2022

Инструкция администратора системы МСР

ВЕРСИЯ 1.0

Оглавление

1. Требования для деплоя приложения

1.1. Кластер Kubernetes.

1.2. Клиентские утилиты для управления кластером Kubernetes.

1.3. База данных PostgreSQL.

1.4. DNS-домен.

2. Подготовка к деплою

2.1. Установка и настройка кластера Kubernetes

Требования к кластеру Kubernetes:

- Наличие `Ingress Controller`.
- Поддержка `PV` и `PVC` (необходимо только в случае, если PostgreSQL будет установлен в Kubernetes).

Kubernetes-кластер может быть развернут на собственных ресурсах, либо можно использовать Managed Kubernetes в каком-либо облаке.

Поддерживаются версии Kubernetes 1.22 и выше.

2.1.1. Managed Service for Kubernetes в Yandex Cloud

Создайте кластер Managed Service for Kubernetes и группу узлов, и управляйте ими с помощью `kubectl` — командной оболочки Kubernetes.

2.1.1.1. Перед началом работы

Чтобы начать работать с сервисом Managed Service for Kubernetes:

1. Перейдите в консоль управления, затем войдите в Yandex Cloud или зарегистрируйтесь, если вы еще не зарегистрированы.
2. На странице биллинга убедитесь, что у вас подключен платежный аккаунт, и он находится в статусе `ACTIVE` или `TRIAL_ACTIVE`. Если платежного аккаунта нет, создайте его.
3. Если у вас еще нет каталога, создайте его.
4. Установите Kubernetes CLI (`kubectl`).
5. Убедитесь, что у вас достаточно свободных ресурсов в облаке.
6. Если у вас еще нет сети, создайте ее.
7. Если у вас еще нет подсетей, создайте их в зонах доступности, где будут созданы кластер Kubernetes и группа узлов.

8. Создайте сервисные аккаунты:

    - С ролью `editor` на каталог, в котором создается кластер. От имени этого сервисного аккаунта будут создаваться ресурсы, необходимые кластеру Managed Service for Kubernetes.
    - С ролью `container-registry.images.puller` на каталог с реестром Docker-образов. От его имени узлы будут скачивать из реестра необходимые Docker-образы.

    Вы можете использовать один и тот же сервисный аккаунт для обеих операций.

    Примечание

    Для создания кластера с туннельным режимом его сервисному аккаунту необходима роль `k8s.tunnelClusters.agent`.

2.1.1.2. Создайте кластер Managed Service for Kubernetes

1. В консоли управления выберите каталог, в котором нужно создать кластер Managed Service for Kubernetes.
2. Выберите сервис **Managed Service for Kubernetes**.
3. Нажмите кнопку **Создать кластер**.
4. Введите имя кластера. Оно должно быть уникальным в рамках каталога.
5. (Опционально) Введите описание кластера.
6. **Сервисный аккаунт для ресурсов** — укажите сервисный аккаунт с ролью editor, который будет использоваться для создания ресурсов.
7. **Сервисный аккаунт для узлов** — укажите сервисный аккаунт с ролью `container-registry.images.puller`, который будет использоваться узлами для доступа к реестру Docker-образов.
8. Укажите релизный канал. Эту настройку невозможно изменить после создания кластера.
9. В блоке **Конфигурация мастера**:

- **Версия Kubernetes** выберите версию Kubernetes, которая будет установлена на мастере.
- **Публичный адрес** выберите способ назначения адреса:
    - **Автоматически** — чтобы назначить случайный IP-адрес из пула адресов Yandex Cloud.
    - **Без адреса** — чтобы не назначать публичный IP-адрес.

- **Тип мастера** — выберите тип мастера:
    - **Зональный** — будет создан один хост-мастер в выбранной зоне доступности. Укажите облачную сеть и выберите в ней подсеть для размещения хоста-мастера.
    - **Региональный** — в каждой зоне доступности будет создано по одному хосту-мастеру. Укажите облачную сеть и подсеть для каждой зоны доступности.

- Выберите группы безопасности для сетевого трафика кластера.

10. В блоке **Сетевые настройки кластера**:

- **CIDR кластера** — укажите диапазон IP-адресов, из которого будут выделяться IP-адреса для подов.
- **CIDR сервисов** — укажите диапазон IP-адресов, из которого будут выделяться IP-адреса для сервисов.
- Задайте маску подсети узлов и максимальное количество подов в узле.

11. Нажмите кнопку **Создать кластер**.

2.1.1.3. Добавьте учетные данные в конфигурационный файл kubectl

Если у вас еще нет интерфейса командной строки Yandex Cloud, установите и инициализируйте его.

По умолчанию используется каталог, указанный в профиле CLI. Вы можете указать другой каталог с помощью параметра `--folder-name` или `--folder-id`.

Чтобы добавить учетные данные кластера Managed Service for Kubernetes в конфигурационный файл `kubectl`:

1. Выполните команду:

2. `yc managed-kubernetes cluster get-credentials test-k8s-cluster --external`

    - По умолчанию учетные данные добавляются в директорию `$HOME/.kube/config`.
    - Если необходимо изменить расположение конфигураций, используйте флаг `--kubeconfig <путь к файлу>`.

3. Проверьте конфигурацию `kubectl` после добавления учетных данных:

4. `kubectl config view`

Результат выполнения команды:

```
apiVersion: v1
clusters:
  - cluster:
    certificate-authority-data: DATA+OMITTED
```

…

2.1.1.4. Создайте группу узлов

Чтобы создать группу узлов:

1. В консоли управления выберите каталог, в котором создан нужный кластер Managed Service for Kubernetes.

2. В списке сервисов выберите **Managed Service for Kubernetes**.
3. Выберите кластер Managed Service for Kubernetes, для которого необходимо создать группу узлов.
4. На странице кластера Managed Service for Kubernetes перейдите на вкладку **Управление узлами**.
5. Нажмите кнопку **Создать группу узлов**.
6. Введите имя и описание группы узлов.
7. Укажите **Версию Kubernetes** для узлов.
8. В блоке **Масштабирование** выберите его тип:

    - **Фиксированный**, чтобы количество узлов в группе оставалось неизменным. Укажите количество узлов в группе.
    - **Автоматический**, чтобы управлять количеством узлов в группе с помощью автоматического масштабирования кластера.

9. В блоке **В процессе создания и обновления разрешено** укажите максимальное количество виртуальных машин, на которое можно превысить и уменьшить размер группы.

10. В блоке **Вычислительные ресурсы**:

    - Выберите платформу.
    - Укажите необходимое количество vCPU и гарантированную долю vCPU, а также объем RAM.
    - (опционально) Укажите, что ВМ должна быть прерываемой.

11. В блоке **Хранилище**:

    - Укажите **Тип диска** для узлов группы:

    - **HDD** — стандартный сетевой диск, сетевое блочное хранилище на HDD-накопителе.
    - **SSD** — быстрый сетевой диск, сетевое блочное хранилище на SSD-накопителе.
    - **Нереплицируемый SSD** — сетевой диск с повышенной производительностью, реализованной за счет устранения избыточности.
- Укажите размер дисков для узлов группы.

12. В блоке **Сетевые настройки**:

    - В поле Публичный адрес выберите способ назначения адреса:
        - **Автоматически** — чтобы назначить случайный IP-адрес из пула адресов Yandex Cloud.
        - **Без адреса** — чтобы не назначать публичный IP-адрес.
    - Выберите группы безопасности.
    - Выберите зону доступности и подсеть для размещения узлов группы.

13. В блоке **Доступ** укажите данные для доступа к узлам группы по SSH:

    - **Логин** — укажите имя пользователя.
    - **SSH-ключ** — вставьте содержимое файла публичного ключа.

14. Нажмите кнопку **Создать**.

2.1.2. Managed Service for Kubernetes в VK Cloud

2.1.2.1. Описание

Кластер Kubernetes состоит из набора машин, так называемых узлов (node's), которые запускают контейнеризированные приложения. Кластер должен иметь как минимум один рабочий узел.

На рабочих узлах размещены поды (pod's), являющиеся компонентами приложения. Внутренние сервисы Kubernetes управляют рабочими узлами и подами в кластере. В промышленных управляющие сервисы обычно запускается на нескольких компьютерах, а кластер, как правило, развёртывается на нескольких узлах, гарантируя отказоустойчивость и высокую надёжность.

Топология кластеров в VK Cloud включает в себя понятие мастер-узлов, на которых располагаются управляющие сервисы и групп рабочих узлов (Node Group), на которых запущены приложения пользователь. Каждый кластер Kubernetes может содержать несколько групп рабочих узлов, каждая из которых создана на базе определенного шаблона виртуальной машины.

2.1.2.2. Создание кластера

Для создания кластера следует перейти в раздел панели управления VK Cloud "Контейнеры" и нажать кнопку "Создать кластер" или "Добавить", если у вас уже есть кластеры Kubernetes.

Затем следует выбрать желаемую конфигурацию кластера. Преднастроенные типы сред отличаются параметрами масштабирования и активированными расширениями:

- Dev среда (разработка)
- Staging среда (пред-релизная среда)
- Production (боевое или живое окружение, для такой среды рекомендуется минимум 3 мастер-ноды)

Также вы можете выбрать какие предустановленные аддоны будут активированы в кластере:

- Мониторинг на базе Prometheus Operator и Grafana;
- Docker Registry, хранящий данные images в объектном хранилище VK Cloud;
- Nginx Ingress Controller, для которого дополнительно создается балансировщик нагрузки.

После нажатия кнопки "Следующий шаг" необходимо выбрать конфигурацию кластера:

Мастер создания отражает следующие поля конфигурации:

Поле	**Описание**
Тип виртуальной машины Master	Конфигурация головного узла - количество CPU и оперативной памяти управляющей машины ("Master"). Для продуктивных сред, рекомендуется минимальная конфигурация CPU=2 RAM=4Gb.
Количество узлов Master	Количество управляющих машин (машин типа "Master"). Для production-сред рекомендуется минимум 3 мастер-ноды.
Размер диска на Master-узле, GB	Объем диска на управляющих Master-нодах.
Тип виртуальной машины Node	Тип конфигурации для рабочего узла - количество CPU и оперативной памяти.

Поле	**Описание**
Количество узлов Node	Количество рабочих узлов.
Размер диска на Node-узле, GB	Объем диска для рабочих узлов.
Тип дисков для Master и Node-узлов	Тип дисков (HDD/SSD).
Версия Kubernetes	Версия Kubernetes, которая будет установлена на ноды кластера.
Сеть	Сеть, в которой будет развернут кластер, если она создана. Если сети нет, то для кластера создастся собственная приватная сеть.
Использовать сеть Load Balancer	По умолчанию балансировщик нагрузки создается в приватной сети кластера, указанной в пункте Сеть. Это опция позволяет создать балансировщик нагрузки в другой приватной сети доступной в рамках текущего проекта.
Использовать подсеть пода	Опция позволяет изменить дефолтную сеть для подов. В терминологии kubernetes это `pod-cidr-network`

Поле	Описание
Назначить внешний IP	Если этот переключатель включен, то будет назначен публичный IPv4 адрес для балансировщика нагрузки, управляющего Kubernetes API Server, а также для балансировщика нагрузки, отвечающего за Ingress Controller.Также можно выбрать создание кластера без внешнего IP для обоих балансировщиков. В этом случае для доступа к кластеру потребуется создание VPN-соединения.
Имя кластера	Имя кластера.
Ключ виртуальной машины	Если ключевая пара уже создана, рекомендуется указать её. Необходимо убедиться, что файл от нее сохранен. Также можно создать новую ключевую пару - она нужна для подключения к нодам кластера по ssh.

На следующем шаге будет предложено выбрать количество узлов Node в Node Group по умолчанию. На этапе создания кластера вы можете добавить несколько Node Group разного размера с разными размерами рабочих узлов. Также для каждого Node Group можно активировать параметры автоматического масштабирования.

Для завершения создания кластера потребуется некоторое время.

Все операции, перечисленные выше могут быть выполнены с помощью VK Cloud Terraform Provider.

2.2. Установка и настройка клиентских утилит кластером Kubernetes

Для управления кластером Вам понадобится утилита `kubectl`, установить её можно по инструкции ниже. Версия утилиты должна совпадать с версией кластера Kubernetes.

Также для деплоя приложения Вам понадобится утилита `helm`, которую можно установить по инструкции ниже. Необходимо использовать версию >= 3.10.

После создания кластера Kubernetes у Вас должен быть `kubeconfig`-файл – файл конфигурации для подключения к кластеру. Его нужно поместить по пути `~/.kube/config`. Подробнее про `kubeconfig` можно почитать в инструкции.

Для проверки подключения к кластеру можно выполнить следующую команду в консоли:

```
kubectl get node
```

В результате вы должны получить список узлов вашего кластера. Пример результата:

```
NAME          STATUS   ROLES    AGE     VERSION
kube-node-0   Ready    <none>   7d13h   v1.22.4
kube-node-1   Ready    <none>   7d13h   v1.22.4
```

2.2.1. Установка и настройка kubectl

Инструмент командной строки Kubernetes kubectl позволяет запускать команды для кластеров Kubernetes. Вы можете использовать `kubectl` для развертывания приложений, проверки и управления ресурсов кластера, а также для просмотра логов. Полный список операций `kubectl` смотрите в Overview of kubectl.

2.2.1.1. Подготовка к работе

Используемая вами мажорная версия `kubectl` не должна отличаться от той, которая используется в кластере. Например, версия v1.2 может работать с версиями v1.1, v1.2 и v1.3. Использование последней версии `kubectl` поможет избежать непредвиденных проблем.

2.2.1.2. Установка kubectl в Linux

2.2.1.2.1. Установка двоичного файла kubectl с помощью curl в Linux

1. Загрузите последнюю версию с помощью команды:
2. ```
   curl -LO
   https://storage.googleapis.com/kubernetes-
   release/release/`curl -s
   https://storage.googleapis.com/kubernetes-
   release/release/stable.txt`/bin/linux/amd64/kubect
   l
   ```

   Чтобы загрузить определенную версию, вставьте в фрагмент команды `$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)` нужную версию.

   Например, команда загрузки версии v1.25.0 для Linux будет выглядеть следующим образом:

   ```
   curl -LO
   https://storage.googleapis.com/kubernetes-
   release/release/v1.25.0/bin/linux/amd64/kubectl
   ```
3. Сделайте двоичный файл `kubectl` исполняемым:
4. ```
   chmod +x ./kubectl
   ```
5. Переместите двоичный файл в директорию из переменной окружения PATH:
6. ```
   sudo mv ./kubectl /usr/local/bin/kubectl
   ```
   ```

7. Убедитесь, что установлена последняя версия:
8. `kubectl version --client`

2.2.1.2.2. Установка с помощью встроенного пакетного менеджера

- Ubuntu, Debian или HypriotOS

```
sudo apt-get update && sudo apt-get install -y apt-
transport-https
curl -s
https://packages.cloud.google.com/apt/doc/apt-key.gpg
| sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-
xenial main" | sudo tee -a
/etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubectl
```

2.2.1.2.3. Установка с помощью стороннего пакетного менеджера

- Snap
- Homebrew

Если вы используйте Ubuntu или другой Linux-дистрибутив, в котором есть пакетный менеджер `snap`, `kubectl` доступен в виде приложения snap.

```
snap install kubectl --classic
kubectl version
```

2.2.1.3. Установка kubectl в macOS

2.2.1.3.1. Установка двоичного файла kubectl с помощью curl в macOS

1. Загрузите последнюю версию:
2. `curl -LO "https://storage.googleapis.com/kubernetes-`

```
release/release/$(curl -s
https://storage.googleapis.com/kubernetes-
release/release/stable.txt)/bin/darwin/amd64/kubec
tl"
```

Чтобы загрузить определенную версию, вставьте в фрагмент команды `$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)` нужную версию.

Например, команда загрузки версии v1.25.0 для macOS будет выглядеть следующим образом:

```
curl -LO
https://storage.googleapis.com/kubernetes-
release/release/v1.25.0/bin/darwin/amd64/kubectl
```

3. Сделайте двоичный файл `kubectl` исполняемым:
4. `chmod +x ./kubectl`
5. Переместите двоичный файл в директорию из переменной окружения PATH:
6. `sudo mv ./kubectl /usr/local/bin/kubectl`
7. Убедитесь, что установлена последняя версия:
8. `kubectl version --client`

2.2.1.3.2. Установка с помощью Homebrew в macOS

Если вы используете macOS и Homebrew, то `kubectl` можно установить с помощью пакетного менеджера Homebrew.

1. Выполните команду установки:

   ```
   brew install kubectl
   ```

   Или:

   ```
   brew install kubernetes-cli
   ```

2. Убедитесь, что установлена последняя версия:

```
3. kubectl version –client
```

2.2.1.3.3. Установка с помощью Macports в macOS

Если вы используйте macOS и Macports, то `kubectl` можно установить с помощью пакетного менеджера Macports.

1. Выполните команду установки:

```
2. sudo port selfupdate
3. sudo port install kubectl
```

4. Убедитесь, что установлена последняя версия:

```
5. kubectl version --client
```

2.2.1.4. Установка kubectl в Windows

2.2.1.4.1. Установка двоичного файла kubectl с помощью curl в Windows

1. Загрузите последнюю версию v1.25.0 по этой ссылке.

   Либо, если у вас установлен `curl`, выполните команду ниже:

   ```
   curl -LO
   https://storage.googleapis.com/kubernetes-
   release/release/v1.25.0/bin/windows/amd64/kubectl.
   exe
   ```

   Последнюю стабильную версию (например, при написании скриптов) вы можете узнать из файла по ссылке `https://storage.googleapis.com/kubernetes-release/release/stable.txt`.

2. Переместите двоичный файл в директорию из переменной окружения PATH:
3. Убедитесь, что версия `kubectl` совпадает загружённой:

```
4. kubectl version --client
```

Заметка: Docker Desktop for Windows добавляет собственную версию `kubectl` в переменную окружения PATH. Если у вас установлен Docker Desktop, вам придётся поместить путь к установленному двоичному файлу перед записью, добавленной установщиком Docker Desktop, либо же удалить вовсе `kubectl`, поставляемый вместе с Docker Desktop.

2.2.1.4.2. Установка с помощью Powershell из PSGallery

Если вы работаете в Windows и используете менеджер пакетов `Powershell Gallery`, вы можете установить и обновить `kubectl` с помощью `Powershell`.

1. Выполните команды по установке (обязательно укажите `DownloadLocation`):

```
Install-Script -Name install-kubectl -Scope CurrentUser -Force
install-kubectl.ps1 [-DownloadLocation <path>]
```

   Заметка: Если вы не укажете `DownloadLocation`, то `kubectl` будет установлен во временную директорию пользователя.

   Установщик создаст `$HOME/.kube` вместе с конфигурационным файлом.

4. Убедитесь, что установлена последняя версия:

```
kubectl version --client
```

   Заметка: Обновить `kubectl` можно путём выполнения двух команд, перечисленных в шаге 1.

2.2.1.4.3. Установка в Windows с помощью Chocolatey или Scoop

Для установки `kubectl` в Windows вы можете использовать либо менеджер пакетов Chocolatey , либо установщик в командной строке Scoop.

- choco
- scoop

```
choco install kubernetes-cli
```

2. Убедитесь, что установлена последняя версия:

```
kubectl version --client
```

3. Перейдите в домашнюю директорию:

```
cd %USERPROFILE%
```

5. Создайте директорию `.kube`:

```
mkdir .kube
```

7. Перейдите в созданную только что директорию `.kube`:

```
cd .kube
```

9. Настройте `kubectl`, чтобы возможно было использовать удаленный кластер Kubernetes:

```
New-Item config -type file
```

Заметка: Отредактируйте конфигурационный файл, используя ваш любимый текстовый редактор или обычный Notepad.

2.2.1.4.4. Установка kubectl из SDK Google Cloud

Вы можете использовать `kubectl` из SDK Google Cloud, который использует этот CLI-инструмент.

1. Установите Google Cloud SDK.
2. Выполните команду для установки `kubectl`:

3. `gcloud components install kubectl`
4. Убедитесь, что установлена последняя версия:
5. `kubectl version --client`

2.2.1.5. Проверка конфигурации kubectl

Чтобы `kubectl` мог найти и получить доступ к кластеру Kubernetes, нужен файл `kubeconfig`, который создаётся автоматически при создании кластера с помощью скрипта `kube-up.sh` или при успешном развертывании кластера Minikube. По умолчанию конфигурация `kubectl` находится в `~/.kube/config`.

Посмотрите на состояние кластера, чтобы убедиться, что kubectl правильно сконфигурирован:

```
kubectl cluster-info
```

Если вы видите URL-ответ, значит `kubectl` корректно настроен для работы с вашим кластером.

Если вы видите сообщение следующего содержания, то значит `kubectl` настроен некорректно или не может подключиться к кластеру Kubernetes:

```
The connection to the server <server-name:port> was refused - did you specify the right host or port?
```

Например, если вы собираетесь запустить кластер Kubernetes на своем ноутбуке (локально), вам потребуется сначала установить специальный для этого инструмент, например Minikube, а затем снова выполнить указанные выше команды.

Если команда `kubectl cluster-info` возвращает URL-ответ, но вы не можете подключиться к своему кластеру, чтобы убедиться, что он правильно настроен, воспользуйтесь этой командой:

```
kubectl cluster-info dump
```

2.2.1.6. Дополнительная конфигурация kubectl

2.2.1.6.1. Включение автодополнения ввода shell

kubectl поддерживает автодополнение (автозаполнение) ввода в Bash и Zsh, которое сэкономит вам много времени на набор команд.

Ниже приведены инструкции по настройке автодополнения для Bash (для Linux и macOS) и Zsh.

- Bash в Linux
- Bash в macOS
- Zsh

2.2.1.6.2. Основные сведения

Скрипт дополнения ввода `kubectl` для `Bash` может быть сгенерирован с помощью команды `kubectl completion bash`. Подключение скрипта дополнения ввода в вашу оболочку включает поддержку автозаполнения ввода для `kubectl`.

Однако скрипт дополнения ввода зависит от `bash-completion`, поэтому вам нужно сначала установить этот пакет (вы можете выполнить команду `type _init_completion`, чтобы проверить, установлен ли у вас уже `bash-completion`).

2.2.1.6.3. Установка bash-completion

`bash-completion` можно установить через многие менеджеры пакеты (см. здесь). Вы можете установить его с помощью `apt-get install bash-completion` или `yum install bash-completion` и т.д.

Приведенные выше команды создадут файл `/usr/share/bash-completion/bash_completion`, который является основным

скриптом bash-completion. В зависимости от используемого менеджера пакетов, вы можете подключить этот файл в файле ~/.bashrc.

Чтобы убедиться, что этот скрипт выполняется, перезагрузите оболочку и выполните команду type _init_completion. Если команда отработала успешно, установка сделана правильно, в противном случае добавьте следующее содержимое в файл ~/.bashrc:

```
source /usr/share/bash-completion/bash_completion
```

Перезагрузите вашу оболочку и убедитесь, что bash-completion правильно установлен, напечатав в терминале type _init_completion.

2.2.1.6.4. Включение автодополнения ввода kubectl

Теперь нужно убедиться, что скрипт дополнения ввода kubectl выполняется во всех сессиях командной оболочки. Есть два способа сделать это:

- Добавьте запуск скрипта дополнения ввода в файл ~/.bashrc:
- `echo 'source <(kubectl completion bash)' >>~/.bashrc`
- Добавьте скрипт дополнения ввода в директорию /etc/bash_completion.d:
- `kubectl completion bash >/etc/bash_completion.d/kubectl`
- Если у вас определён псевдоним для kubectl, вы можете интегрировать его с автодополнением оболочки:
- `echo 'alias k=kubectl' >>~/.bashrc`
- `echo 'complete -F __start_kubectl k' >>~/.bashrc`

Заметка: Все скрипты дополнения ввода bash-completion находятся в `/etc/bash_completion.d`.

Оба подхода эквивалентны. После перезагрузки вашей оболочки, должны появляться дополнения ввода `kubectl`.

2.2.2. Установка Helm

В этом руководстве показано, как установить Helm CLI. Helm можно установить либо из исходного кода, либо из бинарных файлов.

2.2.2.1. Из бинарных файлов

Каждый релиз Helm содержит бинарные версии для разных операционных систем. Эти двоичные версии можно загрузить и установить вручную.

1. Загрузите нужную версию
2. Распакуйте ее (`tar -zxvf helm-v3.0.0-linux-amd64.tar.gz`)
3. Найдите бинарный файл `helm` в распакованной директории и переместите его в нужное место (`mv linux-amd64/helm /usr/local/bin/helm`)

Оттуда вы сможете запустить клиент и добавить стабильный репозиторий: `helm help`.

2.2.2.2. Из сценария установки

У Helm есть сценарий установки, который автоматически загружает последнюю версию Helm и устанавливает ее локально.

```
$ curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main
/scripts/get-helm-3
```

```
$ chmod 700 get_helm.sh
$ ./get_helm.sh
```

2.2.2.3. Из исходного кода (Linux, macOS)

Сборка Helm из исходного кода немного сложнее, но это лучший способ, если вы хотите протестировать последнюю версию Helm.

У вас должна быть рабочая среда Go.

```
$ git clone https://github.com/helm/helm.git
$ cd helm
$ make
```

При необходимости он извлечет зависимости и кэширует их, а также проверит конфигурацию. Затем он скомпилирует helm и поместит его в `bin/helm`.

2.2.3. Организация доступа к кластеру с помощью файлов kubeconfig

Используйте файлы `kubeconfig` для организации информации о кластерах, пользователях, пространствах имен и механизмах аутентификации. Инструмент командной строки `kubectl` использует файлы `kubeconfig` для поиска информации, необходимой для выбора кластера и связи с сервером API кластера.

Предупреждение. Используйте файлы `kubeconfig` только из надежных источников. Использование специально созданного файла `kubeconfig` может привести к выполнению вредоносного кода или раскрытию файла. Если вам необходимо использовать ненадежный файл `kubeconfig`, сначала внимательно проверьте его.

По умолчанию `kubectl` ищет файл с именем `config` в каталоге `$HOME/.kube`. Вы можете указать другие файлы `kubeconfig`, установив переменную среды `KUBECONFIG` или установив флаг `–kubeconfig`.

2.2.3.1. Поддержка нескольких кластеров, пользователей и механизмов аутентификации

Предположим, у вас есть несколько кластеров, и ваши пользователи и компоненты аутентифицируются различными способами. Например:

- Работающий `kubelet` может аутентифицироваться с использованием сертификатов.
- Пользователь может аутентифицироваться с помощью токенов.
- У администраторов могут быть наборы сертификатов, которые они предоставляют отдельным пользователям.

С файлами `kubeconfig` вы можете организовать свои кластеры, пользователей и пространства имен. Вы также можете определить контексты, чтобы быстро и легко переключаться между кластерами и пространствами имен.

2.2.3.2. Элемент context

Элемент `context` в файле `kubeconfig` используется для группировки параметров доступа под удобным именем. Каждый контекст имеет три параметра: кластер, пространство имен и пользователь. По умолчанию инструмент командной строки `kubectl` использует параметры из текущего контекста для связи с кластером.

Чтобы выбрать текущий контекст:

```
kubectl config use-context
```

2.2.3.3. Переменная среды KUBECONFIG

Переменная среды `KUBECONFIG` содержит список файлов `kubeconfig`. Для Linux и MacOS список разделен двоеточиями. Для Windows список разделен точкой с запятой. Переменная среды `KUBECONFIG` не требуется. Если переменная среды `KUBECONFIG` не существует, `kubectl` использует файл `kubeconfig` по умолчанию, `$HOME/.kube/config`.

Если переменная среды `KUBECONFIG` существует, `kubectl` использует конфигурацию, которая является результатом слияния файлов, перечисленных в переменной среды `KUBECONFIG`.

2.2.3.4. Ссылки на файлы

Ссылки на файлы и пути в файле `kubeconfig` относятся к расположению файла `kubeconfig`. Ссылки на файлы в командной строке относятся к текущему рабочему каталогу `$HOME/.kube/config`.

2.2.3.5. Прокси

Вы можете настроить `kubectl` на использование прокси-сервера для каждого кластера, используя `proxy-url` в файле `kubeconfig`, например:

```
apiVersion: v1
kind: Config

clusters:
- cluster:
    proxy-url: http://proxy.example.org:3128
    server: https://k8s.example.org/k8s/clusters/c-
xxyyzz
```

```
  name: development

users:
- name: developer

contexts:
- context:
  name: development
```

2.3. Установка и настройка Postgres

Для создания базы данных Postgres можно также воспользоваться облачными сервисами Yandex Cloud Managed PostgreSQL, VK Cloud PostgreSQL DBaaS.

Поддерживаются версии Postgres 12 и выше.

После создания сервера PostgreSQL необходимо создать базу данных и выделенного пользователя для доступа к базе. Пользователь должен быть владельцем (Owner) базы. Данный пользователь будет использоваться приложением для доступа к базе.

Для деплоя PostgreSQL в кластер Kubernetes можно воспользоваться инструкцией ниже.

После создания базы данных PostgreSQL, необходимо создать Kubernetes-секрет, содержащий пользователя и пароль для подключения. Пример манифеста:

```
apiVersion: v1
kind: Secret
metadata:
  name: postgres-db-secret
data:
  username: base64encoded-username
  password: base64encoded-password
```

Манифест применяется командой:

```
kubectl apply -f my-manifest.yaml
```

2.3.1. Развертывание PostgreSQL с помощью Helm

Helm дает вам быстрый и простой способ развернуть экземпляр PostgreSQL в вашем кластере.

Шаг 1. Добавьте репозиторий Helm

1. Найдите в Artifact Hub диаграмму PostgreSQL Helm, которую хотите использовать. Добавьте репозиторий диаграммы в локальную установку Helm, набрав:

```
helm repo add [имя-репозитория] [адрес-репозитория]
```

Добавление репозитория Bitnami helm, содержащего диаграмму PostgreSQL

2. После добавления репозитория обновите локальные репозитории.

```
helm repo update
```

Система подтверждает успешное обновление.

Шаг 2. Создайте и примените том постоянного хранилища

Данные в вашей базе данных Postgres должны сохраняться при перезапуске модуля.

1. Для этого создайте ресурс PersistentVolume в файле YAML с помощью текстового редактора, такого как `nano`.

```
nano postgres-pv.yaml
```

Содержимое файла определяет:

Сам ресурс.

Класс хранения.

Объем выделенного хранилища.

Режимы доступа.

Путь монтирования в хост-системе.

В этом примере используется следующая конфигурация:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: postgresql-pv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

2. Сохраните файл и выйдите. Затем примените конфигурацию с помощью `kubectl`:

```
kubectl apply -f postgres-pv.yaml
```

Шаг 3. Создайте и примените постоянную заявку на объем

1. Создайте Persistent Volume Claim (PVC), чтобы запросить хранилище, выделенное на предыдущем шаге.

```
nano postgres-pvc.yaml
```

В примере используется следующая конфигурация:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```

```
    name: postgresql-pv-claim
spec:
    storageClassName: manual
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 10Gi
```

2. Сохраните файл и выйдите. Примените конфигурацию с помощью `kubectl`:

```
kubectl apply -f postgres-pvc.yaml
```

3. Используйте `kubectl get`, чтобы проверить, успешно ли PVC подключен к PV:

```
kubectl get pvc
```

Шаг 4: Установите Helm Chart

Установите helm chart с помощью команды `helm install`. Добавьте флаги `--set` в команду, чтобы подключить установку к созданному вами PVC и включить разрешения тома:

```
helm install [имя-релиза] [имя-репо] --set
persistence.existingClaim=[имя-pvc] --set
volumePermissions.enabled=true
```

Шаг 5. Подключитесь к клиенту PostgreSQL

1. Экспортируйте переменную среды `POSTGRES_PASSWORD`, чтобы иметь возможность войти в экземпляр PostgreSQL:

```
export POSTGRES_PASSWORD=$(kubectl get secret --
namespace default psql-test-postgresql -o
jsonpath="{.data.postgresql-password}" | base64 --
decode)
```

2. Откройте другое окно терминала и введите следующую команду перенаправления портов, чтобы перенаправить порт Postgres:

```
kubectl port-forward --namespace default svc/psql-
test-postgresql 5432:5432
```

3. Сверните окно переадресации портов и вернитесь к предыдущему. Введите команду для подключения к `psql`, клиенту PostgreSQL:

```
PGPASSWORD="$POSTGRES_PASSWORD" psql --host
127.0.0.1 -U postgres -d postgres -p 5432
```

3. Деплой приложения MCP

3.1. Копирование образов микросервисов

Примечание. Данный пункт необходимо выполнять только в том случае, если вы используете свой Container Registry для деплоя в кластер Kubernetes.

Вам необходимо скопировать образы в свой Container Registry с помощью утилиты `crane` или `docker`.

Пример:

```
crane cp registry.mcp.btn.bz/mcp/backend/ssa-service:1.0.0 my.registry.com/ssa-service:1.0.0
```

Полный список образов, которые нужно скопировать:

- `registry.mcp.btn.bz/mcp/admin-react`
- `registry.mcp.btn.bz/mcp/backend/arm-admin-service`
- `registry.mcp.btn.bz/mcp/backend/associate-rule-service`
- `registry.mcp.btn.bz/mcp/backend/async-task-service`
- `registry.mcp.btn.bz/mcp/backend/auth-service`
- `registry.mcp.btn.bz/mcp/backend/client-service`
- `registry.mcp.btn.bz/mcp/backend/cluster-service`
- `registry.mcp.btn.bz/mcp/backend/data-service`
- `registry.mcp.btn.bz/mcp/backend/product-service`
- `registry.mcp.btn.bz/mcp/backend/rating-service`
- `registry.mcp.btn.bz/mcp/backend/scheduler-service`
- `registry.mcp.btn.bz/mcp/backend/shelf-service`
- `registry.mcp.btn.bz/mcp/backend/ssa-service`
- `registry.mcp.btn.bz/mcp/backend/tg-bot-service`
- `registry.mcp.btn.bz/mcp/backend/web-api-service`

3.2. Копирование helm-чарта

Для деплоя приложения вам понадобится наш helm-чарт. Скачать его можно с помощью команды:

```
helm pull oci://registry.mcp.btn.bz/charts/mcp --version 1.0.0
```

В результате в текущем рабочем каталоге у вас должен появиться файл `mcp-1.0.0.tgz`

3.3. Создание DNS-записей для приложения

DNS-записи должны указывать на IP-адрес Ingress-контроллера вашего Kubernetes-кластера.

Пример:

```
admin.mcp.my-domain.com A 1.2.3.4
api.mcp.my-domain.com A 1.2.3.4
```

3.4. Подготовка Values для helm чарта

Сохраняем дефолтные Values в файл:

```
helm show values ./mcp-1.0.0.tgz > mcp-values.yaml
```

Вносим изменения в файл mcp-values.yaml:

```
global:
 env:
   - name: PGHOST
     value: "postgresql-server"
   - name: PGPORT
     value: "5432"
   - name: PGDATABASE
     value: "mcp"
   - name: PGUSER
     valueFrom:
       secretKeyRef:
         name: postgres-db-secret
         key: username
   - name: PGPASS
     valueFrom:
       secretKeyRef:
         name: postgres-db-secret
         key: password
```

Здесь нам потребуется указать сервер БД, порт, имя базы и название Kubernetes-секрета, содержащего пользователя и пароль от базы.

Также, если вы используете свой Registry (п. 3.1), то необходимо указать правильное имя образа для каждого сервиса. Пример для сервиса `admin-react`:

```
admin-react:
 enabled: true
 image:
   repository: my.registry.com/admin-react
   tag: "1.0.0"
```

Также необходимо включить и заполнить настройки Ingress.

Для web-api-service:

```
web-api-service:
  ...
  ingress:
   enabled: true
   ...
   hosts:
     - host: api.mcp.my-domain.com
       paths:
         - path: /
           pathType: ImplementationSpecific
```

Для admin фронтенда:

```
admin-react:
 env:
   # set api gateway endpoint for frontend
   - name: REACT_APP_SERVER_ENDPOINT
     value: "http://api.mcp.my-domain.com"
 ingress:
   enabled: true
```

```
...
hosts:
  - host: admin.mcp.my-domain.com
    paths:
      - path: /
        pathType: ImplementationSpecific
```

3.5. Деплой приложения в кластер Kubernetes

Для деплоя приложения выполняем следующую команду в консоли:

```
helm install mcp ./mcp-1.0.0.tgz -atomic -debug -
wait -values mcp-values.yaml
```

Если команда завершилась без ошибок, то значит деплой выполнился успешно. Фронтенд приложения будет доступен по ссылке:

```
http://admin.mcp.my-domain.com
```