Порядок подготовки проекта к системе сборки

Любой проект должен быть подготовлен для автоматической сборки с использованием конвеера CI/CD. Для этого в структуру проекта должны быть внесены соответствующие изменения. Код проекта должен поддерживать конфигурирование через переменные окружения, с использованием **dotenv** или аналогичной библиотеки.

Начальное пополнение проекта

В структуру проекта должны быть добавлены следующие файлы и директории:

- **contrib** - набор вспомогательных файлов для сборки пакета - шаблоны сервисов, конфигурации веб серверов и т.п.
- **debian** - набор рецептов для сборки Debian пакета
- **Makefile** - основной сборочный сценарий проекта
- **build.sh** - специальный файл обновляющий историю изменений и запускающий сборку. Может использоваться для локальной сборки пакетов
- **.env** - Переменные окружения, используемые при сборке проекта в распространяемый debian пакет
- **.env.develop** - Переменные окружения, используемые при настройке стенда, дополняют собой значения переменных, объявленных в файле **.env**
- **.env.example** - Все доступные переменные окружения. Может включаться в документацию проекта.
- **.env.local** - Локальные переменные при локальной сборке и запуске проекта

Набор файлов шаблонов доступен для скачивания с корпоративного GIT репозитория. В шаблонах файлов в качестве имени проекта используется слово **dummy**, его требуется заменить на корректное имя проекта согласно его пути в репозитории GitLab. Путь анализируется от корня репозитория и не учитывает лидирующие пути содержащие слова *back* и *front* в имени проекта. Все прямые слеши (символ /) заменяются на тире (символ -). Если в пути проекта присутствует суффикс, содержащий слова *front* или *back*, то к имени проекта добавляется суффикс *-frontend* или *-backend*, соответственно. Например, если репозиторий имеет путь *cok/lms/front* то полное имя проекта будет иметь вид *cok-lms-frontend*, а если репозиторий имеет путь cok/video-store, то полное имя проекта будет иметь вид *cok-video-store*. Имя проекта, его тип, ветка репозитория отражаются при сборке как соответствующие переменные окружения и могут использоваться в шаблонах конфигурации.

Переменные окружения

При сборке проекта определен ряд переменных окружения, которые автоматически подставляются в файлы конфигурации и файлы **.env** с параметрами окружения проекта. Основные переменные при сборке проекта:

- **PROJECT_NAME** - Короткое имя проекта, без суффикса
- **PROJECT_FULL_NAME** - Полное имя проекта с суффиксом типа проекта
- **PROJECT_TYPE** - Тип проекта - пустое значение, значения *frontend* или *backend*
- **PROJECT_BRANCH** - Собираемая ветка проекта. Для стендов - имя ветки указывается без префикса *stand/*
- **DB_HOST** - Адрес сервера баз данных
- **DB_PORT** - Порт сервера баз данных
- **DB_NAME** - Наименование базы данных на сервере
- **DB_USER** - Имя пользователя для сервера баз данных
- **DB_PASSWORD** - Пароль пользователя для сервера баз данных
- **DB_SCHEME** - Схема на сервере баз данных. Параметр задан только для баз данных совместимых с PostgreSQL!
- **BACKEND_SERVICE** - Адрес бакенд сервера, значение берется из файла конфигурации .env после сборки проекта
- **PUBLISH_TYPE** - Тип публикации проекта, возможные значения: *none, public, private.*
- **PUBLISH_DOMAIN** - Домен публикации - пустое значение, *dev.ao-nk.site* или *dev.ao-nk.ru* в зависимости от типа публикации, задаваемого переменной ***PUBLISH_TYPE***

Если требуется провести подстановку переменных в файл конфигурации времени сборки, используется синтаксис **%VAR%**, где **VAR** - имя переменной.

Если требуется задать пользовательскую переменную, которая будет использоваться в файлах contrib/ingress.yaml или contrib/watchdog.yaml, используется ключевое слово ***export*** перед именем переменной, например:

```
export MY_VAR="Text based value"
```

Секреты и служебная информация

Информация представляющая собой конфиденциальные данные, такие как логины, пароли, ключи и идентификаторы сервисов не должна напрямую фигурировать в файлах конфигурации проекта. Все служебные данные, которые требуется помещать в переменыне окружения на этапе сборки, задаются в виде секретов Jenkins.

Для обращения к секрету используется следующий синтаксис **.env** файла - **%SECRET_*<NAME>*%**, где ***<NAME>*** - имя секрета для собираемого проекта. Секреты в Jenkins имеют формат именования <PROJECT_NAME>_<NAME> в нижнем регистре. Таким образом, использование подстановки **%SECRET_AUTH_ID%** для проекта **cok/lms/frontend** будет искать значение секрета Jenkins с именем **cok-lms_auth_id**.

Рекомендуется выделять отдельный домен Jenkins для каждого сервиса/группы сервисов в Jenkins для точечного управления полномочиями на их просмотр и изменение. Тип секрета Jenkins указывается как **Text Secret**.

Служебные файлы для публикации в Kubernetes

Для публикации сервиса в Kubernetes, в особенности для создания стенда, используются следующие служебные файлы:

- **contrib/Dockerfile** - Рецепт для упаковки образа контейнера (Docker Image). Упакованный образ размещается в корпоративном Docker Registry по адресу registry.ao-nk.ru с именем ***<PROJECT_FULL_NAME>:<PROJECT_BRANCH>***.
- **contrib/ingress.yaml** - Опциональный файл, указывающий правила публикации сервиса. Возможна подстановка переменных во время стадии подготовки контейнера. Переменные подставляются с использованием синтаксиса Bourne Shell - **${VAR}**.
- **contrib/watchdog.yaml** - Опциональный файл, для проверки активности сервиса, согласно [спецификации k18s](#).
- **contrib/dummy.sql** - Опциональный файл дампа начального состояния базы данных. Используется если сервис не предоставляет миграции или требуется определенное начальное состояние базы данных при первичном запуске сервиса или очистке стенда. SQL сценарий должен соответствовать используемой РСУБД.

Пример содержимого файла ***contrib/wathcdog.yaml***:

```
httpGet:
  path: /healthz
  port: 8080
  httpHeaders:
  - name: Custom-Header
    value: Awesome
initialDelaySeconds: 3
periodSeconds: 3
```

Файл ***contrib/ingress.yaml*** описывает параметры публикации сервиса в корпоративной сети или сети интернет:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ${PROJECT_NAME}-${PROJECT_BRANCH}-ingress
  namespace: dev
  annotations:
    nginx.ingress.kubernetes.io/configuration-snippet: |
      proxy_set_header Upgrade "websocket";
      proxy_set_header Connection "Upgrade";
    nginx.ingress.kubernetes.io/proxy-read-timeout: '3600'
    nginx.ingress.kubernetes.io/proxy-send-timeout: '3600'
    nginx.ingress.kubernetes.io/rewrite-target: /$1
    nginx.ingress.kubernetes.io/use-regex: 'true'
    nginx.org/websocket-services: "${BACKEND_SERVICE}"
spec:
  ingressClassName: nginx
  rules:
    - host: ${PROJECT_NAME}-${PROJECT_BRANCH}.${PUBLISH_DOMAIN}
      http:
        paths:
          - backend:
              service:
                name: ${PROJECT_NAME}-${PROJECT_TYPE}-${PROJECT_BRANCH}
                port:
                  number: 80
            path: /(.*)$
            pathType: ImplementationSpecific
          - backend:
              service:
                name: ${BACKEND_SERVICE}
```

```
      port:
        number: 80
    path: /api/(.*|)$
    pathType: ImplementationSpecific
```

Секция Metadata

- **name**: `${PROJECT_NAME}-${PROJECT_BRANCH}-ingress`
  Имя ресурса Ingress генерируется динамически с использованием переменных:
  - `${PROJECT_NAME}`: Название проекта
  - `${PROJECT_BRANCH}`: Название ветки Git
- **namespace**: `dev`
  Ingress развертывается в namespace `dev`, что указывает на использование для разработки.

Аннотации

1. **Поддержка WebSocket**:

```
nginx.ingress.kubernetes.io/configuration-snippet: |
  proxy_set_header Upgrade "websocket";
  proxy_set_header Connection "Upgrade";
nginx.org/websocket-services: "${BACKEND_SERVICE}"
```

   - Активирует поддержку WebSocket для backend-сервиса
   - Устанавливает необходимые заголовки для WebSocket-соединений
   - `${BACKEND_SERVICE}` указывает, какой сервис обрабатывает WebSocket-соединения
2. **Настройки таймаутов**:

```
nginx.ingress.kubernetes.io/proxy-read-timeout: '3600'
nginx.ingress.kubernetes.io/proxy-send-timeout: '3600'
```

   - Устанавливает таймауты чтения и отправки в 3600 секунд (1 час)
   - Полезно для длительных соединений, таких как WebSocket или загрузка файлов
3. **Обработка путей**:

```
nginx.ingress.kubernetes.io/rewrite-target: /$1
nginx.ingress.kubernetes.io/use-regex: 'true'
```

   - Включает обработку путей на основе регулярных выражений
   - Перезаписывает целевой URL, сохраняя совпавший путь (`$1`)

Спецификация (spec)

1. **Класс Ingress**:

```
ingressClassName: nginx
```

   - Используется NGINX Ingress Controller
2. **Правила маршрутизации**:
   - **Хост**: `${PROJECT_NAME}-${PROJECT_BRANCH}.${PUBLISH_DOMAIN}`
     Динамическое имя хоста, состоящее из:
     - Названия проекта
     - Названия ветки
     - Базового домена
   - **Маршруты**:
     - **Основное приложение**:

```
path: /(.*)$
backend:
  service:
    name: ${PROJECT_NAME}-${PROJECT_TYPE}-${PROJECT_BRANCH}
    port: 80
```

       - Перехватывает весь трафик по корневому пути (`/(.*)`)
       - Направляет на фронтенд-сервис с именем из проекта, типа и ветки

3. **API-эндпоинты**:

```
path: /api/(.*|)$
```

```
backend:
  service:
    name: ${BACKEND_SERVICE}
    port: 80
```

- Направляет весь трафик `/api/` на backend-сервис
- Поддерживает как сегменты пути (`/api/something`), так и корень (`/api/`)

Переменные

Переменная	Описание
`${PROJECT_NAME}`	Название развертываемого проекта
`${PROJECT_BRANCH}`	Название ветки Git (для изоляции сред)
`${PROJECT_TYPE}`	Тип проекта ("frontend" или "backend")
`${PUBLISH_DOMAIN}`	Базовый домен для среды
`${BACKEND_SERVICE}`	Название backend-сервиса для API и WebSocket

Примеры использования

1. **Веб-приложение**:
   - URL: `https://myapp-feature.example.com`
   - Отдает фронтенд-приложение
   - API-запросы на `https://myapp-feature.example.com/api/` направляются на бэкенд
2. **WebSocket-соединения**:
   - Тот же backend-сервис обрабатывает WebSocket-соединения
   - Обновление соединений корректно поддерживается через аннотации

Порты и постоянные тома экземпляра сервиса в Kubernetes

Для автоматического определения портов сервиса и постоянных томов сервиса используются ключевые слова ***Dockerfile*** - **EXPOSE** и **VOLUME.**

EXPOSE определяет номер и тип порта (TCP или UDP) сервиса, например:

```
EXPOSE 80/tcp
EXPOSE 5000/udp
```

VOLUME опеределяет путь, внутри контейнера, содержимое которого сохраняется между сборками сервиса (стенда). По умолчанию размер такого тома ограничен квотой в 2 гигабайта. Пример:

```
VOLUME /var/www/service/runtime/uploads
VOLUME /var/spool/crontab
```

SSH доступ к стенду

По окончанию сборки тестового стенда к нему возможен доступ по протоколу SSH, с под логином root с использованием SSH ключа:

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAdzc2gtcn
NhAAAAAwEAAQAAAQEAjPw9pRxAhQr6TZMSKpBxWz4GSFMu/SS8ZtL6Hsb90vonm4K0/Dcz
gIkP7rMTX2yAf4e/5qjuAn66sdi3Kpn234djunXh3Zb1XzjQ+/nF+sS4EQc73j3YFw14oT
N6dZnsRXKUfbV14qSMGoOR2/Taia0NRnYwvu85pxS4Ru5YUBvrktLxc7qp3T+fD6D/dThe
HavMv93MKqLh4IlsMYR10ccRX8bkpz98NU6ZTf9gbfx+JxCq2ya0jksdEt81IzVHdaGI8y
SEQ5V9GS19xsLV3wkCskDmopEXrAFf5Q2WzL/pJosPmtyA39Cc0dm+Z/ADBMD835KmI8ma
8ZAIhoJPuwAAA8BXAkYaVwJGGgAAAAdzc2gtcnNhAAABAQCM/D2lHECFCvpNkxIqkHFbPg
ZIUy79JLxm0voexv3S+iebgrT8NzOAiQ/usxNfbIB/h7/mqO4Cfrqx2Lcqmfbfh2O6deHd
lvVfOND7+cX6xLgRBzvePdgXDXihM3p1mexFcpR9tXXipIwag5Hb9NqJrQ1GdjC+7zmnFL
hG7lhQG+uS0vFzuqndP58PoP91OF4dq8y/3cwqouHiUiwxhHXRxxFfxuSnP3w1TplN/2Bt
/H4nEKrbJrSOSx0S3zUjNUd1oYjzJIRDlX0ZLX3GwtXfCQKyQOaikResAV/lDZbMv+kmiw
+a3lDf0JzR2b5n8AMEwPzfkqYjyZrxkAiGgk+7AAAAAwEAAQAAAQAettchMNH3igg0vT0g
a75eT9IjiUe723R1/DGEYfqrK1dUoDmYltgQAQwpBvdJ+yPVZLgQYq4TehNnKlzhGZC4at
D1rrfJpBkJqSGSO3x/oLqu7wICbTu17ffhOotLsoBQMuGZr14ixZFGN3Kf1iyEAODbAGWn
Owu21CM/RK6VqOSycAoqmnlsaOqaf71FzbNCbgo9eAQjY0Zo7Oxil2RBOV24ZwazEDiue+
XTsSSe/yrJ0p4nJyBdSaLAFiJsCnG66w63ZJY8yleg389OsUEM2B1UTFe4RUPhinXoXjXy
d8+VvE8MfR3uF8xlB4MwN4j9qqfyApWHhO9/cwtr7tiBAAAAgEixCMxVmen36NMPXzepBS
nuxBmevaKWViHpIYxmKzfFhMPmhK6ix07UyFqc/47Q6p6qPyQQj3AZNnz8Ycibaco0Oi+B
RfzvxpaxIADWIrJBx3AbgU7iB6xqZ1kvQLjeP9o2OqsAZ2b5ni2M/CAmGhPcWXYxpEHjWm
```

```
b3tRLDnaBQAAAAgQDB9yoFfLtwdSyUmeuY3Yd8MbKA2BJpUv7Wbw8xCX+0mncDmXZ+O8L8
MPEYw/4Rn0W5YDpnhB4cUuvkKVmsqHEbDARMSJybkPVGbcfLjake3mYq1bW1cF5njPGHgm
PbSIhN/ilOdwUSa/6nr46YeA3E/WZ0x3jjvpnYJzV5VZcHgQAAAIEAuhNZcZsXt3DWEY/M
zcxzjP4WRx4udUbD9JFMQ27iFtlh8wJEwvBMLGnN9uLwotFDixqizbHBhMLadzK3F6QWzz
aY6dHtZGay0VUeIhA1kv8lGpKhxB+9MByCeYKfJeA40wux/Mw7Jv8SYaIjaSULCft9d4Fw
z3ANjvud08QWFTsAAAAJREVWX2FkbWluAQI=
-----END OPENSSH PRIVATE KEY-----
```

Версия #12
★Баталин Иван Анатольевич создал 16 мая 2025 09:13:55
Баталин Иван Анатольевич обновил 5 июня 2025 08:46:56