Руководство по установке ПО
Aero Platform

Оглавление

1. Глоссарий

Сокращение/Термин	Расшифровка/Значение
WEB-UI	Совокупность веб-страниц, предоставляющая пользовательский интерфейс для взаимодействия с сервисом.
Kubernetes	Платформа с открытым исходным кодом для развертывания, масштабирования, управления и контроля контейнеризованных приложений либо сервисов.
API	(Application Programming Interface) - это набор готовых классов, функций, процедур, структур и констант. Вся эта информация предоставляется самим приложением.
API Gateway	Представляет платформу, предназначенную для эффективной балансировки нагрузки между вычислительными узлами и защиты от злоупотребления ресурсами.
Vault	Это сервис управления секретами, используется для безопасного хранения и управления секретами, такими как пароли, токены и ключи шифрования.
Kafka	Распределенная система

	обработки потоковых данных с открытым исходным кодом. Она позволяет собирать, обрабатывать и хранить большие объемы потоковых данных в режиме реального времени.
Keycloak	Сервер авторизации и управления доступом с открытым исходным кодом.
Opensearch	Поисковая система и аналитическая платформа с открытым исходным кодом. Она обеспечивает надежный и масштабируемый поиск и аналитику для больших объемов данных.
PostgreSQL	Система управления реляционными базами данных с открытым исходным кодом и объектно-реляционными функциями.
Gitlab	Платформа DevOps с открытым исходным кодом, которая объединяет управление кодом, отслеживание проблем, CI/CD и другие инструменты в единый интерфейс.
Redis	Система хранения данных в виде структур
S3	Облачный сервис, позволяющий хранить файлы любого типа и объема

2. Предварительные условия установки

2.1 Совместимость продукта

Модули Ecom платформы, разработанные на технологическом стеке Golang и JS/NextJs, предназначены для развертывания на виртуальных машинах Ubuntu 22.04 и в kubernetes 1.27 и выше.

2.2 Дефолтные порты

Порт	Описание
9092	Подключение клиентов к брокерам Kafka для отправки сообщений
9093	Подключение безопасных клиентов к брокерам Kafka (TLS/SSL)
8200	Подключение к хранилищу секретов Vault
8080	API-сервисы (Admin-Connector, API-Composer, Auth, Cart, Catalog, Content, Geo)
9000	Приложение PHP для динамического контента (Clients)
6379	Кэширование данных (API-Composer, Auth, Cart, Catalog, Clients, Content, Geo)
80/443	Веб-серверы (Clients, CMS, Frontend, API-Gateway)

3. Установка

3.1 Инфраструктурный стек платформы

Вся инфраструктура развернута в **Yandex Cloud**.

Инструменты и технологии

* Managed Service for Kubernetes (3 ноды)
* GitLab для хранения кодовой базы
* Vault для управления секретами
* Krakend в качестве API-Gateway
* PostgreSQL для управления реляционными данными
* Redis для кеширования данных
* Keycloak для аутентификации и авторизации
* Kafka - брокер сообщений
* S3 Object Storage для хранения статической информации
* OpenSearch для поиска и индексации данных
* Rancher платформа для управления Kubernetes-кластерами.

Примечание:
Все перечисленные инструменты и технологии развернуты в Yandex Cloud, что обеспечивает надежность, масштабируемость и безопасность платформы.

3.2 Инструкция по подготовке инфраструктуры

1. Устанавливаем Kubernetes (3 рабочие ноды) либо арендуем Managed Service for Kubernetes
2. Разворачиваем в Kubernetes следующие служебные сервисы и средства Observability:
   учетная запись для доступа к репозиториям:
   https://gitlab.aeroidea.ru
   login: check
   password: OUQrrK4bSn2h

a) Nginx-Ingress-Controller
https://gitlab.aeroidea.ru/platform/iac/nginx-ingress-controller
b) Cert-Manager
https://gitlab.aeroidea.ru/platform/iac/cert-manager
c) Vault
https://gitlab.aeroidea.ru/platform/iac/vault
d) Krakend
https://gitlab.aeroidea.ru/platform/iac/krakend
e) Keycloak
https://gitlab.aeroidea.ru/platform/iac/keycloak
f) Kafka
https://gitlab.aeroidea.ru/platform/iac/kafka
g) Jaeger
https://gitlab.aeroidea.ru/platform/iac/jaeger
h) Superset
https://gitlab.aeroidea.ru/platform-autoprice/iac/superset
i) Kube-Prometheus-Stack
https://gitlab.aeroidea.ru/platform/iac/kube-prometheus-stack

3. Parser-VM:
    a) Создайте ВМ с подходящими параметрами (тип, количество ядер, объем памяти и т. д.).
4. Database-VM:
    a) Создайте ВМ с подходящими параметрами.
    b) Подключитесь к ВМ и установите PostgreSQL и OpenSearch.
5. Gitlab-runner-VM:
    a) Создайте ВМ с подходящими параметрами.
    b) Подключитесь к ВМ и установите gitlab-runner:

```
curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh" | sudo bash
apt-get install -y gitlab-runner
gitlab-runner register (указываем токен репозитория и выбираем тип работы docker)
systemctl enable gitlab-runner
usermod -g www-data gitlab-runner
```

6. S3 Object Storage:

а) Создайте S3-бакет.

б) Предоставьте Database-VM доступ к S3-бакету с помощью IAM-роли или политик доступа.

7. Rancher:

а) Подключиться к ВМ с kubernetes и выполнить следующие команды

```
helm --kube-context=platform install rancher rancher-latest/rancher --create-namespace  --namespace cattle-system  --set hostname=rancher.ваш_домен.ru  --set replicas=1  --set ingress.tls.source=letsEncrypt  --set letsEncrypt.email=email_администратора --set letsEncrypt.ingress.class=nginx
```

измените email_администратора и rancher.ваш_домен.ru на действительные значения

3.3 Инструкция по деплою сервисов

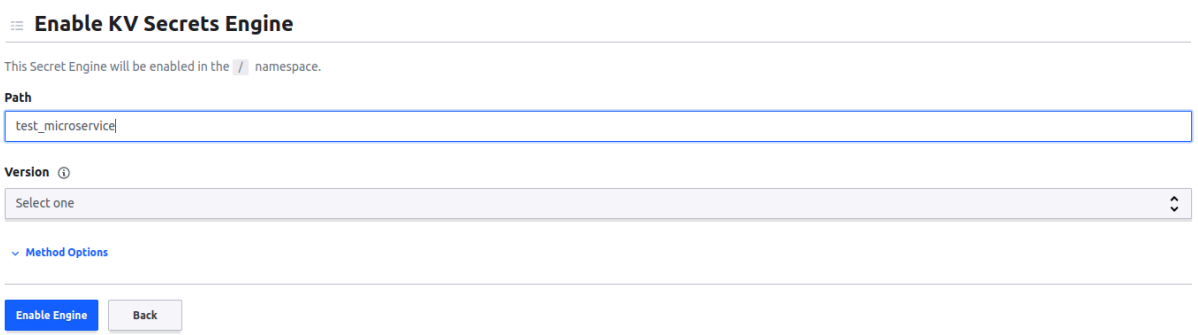
Деплой всех микросервисов в Kubernetes осуществляется посредством gitlab-runner. Для удобства разработан отдельный репозиторий, который включает в себя шаблонизированные решения для тестирования кода, сборки образов и деплоя приложений в зависимости от языка программирования.

Подключение тех или иных шаблонов задается в .gitlab-ci.yml

В качестве инструмента сборки и деплоя служит werf.

Сценарий деплоя нового сервиса:

1. Создаем базу данных в PostgreSQL
2. Создаем в Vault отдельный каталог и заполняем его секретами (переменными)



3. Если нужно, то добавляем в Krakend правила роутинга до api микросервиса

```
{
    "endpoint": "/api/v1/admin/health/check",
    "method": "GET",
    "extra_config": {},
    "output_encoding": "no-op",
    "concurrent_calls": 1,
    "backend": [
        {
            "url_pattern": "/api/v1/health/check",
            "encoding": "no-op",
            "timeout": "60s",
            "sd": "static",
            "extra_config": {
                "github.com/devopsfaith/krakend-circuitbreaker/gobreaker": {
                    "interval": 60,
                    "timeout": 10,
                    "maxErrors": 1
                }
            },
            "host": [
                "{{ .host.admin_connector }}"
            ],
            "disable_host_sanitize": false
        }
    ]
},
```

4. В корне проекта создаем файл werf.yaml (аналог Dockerfile), в котором указываем базовый образ для нашего микросервиса и дополнительные инструкции для сборки образа
5. Создаем .gitlab-ci.yml, в котором подключаем шаблоны ci/cd согласно нашему языку программирования.

.gitlab-ci.yml 130 B

```
variables:
  COVERAGE_LIMIT: "10"
  RUN_TEST_APP: "true"

include:
  - project: 'aero/cicd'
    file: 'gitlab/.gitlab-ci-go.yml'
```

6. Отправляем изменения в gitlab. Этот процесс запустит ci/cd процесс на gitlab-runner.
7. В случае успеха мы увидим новый микросервис в Rancher (см. 4-ый раздел)