



ООО «Среда Разработки»
ИНН/КПП 7731343398/ 772801001
ОГРН 1177746046895

117437, г. Москва, вн.тер.г. Муниципальный округ Коньково,
ул. Академика Арцимовича, д. 17, помещ. 3/1

ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ И УСТАНОВКЕ ПО ДЛЯ МАШИННОГО ОБУЧЕНИЯ НА БАЗЕ SredaFlowAi

Листов 29



СОДЕРЖАНИЕ

1	Введение.....	4
2	Требования к инфраструктуре	5
2.1	Программные требования.....	5
2.2	Аппаратные требования.....	5
2.3	Сетевые требования	5
3	План развертывания.....	6
3.1	Первоначальная настройка БД.....	6
3.1.1	MySQL.....	6
3.1.2	MinIO	8
3.2	Установка сервисов	9
3.2.1	API Gateway	9
3.2.2	Бэкенд-сервисы	22
3.2.3	Фронтенд-сервис	24
4	Проверка развертывания	25
4.1	Проверка состояния подов	25
4.2	Проверка TLS-сертификатов.....	25
4.3	Проверка внешнего IP Ingress Gateway	25
4.4	Функциональная проверка.....	26
5	Типовые ошибки и их устранение.....	27
6	Обновление платформы	28
6.1	Обновление Helm-релиза.....	28



ООО «Среда Разработки»
ИНН/КПП 7731343398/ 772801001
ОГРН 1177746046895

117437, г. Москва, вн.тер.г. Муниципальный округ Коньково,
ул. Академика Арцимовича, д. 17, помещ. 3/1

6.2	Откат релиза.....	28
6.3	Резервное копирование перед обновлением.....	28



1 ВВЕДЕНИЕ

Настоящая инструкция описывает порядок развертывания платформы SredaFlowAi в кластере Kubernetes с использованием Helm в качестве пакетного менеджера. Все компоненты платформы устанавливаются через Helm-чарты - внутренние чарты проекта.

Все команды выполняются с правами администратора кластера (cluster-admin). В тексте символ <значение> обозначает плейсхолдер, который необходимо заменить на реальное значение, sreda.domain.tld - доменное имя платформы.



2 ТРЕБОВАНИЯ К ИНФРАСТРУКТУРЕ

2.1 Программные требования

Компонент	Версия	Комментарий
Kubernetes	≥ 1.26	EKS, GKE, AKS или on-premise
kubectl	≥ 1.26	Соответствует версии кластера
Helm	≥ 3.10	Обязателен для установки всех компонентов
Istio	≥ 1.17	Устанавливается через Helm (см. раздел 3.2.1)
cert-manager	≥ 1.12	Устанавливается через Helm (см. раздел 3.2.1)

2.2 Аппаратные требования

Ресурс	Минимум	Рекомендуется
CPU (воркер-узлы)	8 ядер	16 ядер и более
RAM (воркер-узлы)	16 ГБ	32 ГБ и более
Дисковое пространство	100 ГБ SSD	500 ГБ SSD и более
Количество узлов	3	5 и более

2.3 Сетевые требования

- Доступ к репозиториям SredaFlowAi;
- Настроенный StorageClass в кластере (например, standard);
- Внешний IP-адрес или DNS для Istio Ingress Gateway;
- Открытые порты: 80 (HTTP), 443 (HTTPS), 6443 (Kubernetes API).



3 ПЛАН РАЗВЕРТЫВАНИЯ

Компоненты устанавливаются строго в следующем порядке:

1. Создание namespace sreda-system;
2. Установка баз данных (MySQL (Community Edition), MinIO);
3. Установка Istio (istio-base - istiod - istio-ingressgateway);
4. Установка cert-manager и настройка TLS-сертификатов;
5. Установка Dex (OIDC-провайдер);
6. Установка OIDC AuthService;
7. Настройка маршрутизации (Istio Gateway, VirtualServices, AuthorizationPolicy);
8. Установка бэкенд-сервисов;
9. Установка фронтенд-сервиса;
10. Проверка развертывания.

Создание namespace до начала установки:

```
kubectl create namespace sreda-system
```

```
kubectl create namespace istio-system
```

```
kubectl create namespace cert-manager
```

3.1 Первоначальная настройка БД

3.1.1 MySQL

MySQL (Community Edition) используется для хранения метаданных пайплайнов, экспериментов и ML-артефактов.



Шаг 1. Создать Secret с учетными данными:

```
kubectl create secret generic mysql-custom-secret \  
--from-literal=mysql-root-password=<root_password> \  
--from-literal=mysql-password=<db_password> \  
-n sreda-system
```

Шаг 2. Создать файл значений mysql-values.yaml:

```
global:  
  storageClass: "standard"  
auth:  
  existingSecret: "mysql-custom-secret"  
  database: "sreda_metadata"  
  username: "sreda_user"  
initdbScripts:  
  init_db.sql: |  
    CREATE DATABASE IF NOT EXISTS sreda_pipelines;  
    GRANT ALL PRIVILEGES ON sreda_metadata.* TO 'sreda_user'@'%';  
    GRANT ALL PRIVILEGES ON sreda_pipelines.* TO 'sreda_user'@'%';  
    FLUSH PRIVILEGES;  
primary:  
  persistence:  
    enabled: true  
    size: 20Gi
```

Шаг 3. Установить чарт:

```
helm install mysql ./charts/mysql-*.tgz \  
--namespace sreda-system \  
-f mysql-values.yaml
```



Шаг 4. Проверить запуск пода:

```
kubectl get pods -n sreda-system -l app.kubernetes.io/name=mysql
```

Ожидаемый результат: под в состоянии Running.

3.1.2 MinIO

MinIO используется как объектное хранилище для артефактов пайплайнов и моделей.

Шаг 1. Создать Secret с учетными данными:

```
kubectl create secret generic minio-custom-secret \  
--from-literal=rootUser=sreda_admin \  
--from-literal=rootPassword=SredaPass123! \  
-n sreda-system
```

Шаг 2. Создать файл значений minio-values.yaml:

```
existingSecret: "minio-custom-secret"
```

```
mode: standalone
```

```
service:
```

```
  type: ClusterIP
```

```
  port: 9000
```

```
  consolePort: 9001
```

```
persistence:
```

```
  enabled: true
```

```
  storageClass: "standard"
```

```
  size: 50Gi
```

```
buckets:
```

```
  - name: sreda-artifacts
```

```
    policy: none
```



purge: false

- name: sreda-models

policy: none

purge: false

- name: sreda-pipelines

policy: none

purge: false

Шаг 3. Установить чарт:

```
helm install minio ./charts/minio-*.tgz \  
--namespace sreda-system \  
-f minio-values.yaml
```

Шаг 4. Проверить запуск пода:

```
kubectl get pods -n sreda-system -l app=minio
```

3.2 Установка сервисов

3.2.1 API Gateway

API Gateway состоит из четырех компонентов, устанавливаемых последовательно: Istio (service mesh + ingress), cert-manager (TLS), Dex (OIDC-провайдер), OIDC AuthService (аутентификация запросов).

Istio

Шаг 1. Установить istio-base (CRDs):

```
helm install istio-base ./charts/base-*.tgz \  
--namespace istio-system \  
--set defaultRevision=default
```



Шаг 2. Создать файл значений `istiod-values.yaml`:

global:

meshID: mesh1

multiCluster:

clusterName: SredaCluster

meshConfig:

accessLogFile: /dev/stdout

enableAutoMtls: true

defaultConfig:

proxyMetadata:

OUTPUT_CERTS: "true"

Конфигурация `ExtensionProvider` для `OIDC AuthService`.

Необходима для работы `AuthorizationPolicy` с action: CUSTOM.

extensionProviders:

- name: oidc-authservice-provider

envoyExtAuthzHttp:

service: oidc-authservice.sreda-system.svc.cluster.local

port: 8080

includeRequestHeadersInCheck:

- cookie

- authorization

headersToUpstreamOnAllow:

- sredaflow-userid

- sredaflow-groups



pilot:

autoscaleEnabled: true

autoscaleMin: 1

autoscaleMax: 5

resources:

requests:

cpu: 500m

memory: 2048Mi

Шаг 3. Установить istiod (Control Plane):

```
helm install istiod ./charts/istiod-*.tgz \
```

```
--namespace istio-system \
```

```
-f istiod-values.yaml
```

Шаг 4. Создать файл значений ingress-values.yaml:

service:

type: LoadBalancer

ports:

- name: status-port

port: 15021

targetPort: 15021

- name: http2

port: 80

targetPort: 8080

- name: https

port: 443

targetPort: 8443



autoscaling:

enabled: true

minReplicas: 2

maxReplicas: 5

targetCPUUtilizationPercentage: 80

resources:

requests:

cpu: 100m

memory: 128Mi

limits:

cpu: 2000m

memory: 1024Mi

Шаг 5. Установить Istio Ingress Gateway:

```
helm install istio-ingressgateway ./charts/gateway-*.tgz \
--namespace istio-system \
-f ingress-values.yaml
```

Шаг 6. Проверить запуск подов Istio:

```
kubectl get pods -n istio-system
```

cert-manager

Шаг 1. Создать файл значений cert-manager-values.yaml:

installCRDs: true

replicaCount: 1

resources:

requests:

cpu: 100m



memory: 128Mi

limits:

cpu: 500m

memory: 512Mi

prometheus:

enabled: false

Шаг 2. Установить чарт:

```
helm install cert-manager ./charts/cert-manager-*.tgz \
--namespace cert-manager \
-f cert-manager-values.yaml
```

Шаг 3. Проверить запуск подов:

```
kubectl get pods -n cert-manager
```

Шаг 4. Создать файл root-ca.yaml - корневой CA и ClusterIssuer:

Самоподписанный ClusterIssuer для начальной загрузки CA

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: selfsigned-issuer
spec:
  selfSigned: {}
```

Корневой сертификат CA

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: sredaflow-root-ca
```



namespace: cert-manager

spec:

isCA: true

commonName: sredaflow-root-ca

secretName: sredaflow-root-ca-secret

issuerRef:

name: selfsigned-issuer

kind: ClusterIssuer

ClusterIssuer на основе созданного CA

apiVersion: cert-manager.io/v1

kind: ClusterIssuer

metadata:

name: sredaflow-ca-issuer

spec:

ca:

secretName: sredaflow-root-ca-secret

kubectl apply -f root-ca.yaml

Шаг 5. Создать файл gateway-certificate.yaml - TLS-сертификат для Ingress Gateway:

apiVersion: cert-manager.io/v1

kind: Certificate

metadata:

name: sreda-tls-cert

namespace: istio-system

spec:

secretName: sreda-tls-cert



issuerRef:

name: sredaflow-ca-issuer

kind: ClusterIssuer

dnsNames:

- "sreda.domain.tld"

- "dex.sreda.domain.tld"

kubectl apply -f gateway-certificate.yaml

Шаг 6. Проверить статус сертификата:

kubectl get certificate -n istio-system sreda-tls-cert

Ожидаемый результат: поле READY = True.

Dex (OIDC-провайдер)

Шаг 1. Создать Secret с client secret для платформы:

kubectl create secret generic dex-client-secrets \

--from-literal=SREDA_CLIENT_SECRET=<client_secret> \

-n sreda-system

Шаг 2. Создать файл значений dex-values.yaml:

config:

issuer: https://dex.sreda.domain.tld

storage:

type: kubernetes

config:

inCluster: true

staticClients:



```
- id: sreda-platform
name: "SredaFlowAI Platform"
redirectURIs:
  - "https://sreda.domain.tld/login/"
secretEnv: SREDA_CLIENT_SECRET
```

staticPasswords:

```
- email: "admin@sreda.domain.tld"
  hash: "$2a$10$<bcrypt_hash_замените_на_реальный>"
  username: "admin"
  userID: "08a51652-38b1-4f1f-a034-2e5f5f4756ef"
```

envFrom:

```
- secretRef:
  name: dex-client-secrets
```

Секция staticPasswords.hash должна содержать хэш пароля. Команда для генерации:

```
htpasswd -bnBC 10 "" <password> | tr -d ':\n'
```

Использование тестового хэша в эксплуатационной среде недопустимо.

Шаг 3. Установить чарт:

```
helm install dex ./charts/dex-*.tgz \
--namespace sreda-system \
-f dex-values.yaml
```



OIDC AuthService

Шаг 1. Создать Secret с client secret для AuthService:

```
kubectrl create secret generic oidc-authservice-secret \  
--from-literal=CLIENT_SECRET=<client_secret> \  
-n sreda-system
```

Значение <client_secret> должно совпадать с SREDA_CLIENT_SECRET из Secret dex-client-secrets.

Шаг 2. Создать файл значений authservice-values.yaml:

replicaCount: 2

env:

- name: OIDC_PROVIDER
value: "https://dex.sreda.domain.tld"
- name: CLIENT_ID
value: "sreda-platform"
- name: REDIRECT_URI
value: "https://sreda.domain.tld/login/"
- name: COOKIE_DOMAIN
value: "sreda.domain.tld"
- name: SESSION_MAX_AGE
value: "86400"
- name: OIDC_SCOPES
value: "openid,profile,email,groups"

envFrom:

- secretRef:
name: oidc-authservice-secret



Шаг 3. Установить чарт:

```
helm install oidc-authservice ./charts/oidc-authservice-*.tgz \
--namespace sreda-system \
-f authservice-values.yaml
```

Настройка маршрутизации Istio

Шаг 1. Создать файл routing-manifests.yaml:

Istio Gateway - точка входа для HTTP/HTTPS трафика

```
apiVersion: networking.istio.io/v1alpha3
```

```
kind: Gateway
```

```
metadata:
```

```
  name: sreda-gateway
```

```
  namespace: sreda-system
```

```
spec:
```

```
  selector:
```

```
    istio: ingressgateway
```

```
  servers:
```

```
    - port:
```

```
      number: 80
```

```
      name: http
```

```
      protocol: HTTP
```

```
    hosts:
```

```
      - "sreda.domain.tld"
```

```
      - "dex.sreda.domain.tld"
```

```
    tls:
```

```
      httpsRedirect: true
```

```
    - port:
```



ООО «Среда Разработки»
ИНН/КПП 7731343398/ 772801001
ОГРН 1177746046895

117437, г. Москва, вн.тер.г. Муниципальный округ Коньково,
ул. Академика Арцимовича, д. 17, помещ. 3/1

number: 443

name: https

protocol: HTTPS

hosts:

- "sreda.domain.tld"

- "dex.sreda.domain.tld"

tls:

mode: SIMPLE

credentialName: sreda-tls-cert

VirtualService для Dex

apiVersion: networking.istio.io/v1alpha3

kind: VirtualService

metadata:

name: dex-virtualservice

namespace: sreda-system

spec:

hosts:

- "dex.sreda.domain.tld"

gateways:

- sreda-gateway

http:

- route:

- destination:

host: dex.sreda-system.svc.cluster.local

port:

number: 5556



VirtualService: /login/ → AuthService, остальное → Central Dashboard

apiVersion: networking.istio.io/v1alpha3

kind: VirtualService

metadata:

name: platform-virtualservice

namespace: sreda-system

spec:

hosts:

- "sreda.domain.tld"

gateways:

- sreda-gateway

http:

- match:

- uri:

- prefix: /login/

route:

- destination:

- host: oidc-authservice.sreda-system.svc.cluster.local

- port:

- number: 8080

- route:

- destination:

- host: centraldashboard.sreda-system.svc.cluster.local

- port:

- number: 80

kubectl apply -f routing-manifests.yaml



Шаг 2. Создать файл `auth-policy.yaml` - `AuthorizationPolicy`:

```
apiVersion: security.istio.io/v1beta1
```

```
kind: AuthorizationPolicy
```

```
metadata:
```

```
  name: oidc-protect
```

```
  namespace: istio-system
```

```
spec:
```

```
  selector:
```

```
    matchLabels:
```

```
      istio: ingressgateway
```

```
  action: CUSTOM
```

```
  provider:
```

```
    name: oidc-authservice-provider
```

```
  rules:
```

```
    - to:
```

```
      - operation:
```

```
        hosts:
```

```
          - "sreda.domain.tld"
```

```
        notPaths:
```

```
          - /login/*
```

```
kubectl apply -f auth-policy.yaml
```

`AuthorizationPolicy` с `action: CUSTOM` требует, чтобы в `meshConfig.extensionProviders` `istiod` был зарегистрирован провайдер `oidc-authservice-provider` (настраивается в `istiod-values.yaml`, см. выше). Без этой конфигурации политика не вступит в силу.



3.2.2 Бэкенд-сервисы

Бэкенд SredaFlowAi устанавливается через внутренние Helm-чарты проекта. Перед установкой необходимо создать Secrets с параметрами подключения к БД и объектному хранилищу.

Шаг 1. Создать Secret для подключения к MySQL:

```
kubectcl create secret generic sreda-db-secret \  
--from-literal=DB_PASSWORD=<db_password> \  
-n sreda-system
```

Шаг 2. Создать Secret для подключения к MinIO:

```
kubectcl create secret generic minio-sreda-secret \  
--from-literal=MINIO_SECRET_KEY=<secret_key> \  
-n sreda-system
```

Шаг 3. Подготовить файлы значений для каждого сервиса:

Ниже перечислены примеры файлов, которые необходимо создать перед установкой. Файлы создаются под конкретное окружение.

Файл значений	Сервис	Минимальный состав параметров
sredaflow-values.yaml	Central Dashboard + Platform Core	Домен платформы, namespace, включаемые модули
pipeline-values.yaml	Pipelines Backend + Scheduler	Параметры MySQL, MinIO, лимиты ресурсов
kserve-values.yaml	KServe (инференс моделей)	Ingress Gateway, namespace, StorageClass
training-operator-values.yaml	Training Operator	Namespace, лимиты ресурсов



Файл значений	Сервис	Минимальный состав параметров
spark-values.yaml	Spark Operator	Namespace, настройки Spark History Server

Шаг 4. Установить бэкенд-сервисы:

Основной чарт платформы (Central Dashboard, Profile Controller, Notebook Controller)

```
helm install sredaflow ./charts/sredaflow-*.tgz \
--namespace sreda-system \
--values sredaflow-values.yaml
```

Сервис пайплайнов

```
helm install pipeline ./charts/pipeline-*.tgz \
--namespace sreda-system \
--values pipeline-values.yaml
```

Оператор распределенного обучения

```
helm install training-operator ./charts/training-operator-*.tgz \
--namespace sreda-system \
--values training-operator-values.yaml
```

Spark Operator

```
helm install spark ./charts/spark-*.tgz \
--namespace sreda-system \
--values spark-values.yaml
```

KServe (инференс-эндпоинты)

```
helm install kserve ./charts/kserve-*.tgz \
```



```
--namespace sreda-system \  
--values kserve-values.yaml
```

Путь ./charts/ передает путь к ранее переданным чартам. Чарты передаются архивом.

Шаг 5. Проверить статус всех подов:

```
kubectl get pods -n sreda-system
```

Ожидаемый результат: все поды в состоянии Running или Completed, статусы CrashLoopBackOff / Error отсутствуют.

Шаг 6. Проверить журналы основного сервиса:

```
kubectl logs -n sreda-system -l app=ml-pipeline --tail=100
```

3.2.3 Фронтенд-сервис

Фронтенд входит в состав чарта sredaflow-*.tgz (установленного в разделе 3.2.2) - отдельная установка не требуется.

Убедиться, что сервис centraldashboard существует

```
kubectl get svc centraldashboard -n sreda-system
```

После подтверждения, что сервис включен в sredaflow-*.tgz, выполните проверку работоспособности пода:

```
kubectl get pods -n sreda-system -l app=centraldashboard
```



4 ПРОВЕРКА РАЗВЕРТЫВАНИЯ

4.1 Проверка состояния подов

Все компоненты платформы

```
kubectl get pods -n sreda-system
```

Компоненты Istio

```
kubectl get pods -n istio-system
```

cert-manager

```
kubectl get pods -n cert-manager
```

Ожидаемый результат: все поды в состоянии Running или Completed.

4.2 Проверка TLS-сертификатов

```
kubectl get certificate -n istio-system
```

Поле READY должно быть True.

4.3 Проверка внешнего IP Ingress Gateway

```
kubectl get svc istio-ingressgateway -n istio-system
```

Поле EXTERNAL-IP должно содержать IP-адрес (или DNS-имя при использовании LoadBalancer облачного провайдера). Этот адрес необходимо прописать в DNS для sreda.domain.tld и dex.sreda.domain.tld.



4.4 Функциональная проверка

№	Проверка	Ожидаемый результат
1	Открыть https://sreda.domain.tld в браузере	Редирект на страницу авторизации Dex
2	Войти с учетными данными администратора	Успешный вход, переход на Central Dashboard
3	Создать Jupyter-блокнот	Блокнот запускается и доступен
4	Запустить тестовый пайплайн	Пайплайн выполняется, артефакты сохраняются в MinIO
5	Проверить доступность MinIO Console	Раздел MinIO или прямой порт 9001



5 ТИПОВЫЕ ОШИБКИ И ИХ УСТРАНЕНИЕ

Ошибка	Возможная причина	Способ устранения
Pod в статусе ImagePullBackOff	Недоступен реестр образов	kubectl describe pod <pod_name> -n sreda-system - проверить URL образа и imagePullSecret
Pod в статусе CrashLoopBackOff	Ошибка конфигурации или недоступна БД	kubectl logs <pod_name> -n sreda-system --previous
certificate в статусе READY=False	cert-manager не может выпустить сертификат	kubectl describe certificate sreda-tls-cert -n istio-system - проверить события
Браузер возвращает 503	Istio VirtualService не настроен или под недоступен	kubectl get vs -n sreda-system, проверить endpoint сервиса centraldashboard
Ошибка 401 / редиректа нет	AuthorizationPolicy не активна (нет ExtensionProvider в meshConfig)	Проверить meshConfig.extensionProviders в istiod-values.yaml, затем helm upgrade istiod ...
helm install: already exists	Компонент уже установлен	Использовать helm upgrade вместо helm install
PVC в статусе Pending	Нет подходящего StorageClass	kubectl get sc - проверить наличие и имя StorageClass; скорректировать storageClass в values-файлах



6 ОБНОВЛЕНИЕ ПЛАТФОРМЫ

6.1 Обновление Helm-релиза

Обновить репозитории

```
helm repo update
```

Обновить конкретный релиз (пример для MySQL)

```
helm upgrade mysql bitnami/mysql \  
--namespace sreda-system \  
-f mysql-values.yaml
```

Обновить внутренний чарт (пример для основного сервиса)

```
helm upgrade sredaflow ./charts/sredaflow-*.tgz \  
--namespace sreda-system \  
--values sredaflow-values.yaml
```

6.2 Откат релиза

Просмотреть историю релиза

```
helm history sredaflow -n sreda-system
```

Откатить к предыдущей версии

```
helm rollback sredaflow <revision> -n sreda-system
```

6.3 Резервное копирование перед обновлением

Перед каждым обновлением создать резервную копию:

Дамп MySQL

```
kubectrl exec -n sreda-system deploy/mysql -- \  
--
```



```
mysqldump -u root -p<root_password> --all-databases > sreda_backup_$(date  
+%Y%m%d).sql
```

Экспорт Helm values всех релизов

```
for release in mysql minio dex oidc-authservice istio-base istiod istio-ingressgateway  
cert-manager sredaflow pipeline kserve training-operator spark; do
```

```
helm get values $release -n sreda-system > helm_values_${release}_$(date  
+%Y%m%d).yaml 2>/dev/null || true
```

```
done
```

Экспорт Secrets

```
kubectl get secrets -n sreda-system -o yaml > secrets_backup_$(date  
+%Y%m%d).yaml
```