




ОНЛАЙН-ОБРАЗОВАНИЕ

# Онлайн-образование





# Меня хорошо видно && слышно?

Ставьте  , если все хорошо  
Напишите в чат, если есть проблемы  
заодно проверяем, включена ли запись занятия



Включил Юджин запись ли ты







# Введение в Kubernetes. Часть 2



Непомнящий Евгений

telegram @EvgeniyN



# Правила вебинара



Активно участвуем



Задаем вопрос в чат



Вопросы вижу в чате, могу ответить не сразу

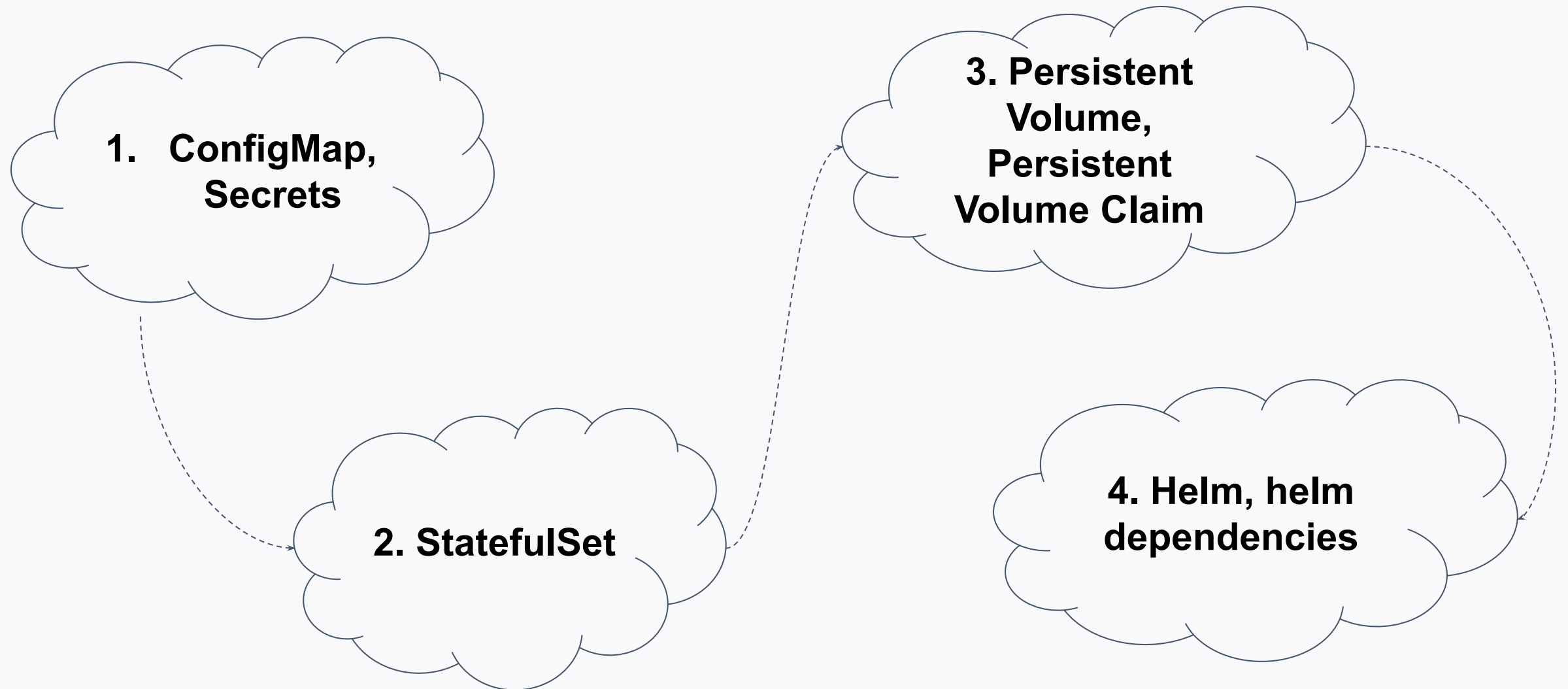
# Цели вебинара | После занятия вы сможете

**1 Иметь представление внутреннем устройстве Kubernetes**

# СМЫСЛ | Зачем вам это уметь, в результате:

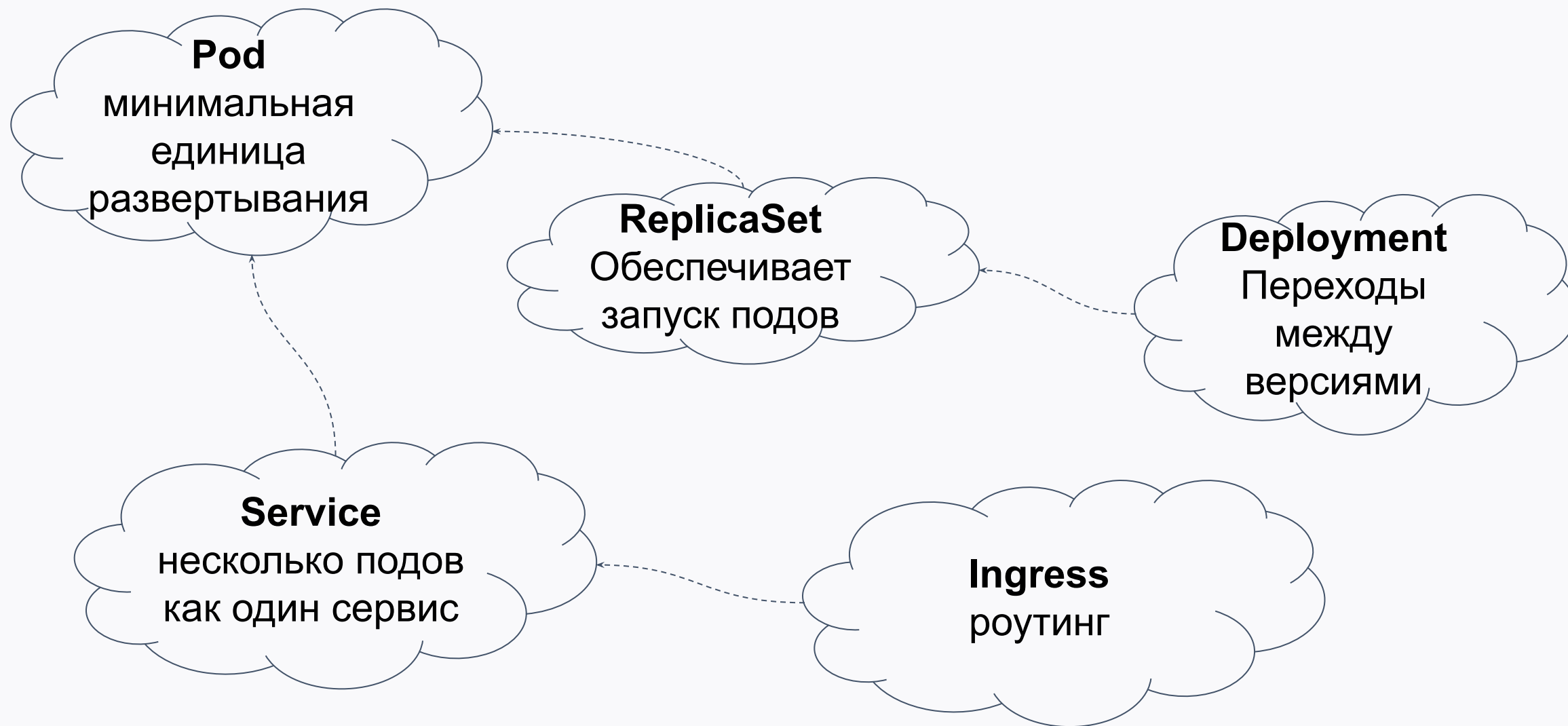
**1 Уметь развернуть приложение с базой данных, обеспечить доступ к нему извне кластера**

# Маршрут вебинара





# Что вы помните с прошлого занятия?





The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this is a semi-transparent network pattern consisting of numerous small dots connected by thin lines, creating a mesh-like effect. The word "ConfigMap" is centered in the middle of the slide in a large, white, sans-serif font.

# ConfigMap



# ConfigMap

Предположим, мы хотим передать в под некоторые параметры

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello-py-app
  template:
    metadata:
      labels:
        app: hello-py-app
    spec:
      containers:
        - name: hello-py
          image: aristoveugene/hello-py:v3
          env:
            - name: DATABASE_URI
              value: postgresql+psycpg2://myuser:passwd@postgres/myapp
          ports:
            - name: web
              containerPort: 80
```

Чем это плохо?

# ConfigMap

Давайте разделим описание ресурса и параметры

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: game-demo
data:
  # property-like keys; each key maps to a simple value
  player_initial_lives: "3"

  # file-like keys
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
```

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-demo-pod
spec:
  containers:
    - name: demo
      image: pg_kub2_config:v2
      env:
        # Define the environment variable
        - name: PLAYER_INITIAL_LIVES # Notice that the case is different here
          # from the key name in the ConfigMap.
          valueFrom:
            configMapKeyRef:
              name: game-demo # The ConfigMap this value comes from.
              key: player_initial_lives # The key to fetch.
      volumeMounts:
        - name: config
          mountPath: "/config"
          readOnly: true
      ports:
        - name: web
          containerPort: 80
  volumes:
    # You set volumes at the Pod level, then mount them into containers inside that Pod
    - name: config
      configMap:
        # Provide the name of the ConfigMap you want to mount.
        name: game-demo
        # An array of keys from the ConfigMap to create as files
        items:
          - key: "game.properties"
            path: "game.properties"
```



# ConfigMap

1. Команда запуска
2. Переменные среды
3. Монтирование в файл
4. Чтение через апи k8s

<https://habr.com/ru/company/flant/blog/498970/>





Secret





# Secret

Как передать пароли, токены, ключи шифрования?

## Секреты

- очень похожи на ConfigMap
- разные типы
- МОЖНО  
ресурсы  
с помощью команд
- МОЖНО  
в переменные  
или в файлы

переменные

Builtin Type	Usage
Opaque	arbitrary user-defined data
kubernetes.io/service-account-token	service account token
kubernetes.io/dockercfg	serialized ~/.dockercfg file
kubernetes.io/dockerconfigjson	serialized ~/.docker/config.json file
kubernetes.io/basic-auth	credentials for basic authentication
kubernetes.io/ssh-auth	credentials for SSH authentication
kubernetes.io/tls	data for a TLS client or server
bootstrap.kubernetes.io/token	bootstrap token data



The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band that contains a white network diagram. This diagram consists of numerous small dots connected by thin white lines, creating a complex web-like structure that spans the width of the slide. Centered within this band is the text "PersistentVolume" in a large, white, sans-serif font.

# PersistentVolume



# Volume

- По умолчанию docker обеспечивает файловую систему, куда можно писать
- Однако эти изменения пропадают при перезапуске контейнера
- У k8s есть свой механизм Volume
- Можно считать, что это виртуальная папка с файлами
- Поведение зависит от типа
- <https://kubernetes.io/docs/concepts/storage/volumes/>

# PersistentVolume / PersistentVolumeClaim

- Pod требует хранилище (Claim)
- k8s связывает это требование с подходящим по классу и объему PersistentVolume
- PersistentVolume могут создаваться “на лету” или вручную
- Для minikube - storage-provisioner
- Данные в таком хранилище переживают перезапуск pod



The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this is a semi-transparent network pattern consisting of numerous small dots connected by thin lines, creating a web-like structure. The text 'StatefulSet' is centered in a large, white, sans-serif font.

# StatefulSet



# StatefulSet

- Коровы vs Домашние питомцы
- Представьте кластер постгреса
- Если один pod умер, его надо перезапустить с тем же именем и на тех же данных
- Нам важно знать наши экземпляры “в лицо”



# StatefulSet vs Deployment

Deployment	StatefulSet
Deployment is used to deploy stateless applications	<i>StatefulSets</i> is used to deploy stateful applications
Pods are interchangeable	Pods are not interchangeable. Each pod has a persistent identifier that it maintains across any rescheduling
Pod names are unique	Pod names are in sequential order
Service is required to interact with pods in a deployment	A Headless Service is responsible for the network identity of the pods
The specified PersistentVolumeClaim is shared by all pod replicas. In other words, shared volume	The specified volumeClaimTemplates so that each replica pod gets a unique PersistentVolumeClaim associated with it. In other words, no shared volume



The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this is a semi-transparent network pattern consisting of white dots connected by thin white lines, creating a web-like effect across the center of the image.

# Веб-сервис + postgres



# Практика

- Postgresql
  - StatefulSet
  - Service - нам для удобства, на проде возможно не нужен
- Веб-сервис
  - Deployment
  - Secret для пароля к БД
  - ConfigMap для параметров
  - Service

# Тест

- Сейчас, мини (1-2 мин)

<https://forms.gle/wGRohzY9ZCMGA4Xw5>

- Потом, 5-10 мин

<https://docs.google.com/forms/d/e/1FAIpQLSdwWQdgzawMw7jMFDAsatHHO6amO463SREo8Hb68H9GEvEU6Q/viewform>



The image features a blue-tinted aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. A semi-transparent blue band with a white network pattern of dots and lines runs horizontally across the middle of the image. The word "Helm" is centered in this band in a white, sans-serif font.

# Helm



# Юзкейс - многократный деплой

- prod, nfr, e2e, stage, dev
- отличия
  - postgres - разные инстансы на prod, nfr и на остальное тестирование, разные базы данных на разное тестирование
  - kafka - один инстанс, разные топики
  - образ - разные версии
  - разное количество реплик
- 50 микросервисов, postgres, kafka, rabbit, ...
- а давайте сделаем второй стенд для e2e



# Юзкейс - многократный деплой

- chart - аналог пакета
- содержимое строится по шаблонам (templates/), синтаксис mustache
- значения по умолчанию - values.yaml
- \_helpers.tpl - общие шаблоны ("функции")

# Юзкейс - установить готовое приложение

- `helm repo add confluentinc  
https://confluentinc.github.io/cp-helm-charts/`
- `helm repo update`
- `helm install cp confluentinc/cp-helm-charts -f cp_values.yaml`



# Юзкейс - зависимости

- Chart.yaml - dependencies
- helm dependency update app-dep
- helm install app app-dep --dry-run



The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City, featuring numerous skyscrapers. A semi-transparent blue band with a white geometric network pattern of dots and lines runs horizontally across the middle of the image, serving as a backdrop for the title text.

# Best practice



# Best practice

10 антипаттернов деплоя в Kubernetes: распространенные практики, для которых есть другие решения

# Опрос

<https://otus.ru/polls/32379/>



# Домашнее задание

Развернуть Постгрес в миникубе

- Устанавливаем minikube
- Разворачиваем PostgreSQL 14 через манифест
- Рекомендую еще развернуть сервис, работающий с БД
- Задание повышенной сложности\*
  - Разворачиваем PostgreSQL 14 с помощью helm



The image features a background of a dense city skyline, likely New York City, viewed from an elevated angle. The entire image is tinted with a blue and green color palette. A network of thin, light blue lines connects various points across the image, creating a digital or technological overlay. The text "Порефлексируем" is centered in the middle of the image, rendered in a bold, white, sans-serif font.


**Порефлексируем**



# Вопросы?

- **ConfigMap и Secret**
- **PersistentVolume**
- **StatefulSet**
- **Helm**





Спасибо за внимание!  
Приходите на следующие вебинары

Непомнящий Евгений