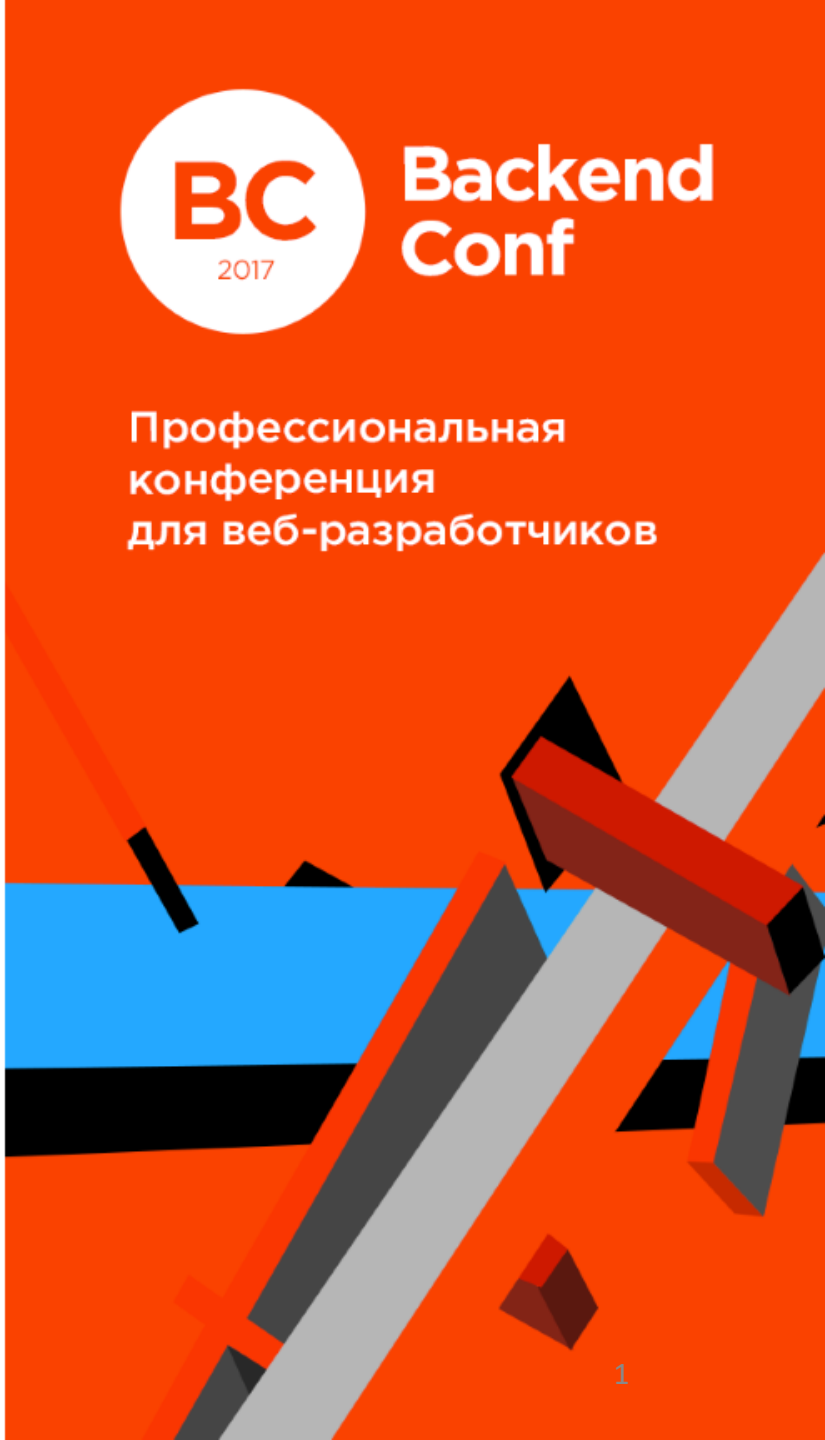
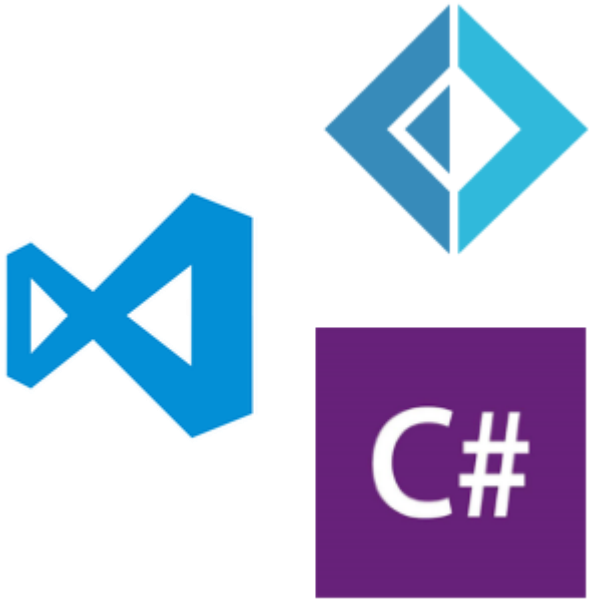
REST-сервисы на ASP.NET Core под Linux в продакшене

Денис Иванов
denis@ivanovdenis.ru
@denisivanov



Обо мне



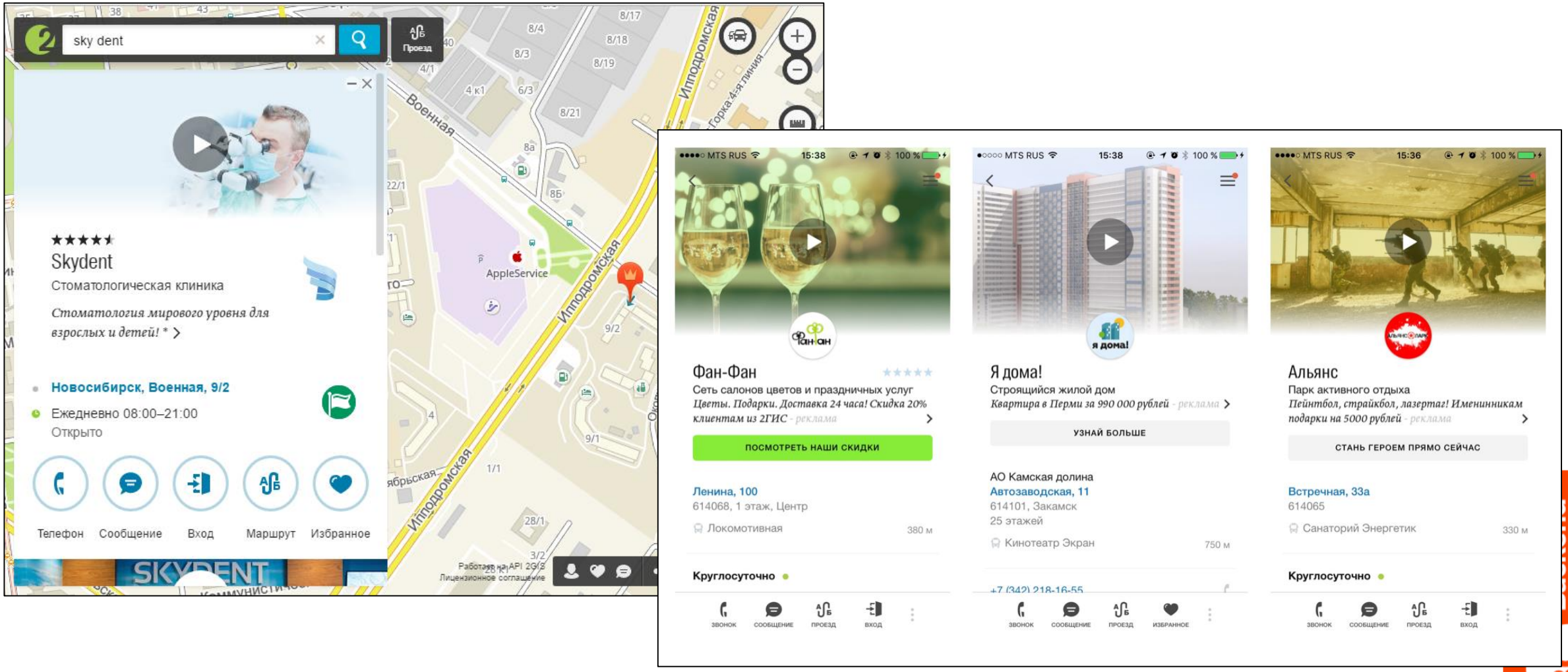
Цель

Поделиться опытом разработки и запуска в продакшен REST-сервисов на ASP.NET Core на Kubernetes

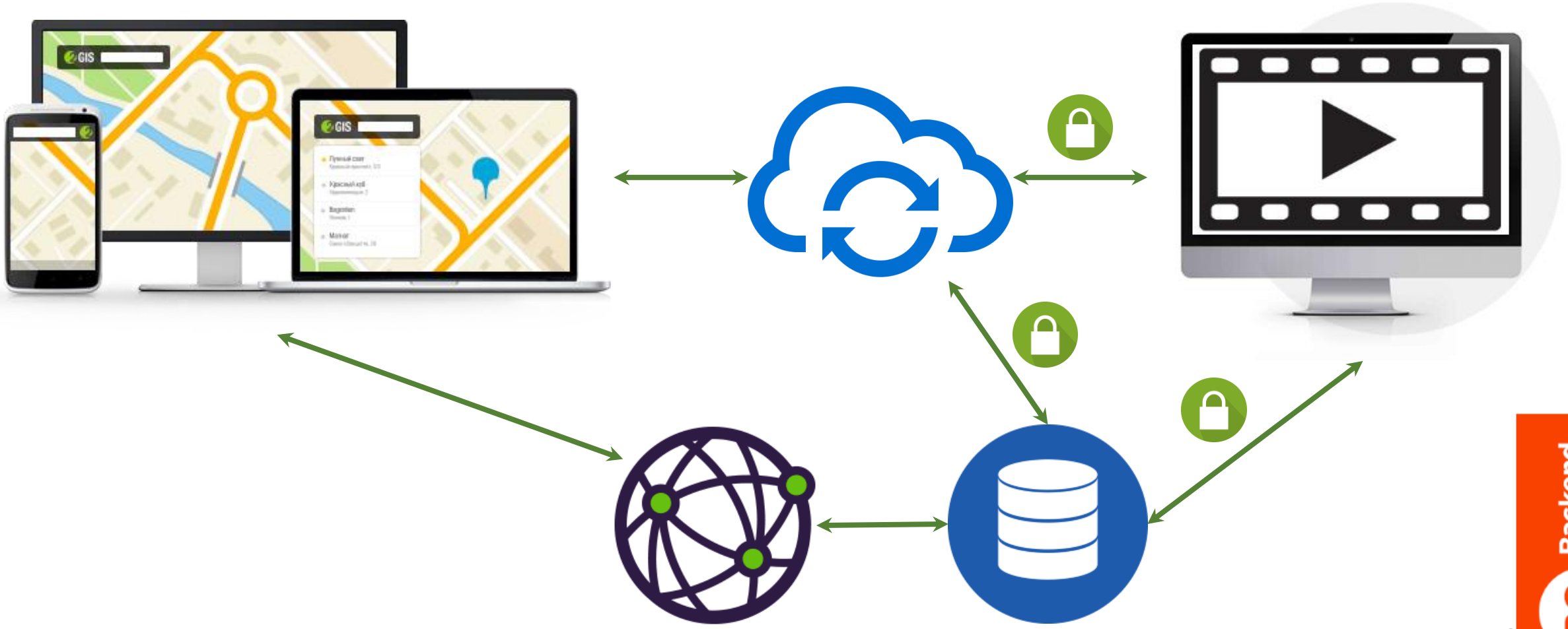
План

- Коротко о сервисе
- On-premise платформа
- .NET Core, ASP.NET Core, базовые фичи
- Билд
- Деплой
- Нагрузочное тестирование
- Performance
  - Кэширование
  - Асинхронность и многопоточность

Коротко о сервисе



Сервис видеорекламы



Сервис видеорекламы

- 99.99% доступность по миру
- Время ответа 200ms*

Сервис видеорекламы

Сервис видеорекламы

Почему Linux

- Существующая on-premise платформа
  - GitLab CI
  - CI starting kit на основе make
  - Docker hub & docker images
- Компоненты на любом технологическом стеке
- Kubernetes

The Twelve-Factor App (1-6)

- Одно приложение – один репозиторий
- Зависимости – вместе с приложением
- Конфигурация через окружение
- Используемые сервисы как ресурсы
- Фазы билда, создания образов и исполнения разделены
- Сервисы – отдельные stateless процессы

https://12factor.net

The Twelve-Factor App (7-12)

- Port binding
- Масштабирование через процессы
- Быстрая остановка и запуск процессов
- Среды максимально похожи
- Логирование в stdout
- Административные процессы

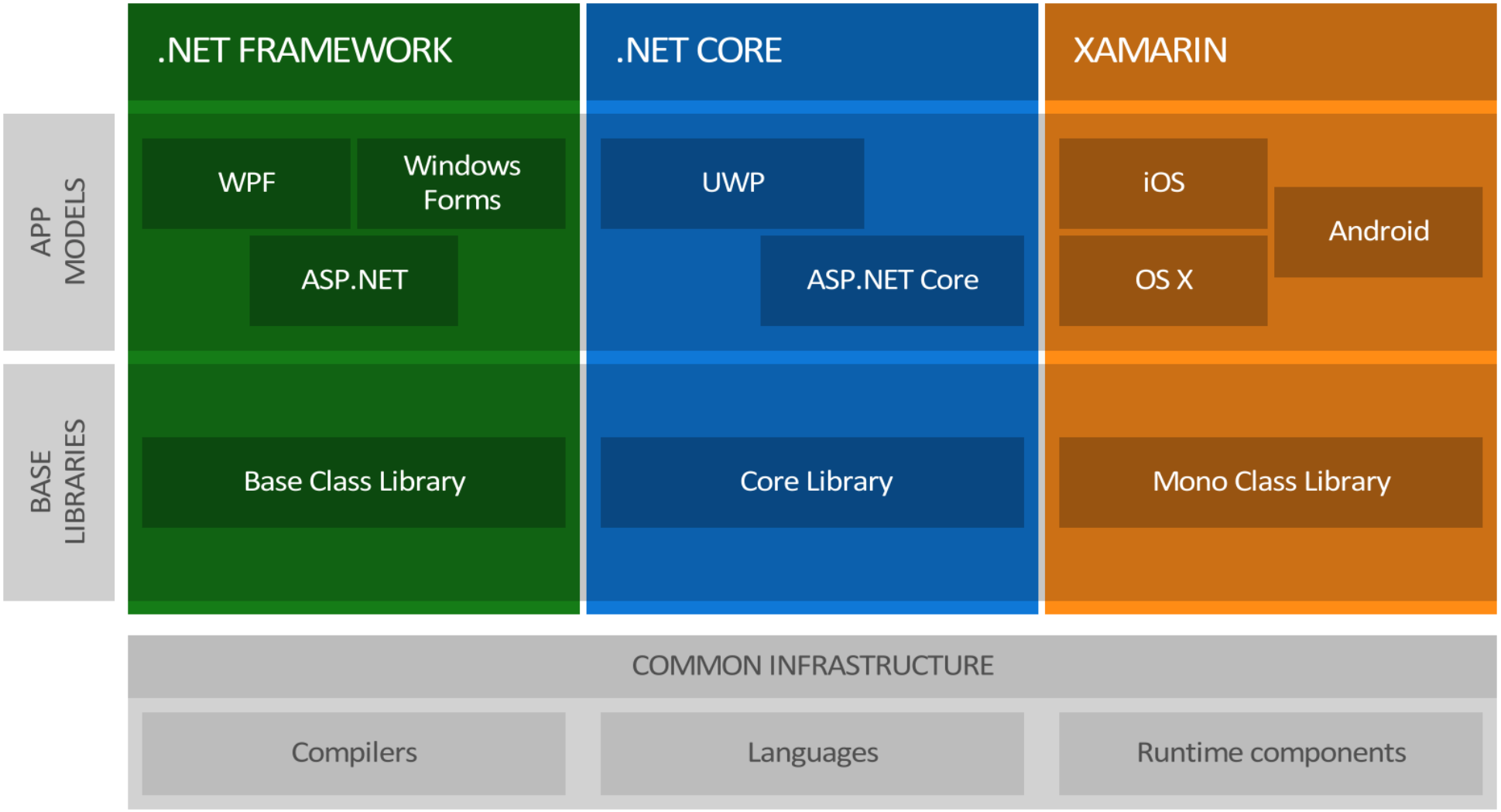
https://12factor.net

Код и презентация

https://github.com/denisivan0v/backend-conf-2017

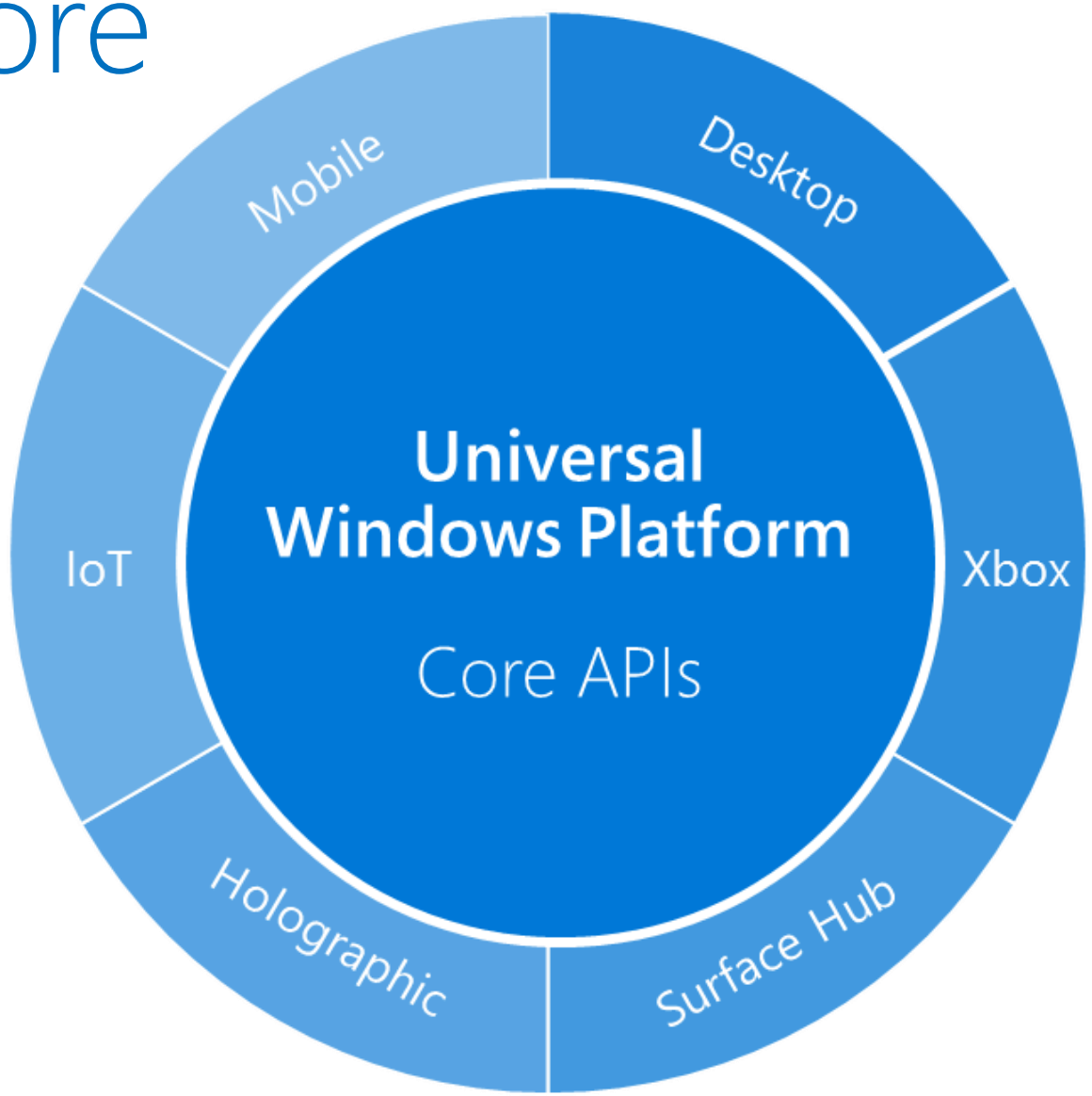
- Коротко о сервисе
- On-premise платформа
- **.NET Core, ASP.NET Core, базовые фичи**
- Билд
- Деплой
- Нагрузочное тестирование
- Performance
  - Кэширование
  - Асинхронность и многопоточность

.NET Core



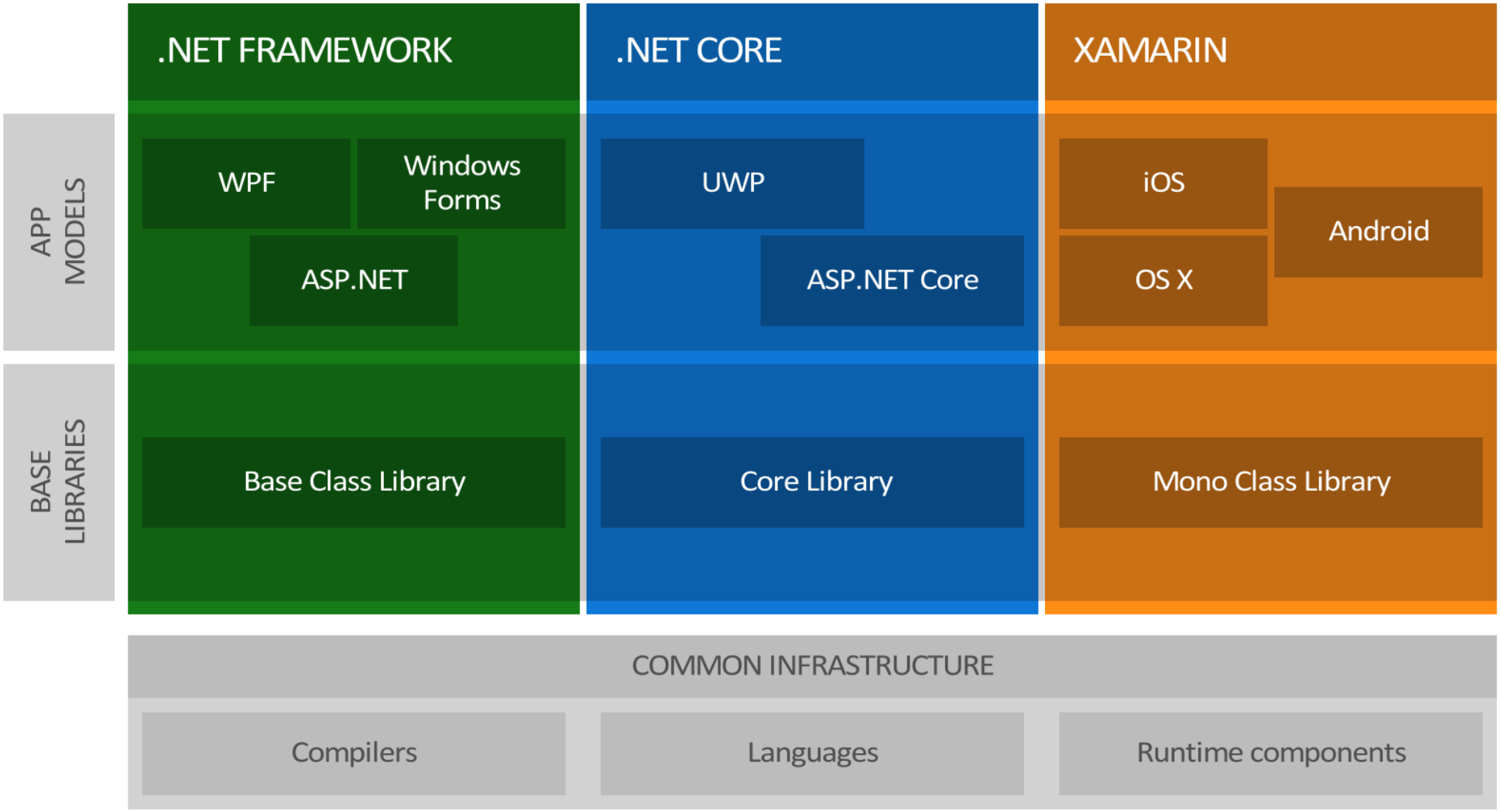
http://www.telerik.com/blogs/the-new-net-core-1-is-here

.NET Core



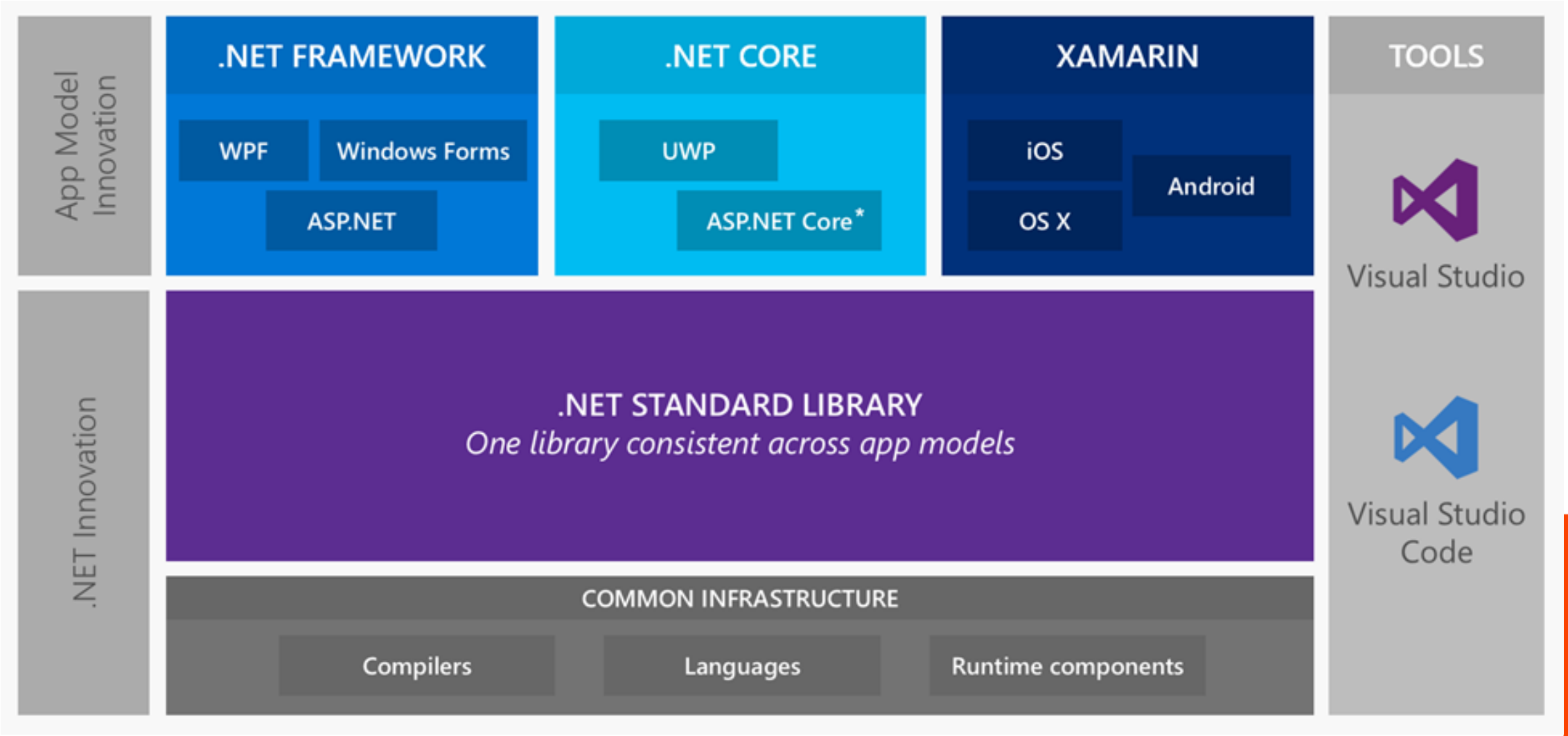
https://blogs.windows.com/buildingapps/2015/04/29/expanding-the-universal-windows-platform-at-build-2015/

.NET Core



http://www.telerik.com/blogs/the-new-net-core-1-is-here

.NET Core

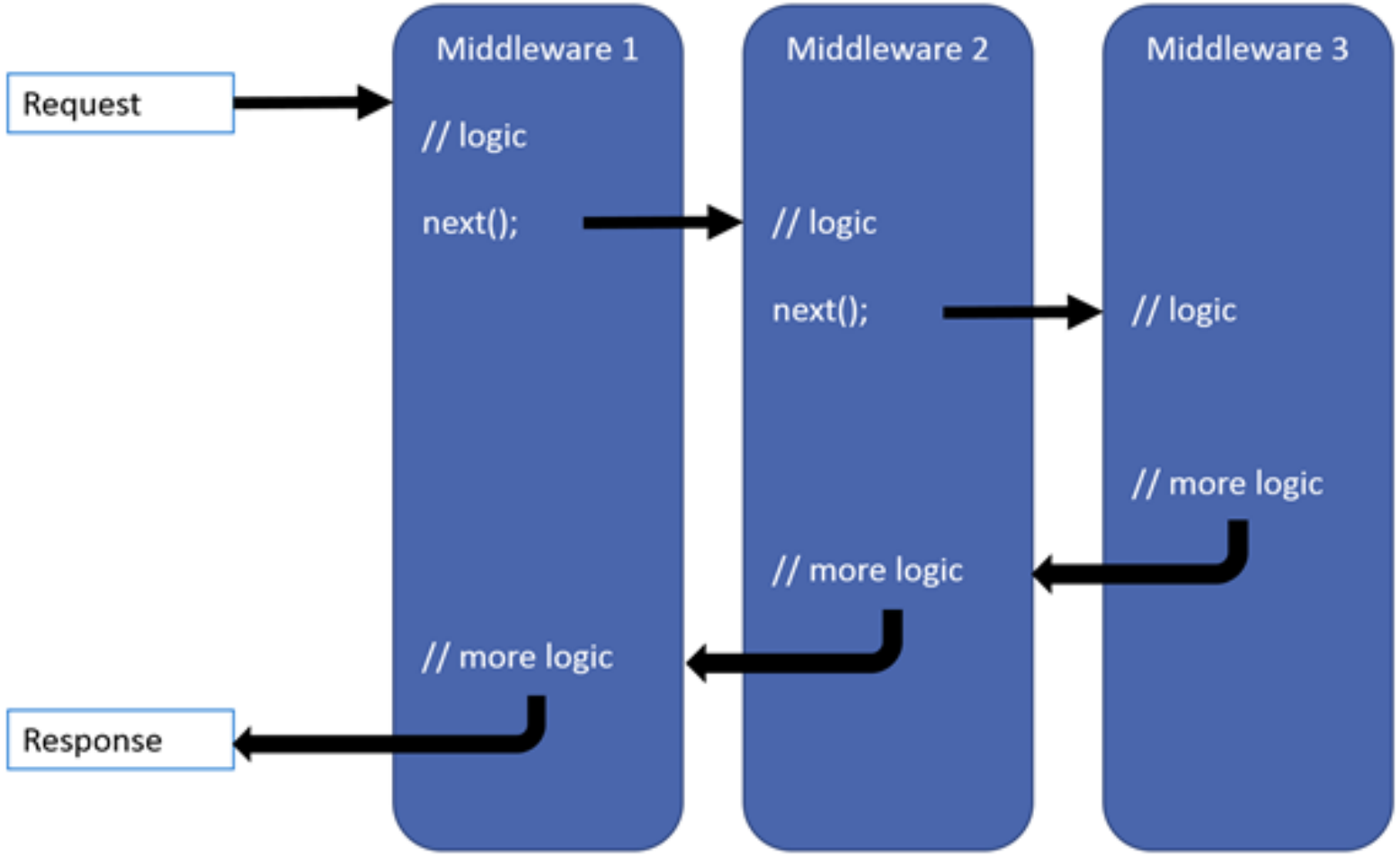


http://www.c-sharpcorner.com/article/getting-started-with-net-core-on-visual-studio-2017/

.NET Core. Self-contained deployment

- Полный контроль зависимостей
- Явное указание платформы при билде (**win10-x64** / **ubuntu.16.04-x64** / **osx.10.12-x64**)
- Только необходимый фреймворк **netstandard1.6**
  - Microsoft.NETCore.Runtime.CoreCLR
  - Microsoft.NETCore.DotNetHostPolicy

ASP.NET Core



```
public sealed class HealthCheckMiddleware
{
   private const string Path = "/healthcheck";
   private readonly RequestDelegate _next;

   public HealthCheckMiddleware(RequestDelegate next)
   {
      _next = next;
   }

   public async Task Invoke(HttpContext context)
   {
      if (!context.Request.Path.Equals(Path, StringComparison.OrdinalIgnoreCase))
      {
         await _next(context);
      }
      else
      {
         context.Response.ContentType = "text/plain";
         context.Response.StatusCode = 200;
         context.Response.Headers.Add(HeaderNames.Connection, "close");
         await context.Response.WriteAsync("OK");
      }
   }
}
```

```
public sealed class HealthCheckMiddleware
{
   private const string Path = "/healthcheck";
   private readonly RequestDelegate _next;

   public HealthCheckMiddleware(RequestDelegate next)
   {
      _next = next;
   }

   public async Task Invoke(HttpContext context)
   {
      if (!context.Request.Path.Equals(Path, StringComparison.OrdinalIgnoreCase))
      {
         await _next(context);
      }
      else
      {
         context.Response.ContentType = "text/plain";
         context.Response.StatusCode = 200;
         context.Response.Headers.Add(HeaderNames.Connection, "close");
         await context.Response.WriteAsync("OK");
      }
   }
}
```

```
public sealed class HealthCheckMiddleware
{
   private const string Path = "/healthcheck";
   private readonly RequestDelegate _next;

   public HealthCheckMiddleware(RequestDelegate next)
   {
      _next = next;
   }

   public async Task Invoke(HttpContext context)
   {
      if (!context.Request.Path.Equals(Path, StringComparison.OrdinalIgnoreCase))
      {
         await _next(context);
      }
      else
      {
         context.Response.ContentType = "text/plain";
         context.Response.StatusCode = 200;
         context.Response.Headers.Add(HeaderNames.Connection, "close");
         await context.Response.WriteAsync("OK");
      }
   }
}
```

```
public sealed class HealthCheckMiddleware
{
   private const string Path = "/healthcheck";
   private readonly RequestDelegate _next;

   public HealthCheckMiddleware(RequestDelegate next)
   {
      _next = next;
   }

   public async Task Invoke(HttpContext context)
   {
      if (!context.Request.Path.Equals(Path, StringComparison.OrdinalIgnoreCase))
      {
         await _next(context);
      }
      else
      {
         context.Response.ContentType = "text/plain";
         context.Response.StatusCode = 200;
         context.Response.Headers.Add(HeaderNames.Connection, "close");
         await context.Response.WriteAsync("OK");
      }
   }
}
```

```
public sealed class HealthCheckMiddleware
{
   private const string Path = "/healthcheck";
   private readonly RequestDelegate _next;

   public HealthCheckMiddleware(RequestDelegate next)
   {
      _next = next;
   }

   public async Task Invoke(HttpContext context)
   {
      if (!context.Request.Path.Equals(Path, StringComparison.OrdinalIgnoreCase))
      {
         await _next(context);
      }
      else
      {
         context.Response.ContentType = "text/plain";
         context.Response.StatusCode = 200;
         context.Response.Headers.Add(HeaderNames.Connection, "close");
         await context.Response.WriteAsync("OK");
      }
   }
}
```

Базовые фичи REST-сервисов

- Логирование
  - Структурное логирование
- Версионирование API
  - SemVer
  - DateTime
- Формальное описание API
  - Swagger

Структурное логирование

https://serilog.net/

```
[18:52:17 INF] Pre-discount tax total calculated at $10.95
[18:52:17 INF] Added CartItem {Id=7e4fc707-958b-4215-ac1b-e20d202b0aea, Descript
ion="Super-Multi-Coated Takumar 50mm f1.4", Total=9.05} to cart; cart contains 1
 items
[18:52:17 ERR] Timer cannot be reset because it is disposed
System.ObjectDisposedException: Cannot access a disposed object.
   at System.Threading.TimerQueueTimer.Change(UInt32 dueTime, UInt32 period)
   at System.Threading.Timer.Change(Int32 dueTime, Int32 period)
   at Serilog.Generator.ActiveAgent.Schedule() in c:\TeamCity\buildAgent\work\de
33ba89868b8a14\src\serilog-generator\ActiveAgent.cs:line 30
```

appsettings.json

```
{
  "Serilog": {
    "MinimumLevel": "Debug",
    "WriteTo": [
      {
        "Name": "Console",
        "Args": {
          "formatter":
            "Serilog.Formatting.Compact.RenderedCompactJsonFormatter,
             Serilog.Formatting.Compact"
        }
      }
    ],
    "Enrich": [ "FromLogContext", "WithThreadId" ]
  }
}
```

appsettings.json

```
{
  "Serilog": {
    "MinimumLevel": "Debug",
    "WriteTo": [
      {
        "Name": "Console",
        "Args": {
          "formatter":
            "Serilog.Formatting.Compact.RenderedCompactJsonFormatter,
             Serilog.Formatting.Compact"
        }
      }
    ],
    "Enrich": [ "FromLogContext", "WithThreadId" ]
  }
}
```

appsettings.json

```
{
  "Serilog": {
    "MinimumLevel": "Debug",
    "WriteTo": [
      {
        "Name": "Console",
        "Args": {
          "formatter":
            "Serilog.Formatting.Compact.RenderedCompactJsonFormatter,
             Serilog.Formatting.Compact"
        }
      }
    ],
    "Enrich": [ "FromLogContext", "WithThreadId" ]
  }
}
```

appsettings.json

```
{
  "Serilog": {
    "MinimumLevel": "Debug",
    "WriteTo": [
      {
        "Name": "Console",
        "Args": {
          "formatter":
            "Serilog.Formatting.Compact.RenderedCompactJsonFormatter,
             Serilog.Formatting.Compact"
        }
      }
    ],
    "Enrich": [ "FromLogContext", "WithThreadId" ]
  }
}
```

appsettings.json

```
{
  "Serilog": {
    "MinimumLevel": "Debug",
    "WriteTo": [
      {
        "Name": "Console",
        "Args": {
          "formatter":
            "Serilog.Formatting.Compact.RenderedCompactJsonFormatter,
             Serilog.Formatting.Compact"
        }
      }
    ],
    "Enrich": [ "FromLogContext", "WithThreadId" ]
  }
}
```

ASP.NET API versioning

- https://github.com/Microsoft/aspnet-api-versioning
- Microsoft REST versioning guidelines

- /api/foo?api-version=1.0
- /api/foo?api-version=2.0-Alpha
- /api/foo?api-version=2015-05-01.3.0
- /api/v1/foo
- /api/v2.0-Alpha/foo
- /api/v2015-05-01.3.0/foo

```
[ApiVersion("1.0")]
[Route(“api/medias")]                          // /api/medias
[Route(“api/{version:apiVersion}/medias")]   // /api/1.0/medias
public sealed class MediasController : Controller
{
   // /api/medias/id?api-version=1.0 or /api/1.0/medias/id
   [HttpGet("{id}")]
   public async Task<IActionResult> Get(long id)
   {
      …
   }
}
```

```
[ApiVersion("1.0")]
[Route("api/medias")]                          // /api/medias
[Route("api/{version:apiVersion}/medias")]     // /api/1.0/medias
public sealed class MediasController : Controller
{
    // /api/medias/id?api-version=1.0 or /api/1.0/medias/id
    [HttpGet("{id}")]
    public async Task<IActionResult> Get(long id)
    {
        …
    }
}
```

```
[ApiVersion("1.0")]
[Route("api/medias")]                                   // /api/medias
[Route("api/{version:apiVersion}/medias")]   // /api/1.0/medias
public sealed class MediasController : Controller
{
    // /api/medias/id?api-version=1.0 or /api/1.0/medias/id
    [HttpGet("{id}")]
    public async Task<IActionResult> Get(long id)
    {
        …
    }
}
```

```
[ApiVersion("1.0")]
[Route("api/medias")]                                   // /api/medias
[Route("api/{version:apiVersion}/medias")]              // /api/1.0/medias
public sealed class MediasController : Controller
{
    // /api/medias/id?api-version=1.0 or /api/1.0/medias/id
    [HttpGet("{id}")]
    public async Task<IActionResult> Get(long id)
    {
        …
    }
}
```

```
[ApiVersion("1.0")]
[Route("api/medias")]                          // /api/medias
[Route("api/{version:apiVersion}/medias")]     // /api/1.0/medias
public sealed class MediasController : Controller
{
    // /api/medias/id?api-version=1.0 or /api/1.0/medias/id
    [HttpGet("{id}")]
    public async Task<IActionResult> Get(long id)
    {
        …
    }
}
```

```
[ApiVersion("1.0")]
[ApiVersion(“2.0")]
[Route(“api/medias")]
[Route(“api/{version:apiVersion}/medias")]
public sealed class MediasController : GatewayController
{
    // /api/medias/id?api-version=1.0 or /api/1.0/medias/id
    [HttpGet("{id}")]
    public async Task<IActionResult> Get(long id)
    {
        …
    }

    // /api/medias/id?api-version=2.0 or /api/2.0/medias/id
    [MapToApiVersion(“2.0")]
    [HttpGet("{id}")]
    public async Task<IActionResult> GetV2(long id)
    {
        …
    }
}
```

```
[ApiVersion("1.0")]
[ApiVersion(“2.0")]
[Route(“api/medias")]
[Route(“api/{version:apiVersion}/medias")]
public sealed class MediasController : GatewayController
{
    // /api/medias/id?api-version=1.0 or /api/1.0/medias/id
    [HttpGet("{id}")]
    public async Task<IActionResult> Get(long id)
    {
        …
    }

    // /api/medias/id?api-version=2.0 or /api/2.0/medias/id
    [MapToApiVersion(“2.0")]
    [HttpGet("{id}")]
    public async Task<IActionResult> GetV2(long id)
    {
        …
    }
}
```

```
[ApiVersion("1.0")]
[ApiVersion("2.0")]
[Route("api/medias")]
[Route("api/{version:apiVersion}/medias")]
public sealed class MediasController : GatewayController
{
    // /api/medias/id?api-version=1.0 or /api/1.0/medias/id
    [HttpGet("{id}")]
    public async Task<IActionResult> Get(long id)
    {
        …
    }

    // /api/medias/id?api-version=2.0 or /api/2.0/medias/id
    [MapToApiVersion("2.0")]
    [HttpGet("{id}")]
    public async Task<IActionResult> GetV2(long id)
    {
        …
    }
}
```

https://github.com/domaindrivendev/Swashbuckle.AspNetCore

```
services.AddSwaggerGen(
    x =>
    {
        IApiVersionDescriptionProvider provider;
        foreach (var description in provider.ApiVersionDescriptions)
        {
            x.SwaggerDoc(description.GroupName, new Info { … });
        }
    });

…

app.UseSwagger();
app.UseSwaggerUI(
    c =>
    {
        IApiVersionDescriptionProvider provider;
        foreach (var description in provider.ApiVersionDescriptions)
        {
            options.SwaggerEndpoint(
                $"/swagger/{description.GroupName}/swagger.json",
                description.GroupName.ToUpperInvariant());
        }
    });
```

```
services.AddSwaggerGen(
    x =>
    {
        IApiVersionDescriptionProvider provider;
        foreach (var description in provider.ApiVersionDescriptions)
        {
            x.SwaggerDoc(description.GroupName, new Info { … });
        }
    });
```

…

```
app.UseSwagger();
app.UseSwaggerUI(
    c =>
    {
        IApiVersionDescriptionProvider provider;
        foreach (var description in provider.ApiVersionDescriptions)
        {
            options.SwaggerEndpoint(
                $"/swagger/{description.GroupName}/swagger.json",
                description.GroupName.ToUpperInvariant());
        }
    });
```

```
services.AddSwaggerGen(
   x =>
   {
      IApiVersionDescriptionProvider provider;
      foreach (var description in provider.ApiVersionDescriptions)
      {
         x.SwaggerDoc(description.GroupName, new Info { … });
      }
   });
```

…

```
app.UseSwagger();
app.UseSwaggerUI(
   c =>
   {
      IApiVersionDescriptionProvider provider;
      foreach (var description in provider.ApiVersionDescriptions)
      {
         options.SwaggerEndpoint(
            $"/swagger/{description.GroupName}/swagger.json",
            description.GroupName.ToUpperInvariant());
      }
   });
```

- Коротко о сервисе
- On-premise платформа
- .NET Core, ASP.NET Core, базовые фичи
- **Билд**
- Деплой
- Нагрузочное тестирование
- Performance
  - Кэширование
  - Асинхронность и многопоточность

```
build:backend-conf-demo:
  image: $REGISTRY/microsoft/aspnetcore-build:1.1.2
  stage: build:app
  script:
    - dotnet restore --runtime ubuntu.16.04-x64
    - dotnet test Demo.Tests/Demo.Tests.csproj
     --configuration Release
    - dotnet publish Demo --configuration Release
     --runtime ubuntu.16.04-x64 --output publish/backend-conf
  tags: [ 2gis, docker ]
  artifacts:
    paths:
      - publish/backend-conf/
```

```
build:backend-conf-demo:
  image: $REGISTRY/microsoft/aspnetcore-build:1.1.2
  stage: build:app
  script:
    - dotnet restore --runtime ubuntu.16.04-x64
    - dotnet test Demo.Tests/Demo.Tests.csproj
     --configuration Release
    - dotnet publish Demo --configuration Release
     --runtime ubuntu.16.04-x64 --output publish/backend-conf
  tags: [ 2gis, docker ]
  artifacts:
    paths:
      - publish/backend-conf/
```

```
build:backend-conf-demo:
  image: $REGISTRY/microsoft/aspnetcore-build:1.1.2
  stage: build:app
  script:
    - dotnet restore --runtime ubuntu.16.04-x64
    - dotnet test Demo.Tests/Demo.Tests.csproj
     --configuration Release
    - dotnet publish Demo --configuration Release
     --runtime ubuntu.16.04-x64 --output publish/backend-conf
  tags: [ 2gis, docker ]
  artifacts:
    paths:
      - publish/backend-conf/
```

```
build:backend-conf-demo:
  image: $REGISTRY/microsoft/aspnetcore-build:1.1.2
  stage: build:app
  script:
    - dotnet restore --runtime ubuntu.16.04-x64
    - dotnet test Demo.Tests/Demo.Tests.csproj
     --configuration Release
    - dotnet publish Demo --configuration Release
     --runtime ubuntu.16.04-x64 --output publish/backend-conf
  tags: [ 2gis, docker ]
  artifacts:
    paths:
      - publish/backend-conf/
```

```
build:backend-conf-demo:
  image: $REGISTRY/microsoft/aspnetcore-build:1.1.2
  stage: build:app
  script:
    - dotnet restore --runtime ubuntu.16.04-x64
    - dotnet test Demo.Tests/Demo.Tests.csproj
     --configuration Release
    - dotnet publish Demo --configuration Release
     --runtime ubuntu.16.04-x64 --output publish/backend-conf
  tags: [ 2gis, docker ]
  artifacts:
    paths:
      - publish/backend-conf/
```

```
build:backend-conf-demo-image:
  stage: build:app
  script:
    - IMAGE=my-namespace/backend-conf TAG=$CI_TAG
      DOCKER_FILE=publish/backend-conf/Dockerfile
      DOCKER_BUILD_CONTEXT=publish/backend-conf
      make docker-build
    - IMAGE=my-namespace/backend-conf TAG=$CI_TAG
      make docker-push
  tags: [ docker-engine, io ]
  dependencies:
    - build:app
```

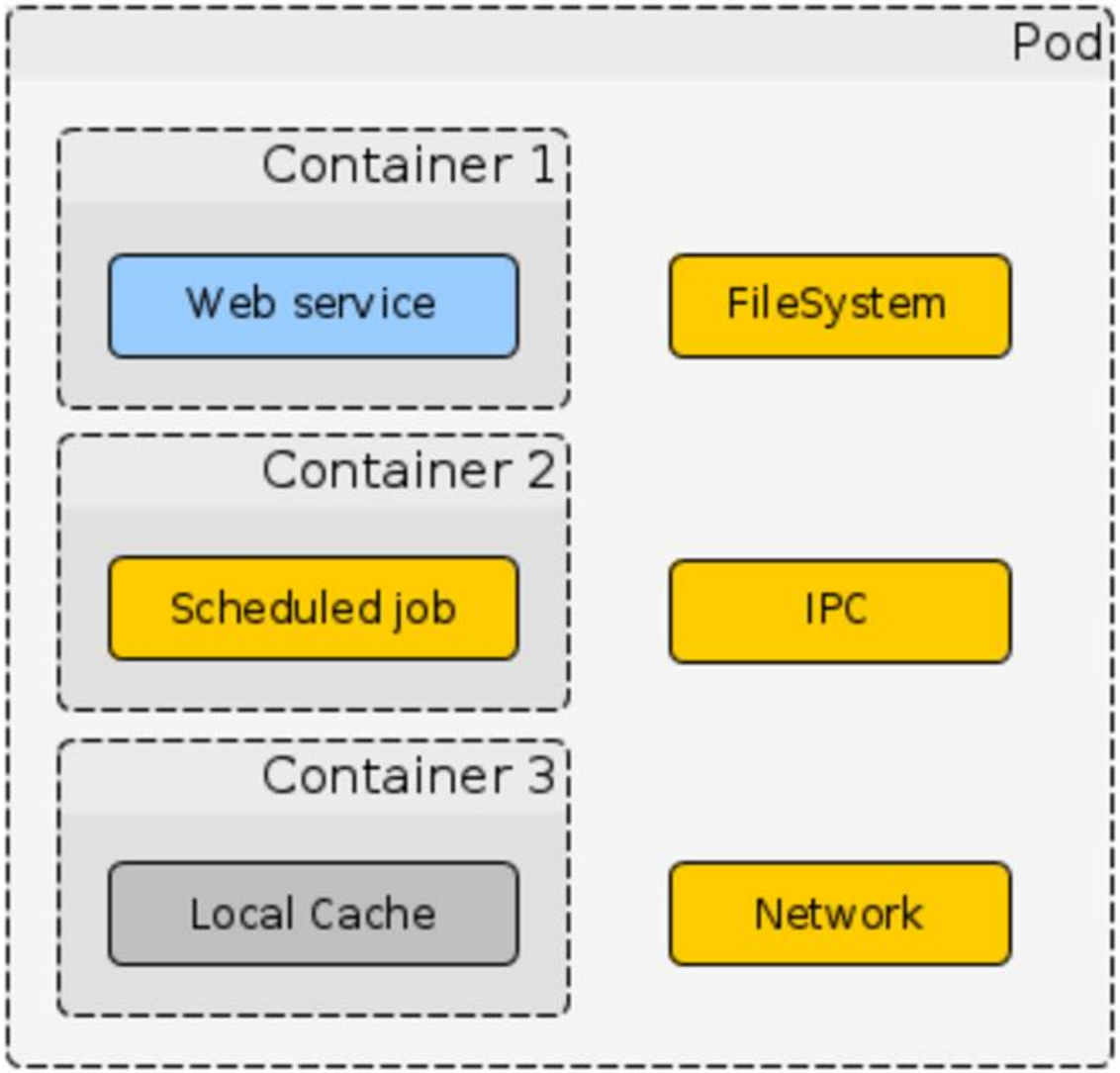
```
build:backend-conf-demo-image:
  stage: build:app
  script:
    - IMAGE=my-namespace/backend-conf TAG=$CI_TAG
      DOCKER_FILE=publish/backend-conf/Dockerfile
      DOCKER_BUILD_CONTEXT=publish/backend-conf
      make docker-build
    - IMAGE=my-namespace/backend-conf TAG=$CI_TAG
      make docker-push
  tags: [ docker-engine, io ]
  dependencies:
    - build:app
```

```
build:backend-conf-demo-image:
  stage: build:app
  script:
    - IMAGE=my-namespace/backend-conf TAG=$CI_TAG
      DOCKER_FILE=publish/backend-conf/Dockerfile
      DOCKER_BUILD_CONTEXT=publish/backend-conf
      make docker-build
    - IMAGE=my-namespace/backend-conf TAG=$CI_TAG
      make docker-push
  tags: [ docker-engine, io ]
  dependencies:
    - build:app
```

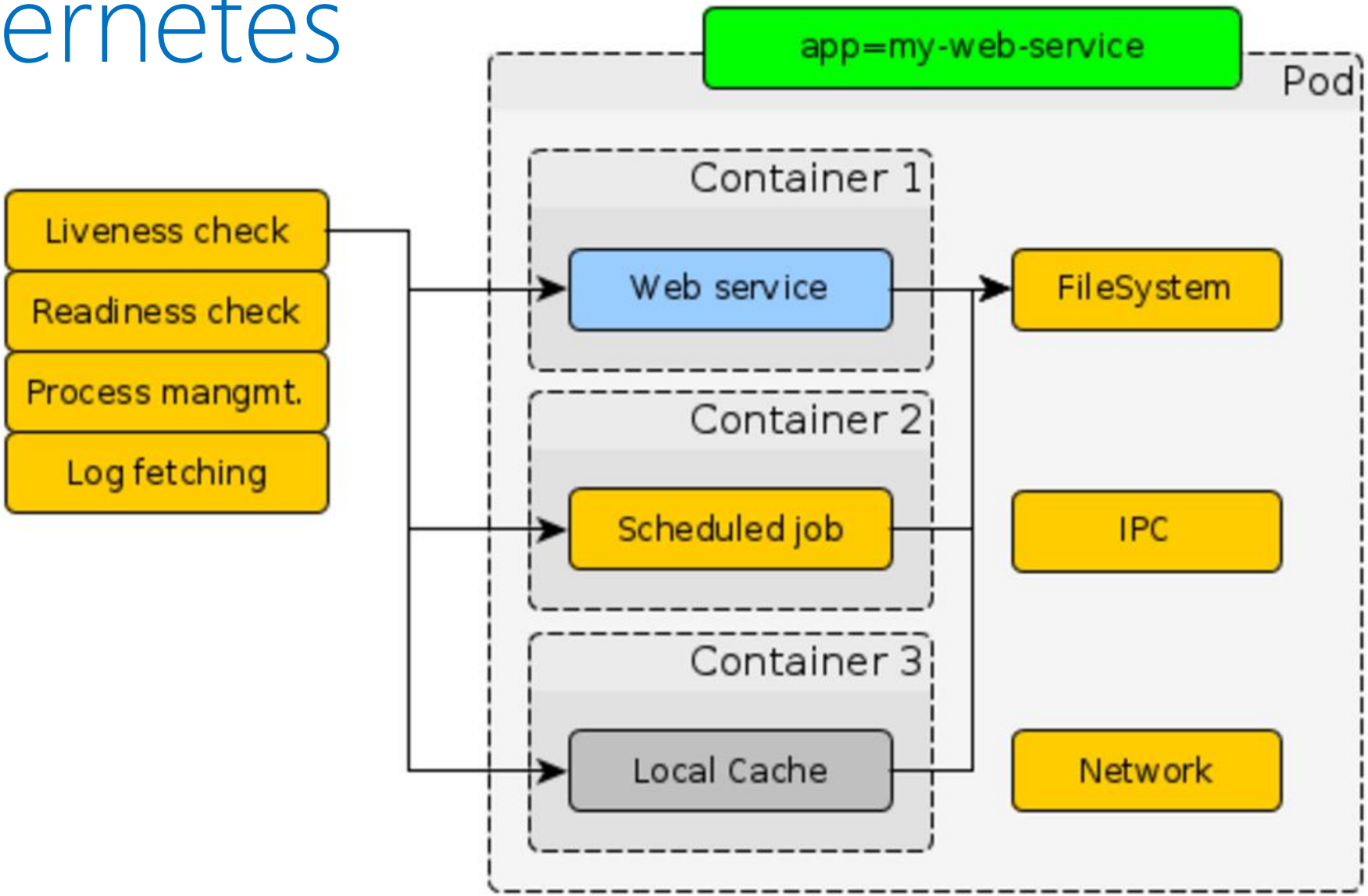
```
build:backend-conf-demo-image:
  stage: build:app
  script:
    - IMAGE=my-namespace/backend-conf TAG=$CI_TAG
      DOCKER_FILE=publish/backend-conf/Dockerfile
      DOCKER_BUILD_CONTEXT=publish/backend-conf
      make docker-build
    - IMAGE=my-namespace/backend-conf TAG=$CI_TAG
      make docker-push
  tags: [ docker-engine, io ]
  dependencies:
    - build:app
```

- Коротко о сервисе
- On-premise платформа
- .NET Core, ASP.NET Core, базовые фичи
- Билд
- **Деплой**
- Нагрузочное тестирование
- Performance
  - Кэширование
  - Асинхронность и многопоточность

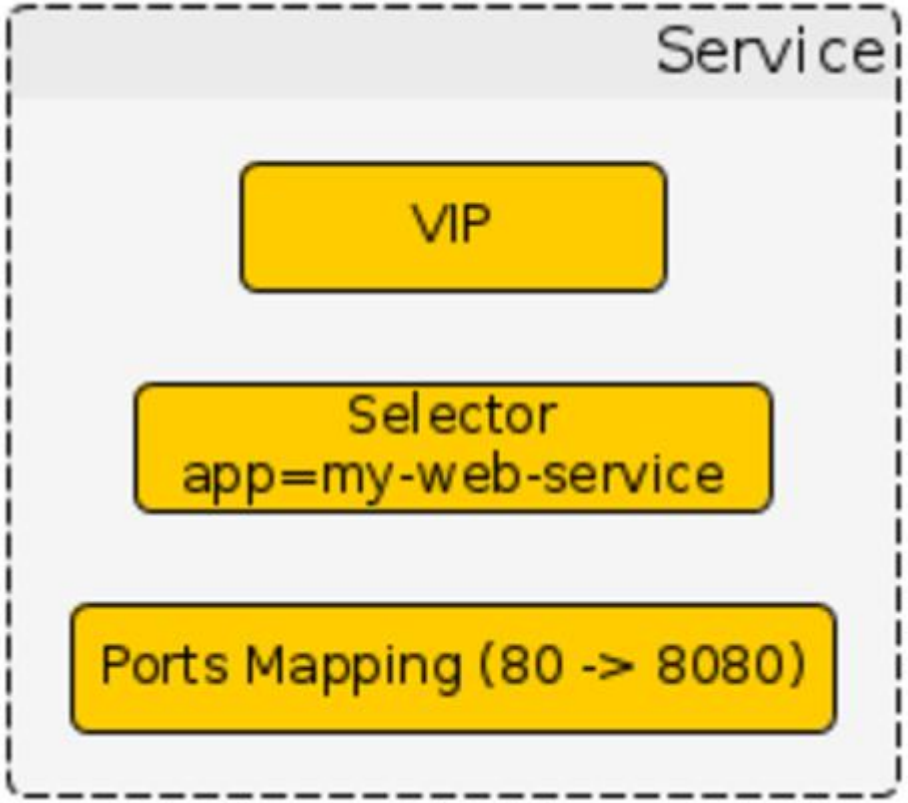
Kubernetes



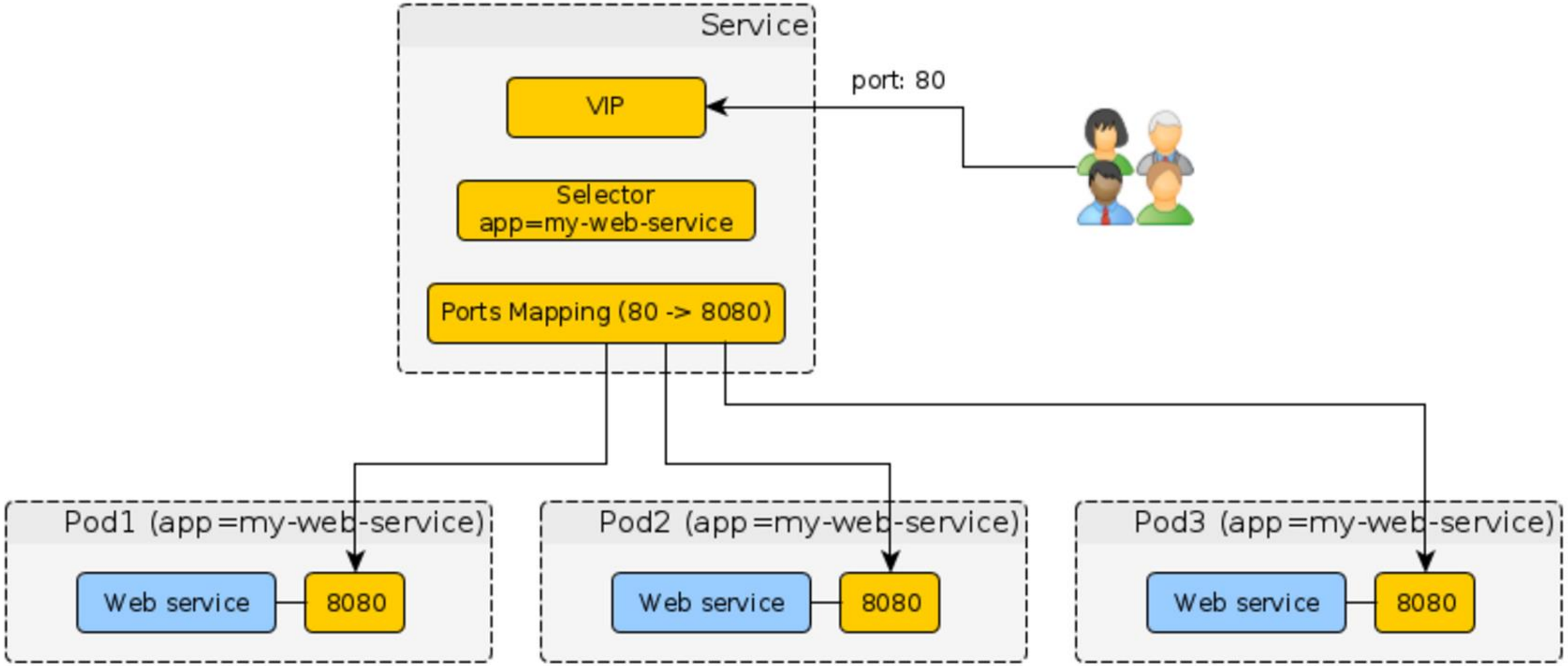
Kubernetes


app=my-web-service
Pod
Liveness check
Readiness check
Process mangmt.
Log fetching
Container 1
Web service
Container 2
Scheduled job
Container 3
Local Cache
FileSystem
IPC
Network

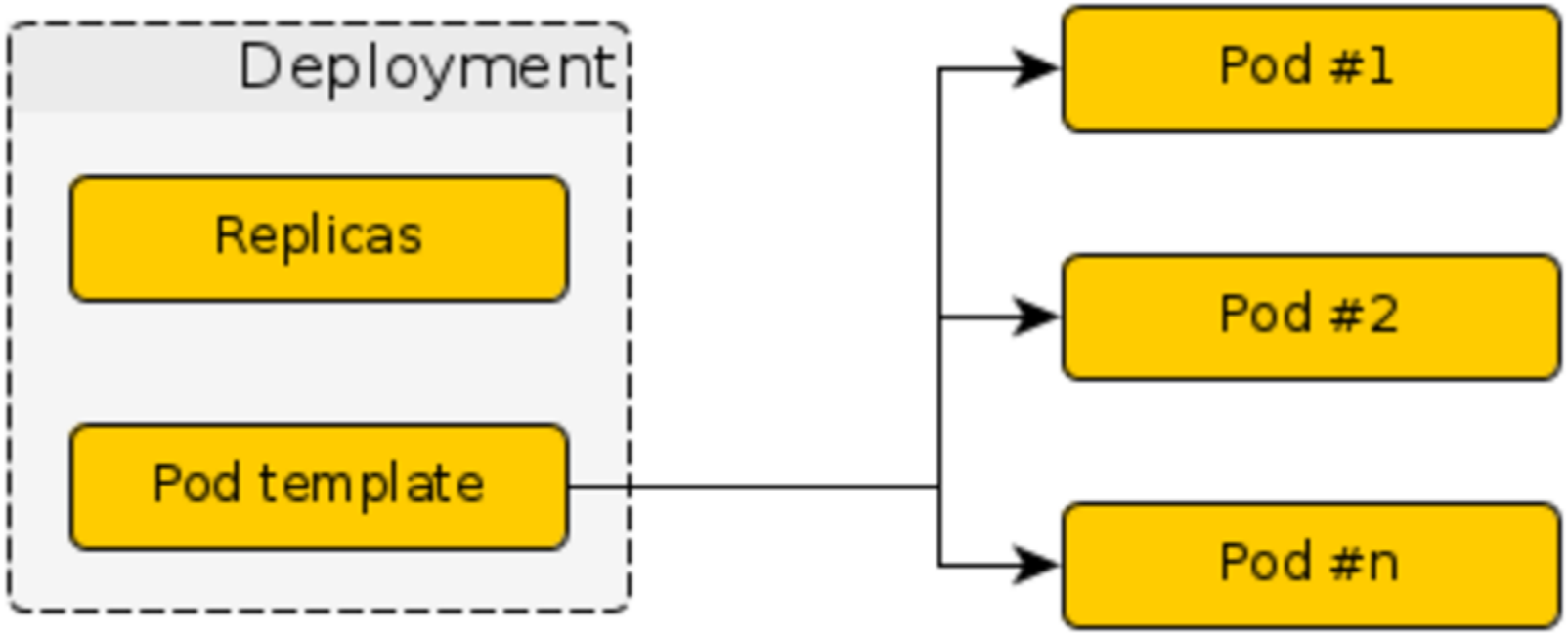
Kubernetes

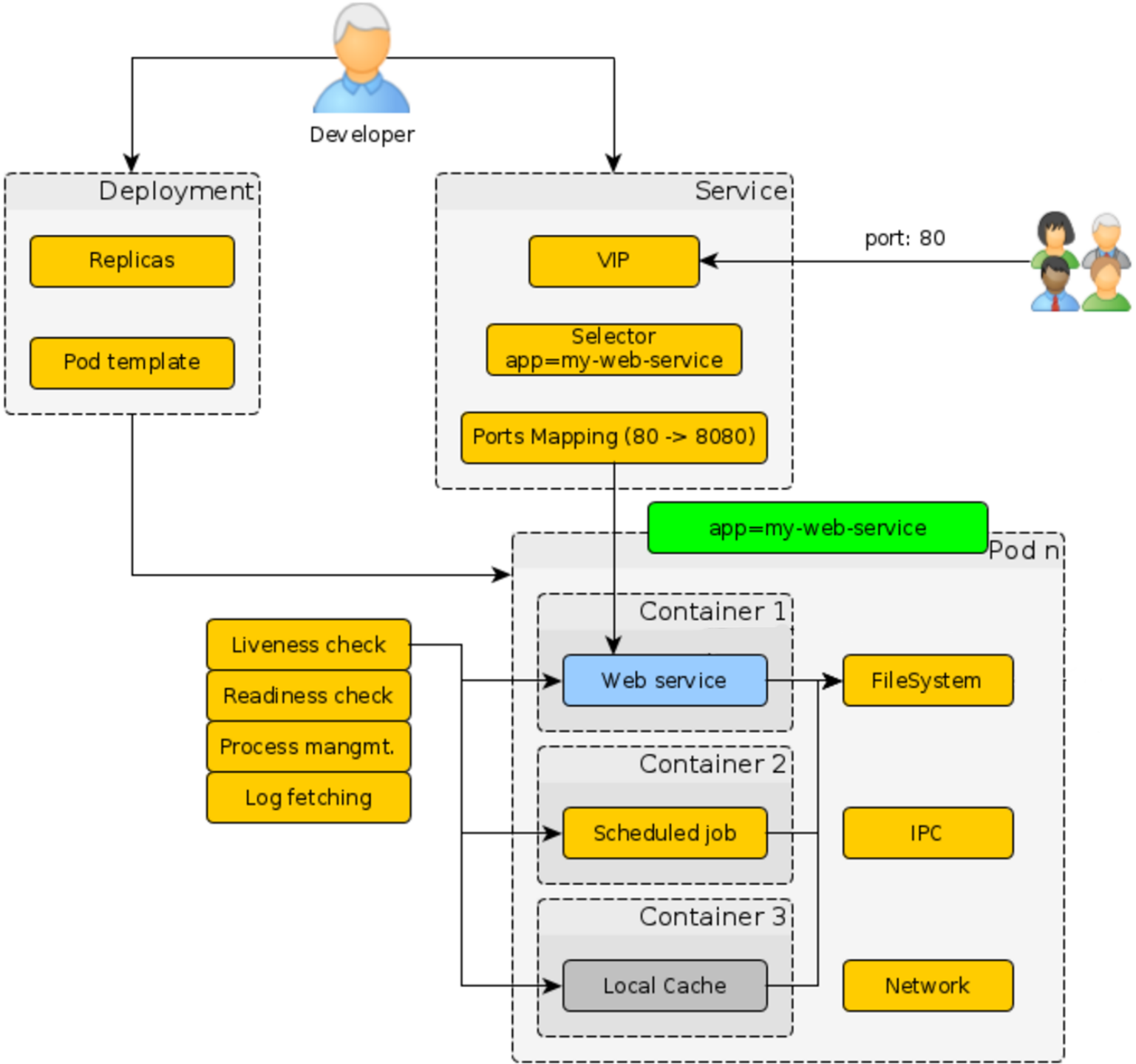


Kubernetes


Service
VIP
port: 80
Selector
app=my-web-service
Ports Mapping (80 -> 8080)
Pod1 (app=my-web-service)
Web service
8080
Pod2 (app=my-web-service)
Web service
8080
Pod3 (app=my-web-service)
Web service
8080

Kubernetes





```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: {{ app_name }}
spec:
  replicas: {{ replicas_count }}
  template:
    metadata:
      labels:
        app: {{ app_name }}
    spec:
      containers:
      - name: backend-conf
        image: {{ image_path }}:{{ image_version }}
        ports:
        - containerPort: {{ app_port }}
        readinessProbe:
          httpGet: { path: '{{ app_probe_path }}', port: {{ app_port }} }
          initialDelaySeconds: 10
          periodSeconds: 10
        env:
        - name: ASPNETCORE_ENVIRONMENT
          value: {{ env }}
```

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: {{ app_name }}
spec:
  replicas: {{ replicas_count }}
  template:
    metadata:
      labels:
        app: {{ app_name }}
    spec:
      containers:
      - name: backend-conf
        image: {{ image_path }}:{{ image_version }}
        ports:
        - containerPort: {{ app_port }}
        readinessProbe:
          httpGet: { path: '{{ app_probe_path }}', port: {{ app_port }} }
          initialDelaySeconds: 10
          periodSeconds: 10
        env:
        - name: ASPNETCORE_ENVIRONMENT
          value: {{ env }}
```

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: {{ app_name }}
spec:
  replicas: {{ replicas_count }}
  template:
    metadata:
      labels:
        app: {{ app_name }}
    spec:
      containers:
      - name: backend-conf
        image: {{ image_path }}:{{ image_version }}
        ports:
        - containerPort: {{ app_port }}
        readinessProbe:
          httpGet: { path: '{{ app_probe_path }}', port: {{ app_port }} }
          initialDelaySeconds: 10
          periodSeconds: 10
        env:
        - name: ASPNETCORE_ENVIRONMENT
          value: {{ env }}
```

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: {{ app_name }}
spec:
  replicas: {{ replicas_count }}
  template:
    metadata:
      labels:
        app: {{ app_name }}
    spec:
      containers:
      - name: backend-conf
        image: {{ image_path }}:{{ image_version }}
        ports:
        - containerPort: {{ app_port }}
        readinessProbe:
          httpGet: { path: '{{ app_probe_path }}', port: {{ app_port }} }
          initialDelaySeconds: 10
          periodSeconds: 10
        env:
        - name: ASPNETCORE_ENVIRONMENT
          value: {{ env }}
```

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: {{ app_name }}
spec:
  replicas: {{ replicas_count }}
  template:
    metadata:
      labels:
        app: {{ app_name }}
    spec:
      containers:
      - name: backend-conf
        image: {{ image_path }}:{{ image_version }}
        ports:
        - containerPort: {{ app_port }}
        readinessProbe:
          httpGet: { path: '{{ app_probe_path }}', port: {{ app_port }} }
          initialDelaySeconds: 10
          periodSeconds: 10
        env:
        - name: ASPNETCORE_ENVIRONMENT
          value: {{ env }}
```

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: {{ app_name }}
spec:
  replicas: {{ replicas_count }}
  template:
    metadata:
      labels:
        app: {{ app_name }}
    spec:
      containers:
      - name: backend-conf
        image: {{ image_path }}:{{ image_version }}
        ports:
        - containerPort: {{ app_port }}
        readinessProbe:
          httpGet: { path: '{{ app_probe_path }}', port: {{ app_port }} }
          initialDelaySeconds: 10
          periodSeconds: 10
        env:
        - name: ASPNETCORE_ENVIRONMENT
          value: {{ env }}
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ app_name }}
  annotations:
    router.deis.io/domains: "{{ app_name }}"
    router.deis.io/ssl.enforce: "{{ ssl_enforce | default('False') }}"
spec:
  ports:
    - name: http
      port: 80
      targetPort: {{ app_port }}
  selector:
    app: {{ app_name }}
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ app_name }}
  annotations:
    router.deis.io/domains: "{{ app_name }}"
    router.deis.io/ssl.enforce: "{{ ssl_enforce | default('False') }}"
spec:
  ports:
    - name: http
      port: 80
      targetPort: {{ app_port }}
  selector:
    app: {{ app_name }}
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ app_name }}
  annotations:
    router.deis.io/domains: "{{ app_name }}"
    router.deis.io/ssl.enforce: "{{ ssl_enforce | default('False') }}"
spec:
  ports:
    - name: http
      port: 80
      targetPort: {{ app_port }}
  selector:
    app: {{ app_name }}
```

```yaml
common:
  replicas_count: 1
  max_unavailable: 0

  k8s_master_uri: https://master.staging.dc-nsk1.hw:6443
  k8s_token: "{{ env='K8S_TOKEN_STAGE' }}"
  k8s_ca_base64: "{{ env='K8S_CA' }}"
  k8s_namespace: my-namespace
  ssl_enforce: true

  app_port: 5000
  app_probe_path: /healthcheck

  image_version: "{{ env='CI_TAG' }}"
  image_path: docker-hub.2gis.ru/my-namespace/backend-conf
  env: Stage

backend-conf-demo:
  app_name: “backend-conf-demo"

  app_limits_cpu: 500m
  app_requests_cpu: 100m
  app_limits_memory: 800Mi
  app_requests_memory: 300Mi

  kubectl:
  - template: deployment.yaml.j2
  - template: service-stage.yaml.j2
```

```yaml
common:
  replicas_count: 1
  max_unavailable: 0

  k8s_master_uri: https://master.staging.dc-nsk1.hw:6443
  k8s_token: "{{ env='K8S_TOKEN_STAGE' }}"
  k8s_ca_base64: "{{ env='K8S_CA' }}"
  k8s_namespace: my-namespace
  ssl_enforce: true

  app_port: 5000
  app_probe_path: /healthcheck

  image_version: "{{ env='CI_TAG' }}"
  image_path: docker-hub.2gis.ru/my-namespace/backend-conf
  env: Stage

backend-conf-demo:
  app_name: “backend-conf-demo"

  app_limits_cpu: 500m
  app_requests_cpu: 100m
  app_limits_memory: 800Mi
  app_requests_memory: 300Mi

  kubectl:
  - template: deployment.yaml.j2
  - template: service-stage.yaml.j2
```

```yaml
common:
  replicas_count: 1
  max_unavailable: 0

  k8s_master_uri: https://master.staging.dc-nsk1.hw:6443
  k8s_token: "{{ env='K8S_TOKEN_STAGE' }}"
  k8s_ca_base64: "{{ env='K8S_CA' }}"
  k8s_namespace: my-namespace
  ssl_enforce: true

  app_port: 5000
  app_probe_path: /healthcheck

  image_version: "{{ env='CI_TAG' }}"
  image_path: docker-hub.2gis.ru/my-namespace/backend-conf
  env: Stage

backend-conf-demo:
  app_name: “backend-conf-demo"

  app_limits_cpu: 500m
  app_requests_cpu: 100m
  app_limits_memory: 800Mi
  app_requests_memory: 300Mi

  kubectl:
  - template: deployment.yaml.j2
  - template: service-stage.yaml.j2
```

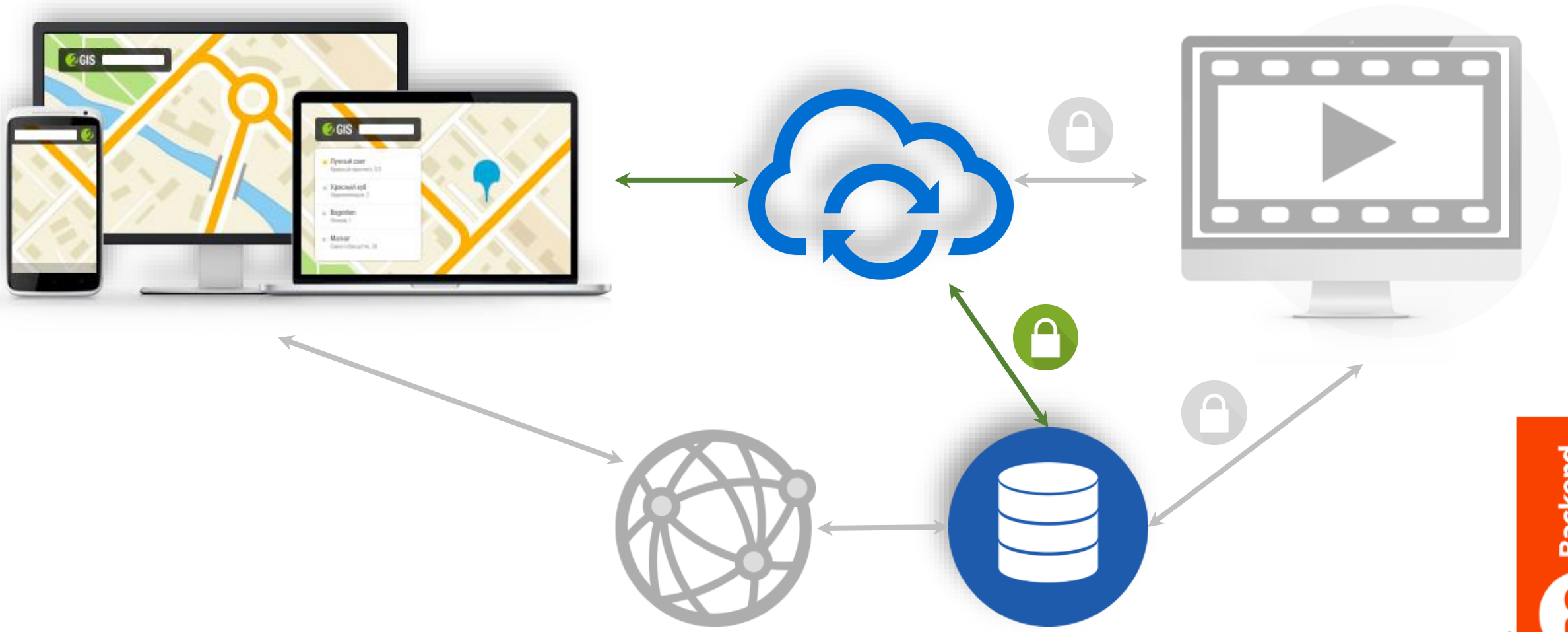
```yaml
common:
  replicas_count: 1
  max_unavailable: 0

  k8s_master_uri: https://master.staging.dc-nsk1.hw:6443
  k8s_token: "{{ env='K8S_TOKEN_STAGE' }}"
  k8s_ca_base64: "{{ env='K8S_CA' }}"
  k8s_namespace: my-namespace
  ssl_enforce: true

  app_port: 5000
  app_probe_path: /healthcheck

  image_version: "{{ env='CI_TAG' }}"
  image_path: docker-hub.2gis.ru/my-namespace/backend-conf
  env: Stage

backend-conf-demo:
  app_name: “backend-conf-demo"

  app_limits_cpu: 500m
  app_requests_cpu: 100m
  app_limits_memory: 800Mi
  app_requests_memory: 300Mi

  kubectl:
  - template: deployment.yaml.j2
  - template: service-stage.yaml.j2
```

```
deploy:backend-conf-demo-stage:
  stage: deploy:stage
  when: manual
  image: $REGISTRY/2gis-io/k8s-handle:latest
  script:
    - export ENVIRONMENT=Stage
    - k8s-handle deploy
      --config config-stage.yaml
      --section backend-conf --sync-mode True
  only:
    - tags
  tags: [ 2gis, docker ]
```

```
deploy:backend-conf-demo-stage:
  stage: deploy:stage
  when: manual
  image: $REGISTRY/2gis-io/k8s-handle:latest
  script:
    - export ENVIRONMENT=Stage
    - k8s-handle deploy
      --config config-stage.yaml
      --section backend-conf --sync-mode True
  only:
    - tags
  tags: [ 2gis, docker ]
```

- Коротко о сервисе
- On-premise платформа
- .NET Core, ASP.NET Core, базовые фичи
- Билд
- Деплой
- **Нагрузочное тестирование**
- Performance
  - Кэширование
  - Асинхронность и многопоточность

Нагрузочный контур



```scala
import Scenario._
import io.gatling.core.Predef._
import scala.concurrent.duration._
import scala.language.postfixOps

class LoadTest extends Simulation {
  val asserts = Seq(
    global.requestsPerSec.gte(17),
    global.failedRequests.count.is(0),
    details("Get").responseTime.percentile3.lte(700)
  )

  val injectionSteps = Seq(
    rampUsersPerSec(1) to 20 during (30 seconds),
    constantUsersPerSec(20) during (120 seconds)
  )

  setUp(scn().inject(injectionSteps).protocols(httpConf))
    .maxDuration(180 seconds)
    .assertions(asserts)
}
```

```scala
import Scenario._
import io.gatling.core.Predef._
import scala.concurrent.duration._
import scala.language.postfixOps

class LoadTest extends Simulation {
  val asserts = Seq(
    global.requestsPerSec.gte(17),
    global.failedRequests.count.is(0),
    details("Get").responseTime.percentile3.lte(700)
  )

  val injectionSteps = Seq(
    rampUsersPerSec(1) to 20 during (30 seconds),
    constantUsersPerSec(20) during (120 seconds)
  )

  setUp(scn().inject(injectionSteps).protocols(httpConf))
    .maxDuration(180 seconds)
    .assertions(asserts)
}
```

```scala
import Scenario._
import io.gatling.core.Predef._
import scala.concurrent.duration._
import scala.language.postfixOps

class LoadTest extends Simulation {
  val asserts = Seq(
    global.requestsPerSec.gte(17),
    global.failedRequests.count.is(0),
    details("Get").responseTime.percentile3.lte(700)
  )

  val injectionSteps = Seq(
    rampUsersPerSec(1) to 20 during (30 seconds),
    constantUsersPerSec(20) during (120 seconds)
  )

  setUp(scn().inject(injectionSteps).protocols(httpConf))
    .maxDuration(180 seconds)
    .assertions(asserts)
}
```

```scala
import Scenario._
import io.gatling.core.Predef._
import scala.concurrent.duration._
import scala.language.postfixOps

class LoadTest extends Simulation {
  val asserts = Seq(
    global.requestsPerSec.gte(17),
    global.failedRequests.count.is(0),
    details("Get").responseTime.percentile3.lte(700)
  )

  val injectionSteps = Seq(
    rampUsersPerSec(1) to 20 during (30 seconds),
    constantUsersPerSec(20) during (120 seconds)
  )

  setUp(scn().inject(injectionSteps).protocols(httpConf))
    .maxDuration(180 seconds)
    .assertions(asserts)
}
```

```yaml
.perf:template: &perf_template
  stage: test:perf
  environment: perf
  only:
    - master
    - /^perf.*$/
  variables:
    PERF_TEST_PATH: "tests/perf"
    PERF_ARTIFACTS: "target/gatling"
    PERF_GRAPHITE_HOST: "graphite-exporter.perf.os-n3.hw"
    PERF_GRAPHITE_ROOT_PATH_PREFIX: "gatling.service-prefix"
  image: $REGISTRY/perf/tools:1
  artifacts:
    name: perf-reports
    when: always
    expire_in: 7 day
    paths:
      - ${PERF_TEST_PATH}/${PERF_ARTIFACTS}/*
  tags: [perf-n3-1]
```

```
perf:run-tests:
  <<: *perf_template
  script:
    - export PERF_GRAPHITE_ROOT_PATH_PREFIX
PERF_GRAPHITE_HOST
    - export PERF_APP_HOST=http://${APP_PERF}.web-
staging.2gis.ru
    - cd ${PERF_TEST_PATH}
    - ./run_test.sh --capacity
    - ./run_test.sh --resp_time
  after_script:
    - perfberry-cli logs upload --dir
${PERF_TEST_PATH}/${PERF_ARTIFACTS} --env ${APP_PERF}.web-
staging.2gis.ru gatling ${PERFBERRY_PROJECT_ID}
```

```
perf:run-tests:
  <<: *perf_template
  script:
    - export PERF_GRAPHITE_ROOT_PATH_PREFIX
PERF_GRAPHITE_HOST
    - export PERF_APP_HOST=http://${APP_PERF}.web-
staging.2gis.ru
    - cd ${PERF_TEST_PATH}
    - ./run_test.sh --capacity
    - ./run_test.sh --resp_time
  after_script:
    - perfberry-cli logs upload --dir
${PERF_TEST_PATH}/${PERF_ARTIFACTS} --env ${APP_PERF}.web-
staging.2gis.ru gatling ${PERFBERRY_PROJECT_ID}
```

```
perf:run-tests:
  <<: *perf_template
  script:
    - export PERF_GRAPHITE_ROOT_PATH_PREFIX
PERF_GRAPHITE_HOST
    - export PERF_APP_HOST=http://${APP_PERF}.web-
staging.2gis.ru
    - cd ${PERF_TEST_PATH}
    - ./run_test.sh --capacity
    - ./run_test.sh --resp_time
  after_script:
    - perfberry-cli logs upload --dir
${PERF_TEST_PATH}/${PERF_ARTIFACTS} --env ${APP_PERF}.web-
staging.2gis.ru gatling ${PERFBERRY_PROJECT_ID}
```

- Коротко о сервисе
- On-premise платформа
- .NET Core, ASP.NET Core, базовые фичи
- Билд
- Деплой
- Нагрузочное тестирование
- **Performance**
  - Кэширование
  - Асинхронность и многопоточность

Кеширование

```
[AllowAnonymous]
[HttpGet("{id}")]
[ResponseCache(
   VaryByQueryKeys = new[] { "api-version" },
   Duration = 3600)]
public async Task<IActionResult> Get(long id)
{
   …
}
```

Кеширование

```
[ResponseCache(
    VaryByQueryKeys = new[] { "api-version" },
    Duration = 3600)]
```

- На клиенте
  - Cache-Control header (HTTP 1.1 Caching)

Кеширование

```
[ResponseCache(
  VaryByQueryKeys = new[] { "api-version" },
  Duration = 3600)]
```

- На клиенте
  - Cache-Control header (HTTP 1.1 Caching)
- На сервере
  - Response Caching Middleware (docs)

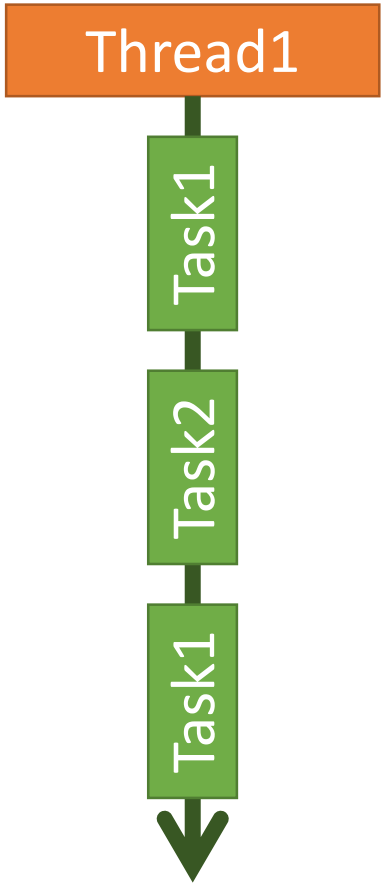
Performance

Асинхронность

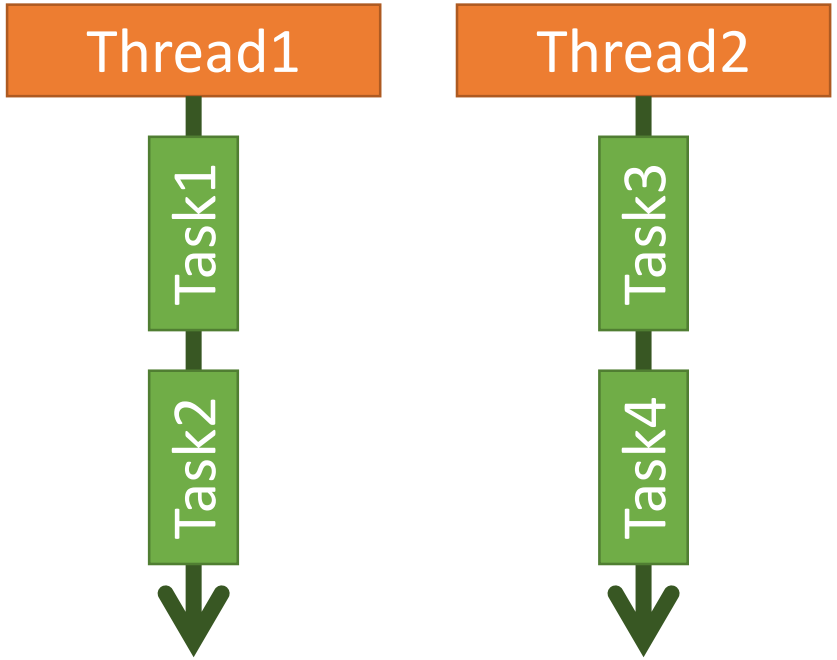
Многопоточность

Performance

Асинхронность



Многопоточность



```
var data =
 await _remoteService.IOBoundOperationAsync(timeoutInSec: 1);
var result = new List<string>[data.Count];

foreach (var item in data)
{
  var detailed =
    await _remoteService.IOBoundOperationAsync(timeoutInSec: 5);
  result.Add(string.Join(", ", detailed));
}
```

```
var data =
 await _remoteService.IOBoundOperationAsync(timeoutInSec: 1);
var result = new List<string>[data.Count];

foreach (var item in data)
{
  var detailed =
    await _remoteService.IOBoundOperationAsync(timeoutInSec: 5);
  result.Add(string.Join(", ", detailed));
}
```

```
var data =
 await _remoteService.IOBoundOperationAsync(timeoutInSec: 1);
var result = new string[data.Count];

var tasks = data.Select(
 async (item, index) =>
  {
   var detailed =
    await _remoteService.IOBoundOperationAsync(timeoutInSec: 5);
   result[index] = string.Join(", ", detailed)
  });

await Task.WhenAll(tasks);
```

```
var data =
 await _remoteService.IOBoundOperationAsync(timeoutInSec: 1);
var result = new string[data.Count];

var tasks = data.Select(
 async (item, index) =>
  {
   var detailed =
    await _remoteService.IOBoundOperationAsync(timeoutInSec: 5);
   result[index] = string.Join(", ", detailed)
  });

await Task.WhenAll(tasks);
```

```
var data =
 await _remoteService.IOBoundOperationAsync(timeoutInSec: 1);
var result = new string[data.Count];

var tasks = data.Select(
 async (item, index) =>
  {
   var detailed =
    await _remoteService.IOBoundOperationAsync(timeoutInSec: 5);
   result[index] = string.Join(", ", detailed)
  });

await Task.WhenAll(tasks);
```

Нагрузочное тестирование

- Лимиты по памяти и процессору
  - 384Mb и 1,5 CPU

Нагрузочное тестирование

- Лимиты по памяти и процессору
  - 384Mb и 1,5 CPU
- Синхронный (capacity) тест
  - ~24 RPS (без кэша)

Нагрузочное тестирование

- Лимиты по памяти и процессору
  - 384Mb и 1,5 CPU
- Синхронный (capacity) тест
  - ~24 RPS (без кэша)
  - ~400 RPS (включен серверный кэш)

Нагрузочное тестирование

- Лимиты по памяти и процессору
  - 384Mb и 1,5 CPU
- Синхронный (capacity) тест
  - ~24 RPS (без кэша)
  - ~400 RPS (включен серверный кэш)

- Асинхронный (load) тест
  - Прошёл

Вместо заключения

Вместо заключения

- Не бойтесь использовать .NET Core в продакшене

Вместо заключения

- Не бойтесь использовать .NET Core в продакшене
- Не бойтесь использовать Linux и .NET Core

Вместо заключения

- Не бойтесь использовать .NET Core в продакшене
- Не бойтесь использовать Linux и .NET Core
- Docker и Kubernetes сильно упрощают жизнь

Вместо заключения

- Не бойтесь использовать .NET Core в продакшене
- Не бойтесь использовать Linux и .NET Core
- Docker и Kubernetes сильно упрощают жизнь
- Оптимизируйте приложения

Вместо заключения

- Не бойтесь использовать .NET Core в продакшене
- Не бойтесь использовать Linux и .NET Core
- Docker и Kubernetes сильно упрощают жизнь
- Оптимизируйте приложения
  - Многое есть из коробки
  - Думать все равно надо

Спасибо!

https://github.com/denisivan0v/backend-conf-2017

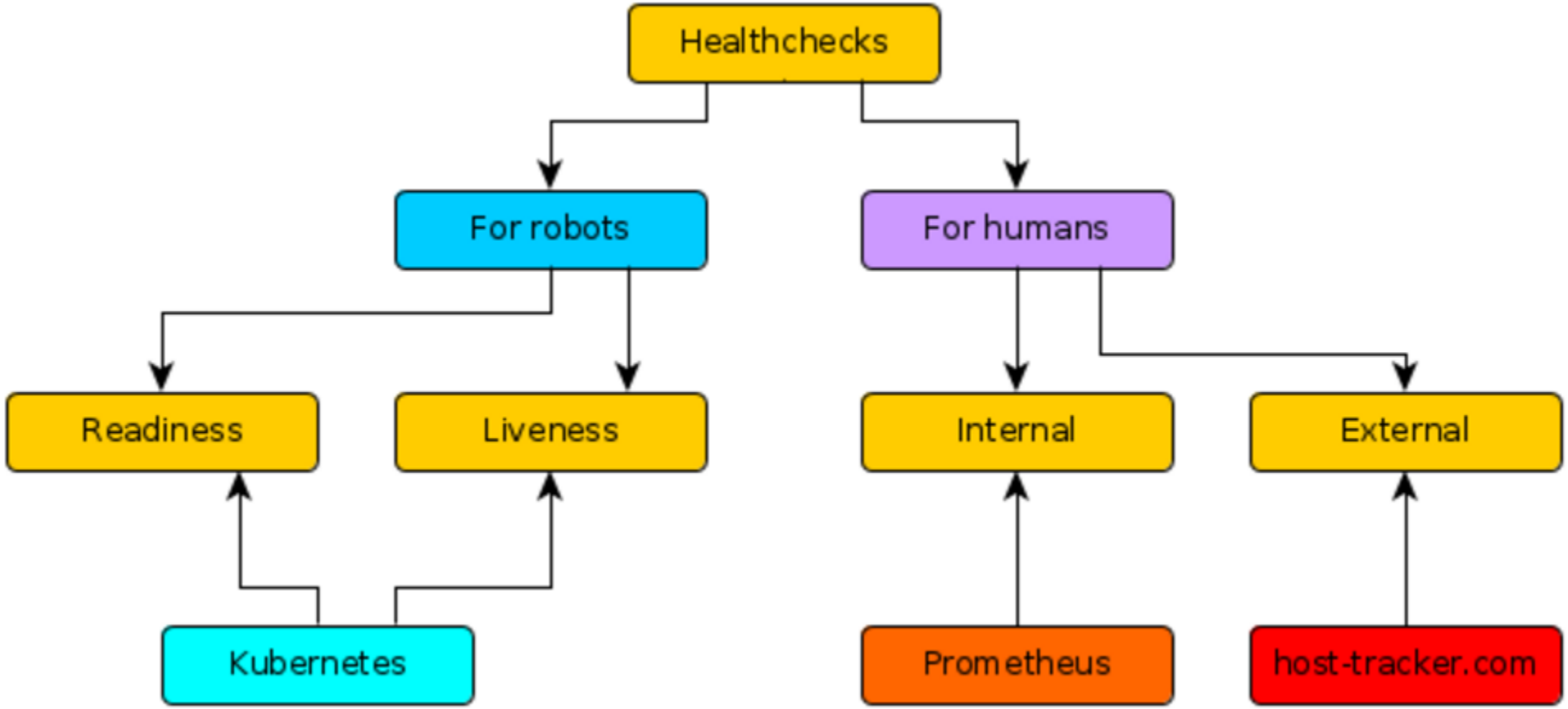
Вопросы?

Денис Иванов

@denisivanov

denis@ivanovdenis.ru
https://github.com/denisivan0v

Эксплуатация



Эксплуатация

- Prometheus server (/metrics)