Инструкция по установке и настройке компонентов

DevKube

DevKube – платформа для управления жизненным циклом контейнерных приложений

ООО " Аплана. Центр Разработки ", г. Москва

2025 г.

Содержание

Введение....3
Планирование....4
Развёртывание кластера....5
  Подготовка....5
  Создание....5
Операторы....8
Локальное хранилище....9
Проекты....10
  Создание проекта....10
  Создание пространств имён....11
Домены....13
Установка....14
  GitLab....15
  Harbor....17
  Jenkins....17
  Nexus....18
  SonarQube....18
    Подготовка....19
    Установка....22
Настройка интеграции....23
  Распределение по имени....23
  Распределение публичных проектов....23
  Публичный инструмент....24

Введение

После инициализации кластера администратор уже имеет функционал для управления и мониторинга кластеров.

В качестве дополнительных функций, администратор может установить в платформу такие компоненты как:

- Gitlab;
- Jenkins;
- Nexus;
- Harbor;
- SonarQube.

Перед тем, как устанавливать данные компоненты, необходимо выполнить ряд подготовительных мероприятий.

Планирование

Все компоненты будут установлены в виде готовых приложений. В связи с этим, их установку рекомендуется выполнять на выделенный кластер.

Минимальные системные требования для такого кластера будут следующие:

Имя объекта	*ЦП*	*ОЗУ*	*Диск*
Infra-master-1	8	16	150Gb SSD
Infra-master-2	8	16	150Gb SSD
Infra-master-3	8	16	150Gb SSD
Infra-worker-1	16	32	150Gb SSD + 150Gb SSD
Infra-worker-2	16	32	150Gb SSD + 150Gb SSD
Infra-worker-3	16	32	150Gb SSD + 150Gb SSD
Итого	72	144	1.45Tb SSD

Если выделение кластера для инфраструктуры не планируется, данный сайзинг необходимо учесть при увеличении мощностей global кластера.

Развёртывание кластера

Подготовка

В инсталляционном пакете присутствует скрипт с именем **init.sh**. Его необходимо перенести на каждую машину планируемого кластера. Для этого, на машине с инсталляционным пакетом переходим в директорию, в которой он установлен и выполняем команду

```
scp init.sh <username>@<host_ip>:<folder>
```

где.

- <username> - имя пользователя удалённой машины;
- <host_ip> - IP-адрес удалённой машины;
- <folder> - директория, в которую будет помещён скрипт.

После этого, запускаем выполнение данного скрипта на каждой машине

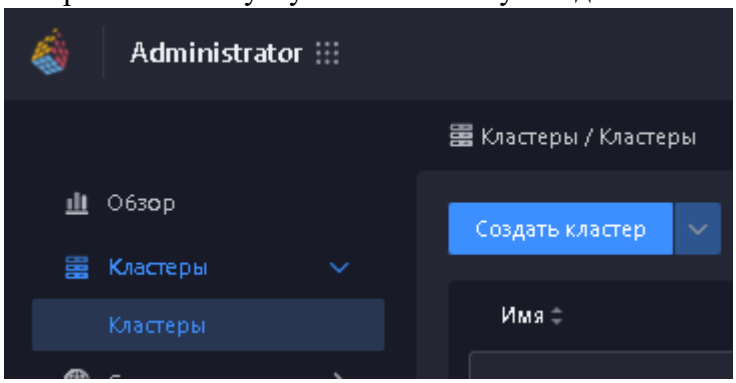
```
bash <folder>/init.sh
```

Скрипт выполнит первоначальную настойку машины, после чего её можно будет включить в кластер.

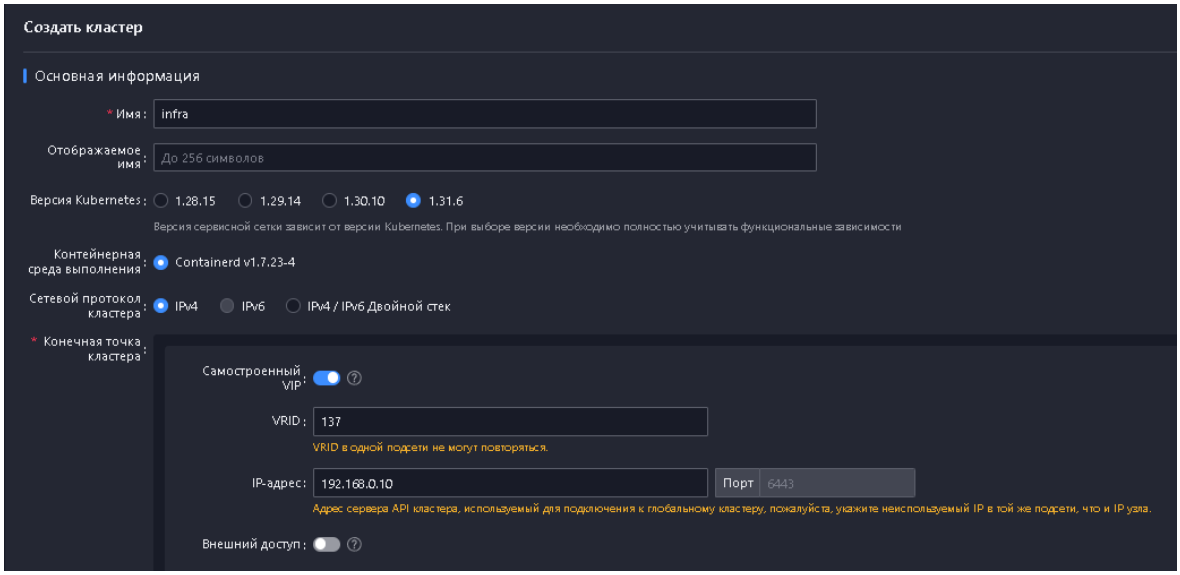
Создание

Далее, переходим в WEB-интерфейс платформы DevKube. Добавлять или изменять кластеры может только пользователь с ролью администратора платформы. Соответственно, нам необходимо авторизоваться под пользователем, обладающим данными правами. Для того чтобы создать кластер, необходимо:

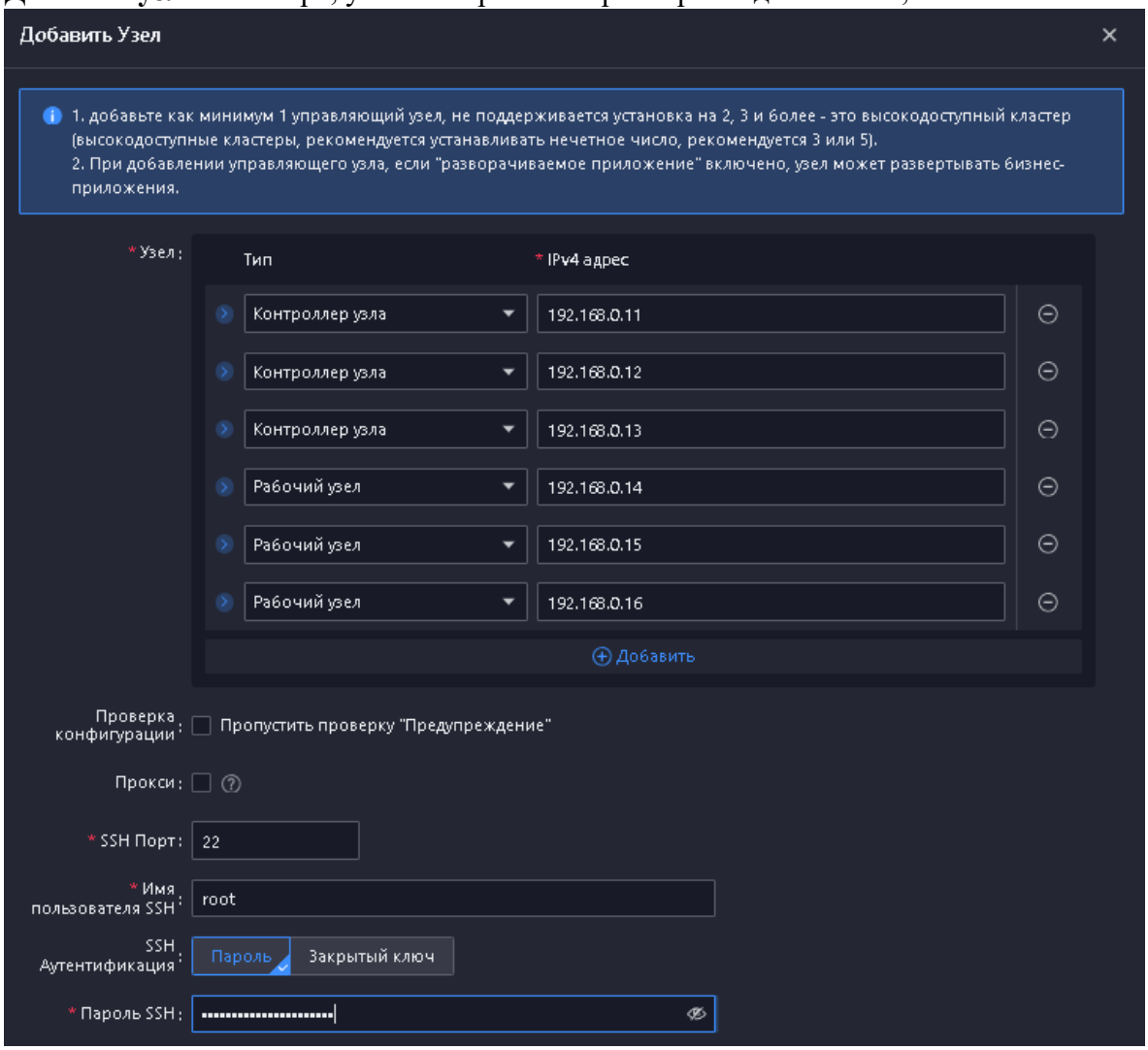
1. Перейти в раздел **Administrator**;
2. Открыть вкладку **Кластеры** и перейти в соответствующий раздел;
3. В верхнем левом углу нажать кнопку **Создать кластер;**



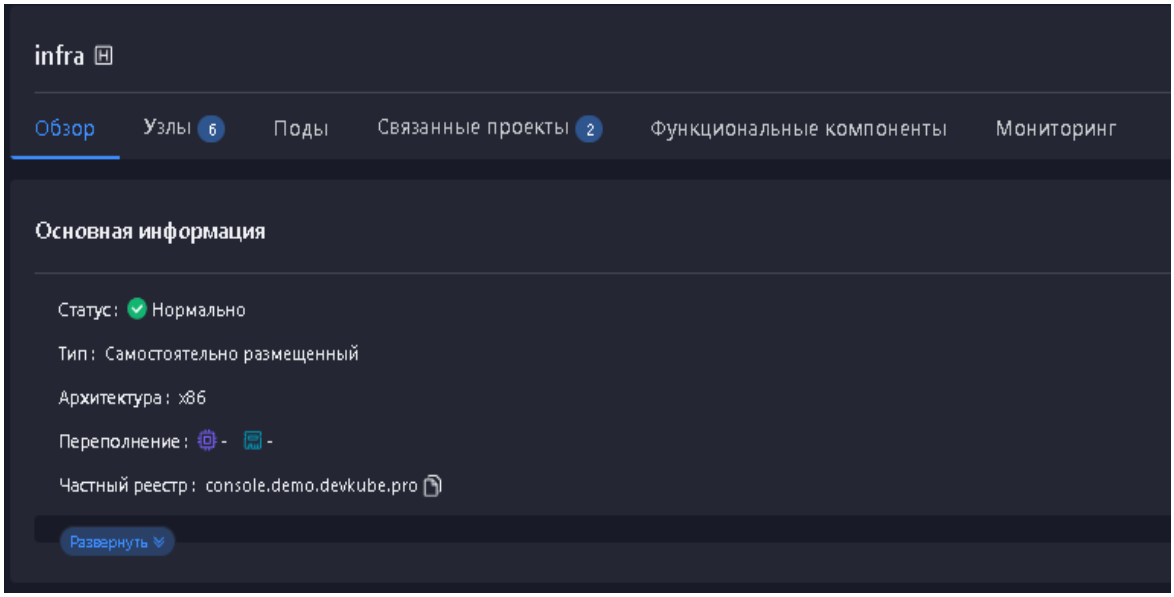
4. В появившейся форме указать **имя** нового кластера;
5. Указать **конечную точку кластера**(Виртуальный IP-адрес для балансировщика нагрузки);



6. Добавить **узлы** кластера, указав их роль и параметры подключения;



7. Нажать кнопку **Создать**;
8. Дождаться перехода кластера в состояние **Нормально**.


infra
Обзор
Узлы 6
Поды
Связанные проекты 2
Функциональные компоненты
Мониторинг
Основная информация
Статус: Нормально
Тип: Самостоятельно размещенный
Архитектура: x86
Переполнение: - -
Частный реестр: console.demo.devkube.pro
Развернуть

Операторы

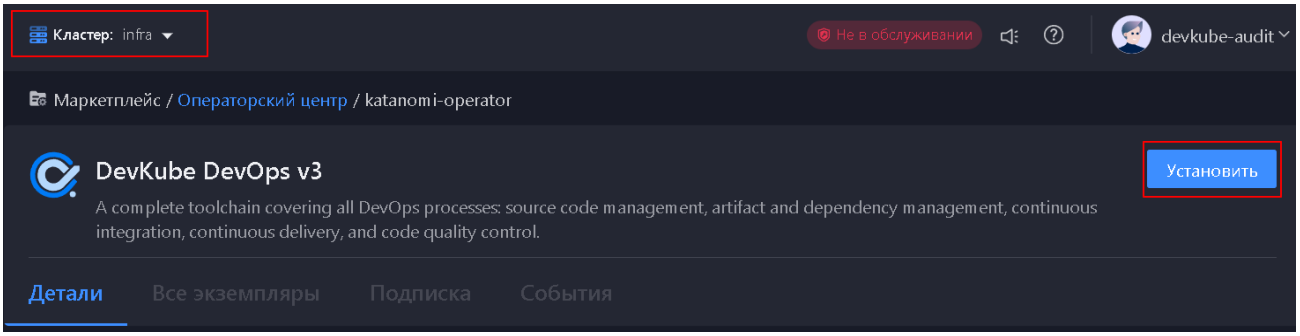
Следующим шагом перед развёртыванием компонентов – установка операторов.

Операторы – приложения, помогающие управлять, отслеживать и получать статус работы приложений. Именно с помощью операторов происходит развёртывание компонентов.

Все операторы находятся во вкладке **Маркетплейс -> Операторский центр**. Именно здесь производится их установка.

Важно заметить. что все компоненты платформы устанавливаются на определённый кластер. Поэтому, перед тем как установить оператор, убедитесь, что кластер в левом верхнем углу выбран верно. Именно на этот кластер в дальнейшем будут установлены приложения.

Для того, чтобы установить оператор достаточно нажать на него и кликнуть кнопку **Установить** в верхнем правом углу экрана.



Неймспейс для развёртывания оператора можно оставить по-умолчанию. Важно чтобы каждый оператор имел свой неймспейс. Это необходимо для успешного проведения дальнейших обновлений.

Локальное хранилище

Каждое приложение требует постоянного хранилища для данных. Поэтому, после установки операторов необходимо создать **локальное хранилище**. В роли такого хранилища будет выступать TopoLVM кластер. По аналоги с операторами, важно, чтобы данный кластер был развёрнут на том же кластере.

Для того чтобы развернуть TopoLVM кластер, необходимо:

1. Перейти во вкладку **Хранилище – Локальное хранилище**;
2. Нажать кнопку **Создать кластер**;
3. Выбрать **узлы хранения***;
4. Добавить присоединённые **диски**;
5. Задать **имя** для StorageClass;
6. Дождаться инициализации кластера и получения статуса дисков.

*На этапе проектирования кластера, были подключены дополнительные диски к машинам с ролью worker. А значит, из списка хостов кластера необходимо выбрать именно worker. Данная схема обязательна. В случае, если узлами хранения будут выступать хосты с ролью master, система не сможет использовать данные диски, т.к. Kubernetes не сможет запланировать на них поды.

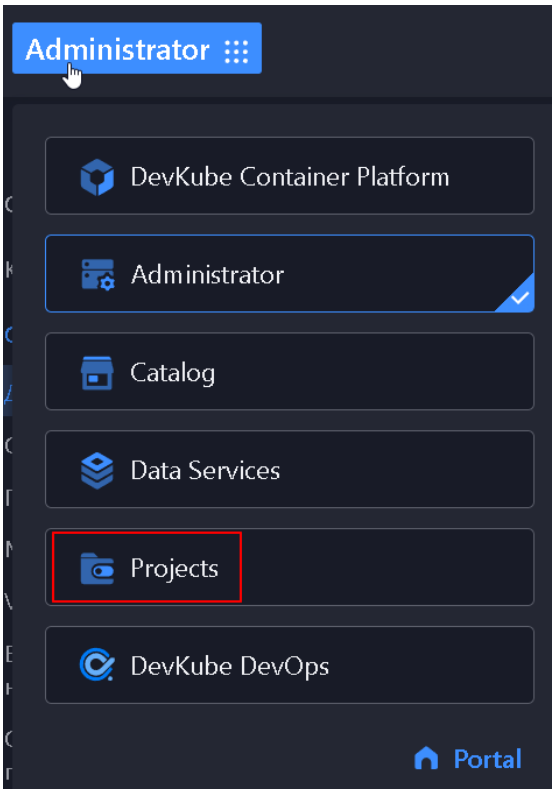
Проекты

Создание проекта

Перед тем как развернуть приложения, необходимо создать проект, в котором данные приложения будут разворачиваться. У каждого проекта есть свои неймспейсы для развёртывания подов. Рекомендуется, для каждого приложения создать свой неймспейс для того, чтобы избежать возможных ошибок.

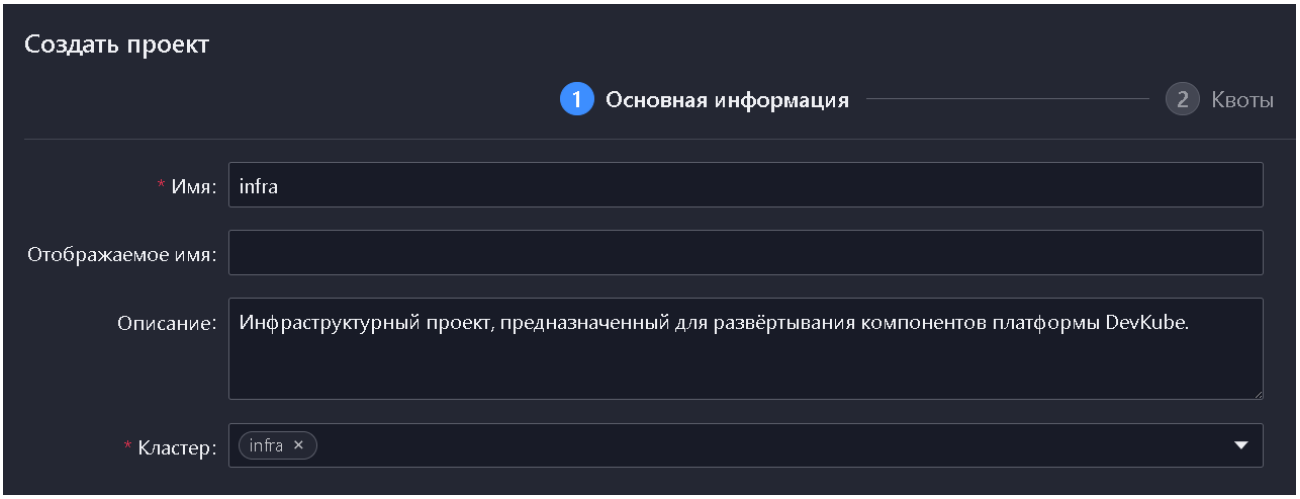
Выполнять создание неймпейсов может пользователь с ролью **администратора проекта**.

Для примера, создадим проект infra. Для этого, нужно открыть верхнее левое меню и перейти в раздел **Projets**.



На данной странице будет представлен список всех существующих в платформе проектов, их лимиты, статус и время создания. Для того, чтобы создать новый проект, необходимо нажать кнопку **Создать проект**.

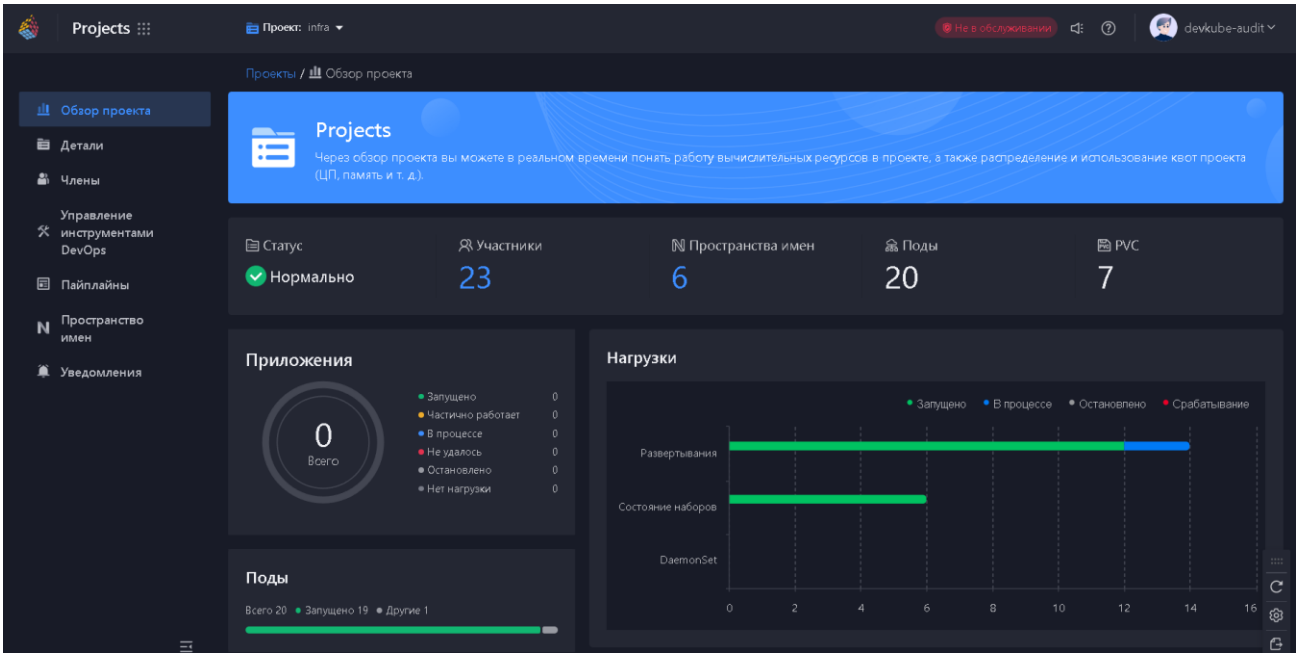
В открывшемся меню необходимо задать имя проекта, выбрать кластер, в котором он будет развёртываться, а также, по желанию, ввести краткое описание проекта.



Переходим далее и попадаем в окно настройки ограничений проекта. Для инфраструктурных проектов лимиты создавать не обязательно.

Создание пространств имён

После создания проекта и перехода в него у нас открывается панель мониторинга проекта. Здесь будет отображаться информация о приложениях проекта, а также количестве пространств имён(неймспейсов) данного проекта.

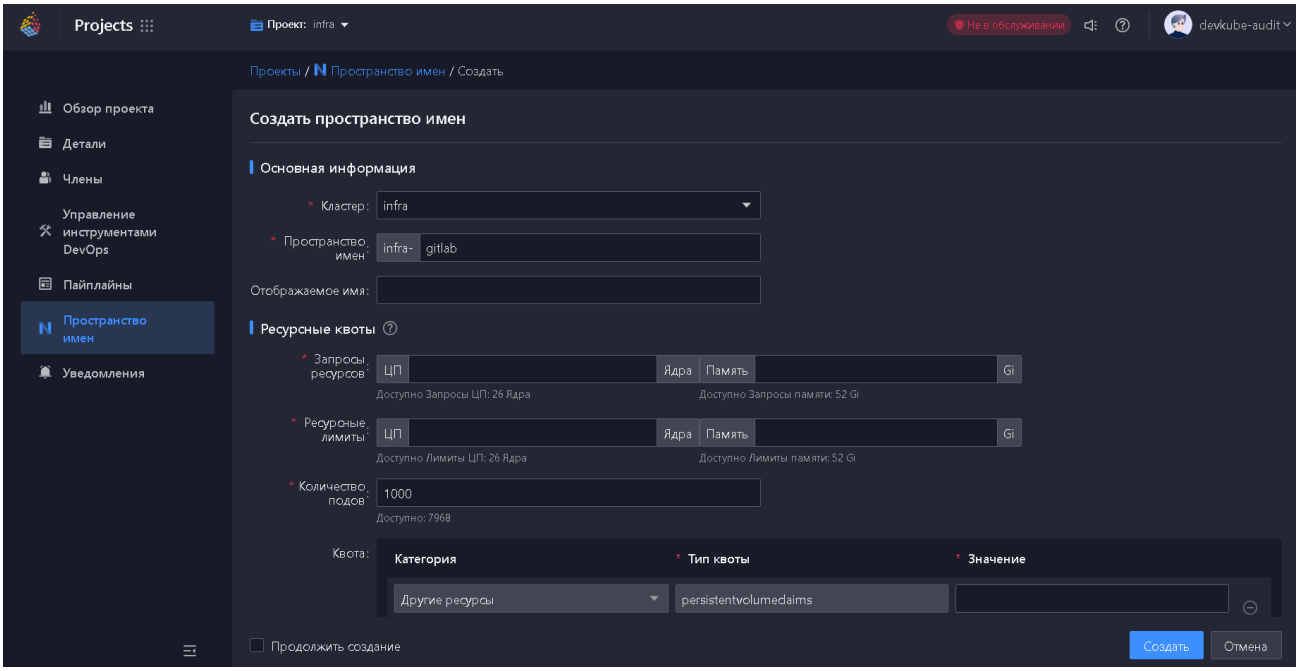


Для создания неймспейса необходимо перейти в раздел **Пространство имён**.

В верхнем левом углу открывшегося окна нажимаем кнопку **Создать пространство имён**.

Настройка немспейса выглядит следующим образом:

1. Вводим **имя** немспейса;
2. По желанию, вводим **лимиты** для неймпейса;
3. Если проект использует несколько кластеров, то выбираем целевой **кластер**.

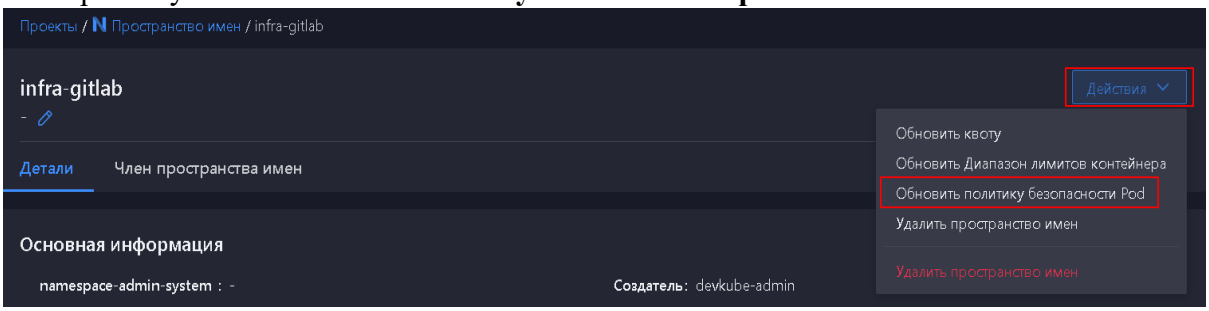


Повторяем процедуру создания неймспейса для всех компонентов платформы.

Перед тем как начать развёртывание компонентов, необходимо подготовить пространства имён, в которых они будут развёрнуты.

Для этого:

1. Переходи в панель управления **проектами**;
2. Выбираем нужный **проект**;
3. Открываем раздел **пространств имён**;
4. Выбираем нужный **неймспейс** и открываем его;
5. В верхнем правом углу нажимаем на кнопку **Действия**;
6. Выбираем пункт **Обновить политику безопасности pod**.



Инфраструктурные приложения требуют запуск подов в привилегированном режиме. Поэтому необходимо изменить политику безопасности подов на **privileged**.

Домены

Для удобства доступа к инфраструктуре, необходимо настроить домены, по которым будет осуществляться доступ инфраструктуре. Разумеется, без добавления ресурсных записей на DNS-сервер в данном случае не обойтись, поэтому первоначально необходимо запланировать и сконфигурировать зону.

Для примера, рассмотрим официальный демонстрационный стенд платформы DevKube, console.demo.devkube.pro.

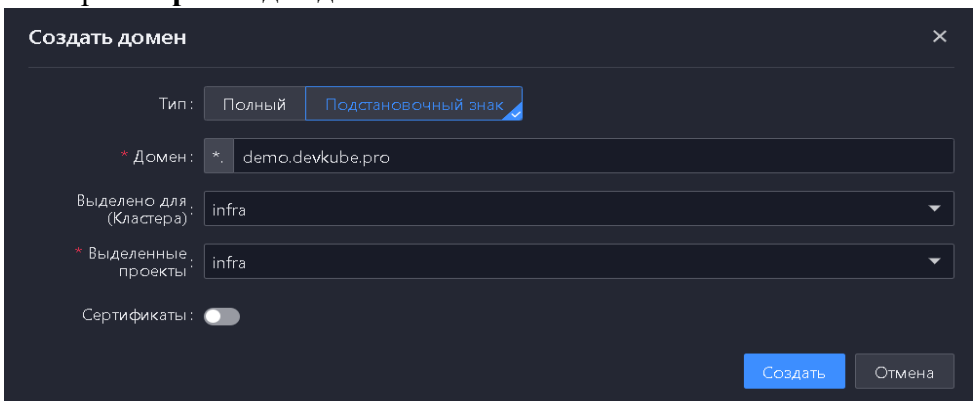
На DNS-сервере настроена зона demo.devkube.pro. При планировании инфраструктуры, были внесены записи для компонентов платформы:

Название	*Тип записи*	*Описание*
console	A	Главная страница портала DevKube
gitlab	A	Платформенный Gitlab
jenkins	A	Платформенный Jenkins
sonarqube	A	Платформенный SonarQube
harbor	A	Платформенный Harbor
nexus	A	Платформенный Nexus

Для того, чтобы сервисы могли отвечать по этим адресам, необходимо добавить WildCard-домен в платформу.

Для этого:

1. Возвращаемся в раздел **Administrator**;
2. Переходим во вкладку **Сеть - Домены**;
3. Нажимаем кнопку **Создать домен**;
4. Выбираем тип **Подстановочный знак**;
5. Вводим **имя** нашего домена;
6. Выбираем **кластер,** на который данный домен будет распространяться;
7. Выбираем **проект** для домена.



8. По желанию, можно добавить **сертификат** для выпускаемого домена;
9. Нажать кнопку **Создать**.

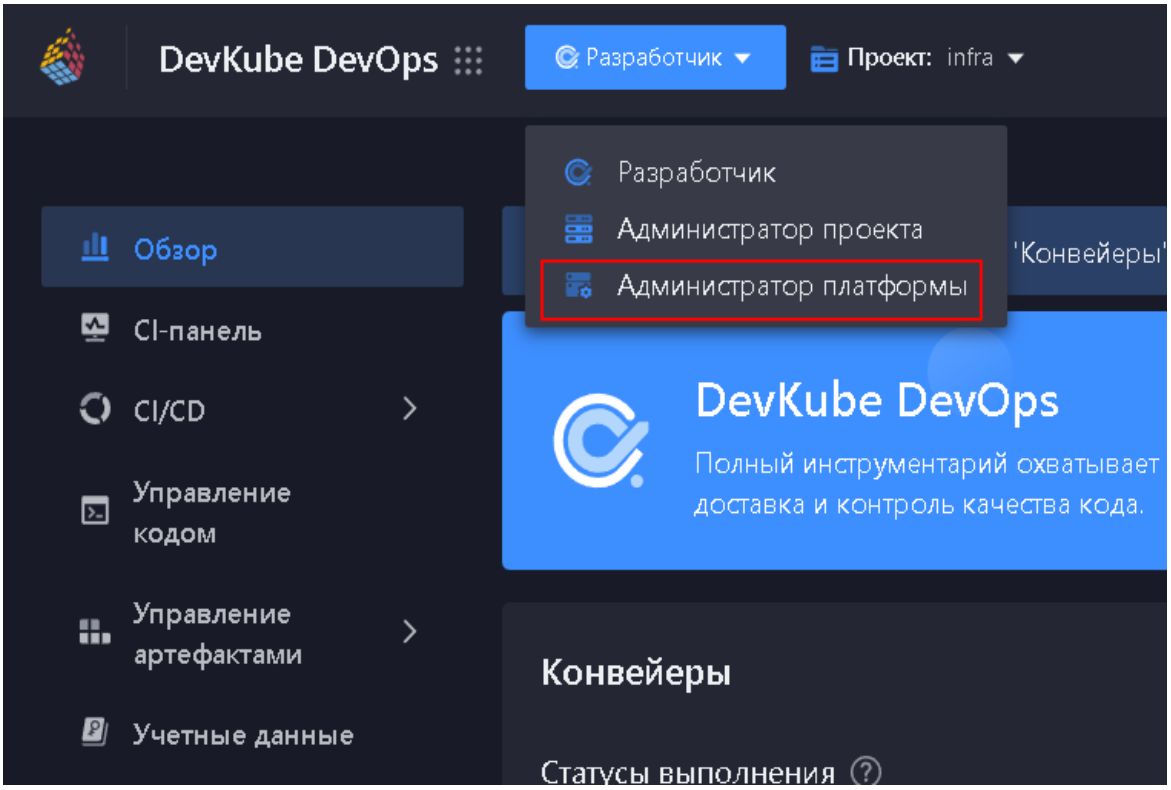
Установка

Установку компонентов платформы необходимо выполнять только после того, как выполнены все предыдущие шаги. Невыполнение одного из шагов влечёт за собой невозможность установки приложений.

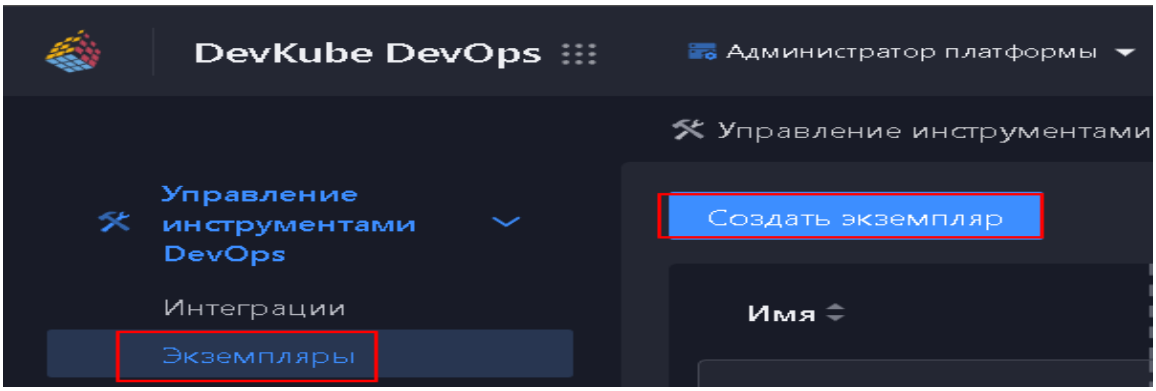
Установка компонентов платформы осуществляется пользователем с ролью **администратора платформы**.

Для того, чтобы создать экземпляр приложения, необходимо:

1. Открыть главное меню платформы и перейти в раздел DevKube DevOps;
2. В верхнем левом углу выбрать **Администратор платформы**;



3. Перейти в раздел **экземпляры**;
4. Нажать кнопку **Создать экземпляр**(Важно обратить внимание на кластер для развёртывания).



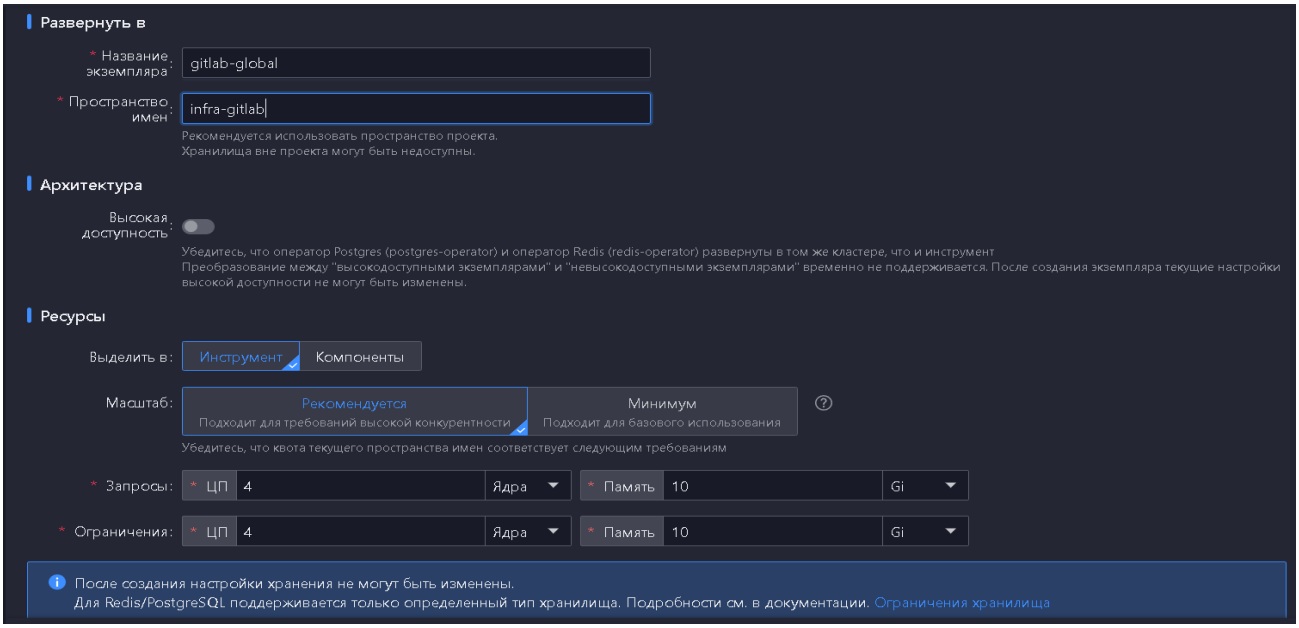
В открывшемся окне выбираем компонент для развёртывания.

Некоторые компоненты требуют дополнительных настроек. Далее, рассмотрим установку каждого компонента подробнее.

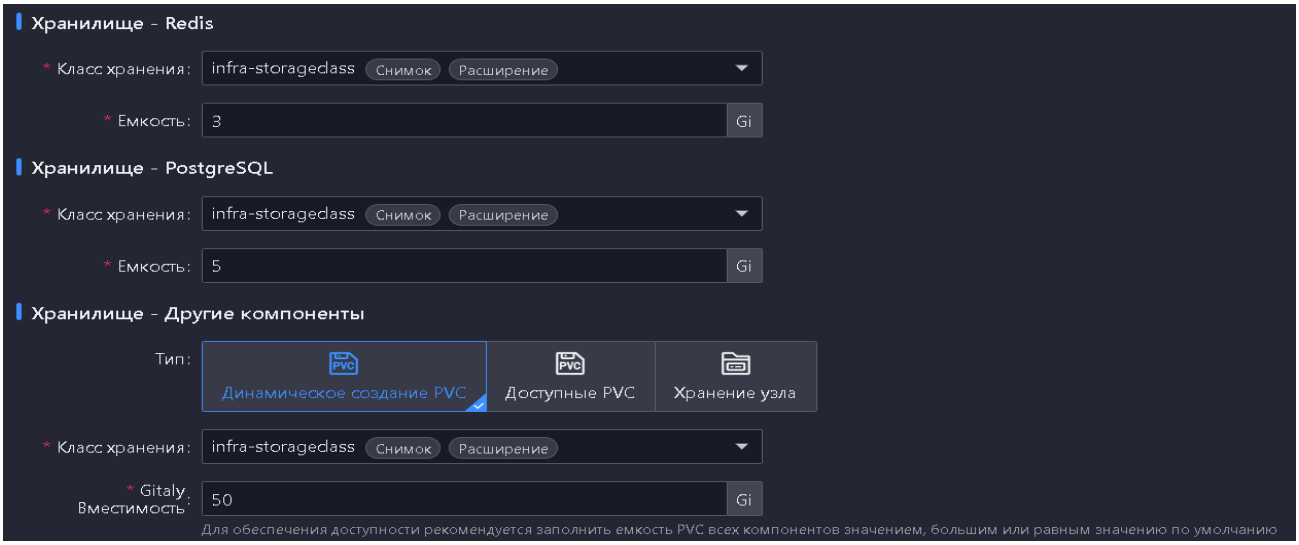
GitLab

Gitlab для своей работы требует установки Redis и PostgreSQL. Инструменты платформы позволяют сделать это прямо из Web-интерфейса без необходимости ручного конфигурирования yaml-файлов и создания манифестов Kubernetes.

При выборе создания экземпляра GitLab перед нам появляется окно развёртывания. Для его корректной работы рекомендуется указать минимум 4 ядра и 10 гб оперативной памяти.

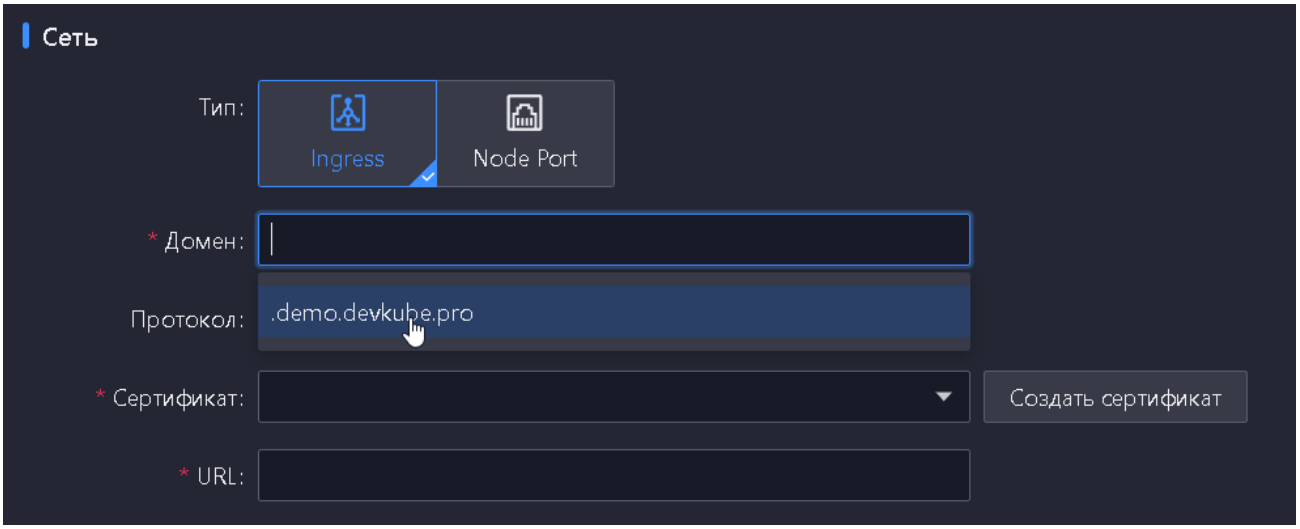


Далее, нам необходимо сконфигурировать хранилище нашего экземпляра. Рекомендуем перед тем, как разворачивать приложения, рассчитать хранилище под каждый сервис с учётом лимита текущего хранилища.



После настройки хранения данных. Перейдём к ингрессу.

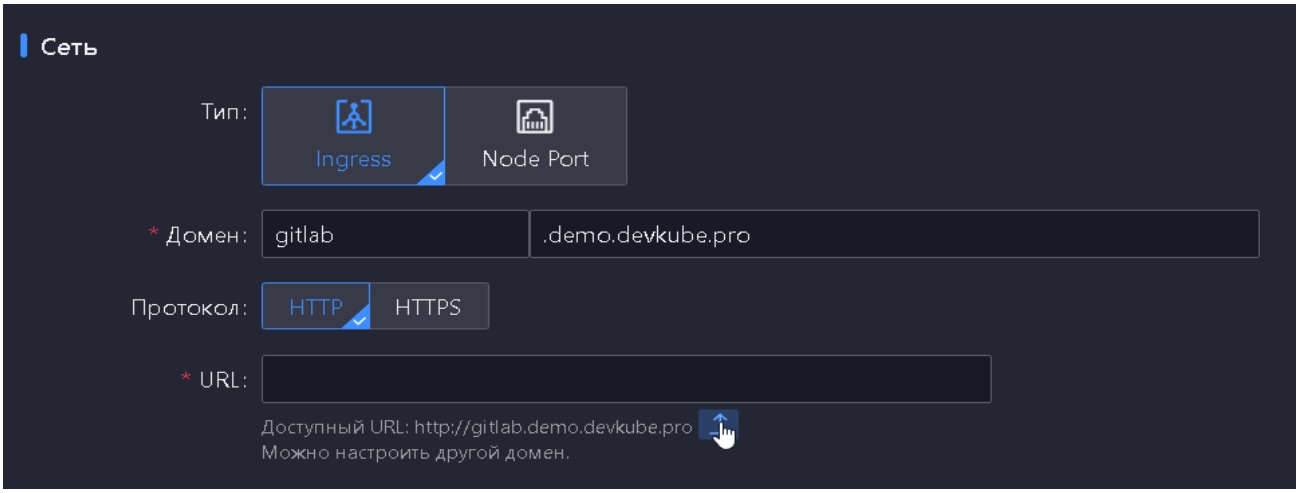
Кликнув по полю ввода домена, система предложит нам ранее введённый нами wildcard-домен.



После нажатия на него. Домен подставится в поля для ввода, а нам останется лишь ввести поддомен. После чего, выбираем протокол, по которому будет генерироваться ссылка на наш экземпляр. При выборе HTTPS появится поле сертификата.

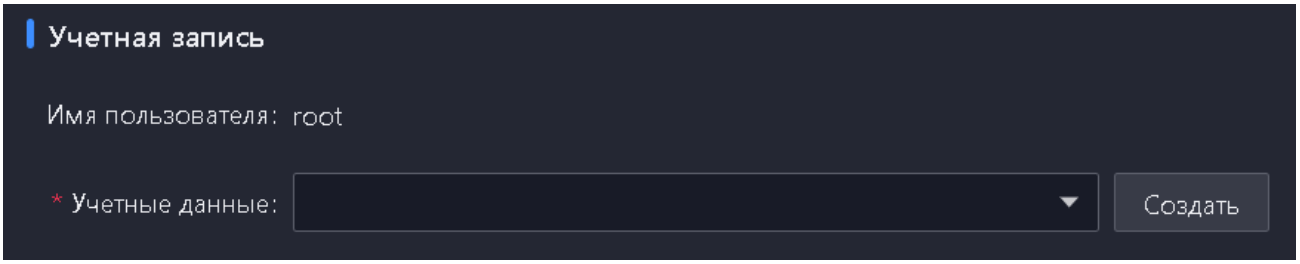
В поле сертификата, мы можем создать сертификат для нашего домена. Стоит отметить, что данная функция не генерирует ssl-сертификат для домена, а позволяет импортировать в платформу уже имеющийся сертификат.

Далее вводим URL нашего экземпляра или подставляем его автоматически, нажав на стрелку внизу.

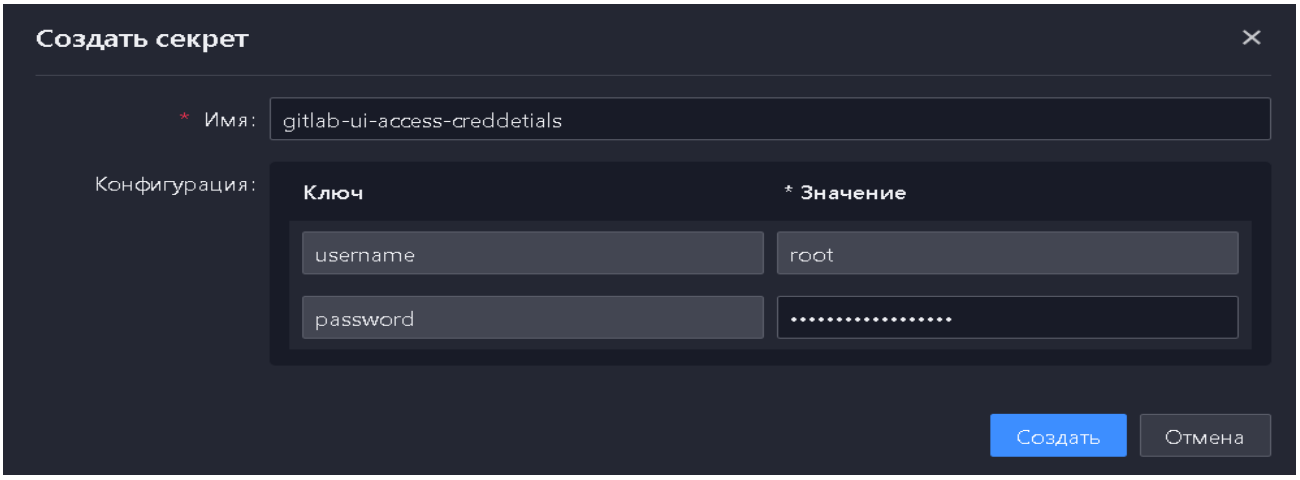


Следующим шагом, создаём креды для доступа администратора экземпляра.

Возле поля выбора учётных данных необходимо нажать кнопку **Создать**.



Вводим имя секрета, вводим пароль пользователя и сохраняем секрет.



После чего, нажимаем кнопку **Создать** в окне создания экземпляра.

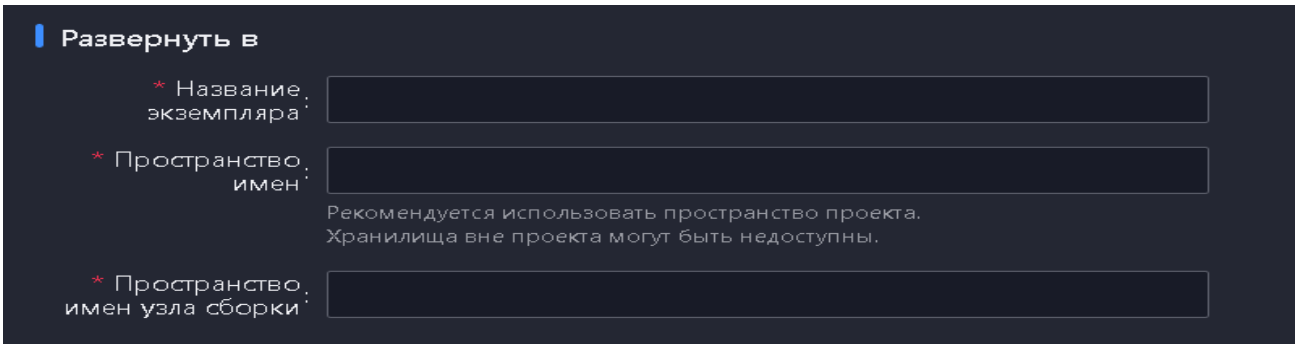
По завершению выполнения, будет создан экземпляр приложения GitLab внутри платформы DevKube.

Harbor

Процесс установки harbor внутри платформы полностью идентичен процессу установки GitLab.

Jenkins

На текущий момент, установка Jenkins имеет ряд существенных отличий от установки GitLab. Для его установки не требуется размещение PostgreSQL и Redis, что значительно упрощает конфигурирование. Но, в отличие от GitLab, Jenkins имеет дополнительное поле – **Пространство имён узла сборки**.



Это поле отвечает за развёртывание узлов сборки(агентов) Jenkins, с помощью который будет выполняться процесс CI/CD.

Администратор платформы может размещать узлы сборки в том же неймспейсе с основным приложением*. Однако, платформа не ограничивает пользователя в этом плане и позволяет размещать узлы в другом неймспейсе, что даёт возможность гибко настраивать лимиты для как самого приложения, так и для агентов отдельно.

*В платформе реализовано динамическое создание узлов сборки при старте выполнения конвеера. При запуске задания на сборку в указанном неймспейсе Kubernetes будет создан run-time под, который будет удалён после выполнения задания сборки.

Так же, важно отметить, что оператор Jenkins на момент релиза 4.0.2 не обладает возможностью создания PVC. Соответственно, его необходимо создать вручную.

Либо, указать в качестве точки хранения HostPath любой worker-ноды. Например, /data/devkube/jenkins.

Nexus

В виду сложности создания пароля администратора при инициализации Nexus, данный функционал платформой не предусмотрен, что потребует навыков работы с Kubernetes.

При создании экземпляра, так же указывается его имя, неймспейс, ресурсы для приложения, а также адрес для ингресса.

После нажатия кнопки **создать**, экземпляр nexus будет доступен по указанному адресу.

Для того, чтобы узнать сгенерированный nexus пароль, нам потребуется подключиться к одному из мастеров кластера Kubernetes, на котором он был установлен.

После подключения к машине, необходимо выполнить команду

```
kubectl get secret nexus-init-password -n <namespace> -o
jsonpath='{.data.ADMIN_PASSWORD}' | base64 --decode && echo
```

Где,

<namespace> - неймспейс, в котором был развернут nexus.

После чего, переходим по адресу nexus’а, вводим init-пароль и задаём новый.

На этом, установка nexus завершена.

SonarQube

На момент релиза 4.0.2 полноценная автоматизированная установка SonarQube не реализована.

Для его развёртывания потребуются знания Kubernetes.

SonarQube при установке требует подключение к базе данных, а значит, необходимо её создать. Администратор может самостоятельно развернуть выделенную машину под данные цели и в таком случае знаний Kubernetes для развёртывания не потребуется.

Однако, в случае если необходимо развернуть сервер БД внутри платформы, можно сделать это внутри того же неймспейса, в котором планируется развёртывание SonarQube.

Подготовка

Начнём с манифестов. Для наших целей нам потребуется подключиться к уже созданному StorageClass, создать объекты Service, PVC, Deployment и Secret. В Secret мы будем хранить логин и пароль от БД, Service – будет отвечать за доступность нашей базы внутри кластера по DNS-имени, PVC – хранилище, а Deployment – манифест PostgreSQL.

Для каждого объекта мы создадим файлы:

Secret

secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: sonarqube-db-secret
  namespace: infra-sonarqube
type: Opaque
data:
  POSTGRES_USER: <base64_username>
  POSTGRES_PASSWORD: <base64_password>
  POSTGRES_DB: <base64_db>
```

Где,

<base64_*> - результат выполнения команды

```
echo -n '<username/password/db_name>' | base64
```

Таким образом мы получим файл, который останется только применить

```
kubectl apply -f secret.yaml
```

Service

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: sonarqube-db
  namespace: infra-sonarqube
spec:
  selector:
    app: sonarqube-db
  ports:
  - protocol: TCP
    port: 5432
    targetPort: 5432
  type: ClusterIP
```

Применяем его, аналогично секрету

```
kubectl apply -f service.yaml
```

PVC

pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: sonarqube-db-pvc
  namespace: infra-sonarqube
spec:
  storageClassName: <storageclass_name>
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

Где,

<storageclass_name> - созданного StorageClass при создании TopoLVM кластера.

Применяем манифест

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

Deployment

deploy.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sonarqube-db
  namespace: infra-sonarqube
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sonarqube-db
  template:
    metadata:
      labels:
        app: sonarqube-db
    spec:
      containers:
      - name: postgres
        image: postgres:13
        ports:
        - containerPort: 5432
        env:
        - name: POSTGRES_USER
          valueFrom:
            secretKeyRef:
              name: sonarqube-db-secret
              key: POSTGRES_USER
        - name: POSTGRES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: sonarqube-db-secret
              key: POSTGRES_PASSWORD
        - name: POSTGRES_DB
          valueFrom:
            secretKeyRef:
              name: sonarqube-db-secret
              key: POSTGRES_DB
        volumeMounts:
        - mountPath: /var/lib/postgresql/data
          name: sonarqube-persistent-storage
      volumes:
      - name: sonarqube-persistent-storage
        persistentVolumeClaim:
          claimName: sonarqube-db-pvc
```

И так же его применяем

```
kubectl apply -f deploy.yaml
```

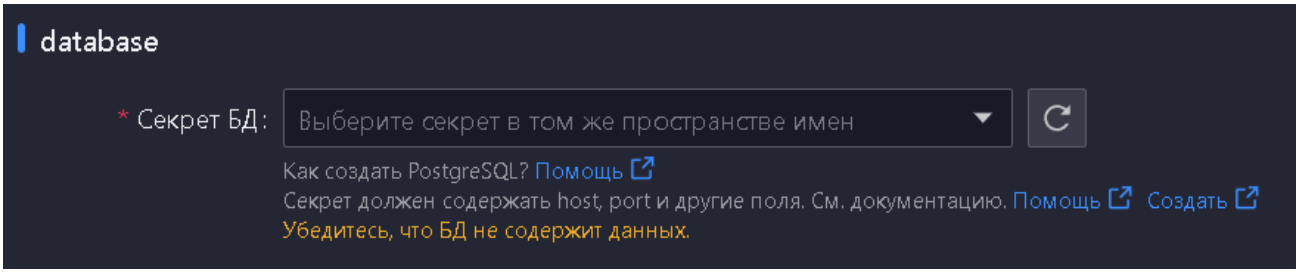
Как итог, в неймспейсе **infra-sonarqube** у нас развернётся готовый экземпляр PostgreSQL интеграции с SonarQube.

Установка

После установки PostgreSQL можно перейти к SonarQube.

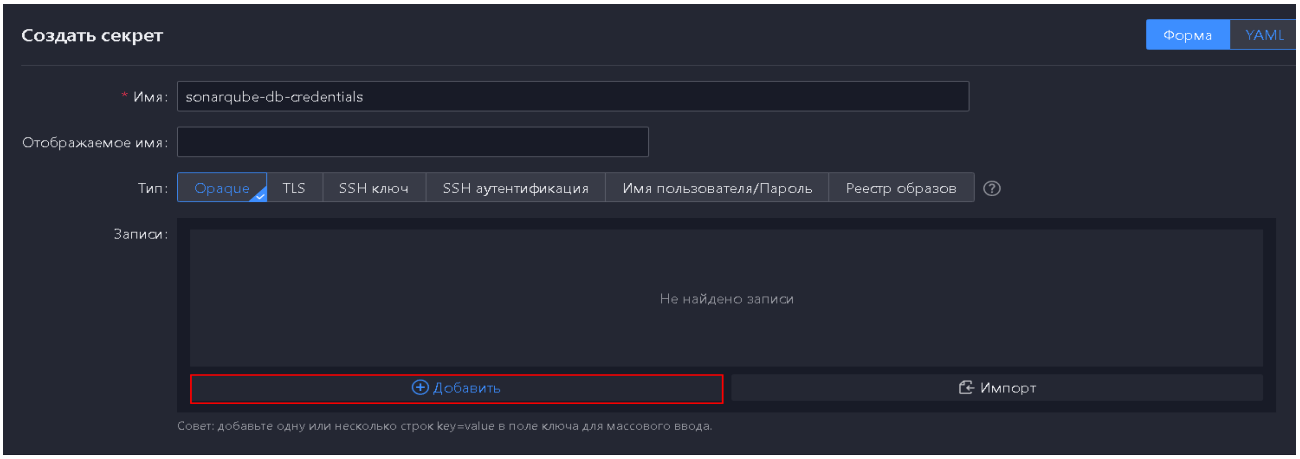
Процесс установки аналогичен Nexus, за исключением того, что в данном случае нам необходимо создать секрет для подключения к БД.

Для того, чтобы создать секрет, необходимо нажать соответствующую кнопку **Создать**.



В появившемся окне вводим **имя** секрета и выбираем тип **Opaque.**

Необходимо добавить несколько полей с помощью кнопки **Добавить**.



Далее, добавляем следующие ключи:

Ключ	*Значение*
host	sonarqube-db.infra-sonarqube.svc
database	<db_name>
port	5432
sslmode	disable
username	<username>
password	<password>

И сохраняем секрет.

Далее применяем секрет в SonarQube и создаём экземпляр.

Установка SonarQube окончена.

Настройка интеграции

После установки экземпляров пользователь с ролью администратора платформы может интегрировать экземпляры с проектами.

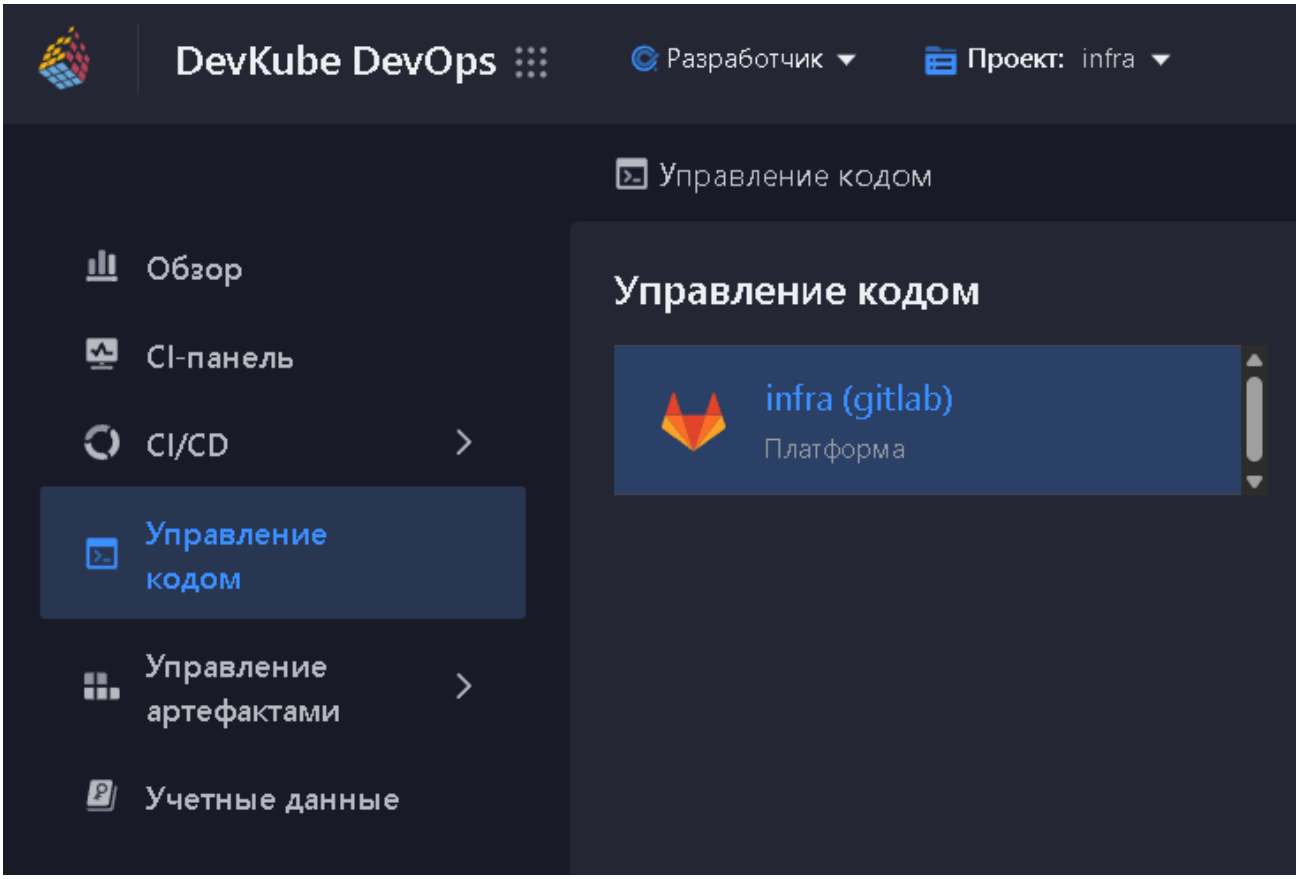
Есть несколько типов интеграций:

- Распределение проектов по имени;
- Распределение публичных проектов;
- Публичный инструмент.

Распределение по имени

При настройке распределения проектов по имени, разработчики будут видеть только те проекты, которые соответствуют имени проекта внутри платформы.

Условно, если разработчик занимается разработкой инфраструктурных решений и имеет доступ к проекту infra, он будет видеть репозитории проекта infra.



При этом, важно отметить, что разработчик может быть прикреплён к нескольким проектам. Для этого, за разработчиком сохранена возможность выбора проекта в верхнем левом углу экрана.

Возможностью распределения проектов по имени обладают такие инструменты, как GitLab, Harbor, Docker Registry и Nexus.

Распределение публичных проектов

Данное распределение даёт доступ всем проектам к публичным репозиториям.

Отлично подойдёт на случай, если исходный код приложения внутри компании является открытым и есть необходимость делиться им с другими командами.

Аналогично распределению по имени, такой возможностью обладают GitLab, Harbor, Docker Registry и Nexus.

Публичный инструмент

Данная интеграция даёт возможность всем командам разработки(проектам) получить доступ к данному инструменту. Это позволяет совместно использовать один инструмент для выполнения различных типов задач.

Данным функционалом располагают такие инструменты как Jenkins и SonarQube.