ИННОТЕХ

ИСПОЛЬЗОВАНИЕ HELM-ЧАРТОВ БЕЗ НАПИСАНИЯ HELM-ЧАРТОВ

КТО Я

РУСЛАН ГАЙНАНОВ

Ведущий инженер DevOps, ГК «Иннотех»

- 5+ лет в devops и IT
- 4+ лет с Kubernetes (CKA/CKAD certified)
- 100+ различных микросервисов
- 20+ различных пайплайнов
- Несколько раз ронял прод

О ЧЁМ ПОГОВОРИМ

- Вызовы
- Типы helm-чартов
- Универсальный helm-чарт
- Результаты
- Open-source
- Выводы

ВЫЗОВЫ

ВЫЗОВЫ

ИННОТЕХ

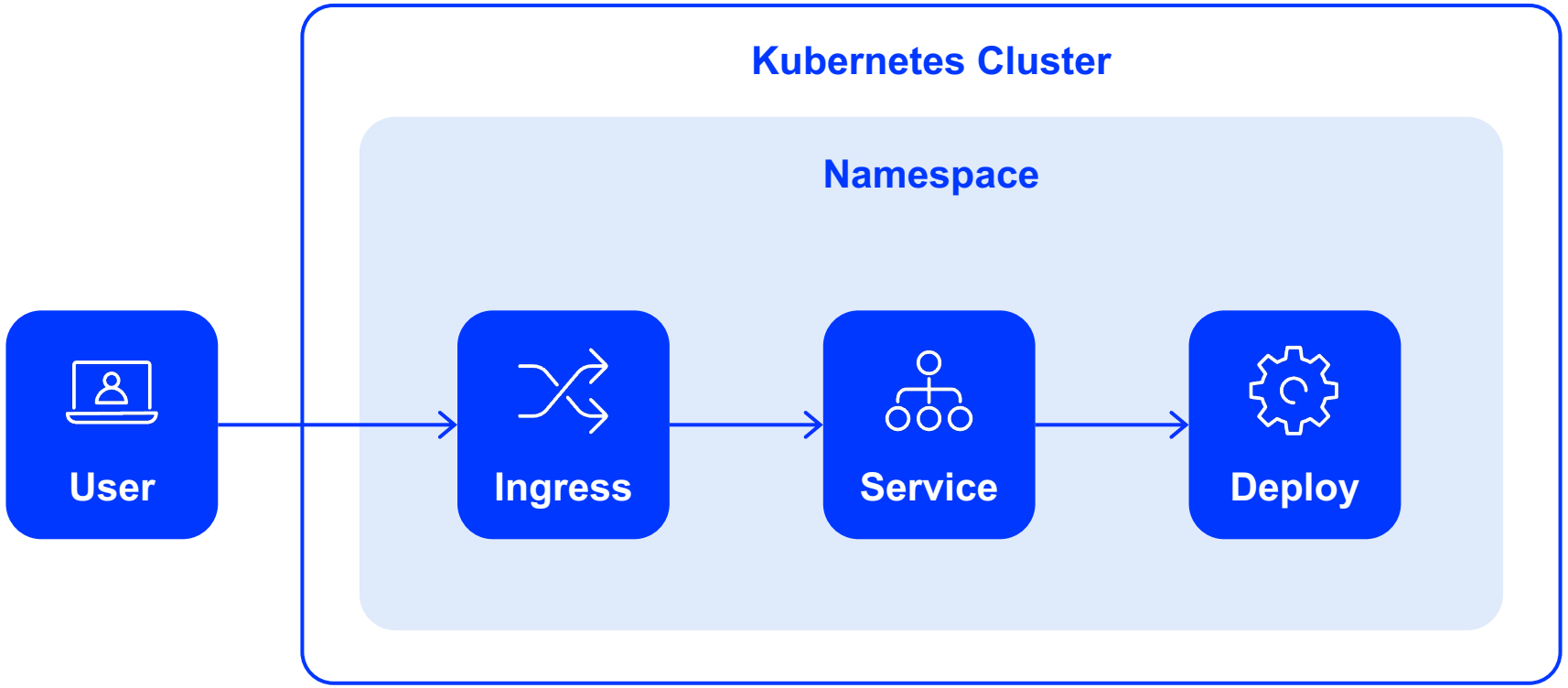
Как легко доставлять код множества различных приложений до прода?

Как следовать подходу DRY (Don’t-Repeat-Yourself)?

Как обеспечить гибкость настроек?

Используемый нами тех.стек:

ПРОСТОЙ МИКРОСЕРВИС В K8S



ИННОТЕХ

МАНИФЕСТЫ K8S ДЛЯ ПРОСТОГО МИКРОСЕРВИСА

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-backend
spec:
  selector:
    matchLabels:
      app: app-backend
  template:
    metadata:
      labels:
        app: app-backend
    spec:
      containers:
      - name: app-backend
        image: app-backend:${TAG}
        ports:
        - containerPort: 8080
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: app-backend
  labels:
    name: app-backend
spec:
  selector:
    name: app-backend
  type: ClusterIP
  ports:
    - port: 80
      protocol: TCP
      targetPort: 8080
```

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: app-backend
  labels:
    name: app-backend
spec:
  rules:
  - host: ${URL}
    http:
      paths:
      - pathType: Prefix
        path: /
        backend:
          service:
            name: app-backend
            port:
              number: 80
```

ИННОТЕХ

МАНИФЕСТЫ K8S ДЛЯ ПРОСТОГО МИКРОСЕРВИСА

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-backend
spec:
  selector:
    matchLabels:
      name: app-backend
  template:
    metadata:
      labels:
        name: app-backend
    spec:
      containers:
      - name: app-backend
        image: app-backend:${TAG}
        ports:
        - containerPort: 8080
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: app-backend
  labels:
    name: app-backend
spec:
  selector:
    name: app-backend
  type: ClusterIP
  ports:
    - port: 80
      protocol: TCP
      targetPort: 8080
```

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: app-backend
  labels:
    name: app-backend
spec:
  rules:
  - host: ${URL}
    http:
      paths:
      - pathType: Prefix
        path: /
        backend:
          service:
            name: app-backend
            port:
              number: 80
```

ИННОТЕХ

МАНИФЕСТЫ K8S ДЛЯ ПРОСТОГО МИКРОСЕРВИСА

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-backend
spec:
  selector:
    matchLabels:
      name: app-backend
  template:
    metadata:
      labels:
        name: app-backend
    spec:
      containers:
      - name: app-backend
        image: app-backend:${TAG}
        ports:
        - containerPort: 8080
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: app-backend
  labels:
    name: app-backend
spec:
  selector:
    name: app-backend
  type: ClusterIP
  ports:
    - port: 80
      protocol: TCP
      targetPort: 8080
```

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: app-backend
  labels:
    name: app-backend
spec:
  rules:
  - host: ${URL}
    http:
      paths:
      - pathType: Prefix
        path: /
        backend:
          service:
            name: app-backend
            port:
              number: 80
```

ШАБЛОНЫ HELM ДЛЯ ПРОСТОГО МИКРОСЕРВИСА

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "app-backend.fullname" . }}
  labels:
    {{- include "app-backend.labels" . | nindent 4 }}
spec:
  selector:
    matchLabels:
      {{- include "app-backend.selectorLabels" . | nindent 6 }}
  template:
    metadata:
      labels:
        {{- include "app-backend.selectorLabels" . | nindent 8 }}
    spec:
      containers:
        - name: {{ .Chart.Name }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .Chart.AppVersion }}"
          ports:
            - containerPort: {{ .Values.service.port }}
```

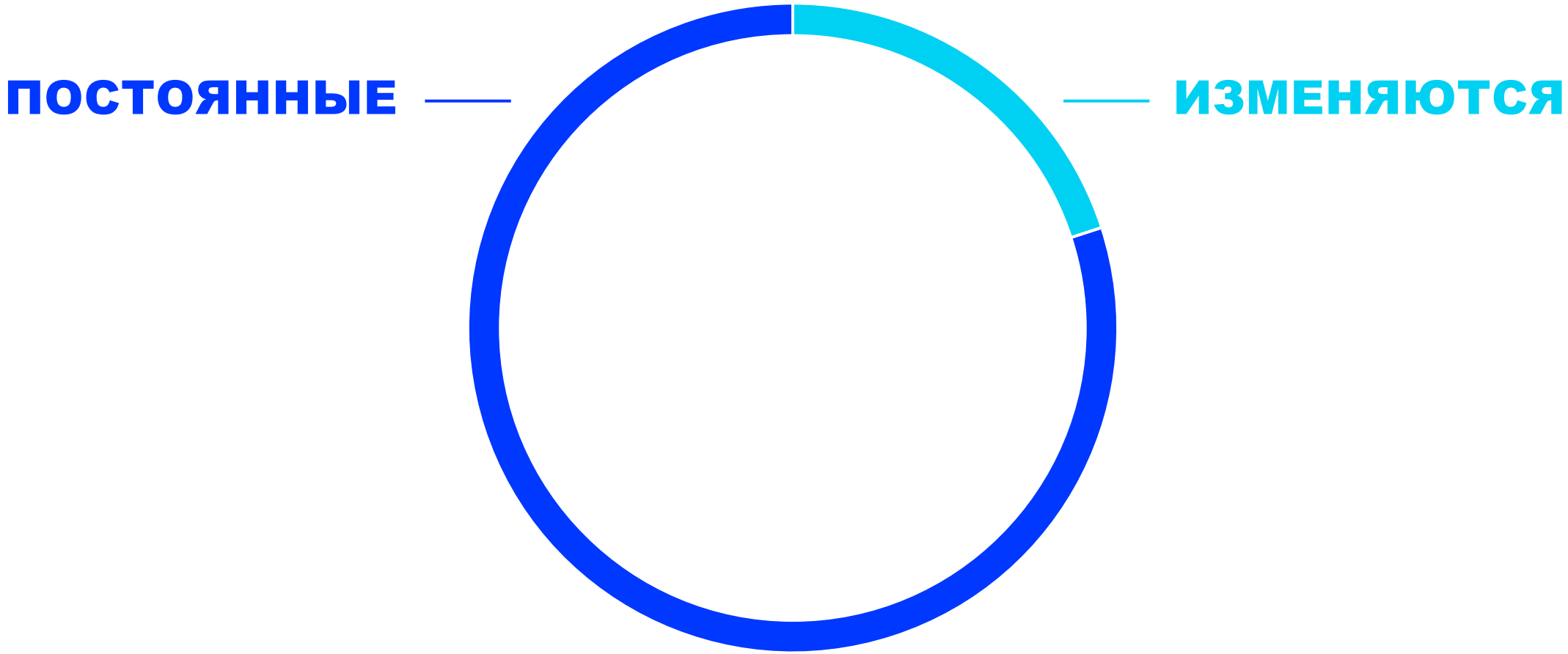
ШАБЛОНЫ HELM ДЛЯ ПРОСТОГО МИКРОСЕРВИСА

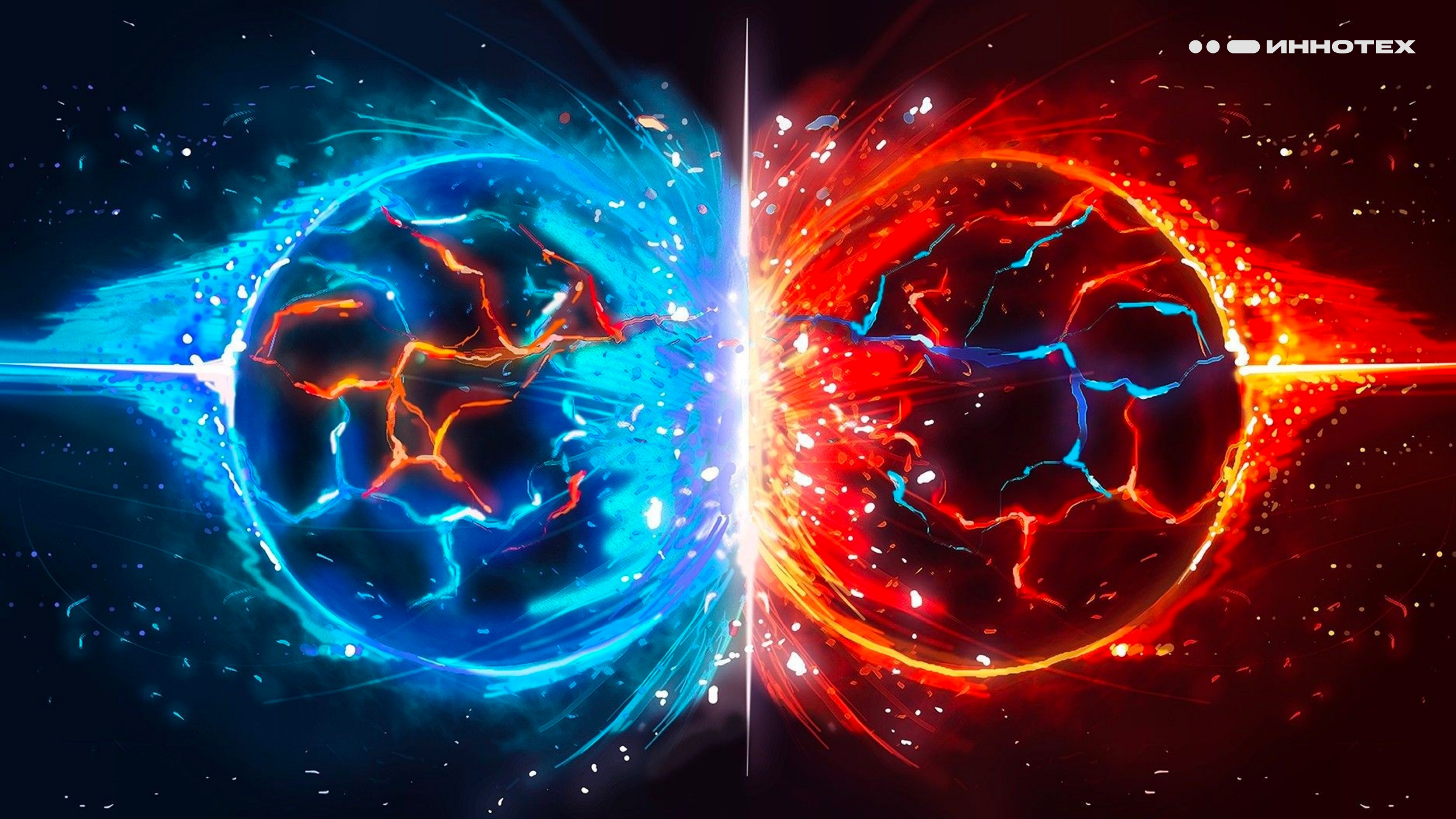
```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ include "app-backend.fullname" . }}
  labels:
    {{- include "app-backend.labels" . | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: {{ .Values.service.port }}
      protocol: TCP
  selector:
    {{- include "app-backend.selectorLabels" . | nindent 4 }}
```

ШАБЛОНЫ HELM ДЛЯ ПРОСТОГО МИКРОСЕРВИСА

```
{{- if .Values.ingress.enabled -}}
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: {{ include "app-backend.fullname" }}
  labels:
    {{- include "app-backend.labels" . | nindent 4 }}
spec:
  rules:
    {{- range .Values.ingress.hosts }}
    - host: {{ .host | quote }}
      http:
        paths:
          {{- range .paths }}
          - path: {{ .path }}
            backend:
              service:
                name: {{ include "app-backend.fullname" }}
                port:
                  number: {{ .Values.service.port }}
          {{- end }}
    {{- end }}
{{- end }}
```

ПРОЦЕНТ ИЗМЕНЯЕМЫХ ПАРАМЕТРОВ МЕЖДУ ПРИЛОЖЕНИЯМИ




ИННОТЕХ

ТИПЫ HELM-ЧАРТОВ

Application & library

HELM-ЧАРТ ПРИЛОЖЕНИЯ (APPLICATION CHART)

Классический helm-чарт содержащий шаблоны манифестов создаваемых ресурсов Kubernetes для некоторого **приложения** с их параметрами (values).

Используется для деплоя через создание релизов:

```
helm install my-release ./backend

NAME: my-release
LAST DEPLOYED: Mon Aug  7 19:15:05 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
```

```
backend/
  Chart.yaml
  values.yaml
  templates/
  templates/deployment.yaml
  templates/service.yaml
  templates/ingress.yaml
  templates/NOTES.txt
_____________________________
# Chart.yaml (application)
name: backend
type: application
version: 1.0.0
```

ОБЩИЙ HELM-ЧАРТ (LIBRARY CHART)

Это чарт с **общими шаблонами** создаваемых ресурсов Kubernetes или их частей.

Не может быть использован при деплое и создании релиза напрямую.

```
common/
  Chart.yaml
  values.yaml
  templates/
  templates/_names.tpl
  templates/_labels.tpl
  templates/_tplvalues.tpl
  templates/_utils.tpl
  templates/...
_____________________________________

# Chart.yaml (library)
name: common
type: library
version: 1.0.0
```

ОБЩИЙ HELM-ЧАРТ (LIBRARY CHART)

Это чарт с **общими шаблонами** создаваемых ресурсов Kubernetes или их частей.

Не может быть использован при деплое и создании релиза напрямую.

```
helm install my-release ./common

Error: INSTALLATION FAILED: library
charts are not installable
```

```
common/
  Chart.yaml
  values.yaml
  templates/
  templates/_names.tpl
  templates/_labels.tpl
  templates/_tplvalues.tpl
  templates/_utils.tpl
  templates/...
---------------------------------

# Chart.yaml (library)
name: common
type: library
version: 1.0.0
```

ОБЩИЙ HELM-ЧАРТ (LIBRARY CHART)

Это чарт с **общими шаблонами** создаваемых ресурсов Kubernetes или их частей.

Не может быть использован при деплое и создании релиза напрямую.

Подключается как **зависимость** к чартам приложений.

```yaml
# Chart.yaml (library)
name: common
type: library
version: 1.0.0

_____________________________________
# Chart.yaml (application)
name: backend
type: application
version: 1.0.0

dependencies:
- name: common
  version: ^1.0.0
  repository: "@dbp"
```

КАК ЭТО РАБОТАЕТ

Общий шаблон

```
# common/templates/_labels.tpl
{{- define "common.labels.standard" -}}
app.kubernetes.io/name: {{ include "common.names.name" . }}
helm.sh/chart: {{ include "common.names.chart" . }}
app.kubernetes.io/instance: {{ .Release.Name }}
app.kubernetes.io/managed-by: {{ .Release.Service }}
{{- end -}}
```

https://github.com/bitnami/charts/blob/main/bitnami/common/templates/_labels.tpl

КАК ЭТО РАБОТАЕТ

Использование шаблона

```
# backend/templates/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ template "common.names.fullname" . }}
  labels: {{- include "common.labels.standard" . | nindent 4 }}
spec:
  ...
```

https://github.com/bitnami/charts/blob/main/bitnami/common/templates/_labels.tpl

КАК ЭТО РАБОТАЕТ

Результат

```yaml
# backend/templates/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-backend
  labels:
    app.kubernetes.io/instance: app-backend
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: app-backend
    helm.sh/chart: universal-chart-2.1.3
spec:
  ...
```

https://github.com/bitnami/charts/blob/main/bitnami/common/templates/_labels.tpl

ВАРИАНТЫ ГОТОВЫХ РЕШЕНИЙ

ОБЩИЙ HELM-ЧАРТ ОТ BITNAMI: ОПИСАНИЕ

Включает в себя функции:

- Получения имен ресурсов, стандартных меток и селекторов
- Настройки affinities (hard & soft, pods & nodes)
- Создания и извлечения данных секретов
- Различные вспомогательные функции (tplvalues.Render)

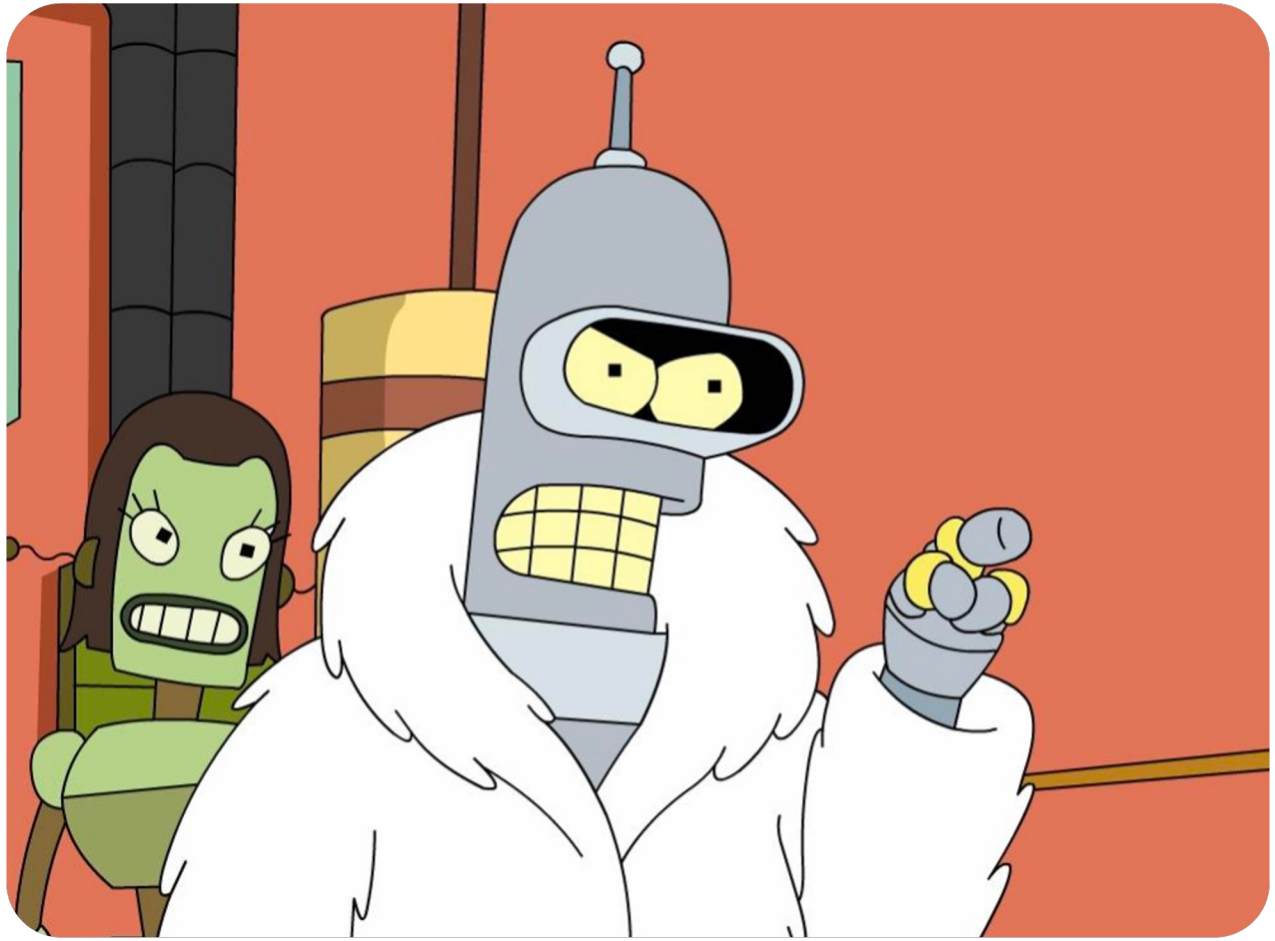
Подключение:

```
dependencies:
  - name: common
    repository: oci://registry-1.docker.io/bitnamicharts
    tags:
      - bitnami-common
    version: 2.x.x
```

ОБЩИЙ HELM-ЧАРТ ОТ BITNAMI: ИСПОЛЬЗОВАНИЕ

```
# backend/templates/configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ template "common.names.fullname" . }}
  namespace: {{ include "common.names.namespace" . | quote }}
  labels: {{- include "common.labels.standard" . | nindent 4 }}
  annotations:
    {{- include "common.tplvalues.render" ( dict "value" .Values.commonAnnotations "context" $ ) }}
spec:
  ...
```

НАШ LIBRARY-ЧАРТ: DBP-COMMON



НАШ LIBRARY-ЧАРТ: DBP-COMMON

Всё что есть в bitnami/common

ИННОТЕХ

НАШ LIBRARY-ЧАРТ - DBP-COMMON

Всё что есть в bitnami/common

Функции создания полных манифестов для типовых app: deploy, svc, ingress

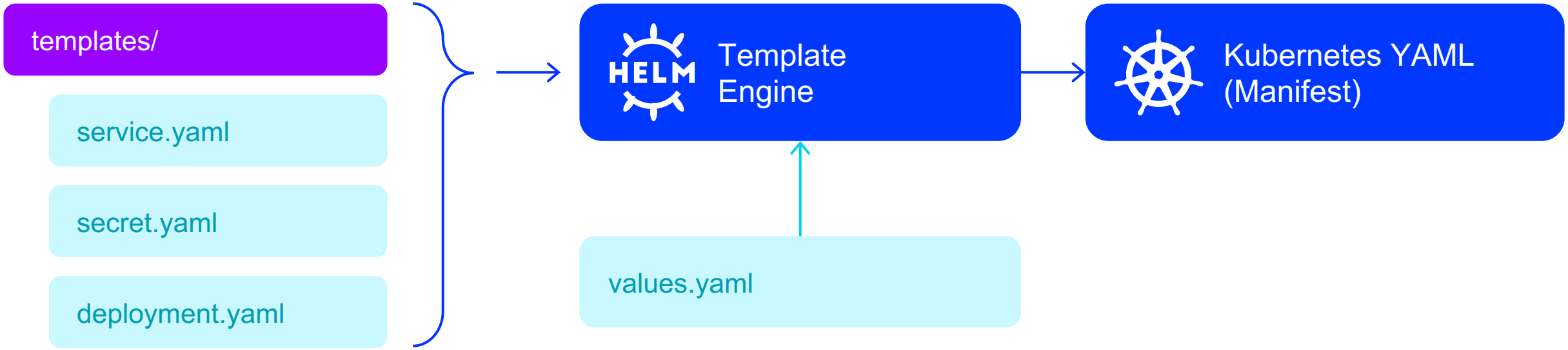
```
# backend/templates/deployment.yaml
{{ include "dbp-common.backend.deployment" . }}

# backend/templates/service.yaml
{{ include "dbp-common.backend.service" . }}

# backend/templates/ingress.yaml
{{ include "dbp-common.backend.ingress" . }}
```

```yaml
# backend/values.yaml
ingress:
  prefix: "demo-app"
  paths:
    - path: /
deployment:
  image:
    name: dbp/demo-app
appConfig:
  ports:
    main: 8080
```

ИННОТЕХ

КАКИЕ ПРОБЛЕМЫ ПРИ ЭТОМ ОСТАЛИСЬ?

Всё-равно необходим отдельный helm-чарт для приложений



КАКИЕ ПРОБЛЕМЫ ПРИ ЭТОМ ОСТАЛИСЬ?

Структура параметров внутри values не всегда очевидная

```
# cerebro

{{- if .Values.image.pullSecrets }}
imagePullSecrets:
{{- range .Values.image.pullSecrets }}
- name: {{ . }}
{{- end }}
{{- end }}
```

```
# sonarqube

{{- if .Values.image.pullSecret }}
imagePullSecrets:
- name: {{ .Values.image.pullSecret }}
{{- end }}
```

```
# prometheus

{{- if .Values.imagePullSecrets }}
imagePullSecrets:
{{ toYaml .Values.imagePullSecrets
    | indent 2 }}
{{- end }}
```

КАКИЕ ПРОБЛЕМЫ ПРИ ЭТОМ ОСТАЛИСЬ?

Отсутствуют необходимые функции в готовых library-чартах

КАКИЕ ПРОБЛЕМЫ ПРИ ЭТОМ ОСТАЛИСЬ?

Сложно обеспечить обратную совместимость изменений и отладку

```
# helm upgrade --install -f values.yaml --timeout 600 --wait backend ./helm/chart

UPGRADE FAILED

Error: timed out waiting for the condition

  Error: UPGRADE FAILED: timed out waiting for the condition
```

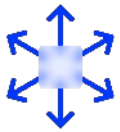
HELM-CHART

Helm-чарт для упаковки и деплоя приложения

UNIVERSAL-CHART

Универсальный Helm-чарт для деплоя различных приложений

ПРЕИМУЩЕСТВА И ВОЗМОЖНОСТИ



Упаковка различных приложений, используя один общий чарт

- Изменяется только values
- Без публикации helm-чартов (helm package & helm push)

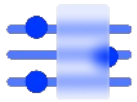
Переопределение параметров ресурсов k8s

Удобно при использовании общего umbrella-chart

- Через секцию global
- Через аргумент --set-string

Добавление своих шаблонов и манифестов

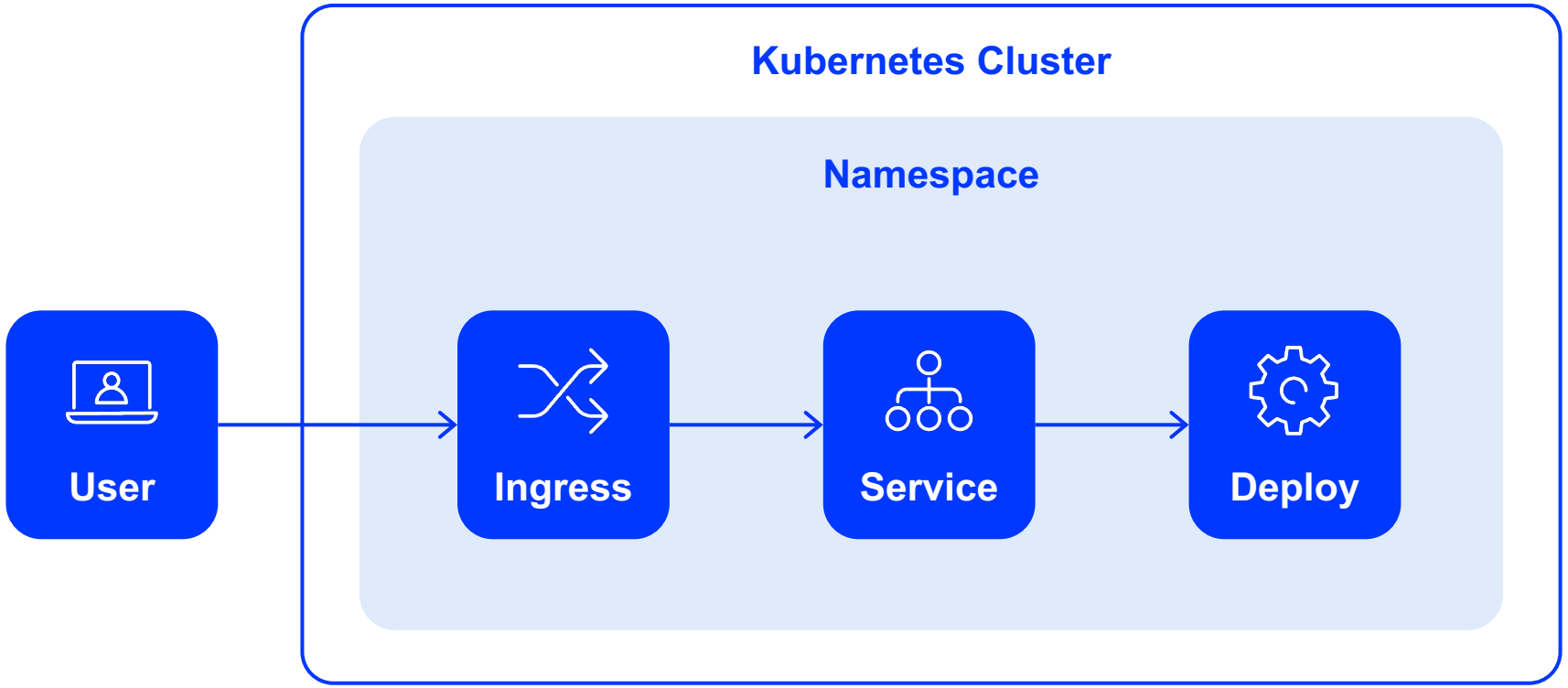
- Через PR в проект universal-chart
- Подключив universal-chart как dependency
- Через секцию настроек extraDeploy



Централизованное управление именами ресурсов

Не требуется менять инструменты

КАК ЭТО ИСПОЛЬЗОВАТЬ (ВАРИАНТ 1)



КАК ЭТО ИСПОЛЬЗОВАТЬ (ВАРИАНТ 1)

1. Пример: traefik/whoami

КАК ЭТО ИСПОЛЬЗОВАТЬ (ВАРИАНТ 1)

1. **Пример: traefik/whoami**

2. **Файл с описанием создаваемых ресурсов: values.yaml**
   - Deployments
   - Services
   - Ingresses

```yaml
deployments:
  whoami:
    containers:
      whoami:
        image: traefik/whoami
        imageTag: v1.10.1
        ports:
          - name: web
            containerPort: 80
```

ИННОТЕХ

КАК ЭТО ИСПОЛЬЗОВАТЬ (ВАРИАНТ 1)

1. **Пример: traefik/whoami**

2. **Файл с описанием создаваемых ресурсов: values.yaml**
   - Deployments
   - Services
   - Ingresses

```yaml
services:
  whoami:
    ports:
      web:
        protocol: TCP
        port: 80
        targetPort: 80
```

КАК ЭТО ИСПОЛЬЗОВАТЬ (ВАРИАНТ 1)

1. **Пример: traefik/whoami**

2. **Файл с описанием создаваемых ресурсов: values.yaml**
   - Deployments
   - Services
   - Ingresses

```yaml
ingresses:
  whoami-127-0-0-1.nip.io:
    hosts:
    - paths:
      - serviceName: whoami
        servicePort: web
```

КАК ЭТО ИСПОЛЬЗОВАТЬ (ВАРИАНТ 1)

1. **Пример: traefik/whoami**

2. **Файл с описанием создаваемых ресурсов: values.yaml**
   - Deployments
   - Services
   - Ingresses

3. **Запустить установку**

```
helm repo add gromr1 https://gromr1.github.io/nxs-universal-chart
helm repo update
helm install inno gromr1/universal-chart --values values.yaml
```

```
NAME: inno
LAST DEPLOYED: Thu Aug 24 14:20:09 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

ИННОТЕХ

КАК ЭТО ИСПОЛЬЗОВАТЬ (ВАРИАНТ 1)

1. **Пример: traefik/whoami**
2. **Файл с описанием создаваемых ресурсов: values.yaml**
   - Deployments
   - Services
   - Ingresses
3. **Запустить установку**
4. **Проверить**

```
curl whoami-127-0-0-1.nip.io:8080
```

```
Hostname: inno-whoami-6d7f995ff9-dqj69
IP: 127.0.0.1
IP: ::1
IP: 10.244.0.16
IP: fe80::3c17:35ff:fef1:2da2
RemoteAddr: 10.244.0.12:52210
GET / HTTP/1.1
Host: whoami-127-0-0-1.nip.io:8080
User-Agent: curl/7.87.0
Accept: */*
X-Forwarded-For: 127.0.0.1
X-Forwarded-Host: whoami-127-0-0-1.nip.io:8080
X-Forwarded-Port: 80
X-Forwarded-Proto: http
X-Forwarded-Scheme: http
X-Real-Ip: 127.0.0.1
X-Request-Id: b4330aa0e5b61cf75eb954e222490444
X-Scheme: http
```

ЧТО ЕЩЕ ВОЗМОЖНО (ПРИМЕРЫ)

```yaml
version: "1.0"

jobs:
  test1:
    containers:
    - name: test
      image: testA
      imageTag: "{{ .Values.version }}"
  test2:
    containers:
    - name: test
      image: testB
      imageTag: "{{ .Values.version }}"
```

https://github.com/GRomR1/nxs-universal-chart/tree/dev/samples

ЧТО ЕЩЕ ВОЗМОЖНО (ПРИМЕРЫ)

```yaml
envs:
  FOO: bar
secretEnvs:
  BAR: foo

deployments:
  whoami:
    containers:
      whoami:
        image: traefik/whoami
        imageTag: v1.10.1
        envSecrets:
        - secret-envs
        envConfigmaps:
        - envs
```

https://github.com/GRomR1/nxs-universal-chart/tree/dev/samples

ЧТО ЕЩЕ ВОЗМОЖНО (ПРИМЕРЫ)

```yaml
services:
  whoami:
    ports:
    - name: web
      protocol: TCP
      port: 80
      targetPort: 8080

serviceMonitors:
  whoami-web:
    endpoints:
    - interval: 30s
      port: web
      path: /
```

https://github.com/GRomR1/nxs-universal-chart/tree/dev/samples

ЧТО ЕЩЕ ВОЗМОЖНО (ПРИМЕРЫ)

```yaml
ingresses:
  whoami-127-0-0-1.nip.io:
    hosts:
    - paths:
      - serviceName: whoami-web
        servicePort: web
    certManager:
      issuerType: issuer
      issuerName: selfsigned-ca-issuer

issuers:
  selfsigned-issuer:
    kind: ClusterIssuer
    selfSigned: {}
  selfsigned-ca-issuer:
    ca:
      secretName: selfsigned-ca
```

https://github.com/GRomR1/nxs-universal-chart/tree/dev/samples

ЧТО ЕЩЕ ВОЗМОЖНО (ПРИМЕРЫ)

```yaml
extraDeploy:
  net-pol: |-
    apiVersion: networking.k8s.io/v1
    kind: NetworkPolicy
    metadata:
      name: access-nginx
    spec:
      podSelector:
        matchLabels:
          app: nginx
      ingress:
      - from:
        - podSelector:
            matchLabels:
              access: "true"
```

https://github.com/GRomR1/nxs-universal-chart/tree/dev/samples

КАК ЭТО ИСПОЛЬЗОВАТЬ (ВАРИАНТ 2)

1. **Подключение как зависимость в свой чарт**
   Chart.yaml

```yaml
version: 1.0.0
name: some-stateful-app
type: application

dependencies:
- name: universal-chart
  version: ^2.0.0
  repository: "@gromr1"
  alias: settings
```

КАК ЭТО ИСПОЛЬЗОВАТЬ (ВАРИАНТ 2)

1. **Подключение как зависимость в свой чарт**
   Chart.yaml

2. **Файл с описанием создаваемых ресурсов: values.yaml**
   В секции settings (имя alias)

```yaml
settings:
  statefulsets:
    cluster:
      replicas: 3
      serviceName: cluster-headless
      containers:
        app:
          image: >-
            '{{ $.Values.global.repository }}
                /infra/some-stateful-app'
          imageTag: "1.1.0"
```

КАК ЭТО ИСПОЛЬЗОВАТЬ (ВАРИАНТ 2)

1. **Подключение как зависимость в свой чарт**
   Chart.yaml

2. **Файл с описанием создаваемых ресурсов: values.yaml**
   В секции settings (имя alias)

3. **Запустить установку**
   - `helm dependency build ./helm/some-stateful-app`
   - `helm install ./helm/some-stateful-app`

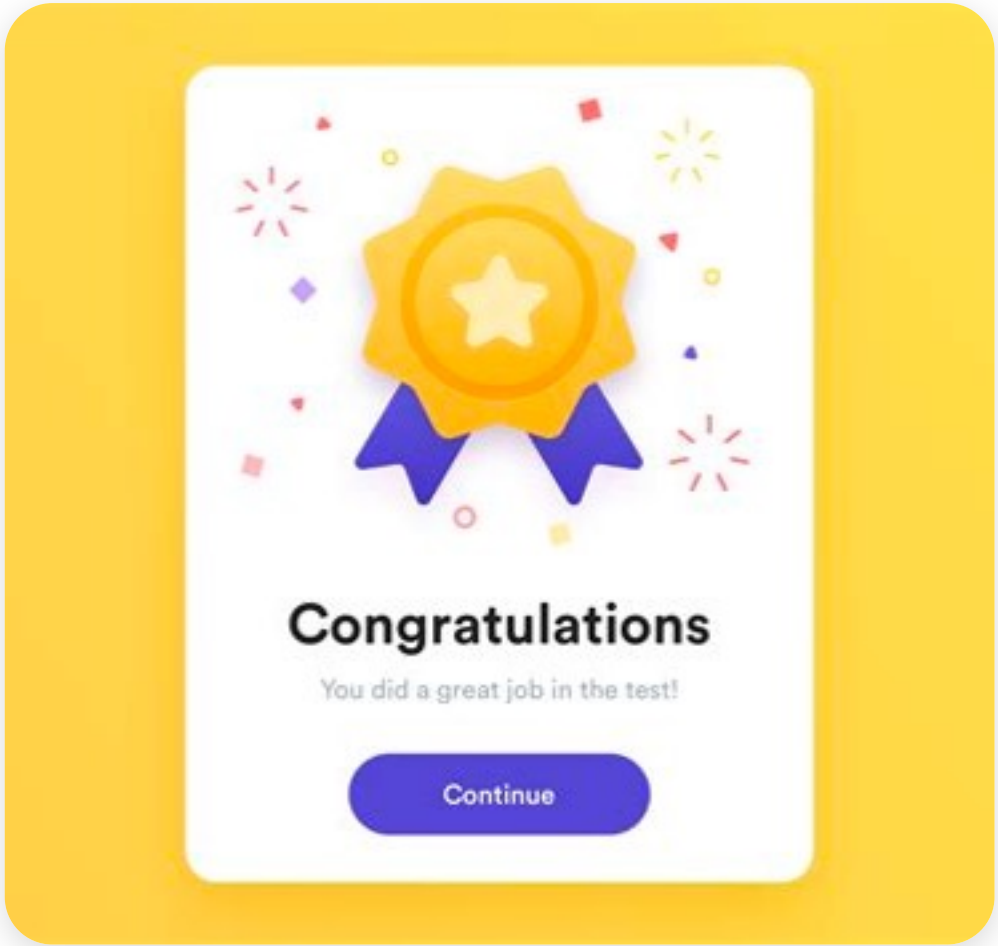
КАК ЭТО ИСПОЛЬЗОВАТЬ (ВАРИАНТ 2)

1. **Подключение как зависимость в свой чарт**
   Chart.yaml

2. **Файл с описанием создаваемых ресурсов: values.yaml**
   В секции settings (имя alias)

3. **Запустить установку**
   - `helm dependency build ./helm/some-stateful-app`
   - `helm install ./helm/some-stateful-app`

4. **Проверить**



КАКОЙ ВАРИАНТ ВЫБРАТЬ?

Вариант 1 (только values.yaml)

Достаточно стандартных ресурсов (deploy, svc, ingress)

Только values.yaml

Деплой приложений

Вариант 2 (через dependency)

Требуются дополнительные CR, функции и шаблоны

Полноценный helm-чарт

Упаковка и деплой приложений

РЕЗУЛЬТАТЫ

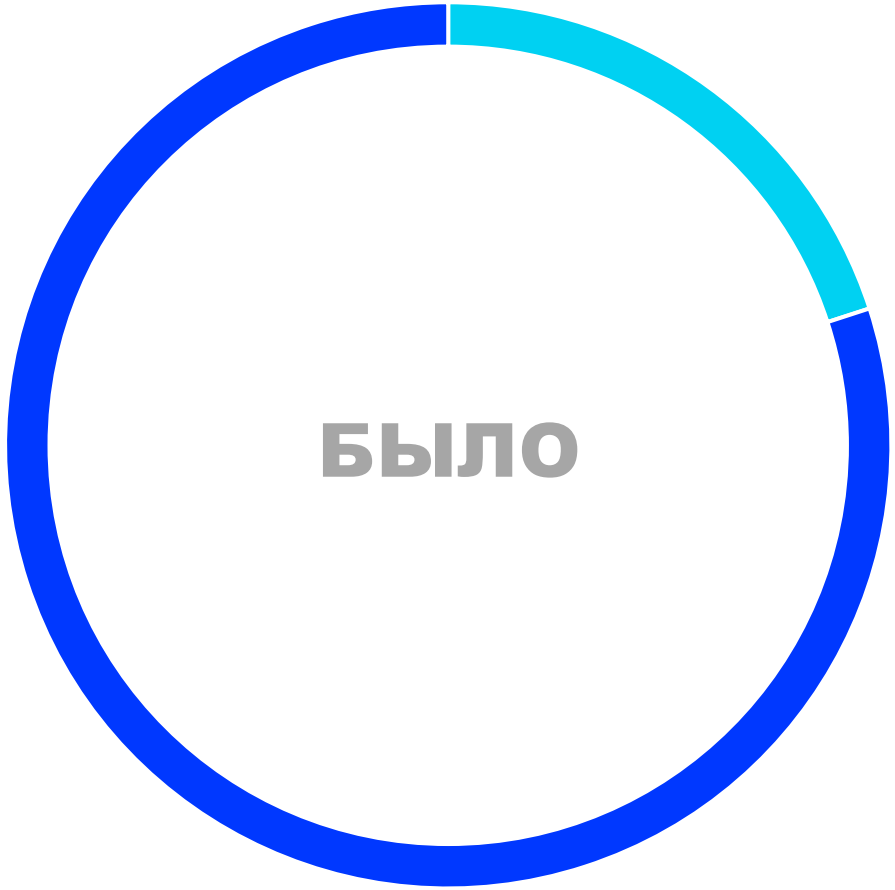
РЕЗУЛЬТАТЫ

ГДЕ ИСПОЛЬЗУЕТСЯ У НАС

- Classic Web App
- OpenSearch
- OpenSearch Dashboards
- FluentD

КАКИХ РЕЗУЛЬТАТОВ МЫ ДОСТИГЛИ

Процент изменяемых параметров
между приложениями

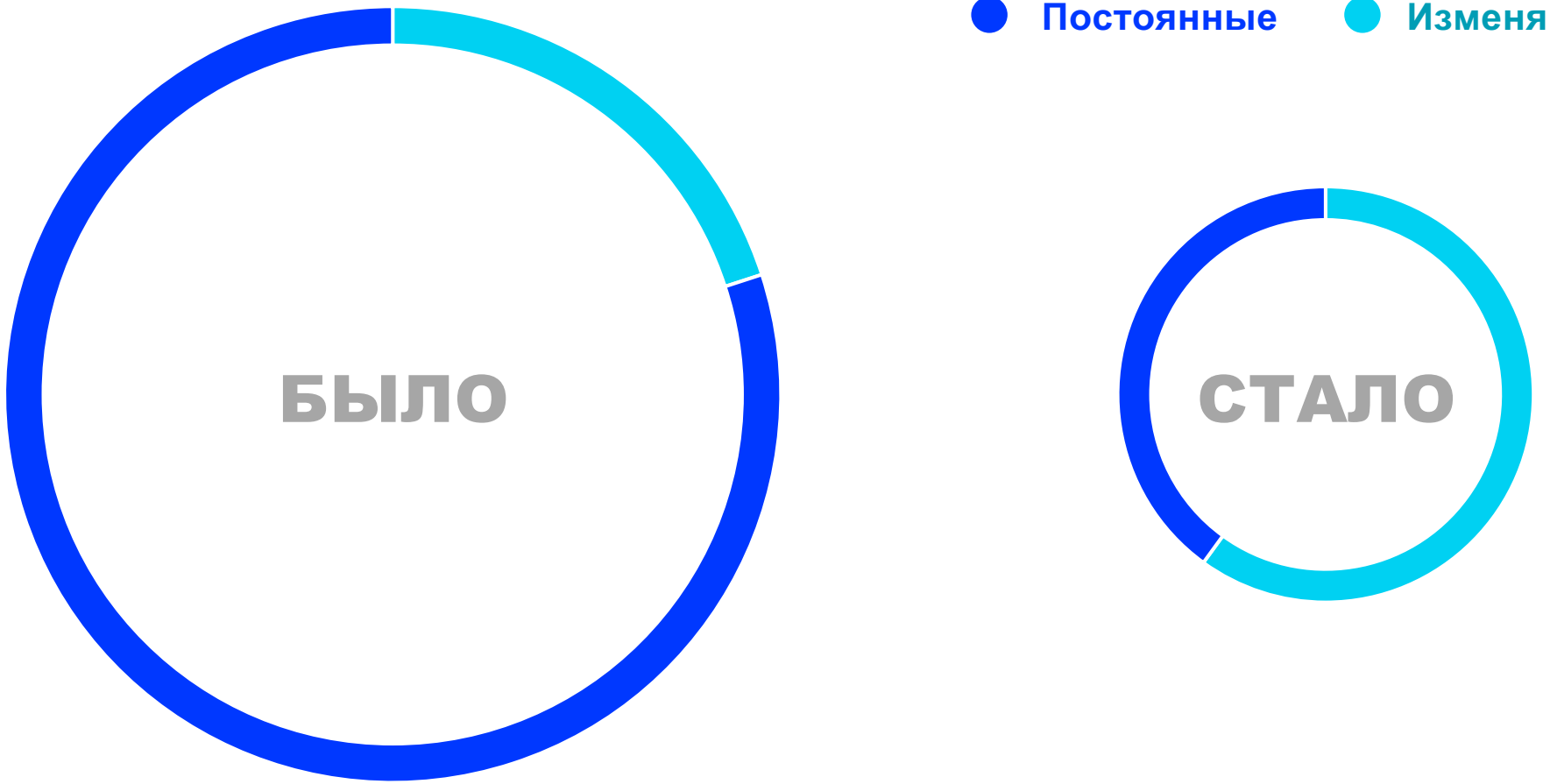


Постоянные

Изменяются

КАКИХ РЕЗУЛЬТАТОВ МЫ ДОСТИГЛИ

Процент изменяемых параметров
между приложениями

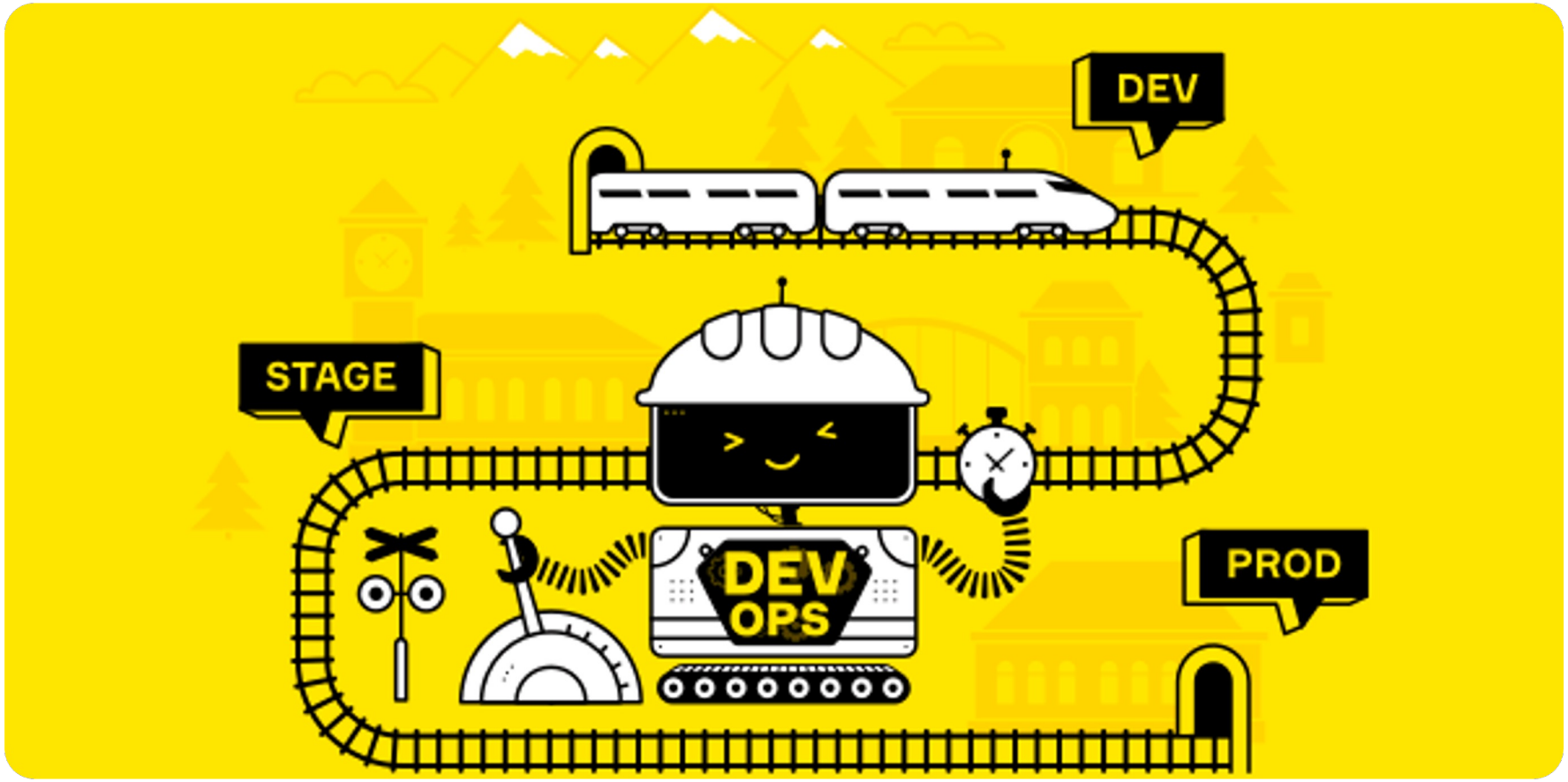


OPEN-SOURCE

ИННОТЕХ

АВТОРЫ И ИСХОДНИКИ

Проект nixys/nxs-universal-chart
(Apache 2.0)

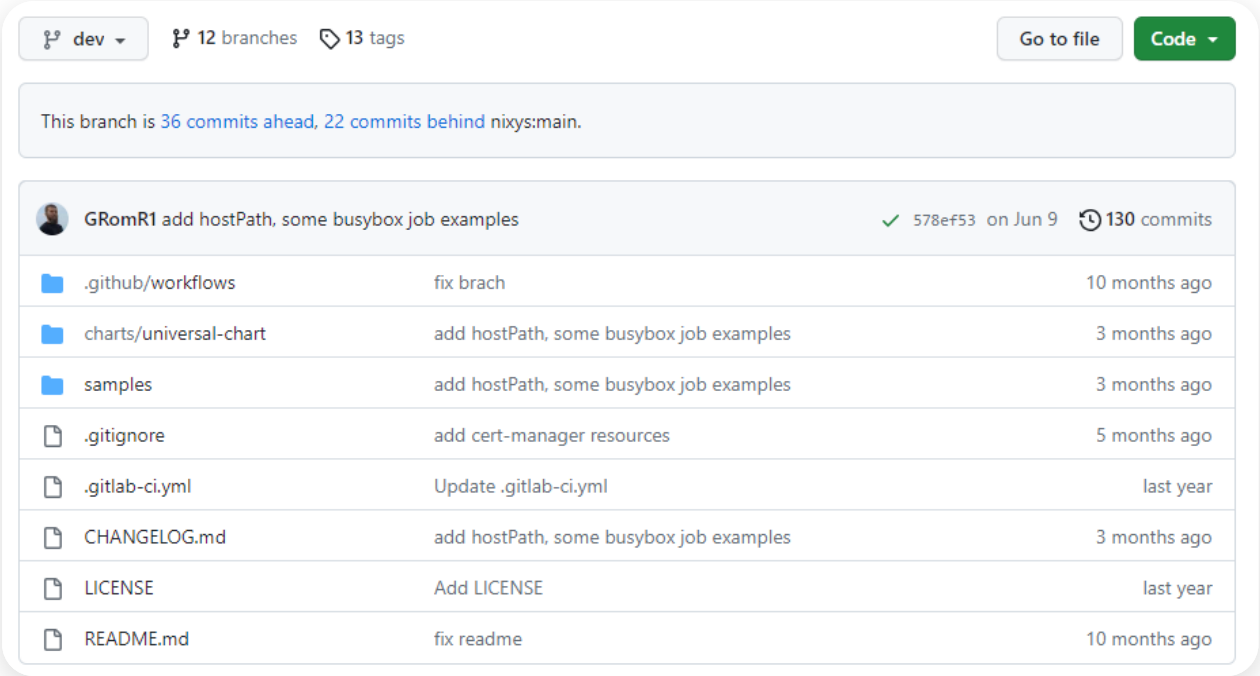


ФОРК ПРОЕКТА

Проект: [GRomR1/nxs-universal-chart](#)
(Apache 2.0)

Добавлено:

- Множество простых примеров
- Поддержка ресурсов [cert-manager](#)
- Опции для отключения создания ресурсов
- Поддержка различных способов определения ресурсов (dict & list)
- Поддержка переопределения namespace
- Релизы через GitHub Actions
  и публикация их на GH Pages
- Unit-тесты через [kind](#), [conftest](#) и [ct](#)



ПЛАНЫ

Проект: GRomR1/nxs-universal-chart
(Apache 2.0)

Планы:

- Генерация документации
- Валидация схем
- Больше тестов
- Новые типы ресурсов

ВМЕСТО ЗАКЛЮЧЕНИЯ

ВЫВОДЫ

ЕСЛИ НАХОДИТЕСЬ НА СТАРТЕ

- Оценить возможный рост
- Подумать о масштабировании заранее
- Использовать шаблоны и include там, где возможно

ЕСЛИ УЖЕ ИСПОЛЬЗУЕТЕ HELM

- Посчитать ресурсы на поддержку решений
- Возможно универсальный чарт сможет их уменьшить
- Один helm-чарт
- Унификация настроек
- Уменьшение количества систем (~~chart registry~~)
- Ниже порог входа

СПАСИБО ЗА ВНИМАНИЕ!

ИННОТЕХ

РУСЛАН ГАЙНАНОВ

Ведущий инженер DevOps, ГК «Иннотех»

t.me/gainanovrus

github.com/GRomR1