Общество с ограниченной ответственностью «Философия.ИТ»

Россия, 107023, г. Москва, ул. Измайловский Вал, д. 30
Тел.: +7 (495) 988-37-38, факс: +7 (495) 988-37-38,
e-mail: customer@fil-it.ru

ИНН 7713728490
КПП 771901001
ОГРН 1117746379145

Сопроводительная документация по развертыванию Программы для ЭВМ «Микросервисная платформа интеграций (МПИ.ESB)»

На 11 листах

2023

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Термин, определение, сокращение	Определение
СХД, ЭХ	Система хранения данных, электронное хранилище
API, Application Programming Interface	Описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой
HTTP, HyperText Transfer Protocol	Протокол прикладного уровня передачи данных
HTTPS, HyperText Transfer Protocol Secure	Расширение протокола HTTP, для поддержки шифрования в целях повышения безопасности
Kubernetes	Программное обеспечение с открытым кодом для развертывания контейнеров и управления ими в большом масштабе
MS AD, AD (Microsoft Active Directory)	Службы каталогов корпорации Microsoft для операционных систем семейства Windows Server
MongoDB	Документоориентированная система управления базами данных
PostgreSQL	Свободная объектно-реляционная система управления базами данных (СУБД)
WSDL, Web Services Description Language	Язык описания веб-сервисов и доступа к ним, основанный на языке XML
Astra Linux Common Edition	Операционная система специального назначения на базе ядра Linux, созданная для комплексной защиты информации и построения защищённых автоматизированных систем
XML, eXtensible Markup Language	Расширяемый язык разметки - рекомендованный Консорциумом Всемирной паутины (W3C) язык разметки

1. Общие положения

1.1 Полное наименование Программы для ЭВМ, обозначение

Полное наименование Программы для ЭВМ: Микросервисная платформа интеграций (МПИ.ESB).

Краткое наименование (обозначение) Программы для ЭВМ: МПИ.ESB.

1.2 Разработчик Программы для ЭВМ

Полное наименование: Общество с ограниченной ответственностью «Философия.ИТ».

Сокращенное наименование: ООО «Философия.ИТ».

1.3 Назначение документа

Настоящий документ входит в комплект эксплуатационной документации по Микросервисная платформа интеграций (МПИ.ESB) (далее – Система) и предназначен для пользователей Системы.

2. Описание требований

2.1 Минимальные аппаратные требований

Минимальные аппаратные требования указаны в таблице **Ошибка! Источник ссылки не найден.**.

Таблица 1 - Требования к серверам по развертыванию платформы

Оборудование	Кол-во серверов	CPU, ядер	RAM, Gb	HDD/SSD, Gb	Модель CPU	ОС
Kubernetes	1 master-node	4	4	SSD 100 G	Intel® Xeon®	Astra Linux (Smolensk 1.6)
	3 worker-node	16	32	HDD 500 G	Intel® Xeon®	Astra Linux (Smolensk 1.6)
Grafana	1	4	8	HDD 100 G	Intel® Xeon®	Astra Linux (Smolensk 1.6)
Victoria Metrics	1	8	16	HDD 2 T	Intel® Xeon®	Astra Linux (Smolensk 1.6)
Kibana	1	4	8	HDD 100 G	Intel® Xeon®	Astra Linux (Smolensk 1.6)
Elasticsearch	1	16	64	SSD 4 T	Intel® Xeon®	Astra Linux (Smolensk 1.6)
Logstash	1	4	8	HDD 100 G	Intel® Xeon®	Astra Linux (Smolensk 1.6)
Kafka	1	8	8	SSD 500 G	Intel® Xeon®	Astra Linux (Smolensk 1.6)
PostgreSQL	1	8	16	SSD 1 T	Intel® Xeon®	Astra Linux (Smolensk 1.6)
Keycloak	1	5	8	HDD 100 ГЬ	Intel® Xeon®	Astra Linux (Smolensk 1.6)

2.2 Программное обеспечение платформы

В данном разделе в виде таблицы приводится состав комплекса программныхсредств платформы с отображением версий используемого ПО

Системы.

Таблица 2 - Состав комплекса средств и версий ПО Системы

№ п.п	Программный компонент	Версия
1	KeyCloak	20.0.1
2	Apache Kafka	3.1
3	Gitlab	v.14
4	PostgreSQL	14.1
5	Kubernetes	1.23.5
6	Grafana	8.5.1
7	Victoria Metrics	1.76.1
8	Kibana	8.1.3
9	Elasticsearch	8.1.3
10	Logstash	8.1.3
11	Apache NiFi	1.18.0

3. Программное обеспечение серверов

Система подходит для работы с ОС Astra Linux SE 1.6 Смоленск. Программное обеспечение, устанавливаемое исполнителем:

Сервер Postgres:

Postgresql - https://ftp.postgresql.org/pub/source/v14.4/postgresql-14.4.tar.gz офф сайт https://www.postgresql.org/

Сервер Nexus:

Nexus3 - https://download.sonatype.com/nexus/3/nexus-3.49.0-02-unix.tar.gz офф сайт https://help.sonatype.com/

Сервер Kafka:

Apache Kafka - https://archive.apache.org/dist/kafka/3.1.0/kafka_2.13-3.1.0.tgz офф сайт https://kafka.apache.org/

Сервер Keycloak:

Keycloak - https://github.com/keycloak/keycloak/releases/download/18.0.2/keycloak-18.0.2.tar.gz офф сайт https://www.keycloak.org/

Сервер Мониторинга:

Victoria Metrics - https://github.com/VictoriaMetrics/VictoriaMetrics/releases/download/v1.76.1/victoria-metrics-amd64-v1.76.1-cluster.tar.gz офф сайт https://docs.victoriametrics.com/

Grafana - https://dl.grafana.com/oss/release/grafana-8.5.1.linux-amd64.tar.gz офф сайт https://grafana.com/

Сервер ELK:

Elasticsearch – архив предоставлен в комплекте поставки. Офф сайт https://www.elastic.co/elasticsearch/

Logstash - архив предоставлен в комплекте поставки. Офф сайт https://www.elastic.co/logstash/

Kibana - архив предоставлен в комплекте поставки. Офф сайт https://www.elastic.co/kibana/

4. Развертывание

На рабочей станции, с которой будет производится установка, должны быть установлены следующие компоненты:

1) python version = 3.10.7
2) ansible [core 2.12.0]
3) Helm 3.10.0
4) openssh
5) kubectl 4.5.7+

4.1 Развертывание Gitlab runner

1. На рабочей станции должен в директории ~/.kube должен быть файл conf с настроенным подключением к кластеру kubernetes
2. Требуется создать группу в gitlab, где будут размешены проекты для установки микросервисов.
3. Для установки gitlab runner требуется создать файл runner-values.yml и добавить в него следующие строчки:

   gitlabUrl: [адрес gitlab]

   runnerRegistrationToken: [Регистрационный токен группы в gitlab]

   Регистрационный токен можно увидеть, зайдя в веб интерфейсе гитлаба в группу и выбрать Settings > CI/CD > Runners > Registration Token
4. Далее требуется ввести команды:

helm repo add gitlab https://charts.gitlab.io

helm install --namespace gitlab-runner --create-namespace --name gitlab-runner -f runner-values.yml gitlab/gitlab-runner

4.2 Развертывание Шины данных

1. Требуется создать namespace в kubernetes, куда будет производиться установка компонентов шины. Сделать это можно командой
   kubectl create namespace [имя неймспейса]
2. Затем требуется создать в ранее созданной группе в gitlab репозиторий ci-tools и загрузить туда содержимое архива ci-tools.tar.gz
3. Так же требуется создать RBAC сущности для работы ci/cd
   Для этого из архиве ci-tools.tar.gz необходимо распаковать файл k8s/ci-rbac.yml и ввести команду:
   kubectl apply –f k8s/ci-rbac.yml –n [имя неймспейса]
4. В настройках группы в Settings > CI/CD > Variebles треьуется создать переменную KUBECONFIG в которую необходимо поместить следующие данные:

```
apiVersion: v1
kind: Config
clusters:
 - name: prod
 cluster:
 server: [адрес апи сервера кластера kubernetes]
 certificate-authority-data: [сертификат кластера kubernetes]
users:
 - name: prod
 user:
 token: [токен пользователя кластера kubernetes]
contexts:
 - name: prod
 context:
 user: prod
 cluster: prod
 namespace: [имя неймспейса]
current-context: prod
```

5. Где токен пользователя можно получить следующим образом:
   Введя команду kubectl get sa ci-cd -n [имя неймспейса] -o yaml можно получить результат, где нас интересует поле вида

secrets:

- name: ci-cd-token-dxjqq

Получив это имя, с помощью команды

kubectl get secrets [имя секрета] -n [имя неймспейса] -o yaml

Получаем вывод, где нас интересует поле token

Текст, находящийся в нем зашифрован в base64, его необходимо расшифровать.

6. так же необходима техническая учетная запись в gitlab, для которой необходимо создать Access Token
7. Далее необходимо распаковать предоставленный архив mip-config.tar.gz и загрузить его в репозиторий в gitlab
8. Внутри этого репозитория требуется произвести конфигурацию сервисов. Внутри директории каждого сервиса требуется отредактировать файл application.properties, указав следующие настройки

   Route-service

   ```
   kubernetes.projects.url=http://[ip апи сервера kubernetes]
   spring.datasource.username = [имя пользователя postgresql]
   spring.datasource.password = [пароль к postgresql]
   gitlab.api.deploy=[адресс сервера gitlab]/api/v4
   gitlab.private-token=[токен технической учетной записи gitlab]
   spring.datasource.url = jdbc:postgresql://[ip postgresql]:[порт postgresql]/[имя базы в postgresql]
   spring.kafka.producer.bootstrap-servers=[ip сервера с kafka]:[порт сервера с kafka]
   spring.security.oauth2.resourceserver.jwt.issuer-uri=http://[ip сервера с keycloak]:[порт keycloak]/realms/[реал к которому будет подключено приложение]
   ```

9. Для установки микросервисов backend’а потребуется распаковать предоставленные архивы, каждый в отдельный проект в gitlab внутри группы.
10. Для некоторых сервисов нужно указать переменные в настройках репозитория Settings > CI/CD > variebles

    1) Route service

    VALUES

    ```
    nodePort: 30344
    kubeapi:
      enable: true
      conf: "$KUBECONFIG"
    ```

2) Mip-config-service

VALUES

```
readinessProbe:
  httpGet:
  exec:
    command:
      - ls
  initialDelaySeconds: 15
  timeoutSeconds: 3
  periodSeconds: 5
  failureThreshold: 6

livenessProbe:
  exec:
    command:
      - ls
  httpGet:
  initialDelaySeconds: 60
  timeoutSeconds: 4
  periodSeconds: 60
env:
  JAVA_OPTS: "-Dserver.port=8080"
  XDG_CONFIG_HOME: "/tmp/"
```

11. Далее необходимо запустить pipeline для установки сервисов, сделать это требуется в следующем порядке

    Mip-config-service

    Route-service

12. Для запуска pipelibe во вкладке ci/cd > pipelines надо выбрать run pipeline
13. после установки Route-service необходимо подключиться к Postgresql и выполнить там следующие запросы:

    INSERT INTO esb_config.rule (keycloak_role_name, resource_id, resource_type, permission) VALUES ('default-roles-mip', null, 'route', 'create');

    INSERT INTO esb_config.rule (keycloak_role_name, resource_id, resource_type, permission) VALUES ('default-roles-mip', null, 'system', 'create');

    INSERT INTO esb_config.rule (keycloak_role_name, resource_id, resource_type, permission) VALUES ('default-roles-mip', null, 'scheduler', 'create');

    INSERT INTO esb_config.rule (keycloak_role_name, resource_id, resource_type, permission) VALUES ('default-roles-mip', null, 'service', 'create');

    INSERT INTO esb_config.rule (keycloak_role_name, resource_id, resource_type,

permission) VALUES ('default-roles-mip', null, 'roles', 'create');

14. Для установки frontend’а необходимо распаковать архив frontend.tar.gz и загрузить его в gitlab
15. В полученном репозитории в файле .gitlab-ci.yml необходимо переопределить переменные указанные в variebles:
    REACT_APP_KAFKA_URL: [адрес kafka]
    REACT_APP_KUBERNETES_URL: [адрес kubernetes dashboard]
    REACT_APP_KIBANA_URL: [адрес Kibana]
    REACT_APP_GRAFANA_URL: [адрес Grafana]
    REACT_APP_NIFI_URL: [адрес nifi]
    REACT_APP_ELASTIC_URL: [адрес elasticsearch]
16. Для запуска pipelibe во вкладке ci/cd > pipelines надо выбрать run pipeline

4.3 Развертывание Nifi 1.18

1. Для разворачивания необходимо отправить ssh ключ с рабочей станции, на сервер, куда будет устанавливаться Nifi
   Сделать это можно следующей командой:
   ssh-copy-id [имя пользователя на сервре]@[ip адрес сервера]
2. На рабочую станцию необходимо загрузить предоставленный архив administration.tar.gz и распаковать его.
3. В распакованных файлах необходимо создать файл inventory в котором нужно указать адрес сервера, на который будет производиться установка Nifi

```
[nifi]
akkuju-pentaho01
```

4. Далее требуется создать файл group_vars/nifi/main.yml, и указать в нем следующие настройки ldap

```
ldap:
  manager_dn:
  manager_password:
  url:
  user_search_base:
  object_class: person
  user_search_scope: SUBTREE
  user_search_filter:
  user_identity_attribute: sAMAccountName
  user_group_name_attribute: memberOf
  group_search_base:
  group_object_class: group
```

```
group_search_scope: SUBTREE
group_search_filter:
group_name_attribute: cn
group_member_attribute: member
initial_admin_identity:
identity_strategy: USE_USERNAME
```

5. далее требуется ввести в консоль команду
   ansible-playbook nifi.yml –i [путь к ранее созданному файлу inventory] –u [имя пользователя, через которого будет производиться установка]

4.4 Развертывание WSO2

1. Для установки wso2 необходимо, чтобы в кластере kubernetes был установлен ingress controller. Если его нет, его можно установить следующей коммандой
   ```
   helm upgrade --install ingress-nginx ingress-nginx

   --repo https://kubernetes.github.io/ingress-nginx \

   --namespace ingress-nginx --create-namespace
   ```

2. Распаковать предоставленный архив wso2.tar.gz.

3. Подключиться к PostgreSQL и создать там пользователя и базу
   create user wso2user;
   alter user "wso2user" with encrypted password 'wso2password';
   create database apim;
   create database shared;
   create database carbon_cp_1;
   create database carbon_cp_2;
   alter database apim owner to wso2user;
   alter database shared owner to wso2user;
   alter database carbon_cp_1 owner to wso2user;
   alter database carbon_cp_2 owner to wso2user;
   grant all privileges on database apim to wso2user;
   grant all privileges on database shared to wso2user;
   grant all privileges on database carbon_cp_1 to wso2user;
   grant all privileges on database carbon_cp_2 to wso2user;

   где wso2password необходимо заменить на пароль.

4. В папке sqls находятся скрипты для создания таблиц, так так же необходимо применить.

5. Так же в предоставленном архиве есть файл wso2/templates/wso2-pg_secret.yaml, к котором в строчке postgresql-password: d3NvMnBhc3N3b3Jk надо заменить d3NvMnBhc3N3b3Jk на созданный пароль в base64

6. Ввести команду

   helm install wso2 -n wso --create-namespace