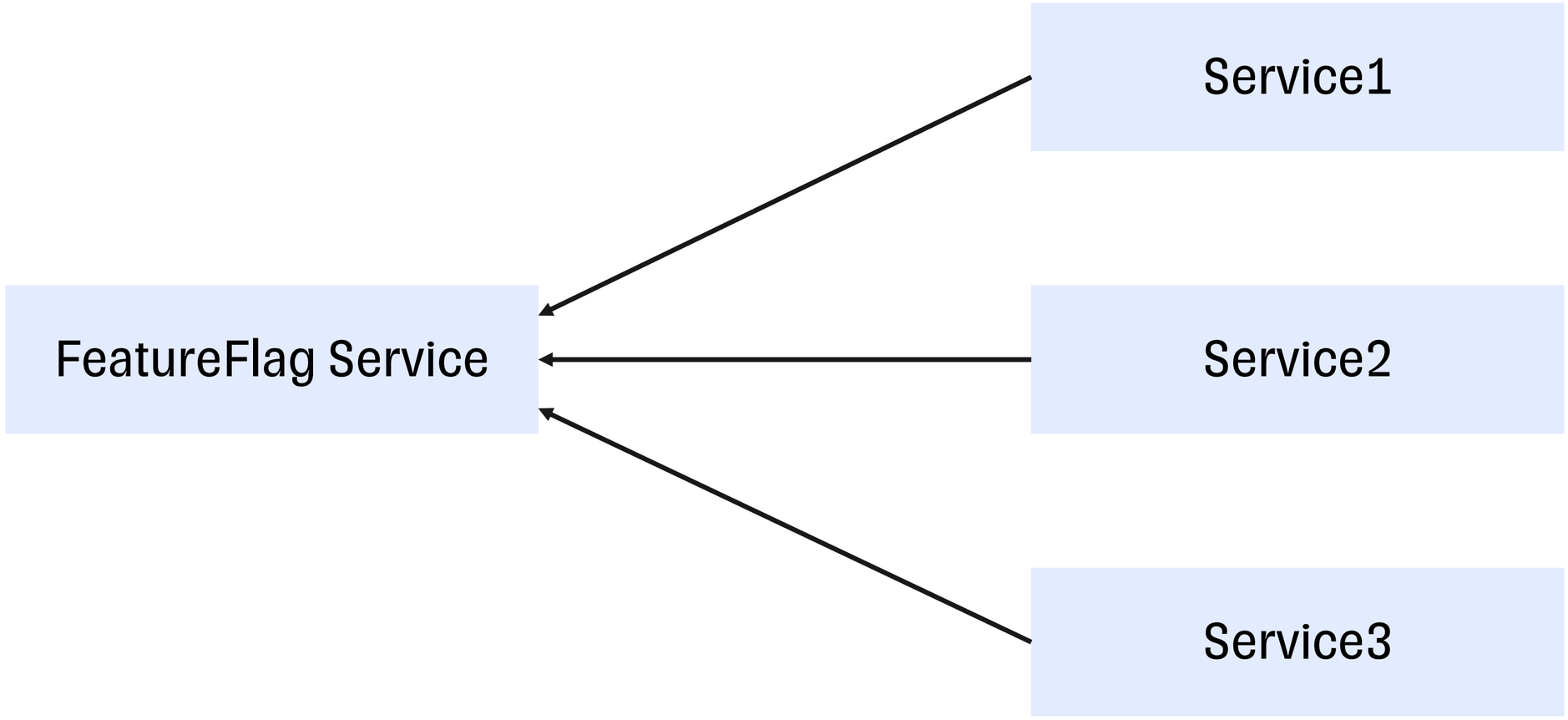


Итог

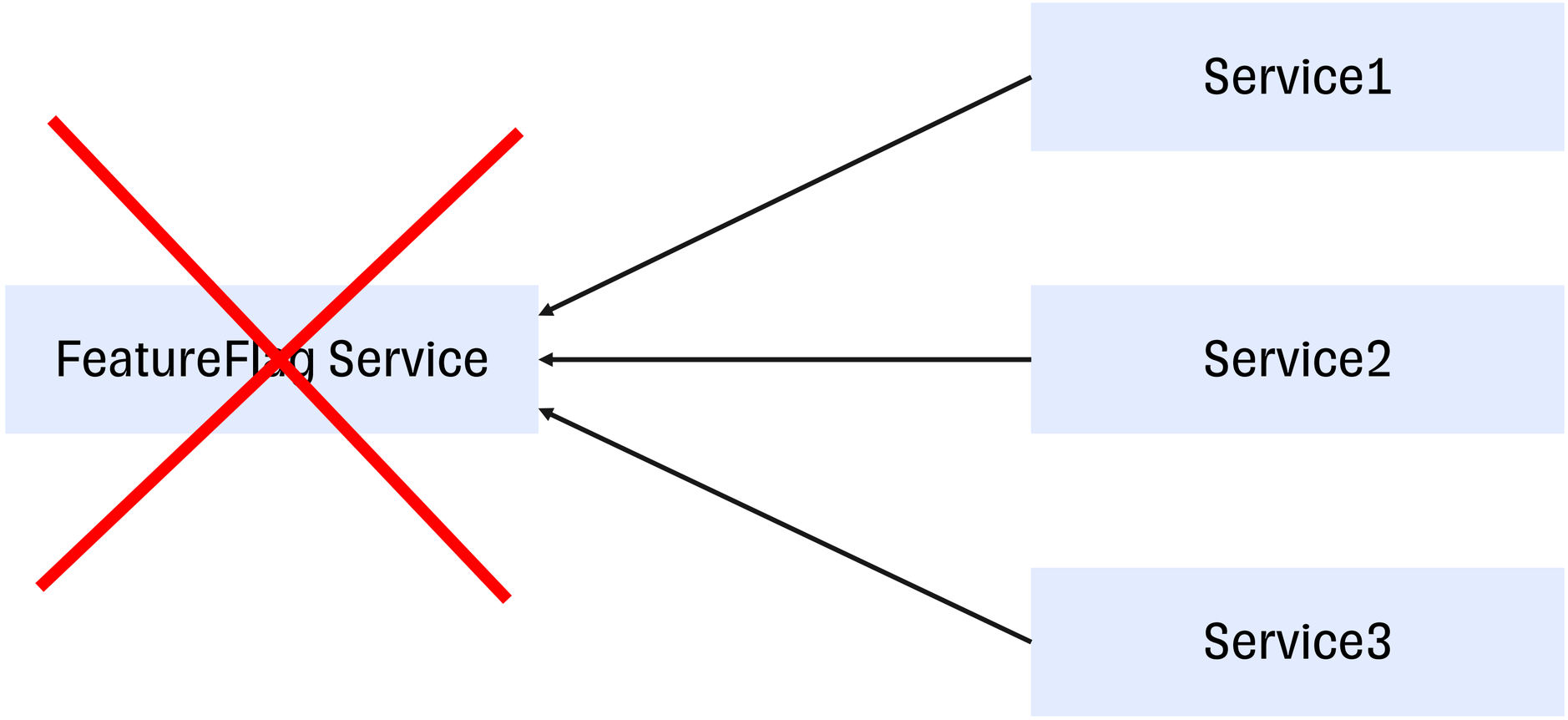
- Имеем возможность включать и выключать определённый функционал без изменения кода приложения прямо в рантайме.
- Тестировщик сам по необходимости включает/выключает определённые функции.
- По запросу бизнеса можем включать определённый функционал на проде без релиза.

Возможные проблемы

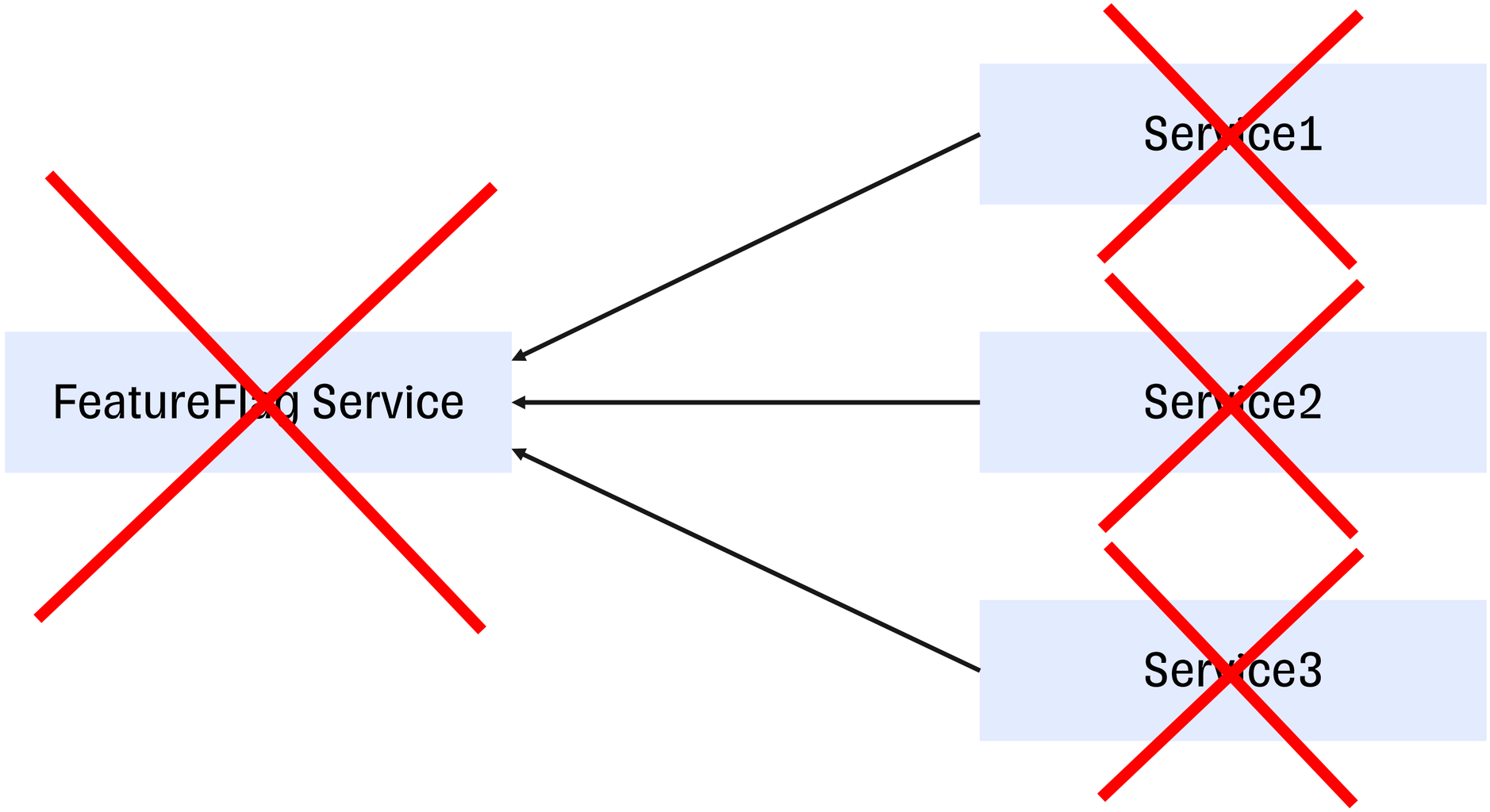
Дополнительная точка отказа



Дополнительная точка отказа

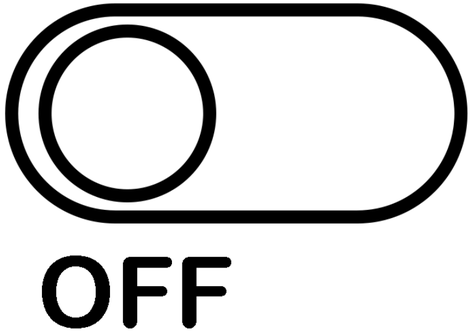


Дополнительная точка отказа

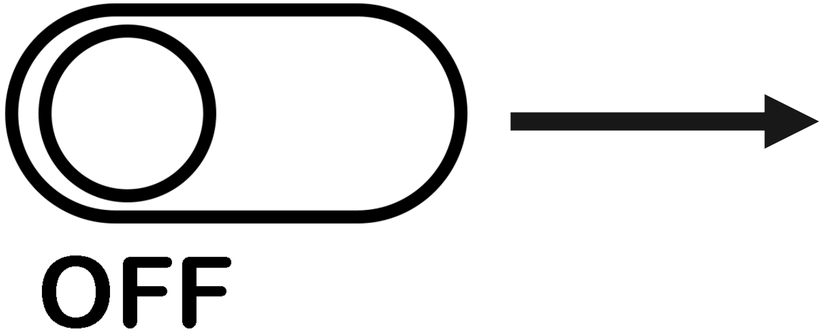


Накопление техдолга

Накопление техдолга

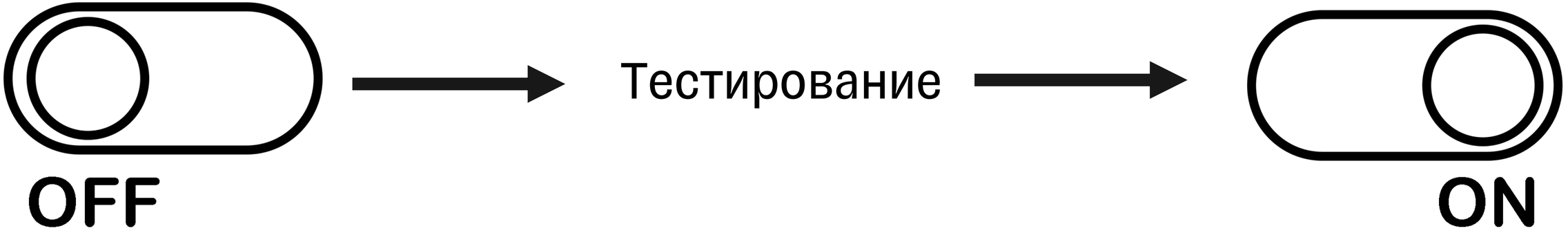


Накопление техдолга



Тестирование

Накопление техдолга

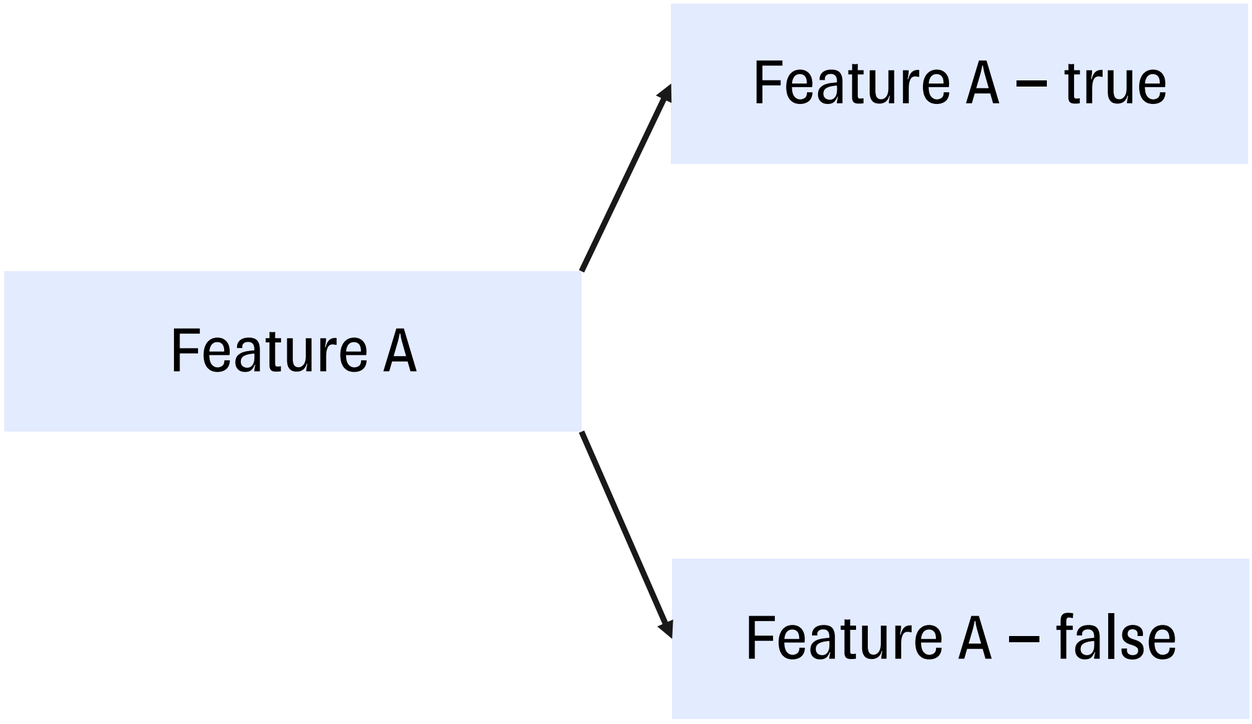


Накопление техдолга

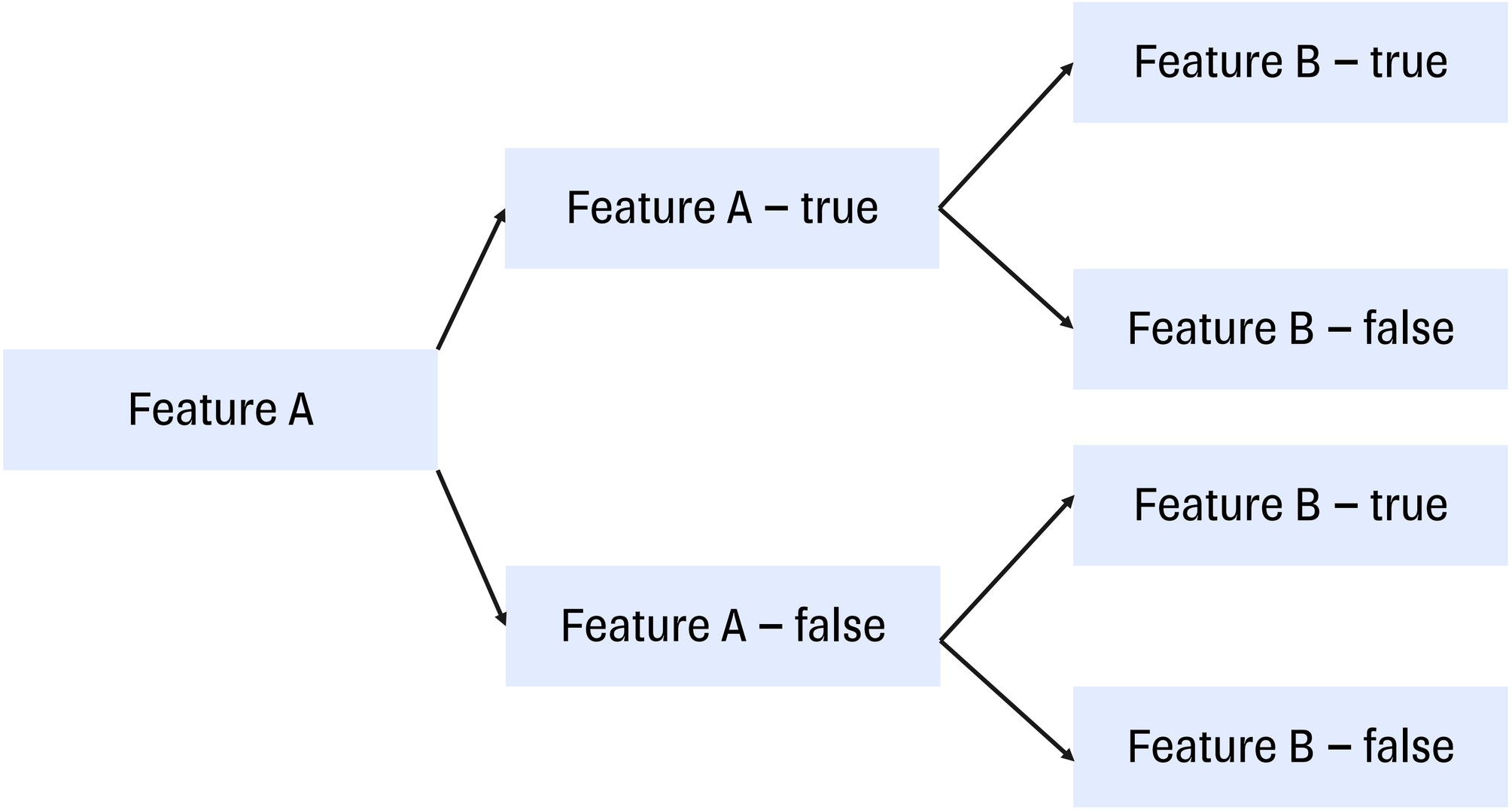
Высокая кардинальность фич

Feature A

Высокая кардинальность фич



Высокая кардинальность фич



Вывод

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/Unleash/unleash - репозиторий проекта Unleash

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

https://github.com/fbeltrao/ConfigMapFileProvider - почему нужно прокидывать FileProvider и реализация для старых версий .NET

https://github.com/kawwik/feature-flags - мой репозиторий с примерами

Ваши вопросы