Руководство по установке

компонента Устранение Сбойных Ситуаций (код: USSX)

продукта Platform V Incident Manager (код: USS)

Содержание

Руководство по установке компонента Устранение Сбойных Ситуаций (USSX)

В руководстве приведены инструкции по установке Продукта.

Основные понятия

В таблице приведены основные аббревиатуры и сокращения:

Аббревиатура, сокращение	**Определение**
БД	База данных
ИС	Информационная система
ПО	Программное обеспечение
СУБД	Система управления базами данных
ТУЗ	Техническая учетная запись
КБ	Корпоративный блок
РБ	Розничный блок
УЦ	Удостоверяющий центр
API	Application Programming Interface. Интерфейс прикладного программирования
Common репозиторий	Репозиторий глобальных параметров среды. Репозиторий с общими настройками продукта Platform V DevOps Pipeline Management (DPM), при условии использования опционально компонента Deploy Tools (CDJE) продукта Platform V DevOps Tools (DOT)
GitLab CE	Инструмент для хранения и управления репозиториями Git; приложение с открытым исходным кодом под лицензией MIT, в основном используемое для хостинга хранилищ Git
Jenkins	Продукт, распространяемый под лицензией Apache 2.0. Программная система с открытым исходным кодом на Java, предназначенная для обеспечения процесса непрерывной интеграции программного обеспечения. Jenkins job или джоба Jenkins по тексту обозначена как «задание Jenkins»
Istio	Сервис интеграции и оркестрации микросервисов в облаке, распространяемый под лицензией Apache 2.0 (рекомендуется использование продукта Platform V Synapse Service Mesh (SSM))
Kafka	Распределенный программный брокер сообщений, проект с открытым исходным кодом под лицензией Apache 2.0 (рекомендуется использовать продукт Platform V Corax (KFK))
Keystore	Хранилище закрытых ключей и клиентских SSL сертификатов
Nexus-Public	Репозиторий для хранения дистрибутивов компонентов, проект с открытым исходным кодом под лицензией EPL v.1
Pipeline DevOps	Процесс разработки по типу конвейера с использованием опционально компонента Deploy Tools (CDJE) продукта Platform V DevOps Tools (DOT)
PSQL	СУБД PostgreSQL (рекомендуется использование продукта Platform V Pangolin SE (PSQ))

В таблице приведены основные термины:

Термин	Определение
Платформа, Platform V	Набор продуктов Platform V, правообладателем которых является АО «СберТех». Перечень таких продуктов обозначен в документации на конкретный продукт
Продукт	Компонент Устранение Сбойных Ситуаций (USSX) продукта Platform V Incident Manager (USS)
Среда контейнеризации	Kubernetes (рекомендуется), поддержана опциональная совместимость с OpenShift 4+

Системные требования

Настройки безопасности окружения и перечень платформенных (дополнительных внешних) продуктов, используемых для установки, настройки и контроля в конечной информационной системе (далее — ИС), выбираются при разработке конечной ИС, исходя из характера обрабатываемой в ней информации и иных требований информационной безопасности (далее — ИБ), предъявляемых к ней.

Системное программное обеспечение

Ниже представлены категории системного программного обеспечения, которые **обязательны для установки**, настройки, контроля и функционирования Продукта. В каждой категории перечислены все поддерживаемые продукты сторонних правообладателей. Отдельно обозначены варианты, которые рекомендует АО «СберТех» (маркировка «Рекомендовано» в столбце Комментарий). Клиенту необходимо выбрать один из продуктов в каждой категории, исходя из условий использования конечной ИС.

Операционная система

Наименование	Версия	Применение
ОС Альт 8 СП	10.0 и выше	Рекомендовано
Red Hat Enterprise Linux	3.10 и выше	Предусмотрена опциональная совместимость

Среда контейнеризации

Необходимое окружение для разворачивания компонента — Kubernetes версии 1.23 и выше или OpenShift (опционально) версии 4.2 и выше (далее — среда контейнеризации).

Наименование	Версия	Комментарий
Kubernetes	1.23 и выше	Рекомендовано
Red Hat OpenShift	4.2 и выше	Предусмотрена опциональная совместимость

Инструмент контейнеризации

Наименование	Версия	Комментарий
Docker CE		Рекомендовано

Инструмент сборки, тестирования, развертывания контейнеризированных приложений

Наименование	Версия	Комментарий
Jenkins		Рекомендовано

Java-машина

Наименование	Версия	Комментарий
OpenJDK	1.8 и выше	Рекомендовано
OracleJDK	1.8 и выше	Предусмотрена опциональная совместимость

Система управления базами данных (СУБД)

Наименование	Версия	Комментарий
PostgreSQL	42.2.5 и выше	Рекомендовано
Oracle Database	–	Предусмотрена опциональная совместимость
Oracle GoldenGate	–	Предусмотрена опциональная совместимость

Правообладателем также рекомендована СУБД, основанная на PostgreSQL, – Platform V Pangolin SE, см. раздел «Платформенные зависимости».

Сервер приложений

Наименование	Версия	Комментарий
Apache Tomcat	9.0.39 и выше	Рекомендовано

Браузер

Наименование	Версия	Комментарий
Яндекс		Рекомендовано
Google Chrome		Предусмотрена опциональная совместимость

Инструмент управления проектом

Наименование	Версия	Комментарий
Apache Maven	3.0 и выше	Рекомендовано

Сервис централизованного хранения репозиториев артефактов (хранилище артефактов)

Наименование	Версия	Комментарий
Nexus-Public	2.5.1 и выше	Рекомендовано
Nexus Repository Manager PRO	-	Предусмотрена опциональная совместимость
Nexus Repository Manager OSS	-	Предусмотрена опциональная совместимость

Сервис централизованного хранения репозиториев исходного кода

Наименование	Версия	Комментарий
GitLab	15.0 и выше	Рекомендовано
Bitbucket	7.6 и выше	Предусмотрена опциональная совместимость

Система мониторинга сборки

Наименование	Версия	Комментарий
Prometheus	2.31 и выше	Рекомендовано

Система для визуализации численных метрик (предоставленных, например, Prometheus):

Наименование	Версия	Комментарий
Grafana	2.5.0 и выше	Рекомендовано

Удостоверяющий центр

Наименование	Версия	Комментарий
EJBCA Community	-	Рекомендовано

Сервис интеграции и оркестрации микросервисов в облаке

Наименование	Версия	Комментарий
Istio	1.12 и выше	Рекомендовано

Правообладателем также рекомендован сервис интеграции и оркестрации микросервисов в облаке, основанный на Istio, – Platform V Synapse Service Mesh, см. раздел «Платформенные зависимости».

Платформенные зависимости

Для настройки, контроля и функционирования Продукта реализована интеграция с программными продуктами, правообладателем которых является АО «СберТех»:

Наименование продукта	Код	Версия	Код и наименование компонента	Обязательность установки	Описание
Platform V Pangolin SE	PSQ	4.6.0. и выше	-	Обязательно	Система управления базами данных, основанная на PostgreSQL
Platform V Backend	#BD	03.035	APLJ Прикладной журнал	Опционально	Сервис для обеспечения отказоустойчивости приложения на уровне базы данных
Platform V Audit SE	AUD	1.1	AUDT Аудит	Опционально	Сервис для аудирования событий
Platform V Synapse Service Mesh	SSM	1.12 и выше	IGEG Граничный прокси	Опционально	Сервис для обеспечения управляемого вызова интеграционных сервисов прикладной части

Platform V Monitor	OPM	3.27.11	LOGA Журналирование	Опционально	Сервис для хранения лог-файлов
Platform V Monitor	OPM	6.0.0-02	MONA Объединенный мониторинг Unimon	Опционально	Сервис для сбора прикладных и инфраструктурных метрик и отправки их в целевую систему хранения
Platform V Backend	#BD	1.8.0-4.1.21	OTTS One-Time Password / One-Time-Token	Опционально	Сервис для аутентификации и авторизации межсервисных взаимодействий
Platform V IAM SE	IAM		AUTH IAM Proxy	Опционально	Сервис управления доступом и информационными ресурсами
Platform V IAM SE	IAM		AUTZ Объединенный сервис авторизации	Опционально	Сервис авторизации

Примечание:

- **Обязательно** — компонент или продукт необходим для функционирования сервиса.
- **Опционально** — необязательный для функционирования сервиса компонент или продукт, рекомендуется его установка, но допускается использование аналога других производителей.

Аппаратные требования

Для установки Продукта требуется нижеописанная конфигурация аппаратного обеспечения.

Продукт состоит из двух модулей – *USS* и *ERRM*, которые разворачиваются каждый в своем пространстве имен.

Квота на проект: 50 CPU, 128 ГБ.

Квота на pod с учетом sidecar-контейнеров:

- модуль USS: 26 CPU, 64 ГБ;
- модуль ERRM: 26 CPU, 64 ГБ.

Подготовка окружения

Подготовка к развертыванию Продукта включает в себя выполнение следующих шагов:

1. Заведите ТУЗ для доступа.

2. Запросите права на чтение репозитория, содержащего код задания Jenkins для развертывания.
3. Заведите заявку на предоставления доступа к registry для ТУЗ.
4. Импортируйте задание склейки в Jenkins (xml в Jenkins, а остальные необходимые файлы в Gitlab CE).
5. Импортируйте задание развертывания в Jenkins (xml в Jenkins, а остальные необходимые файлы в Gitlab CE).
6. Задайте права и полномочия в Jenkins для запуска Pipeline DevOps. Есть специальное задание для создания полномочий в Jenkins с правами на доступ к среде контейнеризации. Опционально можно сделать другим способом.
7. Задайте права и полномочия в Jenkins, содержащую ТУЗ для доступа к Nexus-Public, где будет лежать дистрибутив приложения.
8. Создайте git репозиторий (получение прав) для хранения файлов конфигурации.
9. Создайте клиентский сертификат.
10. Сгенерируйте сертификаты для компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD) и создайте секреты (опционально).
11. Создайте ServiceAccount Kubernetes или OpenShift (опционально). Service-acc-ingress необходимо привязать imagePullSecret, выполнив команды:

```
kind: ServiceAccount
apiVersion: v1
metadata:
name: default
namespace: tribe-pt-dev-ussx-k8s
uid: <значение>
resourceVersion: '<значение>'
creationTimestamp: '2022-04-28T12:06:26Z'
secrets:
 - name: default-token-9x95x
imagePullSecrets:
 - name: tuz-image-pull
```

```
kind: Secret
apiVersion: v1
metadata:
name: default-token-9x95x
namespace: tribe-pt-dev-ussx-k8s
uid: <значение>
resourceVersion: '<значение>'
```

```
creationTimestamp: '2022-04-28T12:06:26Z'
annotations:
kubernetes.io/service-account.name: default
kubernetes.io/service-account.uid: <значение>
managedFields:
data:
ca.crt: >-
<значение>
namespace:
<значение>
token: >-
<значение>
type: kubernetes.io/service-account-token
```

Состав дистрибутива

В данный момент в дистрибутиве содержатся темплейты для установки в среду контейнеризации Kubernetes или Red Hat OpenShift 4+ (опционально) и миграции баз данных.

Дистрибутив имеет следующую структуру:

Distr.zip

```
|package/
   |________bh/
                |________app-api/. \<- jar файлы Апп-апи
                |________auth-api/. \<-jar файлы  для Аус-апи
                |________datamart-api/. \<-jar файлы для Датамарт-апи
                |________inner-connector-api/. \<-jar файлы  для Иннер-коннектор-апи
   |________docker/
                |________app-api/. \<- docker файлы Апп-апи
                |________auth-api/. \<-jar файлы  для Аус-апи
                |________datamart-api/. \<-jar файлы  для Апп-апи
                |________inner-connector-api/. \<-jar файлы  для Иннер-коннектор-апи
                |________frontend/. \<-jar файлы  для frontend
   |________pl/
                |________frontend/. \<- файлы статики  для frontend
   |________db/
                |________app-api.zip \<-Миграции для Апп-апи
                |________ auth-api.zip \<-Миграции для Аус-апи
                |________ inner-connector-api.zip \<-Миграции для иннера
                |________postgresql.jar
                |________liquibase.jar
                |________version.conf
   |conf/deploy_job/ - Джоба деплоя
   |conf/recreate_distrib_job/ Джоба склейки
   |conf/k8s/base
               |________isio/*.yaml
               |________app-api/*.yaml
               |________auth-api/*.yaml
```

Установка

Настройка сертификатов для компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD)

Для настройки сертификата для компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD) (является опциональным) необходимо следующее.

1. Сгенерируйте ключевую пару. Пример (создание происходит с помощью инструмента keytool с примером команды):

   ```
   keytool -genkey -keyalg EC -sigalg SHA256withECDSA -keystore ${module_id}.p12 -storetype PKCS12 -keysize 256 -dname "CN=${module_id}" -alias ${module_id}
   ```

2. Сгенерируйте запрос на сертификат. Пример:

   ```
   keytool -certreq -keyalg EC -sigalg SHA256withECDSA -keystore ${module_id}.p12 -storetype PKCS12 -alias ${module_id} > ${module_id}_cert_req.pem
   ```

   В результате выполнения данной команды будет создан файл `CSR-**${**module_id}_cert_req.pem.**`

3. Полученный файл передайте администратору компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD) в заявке на генерацию сертификата для модуля. В заявке нужно в явном виде указать, что сертификат должен быть сгенерирован из приложенного запроса по данной инструкции. Результатом выполнения заявки будет два файла: сертификат УЦ компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD) и сертификат модуля.

4. Добавьте полученные сертификаты в keystore.

5. Получите сертификат публичного ключа компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD) одним из нижеописанных способов.

   Сертификат публичного ключа компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD) необходим для корректной работы клиента компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD) и используется для проверки подписи компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD).

   Необходимо скачать сертификат и импортировать его в хранилище самостоятельно (должны быть установлены утилиты curl и keytool).

Получение сертификата компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD):

```
# Download OTTS Service certificate
curl -X GET "https://<ott-service-host>:8442/ejbca/publicweb/webdist/ce
rtdist?cmd=lastcert&subject=cn%3Dott-service" --insecure > ott-service.
crt
### Use PKCS12
# Import OTTS Service certificate into PKCS12 store
keytool -importcert -file ott-service.crt -keystore truststore.p12 -sto
retype PKCS12 -storepass <password> -alias "ott-service" -noprompt
### or JKS
# Import OTTS Service certificate into JKS store
keytool -importcert -file ott-service.crt -keystore truststore.jks -sto
retype JKS -storepass <password> -alias "ott-service" -noprompt
```

Необходимо загрузить подготовленное хранилище.

Также для удобства создан скрипт с возможностью генерации такого сертификата. Для генерации запроса на сертификат компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD) выполните:

```
./cert_generate.py -CR -OTT -c ussx для ussx
```

Создание ТУЗ

В работе системы АС используется техническая учетная запись, например, *uss*.

Чтобы получить техническую учетную запись, обратитесь к системному администратору или оформите заявку на получение технической учетной записи внутри своей организации.

Для доступа к базе данных эта учетная запись добавляется отдельно, при этом необходимо гарантировать следующий набор прав:

- create table;
- schema;
- index;
- Grant select;
- update;
- delete;
- truncate;
- insert;
- trigger;
- execute;
- references;
- connect;

- temporary;
- usage;
- alter table.

Отсутствие следующих прав приводит к ошибкам на этапе установки:

- create table;
- schema.

При отсутствии остальных будут возникать ошибки в процессе работы самого Продукта.

Создание клиентского сертификата

Для настройки https, а также для общения с рядом технологических сервисов необходимо создать клиентские сертификаты приложения.

Создание подразумевает два этапа:

1. Создание запросов на сертификаты.
2. Отправление запросов на создание сертификатов в удостоверяющий центр.

Для создания запроса необходима консоль с sh.

Чтобы создать закрытый ключ, выполните следующие действия:

1. Создайте в консоли директорию для сертификатов, выполнив команду:

   ```
   mkdir -p certs && cd certs
   ```

2. После создания директории создайте запрос. Создайте закрытый ключ, воспользовавшись утилитой OpenSSL:

   ```
   OpenSSL> genrsa -aes256 -out ourPrivatKey.key 2048
   ```

   Крайне важно запомнить пароль к контейнеру, который мы будем вводить при создании ключа.

3. В результате в директории certs появится закрытый ключ ourPrivatKey.key.

Чтобы создать открытый ключ, выполните следующие действия:

1. Воспользуйтесь утилитой OpenSSL:

   ```
   OpenSSL> req -new -key ourPrivateKey.key -out ourCertificate.csr
   ```

2. Далее консоль запросит пароль для подтверждения. Введите пароль, который был задан на предыдущем шаге.

3. Заполните информацию в сертификате: Country Name: RU; State or Province: ; Organization Name: XXXX; Organizational Unit Name: XXXX; Common Name: ; Emal

Address: . В результате на выходе будет получен запрос на открытую часть сертификата, который необходимо направить коллегам из УЦ, для создания сертификата.

4. После получения сертификатов разделите на корневой, промежуточный и клиентский сертификат. Для этого откройте сертификат для просмотра, нажмите на экспортируемый сертификат и выберите *Экспортировать в файл*, в качестве кодировки выберите *Файлы X.509 (.CER) в кодировке Base64* все эти сертификаты сохраните отдельными файлами.

5. После чего преобразуйте в OpenSSL клиентский сертификат и дешифруйте приватный ключ, для этого обратитесь к OpenSSL:

   ```
   OpenSSL> x509 -inform der -in ourCertificate.cer -out ourSertificate.pe
   m
   ```

6. Далее дешифруйте приватный ключ:

   ```
   OpenSSL> rsa -in ourPrivateKey.key -out unencryptPrivateKey.key
   ```

   Команда попросит ввести пароль, который был задан в первых шагах.

7. Далее соберите данные ключи в секреты среды контейнеризации, для чего выполните следующее:

   - Авторизуйтесь в среде контейнеризации, зайдите в необходимый проект, в котором планируется устанавливать приложение.
   - Пройдите в WorkLoads → Secrets → Create Secret → Key/Value Secret.

Для создания секрета *istio-ingressgateway-ca-certs* необходимо следующее.

Чтобы создать цепочку клиентских сертификатов, выполните следующие действия:

1. Перейдите в WorkLoads → Secrets → Create Secret → Key/Value Secret имя секрета–istio-ingressgateway-certs

   - ключ – tls.crt;
   - значение – текстовое содержимое клиентского ключа в формате OpenSSL созданного ранее.

2. Вторым ключом добавьте дешифрованный ключ, чтобы хранить их в 1-ом секрете:

   - ключ – tls.key;
   - значение – текстовое значение дешифрованного ключа, созданного ранее.

Инструкция создания клиентских сертификатов представлена в данном документе. Для настройки https, а также для общения с рядом технологических сервисов необходимо создать клиентские сертификаты. Создание делится на 2 этапа:

- создание запросов на сертификаты;
- далее запросы необходимо отправить на создание сертификатов в удостоверяющий центр.

1. Для создания запроса необходима консоль с sh, в которой нужно создать директорию для сертификатов и перейти в нее, выполнив команду Создание директории `mkdir -p certs && cd certs.` После создания директории создайте запрос. Сначала создайте закрытый ключ, воспользовавшись утилитой OpenSSL `OpenSSL> genrsa -aes256 -out ourPrivatKey.key 2048` Крайне важно запомнить пароль к контейнеру, который будете вводить при создании ключа. В результате в директории certs появится закрытый ключ ourPrivatKey.key.

2. Создайте открытый ключ, для чего воспользуйтесь утилитой OpenSSL `OpenSSL> req -new -key ourPrivateKey.key -out ourCertificate.csr` Консоль запросит пароль для продолжения, введите пароль, который задавали в прошлом шаге. Далее необходимо заполнить информацию в сертификате:

   Country Name: RU; State or Province: <оставить пустым>;

   Organization Name: <указать наименование>;

   Organizational Unit Name: <указать наименование>;

   Common Name: our hostname;

   Emal Address: <оставить пустым>.

В результате на выходе получите запрос на открытую часть сертификата, который необходимо направить коллегам из УЦ, для создания сертификата. Получение сертификатов необходимо разделить на корневой, промежуточный и клиентский сертификат. Для этого откройте сертификат для просмотра, нажмите на экспортируемый сертификат и выберите экспортировать в файл, в качестве кодировки выберите «*Файлы X.509 (.CER) в кодировке Base64»*. Все эти сертификаты сохраните отдельными файлами. После чего преобразуйте в OpenSSL клиентский сертификат и дешифруйте приватный ключ. Для этого вновь обратитесь к OpenSSL:

```
OpenSSL> x509 -inform der -in ourCertificate.cer -out ourSertificate.pem
```

Далее дешифруйте приватный ключ:

```
OpenSSL> rsa -in ourPrivateKey.key -out unencryptPrivateKey.key
```

Команда попросит ввести пароль, который был задан на первых шагах. Далее соберите данные ключи в секреты среды контейнеризации:

1. Авторизуйтесь.

2. Зайдите в проект.

3. Пройдите по цепочке: WorkLoads → Secrets → Create Secret → Key/Value Secret.
4. Создайте цепочку клиентских сертификатов.
5. Перейдите в WorkLoads → Secrets → Create Secret → Key/Value Secret. Где имя секрета - istio-ingressgateway-certs, ключ – tls.crt, значение – текстовое содержимое клиентского ключа в формате OpenSSL, созданного ранее. Вторым ключом необходимо добавить дешифрованный ключ, чтобы хранить их в 1-ом секрете: ключ – tls.key, значение – текстовое значение дешифрованного ключа, созданного ранее. Для удобства создания сертификата создан скрипт. Для начала необходимо заполнить san.cnf.

```
[ req ]
default_bits       = 2048
distinguished_name = req_distinguished_name
req_extensions     = req_ext
prompt = no

[ req_distinguished_name ]
countryName                 = RU
stateOrProvinceName         = MOW
localityName                = MOW
organizationName            = NAME
commonName                  = <URL>

[ req_ext ]
subjectAltName = @alt_names

[alt_names]
DNS.1 = <URL>
DNS.2 = <URL>
DNS.3 = <URL>
DNS.4 = <URL>
```

Далее необходимо запустить скрипт:

`./cert_generate.py -CR -server -req san.cnf -c CommonName`.

После чего придет запрос на создание сертификата.

Создание секретов для среды контейнеризации (опционально)

Для выполнения команд необходим установленный клиент среды контейнеризации Kubernetes или Red Hat OpenShift 4+ (опционально).

Необходимо получить следующие секреты:

- ignite-se-cluster-secrets;
- kafka-secret;
- secret-ingressgateway-ca-certs;

- secret-ingressgateway-certs;
- secret-uss-eggrm;
- secret-uss-ott;
- secret-uss-ott-certs.

Создание секрета

За основу возьмите yaml-файл примерно такого содержания:

```
kind: Secret
apiVersion: v1
metadata:
name: <имя секрета>
data:
<key>: <value>
type: Opaque
```

secret-ingressgateway-ca-certs и **secret-ingressgateway-certs** – для хранения сертификатов Envoy.

Файлы подготавливаются при создании клиентского сертификата:

1. Логин в среде контейнеризации: `oc login --token=<openshift_user_token> --server=<openshift_api_url>:<значение>`

2. Создание секрета istio-certs (вывод содержимого в консоль): `oc create secret generic istio-certs --from-file=root-ca.pem --from-file=envoy.crt --from-file=envoy.key --dry-run -o yaml > secret-istio-certs.yaml`

3. Создание секрета secret-ingressgateway-ca-certs: `oc create secret generic secret-ingressgateway-ca-certs --from-file=ca.pem=root-ca.pem --dry-run -o yaml > secret-ingressgateway-ca-certs.yaml`

4. Создание секрета secret-ingressgateway-certs: `oc create secret generic secret-ingressgateway-certs --from-file=tls.key=envoy.key --from-file=tls.crt=envoy.crt --dry-run -o yaml > secret-ingressgateway-certs.yaml`

5. Применить задание Jenkins для шифрования файлов.

6. Выложить зашифрованный файл секрета в git repository в папку /config///secrets.

Для простоты использования, секреты можно создать из ранее подготовленных сертификатов с помощью скрипта:

```
./cert_generate.py -h
   usage: cert_generate.py [-h] -c CN [-p PWD] [-r ROOT] [-i INTERMEDIATE]
[-C CLIENT] [-CR] [-I] [-kafka] [-server] [-req REQ]
[-IngressSecret] [-secretname SECRETNAME]
[-OSKafkaSecret] [-Certs CERTS] [-Keys KEYS] [-OTT]
```

```
[-OTTSecret] [-ottpem OTTPEM] [-rootsber ROOTSBER]
Create JKS with you optional arguments:
-h, --help show this help message and exit
-c CN, --cn CN common name
-p PWD, --pwd PWD password for jks
-r ROOT, --root ROOT root certificate
-i INTERMEDIATE, --intermediate INTERMEDIATE intermediate cert
-C CLIENT, --client CLIENT client cert
-CR, --create need create jks
-I, --Import need add cert to exist jks
-kafka Crete jks for kafka
-server create openshift ingress cert from req file
-req REQ req файл с параметрами создаваемого сертификата
-IngressSecret create OpenShift secret for Ingress/Egress from cert key and t
rusted CA
-secretname SECRETNAME имя создаваемого сертификата
-OSKafkaSecret create OpenShift secret for kafka from cert key and trusted CA
-Certs CERTS массив сертификатов для Kafka (Kafka Application Journal), в пор
ядке, указанном для ключей
-Keys KEYS массив ключей, в порядке, указанном для сертификатов для Kafka (Ka
fka Application Journal)
-OTT create cert for OTT
-OTTSecret create secret for OTT
-ottpem OTTPEM Ott pem from ott ZNO
-rootsber ROOTSBER root pem from ott ZNO
```

Создать секрет для компонента One-Time Password / One-Time-Token (OTTS) продукта Platform V Backend (#BD): `./cert_generate.py -OTTSecret -c CommonName -secretname secret-errm-ott-certs`

Где CommonName – созданный на предыдущем шаге, этим же скриптом, сертификат.

Создать kafka-secret:

```
./cert_generate.py -server -req san.cnf -c <hostname> -OSKafkaSecret -secretn
ame kafka-cert -r ../RootCa.cer -i ../intermediate.cer -Certs <hostname>/<hos
tname>,<hostname>/<hostname> -Keys APP_cert,AUTH_cert
```

Создать серверный сертификат:

```
./cert_generate.py -server -req san.cnf -c <hostname> -IngressSecret -secretn
ame secret-ingressgateway-certs -r ../RootCa.cer -i ../intermediate.cer
```

Конфигурация

Все стендо-зависимые параметры (минимальная конфигурация) и параметры основных интеграций указываются в файле deploy/ansible/vars/varisbles.yaml. Полный список возможных параметров представлен ниже.

Файл .env

NAMESPACE: название namespace (пространства имен)
REGISTRY_URL: URL registry с docker-образами

############################## Интеграция с компонентом Аудит (AUDT) продукта Platform V Audit SE (AUD)

AUDIT2_PROXY_URL: целевой URL компонента Аудит (AUDT) продукта Platform V Audit SE (AUD)

############################## Интеграция с компонентом Прикладной Журнал (APLJ) продукта Platform V Backend (#BD)
KAFKA_PJ_PORT: порт Kafka Application Journal, обычно 9093
KAFKA_PJ_HOST_3: хост сервера №1 Kafka Application Journal
KAFKA_PJ_HOST_2: хост сервера №2 Kafka Application Journal
KAFKA_PJ_HOST_1: хост сервера №3 Kafka Application Journal
KAFKA_PJ_HOST_4: хост сервера №4 Kafka Application Journal
KAFKA_PJ_HOST_5: хост сервера №5 Kafka Application Journal
KAFKA_PJ_HOST_6: хост сервера №6 Kafka Application Journal
KAFKA_PJ_IP_1: ip адрес сервера Kafka №1 Application Journal
KAFKA_PJ_IP_2: ip адрес сервера Kafka №2 Application Journal
KAFKA_PJ_IP_3: ip адрес сервера Kafka №3 Application Journal
KAFKA_PJ_IP_4: ip адрес сервера Kafka №4 Application Journal
KAFKA_PJ_IP_5: ip адрес сервера Kafka №5 Application Journal
KAFKA_PJ_IP_6: ip адрес сервера Kafka №6 Application Journal
KAFKA_PJ_BOOTSTRAP: список ip:порт всех серверов Kafka Application Journal через запятую
STANDIN_CLIENT_ZONEID: зона USS в Application Journal
STANDIN_CLIENT_REFRESH_STATE_PERIOD: частота обновления состояния в Application Journal в секундах, обычно 100
STANDIN_CLIENT_CONCURRENCY: количество потоков для репликации журналов, обычно 10
STANDIN_CLIENT_PRODUCER_SSL_KEYSTORE_LOCATION: путь к keystore с сертификатами для доступа producer'а к Kafka Application Journal, обычно /opt/certs/kafka/keystore.jks
STANDIN_CLIENT_PRODUCER_SSL_TRUSTSTORE_LOCATION: путь к truststore с сертификатами для доступа producer'а к Kafka Application Journal, обычно /opt/certs/kafka/truststore.jks
STANDIN_CLIENT_PRODUCER_SSL_IDENTIFICATION_ALGORITHM: алгоритм идентификации producer'а в Kafka Application Journal, для автоматического выбора оставляется пустым
STANDIN_CLIENT_CONSUMER_SSL_KEYSTORE_LOCATION: путь к keystore с сертификатами для доступа consumer'а к Kafka Application Journal, обычно /opt/certs/kafka/keystore.jks
STANDIN_CLIENT_CONSUMER_SSL_TRUSTSTORE_LOCATION: путь к truststore с сертификатами для доступа consumer'а к Kafka Application Journal, обычно /opt/certs/kafka/truststore.jks
STANDIN_CLIENT_CONSUMER_SSL_IDENTIFICATION_ALGORITHM: алгоритм идентификации consumer'а в Kafka Application Journal, для автоматического выбора оставляется пустым

```
##################################### ISTIO
PROXY_IMAGE: адрес образа istio proxyv2

############################## INGRESS
ISTIO_CONTROL_PLANE: адрес контрольной панели istio
INGRESSGATEWAY_NAME: название gateway на ingress'е, обычно dp-uss-app-ingress
INGRESSGATEWAY_ISTIO_LABEL: лейбл для сущностей ingress istio, обычно
<название-namespace>-uss-<rb/kb>-ingress
INGRESSGATEWAY_APP_LABEL: лейбл для прикладных сущностей ingress, обычно
<название-namespace>-uss-<rb/kb>-ingress
INGRESSGATEWAY_CERTS_SECRET_NAME: название серверных секретов для ingress
gateway, обычно secret-ingressgateway-certs
INGRESSGATEWAY_CA_CERTS_SECRET_NAME: название секретов с УЦ для ingress
gateway, обычно secret-ingressgateway-ca-certs
INGRESSGATEWAY_CERTS_MOUNT_PATH: путь к серверным секретам для ingress
gateway, обычно /etc/istio/ingressgateway-certs
INGRESSGATEWAY_CA_CERTS_MOUNT_PATH: путь к секретам с УЦ для ingress gateway,
обычно /etc/istio/ingressgateway-ca-certs

############################## EGRESS
EGRESSGATEWAY_NAME: название gateway на egress'е, обычно dp-uss-app-egress
EGRESSGATEWAY_ISTIO_LABEL: лейбл для сущностей egress istio, обычно
<название-namespace>-uss-<rb/kb>-egress
EGRESSGATEWAY_APP_LABEL: лейбл для прикладных сущностей egress, обычно
<название-namespace>-uss-<rb/kb>-egress
EGRESSGATEWAY_CERTS_SECRET_NAME: название серверных секретов для egress
gateway, обычно  secret-ingressgateway-certs
EGRESSGATEWAY_CA_CERTS_SECRET_NAME: название секретов с УЦ для egress
gateway, обычно secret-ingressgateway-ca-certs
EGRESSGATEWAY_CERTS_MOUNT_PATH: путь к серверным секретам для egress gateway,
обычно /etc/istio/ingressgateway-certs
EGRESSGATEWAY_CA_CERTS_MOUNT_PATH: путь к секретам с УЦ для egress gateway,
обычно /etc/istio/ingressgateway-ca-certs

##################################### Компонент One-Time Password / One-
Time-Token (OTTS) продукта Platform V Backend (#BD)

OTT_SERVICE_HOSTS: список хост:порт всех серверов OTT через запятую
OTT_SERVICE_URL: основной адрес сервиса OTT
OTT_CERTSTORE_PATH: путь к keystore с сертификатами для доступа к OTT, обычно
/mnt/secrets/error-processor-cloud.p12
OTT_TRUST_STORE_PATH: путь к truststore с сертификатами для доступа к OTT,
обычно /mnt/secrets/ott_service_truststore.p12
OTT_CLIENT_CERT_ALIAS: название сервиса в компоненте One-Time Password / One-
Time-Token (OTTS) продукта Platform V Backend (#BD), обычно error-processor-
cloud
OTT_SERVICE_SI_HOSTS: хост:порт standin компонента One-Time Password / One-
Time-Token (OTTS) продукта Platform V Backend (#BD)
OTT_MODULE_ID: идентификатор модуля в компоненте One-Time Password / One-
Time-Token (OTTS) продукта Platform V Backend (#BD), обычно error-processor-
```

```
cloud
OTT_IMAGE: адрес образа ott-client-api-v2

#################################### Квота USS
USS_REPLICA_COUNT: количество реплик приложения, обычно 1
USS_CPU_LIMIT: количество ядер, выделяемых приложению из квоты, обычно 1
USS_MEM_LIMIT: количество оперативной памяти, выделяемых приложению из квоты,
обычно 6000Mi
USS_CPU_REQ: максимальное количество ядер, выделяемых приложению из квоты,
обычно 1
USS_MEM_REQ: максимальное количество оперативной памяти, выделяемых
приложению из квоты, обычно 6000Mi
SERVER_PORT: порт, на котором поднято приложение внутри контейнера, 8080
USS_CPU_ISTIO_REQ: количество ядер, выделяемых istio sidecar из квоты, обычно
100m
USS_CPU_ISTIO_LIM: максимальное количество ядер, выделяемых istio sidecar из
квоты, обычно 100m
USS_MEM_ISTIO_REQ: количество оперативной памяти, выделяемых istio sidecar из
квоты, обычно 1000Mi
USS_MEM_ISTIO_LIM: максимальное количество оперативной памяти, выделяемых
istio sidecar из квоты, обычно 1000Mi
USS_MEM_FB_LIMIT: количество оперативной памяти, выделяемых fluent-bit из
квоты, обычно 32Mi
USS_MEM_FB_REQ: максимальное количество оперативной памяти, выделяемых
fluent-bit из квоты, обычно 32Mi
USS_CPU_FB_REQ: количество ядер, выделяемых fluent-bit из квоты, обычно 200m
USS_CPU_FB_LIMIT: максимальное количество ядер, выделяемых fluent-bit из
квоты, обычно 200m

############################## DATABASES
MASTER_DATABASE_IP: ip адрес основной БД
MASTER_DATABASE_PORT: порт основной БД
MASTER_DATABASE_HOST: хост основной БД
STANDIN_DATABASE_HOST: хост standin БД
STANDIN_DATABASE_PORT: порт standin БД
STANDIN_DATABASE_IP: ip адрес standin БД

#################################### Настройки USS-APP
SERVER_PORT: порт, на котором поднимется приложение внутри контейнера, 8080
POSTGRES_ENABLED: включение/выключение режима работы с PGSE Platform V
SUDIR_ENABLED: включение/выключение режима работы с компонентом IAM Proxy
(AUTH) продукта Platform V IAM SE (IAM)
DATASOURCE_MAIN_CLASS: класс драйвера для основного datasource, для PGSE
Platform V - org.postgresql.Driver
DATASOURCE_MAIN_URL: URL основной БД
INT_MASTER_DATABASE_PORT: порт основной БД
DATASOURCE_MAIN_USER: имя пользователя в основной БД
DATASOURCE_MAIN_MAXPOOLSIZE: максимальный размер пула соединений для основной
бд, обычно 100
DATASOURCE_MAIN_MINPOOLSIZE: минимальный размер пула соединений для основной
```

```
бд, обычно 60
DATASOURCE_STANDIN_CLASS: класс драйвера для standin datasource, для PGSE
Platform V - org.postgresql.Driver
DATASOURCE_STANDIN_URL: URL standin БД
INT_STANDIN_DATABASE_PORT: порт standin БД
DATASOURCE_STANDIN_USER: имя пользователя в standin БД
DATASOURCE_STANDIN_MINPOOLSIZE: минимальный размер пула соединений для
standin бд, обычно 60
DATASOURCE_STANDIN_MAXPOOLSIZE: максимальный размер пула соединений для
standin бд, обычно 100
SPAS_SERVER_URL: адрес компонента Объединенный сервис авторизации (AUTZ)
продукта Platform V IAM SE (IAM)
SPAS_MAX_CLIENT_POOL_SIZE: количество потоков для клиента ОСА, обычно 10
SEND_MESSAGE_TO_JOURNAL_USS: включение/выключение репликации данных через
Application Journal
STANDIN_CLIENT_USS_APP_ZONEID: зона USS в Application Journal
USS_SUDIR_ROUTE: название route для входа через компонент IAM Proxy (AUTH)
продукта Platform V IAM SE (IAM)
USS_SUDIR_GEO_ROUTE: название route с поддержкой георезервирования для входа
через компонент IAM Proxy (AUTH) продукта Platform V IAM SE (IAM)
SPAS_HOSTNAME: хост сервиса ОСА
SPAS_PORT: порт сервиса ОСА
SPAS_IP: ip адрес сервиса ОСА
SPAS_KEYSTORE_PKEY: путь к приватному ключу к keystore компонента
Объединенный сервис авторизации (AUTZ) продукта Platform V IAM SE (IAM),
обычно оставляется пустым
SPAS_KEYSTORE_CERT: путь к keystore компонента Объединенный сервис
авторизации (AUTZ) продукта Platform V IAM SE (IAM), обычно оставляется
пустым
SPAS_TRUSTED_CERT: путь к truststore компонента Объединенный сервис
авторизации (AUTZ) продукта Platform V IAM SE (IAM), обычно оставляется
пустым

############################## FLUENT
FLUENTBIT_IMAGE: адрес образа fluent-bit
LOGGER_HOST: хост сервиса Logger Platform V
LOGGER_PORT: порт сервиса Logger Platform V
STAND_VERSION: идентификатор стенда для Logger Platform V

############################## Дополнительные настройки
IGNITE_CLIENT_ENABLED: включение/выключение режима работы с Ignite SE
Platform V
SAME_DB_ENABLED: настройка топологии БД, должна быть всегда false
CONTOUR_ID: идентификатор контура КБ/РБ
SUDIR_LOGOUT_URL: URL страницы логина сервера компонента IAM Proxy (AUTH)
продукта Platform V IAM SE (IAM)
AUDIT_DEV_MODE: включение/выключение интеграции с компонентом Аудит (AUDT)
продукта Platform V Audit SE (AUD)
STANDIN_CLOUD_CLIENT_STUB: включение/выключение интеграции с Application
Journal
```

```
OTT_CERTSTORE_TYPE: тип keystore для OTT, обычно PKCS12
AUDIT_CLIENT_CRT: имя файла сертификата для компонента Аудит (AUDT) продукта
Platform V Audit SE (AUD), обычно tls.crt
AUDIT_CLIENT_KEY: имя файла приватного ключа сертификата для компонента
Компонент Аудит (AUDT) продукта Platform V Audit SE (AUD), обычно tls.key
NOTIFICATIONS_ENABLED: включение/выключение отправки уведомлений
NOTIFICATIONS_KEYSTORE_PATH: путь к keystore с сертификатами почтового
сервера, обычно /opt/certs/kafka/keystore.jks
NOTIFICATIONS_TRUSSTORE_PATH: путь к truststore с сертификатами почтового
сервера, обычно /opt/certs/kafka/truststore.jks
NOTIFICATIONS_MAIL_USERNAME: логин в почтовом сервере
NOTIFICATIONS_MAIL_FROM: почтовый адрес, с которого будет осуществляться
рассылка уведомлений по почте
NOTIFICATIONS_SMS_USERNAME: логин в сервисе CMC
NOTIFICATIONS_SMS_FROM: адрес, с которого будет осуществляться рассылка
уведомлений по CMC
NOTIFICATIONS_SMS_TO: целевой адрес для рассылки уведомлений по CMC
NOTIFICATIONS_SMTPS_DEBUG_ENABLED: включение/выключение подробного
логирования для отправки уведомлений
NOTIFICATIONS_SMTPS_HOST: адрес SMTP почтового сервера
NOTIFICATIONS_SMTPS_PORT: порт SMTP почтового сервера, обычно 587
NOTIFICATIONS_SMTPS_PORT_CM: внешний порт для отправки уведомлений, обычно
8443

CORRSCEN_PORT: целевой порт для вызова корректирующих сценариев, обычно 443
CORRSCEN_HOST_1: хост для вызова корректриующего сценария №1
```

Интеграция с компонентом Объединенный мониторинг Unimon (MONA) продукта Platform V Monitor (OPM)

Данный компонент предназначен для сбора прикладных и инфраструктурных метрик и отправки их в целевую систему хранения. С детальным описанием можно ознакомиться в документации к продукту Platform V Monitor (OPM).

Интеграция с компонентом Журналирование (LOGA) продукта Platform V Monitor (OPM)

Для интеграции с компонентом корректно заполните enviroment файл стенда (см. документацию для компонента Журналирование (LOGA) продукта Platform V Monitor (OPM)). За настройки подключения к компоненту отвечают параметры:

- LOGGER_GW: – предпочитаемый хост логгера;
- LOGGER_ENDP: - нода балансировки;
- LOGGER_ENDP_2 - нода балансировки;
- LOGGER_PORT: - порт логгера (443);
- LOGGER_PROTO: - протокол HTTP/HTTPS (HTTPS).

Больше никаких действий для настройки интеграции с данным компонентом не требуется.

Pipeline DevOps для развертывания проекта Продукта в среду контейнеризации Kubernetes или Red Hat OpenShift 4+ (опционально)

Запуск осуществляется в задании Jenkins типа pipeline, definition у которого выставлен как Jenkinsfile из данного репозитория. Данный Pipeline DevOps можно сразу настроить на все стенды среды контейнеризации Kubernetes или Red Hat OpenShift 4+ (опционально), для достижения универсальности созданных template.

Секреты стендов

Каждый стенд может иметь уникальный список секретов, например, пароли в БД. Внутри Git секреты сохранены в виде зашифрованных с помощью ansible-vault файлов. Является опциональным шагом, т.к. для минимальной конфигурации достаточно двух секретов (к основной и SI БД), указываемые в Jenkins, указываемые с ID секрета: MAIN_DB, STANDIN_DB.

При создании файлов секретов, все параметры, передаваемые внутри value должны быть зашифрованы в base64.

Существует отдельный pipe, который на входе имеет четыре параметра:

- gitSshUrl – репозиторий, в котором временно находятся секреты, которые необходимо шифровать;
- gitBranch – ветка репозитория, в которой находятся секреты;
- folder – папка, в которой находятся секреты;
- workMode – режим работы скрипта (encrypt – шифруем, decrypt – расшифровываем).

В результате получаем архив, прикрепленный к результату сборки, в котором находятся зашифрованные или расшифрованные секреты (в зависимости от типа задачи), которые нужно подложить в репозиторий в папку secrets/example, где example – имя стенда, для которого предназначаются секреты.

Jenkinsfile

В Jenkinsfile находится сам скрипт, в котором объявлен так же ряд переменных.

Сборка артефакта продукта через Jenkins

Внешний вид задания Jenkins (с примером заполнения):

Parameters

AGENT

clearAgent

Jenkins агент для исполнения job

jdk

openjdk-11.0.2_linux

Версия jdk пример: openjdk-1.8.0-linux

maven

Maven 3.6.2

Версия maven пример: Maven 3.5.4

OWNED_DISTR_URL

Ссылка на owned дистрибутив в Nexus

PUBLISH_URL

Ссылка на Nexus репозиторий для загрузки дистрибутива

PARTY_DISTR_URL

Ссылка на party дистрибутив в Nexus

DOCKER_REGISTRY

Docker репозиторий

DOCKER_REGISTRY_PATH

Путь к дистрибутиву в Docker репозитории

DOCKER_DIR

Директория для хранения дистрибутива в Docker репозитории

MAVEN_XML

MavenSetting

Id Maven Setting.xml файла в Jenkins

nexusSecrets

Id Nexus секрета в Jenkins

baseImageJava

Базовый docker образ Java

VERSION

4.3.27

Версия релиза

ARTIFACT_ID

ID артефакта

GROUP_ID

ID группы

REPOSITORY_ID

ID репозитория

MODULE

Выберите модуль для сборки

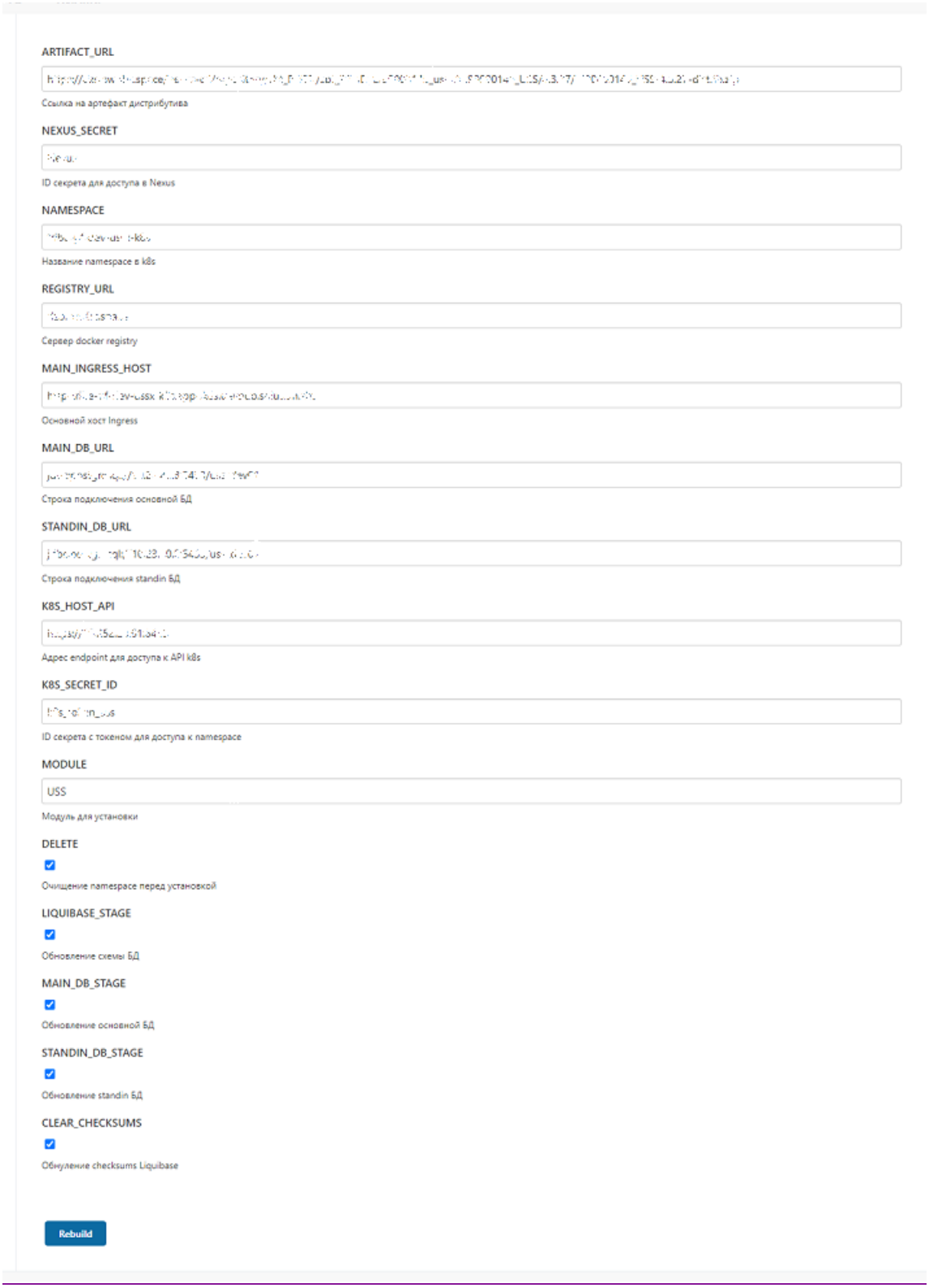
☐ OPENSHIFT

Параметры:

- AGENT – Jenkins-агент для исполнения задания Jenkins;
- jdk – версия jdk;
- maven – версия maven;
- OWNED_DISTR_URL – ссылка на owned дистрибутив в Nexus-Public;
- PUBLISH_URL – ссылка на Nexus-Public репозиторий для загрузки дистрибутива;
- PARTY_DISTR_URL – ссылка на party дистрибутив в Nexus-Public;
- DOCKER_REGISTRY – Docker репозиторий;
- DOCKER_REGISTRY_PATH – путь к дистрибутиву в Docker репозитории;
- DOCKER_DIR – директория для хранения дистрибутива в Docker репозитории;
- MAVEN_XML – ID Maven Setting.xml файла в Jenkins;
- nexusSecrets – ID Nexus-Public секрета в Jenkins;
- baselmageJava – базовый Docker образ Java;
- VERSION – версия релиза;
- ARTIFACT_ID – ID артефакта;
- GROUP_ID – ID группы;
- REPOSITORY_ID – ID репозитория;
- MODULE – модуль для сборки;
- OPENSHIFT – среда контейнеризации (опционально).

Развертывание Продукта через Jenkins

Внешний вид задания Jenkins (с примером заполнения):


ARTIFACT_URL
Ссылка на артефакт дистрибутива
NEXUS_SECRET
ID секрета для доступа в Nexus
NAMESPACE
Название namespace в k8s
REGISTRY_URL
Сервер docker registry
MAIN_INGRESS_HOST
Основной хост Ingress
MAIN_DB_URL
Строка подключения основной БД
STANDIN_DB_URL
Строка подключения standin БД
K8S_HOST_API
Адрес endpoint для доступа к API k8s
K8S_SECRET_ID
ID секрета с токеном для доступа к namespace
MODULE
USS
Модуль для установки
DELETE
Очищение namespace перед установкой
LIQUIBASE_STAGE
Обновление схемы БД
MAIN_DB_STAGE
Обновление основной БД
STANDIN_DB_STAGE
Обновление standin БД
CLEAR_CHECKSUMS
Обнуление checksums Liquibase
Rebuild

Параметры:

- ARTIFACT_URL – ссылка на артефакт дистрибутива;
- NEXUS_SECRET – ID секрета для доступа в Nexus-Public;
- NAMESPACE – название пространства имен в k8s;
- REGISTRY_URL – ссылка на Docker registry;
- MAIN_INGRESS_HOST – основной хост Ingress;
- MAIN_DB_URL – строка подключения основной БД;
- STANDIN_DB_URL – строка подключения StandIn БД;
- K8S_HOST_API – адрес endpoint для доступа к API k8s;
- K8S_SECRET_ID – ID секрета с токеном для доступа к пространству имен;
- MODULE – модуль для установки;
- DELETE – очищение пространства имен перед установкой;
- LIQUIBASE_STAGE – обновление схемы БД;
- MAIN_DB_STAGE – обновление основной БД;
- STANDIN_DB_STAGE – обновление StandIn БД;
- CLEAR_CHECKSUMS – обнуление checksums Liquibase.

СУБД PSQL

Чтобы подготовить базу данных, выполните следующие действия:

- В СУБД PSQL проверьте наличие ролей as_admin, as_TUZ. При разворачивании в СУБД PSQL роли входят в поставку. При использовании СУБД PSQL, предварительно создать роли (см. скрипт создания ролей).

```
-- скрипт проверки на наличие ролей
SELECT rolname FROM pg_roles WHERE rolname in ('as_TUZ', 'as_admin');

-- скрипт создания ролей
CREATE ROLE as_admin;
CREATE ROLE "as_TUZ";
```

- Создайте пользователя владельца, от имени которого выполняется накат скриптов по разворачивании БД:

```
-- создание пользователя USS
CREATE USER USS WITH
    PASSWORD '***************'
    LOGIN
    NOCREATEDB
    NOCREATEROLE
    NOSUPERUSER
    NOINHERIT
    NOREPLICATION
    NOBYPASSRLS;

-- Устанавливаем порядок поиска схем для пользователя
ALTER ROLE USS SET search_path = "$user";
```

```
GRANT as_admin TO USS;
ALTER ROLE USS SET role TO 'as_admin';
```

- Создайте табличные пространства:

```
-- на сервере создать папки для пространств
mkdir /mnt/uss_l
mkdir /mnt/uss_t

--в СУБД выполнять пользователем с правами db_admin (PSQL)

CREATE TABLESPACE uss_t
OWNER as_admin
LOCATION '/pgdata/11/';

CREATE TABLESPACE uss_l
OWNER as_admin
LOCATION '/pgdata/11/';

GRANT CREATE ON TABLESPACE uss_t TO as_admin;
GRANT CREATE ON TABLESPACE uss_l TO as_admin;

GRANT CREATE ON TABLESPACE uss_t TO "as_TUZ";
GRANT CREATE ON TABLESPACE uss_l TO "as_TUZ";
```

- Создайте БД:

```
--выполнять пользователем с правами db_admin (uss)
CREATE DATABASE uss
WITH
OWNER = db_admin
ENCODING = 'UTF8'
LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8'
TABLESPACE = uss_l
CONNECTION LIMIT = -1;

GRANT CREATE ON DATABASE uss TO as_admin;
```

- Создайте схемы:

```
-- Схема для владельца схемы приложения
CREATE SCHEMA uss AUTHORIZATION as_admin;
GRANT ALL ON SCHEMA uss TO as_admin;
GRANT USAGE ON SCHEMA uss TO "as_TUZ";
```

- Предоставьте дефалтовые привилегии на схему для ролей as_admin, as_TUZ:

```
GRANT USAGE ON SCHEMA uss TO "as_TUZ";
GRANT SELECT, UPDATE, INSERT, DELETE ON ALL TABLES IN SCHEMA uss TO "as
_TUZ";
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA uss TO "as_TUZ";
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA uss TO "as_TUZ";
```

```
GRANT EXECUTE ON ALL ROUTINES IN SCHEMA uss TO "as_TUZ";
GRANT EXECUTE ON ALL PROCEDURES IN SCHEMA uss TO "as_TUZ";
ALTER DEFAULT PRIVILEGES FOR ROLE as_admin IN SCHEMA uss GRANT SELECT,
INSERT, UPDATE, DELETE ON TABLES TO "as_TUZ";
ALTER DEFAULT PRIVILEGES FOR ROLE as_admin IN SCHEMA uss GRANT ALL PRIV
ILEGES ON SEQUENCES TO"as_TUZ";
ALTER DEFAULT PRIVILEGES FOR ROLE as_admin IN SCHEMA uss GRANT EXECUTE
ON FUNCTIONS TO"as_TUZ";
ALTER DEFAULT PRIVILEGES FOR ROLE as_admin IN SCHEMA uss GRANT EXECUTE
ON ROUTINES TO"as_TUZ";
ALTER DEFAULT PRIVILEGES FOR ROLE as_admin IN SCHEMA uss GRANT USAGE ON
TYPES TO "as_TUZ";

GRANT ALL PRIVILEGES ON SCHEMA uss TO "as_admin";
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA uss TO "as_admin";
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA uss TO "as_admin";
GRANT ALL PRIVILEGES ON ALL FUNCTIONS IN SCHEMA uss TO "as_admin";
GRANT ALL PRIVILEGES ON ALL ROUTINES IN SCHEMA uss TO "as_admin";
GRANT ALL PRIVILEGES ON ALL PROCEDURES IN SCHEMA uss TO "as_admin";
ALTER DEFAULT PRIVILEGES FOR ROLE as_admin IN SCHEMA uss GRANT ALL PRIV
ILEGES ON TABLES TO"as_admin";
ALTER DEFAULT PRIVILEGES FOR ROLE as_admin IN SCHEMA uss GRANT ALL PRIV
ILEGES ON SEQUENCES TO"as_admin";
ALTER DEFAULT PRIVILEGES FOR ROLE as_admin IN SCHEMA uss GRANT ALL PRIV
ILEGES ON FUNCTIONS TO"as_admin";
ALTER DEFAULT PRIVILEGES FOR ROLE as_admin IN SCHEMA uss GRANT ALL PRIV
ILEGES ON ROUTINES TO"as_admin";
ALTER DEFAULT PRIVILEGES FOR ROLE as_admin IN SCHEMA uss GRANT ALL PRIV
ILEGES ON TYPES TO"as_admin";
```

- Создайте пользователя системы uss_appl:

```
--Выполнять пользователем с правами dba
CREATE USER uss_appl WITH
    PASSWORD '***************'
    LOGIN
    NOCREATEDB
    NOCREATEROLE
    NOSUPERUSER
    NOINHERIT
    NOREPLICATION
    NOBYPASSRLS;

ALTER ROLE uss_appl SET search_path = uss;
ALTER ROLE uss_appl SET role TO "as_TUZ";
GRANT "as_TUZ" TO uss_appl;
```

Обновление

Обновление Продукта осуществляется стандартным образом аналогично установке с актуализацией env файлов, в случае необходимости, а также изменением версии релиза в параметрах развертывания задания Jenkins. Установку новой версии Продукта необходимо осуществить согласно описанным шагам в пункте *Установка дистрибутива* текущего документа. Дополнительных настроек не требуется.

Удаление

Удаление Docker Images выполняется стандартными механизмами Kubernetes или Red Hat OpenShift 4+ (опционально).

Проверка работоспособности

В данный момент работоспособность компонент определяется путем запроса на сервере приложения http://host/component-name/actuator/health, (это ссылка, по которой можно получить статус приложения “OK” или не “OK”). Где:

- host – адрес сервера;
- component-name – имя компонента.

http://auth-api/actuator/health

http://app-api/actuator/health

http://inner-connector/actuator/health

В том числе для проверки работоспособности веб-интерфейса Продукта необходимо:

1. Успешно войти в АРМ продукта под пользователем, которому назначена определенная роль.

2. Перейти по всем разделам сайдбара Продукта и убедиться, что при переходе отсутствуют ошибки.

После установки компонентов необходимо проверить отсутствие ошибок в системных журналах подов компонента в среде контейнеризации Kubernetes или Red Hat OpenShift 4+ (опционально) и, в случае инсталляции в конфигурации с включенным компонентом Граничный прокси (IGEG) продукта Platform V Synapse Service Mesh (SSM) (является опциональным), проверить в системных журналах подов компонента Граничный прокси (IGEG) продукта Platform V Synapse Service Mesh (SSM) наличие http-запросов со статусом отличным от 200.

Откат

Для отката Продукта к ранним версиям ПО после установки обновлений необходимо восстановить предыдущие версии Docker Images, либо путем включения

предыдущего Replication Controller (предыдущая версия конфигурации в среде контейнеризации Kubernetes или Red Hat OpenShift 4+ (опционально)), либо путем наката предыдущего релиза.

1. Откат к предыдущей версии представляет собой удаление и установку последней стабильной версии (запуск задания Jenkins установки с параметром delete и указанием предыдущей версии релиза).

2. Для отката необходимо выполнить установку предыдущей стабильной версии в соответствии с инструкцией, пункт *Установка дистрибутива* текущего документа.

3. Возможен откат БД средствами СУБД PostgreSQL (рекомендуется использование продукта Platform V Pangolin SE (PSQ)).

Часто встречающиеся проблемы и пути их устранения

При ошибке установки SQL-скриптов посмотрите лог сборки в Jenkins на предмет ошибки, проверьте подключение к БД, выполните необходимые правки в common репозитории Pipeline DevOps или репозитории с конфигурацией Продукта, если ошибка связана с настройками, или обратитесь к команде разработки Продукта для внесения изменений в SQL-скрипты и пересборки дистрибутива.

При ошибке установки в среде контейнеризации Kubernetes или Red Hat OpenShift 4+ (опционально) посмотрите лог сборки в Jenkins на предмет ошибки, выполните необходимые правки в common репозитории Pipeline DevOps или репозитории с конфигурацией Продукта, если ошибка связана с настройками, или обратитесь к команде разработки Продукта для внесения изменений в SQL-скрипты и пересборки дистрибутива.

При ошибке авторизации убедитесь, что пользователь заведен в компоненте IAM Proxy (AUTH) продукта Platform V IAM SE (IAM).

Чек-лист валидации установки

При установке необходимо проверить результат выполнения следующих плейбуков:

- DB_UPDATE – проверить, что применена последняя миграция скриптов из папки liquibase/scripts дистрибутива;
- OPENSHIFT_DEPLOY – проверить, что в проектной области среды контейнеризации Kubernetes или Red Hat OpenShift 4+ (опционально) появились ConfigMaps, VirtualService, DeploymentConfig, Service, Route, ServiceEntry, Gateway, DestinationRule и создались POD;
- OPENSHIFT_INGRESS_EGRESS_DEPLOY – проверить, что в проектной области среды контейнеризации Kubernetes или Red Hat OpenShift 4+ (опционально), появились POD INGRESS и EGRESS.