ООО «РусБИТех-Астра»

Программный комплекс «Astra Dev Platform»

Инструкция по установке

Москва

2025 год

ОГЛАВЛЕНИЕ:

1. Общие сведения

1.1 Назначение документа

Настоящий документ описывает установку Astra Dev Platform (далее Платформа).

1.2 Платформа

Astra Dev Platform – это DevOps-платформа созданная на основе продуктов GitFlic и Боцман, работающий под управлением защищенной операционной системы Astra Linux Special Edition и обеспечивающая полный жизненный цикл разработки, тестирования, доставки и эксплуатации программного обеспечения в enterprise-среде.

Цель платформы — создание DevOps-цикла, где разработанный программный код, собирается, тестируется, развертывается, эксплуатируется и мониторится.

Ключевые принципы:

–Автоматизация: автоматизация DevOps-цикла;

–Изоляция и безопасность: RBAC, GPG-подписи, политики доступа;

–Масштабируемость: поддержка роста команд, проектов и нагрузки.

Платформа объединяет: управление кодом, задачами, CI/CD конвейеров, оркестрацию и эксплуатацию контейнерных приложений.

Программный комплекс предназначен для развертывания в локальной или частной облачной инфраструктуре и ориентировано на высокие требования к безопасности, отказоустойчивости и контролю.

2. Требования к инфраструктуре

Перед началом установки обязательно убедитесь, что вы выполняете все требования из этого чек-листа

Для вашего удобства чек-боксы кликабельны, просто пройдитесь по всему списку, и как только все галочки будут зелёными - вы готовы к установке

Online установка

- В моей инфраструктуре есть DNS-сервер, и я могу создавать A-записи в нём
- В моей инфраструктуре есть NTP-сервер, и машины, которые я буду использовать, настроены на него

- У всех серверов, которые я использую, есть доступ до нашего harbor
- У меня есть возможность создать\использовать 7 машин для установки управляющего кластера боцмана (3 мастера и 3 воркера и АРМ администратора)
- Все машины, которые я планирую использовать, находятся в плоской сети (один broadcast сегмент), между ними нет ограничения в сетевом взаимодействии
- У меня есть IP адрес в сети, которую будут использовать мои серверы, и который не находится в DHCP-пуле. Я готов использовать этот IP адрес как адрес kube-api
- Я использую поддерживаемую операционную систему
- Мои серверы доверяют SSL сертификату регистри, который я использую
- Мои серверы соответствуют минимальным системным требованиям

Минимальные системные требования для узлов

Резервация ресурсов

При установке в гипервизорах крайне рекомендуется резервировать ресурсы для управляющего кластера.

Controlplane

Дисковая система

Для ControlPlane узлов критически важно использовать быстрые диски, обеспечить низкую latency и достаточное количество IOPS (1000+ для ненагруженных кластеров)

Параметр	**Минимальная конфигурация**
CPU (vCPU)	4
RAM (GB)	8
Storage (GB)	45 (65 для Yandex Cloud и 100 для VK)

Yandex Cloud и VK Cloud обеспечивает лимитированное число IOPS для своих SSD.

Для обеспечения работы ETCD нужны узлы ControlPlane с размером диска:

- Yandex Cloud - от 65GB Документация Yandex Cloud
- VK Cloud - от 100GB Документация VK Cloud

Worker

Параметр	**Минимальная конфигурация**
CPU (vCPU)	4
RAM (GB)	12
Storage (GB)	60

Арм администратора

Параметр	**Минимальная конфигурация**
CPU (vCPU)	4
RAM (GB)	8
Storage (GB)	40

Минимально необходимые квоты для облаков

Все значение рассчитаны на 1 кластер с минимальной конфигурацией. С возможностью обновления. Для создания подчиненного кластера увеличивайте квоты минимум x2.

Yandex cloud

Тип Ресурса	Значение
Число дисков	7
Общий объём SSD-дисков	440 GB
Количество vCPU виртуальных машин	28
Общий объем RAM виртуальных машин	72 GB
Количество виртуальных машин	7
Количество статических маршрутов	1
Количество облачных сетей	1
Количество подсетей	1
Количество всех публичных IP-адресов	2
Количество таблиц маршрутизации	1

VK Cloud

Тип Ресурса	Значение
Виртуальных машин	7

Тип Ресурса	Значение
vCPU	28
RAM	73728 МБ
Диски	7
Размер дисков	580 ГБ
IP-адреса Neutron	7

3. Подготовка

Шаг 1. Подготовка дистрибутива

Перенесите архив с дистрибутивом на АРМ администратора любым удобным способом, например с помощью scp:

Команду необходимо запускать с той машины, на которой размещен дистрибутив. Если дистрибутив уже на АРМ администратора то пропустите команду переноса.

```
scp bootsman-VERSION.zip USER@ADDRESS_ARM:/home/USER/
```

Подключитесь к АРМ Администратора с помощью SSH:

```
ssh USER@ADDRESS_ARM
```

Распакуйте архив:

```
unzip bootsman-VERSION.zip
```

В зависимости от настроек распаковщика у вас должно произойти следующее:

1. Создастся дополнительная поддиректория с содержимым архива; В этом случае перейдите в директорию:

```
2. cd bootsman-VERSION
```

3. Содержимое архива будет размещено в текущей директории.

Выдайте права на исполнение:

```
chmod +x bootsmanctl
```

Проверьте корректность выдачи прав:

```
ls -l bootsmanctl
```

Вывод проверки:

```
-rwxr-xr-x 1 as as 104451619 мар 18 13:58 bootsmanctl
```

"x" сигнализирует о том, что файл исполняемый.

Шаг 2. Добавление лицензии

Лицензия и ее файлы должна располагаться в той же директории, где и дистрибутив.

Используйте удобный для вас способ: Добавление лицензии в окружение:

```
export LICENSE=ZzSGquD8GWb2zthw7XJR
```

Или добавьте в файл:

```
mkdir -p .bootsmanctl
cat << EOF >> .bootsmanctl/license.lic
ZzSGquD8GWb2zthw7XJR
EOF
```

АРМ готов к работе

Шаг 3. Образы ОС

Работа GPU возможна только на узлах с ядром linux 6.1+

Перед началом установки подготовьте образы с желаемой ОС. Мы предлагаем два варианта:

- Использовать подготовленный командой Bootsman образ
- Подготовить образ самим в соответствии с нашими инструкциями

Подготовка образа

Создайте инстанс в облаке или гипервизоре и проведите настройку.

Базовые настройки

Выключите SWAP и включите компоненты ядра

```
swapoff -a && sed -ri '/\sswap\s/s/^#?/#/' /etc/fstab
modprobe overlay && modprobe br_netfilter
mkdir /etc/containerd
```

Создайте файл /etc/sysctl.d/sysctl.conf с помощью команды

```
cat <<EOF > /etc/sysctl.d/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-ip6tables=1
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.disable_ipv6=0
net.ipv4.tcp_congestion_control=bbr
vm.overcommit_memory=1
kernel.panic=10
kernel.panic_on_oops=1
fs.inotify.max_user_instances=8192
fs.inotify.max_user_watches=524288
EOF
```

Создайте файл /etc/modules-load.d/containerd.conf с помощью команды

```
cat <<EOF > /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF
```

Отключение брандмауэра

```
ufw disable
```

Загрузка kubernetes компонентов

Загрузите с нашего публичного S3-хранилища дополнительные файлы:

```
- containerd
```

- k8s_${Kubernetes_Version}/(deb|rpm)/*

Kubernetes_Version - соответствует желаемой версии Kubernetes deb - для Astra и Ubuntu rpm - для RedOS

Установка kubernetes компонентов

Установите зависимости kubernetes

```
apt install -y socat ebtables ethtool conntrack open-iscsi nfs-common
```

Установите containerd из скачанного архива

```
tar -C / -xvf cri-containerd-cni-*.tar.gz
```

Установите Kubernetes пакеты

```
for pkg in cri-tools kubernetes-cni kubectl kubelet kubeadm; do
dpkg --install $pkg*.deb
done
apt-mark hold cri-tools kubernetes-cni kubectl kubelet kubeadm
```

Далее примените изменения в sysctl и включите containerd

```
sysctl --system
systemctl daemon-reload && systemctl enable containerd && systemctl start containerd
```

Дополнительные сетевые настройки

Создайте юнит для автозапуска настроек

```
cat <<EOF > /etc/systemd/system/ethtool-conf.service
[Unit]
After=network.target
[Service]
ExecStart=/usr/local/bin/ethtool-conf.sh
[Install]
WantedBy=default.target
EOF
```

И сам скрипт

Используйте имя вашего основного интерфейса вместо ${INTERFACE_NAME}

```
cat <<EOF > /usr/local/bin/ethtool-conf.sh
#!/bin/bash

ethtool -K ${INTERFACE_NAME} tx-udp_tnl-csum-segmentation off
ethtool -K ${INTERFACE_NAME} tx-udp_tnl-segmentation off
EOF
chmod 744 /usr/local/bin/ethtool-conf.sh
chmod 664 /etc/systemd/system/ethtool-conf.service
systemctl daemon-reload
systemctl enable ethtool-conf.service --now
```

Настройка Cloud-init

Очистите cloud-init и выключите инстанс

```
cat /dev/null > /etc/machine-id
cloud-init clean --logs
```

Поддержка GPU

Для создания образа с поддержкой GPU для baremetal или vsphere инсталяций потребуются дополнительные настройки

Добавьте дополнительную опцию загрузки ядра: в файл /etc/default/grub, в строку GRUB_CMDLINE_LINUX_DEFAULT=

```
pci=nocrs,realloc
```

Итоговый файл должен иметь вид:

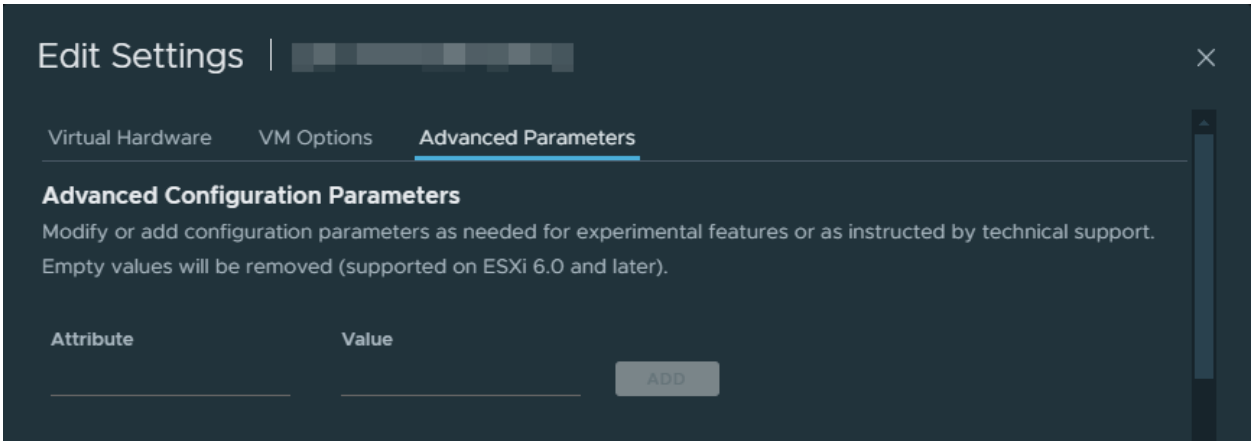
```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="parsec.mac=0 parsec.max_ilev=0 quiet net.ifnames=0 pci=nocrs,realloc"

cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-6.1.90-1-generic root=UUID=c3e1f0eb-c6f9-49fe-b726-f28ff6755df4 ro parsec.mac=0 parsec.max_ilev=0 quiet net.ifnames=0 pci=nocrs,realloc
```

Выключите машину

```
shutdown now
```

Для гипервизора vsphere потребеются дополнительные опции машин в Advanced Parameters

Для этого выберете выключенную машину для образа, заходим в Settings > Advanced Parameters



Добавьте дополнительные параметры

```
pciPassthru.64bitMMIOSizeGB=128
pciPassthru.use64bitMMIO=TRUE
```

Создание образа

Выключите машину, если не выключили в рамках настройки GPU

```
shutdown now
```

После выключения подготовленного инстанса сконвертируйте его образ. В зависимости от использованного облака, воспользуйтесь инструкцией Yandex или инструкцией VK

Когда статус образа перейдет в статус Ready, подготовленный инстанс можно удалять.

Подготовленные образы операционных систем

Yandex Cloud

Для того, чтобы воспользоваться образами, вам понадобится скопировать их к себе в облако.

Для этого выполните команду:

`yc compute image create --name IMAGE_NAME --source-image-id SOURCE_IMAGE_ID --cloud-id bpfj5pch0mue9u8c2ivf`

Где

- `IMAGE_NAME` имя образа, с которым он сохранится у вас в облаке
- `SOURCE_IMAGE_ID` id образа из таблицы ниже

OS	k8s version	id
Astra linux 1.7	v1.28.4	fd8q426d72ejqihvmfvj
Astra linux 1.7	v1.29.3	fd80pubsuot46h500lnc
Astra linux 1.7	v1.30.7	fd8ao375bhd6b6a0ufgv
Astra linux 1.7	v1.31.3	fd807ep6vusd03n2ahmm
Astra linux 1.8	v1.28.4	fd8i7dsbfkqhef2l16b2
Astra linux 1.8	v1.29.3	fd8jselg2l7tg0ersqe1
Astra linux 1.8	v1.30.7	fd8cf5dmhveshlv1t71h
Astra linux 1.8	v1.31.3	fd8du6t4s01e8jdnmeke

VK Cloud/vSphere/Brest

Чтобы воспользоваться образами, выберите образ, скачайте, распакуйте и импортируйте в интересующую вас среду.

`s3-bucket: https://storage-console.bootsman.host/browser/bootsman-public`

`path: template-os/vk/`

4. Установка управляющего кластера

Шаг 1. Выполните команду установки пакетов:

conntrack ethtool ebtables socat open-iscsi nfs-common

```
sudo apt install -y conntrack ethtool ebtables socat open-iscsi nfs-common
```

Шаг 2. Установка host-agent

Установите host-agent боцмана из комплекта поставки на каждый узел. Используйте файлы из комплекта поставки в зависимости от используемой ОС:

```
byoh-agent-version.deb
```

Произведите установку на каждом узле:

```
sudo dpkg --install /home/bootsman/byoh-agent-version-amd64.deb
```

Настройка ролей будущих узлов

Для стабильности и отказоустойчивости кластера рекомендуется ручное назначение трех Master и трех Worker узлов (согласно минимальной конфигурации).

Использование ролей является не обязательным условием для работы платформы - если роли не указать, установщик распределит роли Master и Worker на выделенных ресурсах в случайном порядке.

Что бы добавить ролевую метку на каждый узел используйте команду:

```
export ROLE=NODE_ROLE
sudo sed -i "s#ExecStart=\"/usr/bin/byoh-agent\"#ExecStart=\"/usr/bin/byoh-agent\" --label role=${ROLE}#" /usr/lib/systemd/system/host-agent.service
```

Проверьте результат:

```
cat /usr/lib/systemd/system/host-agent.service
```

Итоговый сервис должен иметь вид:

```
[Unit]
Description=host-agent service
After=network-online.target
[Service]
```

```
Type=simple
WorkingDirectory=/root/.byoh
ExecStart=/usr/bin/byoh-agent --label role=master
User=root
Group=root
Restart=always
[Install]
WantedBy=multi-user.target
```

Выполните команду сохранения полученного результата и перезапуска агента:

```
systemctl daemon-reload && systemctl restart host-agent
```

Шаг 3. Подготовка конфигурационного файла (bootsman.config.yaml)

На АРМ создайте конфиг файл `bootsman.config.yaml` в той же директории, где расположен инсталлятор-`bootsmanctl`

```
touch bootsman.config.yaml
```

И откройте его для редактирования любым текстовом редакторе, например nano:

```
nano bootsman.config.yaml
```

```
kubernetesVersion: KUBERNETES_VERSION
# Раскоментируйте блок в случае airgap установки
# registry:
#  username: PRIVATE_REGISTRY_USERNAME
#  password: PRIVATE_REGISTRY_PASSWORD
#  path: PRIVATE_REGISTRY_URL
#  insecure: PRIVATE_REGISTRY_INSECURE
sshAuthorizedKeys:
 - ssh-ed25519 AAAAC3NzaC1lZKKtLOCNv2g bootsman@stsoft.ru
wait:
 clusterCtlTimeout: 30m0s
 operationTimeout: 40m0s
 operationRetryInterval: 15s
 objCreateTimeout: 4m0s
```

```
  objCreateInterval: 5s
infrastructure:
  clusterNetwork:
    podCidrBlocks: ["POD_CIDR"]
    servicesCidrBlocks: ["SERVICE_CIDR"]
  baremetalProxy:
    currentIp: CURRENT_MACHINE_IP
    currentPort: KUBE_API_PORT
controlPlane:
  endpoint:
    host: K8SCPIP
    port: KUBE_API_PORT
  replicas: KCP_REPLICAS
  label: CONTROLPLANE_LABEL
  kubeletExtraArgs: {}
  machineHealthCheck:
     timeoutDuration: 10m0s
     maxUnhealthy: 100%
workerPool:
  replicas: WORKER_REPLICAS
  label: WORKER_LABEL
  kubeletExtraArgs: {}
  machineHealthCheck:
     timeoutDuration: 10m0s
     maxUnhealthy: 40%
web:
  bootstrapPassword: BOOTSMAN_WEB_BOOTSTRAPPASSWORD
  hostname: BOOTSMAN_WEB_HOSTNAME
```

Pod и Service cidr можно изменить, следуя правилам: маска блоков должна быть не менее 16, блоки не должны пересекаться

Пример заполненного файла конфигурации:

AirgapOnline

```
kubernetesVersion: v1.28.4
registry:
  username: demo
  password: demo
  path: redos-quay.stsoft.lan/bootsman-v210
  insecure: true
sshAuthorizedKeys:
  - ssh-ed25519 AAAAC3NzaC1lZKKtLOCNv2g bootsman@stsoft.ru
wait:
  clusterCtlTimeout: 30m0s
  operationTimeout: 40m0s
  operationRetryInterval: 15s
  objCreateTimeout: 4m0s
  objCreateInterval: 5s
infrastructure:
  clusterNetwork:
    podCidrBlocks: ["172.20.0.0/16"]
    servicesCidrBlocks: ["172.21.0.0/16"]
  baremetalProxy:
    currentIp: 10.104.167.234
    currentPort: 6443
controlPlane:
  endpoint:
    host: ""
    port: 6443
  replicas: 3
  label: master
  kubeletExtraArgs: {}
  machineHealthCheck:
```

```
  timeoutDuration: 10m0s
  maxUnhealthy: 40%
workerPool:
 replicas: 3
 label: worker
 kubeletExtraArgs: {}
 machineHealthCheck:
  timeoutDuration: 10m0s
  maxUnhealthy: 40%
web:
 bootstrapPassword: somePassword123$
 hostname: rancher.redos-v21-ag.dev.stsoft.lan
```

Шаг 4. Запуск установки

Установка платформы "Боцман" запускается с заранее подготовленного АРМ

Выполните команду запуска процесса установки:

```
./bootsmanctl mgmt create cluster-name -i baremetal -c bootsman.config.yaml
```

Вывод инсталляции должен окончиться так:

```
2023-09-10T17:49:46Z | INFO  | finished
```

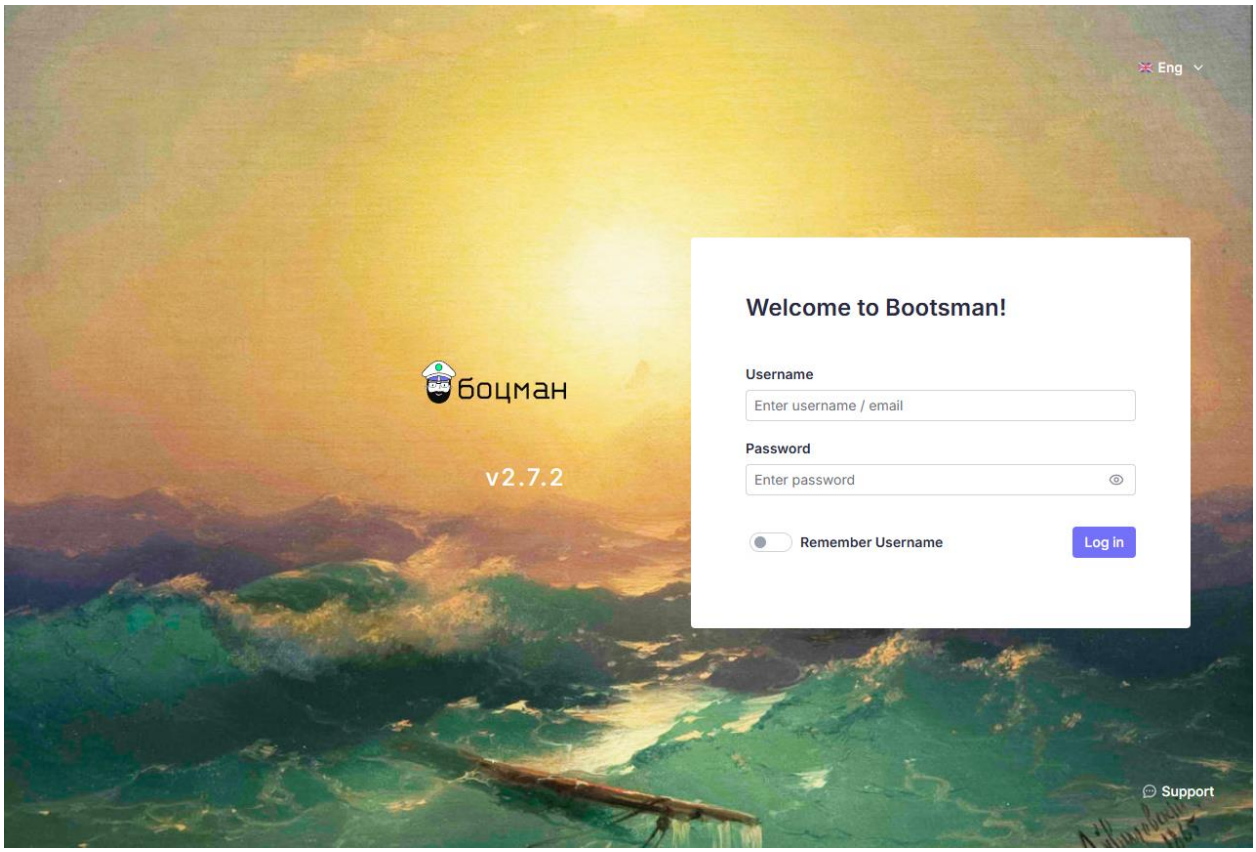
В процессе установки кластера будет формироваться журнал событий.

Доступ в интерфейс

Для успешного доступа в интерфейс нужно добавить запись в ваш DNS-сервер.

Запись состоит из двух параметров: IP-адрес одного из узлов роли worker и доменное имя (Оно указывалось в bootsman.config.yaml в виде опции BOOTSMAN_WEB_HOSTNAME).

После добавления записи, произведите вход в web-панель



login: admin

password: BOOTSMAN_WEB_BOOTSTRAPPASSWORD

Вы авторизовались под пользователем admin. Далее рекомендуем сменить этот пароль с помощью стандартного меню управления пользователями.

5. Установка подчиненного кластера

Для работы кластера Bootsman с внешним ETCD потребуется дополнительная подготовка. Подробнее

Перед началом установки подчинённого кластера следует удовлетворить следующие требования:

- Успешная установка управляющего кластера Bootsman
- DNS-имя управляющего кластера должно разрешаться на каждом узле будущего кластера
- Доступ подчинённого кластера к управляющему

- Сумма символов будущего имени кластера и имени Namespace не должно превышать 30 символов

Установку можно провести двумя способами:

- Использовать установщик Bootsman
- Применить yaml-манифесты

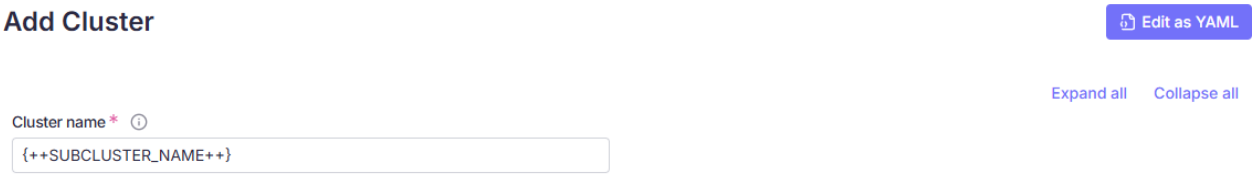
Разворачивание с помощью установщика

Во время заполнения формы вы можете в любой момент переключится в режим yaml

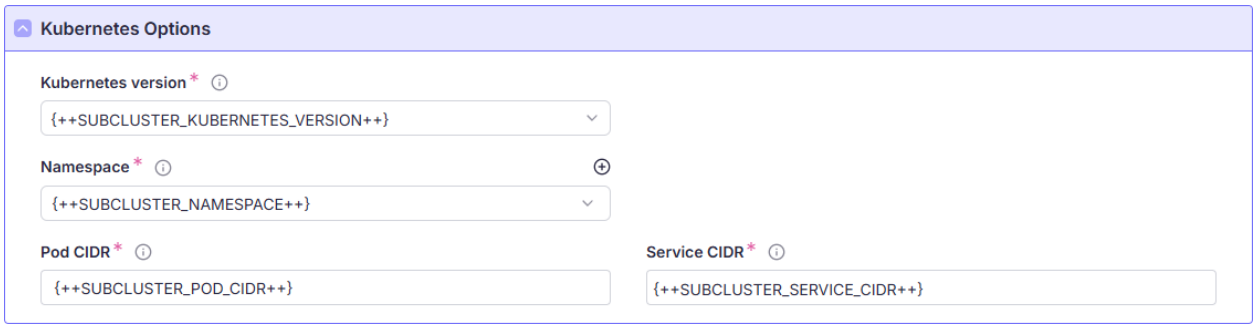
Для создания подчинённого кластера авторизуйтесь в управляющий кластер и нажмите кнопку 

Вас перенаправит на страничку установщика для заполнения данных о будущем кластере.

Сначала укажите имя нового подчинённого кластера



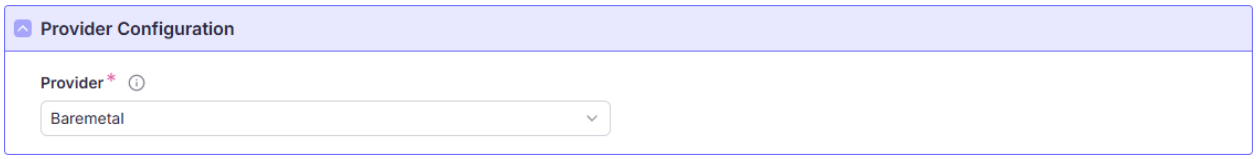
Kubernetes Options



Имя параметра	Описание
SUBCLUSTER_KUBERNETES_VERSION	Версия Kubernetes. Поддерживаемые версии
SUBCLUSTER_NAMESPACE	Namespace управляющего кластера, который будет содержать объекты подчинённого кластера
SUBCLUSTER_POD_CIDR	Блок адресов для Pods. Должен не пересекаться с другими блоками внутри одного Kubernetes кластера. Допустимый размер блока /[1-20] Пример: `podsCidrBlocks: ["172.20.0.0/16"]`
SUBCLUSTER_SERVICE_CIDR	Блок адресов для Service. Должен не пересекаться с другими

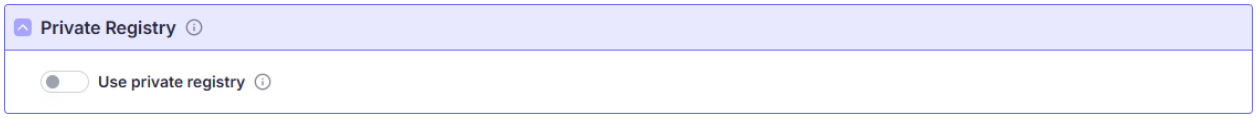
Имя параметра	Описание
	блоками внутри одного Kubernetes кластера. Допустимый размер блока /[1-20] Пример: servicesCidrBlocks: ["172.21.0.0/16"]

Provider Configuration

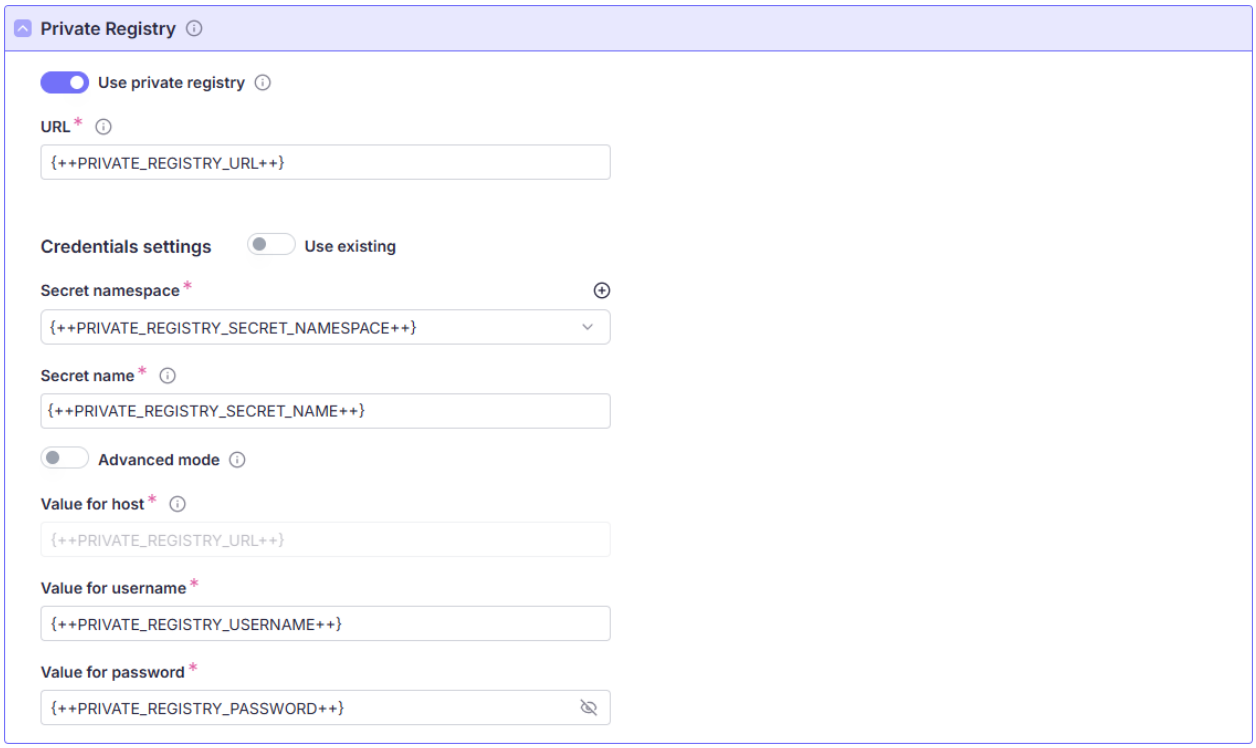


Private Registry

В случае, если вы хотите использовать наш репозиторий образов, то переключите параметр "Use private registry" в выключенное состояние и продолжайте заполнять в соотвествии с следующим разделом.



Если хотите произвести установку из локального репозитория, то заполните поля из формы ниже



В случае использования уже существующего секрета на управляющем кластере, следует использовать настройку "Credential settings" "Use existing" Формы помогут найти секрет и указать имена полей, которые соответствуют пользователю и паролю

Advanced mode поможет вам задать специфические имена ключей в секрете. По умолчанию, username - пользователь, password - пароль

Имя параметра	**Описание**
PRIVATE_REGISTRY_URL	URL вашего репозитория образов. Если установка будет производится online блок registry можно удалить
PRIVATE_REGISTRY_SECRET_NAMESPACE	Namespace, в котором размещен

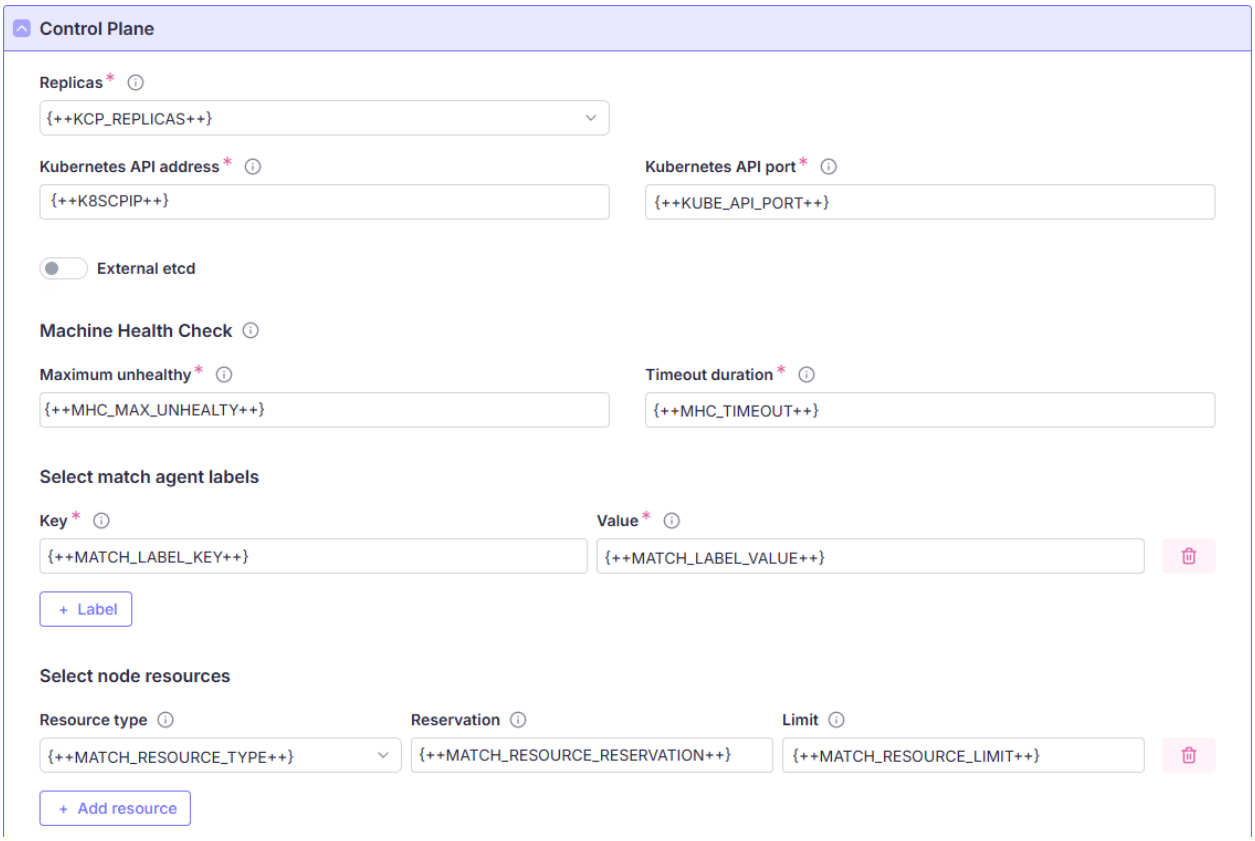
Имя параметра	Описание
	secret для авторизации в приватный репозиторий образов
PRIVATE_REGISTRY_SECRET_NAME	Имя secret для авторизации в приватный репозиторий образов
PRIVATE_REGISTRY_USERNAME	Имя пользователя в репозитории образов. Если установка будет производится online блок registry можно удалить
PRIVATE_REGISTRY_PASSWORD	Пароль для пользователя в репозитории образов. Если установка будет производится online блок registry можно удалить
PRIVATE_REGISTRY_SECRET_HOST_KEY	При использовании

Имя параметра	Описание
	существующего secret. Имя ключа, содержащее параметр HOST
PRIVATE_REGISTRY_SECRET_USERNAME_KEY	При использовании существующего secret. Имя ключа, содержащее параметр USERNAME
PRIVATE_REGISTRY_SECRET_PASSWORD_KEY	При использовании существующего secret. Имя ключа, содержащее параметр PASSWORD

Control Plane

Предупреждение

K8SCPIP - это IP-адрес, который будет использоваться как адрес kube-apiserver. Он должен быть зарезервирован под задачу обеспечения отказоустойчивости IP сервера и не должен выделяться каким либо другим ресурсом



Для работы кластера Bootsman с внешним ETCD потребуется дополнительная подготовка. Подробнее

Имя параметра	**Описание**
KCP_REPLICAS	Количество узлов с ролью master. Может принимать значения 3 или 5
K8SCPIP	IP-адрес балансировщика kube-apiserver для обеспечения отказоустойчивости.

Имя параметра	Описание
	Адрес должен быть свободен и не состоять в dhcp-пулах Подробнее
KUBE_API_PORT	Порт kube-api. По-умолчанию 6443
MHC_MAX_UNHEALTY	Пороговое значение для MachineHealthCheck, при превышении которого не будет выполняться исправления. Для узлов с ролью master, значение по-умолчанию 100%. Для узлов с ролью worker, значение по-умолчанию 40% Подробнее
MHC_TIMEOUT	Время, которое дается узлу на самостоятельное восстановление, прежде чем вмешается MachineHealthCheck. По-умолчанию 10m Подробнее

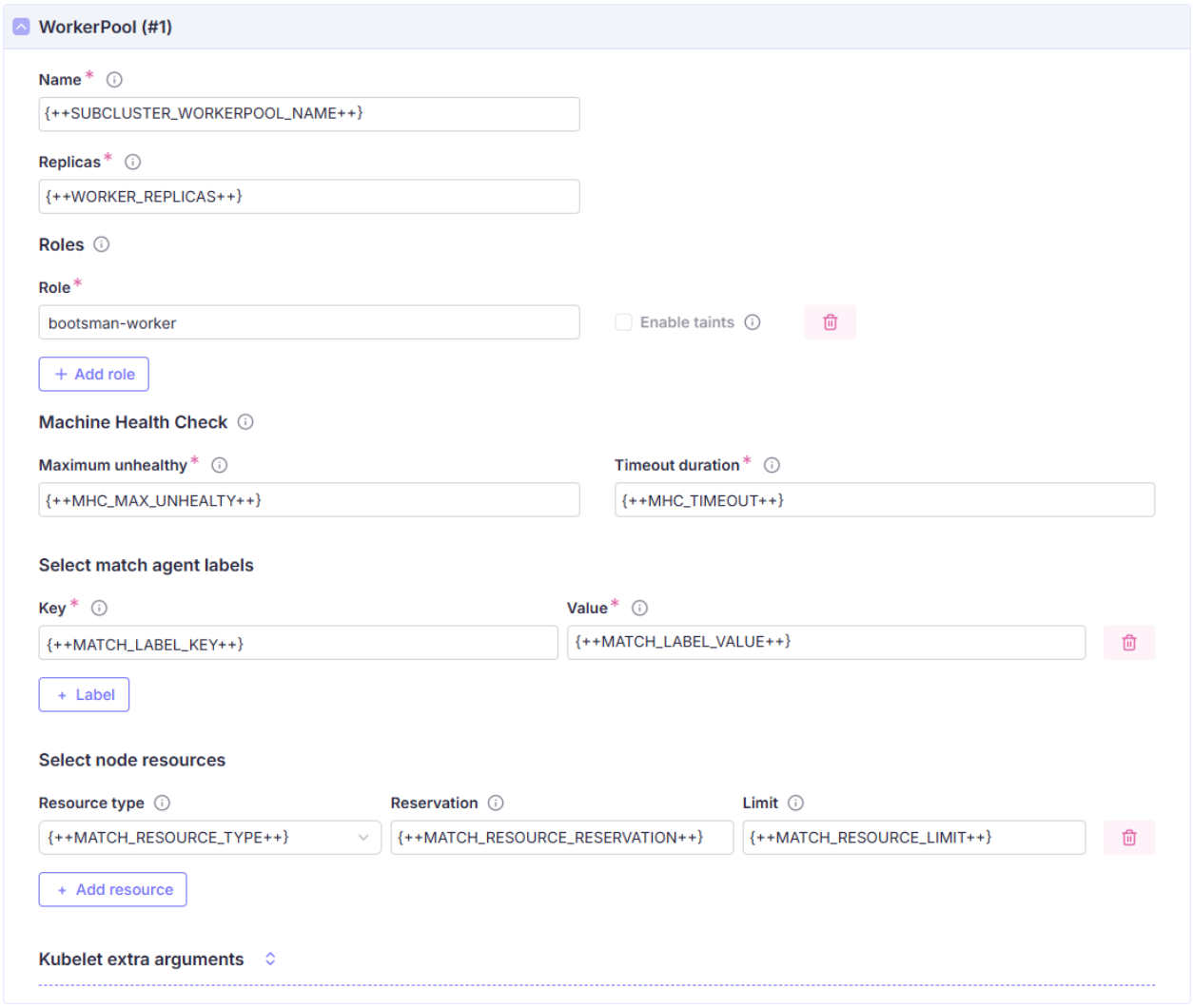
Имя параметра	Описание
MATCH_LABEL_KEY	Опционально. Выбор узла из списка ByoHost по лейблу
MATCH_LABEL_VALUE	Выбор узла из списка ByoHost по лейблу
MATCH_RESOURCE_TYPE	Опционально. Выбор узла из списка ByoHost по типу ресурсов
MATCH_RESOURCE_RESERVATION	Опционально. Минимальное требуемое число ресурсов. Узлы с меньшим значением не будут использованы
MATCH_RESOURCE_LIMIT	Опционально. Максимальное требуемое число ресурсов. Узлы с большим значением не будут использованы

WorkerPool

Вы не ограничены в одном типе ВМ, предназначенных для нагрузки. Для создания дополнительного пула виртуальных машин используйте кнопку

Для работы графических ускорителей установите значения для выбора узлов по GPU (MATCH_RESOURCE_TYPE и MATCH_RESOURCE_RESERVATION)

Если потребуется используйте дополнительные селекторы по LABEL



Имя параметра	Описание
SUBCLUSTER_WORKERPOOL_NAME	Имя пула виртуальных машин

Имя параметра	Описание
WORKER_REPLICAS	Число узлов в workerpool. Сумма узлов во всех workerpool должна быть >=3
Roles	Для большинства инсталляций используйте роль bootsman-worker. Подробнее о ролях
MHC_MAX_UNHEALTY	Пороговое значение для MachineHealthCheck, при превышении которого не будет выполняться исправления. Для узлов с ролью master, значение по-умолчанию 100%. Для узлов с ролью worker, значение по-умолчанию 40% Подробнее
MHC_TIMEOUT	Время, которое дается узлу на самостоятельное восстановление, прежде чем вмешается MachineHealthCheck.

Имя параметра	Описание
	По-умолчанию 10m Подробнее
MATCH_LABEL_KEY	Опционально. Выбор узла из списка ByoHost по лейблу
MATCH_LABEL_VALUE	Выбор узла из списка ByoHost по лейблу
MATCH_RESOURCE_TYPE	Опционально. Выбор узла из списка ByoHost по типу ресурсов
MATCH_RESOURCE_RESERVATION	Опционально. Минимальное требуемое число ресурсов. Узлы с меньшим значением не будут использованы
MATCH_RESOURCE_LIMIT	Опционально. Максимальное требуемое число ресурсов. Узлы с большим значением не будут использованы

Установка с помощью yaml-манифестов

Если вам удобнее способ применения yaml-манифестов, то ниже есть примеры. Заполните их в соответствии с описанием из главы, представленной ранее.

Для применения yaml-манифестов можно использовать kubectl

Манифест secret_private_registry.yaml требуется только для установки из вашего приватного репозитория, в случае установки из репозитория Bootsman - пропустите.

```
kubectl apply -f secret_private_registry.yaml
kubectl apply -f Cluster.yaml
kubectl apply -f Workerpool.yaml
```

secret_private_registry.yamlCluster.yamlWorkerpool.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: PRIVATE_REGISTRY_SECRET_NAME
  namespace: PRIVATE_REGISTRY_SECRET_NAMESPACE
stringData:
  host: PRIVATE_REGISTRY_HOST
  username: PRIVATE_REGISTRY_USERNAME
  password: PRIVATE_REGISTRY_PASSWORD
```

6. Установка GitFlic в Боцман

- Начиная с версии 4.6.1 изменился репозиторий для подключения с helm !
- Начиная с версии 4.6.1 все предыдущие версии чарта помечены как deprecated и не рекомендуются к развертыванию!
- Начиная с версии 4.6.1 версия чарта и версия ПО GitFlic совпадают
- Новый чарт не совместим со старыми версиями чарта (*версии до 4.6.1*)!

Установка развертывания

- Для установки enterprise версии необходимо предварительно загрузить из личного кабинета образ gitflic-server-ee:<Актуальный тег> установить его в ваш реестр контейнеров и указывать его в качестве образа для ПО GitFlic
- По умолчанию используется storageClass.name=default.

- Будьте внимательны! В большинстве платформ контейнеризации параметр `reclaimPolicy` по умолчанию установлен в `Delete`!

С изменениями в версиях и возможностями Helm чарта можно ознакомиться в официальном репозитории

Быстрый запуск

```
helm install gitflic  oci://registry.gitflic.ru/helm/project/gitflic/gitflic-server-helm/gitflic-server \
  --namespace gitflic \
  --create-namespace
```

В данном случае будет развернуто последнее доступное на текущий момент ПО GitFlic. Будет применена конфигурация хранилища (storageClass) установленная по умолчанию в кластере. Сертификаты будут сгенерированы автоматически. Конфигурация для ingress контролера не будет применена. Elasticsearch и его функционал не будут установлены и применены

Доступ к web интерфейсу

```
kubectl -n gitflic port-forward deployments/gitflic 8080:8080
```

Доступ к ssh серверу

```
kubectl -n gitflic port-forward deployments/gitflic 2255:2255
```

Изменение параметров приложения

Изменение baseUrl приложения

Параметр `baseUrl` служит для генерации ссылок внутри приложения и должен содержать в себе конечный домен или хост, по которому будет осуществляться доступ к приложению. Параметр должен быть сформирован с учетом протокола и порта. Данный параметр будет записан в `configMap` приложения. При изменении данного параметра, необходимо перезапустить приложение GitFlic что бы изменения вступили в силу!

Для изменения baseUrl создайте файл `values.yaml` или добавьте в уже существующий директивы:

```
# values.yaml
gitflic:
  baseUrl: "https://example.my.domain:8080"
```

Как получить автоматически сгенерированные сертификаты и ключи

kubectl -n gitflic get secrets gitflic-cert -o jsonpath='{ .data.key\.pem }' | base64 -d > ./key.pem

kubectl -n gitflic get secrets gitflic-cert -o jsonpath='{ .data.key\.pem\.pub }' | base64 -d > ./key.pem.pub

kubectl -n gitflic get secrets gitflic-cert -o jsonpath='{ .data.private_key\.pem }' | base64 -d > ./private_key.pem

kubectl -n gitflic get secrets gitflic-cert -o jsonpath='{ .data.public_key\.pem }' | base64 -d > ./public_key.pem

После выполнения команд, будут созданы следующие файлы:

- key.pem - Приватный ключ для SSH сервера платформы GitFlic
- key.pem.pub - Публичный ключ для SSH сервера платформы GitFlic
- private_key.pem - Приватный ключ для работы платформы GitFlic
- public_key.pem - Публичный ключ для работы платформы GitFlic

Сохраните эти ключи в надежном месте. Изменение этих ключей в процессе эксплуатации платформы GitFlic запрещено и может привести к неработоспособности платформы !

Как установить собственные сертификаты и ключи

Создайте ключевые пары для SSH сервера и сервисов платформы GitFlic:

```
ssh-keygen -t ed25519 -N "" -q -f key.pem
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

Добавьте дополнительные параметры к команде установки:

```
helm install gitflic  oci://registry.gitflic.ru/helm/project/gitflic/gitflic-server-helm/gitflic-server \
  --namespace gitflic \
  --create-namespace \
  --set generate_certificate=false \
  --set-file gitflic.certs.key=key.pem \
  --set-file gitflic.certs.keyPub=key.pem.pub \
  --set-file gitflic.certs.public_key=public_key.pem \
  --set-file gitflic.certs.private_key=private_key.pem
```

Включение "Поиск по коду"

По умолчанию функционал поиска по коду отключен. Данный функционал требует подключения **elasticsearch** к развертыванию и конфигурирования приложения для работы с ним.

Для включения функционала поиска по коду создайте файл values.yaml или добавьте в уже существующий директивы:

```
# values.yaml
elasticsearch:
  install: true
gitflic:
  elasticsearch:
    enable: true
```

Изменение параметров для SMTP сервера

По умолчанию сервер SMTP не настроен и не осуществляет отправку писем.

Для изменения параметров SMTP сервера создайте файл values.yaml или добавьте в уже существующий директивы:

```
# values.yaml
gitflic:
  mail:
    host: "smtp.foo.bar" # Хост SMTP сервера
    port: 587 # Порт SMTP сервера
    username: foo@bar.net # Имя пользователя для подключения к SMTP серверу
    password: Superp@$$word # Пароль пользователя для подключения к SMTP серверу
    sender:
      name: foobar # Имя пользователя отображаемое в качестве отправителя писем
      email: foo@noreply.bar # Email отображаемый в качестве отправителя писем
```

Изменение параметров развертывания

Изменение imagePullPolicy параметра

По умолчанию, для всех развертываний параметр imagePullPolicy установлен в **IfNotPresent**.

Для изменения параметра imagePullPolicy создайте файл values.yaml или добавьте в уже существующий директивы:

```
# values.yaml
redis:
  imagePullPolicy: "Never" # Или Always, в зависимости от ваших требований
postgres:
  imagePullPolicy: "Never" # Или Always, в зависимости от ваших требований
elasticsearch:
  imagePullPolicy: "Never" # Или Always, в зависимости от ваших требований
gitflic:
  imagePullPolicy: "Never" # Или Always, в зависимости от ваших требований
```

Включение лимитов для контейнеров

Для всех контейнеров внутри развертывания, установлены параметры resources.requests в соответствии с минимальными требованиями ПО для функционирования платформы. Лимиты (параметры resources.limits) установлены только для контейнеров **gitflic-server** и **elasticsearch**. Остальные контейнеры используют по умолчанию все доступные ресурсы кластера. Если необходимо ограничить и/или переопределить существующие, используйте в качестве корневой директивы имя сервиса и стандартные директивы для определения ресурсов контейнера. Текущие лимиты могут быть изменены в будущем и/или убраны совсем. Подробнее см. в файле values.yaml

Для изменения лимитов контейнеров создайте файл values.yaml или добавьте в уже существующий директивы:

```
# values.yaml
# Пример для postgresql
postgres:
  containers:
    resources:
```

```
    requests:
      cpu: "500m"
      memory: "512Mi"
    limits:
      cpu: "1000m"
      memory: "2048Mi"
```

Изменение imagePullSecrets параметра

По умолчанию, для всех развертываний параметр imagePullSecrets не установлен. Для доступа к приватным реестрам необходимо создать ресурс типа secret с параметрами для доступа к реестру:

```
kubectl create secret docker-registry my-private-registry \
  --namespace gitflic \
  --docker-server #Адрес до реестра контейнеров \
  --docker-username #Имя пользователя для доступа к реестру \
  --docker-password #Пароль пользователя для доступа к реестру
```

Для указания секрета с данными реестра создайте файл values.yaml или добавьте в уже существующий директивы:

```
# values.yaml
redis:
 imagePullSecrets: "my-private-registry" # Или имя указанное при создании секрета
postgres:
 imagePullSecrets: "my-private-registry" # Или имя указанное при создании секрета
elasticsearch:
 imagePullSecrets: "my-private-registry" # Или имя указанное при создании секрета
rabbitmq:
 imagePullSecrets: "my-private-registry" # Или имя указанное при создании секрета
gitflic:
 imagePullSecrets: "my-private-registry" # Или имя указанное при создании секрета
```

Изменение стандартной зоны для кластера

По умолчанию установлена зона `svc.cluster.local`.

Для изменения зоны кластера создайте файл `values.yaml` или добавьте в уже существующий директивы:

```
cluster:
  zone: # Имя вашей зоны кластера
```

Изменение стандартных образов контейнеров

По умолчанию используются образы из публичного реестра `registry.gitflic.ru`. В случае отсутствия возможности подключения к публичному реестру, рекомендуется скачать образы из данного развертывания, любым доступным способом и прогрузить их в ваш приватный реестр, так как развертывание предполагает использование образов указанных в конфигурации развертывания.

Для изменения стандартных образов создайте файл `values.yaml` или добавьте в уже существующий директивы:

```
# values.yaml
redis:
  image:
    repository: my-private-registry/redis
    tag: "6.2"
postgres:
  image:
    repository: my-private-registry/postgres
    tag: "12"
rabbitmq:
  image:
    repository: my-private-registry/rabbitmq
    tag: "3.13-alpine"
elasticsearch:
  image:
    repository: my-private-registry/elasticsearch
    tag: "7.16.2"
```

```
gitflic:
  image:
    repository: my-private-registry/gitflic-server-enterprise
    tag: "4.6.1"
```

Изменение параметров хранения данных

Изменение storageClassName

- ** Рекомендуется использовать longhorn, в качестве программно определяемого хранилища*
- ** В Платформе контейнеризации "Боцман", longhorn установлен по умолчанию как программно определяемое хранилище*

Для изменения storageClassName создайте файл values.yaml или добавьте в уже существующий директивы:

```
#values.yaml
redis:
  storage:
    storageClassName: #Ваш storageClass, например longhorn
postgres:
  storage:
    storageClassName: #Ваш storageClass, например longhorn
elasticsearch:
  storage.:
    storageClassName: #Ваш storageClass, например longhorn
gitflic:
  storage:
    storageClassName: #Ваш storageClass, например longhorn
```

Выполните установку с применением созданного файла values.yaml:

```
helm install gitflic  oci://registry.gitflic.ru/helm/project/gitflic/gitflic-server-helm/gitflic-server \
    --namespace gitflic \
```

```
  --create-namespace \
  --values ./values.yaml
```

Отключение использования PersistentVolumeClaim

При отключении использования PersistentVolumeClaim, к контейнерам будет применен параметр emptyDir. Это приведет к тому, что для данных не будет выделено отдельного хранилища и все изменения будут записаны внутри контейнера. Не используйте данный параметр в production среде!

Для отключения использования PersistentVolumeClaim создайте файл values.yaml или добавьте в уже существующий директивы:

```
# values.yaml
redis:
  storage:
    usePVC: false

postgres:
  storage:
    usePVC: false

elasticsearch:
  storage:
    usePVC: false

gitflic:
  storage:
    usePVC: false
```

Изменение количества выделяемого места в хранилище

Для изменения количества выделяемого места под хранение данных создайте файл values.yaml или добавьте в уже существующий директивы:

```
# values.yaml

redis:
  storage:
    capacity: 3Gi # Измените в случае если не хватает места под хранение кеша

postgres:
  storage:
    capacity: 5Gi # Измените в случае если БД превысит текущий размер

elasticsearch:
  storage:
    capacity: 20Gi # Измените в случае если размер БД превысит текущий размер

gitflic:
  storage:
    capacity: 20Gi # Измените с учетом ваших предполагаемых размеров репозиториев и/или данных реестра
```

Изменение параметров баз данных

Подключение собственной базы данных Postgresql

Перед подключением собственной БД postgresql необходимо самостоятельно инициализировать базу и установить необходимые расширения: - pgcrypro - pg_trgm

Для подключения собственной базы `postgresql` создайте файл `values.yaml` или добавьте в уже существующий директивы:

```
# values.yaml
postgres:
  install: false
  host: foo.bar.sql #Адрес до хоста postgresql
  port: 5432 # Порт хоста postgresql. Значение по умолчанию 5432
  user: foobar # Пользователь для бд postgresql. Значение по умолчанию gitflic
```

```
  password: foobarpassword # Пароль для доступа к базе данных postgresql
  database: # Имя базы данных. Значение по умолчанию gitflic
```

Выполните установку с применением созданного файла values.yaml:

```
helm install gitflic  oci://registry.gitflic.ru/helm/project/gitflic/gitflic-server-helm/gitflic-server \
    --namespace gitflic \
    --create-namespace \
    --values ./values.yaml
```

Как получить пароль от базы данных Postgresql

В случае установки БД Postgresql из чарта, пароль для доступа к БД будет сгенерирован автоматически и помещен в Secret в том же пространстве имен в котором было выполнено развертывание.

```
kubectl -n gitflic get secrets passwords -o jsonpath='{ .data.postgres }'  | base64 -d
```

Параметры для ingress контроллера

Шаблоны для ingress контроллера работают только с контроллером nginx!

Включение создания шаблона

По умолчанию создание шаблона для ingress контроллера отключено. Автоматическая генерация имени секрета с tls сертификатами сделана для развертываний в который используется сторонний cert-manager для генерации сертификатов, например Let's Encrypt

Для включения создания шаблона создайте файл values.yaml или добавьте в уже существующий директивы:

```
# values.yaml
ingress:
  create: true
  tls:
    secretName: "" # Имя секрета с сертификатами tls. Если оставить пустым, имя секрета будет сгенерировано автоматически
  rules:
    host: "example.gitflic.local" # Хост по которому будет осуществляться доступ
```

Отключение tls сертификатов

Для отключения доступа по HTTPS создайте файл values.yaml или добавьте в уже существующий директивы:

```
# values.yaml
ingress:
  tls:
    enabled: false
```

Добавление аннотаций к шаблону ingress

Для добавления собственных аннотаций создайте файл values.yaml или добавьте в уже существующий директивы:

```
# values.yaml
ingress:
  annotations:
    nginx.ingress.kubernetes.io/force-ssl-redirect: "true" # Добавление принудительного перенаправления на HTTPS протокол
    # Остальные аннотации которые необходимо добавить в шаблон
```

Отключение предустановленных аннотаций

По умолчанию в шаблон включены некоторые аннотация для контроллера. Подробнее см. в файле values.yaml в блоке ingress.annotationsStatic.

Для отключения предустановленных аннотаций создайте файл values.yaml или добавьте в уже существующий директивы:

```
# values.yaml
ingress:
  annotationsStatic: {}
```

Пример полного файла values.yaml

Ниже представлен пример полного файла values.yaml для развертывания в production среде.

Предполагается что:

- В кластере заранее создан storageClassName=longhorn-gitflic с политикой reclaimPolicy=Retain и доп настройками хранения.
- В кластере установлен ingress контроллер nginx. cert-manager отсутствует.
- Сертификаты для tls созданы и добавлены вручную как secret=gitflic-tls-secret.
- Включено принудительное перенаправление на HTTPS протокол.
- В качестве хоста используется адрес https://example.gitflic.local.
- Образ gitflic-ee:4.6.1 скачан из ЛК и загружен в my-private-registry по пути my-private-registry/gitflic-enterprise:4.6.1.
- Создан secret=private-registry-secret для доступа к приватному реестру где расположен образ gitflic-enterprise:4.6.1.
- Включен функционал Поиск по коду.
- Используется стандартная зона кластера svc.cluster.local.
- Сертификаты и ключи для SSH будут сгенерированы автоматически.
- Внешняя БД postgresql **не будет** использована.
- Предполагаемый размер хранилища репозиториев и реестра 100Gb.
- SMTP сервер **не используется**.

```
# values.yaml
redis:
  storage:
    storageClassName: "longhorn-gitflic"

postgres:
  storage:
    storageClassName: "longhorn-gitflic"

elasticsearch:
  install: true
  storage:
    storageClassName: "longhorn-gitflic"
```

```
gitflic:
  image:
    repository: my-private-registry/gitflic-enterprise
    tag: "4.6.1"
  imagePullSecrets: "private-registry-secret"
  storage:
    storageClassName: "longhorn-gitflic"
    capacity: 100Gi
  baseUrl: "https://example.gitflic.local"
  elasticsearch:
    enable: true

ingress:
  create: true
  annotations:
    nginx.ingress.kubernetes.io/force-ssl-redirect: "true"
  tls:
    secretName: "gitflic-tls-secret"
  rules:
    host: "example.gitflic.local"
```

7. Установка ADP

Предварительные действия

Настроено подключение к рабочему кластеру Боцман.

Развертывание

Развертывание стартовой страницы производится в namespace, в котором развернут Gitflic.

- Разархивируйте архив adp.zip

- Перейдите в директорию adp

- Отредактируйте файлы config-map.yaml и ingress.yaml – заполните параметры конфигурации исходя из ваших условий

- Выполните команду kubectl apply -f .