Васильев Егор
Positive Technologies

Тестировать Helm-чарты проще, чем их писать

О себе

- DevOps
- Работал в крупных международных компаниях (Veeam Software, JetBrains, Huawei)
- Веду технический блог http://blog.bissquit.com

Аудитория

- QA-инженеры любого уровня, не работавшие с Kubernetes и которым не приходилось сталкиваться с тестированием инфраструктуры, но которым **очень хочется** влиться в тему
- Инженеры разных специализаций (dev, ops, other), которые хотят повысить качество своих деплоев в Kubernetes

Как возникла идея доклада

QA тестируют приложения, но мало кто тестирует инфраструктуру, хотя

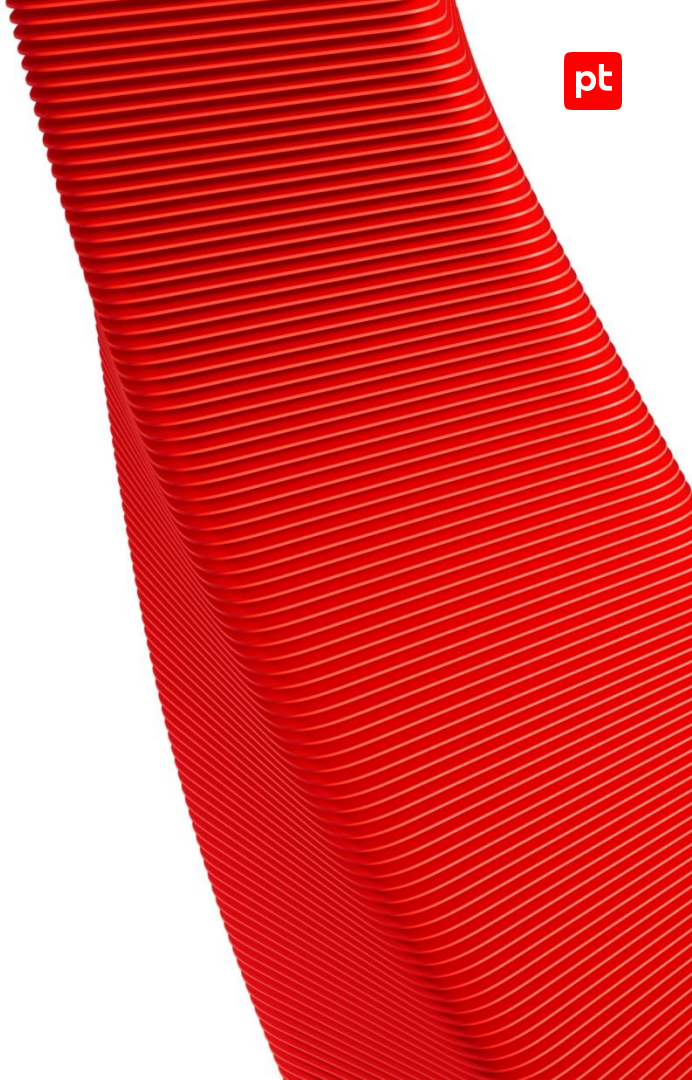
иногда качество инфраструктуры влияет на стабильность работы приложения даже больше, чем качество самого приложения

О чем поговорим?

- Что такое Helm
- Создадим чарт
- Узнаем как тестировать чарты
  - Подготовим окружение
  - Подеплоим чарт в кластер Kubernetes
- Создадим пайплайн

01

Что такое Helm?



Что такое Helm

Как это видит официальная документация:

The package manager for Kubernetes, CNCF graduated

Что такое Helm

Как это видим мы:

Шаблонизатор с GO templates под капотом,

который позволяет сделать множество громоздких yaml-манифестов Kubernetes более гибкими для деплоя в разные окружения

Что такое Helm

```yaml
apiVersion: v1
kind: Service
metadata:
  name: rate-limits-exporter
  labels:
    app.kubernetes.io/name: rate-limits-exporter
spec:
  type: ClusterIP
  ports:
    - port: 80
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app.kubernetes.io/name: rate-limits-exporter
```

Что такое Helm

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

Что такое Helm

_helpers.tpl

```
1 {{- define "rate-limits-exporter.fullname" -}}
  {{- if .Values.fullnameOverride }}
2 {{- .Values.fullnameOverride | trunc 63 | trimSuffix "-" }}
  {{- else }}
  {{- $name := default .Chart.Name .Values.nameOverride }}
3 {{- if contains $name .Release.Name }}
  {{- .Release.Name | trunc 63 | trimSuffix "-" }}
  {{- else }}
  {{- printf "%s-%s" .Release.Name $name | trunc 63 | trimSuffix "-" }}
  {{- end }}
  {{- end }}
  {{- end }}
```

Что такое Helm

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

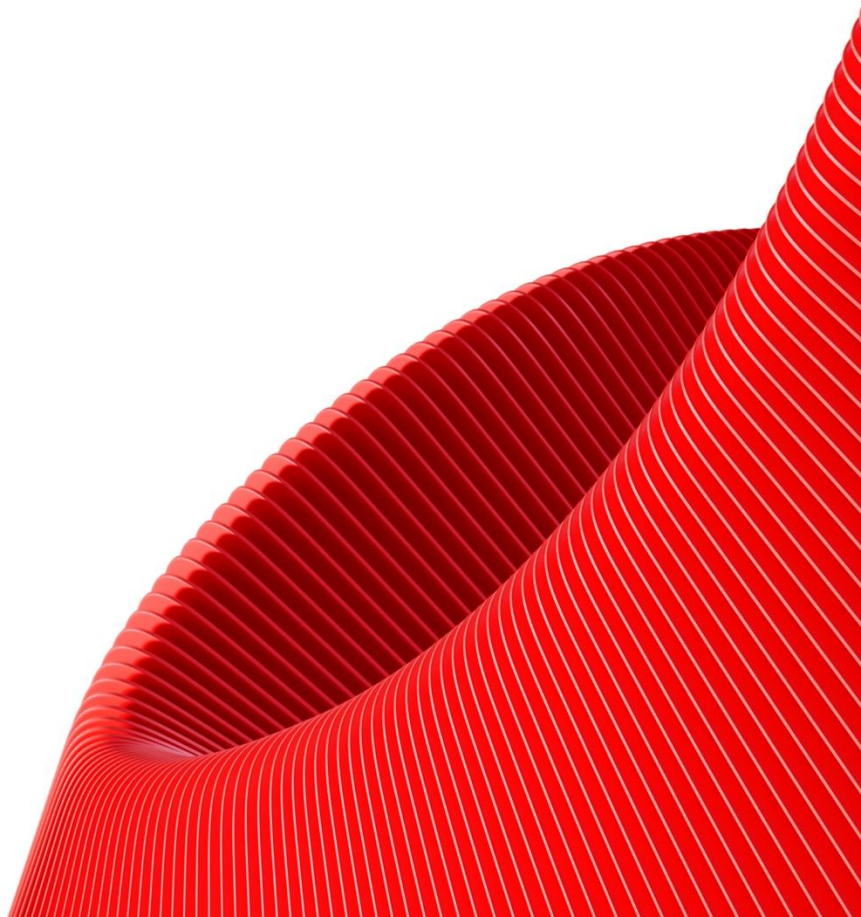
Что такое Helm

values.yaml

```yaml
service:
  type: ClusterIP
  port: 80
```

02

Создаем чарт



Создаем чарт

Пример: кастомный экспортер метрик для Prometheus - rate-limits-exporter (показывает лимиты по количеству скачиваний образов DockerHub)

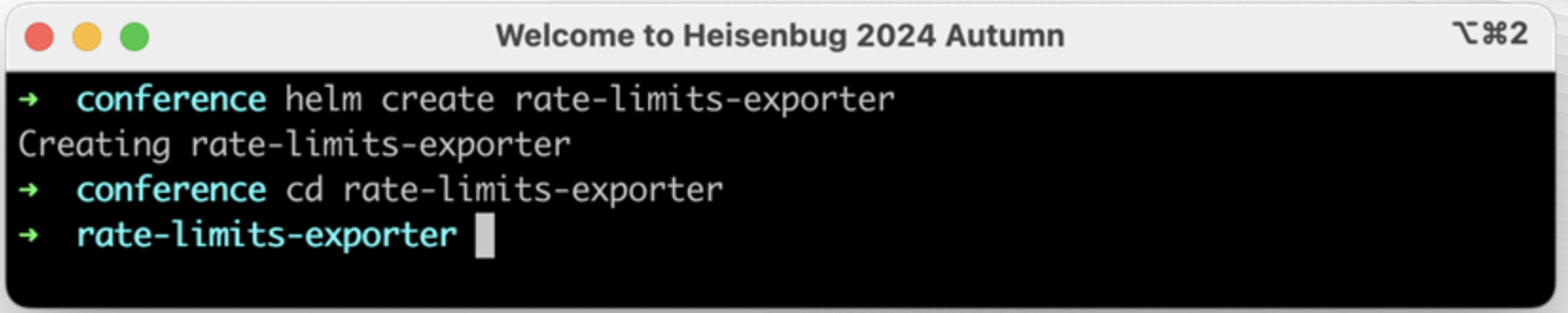
What's the download rate limit on Docker Hub?

Docker Hub limits the number of Docker image downloads, or pulls, based on the account type of the user pulling the image. Pull rate limits are based on individual IP address.

User type	Rate limit
Anonymous users	100 pulls per 6 hours per IP address
Authenticated users	200 pulls per 6 hour period
Users with a paid Docker subscription	Up to 5000 pulls per day

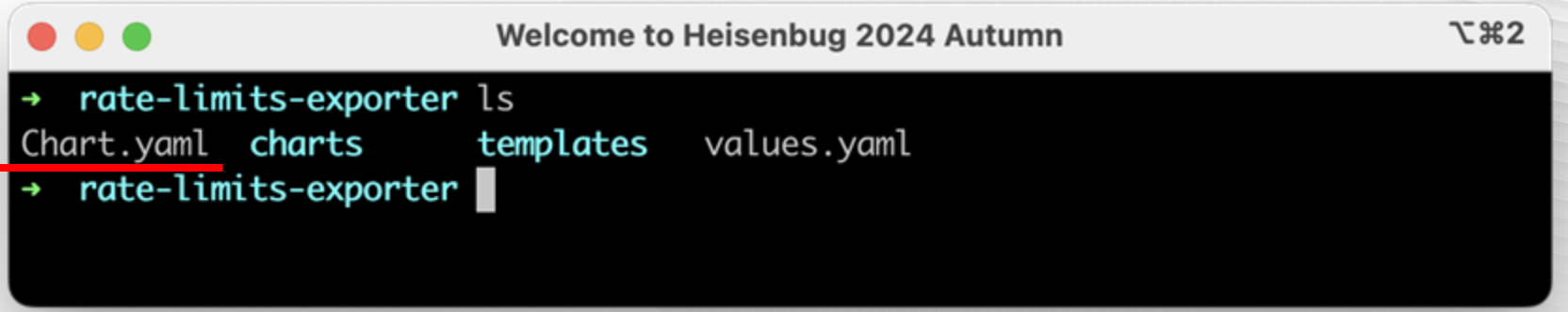
Создаем чарт

- Установим Helm
- Вызовем команду **helm create <chart-name>** для создания начальной структуры каталогов



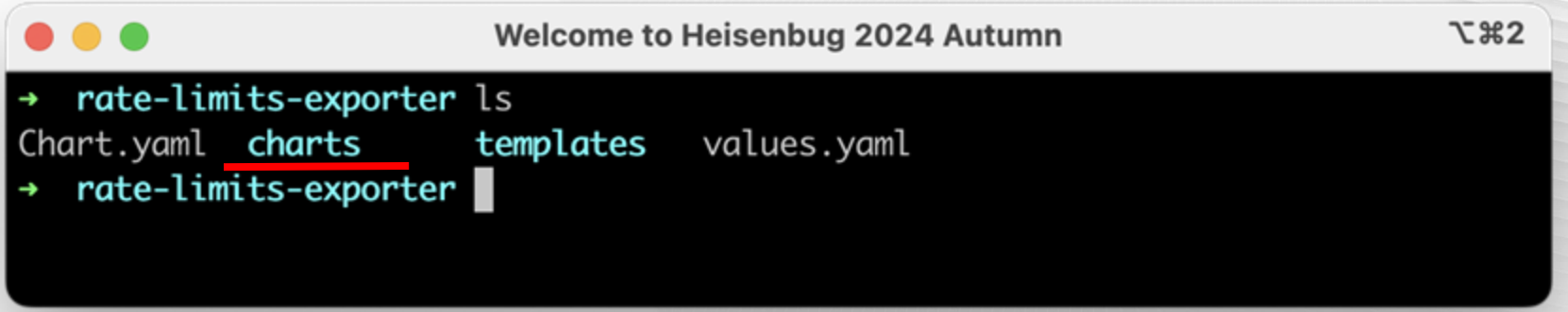
Создаем чарт

Общая информация о чарте (авторы, версии, описание)



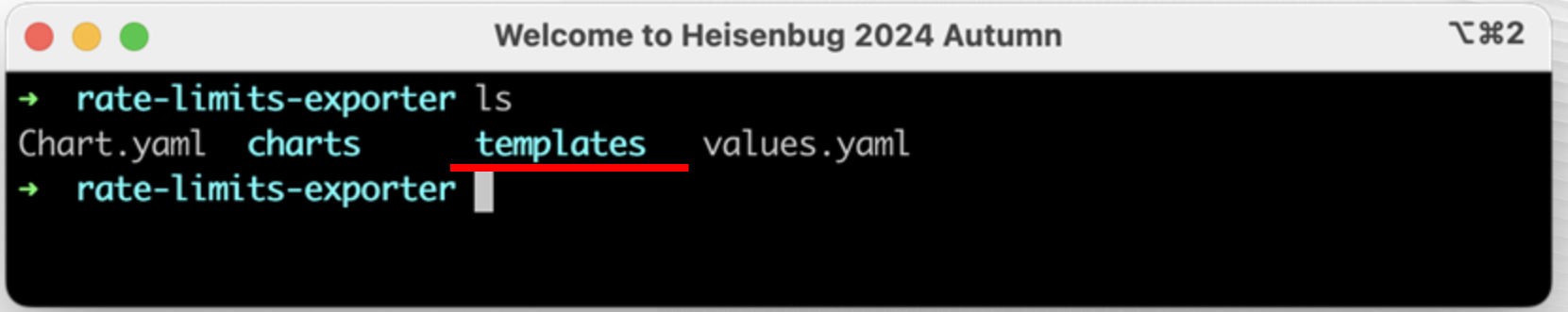
Создаем чарт

Любые другие чарты, от которых наш чарт зависит



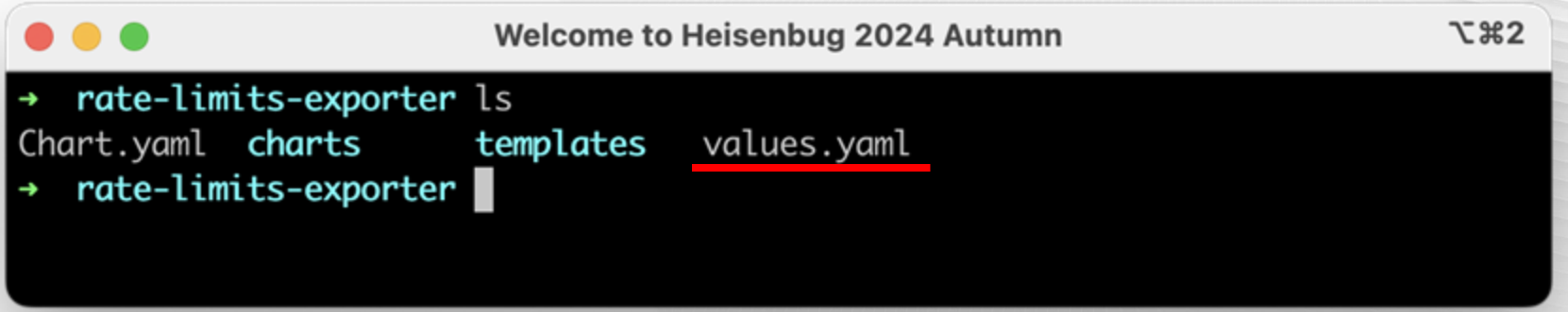
Создаем чарт

Шаблоны нашего чарта (+ values.yaml = Kubernetes manifests)



Создаем чарт

Конфигурация чарта по умолчанию



Создаем чарт

LICENSE - информация о лицензии

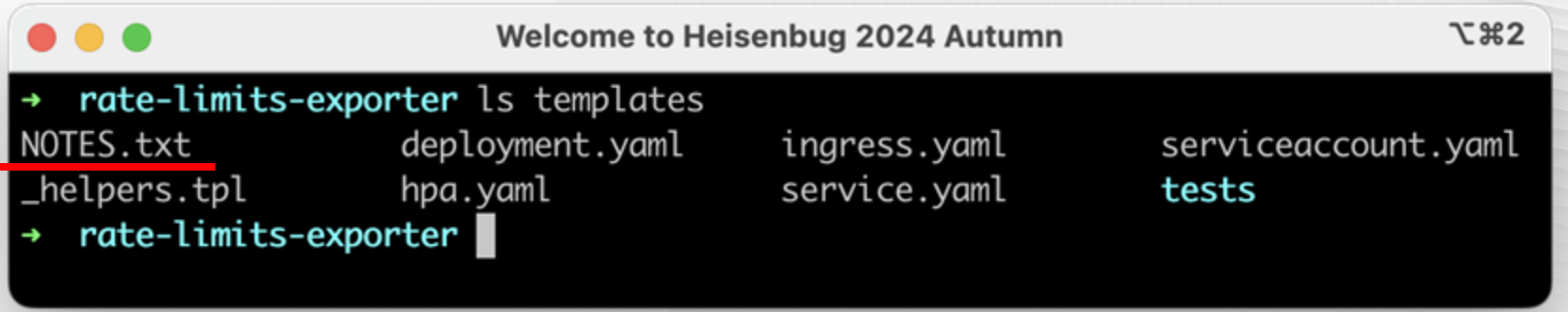
README.md - документация к чарту

values.schema.json - схема формата json, описывает структуру values.yaml

crds/ - каталог с Custom Resource Definitions

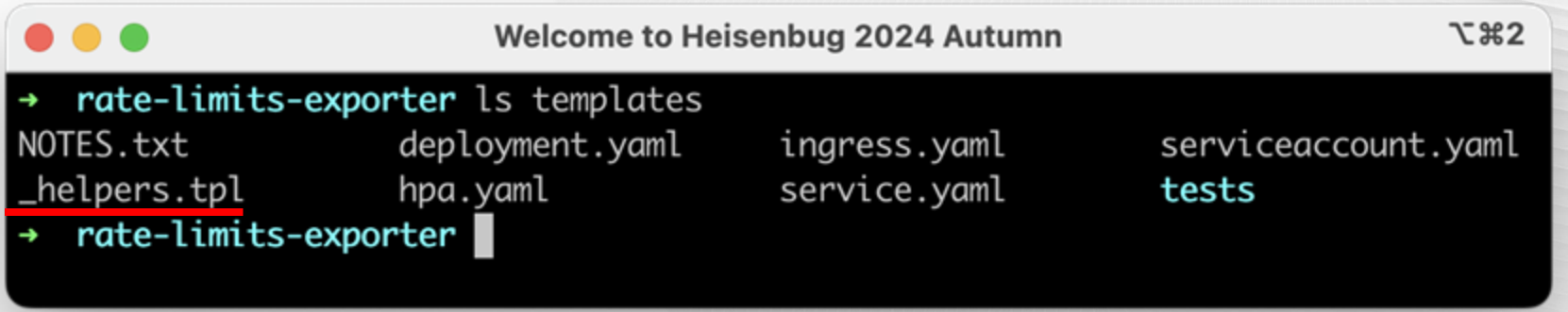
Создаем чарт

Заметки после успешной установки чарта



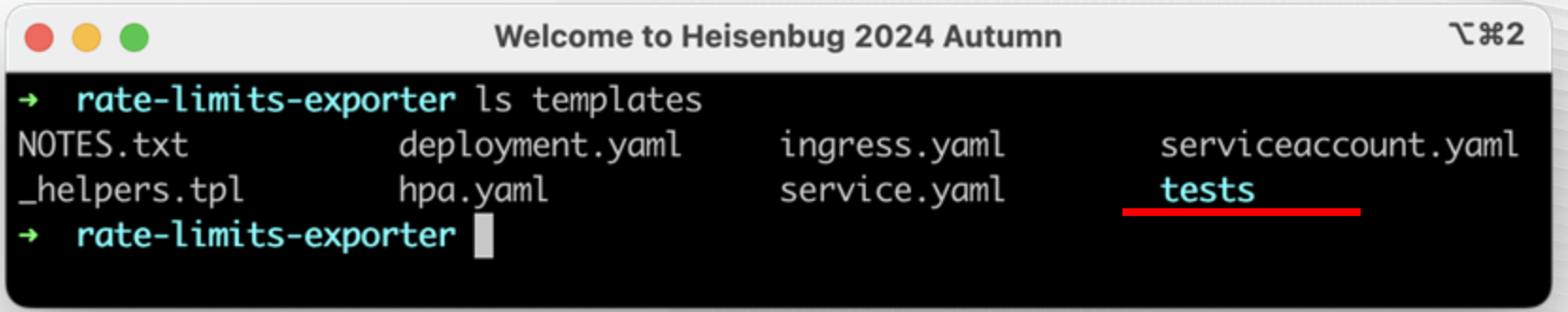
Создаем чарт

Файлы с _ в начале содержат переиспользуемый код



Создаем чарт

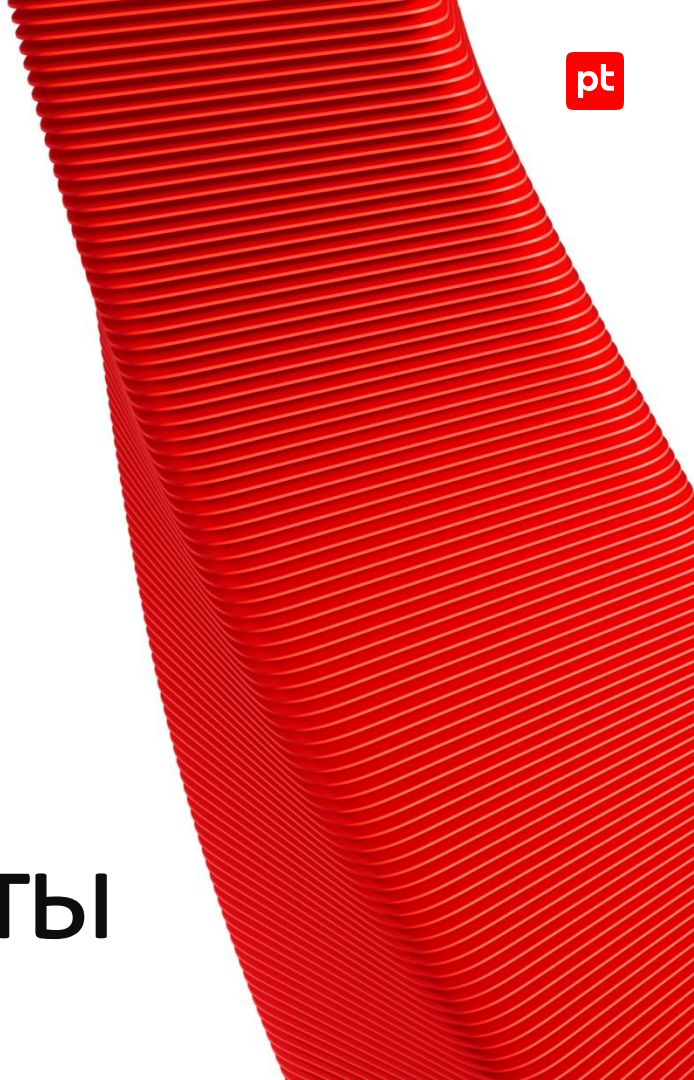
Магия Helm для тестирования чартов



Создаем чарт

Пример Helm-чарта на GitHub

03



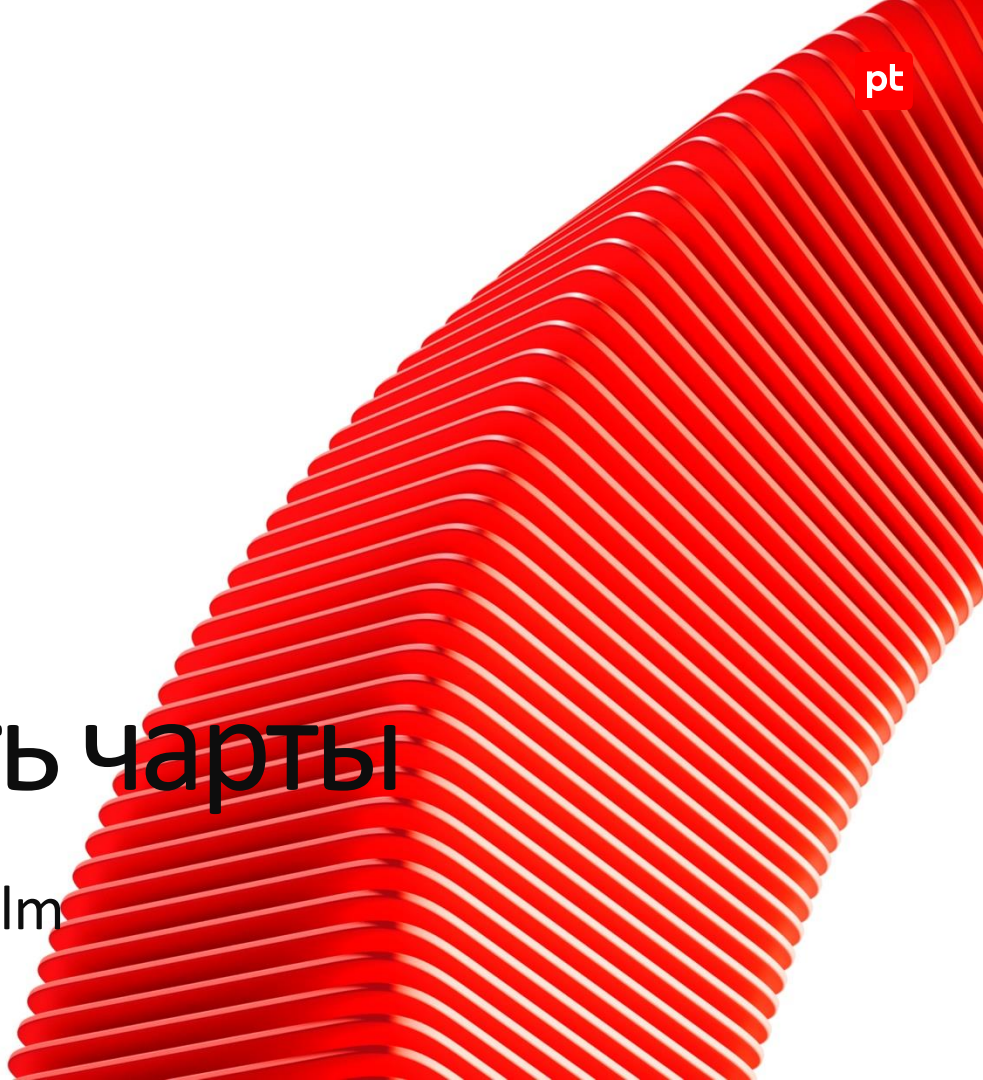
Как тестировать чарты

Тестирование чарта

Наверняка нам понадобится готовый кластер Kubernetes

Стоп! До этого момента мы можем использовать еще множество инструментов, не стоит забегать вперед

03



Как тестировать чарты

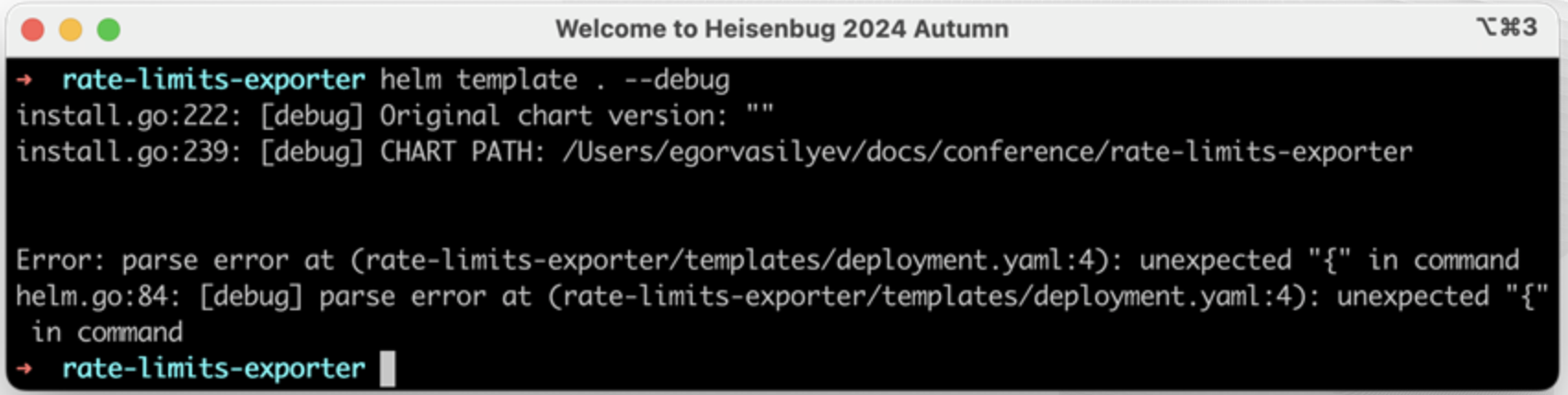
Встроенные инструменты Helm

Тестирование чарта

- Проверим, а способен ли Helm сделать рендер нашего чарта
- Если все хорошо, на выходе получите готовые к деплою манифесты

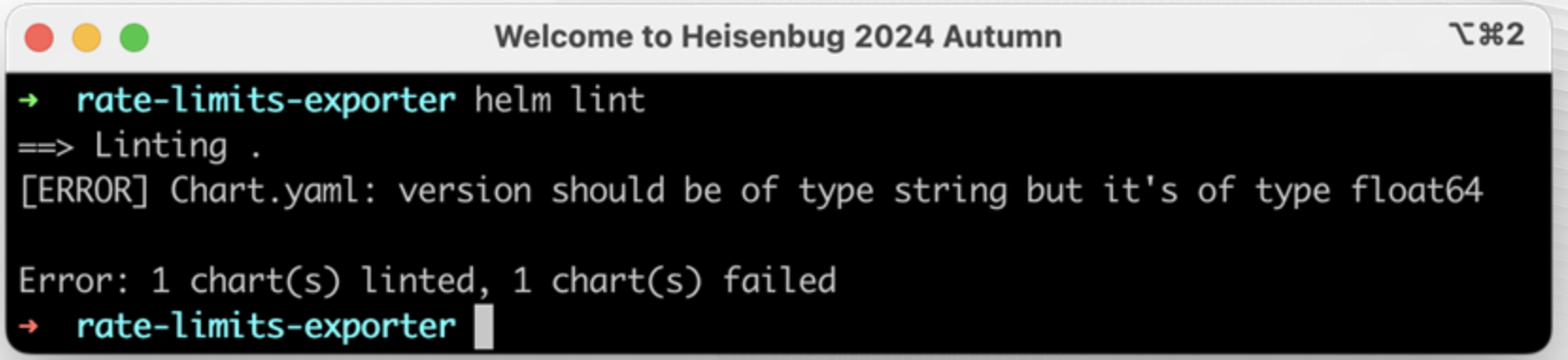
helm template <chart-name>

helm template .



Тестирование чарта

- Полезный встроенный инструмент - **helm lint**, который проверит чарт и выдаст критические ошибки, если чарт не сможет подеплоиться



А еще helm lint проверит соответствие схеме **values.schema.json**

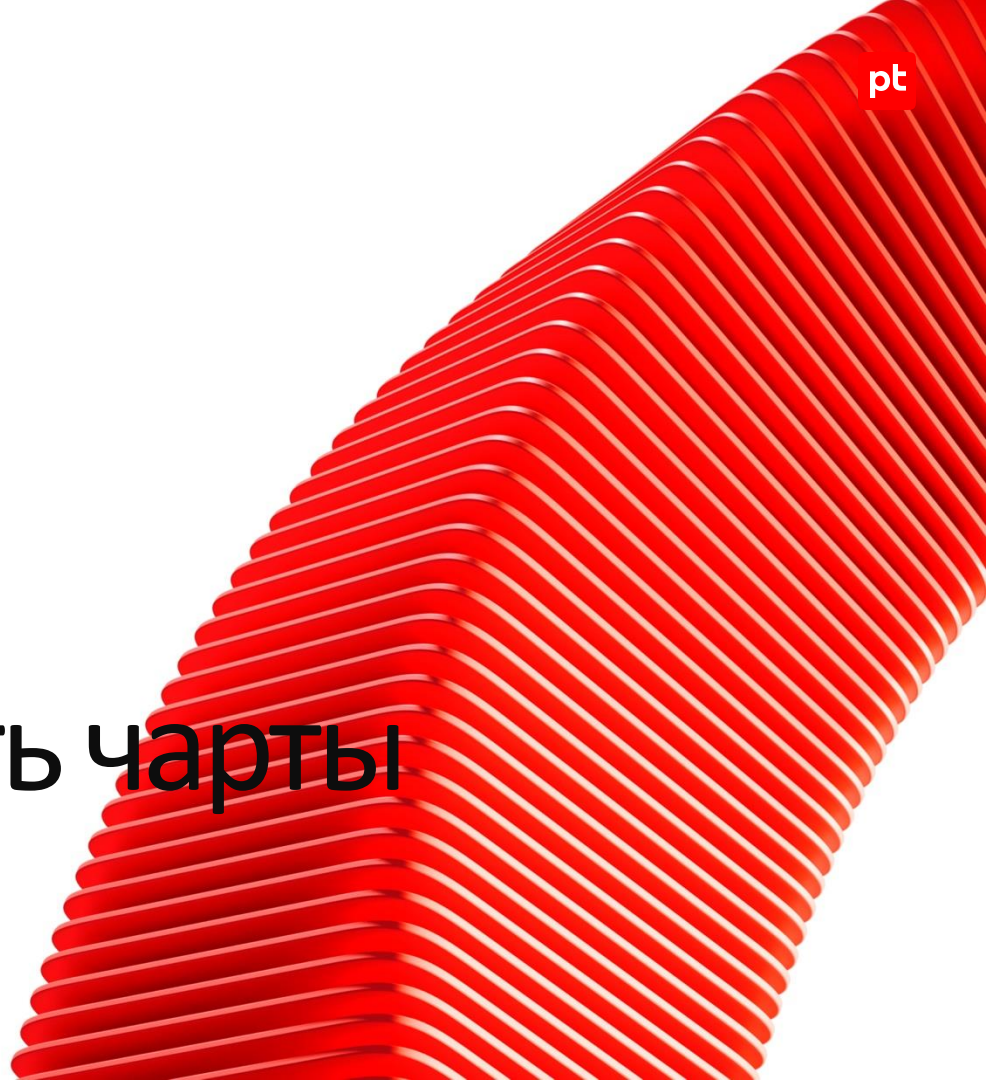
Тестирование чарта

- Если чарт не соответствует принятым соглашениям и рекомендациям, **helm lint** выдаст предупреждение

03

Как тестировать чарты

Валидация схемы



Тестирование чарта

ci passing | homebrew 0.6.7 | go report A+ | reference

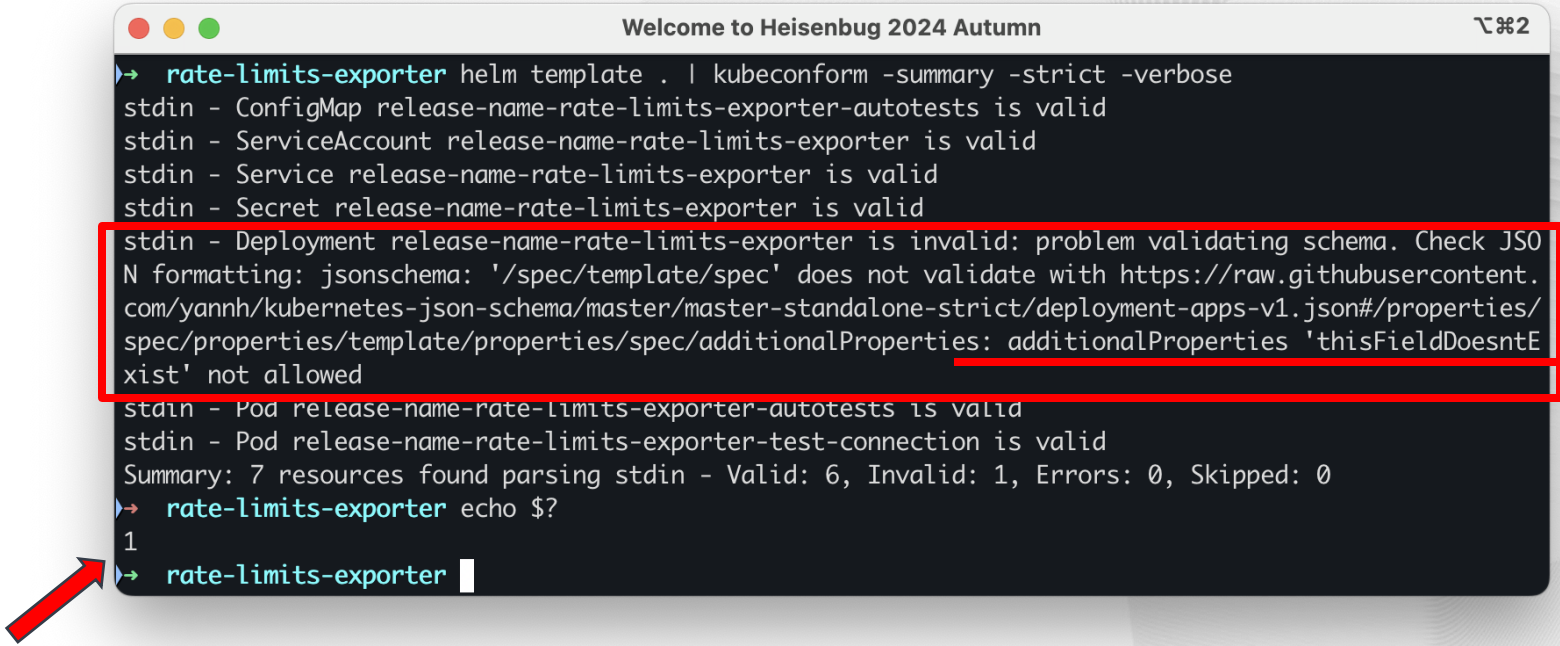
`Kubeconform` is a Kubernetes manifest validation tool. Incorporate it into your CI, or use it locally to validate your Kubernetes configuration!

Тестирование чарта

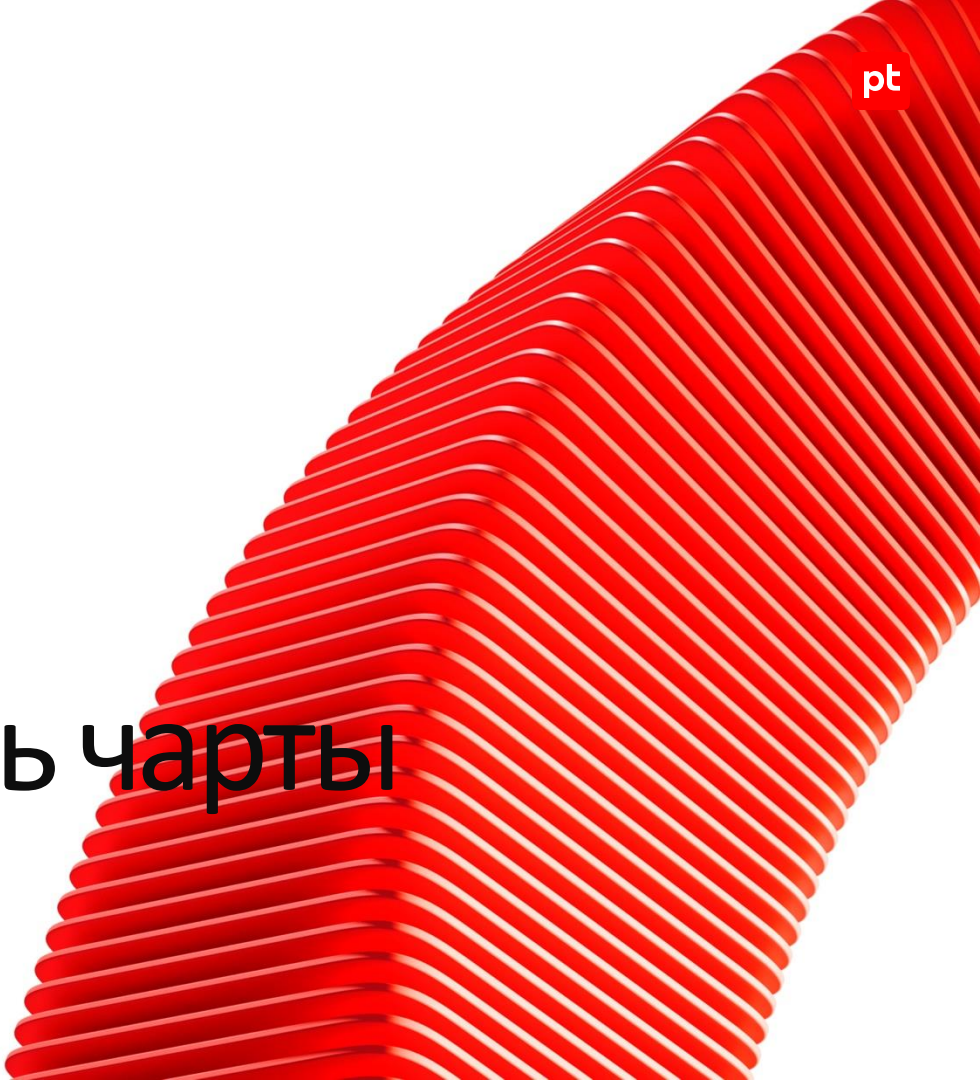
```
spec:
  thisFieldDoesntExist: "some-data"
  {{- with .Values.imagePullSecrets }}
  imagePullSecrets:
    {{- toYaml . | nindent 8 }}
  {{- end }}
  {{- if .Values.serviceAccount.create }}
  serviceAccountName: {{ include "rate-limits-exporter.serviceAccountName" . }}
  {{- end }}
  securityContext:
```

Тестирование чарта

helm template . | kubeconform -summary -strict -verbose



03



Как тестировать чарты

Статические анализаторы

Тестирование чарта

kube-score

go report A+ | Test Go passing | release v1.18.0 | Github stars 2.6k | downloads 2.8M | License MIT

`kube-score` is a tool that performs static code analysis of your Kubernetes object definitions.

The output is a list of recommendations of what you can improve to make your application more secure and resilient.

You can test kube-score out in the browser with the online demo (source).

Тестирование чарта

kube-score

Kubernetes object analysis with recommendations for improved reliability and security.

`kube-score` is a tool that does static code analysis of your Kubernetes object definitions. The output is a list of recommendations of what you can improve to make your application more secure and resilient.

`kube-score` is open-source and available under the MIT-license. For more information about how to use kube-score, see zegl/kube-score on GitHub. Use this website to easily test kube-score, just paste your object definition YAML or JSON in the box below.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: statefulset-test-1
spec:
  template:
    metadata:
      labels:
        app: foo
    spec:
      containers:
      - name: foobar
        image: foo:bar
---
apiVersion: policy/v1beta1
kind: PodDisruptionBudget
```

```
apps/v1/Deployment statefulset-test-1
    [CRITICAL] Container Resources
        · foobar -> CPU limit is not set
```

Тестирование чарта

Идентификатор проверки

Цель

Описание

ID	Target	Description	Enabled
deployment-strategy	Deployment	Makes sure that all Deployments targeted by service use RollingUpdate strategy	default
deployment-replicas	Deployment	Makes sure that Deployment has multiple replicas	default
ingress-targets-service	Ingress	Makes sure that the Ingress targets a Service	default
cronjob-has-deadline	CronJob	Makes sure that all CronJobs has a configured deadline	default
cronjob-restartpolicy	CronJob	Makes sure CronJobs have a valid RestartPolicy	default
container-resources	Pod	Makes sure that all pods have resource limits and requests set. The --ignore-container-cpu-limit flag can be used to disable the requirement of having a CPU limit	default
container-resource-requests-equal-limits	Pod	Makes sure that all pods have the same requests as limits on resources set.	optional
container-cpu-requests-equal-limits	Pod	Makes sure that all pods have the same CPU requests as limits set.	optional
container-memory-requests-equal-limits	Pod	Makes sure that all pods have the same memory requests as limits set.	optional
container-image-tag	Pod	Makes sure that a explicit non-latest tag is used	default

Тестирование чарта

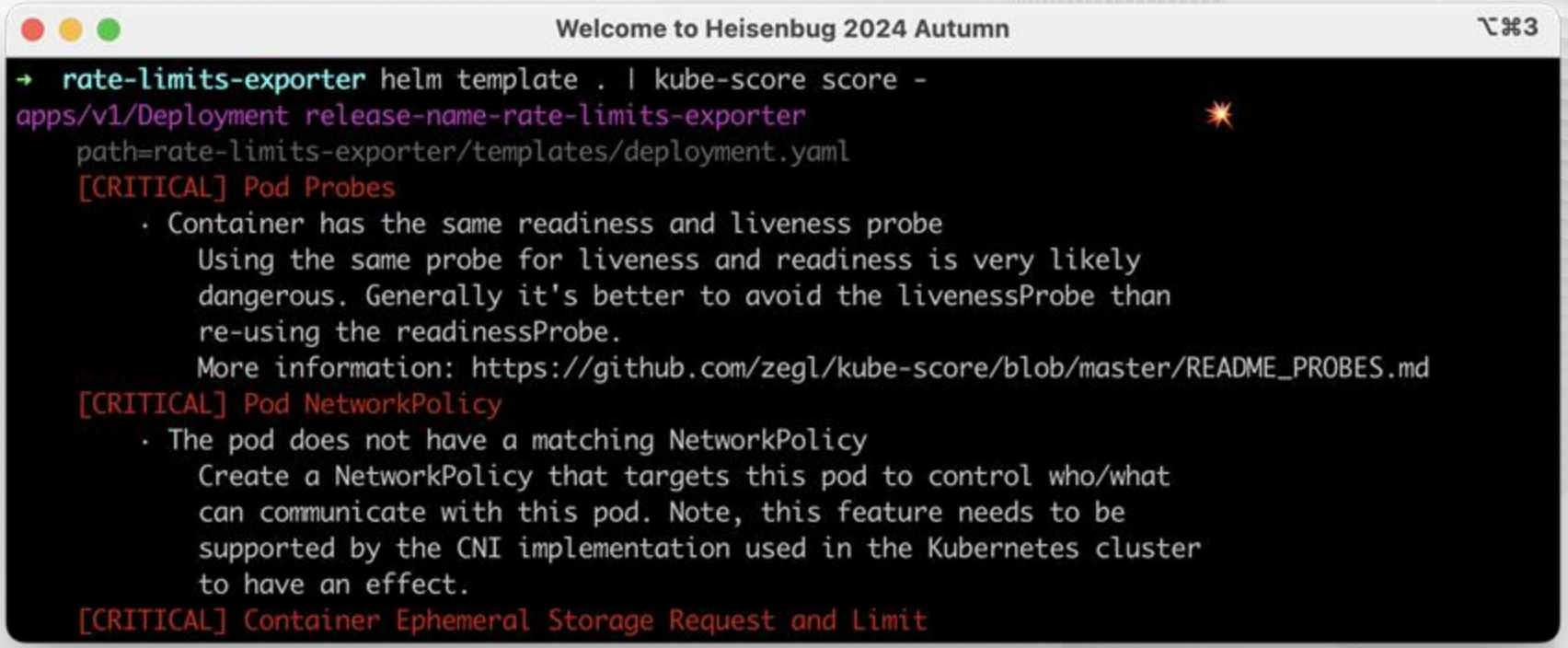
kube-score score --ignore-test pod-networkpolicy

OR

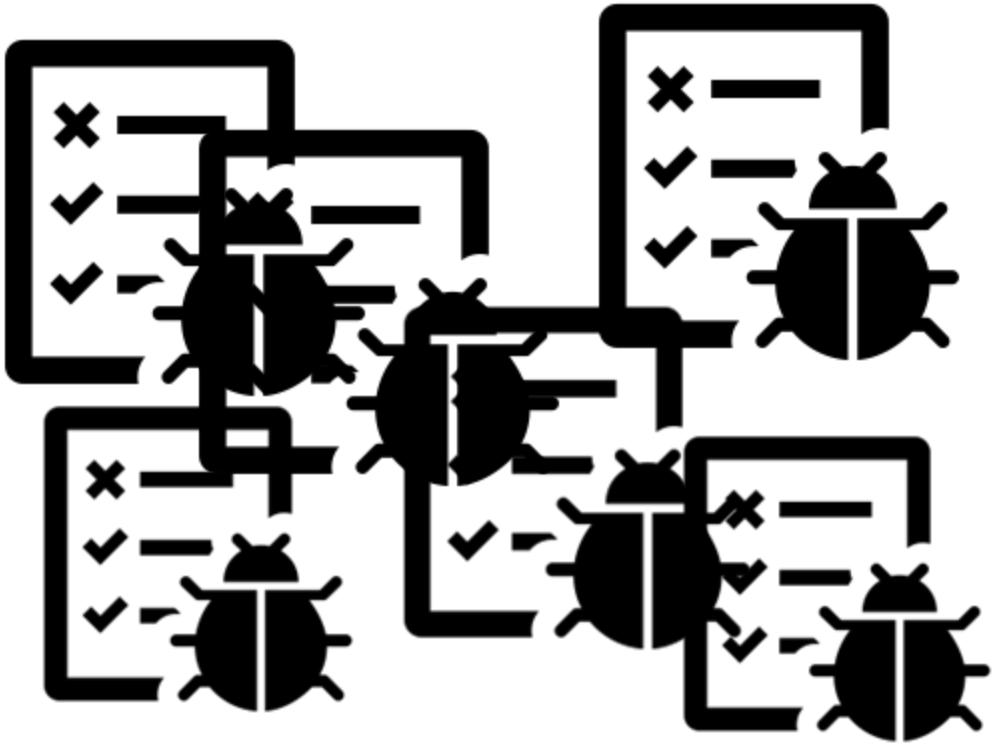
```
apiVersion: v1
kind: Pod
metadata:
  name: "{{ include "rate-limits-exporter.fullname" . }}-test-connection"
  labels:
    {{- include "rate-limits-exporter.labels" . | nindent 4 }}
  annotations:
    "helm.sh/hook": test
    "kube-score/ignore": container-ephemeral-storage-request-and-limit,pod-probes
```

Тестирование чарта

helm template . | kube-score score -



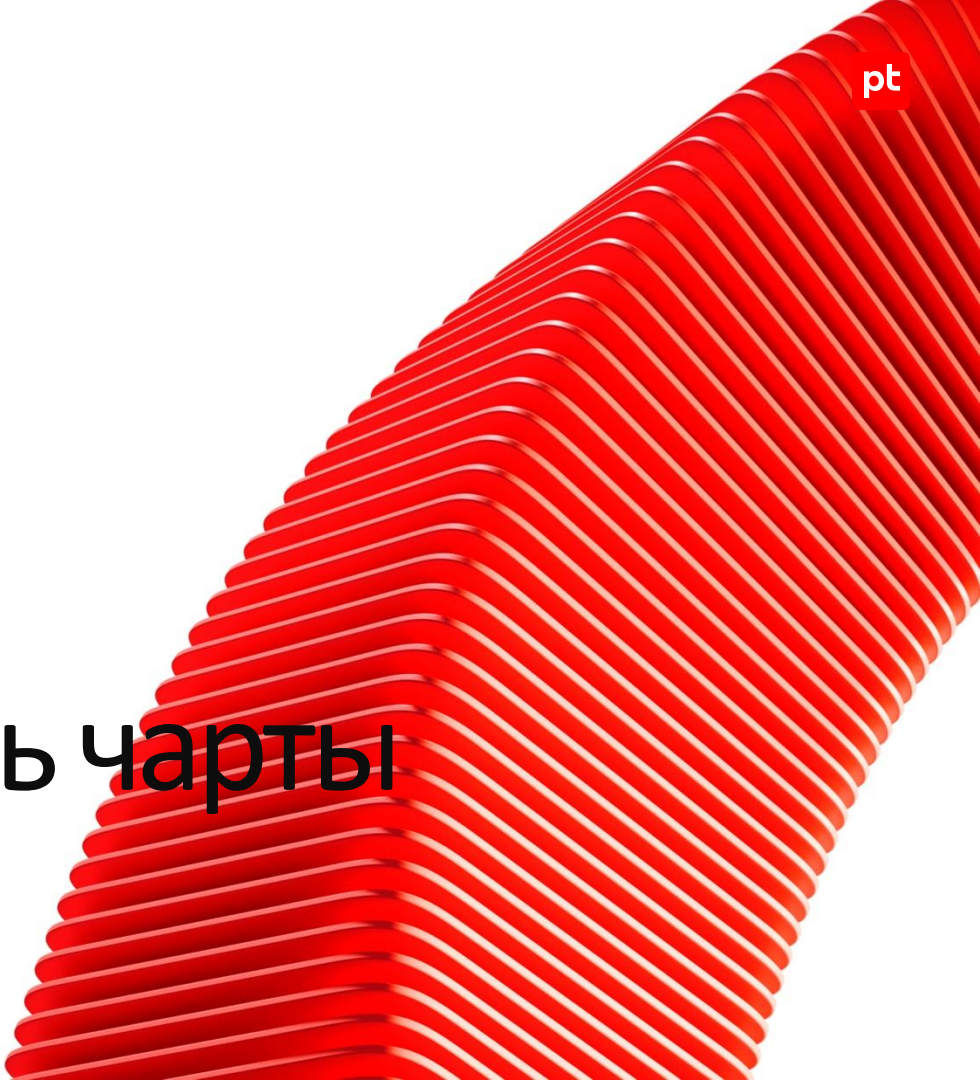
Тестирование чарта



03

Как тестировать чарты

Промежуточные итоги

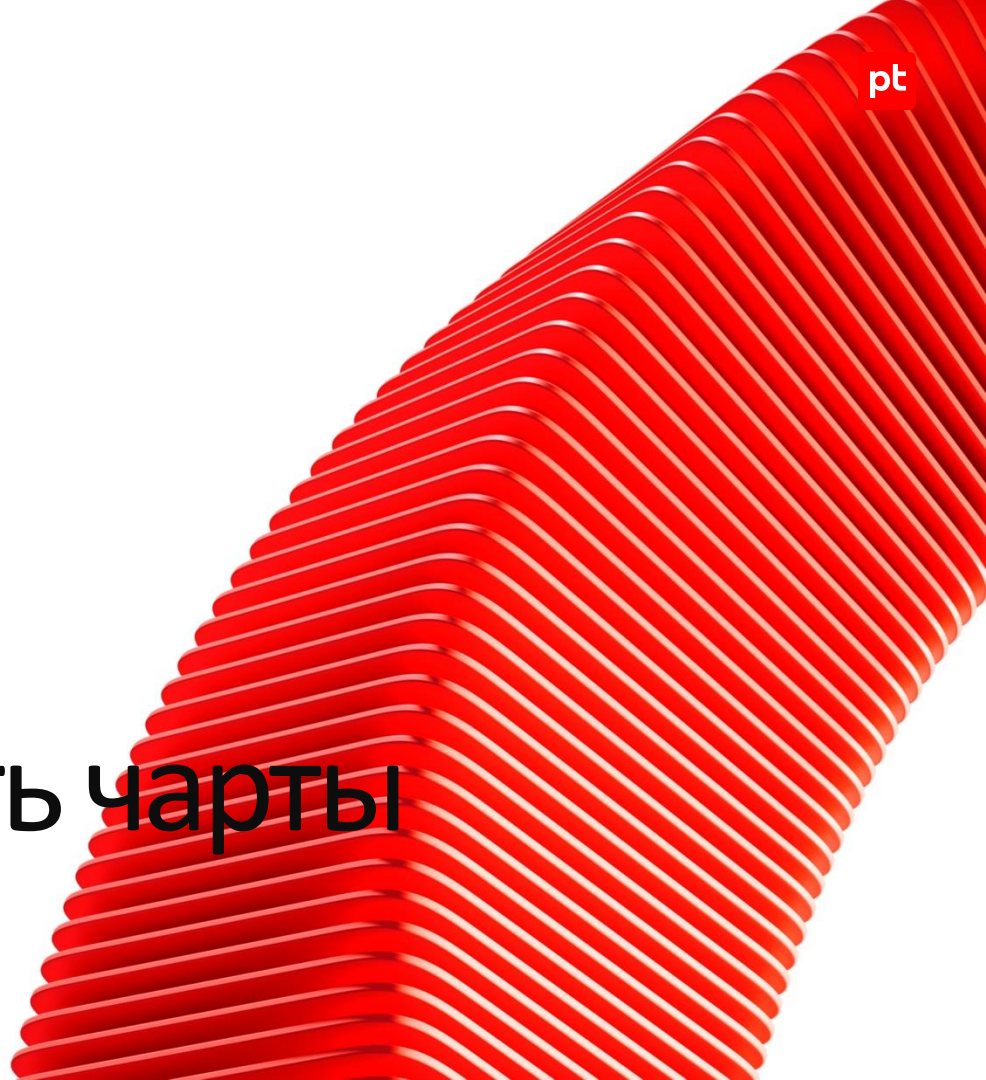


Тестирование чарта

helm template
helm lint
kubeconform
kube-score

Инструменты для тестирования чарта локально без необходимости деплоить в кластер Kubernetes

03

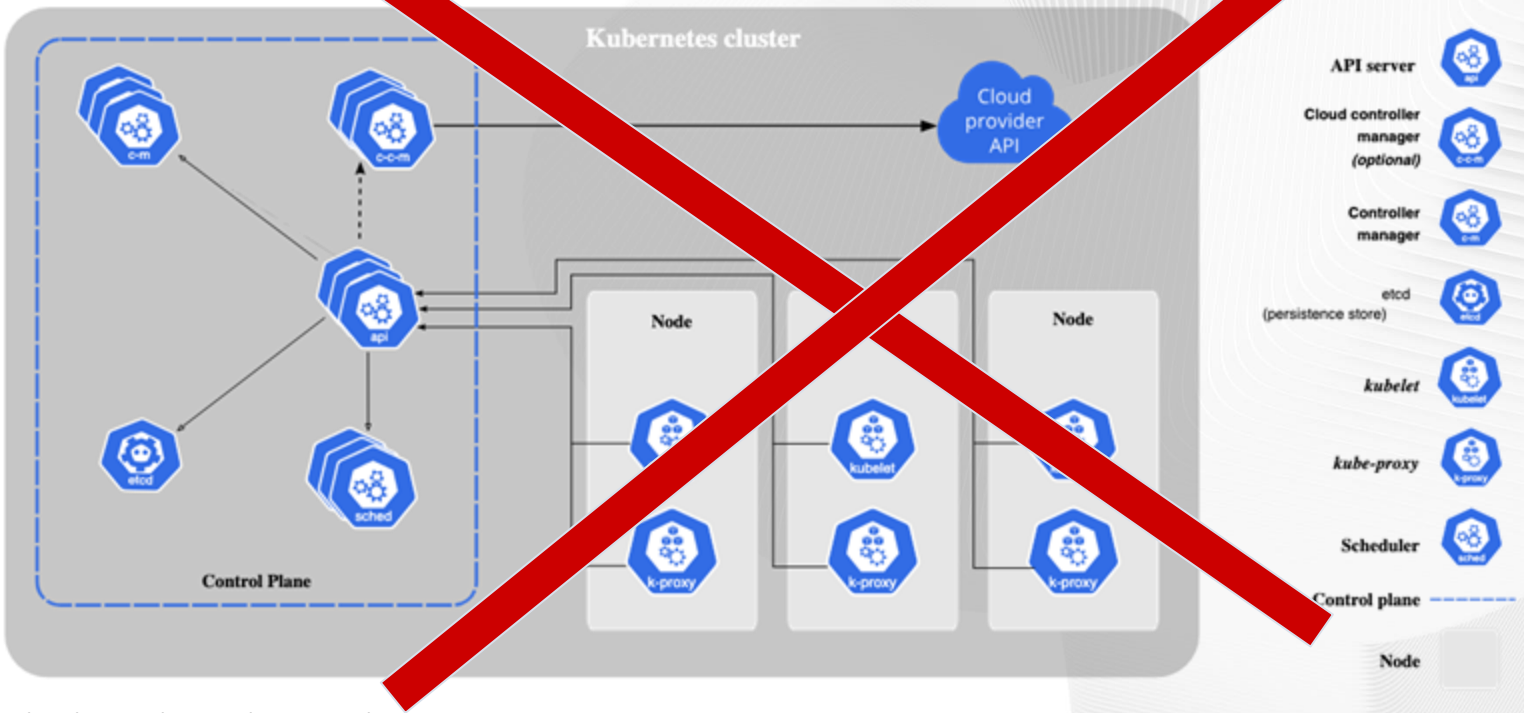


Как тестировать чарты

Готовим окружение

Готовим окружение

Архитектура Kubernetes



https://kubernetes.io/docs/concepts/overview/components/

Готовим окружение

Архитектура Kubernetes

Готовим окружение

Нам понадобится:

- kubectl
- kind

Готовим окружение

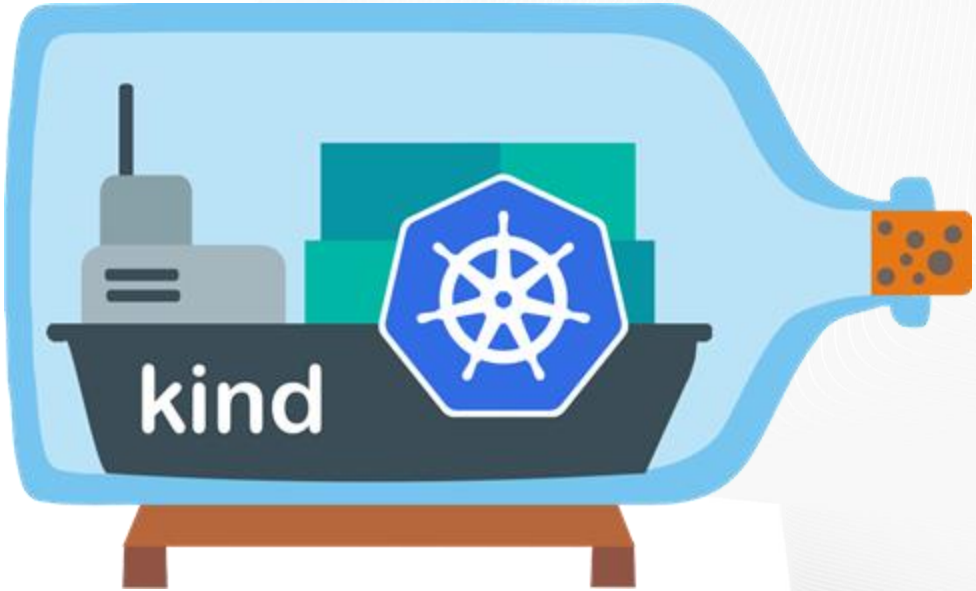
kubectl - Kubernetes command-line tool

supported within one minor version (older or newer) of kube-apiserver (kube-apiserver is at 1.30 - kubectl is supported at 1.31, 1.30, and 1.29)

https://kubernetes.io/releases/version-skew-policy/

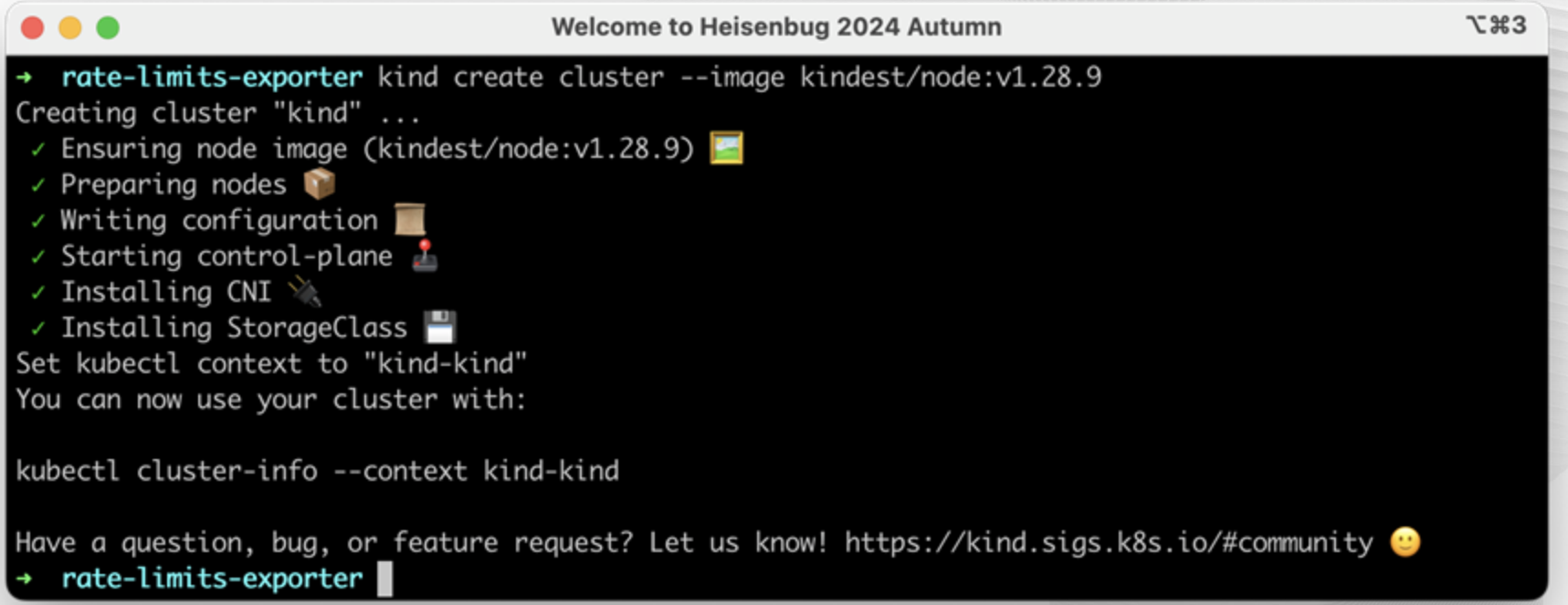
Готовим окружение

kind - Kubernetes in Docker - инструмент для запуска кластеров Kubernetes локально с использованием контейнеров Docker



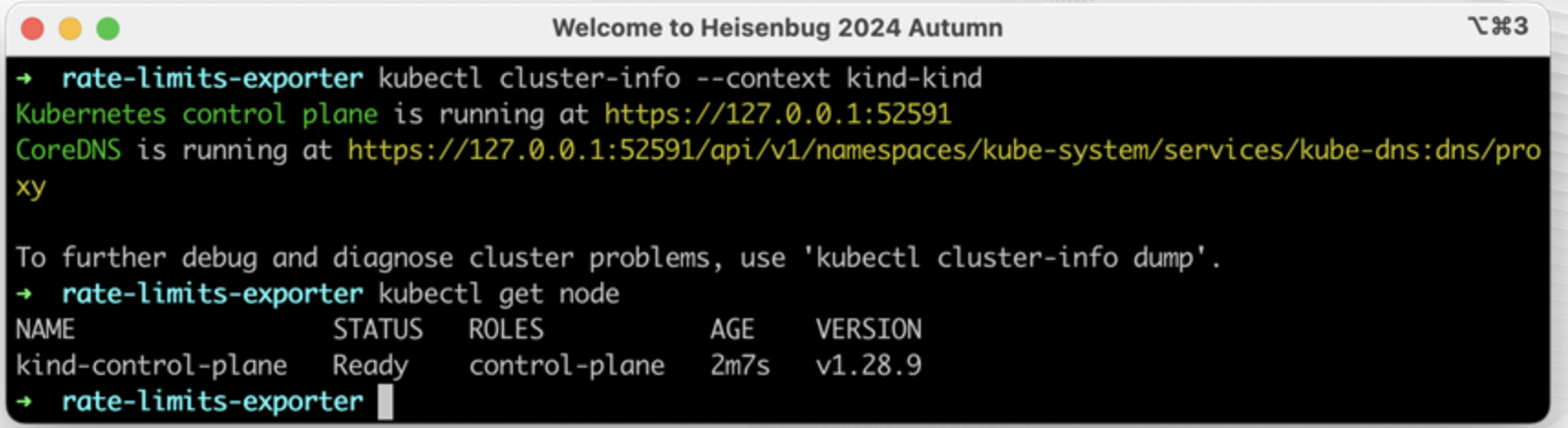
Готовим окружение

kind create cluster --image kindest/node:v1.28.9



Готовим окружение

kubectl cluster-info --context kind-kind
kubectl get node

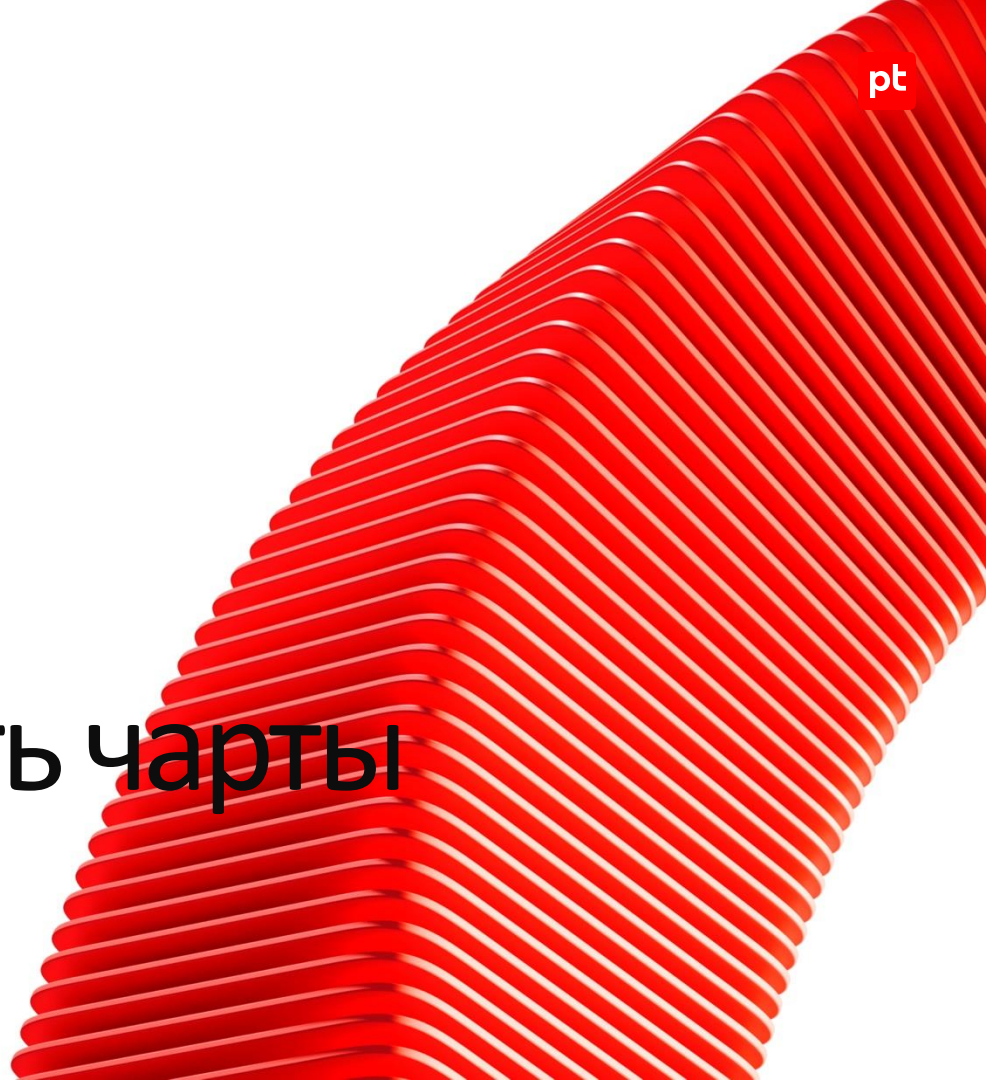


Окружение готово, можно продолжать!

03

Как тестировать чарты

Деплоим чарт

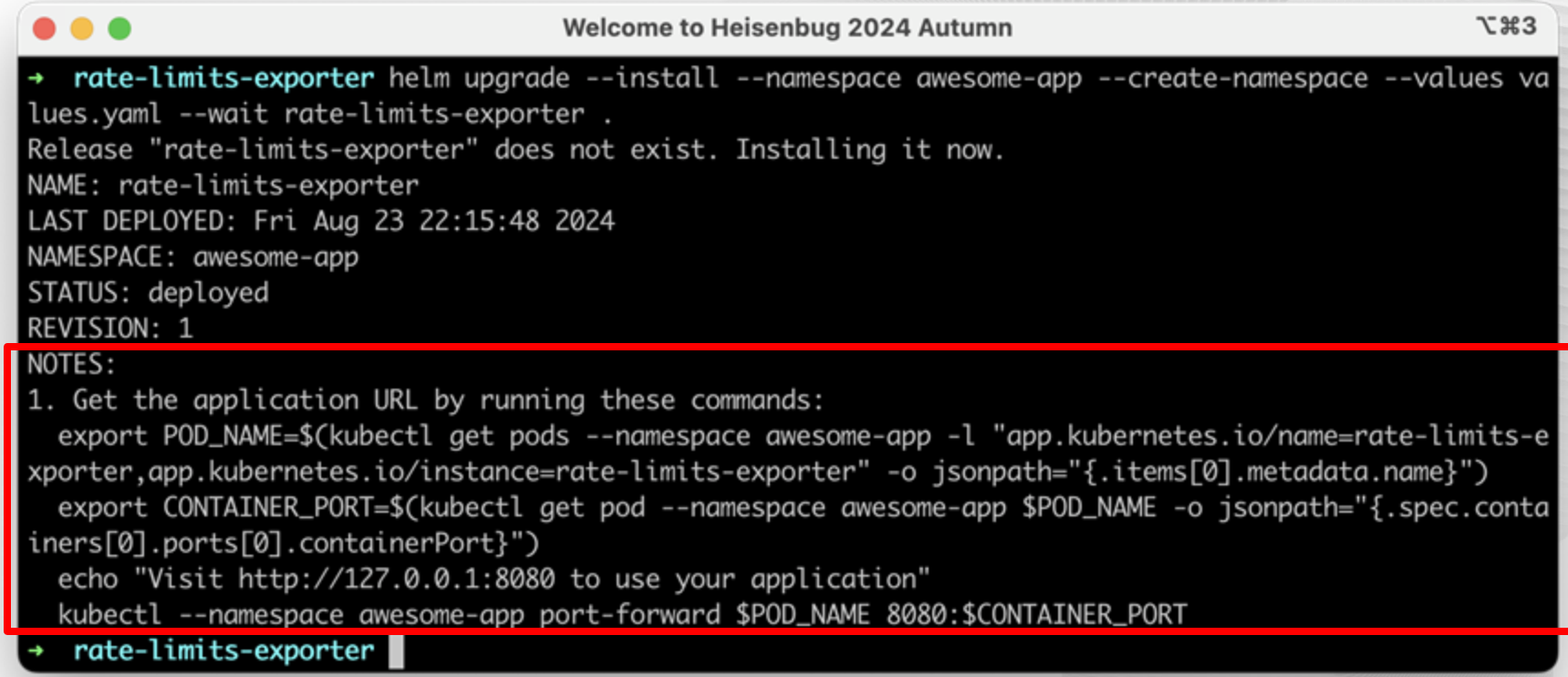


Деплоим чарт

- Используем **helm upgrade –install** для установки чарта
- Команда **helm install** тоже подходит, но при повторном выполнении появится ошибка, поскольку такой Helm-чарт уже будет присутствовать

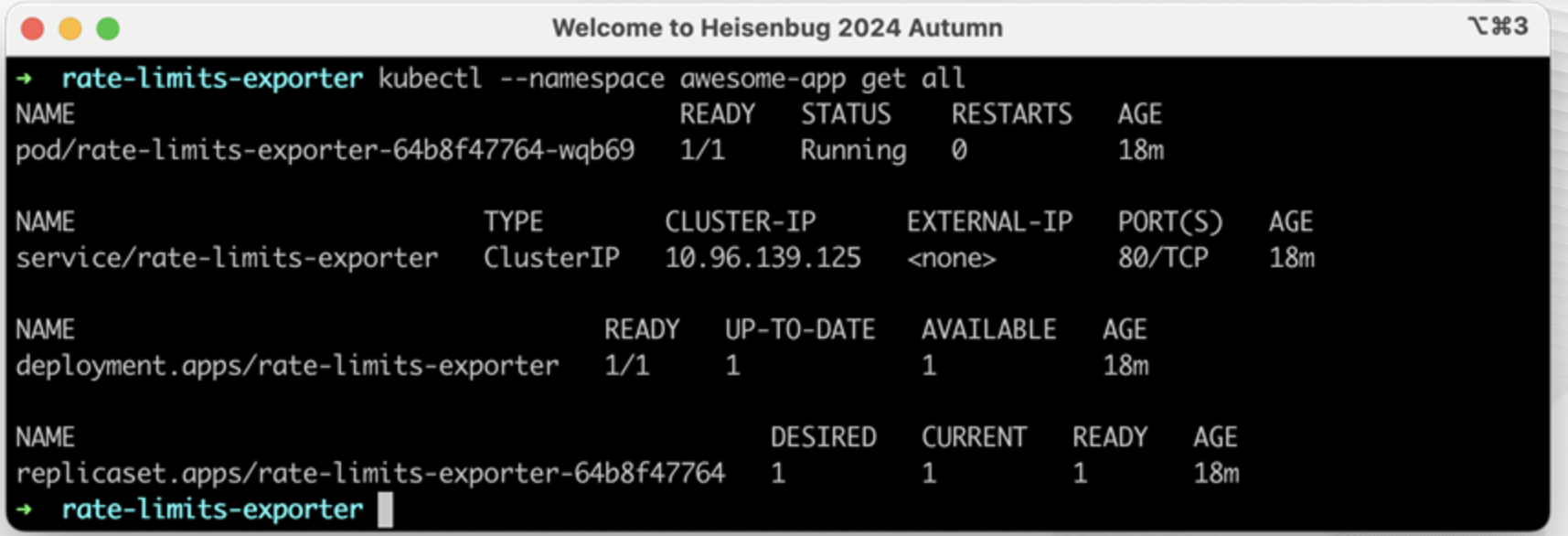
```
helm upgrade \
    --install \
1   --namespace awesome-app \
2   --create-namespace \
3   --values values.yaml \
4   --wait \
    rate-limits-exporter .
```

Деплоим чарт



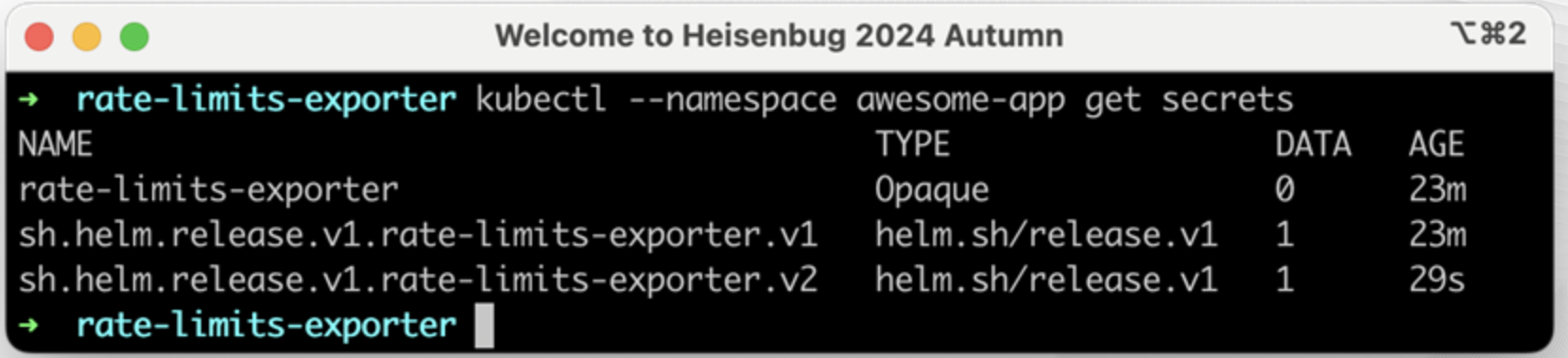
Деплоим чарт

- Проверим какие ресурсы были созданы в кластере
- Выполним **kubectl get,** укажем пространство имен и типа ресурса



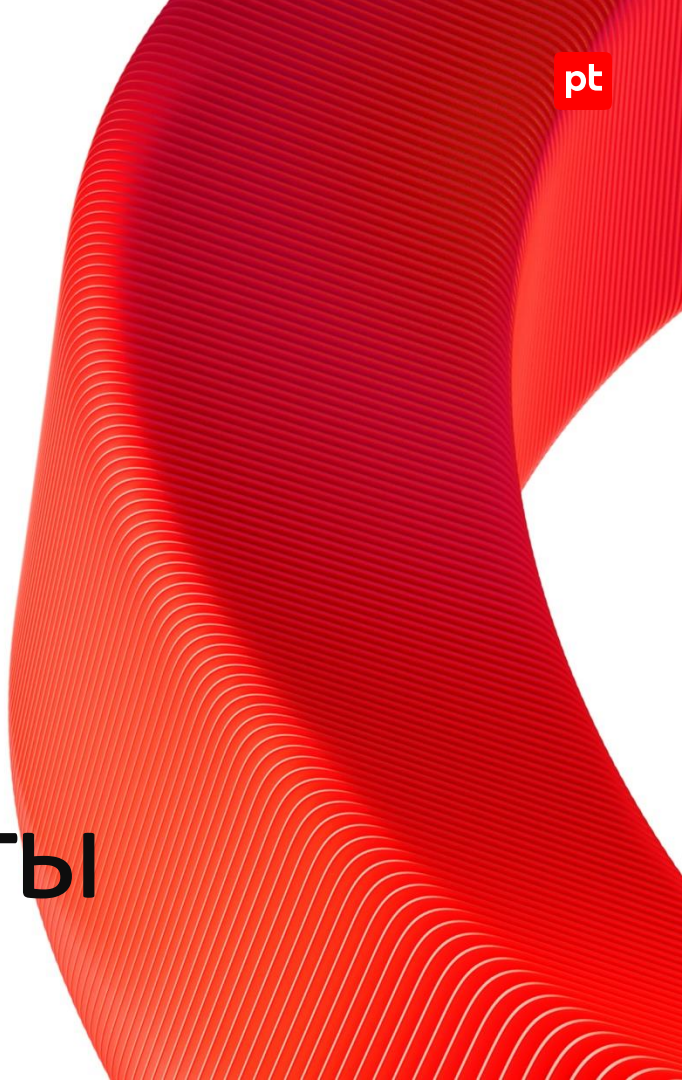
Деплоим чарт

- Важно помнить, что Helm хранит состояние
- Helm 3 делает это по умолчанию в секретах того неймспейса, куда деплоится чарт
- Helm 2 делал это по-другому



03

Как тестировать чарты



Тестируем чарт

- Helm имеет **встроенный инструмент тестирования** чартов
- **Основан на механизме хуков**, которые запускаются при наступлении определенных событий
- Нас интересует конечно же событие **test**

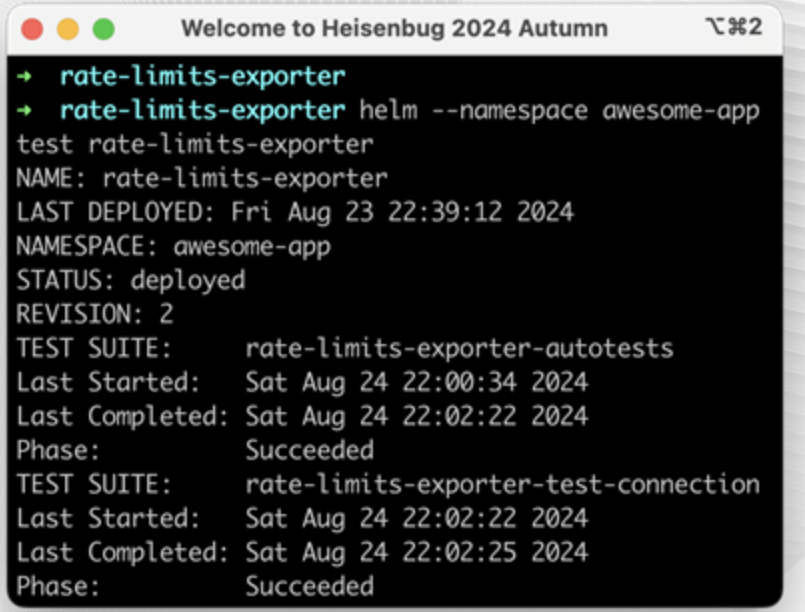
Annotation Value	Description
pre-install	Executes after templates are rendered, but before any resources are created in Kubernetes
post-install	Executes after all resources are loaded into Kubernetes
pre-delete	Executes on a deletion request before any resources are deleted from Kubernetes
post-delete	Executes on a deletion request after all of the release's resources have been deleted
pre-upgrade	Executes on an upgrade request after templates are rendered, but before any resources are updated
post-upgrade	Executes on an upgrade request after all resources have been upgraded
pre-rollback	Executes on a rollback request after templates are rendered, but before any resources are rolled back
post-rollback	Executes on a rollback request after all resources have been modified
test	Executes when the Helm test subcommand is invoked (view test docs)

Тестируем чарт

- Обычный манифест Kubernetes
- Должен иметь аннотацию **`helm.sh/hook: test`**
- **Внутри манифеста - под**, который запускает задачи
- Тест считается успешным, если **контейнер пода завершился** со статусом 0 (exit 0)

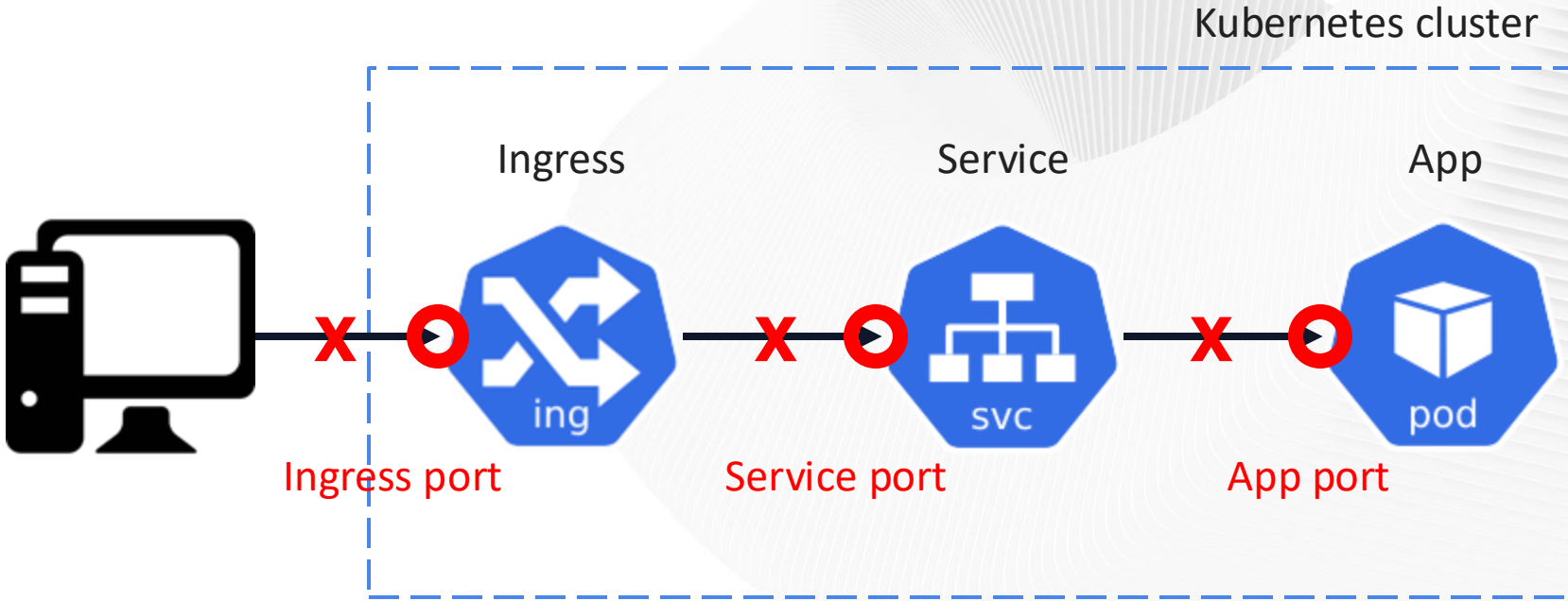
Тестируем чарт

- Можно определить **сколько угодно** тестовых манифестов
- Все тестовые манифесты должны находиться в каталогах **templates/** или **templates/tests/**
- Запускаем тесты командой **helm test**
- Helm дожидается их выполнения и выводит соответствующий статус

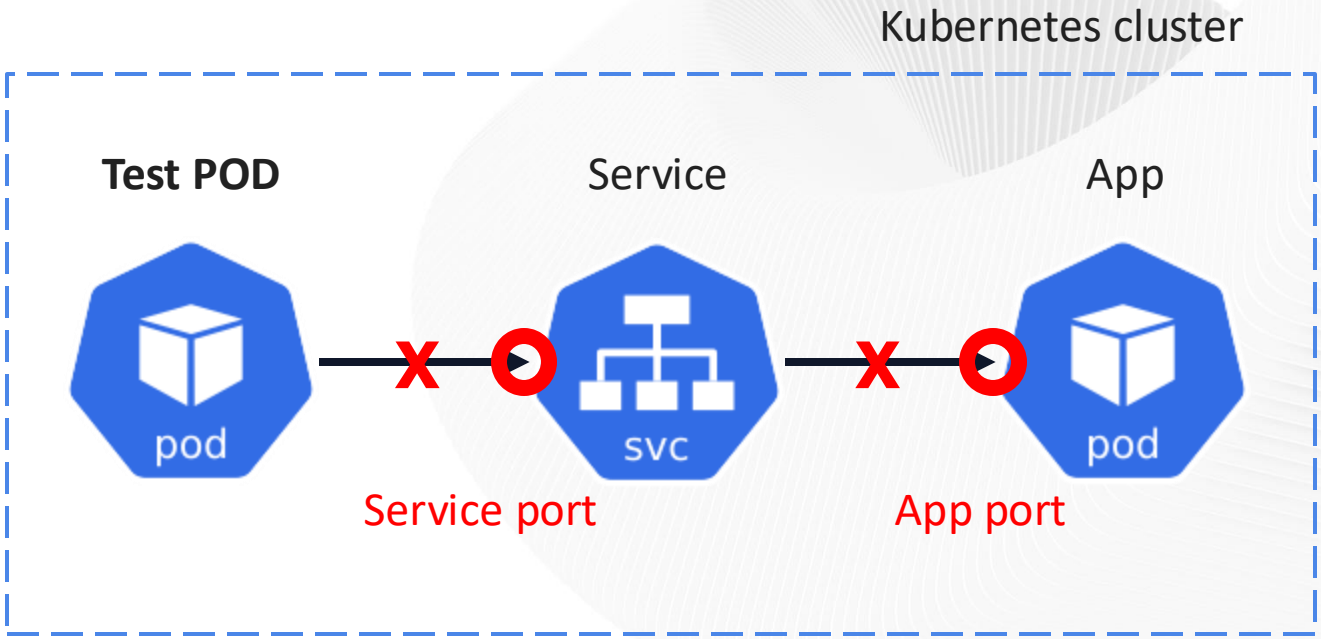


Что именно мы тестируем?

Тестируем чарт



Тестируем чарт



Как будет выглядеть тест?

Тестируем чарт

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: "{{ include "rate-limits-exporter.fullname" . }}-test-connection"
  labels:
    {{- include "rate-limits-exporter.labels" . | nindent 4 }}
  annotations:
    "helm.sh/hook": test
    "kube-score/ignore": pod-probes,container-image-pull-policy,pod-networkpolicy
spec:
  containers:
    - name: wget
      image: busybox:1.36.1
      workingDir: /tmp
      command: ['wget']
      args: ['{{ include "rate-limits-exporter.fullname" . }}:{{ .Values.service.port }}']
```

А что на счет автотестов?

Тестируем чарт

```python
import os
import re
import requests
from pytest_check import check


url = os.getenv('APP_URL', 'http://0.0.0.0:8080')


def test_status():
    r = requests.get(url)
    # using soft assertion here
    check.equal(r.status_code, 200)
    check.equal(r.encoding, 'utf-8')


def test_metrics():
    r = requests.get(f'{url}/metrics')
    assert re.search(r'dockerhub_ratelimit_scrape_error\{.*\} 0|1', r.text)


def test_healthz():
    r = requests.get(f'{url}/healthz')
    assert r.text == 'Ok'
```

Где хранить?

Как запаковать?

Тестируем чарт

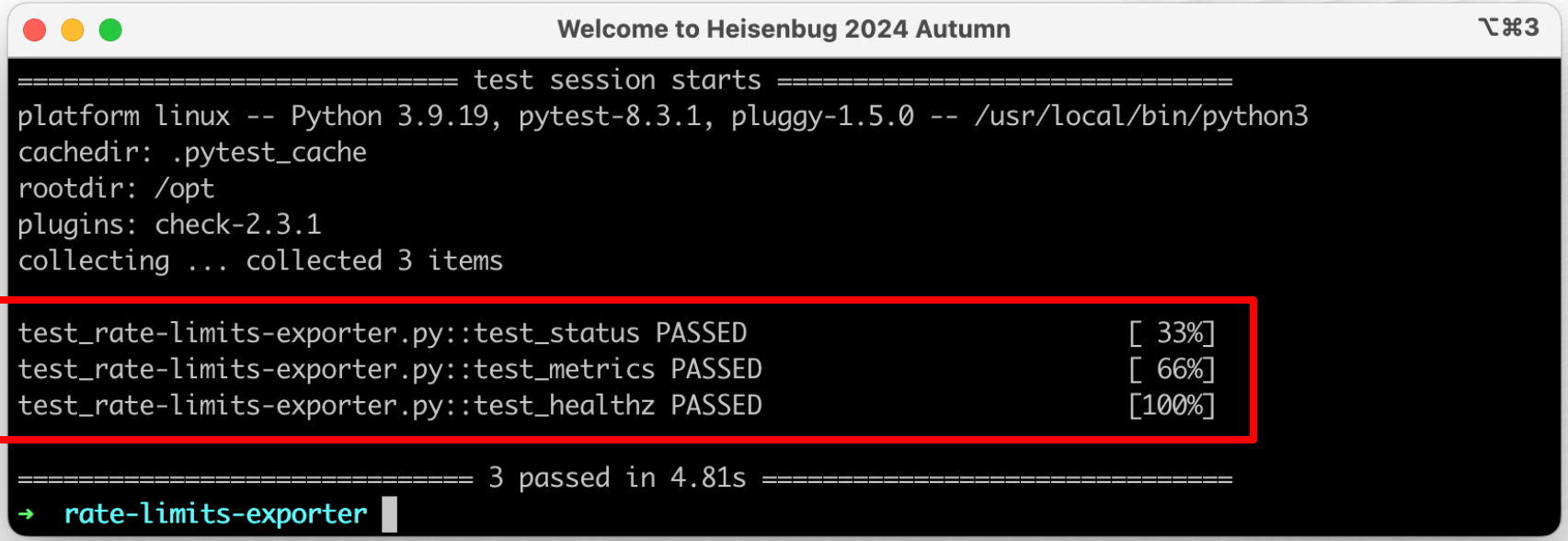
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: "{{ include "rate-limits-exporter.fullname" . }}-autotests"
  labels:
    {{- include "rate-limits-exporter.labels" . | nindent 4 }}
# accessing files using helm https://helm.sh/docs/chart_template_guide/accessing_files/
data:
{{- range $path, $_ := .Files.Glob "auto-tests/*" }}
{{ base $path | indent 2 }}: |
{{ $.Files.Get $path | indent 4 }}
{{- end }}
```

Тестируем чарт

```
apiVersion: v1
kind: Pod
metadata:
  name: "{{ include "rate-limits-exporter.fullname" . }}-autotests"
  labels:
    {{- include "rate-limits-exporter.labels" . | nindent 4 }}
  annotations:
    "helm.sh/hook": test
    "kube-score/ignore": pod-probes,container-image-pull-policy,pod-networkpolicy,container-security-
spec:
  containers:
    - name: autotests
      image: python:3.9.19-slim-bookworm
      workingDir: /opt
      command: ["/bin/bash", "-c"]
      args: ['python3 -m pip install -r requirements.txt ; python3 -m pytest -v']
      env:
        - name: APP_URL
          value: "http://{{ include "rate-limits-exporter.fullname" . }}:{{ .Values.service.port }}"
```

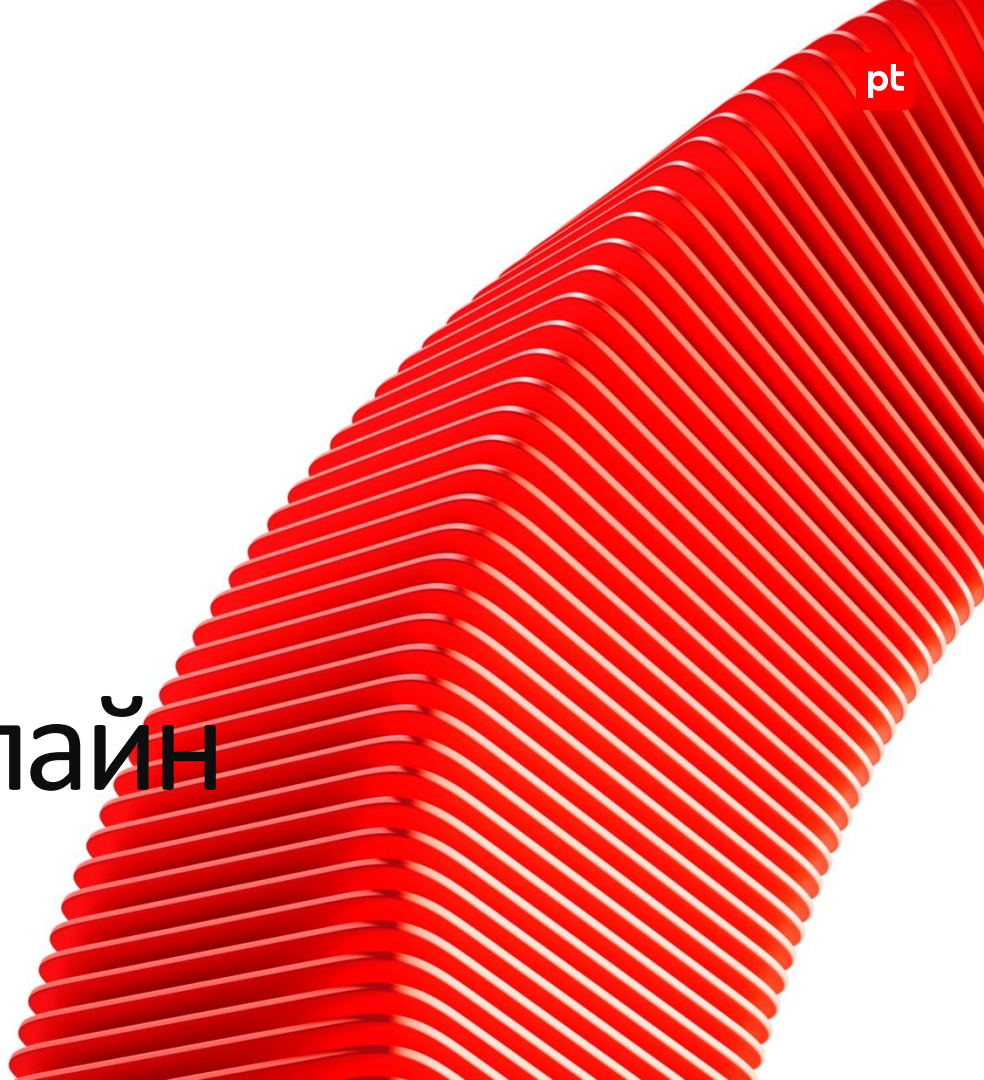
Тестируем чарт

kubectl -namespace awesome-app logs pod/rate-limits-exporter-autotests



04

Создаем пайплайн



Создаем пайплайн

Почему GitHub Actions?

- Доступен всем совершенно бесплатно с безлимитным временем CI/CD (для публичных проектов)
- Минимальный порог входа и большое сообщество

Создаем пайплайн

```
1 - name: 'Install Helm'
    uses: azure/setup-helm@v4.2.0
    with:
      version: ${{ env.HELM_VERSION }}

2 - name: 'Install kubectl'
    uses: azure/setup-kubectl@v3
    with:
      version: ${{ env.KUBERNETES_CLUSTER_VERSION }}

3 - name: 'Install kind and create Kubernetes cluster'
    uses: helm/kind-action@v1
    with:
      version: ${{ env.KIND_VERSION }}
      node_image: kindest/node:${{ env. KUBERNETES_CLUSTER_VERSION }}

4 - name: 'Install kube-score'
    run: |
      curl -Lo ./kube-score.tar.gz https://github.com/zegl/kube-score/r
      tar -zxvf kube-score.tar.gz
      chmod +x kube-score
      sudo mv kube-score /usr/local/bin/
```

Pure shell

GitHub Workflow

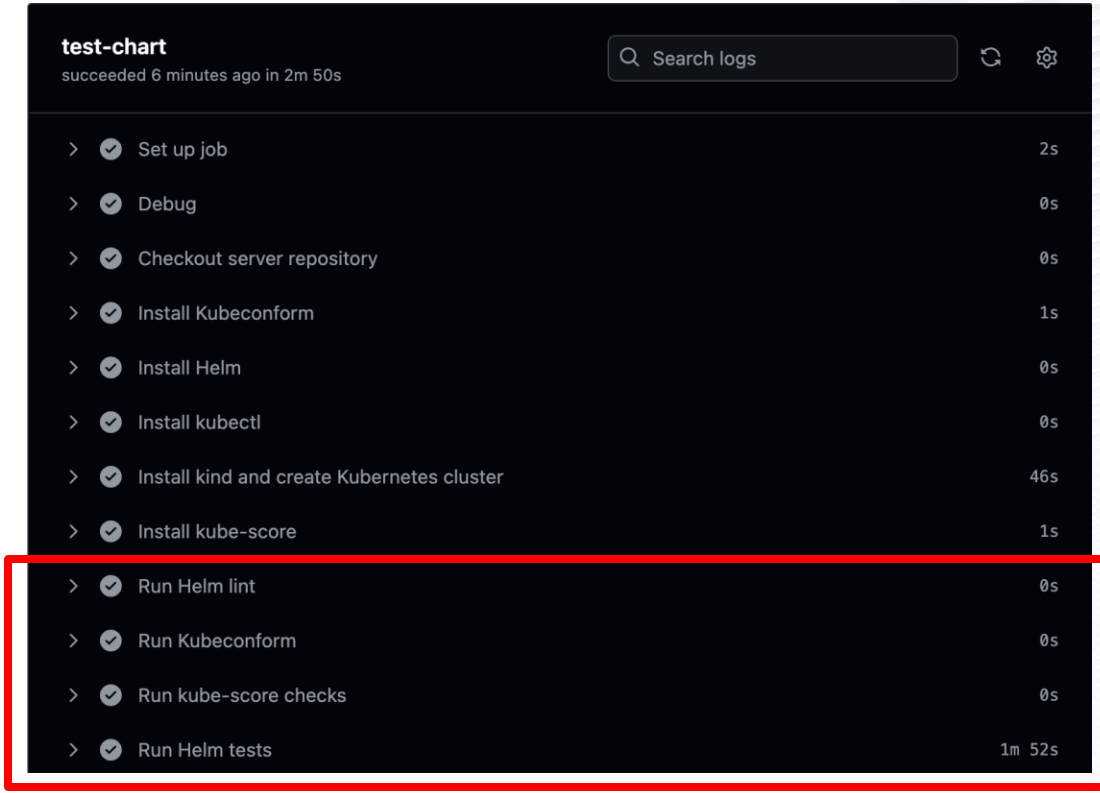
Создаем пайплайн

```
- name: 'Run Helm lint'
  run: |
    helm lint chart

- name: 'Run Kubeconform'
  working-directory: chart
  run: |
    helm template . | kubeconform -kubernetes-version ${KUBERNETES_CLUSTER_VERSION//v}

- name: 'Run kube-score checks'
  working-directory: chart
  run: |
    helm template . | kube-score score -

- name: 'Run Helm tests'
  working-directory: chart
  run: |
    helm upgrade \
      --install \
      --namespace ${NAMESPACE} \
      --create-namespace \
      --wait \
      ${CHART_NAME} .

    helm --namespace ${NAMESPACE} \
      test ${CHART_NAME}
```

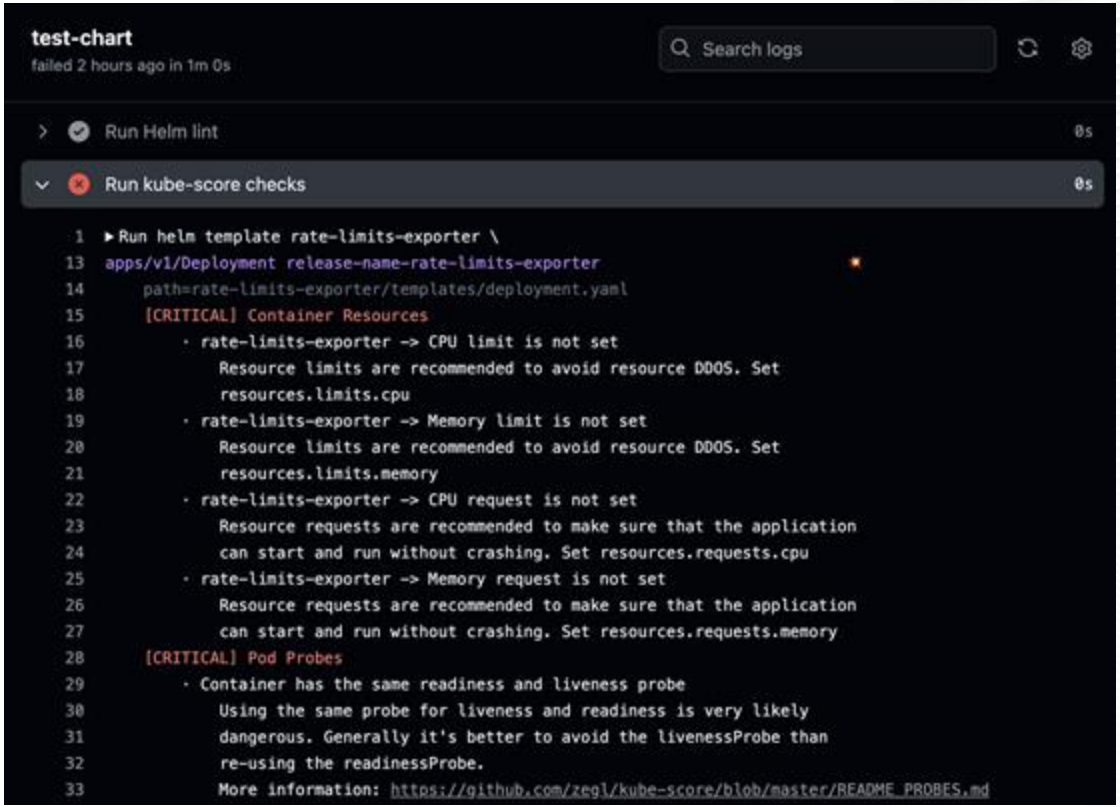
Install

Test

Создаем пайплайн

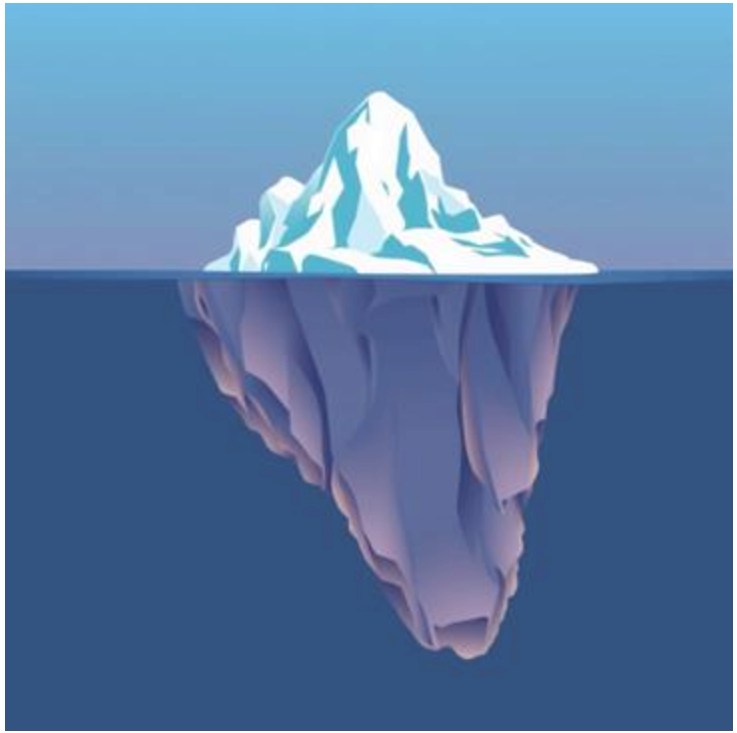


Создаем пайплайн



Заключение

Заключение

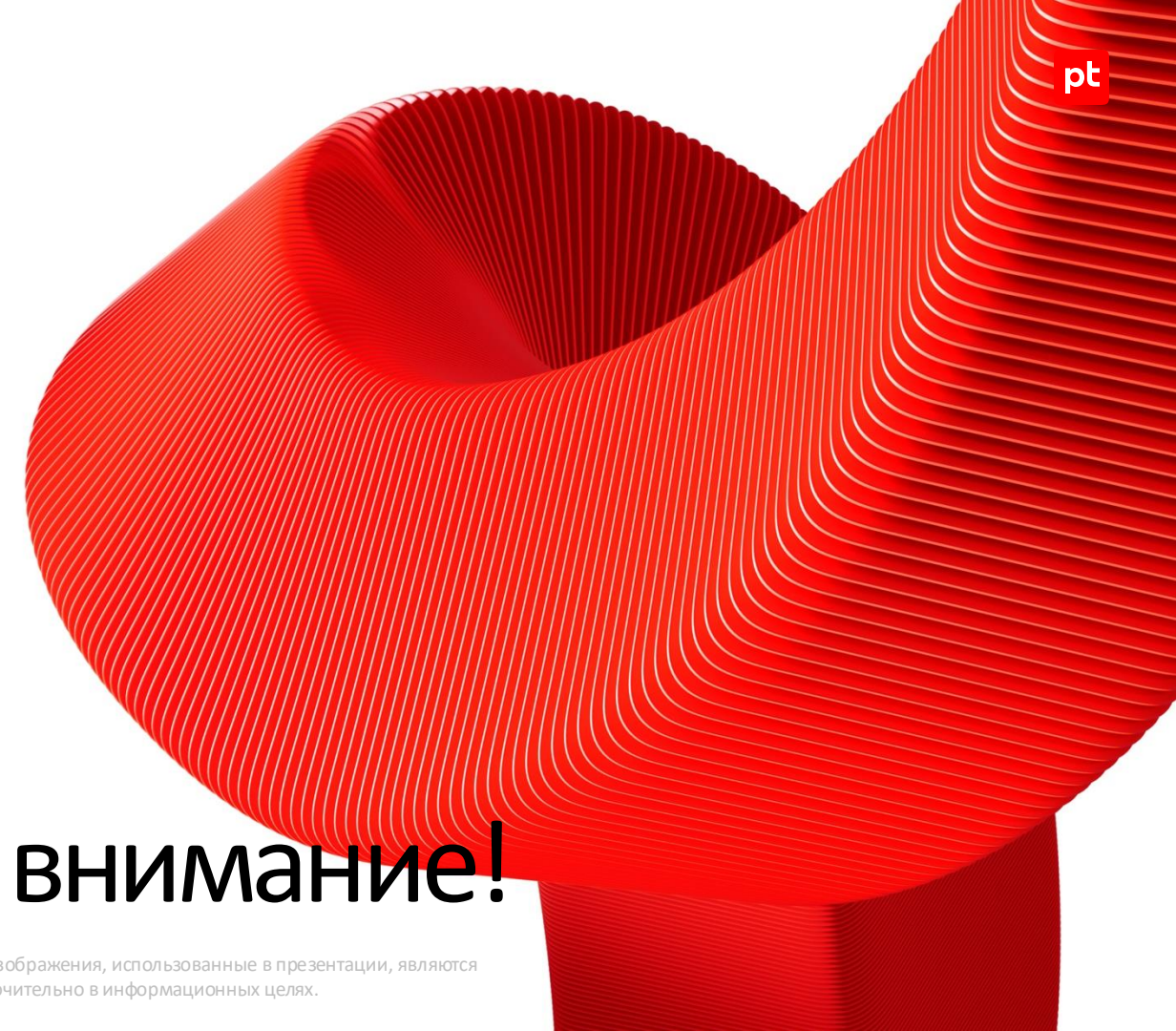


Application

Infrastructure

Image generated by https://nplod.ru/

Вопросы



Спасибо за внимание!