ПРОГРАММНАЯ ЧАСТЬ ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА

«ЭВИРИС»

РУКОВОДСТВО ПО УСТАНОВКЕ И ЭКСПЛУАТАЦИИ

СОДЕРЖАНИЕ

1. Общие сведения 3

2. Установка инфраструктуры 4

2.1. Установка виртуализации Astra Linux 4
2.2. Установка Kubernetes (Deckhouse) 4
2.3. Установка персистентного хранилища (Longhorn) 5
2.4. Установка Docker Registry 6
2.5. Установка S3 MinIO 6

3. Установка и настройка компонентов контура API-GW 8

3.1. Подготовительные шаги 8
3.2. Установка PostgreSQL 9
3.3. Установка HAProxy 9
3.4. Установка Redis 9
3.5. Установка Kafka (с использованием Strimzi Operator) 10
3.6. Установка модулей API-GW 10
3.6.1. dbms-interaction-module 10
3.6.2. auth-module 10
3.6.3. backend-module 11
3.6.4. command-processing-module 11
3.6.5. sse-module 12
3.6.6. proxy-module 12
3.7. Настройка HAProxy для балансировки нагрузки 13

4. Установка и настройка компонентов контура CORE 14

Подготовительные шаги 14
4.1. Настройка NFS-сервера 14
4.2. Установка HAProxy и Keepalived 15
4.3. Установка Hive 16
4.4. Установка Redis 16
4.5. Установка Spark History Server 16
4.6. Установка Livy 17
4.7. Установка Fluent-bit 17
4.8. Установка receiving-control-module 18
4.9. Установка command-orchestrator 18

5. Установка и настройка компонентов контура USER 19

Подготовительные шаги 19

5.1. Установка frontend 19
5.2. Установка preparation-module.....19

1. Общие сведения

Программная часть программно-аппаратного комплекса «Эвирис» (далее – программа) предназначена для автоматизированного обучения (AutoML) и последующего применения моделей машинного обучения. При этом пользователи на всех этапах использования программы имеют доступ исключительно к структуре данных и метаданным, не имея доступа непосредственно к наборам данных, используемых для машинного обучения и применения ML-моделей.

Пользователи взаимодействуют с программой через интерфейс (UI). Описание доступных действий пользователя приведены в документе «Руководство пользователя».

Описание основных доступных действий администратора, требования к программному обеспечению приведены в документе «Руководство администратора».

Для начала работы администратора в программе предварительно должны быть выполнены действия по развертыванию программы, описанные в данном документе.

2. Установка инфраструктуры

2.1. Установка виртуализации Astra Linux

Для установки гипервизора необходимо выполнить следующие команды:

sudo apt install astra-kvm ovmf gir1.2-spiceclientgtk-3.0

sudo adduser имя_пользователя libvirt-admin

После этого можно использовать графический интерфейс для работы с виртуальными машинами.

2.2. Установка Kubernetes (Deckhouse)

Для установки Kubernetes (Deckhouse) необходимо выполнить действия согласно официальной инструкции: https://deckhouse.ru/products/kubernetes-platform/gs/bm/step2.html

Пример сгенерированного конфига Deckhouse лежит в каталоге проекта, каталог Deckhouse.

Необходимо выполнить шаги до пункта: «Создайте NodeGroup worker» — данный пункт выполняем.

После установки вы получаете одну master-ноду. Далее необходимо увеличить количество master- и worker-нод до нужного количества.

1. Для добавления новой master-ноды: на имеющейся master-ноде необходимо выполнить следующую команду:
   d8 k -n d8-cloud-instance-manager get secret manual-bootstrap-for-master -o json | jq '.data."bootstrap.sh"' -r
   Результатом вывода будет зашифрованный в base64 скрипт для установки ноды.
2. Скопировать вывод консоли и перенести его на целевую ноду, которую необходимо добавить, как master-ноду.
3. На целевой ноде необходимо выполнить:
   cat script | base64 -d > script.sh
   chmod +x ./script.sh
   ./script.sh
4. Для проверки корректности добавленной ноды на первом мастере необходимо выполнить следующую команду:
   kubectl get node
   Если команда kubectl вернула ошибку, необходимо проверить наличие контекста подключения к кластеру.

Для добавления worker-ноды необходимо выполнить порядок действий аналогичный порядку добавления master-ноды, за исключением следующей команды:

d8 k -n d8-cloud-instance-manager get secret manual-bootstrap-for-worker -o json | jq '.data."bootstrap.sh"' -r

В проекте используются три независимых кластера Kubernetes: CORE, API-GW, USER (описание их установки приведено в разделах ниже настоящего документа).

2.3. Установка персистентного хранилища (Longhorn)

В проекте используются Redis и Kafka, которым необходимо сохранять свое состояние. Для хранения выбран Longhorn.

После инициализации кластеров K8S необходимо установить Longhorn в кластеры CORE и API-GW.

В каталоге проекта директории core и api_gw содержатся Helm-чарты для установки.

Порядок установки:

1. В рамках подготовки необходимо скопировать каталог проекта на ноду, которая имеет доступ к K8S.
2. Далее необходимо перейти в каталог: cd core/longhorn и на всех worker-нодах установить open-iscsi:
   sudo apt install open-iscsi
3. Создать namespace в K8S:
   kubectl create ns longhorn-system
4. Добавить лейблы безопасности:
   *kubectl label namespace longhorn-system *
   *pod-security.kubernetes.io/enforce=privileged *
   *pod-security.kubernetes.io/audit=privileged *
   *pod-security.kubernetes.io/warn=privileged *
   security.deckhouse.io/pod-policy-action=warn --overwrite
5. Установить Helm-чарт:
   helm install longhorn . -n longhorn-system -f values.yaml
6. После установки удалить storageclass, созданный Longhorn:
   kubectl delete storageclasses longhorn-static
7. Применить свой storageclass из файла storage.yaml:
   kubectl create -f storage.yaml
8. В файле storage.yaml изменить параметр numberOfReplicas: "2" так, чтобы он соответствовал количеству worker-нод.

Порядок установки Longhorn для кластеров CORE и API-GW одинаковый..

2.4. Установка Docker Registry

Так как контур предполагается закрытым, необходимо установить локальное хранилище Docker-образов. Требуется установить по одному registry в каждый контур: CORE, API-GW, USER.

Порядок установки:

1. Скопируйте каталог docker-registry на сервер, где планируется развернуть registry.
2. Создайте структуру каталогов:
   mkdir -p {data/registry,auth,certs}
3. Создайте файл с логином и паролем:
   *docker run --rm *
   *--entrypoint htpasswd *
   httpd:2 -Bbn username password > auth/htpasswd
4. Создайте самоподписанные сертификаты для registry. Для этого используйте скрипт script-cert.sh.
5. В переменной REGISTRY_HOST укажите IP-адрес или доменное имя сервера.
6. Запустите registry:
   docker-compose up -d
   Веб-интерфейс будет доступен на порту 80 данного сервера.
7. При необходимости очистите образы:
   docker prune -a

Порядок установки registry в контурах CORE, API-GW, USER одинаковый.

2.5. Установка S3 MinIO

Для работы MinIO необходимо подготовить диски — отформатировать и смонтировать их в системе.

MinIO запускается в кластерном режиме на двух нодах. Используются два docker-compose файла: один для первой ноды, второй — для второй.

Порядок установки:

1. Скопировать файл docker-compose-node-1.yaml на первую ноду.
2. Скопировать файл docker-compose-node-2.yaml на вторую ноду.
3. В каталоге с файлом выполнить:
   docker-compose up -d

Web-интерфейс MinIO будет доступен на каждой ноде на порту, указанном в директиве --console-address в docker-compose файле.

Логин и пароль задаются через переменные окружения в docker-compose.

4. Установить модуль валидации данных (модуль расположен в каталоге system/minio/data_valid_cont):
    a. Отредактировать файл config.yaml, указав:
        - адрес и креды MinIO (minio);
        - адрес Kafka (kafka);
        - адрес Hive (hive);
    b. Запустить модуль:
    docker-compose up -d

3. Установка и настройка компонентов контура API-GW

3.1. Подготовительные шаги

Перед началом установки компонентов контура API-GW необходимо выполнить следующие подготовительные шаги:

1. Развернуть локальный Docker Registry.
2. Загрузить необходимые образы в registry.
3. Настроить Kubernetes-кластер.
4. Создать Namespace для API-GW.
5. Настроить доступ к registry из Kubernetes.
6. Установить PostgreSQL и HAProxy.

Шаг 1: Загрузка образов в Docker Registry

Для загрузки образов в Docker Registry необходимо:

1. Скопировать каталог api-gw на сервер, где установлен локальный Docker Registry.
2. Загрузить образы командой:

   docker load -i имя_файла_образа
3. Авторизоваться в registry:

   docker login <адрес_registry> -u <логин> -p <пароль>
4. Залить образы в registry:

   docker push <тег_образа>

Шаг 2: Создание Namespace и Secret

Для создания namespace необходимо выполнить команду:

kubectl create ns api-gw

Для создания secret для доступа к registry необходимо выполнить команду:

*kubectl create secret docker-registry docker-registry *

*--docker-server=<адрес_registry> *

*--docker-username=<логин> *

--docker-password=<пароль>

Шаг 3: Настройка доверия сертификатам на worker-нодах

Для настройки доверия сертификатам на worker-нодах необходимо:

1. Скопировать сертификат registry.crt на все worker-ноды:

scp certs/registry.crt administrator@<ip_worker_node>

2. На каждой worker-ноде выполнить:

   sudo cp registry.crt /usr/local/share/ca-certificates/

   sudo update-ca-certificates

   sudo reboot

Шаг 4: Установка PostgreSQL

Для установки PostgreSQL необходимо:

1. Установить PostgreSQL:

   apt update && apt install postgresql

2. Переключиться на пользователя postgres:

   su - postgres

3. Создать базу данных:

   psql

   CREATE DATABASE anklav_management;

   \q

Шаг 5: Установка HAProxy

Для установки HAProxy необходимо выполнить команду:

apt update && apt install haproxy

3.2. Установка PostgreSQL

Описание порядка установки PostgreSQL приведено выше в пункте «Подготовительные шаги», шаг 4 «Установка PostgreSQL» настоящего документа.

3.3. Установка HAProxy

Описание порядка установки HAProxy приведено выше в пункте «Подготовительные шаги», шаг 5 «Установка HAProxy» настоящего документа.

3.4. Установка Redis

Для установки Redis необходимо:

1. Перейти в каталог:

   cd api_gw/redis

2. Установить Redis через Helm с помощью команды:

   helm upgrade --install redis . -n redis --create-namespace -f values.yaml

3.5. Установка Kafka (с использованием Strimzi Operator)

Для установки Kafka необходимо:

1. Установить Strimzi Operator с помощью команды:
   cd api_gw/kafka-cluster/strimzi-kafka-operator
   helm upgrade --install kafka-operator . -n kafka --create-namespace -f values.yaml
2. Применить конфигурацию кластера:
   cd api_gw/kafka-cluster/config
   kubectl apply -f kafka-cluster.yaml

3.6. Установка модулей API-GW

3.6.1. dbms-interaction-module

Для установки модуля dbms-interaction-module необходимо:

1. Перейти в каталог:
   cd api_gw/dbms-interaction-module/helm
2. В файле values.yaml задать переменные:
   - repository — адрес образа;
   - tag — версия тега;
   - imagePullSecrets — имя секрета;
   - env — переменные окружения:
     - DATABASE_USERNAME - имя пользователя БД;
     - DATABASE_PASSWORD - пароль пользователя БД;
     - DATABASE_NAME - имя базы данных;
3. Установить модуль с помощью команды:
   helm upgrade --install dbms-interaction-module . -n api-gw -f values.yaml

3.6.2. auth-module

1. Перейдите в каталог:
   cd api_gw/auth-module/helm
2. В файле values.yaml укажите:
   - repository — адрес образа
   - tag — версию тега
   - imagePullSecrets — имя секрета
   - env — переменные окружения:

- REDIS_SENTINEL_HOSTS (Адрес Redis)

3. Установите модуль:

   helm upgrade --install auth-module . -n api-gw -f values.yaml

3.6.3. backend-module

1. Перейдите в каталог:

   cd api_gw/backend/helm

2. В файле values.yaml укажите:

- repository — адрес образа
- tag — версию тега
- imagePullSecrets — имя секрета
- env — переменные окружения:
  - DBMS_HOST (Хост dbms модуля)
  - DBMS_PORT (Порт dbms)
  - DMBS_INTERCATION_MODULE (Адрес dbms)

3. Установите модуль:

   helm upgrade --install backend-module . -n api-gw -f values.yaml

3.6.4. command-processing-module

1. Перейдите в каталог:

   cd api_gw/command-processing-module/helm

2. В файле values.yaml укажите:

- repository — адрес образа
- tag — версию тега
- imagePullSecrets — имя секрета
- env — переменные окружения:
  - KAFKA_BROKER (Адрес Kafka)
  - DBMS_ADDRESS (Адрес dbms модуля)
  - REDIS_SENTINEL_SERVER1 (Адрес Redis)
  - REDIS_SENTINEL_SERVER2 (Адрес Redis)
  - REDIS_SENTINEL_SERVER3 (Адрес Redis)
  - REDIS_SENTINEL_MASTER (Имя мастера Redis)

3. Установите модуль:

   helm upgrade --install command-processing-module . -n api-gw -f values.yaml

3.6.5. sse-module

1. Перейдите в каталог:

   cd api_gw/sse-module/helm
2. В файле values.yaml укажите:

- repository — адрес образа
- tag — версию тега
- imagePullSecrets — имя секрета
- env — переменные окружения:
    - REDIS_SENTINEL_HOSTS (Адрес Redis)
    - REDIS_SENTINEL_SERVICE_NAME (Имя мастера Redis)
    - REDIS_DB (Основная БД Redis)
    - REDIS_SESSION_DB (Бд для хранения сессий в Redis)

3. Установите модуль:

   helm upgrade --install sse-module . -n api-gw -f values.yaml

3.6.6. proxy-module

1. Перейдите в каталог:

   cd api_gw/nginx/helm
2. В файле values.yaml укажите:
    - repository — адрес образа
    - tag — версию тега
    - imagePullSecrets — имя секрета
    - upstream — адреса модулей:
        - backend_module
        - auth_module
        - execution_module
        - data_upload_module
        - command_processing_module
        - sse_module
        - dbms_module
3. Установите модуль:

   helm upgrade --install proxy-module . -n api-gw -f values.yaml

3.7. Настройка HAProxy для балансировки нагрузки

Для настройки HAProxy для балансировки нагрузки необходимо:

1. Открыть конфигурационный файл:
   vi /etc/haproxy/haproxy.cfg
2. Добавьте конфигурацию из файла api_gw/haproxy/haproxy.cfg.
3. Измените IP-адреса для балансировки:

- Kafka;
- Proxy Module;
- MinIO UI.

4. Перезапустить HAProxy с помощью команды:
   systemctl restart haproxy

4. Установка и настройка компонентов контура CORE

Подготовительные шаги

Перед началом установки компонентов контура CORE необходимо выполнить следующие подготовительные шаги:

1. Развернуть локальный Docker Registry, аналогично тому, как это было сделано для контура API-GW (см. п. 2.4. «Установка локального Docker Registry» настоящего документа).
2. Добавить секреты и сертификат (см. п 3.1. «Подготовительные шаги», шаги 1-3 настоящего документа).

4.1. Настройка NFS-сервера

Для хранения логов Spark необходимо развернуть NFS-сервер. Перед началом установки необходимо убедиться, что Docker Registry установлен и Secret добавлен в Kubernetes (аналогично API-GW).

Для установки NFS-сервера необходимо:

1. Обновить систему с помощью команды:
   apt update && apt upgrade
2. Установить NFS-сервер с помощью команды:
   apt install nfs-kernel-server
3. Создать каталог для общего доступа:
   mkdir -p /opt/nfs
4. Отредактировать файл экспорта:
   vi /etc/exports
5. Добавить строку:
   */opt/nfs *(rw,sync,no_subtree_check,no_root_squash)*
6. Применить изменения:
   exportfs -ra
7. Перезапустить NFS-службу:
   systemctl restart nfs-kernel-server
8. Включить автозапуск службы:
   systemctl enable nfs-kernel-server
9. Проверить доступность каталога:

showmount -e <ip_адрес_текущего_NFS_сервера>

10. Создать структуру под Spark-логи:
    mkdir /opt/nfs/spark-logs
11. Создать группу и пользователей:
    groupadd sharedspark
    useradd --system --uid=199 --gid=sharedspark livy
    useradd --system --uid=185 --gid=sharedspark spark
12. Назначить права:
    chgrp sharedspark /opt/nfs/spark-logs/
    chmod -R 770 /opt/nfs/spark-logs/
13. Скопировать скрипт очистки логов clean-log.sh из каталога проекта в /root и сделать его исполняемым:
    chmod +x /root/clean-log.sh
14. Добавить задачу в crontab для ежечасной очистки:
    crontab -e
15. Добавить строку:
    **/30 * * * * /root/clean-log.sh*

4.2. Установка HAProxy и Keepalived

HAProxy используется для балансировки трафика между нодами MinIO, Keepalived и обеспечивает отказоустойчивость.

Для установки HAProxy и Keepalived необходимо:

1. Установить пакеты с помощью команды:
   apt update && apt install haproxy keepalived
2. Скопировать конфигурационные файлы из каталога core/haproxy.
3. Настроить haproxy.cfg:
   a. Указать IP-адреса MinIO-нод.
   b. Настроить keepalived.conf.
   c. Указать общий VIP-адрес.
4. Выполнить запуск:
   a. Перезапустить HAProxy:
      systemctl restart haproxy
   b. Убедиться, что Keepalived запущен:
      systemctl status keepalived

Для корректной работы необходимо минимум две ноды HAProxy.

4.3. Установка Hive

Hive обеспечивает работу с данными в хранилище MinIO.

Для установки Hive необходимо:

1. Перейти в каталог:

   cd core/hive
2. В файле values.yaml указать:
   - *repository — адрес образа;*
   - *tag — версия тега;*
   - *imagePullSecrets — имя секрета;*
   - *s3 — адрес MinIO и креды для подключения.*
3. Установить модуль:

   helm upgrade --install hive . -n hive --create-namespace -f values.yaml

4.4. Установка Redis

Redis используется для кэширования данных.

Для установки Redis необходимо:

1. Перейти в каталог:

   cd core/redis
2. Установить Redis через Helm:

   helm upgrade --install redis . -n redis --create-namespace -f values.yaml

4.5. Установка Spark History Server

Spark History Server позволяет просматривать историю выполненных заданий.

Для установки Spark History Server необходимо:

1. Перейдите в каталог:

   cd core/spark-history-server
2. В файле values.yaml укажите:
   - *repository — адрес образа*
   - *tag — версию тега*
   - *imagePullSecrets — имя секрета*
   - *storage.pv.nfs — IP-адрес NFS-сервера*
3. Установите модуль:

helm upgrade --install spark-history . -n spark --create-namespace -f values.yaml

4.6. Установка Livy

Livy обеспечивает REST-интерфейс для взаимодействия с Apache Spark.

Для установки Livy необходимо:

1. Перейти в каталог:
   cd core/livy
2. В файле values.yaml указать:
   - *repository — адрес образа;*
   - *tag — версию тега;*
   - *imagePullSecrets — имя секрета;*
   - *env — переменные окружения:*
     - *SPARK_CONF_spark_kubernetes_container_image (адрес образа spark);*
     - *SPARK_CONF_spark_kubernetes_container_image_pullSecrets (cСекрет k8s).*
3. Установить модуль:
   helm upgrade --install livy . -n spark -f values.yaml

4.7. Установка Fluent-bit

Fluent-bit отвечает за сбор и отправку логов.

Для установки Fluent-bit необходимо:

1. Перейти в каталог:
   cd core/fluent-bit
2. В файле values.yaml указать:
   nfsPvPvc — IP-адрес NFS-сервера и путь хранения логов
3. Добавить namespace и лейблы безопасности:
   kubectl create ns fluent-bit
   *kubectl label namespace fluent-bit *
   *pod-security.kubernetes.io/enforce=privileged *
   *pod-security.kubernetes.io/audit=privileged *
   *pod-security.kubernetes.io/warn=privileged *
   security.deckhouse.io/pod-policy-action=warn --overwrite
4. Установите модуль:
   helm upgrade --install fluent-bit . -n fluent-bit --create-namespace -f values.yaml

4.8. Установка receiving-control-module

Модуль предназначен для управления приемом и контролем данных.

Для установки модуля receiving-control-module необходимо:

1. Перейти в каталог:
   cd core/receiving-control-module
2. В файле values.yaml указать:
   - *repository — адрес образа;*
   - *tag — версия тега;*
   - *imagePullSecrets — имя секрета;*
   - *config.kafka — адрес Kafka;*
   - *config.redis — имя мастера и адреса Redis;*
   - *config.minio — адрес и креды MinIO;*
   - *config.orchestrator — адрес оркестратора;*
   - *config.hive — адрес Hive.*
3. Установить модуль:
   helm upgrade --install receiving-control-module . -n spark -f values.yaml

4.9. Установка command-orchestrator

Модуль Command Orchestrator предназначеен для управления выполнением команд в системе.

Для установки модуля command-orchestrator необходимо:

1. Перейти в каталог:
   cd core/command-orchestrator
2. В файле values.yaml указать:
   - *repository — адрес образа;*
   - *tag — версию тега;*
   - *imagePullSecrets — имя секрета;*
   - *config.livy.url — адрес Livy;*
   - *config.images — образы для соответствующих команд;*
   - *config.images.storage.redis — имя мастера и адреса Redis;*
   - *config.images.storage.core_service_url — адрес receiving-control-module.*
3. Установить модуль:
   helm upgrade --install command-orchestrator . -n spark -f values.yaml

5. Установка и настройка компонентов контура USER

Подготовительные шаги

Перед началом установки компонентов контура USER необходимо выполнить следующие подготовительные шаги:

1. Развернуть локальный Docker Registry, аналогично тому, как это было сделано для контуров CORE и API-GW (см. п. 2.4. «Установка локального Docker Registry» настоящего документа).
2. Добавить секреты и сертификат (см. п 3.1. «Подготовительные шаги», шаги 1-3 настоящего документа).

5.1. Установка frontend

Модуль frontend обеспечивает пользовательский интерфейс системы.

Для установки модуля frontend необходимо:

1. Перейти в каталог:
   cd user/frontend
2. В файле values.yaml указать:
   - *repository — адрес образа;*
   - *tag — версию тега;*
   - *imagePullSecrets — имя секрета;*
   - *nginx.locations — адреса сервисов:*
     - *proxy-module (gateway);*
     - *preparation-module (test).*
3. Установить модуль:
   helm upgrade --install frontend . -n user --create-namespace -f values.yaml

5.2. Установка preparation-module

Модуль preparation-module отвечает за подготовку команд перед их выполнением.

Для установки модуля preparation-module необходимо:

1. Перейти в каталог:
   cd user/preparation-module
2. В файле values.yaml указать:
   - *repository — адрес образа;*

- *tag — версия тега;*
- *imagePullSecrets — имя секрета;*
- *env — переменные окружения:*
  - *AUTHENTICATION_URL — адрес proxy-module (часть URL);*
  - *COMMAND_TEMPLATES_URL — адрес proxy-module (часть URL);*
  - *COMMAND_EXECUTOR_URL — адрес proxy-module (часть URL).*

3. Установить модуль:

   helm upgrade --install preparation-module . -n user -f values.yaml