

**ИНСТРУКЦИЯ ПО УСТАНОВКЕ ЭКЗЕМПЛЯРА
ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
«IN DATA ANALYSIS PLATFORM 4.0»
МОДУЛЬ «IN DATA ANALYSIS PLATFORM INDICATORS»**

АННОТАЦИЯ

Данный документ предназначен для специалистов, выполняющих администрирование программного продукта «In Data Analysis Platform 4.0», (далее – In-DAP 4.0), и включает описание действий по установке и настройке модуля In Data Analysis Platform Indicators.

СОДЕРЖАНИЕ

1. Состав In Data Analysis Platform Indicators	4
2. Варианты установки	4
3. Необходимая инфраструктура	4
3.1. Настройка инфраструктуры.....	5
3.1.1. Установка сервера Модуля ИБ:.....	5
3.1.1.1. Установка из дистрибутива:.....	5
3.1.1.2. Установка из контейнера:.....	7
3.1.2. Установка Сервера Postgresql 14	7
3.1.3. Установка Сервера nfs.....	7
3.1.4. Установка Сервера Minio	7
4. Установка и настройка сервисов	12
4.1. Описание манифестов сервисов kubernetes	12

1. Состав In Data Analysis Platform Indicators

In Data Analysis Platform Indicators (далее – In-DAP Indicators) состоит из следующих сервисов:

Backend

- attributeweb service
- documentvalueweb service
- gatewayintegrationapi
- gatewaywebapi
- hierarchyweb service
- historicaldataweb service
- importweb service
- indicatorvalue s
- indicatorweb service
- lifecycletemplateweb service
- menuweb service
- monitoringservices
- objectvalue s
- objectweb service
- permissionweb service
- sendingemailweb service
- taskscheduler
- p gadmin

Frontend

- frontend

Утилита для генерации базы данных

- postgresqlfdatagenerator

2. Варианты установки

Так как приложение состоит из микросервисов, то запуск необходимо осуществлять посредством старта через инструментарий docker-compose, либо через оркестратора контейнеров (docker swarm, nomad, mesos, kubernetes).

3. Необходимая инфраструктура

Для функционирования системы необходима следующая инфраструктура:

- Сервер модуля ИБ;
- Сервер Postgresql 14;
- Сервер NFS;
- Сервер Minio;

3.1. Настройка инфраструктуры

3.1.1. Установка сервера Модуля ИБ:

Сервер Модуля ИБ устанавливается из дистрибутива либо посредством запуска в контейнере.

3.1.1.1. Установка из дистрибутива:

Инициализация Модуля ИБ происходит в процессе установки, во время которой указываются:

- логин и пароль администратора Модуля ИБ;
- параметры подключения к базе данных (адрес сервера базы данных, имя базы данных, имя пользователя базы данных, пароль пользователя базы данных).

Перед началом установки Модуля ИБ необходимо убедиться, что развёрнуты:

- Пакеты Java JDK 8;
- Сервер приложений WildFly 20.0.1;
- PostgreSQL PRO 11 с заведёнными учётной записью пользователя и базой данных.

Установка Модуля ИБ состоит из следующих шагов:

- Установка приложения ПО Модуля ИБ
- Настройка контроля целостности
- Запуск и проверка работы сервиса

Важно! При выполнении команд через терминал, необходимо использовать запуск от привилегированного пользователя с правами *sudo*.

Установка приложения ПО Модуля ИБ

Необходимо скопировать содержимое архива из дистрибутива модуля ИБ в корень каталога WildFly, при этом необходимо согласиться на все слияния папок и замену файлов. (Keycloak-overlay-11.0.2.Tar.gz). В результате будет папка со всеми файлами, необходимыми для Модуля ИБ.

Следующим шагом необходимо перейти в каталог `"/<адрес дистрибутива, в котором расположен корневой дистрибутив сервера приложений`

WildFly> / <Наименование дистрибутива сервера приложений Wildfly> /' и с правами sudo выполнить команду:

```
# ./iamc_setup.sh
```

В результате выполнения команды будет выполнена инициализация и регистрации модуля ИБ, добавления jdbc драйвера, DataSource. Так же будут настроены standalone, так и standalone-ha версии модуля.

***Примечание:** В случае повторного запуска скрипта или обновления существующей установки, возможны появления сообщений о дубликатах ресурсов.*

Переменные окружения, определённые в запускаемом скрипте, будут формировать настройки системы. Для этого необходимо отредактировать файл /дистрибутив, в котором расположен корневой дистрибутив WildFly/Наименование дистрибутива сервера приложений Wildfly/bin/standalone-ha.sh (либо standalone.sh, в зависимости от необходимого режима) заменив значения переменных на актуальные, отредактировав следующие строки:

```
DB_ADDR='адрес PostgreSQL сервера (если порт PostgreSQL сервера отличается от стандартного (5432), его следует указать через двоеточие)'
```

```
DB_DATABASE='имя БД'
```

```
DB_USER='логин учётной записи в БД'
```

```
DB_PASSWORD='пароль учётной записи для БД'
```

```
#DB_SCHEMA='наименование схемы' (заполняется в случае, если в СУБД используется схема)
```

```
#JDBC_PARAMS='настройки JDBC драйвера' (заполняется в случае, если используются специфичные настройки JDBC драйвера)
```

```
cgstmon_enable= Если true, включает модуль контроля целостности (При этом, в случае обнаружения изменений в файлах приложения, запуск сервиса будет остановлен!)
```

```
cgstmon_path= Путь к утилите контроля целостности cgstmon
```

```
cgstmon_log= Путь к файлу логов cgstmon. В логе фиксируется статус прохождения контроля целостности при последнем запуске.
```

```
PROXY_ADDRESS_FORWARDING='true/false' (требуется объявления при работе с обратным прокси, таким как nginx, это позволяет Модулю ИБ определять IP-адрес клиента из HTTP-заголовка)
```

```
BIND_ADDR= 'IP-адрес интерфейса, на котором должен работать Модуль ИБ'
```

```
SERVER_OPTS='-Djboss.bind.address=$BIND_ADDR -bmanagement $BIND_ADDR --server-config=standalone.xml' (При необходимости разнести Модуль ИБ и HAL консоль
```

WildFly по разным интерфейсам, следует явно указать IP-адрес. Параметр `jboss.bind.address` определяет адрес для Модуля ИБ, `bmanagement` – для HAL консоли)

Дополнительно при работе в режиме `standalone`-ha указываются параметры:

`jgroups.bind_addr` - *определяет IP-адрес для сервиса обмена сообщениями JGroup и по умолчанию равен `$BIND_ADDR`*

`jboss.tx.node.id` - *данный параметр позволяет уникально идентифицировать хост при работе с общей БД, должен быть уникален и по умолчанию инициализируется результатом выполнения команды `'hostname'`*

3.1.1.2. Установка из контейнера:

Необходимо запустить образ из поставки, используя сценарии в `/ansible/roles/kubernetes/templates/Services/keycloak`.

Либо в случае использования `kubernetes` запустив пайплайн `keycloak`.

Используются следующие переменные:

`keycloak.appname` – название сервиса

`keycloak.DB_ADDR` – адрес сервера бд

`keycloak.DB_DATABASE` – название ДБ

`keycloak.DB_USER` – пользователь ДБ

`keycloak.DB_PASSWORD` - пароль

`keycloak.HAL_ORIGIN` – dns имя сервера МИБ

`keycloak.KEYCLOAK_FRONTEND_URL` – полный url

3.1.2. Установка Сервера PostgreSQL 14

Осуществляется стандартная установка сервера.

Для доступа сервисов через `kubernetes` есть необходимый сценарий сервиса:

`ansible/roles/kubernetes/templates/postgresql/postgresql_service.yaml.j2`

Необходимо создать следующие бд:

- БД сервиса модуля In-DAP Indicators;
- БД модуля ИБ.

3.1.3. Установка Сервера nfs

Осуществляется стандартная установка сервера.

Необходимо создать `nfs` ресурс для `DAP Indicators`.

Для доступа сервисов через `kubernetes` есть необходимый сценарий сервиса:

`ansible/roles/kubernetes/templates/importwebservice/importwebservice_pv.yaml.j2`

`ansible/roles/kubernetes/templates/importwebservice/importwebservice_pvc.yaml.j2`

3.1.4. Установка Сервера Minio

Осуществляется стандартная установка сервера:

```
yum install wget
wget https://dl.min.io/server/minio/release/linux-amd64/minio
chmod +x minio
firewall-cmd --zone=public --add-port=9000/tcp --permanent
mkdir -p /opt/minio/data
groupadd minio
useradd minio -g minio
chown -R minio:minio /opt/minio
vi /etc/systemd/system/minio.service
```

[Unit]

Description=minio

Documentation=https://docs.minio.io

[Service]

Type=simple

WorkingDirectory=/opt/minio

User=minio

Group=minio

PermissionsStartOnly=true

ExecStart=/opt/minio/minio server /opt/minio/data --console-address ":9001"

Restart=on-success

StandardOutput=journal

StandardError=inherit

SyslogIdentifier=minio

Specifies the maximum file descriptor number that can be opened by this process

LimitNOFILE=65536

Disable timeout logic and wait until process is stopped

TimeoutStopSec=0

SIGTERM signal is used to stop Minio

KillSignal=SIGTERM

SendSIGKILL=no

SuccessExitStatus=0

[Install]

WantedBy=multi-user.target

```
mv minio /opt/minio/
```

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc
```

```
chmod +x mc
```

```
mv mc /opt/minio/
```

```
systemctl enable minio
```

```
systemctl daemon-reload
```

```
systemctl start minio
```

Если подключение происходит не через kubernetes, то следует установить nginx для использования сертификата.

Minio.conf (для nginx)

```
server {
```

```
listen 443 ssl;
```

```
server_name minio-localhost.sc-tatarstan.ru;
```

```
#change max size upload file as you want
```

```
client_max_body_size 500M;
```

```
ssl_certificate /opt/minio/certs/MinioFull.crt;
```

```
ssl_certificate_key /opt/minio/certs/private.key;
```

```
ssl_session_cache shared:SSL:1m;
```

```
ssl_session_timeout 10m;
```

```
ssl_ciphers HIGH:!aNULL:!MD5;
```

```
ssl_prefer_server_ciphers on;
```

```
access_log /var/log/nginx/minio-access-api.log;
```

```

error_log /var/log/nginx/minio-error-api.log;
location / {
proxy_pass http://10.70.35.42:9000;
include proxy_params;
}
}

server {
listen 443 ssl;
server_name minio-localhost-console.sc-tatarstan.ru;

#change max size upload file as you want
client_max_body_size 500M;

ssl_certificate /opt/minio/certs/MinioFull.crt;
ssl_certificate_key /opt/minio/certs/private.key;
ssl_session_cache shared:SSL:1m;
ssl_session_timeout 10m;
ssl_ciphers HIGH:!aNULL:!MD5;
ssl_prefer_server_ciphers on;
access_log /var/log/nginx/minio-access-console.log;
error_log /var/log/nginx/minio-error-console.log;
location / {
proxy_pass http://10.70.35.42:9001;
include proxy_params;
}
}

```

После развертывания контейнера Minio необходимо выполнить следующую настройку:

- Создать бакеты:

```
stscmon.initialbucket
```

```
stscmon.mainbucket
```

Для >21.3 названия бакетов прописаны в секции MinioClientSettings в all.yml

InitialBucketName - наименование начального бакета, с которым работает фронт и stscmonlib.

MainBucketName - наименование рабочего бакета, с которым работает интеграционный сервис In-DAP Indicators.

- Создать policy на каждый бакет.

На текущий момент policy дает полные права на бакет, необходимо ограничить в правах readwrite_stscmon_initialbucket.

Названия policy должны быть равны ролям в клиенте stscmon_minio_client. Допустимо другое наименование policy, однако есть проблема с маппингом ролей в keycloak.

readwrite_stscmon_initialbucket

readwrite_stscmon_initialbucket

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:PutObject",
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::stscmon.initialbucket/*"
      ]
    }
  ]
}
```

readwrite_stscmon_mainbucket

readwrite_stscmon_mainbucket

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

    "Action": [
      "s3:*"
    ],
    "Resource": [
      "arn:aws:s3:::stscmon.initialbucket/*",
      "arn:aws:s3:::stscmon.mainbucket/*"
    ]
  }
]
}

```

- Настроить интеграцию Minio и Keycloak

Выполнить команду через `minio/mc`

```
mc admin config set *serverAlias* identity_openid
```

со следующими параметрами:

```
config_url="https://***/auth/realms/***/.well-known/openid-configuration"
```

```
client_id="stscmon_minio_client"
```

```
client_secret="***"
```

```
redirect_uri="http://***/oauth_callback"
```

```
claim_userinfo=true
```

```
scopes="openid,stscmon_minio"
```

```
claim_name="minio_claim"
```

После изменения конфигурации Minio перезагрузить экземпляр.

Для доступа сервисов через kubernetes есть необходимый сценарий сервиса:

```
ansible/roles/kubernetes/templates/minio/minio_service.yaml.j2
```

а также ингресс для доступа через nginx ingress kubernetes:

```
ansible/roles/kubernetes/templates/minio/minio_service.yaml.j2
```

4. Установка и настройка сервисов

4.1. Описание манифестов сервисов kubernetes

Чтобы запустить сервисы посредством docker compose, в docker swarm, nomd, mesos, etc, а также в самом kubernetes создать конфигурационные файлы. Все сервисы манифестов Kubernetes для GitLab ci/cd по пути: `ansible/roles/kubernetes/templates`.

Переменные для деплоя хранятся в `ansible/inventory/%environment%/group_vars/all.yml`.

Описание переменных:

pgsql_cluster: адрес кластера бд

pgsql_main_db: бд приложения

pgsql_db_password: пароль от бд

nfs_host: адрес сервера nfs

importwebservice_pv_path_on_nfs: путь к nfs шару

ingress_enable_cors: true

pgsql_endpoint: имя pg bouncer

pgsql_app_port: порт для pg bouncer

pgbouncer_ip: адрес pg bouncer

keycloak:

Disabled: false

url: url МИБ

realms: реалм в МИБ

ClientId: stscmon_integration_client

ClientSecret: секрет stscmon_integration_client

ClientGuid: guid stscmon_integration_client

ClientScope: stscmon_internal_signalR

Helper:

ClientSecret: секрет stscmon_helper_client

Также есть переменные по умолчанию в `ansible/roles/kubernetes/defaults/main.yml`

env: "окружение (понятие наименования стенда "k8s_namespace: "namespace для kubernetes"

endpoint: "веб адрес модуля In-DAP Indicators "

docker_registry: docker-hub.it-cloud.tech:5004 (dockerhub)

Limits

default_cpu_container_limit: 1000

default_ram_container_limit: 512

ingress_enable_cors: false

#PGSQL Параметры для ДБ

pgsql_app_name: postgresql

pgsql_cluster: 123

pgsql_endpoint: postgresql

pgsql_app_port: 5432

pgsql_db_username: ""

pgsql_db_password: ""

pgsql_main_db: ""

Redis параметры для Redis

redis_app_name: "redis"

redis_app_port: 6379

#redis_cluster: "10.70.35.77"

redis_replicas_count: 1

redis_image_path: redis

redis_image_version: 6.2.1

redis_cpu_limit: 500m

redis_ram_limit: 512Mi

redis_cpu_request: 10m

redis_ram_request: 128Mi

redis_password: password

#ELK Параметры для ELK

elasticsearch_uri: ""

elasticsearch_requireAuth: "true"

elasticsearch_username: ""

elasticsearch_password: ""

elasticsearch_includeAllProperties: "true"