Инструкция по установке/развертыванию Ensi.Cloud.

Оглавление

Получение доступа к коду

Для того чтобы получить предварительный доступ к коду проекта, необходимо запросить выгрузку архива, который будет содержать все необходимые сервисы, шаблоны генераторов и вспомогательные пакеты. Данный архив является обязательным для успешного развертывания проекта и обеспечивает полный набор компонентов для работы с приложением.

Первичная настройка

После того как все сервисы будут развернуты, необходимо провести первичную настройку.

1. Важно назначить первого суперадмина, что позволит осуществлять управление системой. Это можно сделать с помощью CURL-запроса, используя предоставленный шаблон запроса:

```
curl --request POST \
  --url http://backoffice-users.ensi-
cloud.127.0.0.1.nip.io/api/v1/users/users \
  --header 'Content-Type: application/json' \
```

```
  --data '{
  "client_id": null,
  "is_active": true,
  "login": "Login",
  "name": "Иван",
  "last_name": "Иванов",
  "password": "testPassword775105k#i*"
}'
```

2. Вам также потребуется создать клиента и соответствующего оператора для него через админ-панель. Это необходимо для обеспечения правильной работы системы и управления пользователями.

3. Дополнительно, важно настроить примонтированную папку, чтобы обеспечить корректную работу сервисов. Для этого важно указать путь к папке, которая будет примонтирована, как для сервиса Elastic, так и для сервиса поиска.

   a. В эластик в /usr/share/elasticsearch/config/synonyms
   b. В сервис search в директорию, соответствующую ELASTIC_SYNONYM_LOCAL_CONFIG_FOLDER

4. После настройки кронов и воркеров в соответствии с helm-values, необходимо добавить первичную информацию о товарах. После этого информация будет проиндексирована в Elasticsearch при следующем запуске соответс

Инфраструктура

Список компонентов, используемых в наших проектах для запуска cloud search (и вообще ensi)

- kubernetes, версия >=1.19, три мастера, минимум три ноды по 12 cpu и 24Gb RAM
- ceph, монтируется в кубер как RWX PVC для хранения файлов сервисами
- postgres 15 - основная СУБД
- elasticsearch 8 - поисковой движок
- kafka - для асинхронного межсервисного взаимодействия
- redis - для организации пакетной обработки задач и как кэш

Для организации отказоустойчивости в продуктивных средах компоненты объединяются в кластера.
Для Elasticsarh и kafka используются нативные механизмы.
Postgres и redis реплицируются по модели master-slave.
Везде где возможно настраивается failover с переносом cirtual ip адреса компонента с упавшей машины на живую средствами keepalived.

Разворот в инфраструктуре

1. Для развертывания приложения в Kubernetes обычно используется Helm, который предоставляет универсальный чарт для всех сервисов, а также набор values файлов - по одному или два на каждый сервис.

2. Обновление сервиса сводится к одной команде helm upgrade.

3. Первое развертывание сервиса может потребовать выполнения дополнительных действий, таких как создание PVC для хранения файлов, создание секрета с ключами или запуск какой-то инициирующей команды в самом сервисе.

4. Все различия между сервисами определяются в values файлах и через опцию --set при разворачивании. Сервисы отличаются друг от друга посредством env переменных, набора кронджобов и консюмеров.

5. Сборка и отгрузка сервиса автоматизирована с использованием Jenkins. В пайплайне, помимо кода сервиса, также происходит клонирование репозиториев с Helm чартом и values файлами. Следующим шагом после клонирования происходит установка зависимостей, тестирование и сборка образа.

6. В завершающей стадии, происходит расшифровка values файлов, поскольку некоторые значения внутри них зашифрованы с использованием sops. В конце формируется команда helm upgrade с указанием всех values файлов, только что собранного образа, неймспейса, имени релиза и пути до чарта.

7. Во время отгрузки происходит выполнение helm pre-install hook, который выполняет миграцию структуры БД (если она есть в текущем релизе) и после этого обновляет поды сервиса.

Разворот сервисов на локальной машине

1. Установка Docker

Установка Docker + WSL в случае ОС Windows

Установка Docker делается один раз. Рекомендуется использовать последнюю стабильную версию Docker. Описание способа установки ниже может быть заменено на альтернативные способы установки Docker.

Устанавливаем "Docker Desktop for Windows":https://hub.docker.com/editions/community/docker-ce-desktop-windows.

Устанавливаем "WSL2": https://docs.microsoft.com/ru-ru/windows/wsl/install-win10.

Ставим галочку в настройках докера для интеграции с wsl

Установка Docker compose v2 plugin в случае ОС Linux

Установка Docker делается один раз. Необходимо для работы команды docker compose, которая быстрее чем docker-compose и используется в наших скриптах.

Устанавливаем плагин https://docs.docker.com/compose/cli-command/#install-on-linux

1.1. Структура рабочего пространства проекта и Ensi Local Ctl

1.1.1. Структура директорий рабочего пространства

Для организации рабочего пространства, запуска сервисов ensi и сервисов, необходимых для его работы, используется набор скриптов Ensi Local Ctl , сокращенно ELC.

Ниже показана предпочтительная структура директорий рабочего пространства, однако вы можете изменить её, задав определённые переменные.

```
1.
2├── apps
3│   ├── service-1
4│   ├── service-2
5│   └── domain-1
6│       └── service-3
7├── other
8│   ├── ensi-local-ctl
9│   └── some-repo
10└── packages
11    ├── package-1
12    ├── package-2
13    └── studio.json
```

Директория `apps` содержит в себе сервисы, которые могут находиться на любом уровне вложенности. Директория packages - пакеты, которые необходимо подключать в сервисы во время разработки через symlink.

В директории other (название может быть любым) находится сам ELC, от этого места рассчитываются пути на две другие директории.

1.1.2. Установка Ensi Local Ctl

Для установки ELC необходимо клонировать репозиторий и создать симлинк на скрипт. Симлинк позволит вызывать скрипт в любом месте, а не только в папке с ELC.

`git clone git@gitlab.com:greensight/ensi/devops/ensi-local-ctl.git`

`cd ensi-local-ctl`

`sudo ln -s $PWD/ensi-local-ctl /usr/local/bin/ensi`

Кроме того, инструменты разработки, запускаемые в контейнерах, используют следующие папки:

- `~/.config/composer`
- `~/.cache/composer`
- `~/.npm`

Если их нет, необходимо создать, иначе они будут созданы автоматически с неправильными правами.

Кроме того, необходимо создать отдельную сеть для контейнеров:
docker network create ensi, где ensi - название сети. Вы можете назвать сеть иначе, но тогда вам придётся изменить переменную ELC_NETWORK. Во время создания сети лучше отключить VPN, иначе операция может завершиться ошибкой.

После клонирования репозитория необходимо также выполнить команду ensi global init , находясь в папке ELC

В результате будет создан файл ./global/.elc.env, в котором указаны настройки по-умолчанию. Файл содержит список переменных, значение которых вычислено на основании текущего расположения ELC и с учётом вышеописанной структуры папок. Если ваше рабочее пространство организовано иначе, вам придётся изменить часть из этих переменных.

1.2. Запуск глобальных сервисов

Многие сервисы ensi требуют для своей работы наличие базы данных или брокера очередей. Подобные глобальные сервисы управляются через ELC командами с префиксом global.

```
ensi global start              # запустить сразу всё
ensi global start proxy        # запустить только сервис proxy
ensi global start postgres elasticsearch # запустить сервисы postgres и elastic
```

Полный список операций вы можете получить выполнив команду
`ensi --help`

Глобальные сервисы доступны из сервисов ensi по адресам, совпадающим с их названием. Т.е. для подключения к БД, сервис должен делать соединение на postgres:5432.

Если же вы хотите подключиться БД с хоста, то вам необходимо обращаться на localhost:5432, т.к. порты глобальных сервисов прокидываются на хост.

Отдельно стоит рассмотреть сервис proxy. Это реверс-прокси, который автоматически делает доступными все сервисы в браузере по адресам вида

<service>.ensi.127.0.0.1.nip.io. Это относится как к сервисам ensi, так и к дашбордам глобальных сервисов, например, открыть kafka-ui можно перейдя по адресу http://kafka-ui.ensi.127.0.0.1.nip.io при условии, что глобальный сервис kafka-ui запущен.

1.3. Установка и локальный запуск сервисов

Для установки и запуска сервисов необходимо лишь наличие git и docker на устройстве.

Количество действий, необходимых для запуска, немного отличается в зависимости от типа сервиса

1.3.1. PHP + Laravel: установка

1. Клонируем сервис в директорию apps или куда указывает переменная ELC_APPS_DIR: . Рекомендуется дополнительно создать в apps поддиректории под каждый домен Ensi

   ```
   git clone git@gitlab.com:greensight/ensi/customers/crm.git
   ```

2. Создаем файлы с переменными окружения и заполняем их необходимыми значениями

   ```
   cp .env.example .env
   cp .env.example .env.testing
   ```

3. Собираем docker образ если еще не был собран ранее

   ```
   ensi build
   ```

4. Заходим во внутрь контейнера, дальнейшие команды выполняются внутри него

   ```
   ensi exec sh
   ```

5. Устанавливаем зависимости

   ```
   composer i && npm i
   ```

6. Генерируем ключ

   ```
   php artisan key:generate
   ```

7. Накатываем миграции и тестовые данные

```
ensi build # если докер образ еще не был собран
ensi exec php artisan migrate --seed
```

В простейшем случае сервис лежит в папке apps (или в той, на которую указывает переменная ELC_APPS_DIR), причём в папке сервиса располагается сразу корень его репозитория, без дополнительной вложенности. Сам сервис является php backend сервисом.

В этом случае работать с сервисом можно используя команды
`ensi <args>`
`ensi app <name> <args>`

Это разные способы выполнить команду для определённого сервиса. Первый способ подходит, когда вы находитесь в папке сервиса. Второй не требует находиться в папке сервиса, однако вы должны явно указать к какому сервису относится текущее действие.

В качестве названия сервиса следует использовать путь до сервиса относительно папки ELC_APPS_DIR. Например, чтобы обратиться к сервису, который лежит по пути ELC_APPS_DIR/catalog/pim, необходимо вызывать команду `ensi app catalog/pim <args>`

Перед первым запуском сервиса необходимо собрать его docker образ.
`ensi build`

Эта команда берёт базовый боевой образ и добавляет в него инструменты, необходимые для локальной разработки. По умолчанию все сервисы используют образ с одним названием, поэтому эту команду можно выполнить всего один раз.

После этого сервис можно запустить.

1.3.2. PHP + Laravel: запуск сервиса

`ensi start -` эта команда запускает два контейнера: app и nginx. Сервис доступен в браузере и для других сервисов по адресу http://<name>.ensi.127.0.0.1.nip.io. В случае, когда сервис находится в папке домена, его адрес будет содержать и название домена. Уже известный нам сервис catalog/pim будет иметь адрес http://catalog-pim.ensi.127.0.0.1.nip.io

Можно запустить сервис и не находясь в его папке. Для этого надо модифицировать команду запуска следующим образом:
`ensi app <name> start`

В качестве названия сервиса name следует использовать путь до сервиса относительно папки ELC_APPS_DIR. Например, для сервиса, который лежит в директории ELC_APPS_DIR/catalog/pim, получаем:
`ensi app catalog/pim start`

PHP + Laravel: запуск тестов
1ensi exec composer test
2ensi exec composer test-coverage

1.4. Запуск инструментов разработки

Dev образ приложения отличается от боевого наличием в нём инструментов, необходимых во время разработки. Это могут быть как отдельные программы, вроде composer, так и дополнительные расширения php, например xdebug.

Предпочтительным способом запуска инструментов является вход в контейнер приложения
`ensi exec sh`
`composer install`
`php artisan key:generate`

Можно, однако, и запускать команды не переводя терминал внутрь контейнера
`ensi exec composer install`
`ensi exec php artisan migrate`

1.5. Настройка git хуков

Хуки организованы через husky. Чтобы они начали работать, необходимо выполнить `npm install` в корне сервиса. Во время установки пакета husky перехватывает все хуки гита, и дальше уже на основании своих настроек вызывает тот или иной скрипт из папки `.git_hooks`.

С сервисами поставляются хуки, способные работать как прямо на хосте, так и в контейнерах ELC. По умолчанию они работают на хосте. Чтобы они работали в контейнерах, необходимо выше по файловой системе создать файл `.external-runtime`, который содержит такой текст:
`EXTERNAL_RUNTIME_COMMAND="ensi exec-script"`

Файл должен располагаться выше точки монтирования сервиса в контейнер. Лучше класть его в корень рабочего пространства проекта.

Когда этот файл существует, хуки используют указанную в нём команду для запуска самих себя в контейнере текущего сервиса. Так как скриптов в каждом хуке бывает больше одного, первый из них при своём выполнении запускает контейнер app текущего сервиса, а последующие просто делают exec в работающем контейнере.

2. IРазворот admin-gui-frontend на локальной машине
    1. Необходимо убедиться, что установлен node js LTS версии https://nodejs.org/en/ , yarn https://classic.yarnpkg.com/en/docs/install#mac-stable первой версии и git при необходимости https://git-scm.com/book/ru/v2/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-

%D0%A3%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0-Git

2. Если все установлено, клонируем репозиторий
   `git clone git@gitlab.com:greensight/ensi/admin-gui/admin-gui-frontend.git`
3. Создаем файлы с переменными окружения и заполняем их необходимыми значениями
   `cp .env.example .env.local`
4. Для корректной работы приложения важно указать API_HOST, ссылающийся на API_HOST=https://admin-gui-backend-master-dev.ensi.tech/
5. Устанавливаем зависимости
   `yarn`
6. Запускаем приложение

Команда	**Описание**
yarn dev	Запустить приложение в dev-режиме
yarn prod	Запустить приложение в prod-режиме
yarn build	Собрать приложение
yarn build:analyze	Собрать приложение + bundle analyzer для анализа бандлов
yarn test	Запустить тесты (jest)
yarn start	Запустить сервер

yarn lint	Запустить eslint
yarn format	Запустить prettier
yarn tsc	Запустить Typescript compiler
yarn storybook	Запустить storybook
yarn storybook:build	Собрать storybook
yarn storybook:serve	Запустить сервер для отображения собранного storybook
yarn tokens	Обновить GDS-токены
yarn repack	Переразвернуть приложение, удалив node_modules, yarn cache
yarn prepare	Устанавливает husky

6. Для использования команды yarn tokens необходимо создать gds.config.json и заполнить необходимыми значениями.