Руководство по установке
продукта «Platform V Exchange Rates»
Код компонента: CRT
Поколение 4

ОГЛАВЛЕНИЕ

Руководство по установке

Термины и определения

В таблице приведены основные аббревиатуры и сокращения:

Термин	**Описание**
GitLab, Git	Веб-сервис для проектов и их совместной разработки
CPU	Central Processing Unit, центральный процессор
Istio	Настраиваемая сервисная сеть (service mesh) с открытым исходным кодом, служащая для взаимодействия, мониторинга и обеспечения безопасности контейнеров в кластере Kubernetes. Platform V Synapse Service Mesh или istio
Platform V Synapse Event Transfer Service	Распределенная потоковая платформа
OTT	Platform V One-Time-Token. Сервис для выпуска и подтверждения действительности выпущенных ключей
Synapse, Platform V Synapse Service Mesh	Управляет взаимодействием между сервисами или микро-сервисами
АС	Автоматизированная система
БД	База данных
ЕКПиТ	Единый каталог продуктов и тарифов
ЗНО	Запрос на обслуживание

ОС	Операционная система
ПЖ	компонент Прикладной журнал (APLJ) продукта Platform V Data Tools
Продукт	Platform V Exchange Rates
Среда контейнеризации	Kubernetes (рекомендуется), поддержана опциональная совместимость с OpenShift 4+
ТУЗ	Техническая учетная запись
УЦ	Удостоверяющий центр

Системные требования

В данном разделе перечислены пре−реквизиты и требования к окружению для установки программного продукта, в том числе рекомендуемые настройки безопасности окружения. Также представлен перечень внешних продуктов, используемых для установки, настройки и контроля с указанием выполняемых ими функций.

Пререквизиты

Перед установкой **Продукта** убедитесь, что выполнены все следующие условия:

1. Обязательные условия

- ОС Linux (рекомендован «Альт СП 8»)
- для продукта выделены области в среде контейнеризации;
- развернута СУБД (PostgreSQL или Platform V Pangolin SE).
- выделенные области включены под управление service mesh (istio). Для подключения проекта к istio воспользуйтесь Synapse FAQ;
- в наличии репозиторий Git;
- jenkins с заведенными секретами для доступа к Git, Хранилищу артефактов и Докеру Хранилища;
- в проекте среды контейнеризации присутствует секрет для доступа к приватному хранилищу контейнеров (docker-regestry);
- в проекте среды контейнеризации присутствует сервисный аккаунт (Service Account) и созданы роли для разворачивания в namespaces (project) приложения (типовые роли Deploy или Jenkins CD).

2. Опциональные компоненты

- в окружении развернут компонент Platform V One-Time-Token;
- в окружении развернут компонент Прикладной журнал продукта Platform V Data Tools;
- в окружении развернут компонент Аудит (AUDT) продукта Platform V Audit SE;
- в окружении развернут продукт Platform V Synapse Service Mesh;
- в окружении развернут компонент Журналирование (LOGA) продукта Platform V Monitor;
- БД справочников (ЕКПиТ), а также пользователь и пароль к ней. Необходимы права на чтение (предполагается, что все действия по инициализации схемы данных пользователя были сделаны заранее, а также таблицы в БД не пустые.

Требования

Для пространства среды контейнеризации необходим проект с характеристиками 16 CPU и 32 GB memory.

Используемые внешние продукты

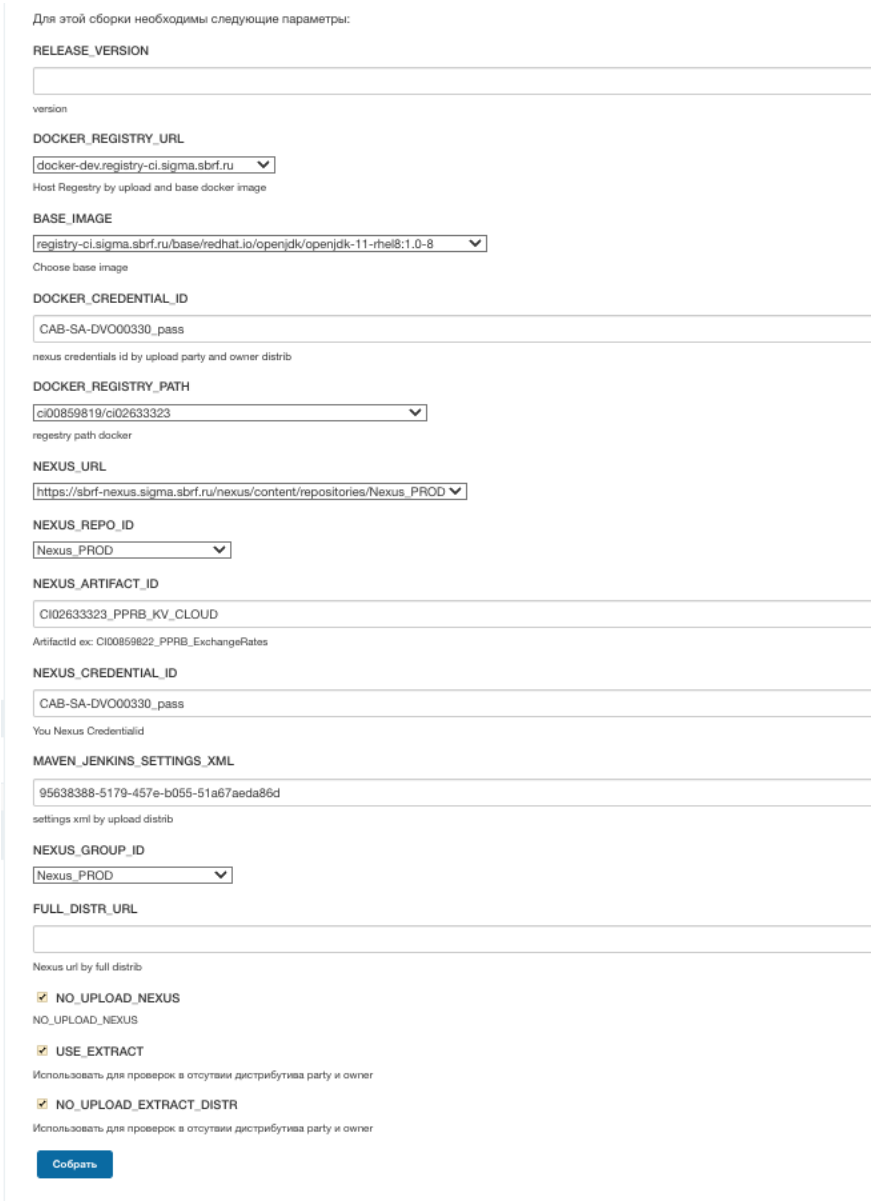
Внешний продукт	**Назначение**
Jenkins	Запускает и выполняет процедуру установки
Git хранилище	Репозиторий для хранения скрипта процедуры установки, секретов среды контейнеризации, параметров установки, хранилищ сертификатов, ключей, паролей к ним для тестов и процесса обновления структуры базы данных
Nexus-Public	Хранение дистрибутивов продукта
Command line tool (kubectl)	Клиент среды контейнеризации для выполнения команд по созданию, модификации, получению, удалению ресурсов среды контейнеризации
Liquibase	Программа для обновления базы данных

Подготовка дистрибутива

Дистрибутив представляет собой zip-файл, опубликованный в хранилище. Дистрибутив содержит разделенный целевой дистрибутив Party-distrib.zip. Включает набор библиотек и зависимостей распространяющихся свободно.

Owned-distrib.zip – содержит основной код продукта. В party-distrib в папке Jenkins расположена джоба ReCreateFullDistrib.groovy". Ее необходимо развернуть, опубликовать в git.

На рисунке представлены необходимые параметры для сборки.



1. При первом запуске нажмите на кнопку **Собрать**. В результате будет создан интерфейс для джобы. Реализована возможность корректировки джобы под свое окружение. Для этого в разделе parametrs задайте типовые настройки окружения (ссылка на docker registry, имя секрета в jeknis для доступа в registry).

```
parameters {
        // metadata
        string(name: 'RELEASE_VERSION',description: 'version')
        //docker
        choice(name: 'DOCKER_REGISTRY_URL',description:'Host Regestry by upload and base docker
image',
                choices: [
                    'docker-dev.registry-ci.sigma.sbrf.ru',
```

```
                    'docker-release.registry-ci.sigma.sbrf.ru',
                    'registry.delta.sbrf.ru',
                    'dzo.sw.sbc.space'
                ]
        )
        choice(name: 'BASE_IMAGE',
                choices: [
                    'registry-ci.sigma.sbrf.ru/base/redhat.io/openjdk/openjdk-11-rhel8:1.0-8',
                    'registry.delta.sbrf.ru/base/redhat/openjdk/openjdk-11-rhel8:1.0-8',
                    'dzo.sw.sbc.space/sbt_dev/ci90000087_asgtmq_dev/alt-sp8/openjdk-11:v0.1'
                        ],
                description: 'Choose base image')
        string(name: 'DOCKER_CREDENTIAL_ID', description: 'nexus credentials id by upload party and
owner distrib', defaultValue: 'CAB-SA-DVO00330_pass')
        choice(name: 'DOCKER_REGISTRY_PATH',
                choices:[
                    "ci00859819/ci02633323",
                    "docker-release/pprb/ci00859819/ci00859822_pprb-currency-rates",
                    "pprb/ci00859819/ci00859822_pprb-currency-rates",
                    "pprb-dev/ci00859819/ci00859822_pprb-currency-rates_dev",
                    "pprb/ci00859819/ci00859822_pprb-currency-rates",
                    "sbt_dev/ci90000071_crtx_dev",
                    "sbt/ci90000071_crtx"
                ],
                description: 'regestry path docker' )
        //nexus
        choice(name: 'NEXUS_URL',
                choices:[
                    "https://sbrf-nexus.sigma.sbrf.ru/nexus/content/repositories/Nexus_PROD",
                    "https://dzo.sw.sbc.space/nexus-cd/repository/sbt_PROD",
                                        "https://sbtnexus.delta.sbrf.ru",
                    "https://sbrf-nexus.ca.sbrf.ru"
                ],
                description: "" )
        choice(name: 'NEXUS_REPO_ID',
```

```
            choices:[
                "Nexus_PROD",
                "Nexus_PROD_OUT",
                "sbt_PROD",
                "SERVICE_PACKAGE_snapshot"

            ]
        )
        string(name: 'NEXUS_ARTIFACT_ID', description: 'ArtifactId ex: CI00859822_PPRB_ExchangeRates', defaultValue: 'CI02633323_PPRB_KV_CLOUD')
        string(name: 'NEXUS_CREDENTIAL_ID',description: 'You Nexus Credentialid', defaultValue: 'CAB-SA-DVO00330_pass')
        string(name: 'MAVEN_JENKINS_SETTINGS_XML', defaultValue: "95638388-5179-457e-b055-51a67aeda86d", description: "settings xml by upload distrib" )
        choice(name: 'NEXUS_GROUP_ID',
            choices:[
                "Nexus_PROD",
                "sbt_PROD.CI90000071_crtx.4g",
                "---"

            ]
        )
        // distrib
        string(name: 'FULL_DISTR_URL', description: 'Nexus url by full distrib')

        booleanParam( defaultValue: true,description: 'NO_UPLOAD_NEXUS',name: 'NO_UPLOAD_NEXUS')
        booleanParam( defaultValue: true, description:'Использовать для проверок в отсутвии дистрибутива party и owner', name: 'USE_EXTRACT')
        booleanParam( defaultValue: true, description:'Использовать для проверок в отсутвии дистрибутива party и owner', name: 'NO_UPLOAD_EXTRACT_DISTR')

    }
```

2. Задайте метку Jenkins agenta, который существует в вашем окружении.
3. Заведите конфигурационный файл Settings.xml для maven.
4. В джобу передайте ссылки на текущий дистрибутив.
5. Выполните сборку.

Установка

Существует два способа установки Продукта Platform V Exchange Rates:

1. Целевой (основной) способ с помощью ручной установки.
2. Опциональный вариант с помощью компонента Deploy tools (CDJE) продукта Platform V DevOps Tools.

Подготовка к развертыванию

Для подготовки к развертыванию продукта выполните следующие шаги:

1. Создание секрета в jenkins для доступа в среду контейнеризации.
2. Создание задачи в jenkins по установке дистрибутива.
3. Создание секрета.
4. Создание файла _passwords.conf в репозитории common, применяется только при опциональной автоматизированной установке с помощью DevOps Tools (Смотри пункт разворачивание джобы компонента Deploy tools (CDJE) продукта Platform V DevOps Tools).
5. Создание файла зависимых параметров установки, применяется только при опциональной автоматизированной установке с помощью DevOps Tools (Смотри пункт разворачивание джобы компонента Deploy tools (CDJE) продукта Platform V DevOps Tools.

Ручная установка

Текущий тип установки является целевым (основным) вариантом установки. Для ручной установки:

1. Скачайте дистрибутив на персональный компьютер.
2. Разархивируйте дистрибутив.
3. Подготовьте директорию.

- Подготовьте yaml манифесты с секретами. Добавьте к именам секретов окончание .unver.
- Создайте yaml манифест с именем secret-crtx.unver и поместите все параметры для _password.conf в него. **Пример**

```
kind: Secret
apiVersion: v1
metadata:
  name: secret-crtx.unver
type: Opaque
data:
  DB_CURRATE_MAIN_LOGIN: <base64 enc >
  DB_CURRATE_MAIN_PASSWORD:  <base64 enc >
```

```
  # База данных CR SI
  DB_CURRATE_STANDIN_LOGIN:  <base64 enc >
  DB_CURRATE_STANDIN_PASSWORD:  <base64 enc >
  #
  DB_EKPIT_MAIN_LOGIN:  <base64 enc >
  DB_EKPIT_MAIN_PASSWORD:  <base64 enc >
  DB_EKPIT_STANDIN_LOGIN:  <base64 enc >
  DB_EKPIT_STANDIN_PASSWORD:  <base64 enc >
  #
  MESSENGER_KAFKA_SSL_KEY_STORE_PASSWORD:  <base64 enc >
  MESSENGER_KAFKA_SSL_TRUST_STORE_PASSWORD:  <base64 enc >
  # пароль от закртытого ключа
  MESSENGER_KAFKA_SSL_PRIVATE_KEY_PASSWORD:  <base64 enc >
  #
  OTT_CERTSTORE_PWD:  <base64 enc >
  OTT_TRUST_STORE_PWD:  <base64 enc >
  # пароль от закртытого ключа
  OTT_CERTSTORE_PRIVATE_KEY_PWD:  <base64 enc >
  #
  AJ_KEYSTORE_PASSWORD:  <base64 enc >
  AJ_TRUSTORE_PASSWORD:  <base64 enc >
  # пароль от закртытого ключа
  AJ_KEY_PRIVATE_KEY_PASSWORD: <base64 enc >
```

4. Установите все secret командой

```
kubectl apply -f <имя файла манифеста> -n <пространство имен для установки>
```

5. В результате будут созданы все секреты из блока extra-secrets в distib.yaml и secret-crtx.unver.
6. Шаблонизируйте в ручную yaml манифесты из дистрибутива.
7. Перейдите в директорию package/conf/k8s/base. В текущей директории расположены поддиректории отвечающие за каждый сервис. В директории istio расположены манифесты для настройки сервисной сети.

Для того чтобы шаблонизировать манифесты в каждой поддиректории воспользуйтесь ansible. **Пример ansible для шаблонизации:**

```
--
- hosts: 127.0.0.1_
```

```
  connection: local
  tasks:
  - name: "template"
    ansible.builtin.template:
      src : package/conf/k8s/base/crtx-cb-cloud-app/dc.yaml
      dest: ./<Директория для хранинния итоговых манифестов>
```

8. Далее вызовите playbook и передайте параметры соответствующему сервису:

```
ansible-playbook ./pipline.yaml --extra-vars "@package/conf/config/parameters/crtx-cb-cloud-app.all.conf" --extra-vars @/package/conf/config/parameters/crtx-istio-sidecar.all.conf  --extra-vars @/package/conf/custom_property.conf
```

9. В примере шаблонизируется сервис crtx-cb-cloud-app и ему передаются параметры с соответствующим именем, а также дополнительные параметры которые общие для всех файлов.
    10. В дистрибутиве находиться файл custom_property.conf.yml в текущий файл можно добавить недостающие переменные, которые напрямую подтягиваться с common репозитория.
    11. Результат работы ansible-playbook.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: crtx-cb-cloud-app
labels:
  app: crtx-cb-cloud-app
spec:
  progressDeadlineSeconds: 300
  replicas: 1
  revisionHistoryLimit: 1
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: "25%"
      maxUnavailable: "25%"
  selector:
    matchLabels:
```

```
      app: crtx-cb-cloud-app
  template:
    metadata:
      labels:
        app: crtx-cb-cloud-app
      annotations:
        sidecar.istio.io/inject: "True"
        sidecar.istio.io/rewriteAppHTTPProbers: "True"
    spec:
      restartPolicy: Always
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 100
            podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: app
                  operator: In
                  values:
                  - crtx-cb-cloud-app
              topologyKey: kubernetes.io/hostname
      containers:
      # app cb
        - resources:
            limits:
              cpu: "1"
              memory: "2Gi"
            requests:
              cpu: "1"
              memory: "2Gi"
          securityContext:
            readOnlyRootFilesystem: true
          readinessProbe:
            httpGet:
```

```
            path: /actuator/health/readiness
            port: 8081
            scheme: HTTP
          initialDelaySeconds: 160
          timeoutSeconds: 2
          periodSeconds: 10
          successThreshold: 1
          failureThreshold: 3
        livenessProbe:
          httpGet:
            path: /actuator/health/liveness
            port: 8081
            scheme: HTTP
          timeoutSeconds: 2
          periodSeconds: 10
          successThreshold: 1
          failureThreshold: 6
        startupProbe:
          failureThreshold: 50
          httpGet:
            path: /actuator/health/liveness
            port: 8081
            scheme: HTTP
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 2
        terminationMessagePath: /dev/termination-log
        name: crtx-cb-cloud-app
        env:
          - name: PROJECT_NAME
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: metadata.namespace
          - name: NODE_NAME
```

```
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: spec.nodeName
          - name: POD_NAMESPACE
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: metadata.namespace
          - name: POD_NAME
            valueFrom:
              fieldRef:
                apiVersion: v1
                fieldPath: metadata.name
        ports:
          - containerPort: 8080
            protocol: TCP
          - containerPort: 8081
            protocol: TCP
        imagePullPolicy: Always
        volumeMounts:
          - { name: temp, mountPath: /tmp }
          - { name: applogs, mountPath: /tmp/logs }
          - { name: logback-config, readOnly: true, mountPath: /app/etc/logger }
          - { name: aj-secrets, readOnly: true, mountPath: /app/certs/aj }
          - { name: configmap-cr-app, readOnly: true, mountPath: /app/etc }
        envFrom:
          - configMapRef:
              name: crtx-cm-cb-cloud-app
          - secretRef:
              name: secret-crtx.${distrib.release.version}
        image:  "test/test@###IMAGE###"
      # logger
    securityContext:
      runAsNonRoot: true
```

```
      imagePullSecrets:
        - name: "default"
      volumes:
        - { name: temp, emptyDir: {} }
        - { name: applogs, emptyDir: {} }
        - name: logback-config
          configMap:
            name: crtx-cm-cb-cloud-app-logback
            defaultMode: 256
        - name: aj-secrets
          secret:
            secretName: secret-aj.${distrib.release.version}
            defaultMode: 256
            optional: true
        - name: configmap-cr-app
          configMap:
            name: crtx-cm-cb-cloud-app
            defaultMode: 256
      dnsPolicy: ClusterFirst
```

12. Далее проверьте этот манифест и замените строки содержащие ${}. Так как efc pipline поддерживает различные механизмы шаблонизации, то могут быть переменные типа .${distrib.release.version} которые не были заменены.
13. Значение текущей переменной замените на unver, выполнив команду:

```
Kubectl apply -f <ваш обработаный. Манифест > -n <пространсвто имен в k8s>
```

Для успешного разворачивания любого сервиса разверните все манифесты в директории сервиса.

Для обеспечения безопасности передаваемых данных используется mTLS + OTT. Все входящие и исходящие межсервисные соединения проходят через ingress/egress proxy. Для входящих/исходящих взаимодействий используются mTLS.

Для исходящего трафика используется два сервиса **Egress-istio**:

- crtx-egress – используется для "полезного трафика" приложения выступает в качестве балансировщика и выходного шлюза на границе сервисной сети;
- crtx-egress-mon – используется для сервисного трафика, предназначен для балансирования и терминации трафика на границе сети, для служб и сервисов аудита, логирования, мониторинга.

Шаблон установки для crtx-egress-2-0 – istio/deployment/egress/*.yaml, Каждый из этих шаблонов разворачивает egress-envoy и OTT сервис, подключенный как sidecar. Для активации авторизации на envoy с применением OTT используется настройка envoyfiltra:

```
  - apiVersion: networking.istio.io/v1alpha3
    kind: EnvoyFilter
    metadata:
      name: crtx-ef-mon
    spec:
      configPatches:
        - applyTo: HTTP_FILTER
          match:
            context: GATEWAY
            listener:
              filterChain:
                filter:
                  name: envoy.http_connection_manager
              portNumber: 5443
          patch:
            operation: INSERT_BEFORE
            value:
              name: envoy.ext_authz
              config:
                failure_mode_allow: false
                grpc_service:
                  google_grpc:
                    stat_prefix: ext_authz
                    target_uri: unix:/mnt/ott-uds-socket/ott.socket
                  timeout: 1s
                with_request_body:
                  allow_partial_message: false
                  max_request_bytes: 1048576
      workloadSelector:
        labels:
          app: ${EGRESS_MON_GATEWAY_NAME}-${CR_NAMESPACE}
```

cr-egress-2-0-mon – использует два порта 8080 для http трафика и 5443 для mtls + OTT в качестве выходного шлюза.

```
    apiVersion: v1
    kind: Service
    metadata:
      name: crtx-svc-egress-mon
      labels:
        app.kubernetes.io/component: istio-discovery
        app.kubernetes.io/name: istio-discovery
        app.kubernetes.io/part-of: istio
        istio.io/rev: basic
        release: istio
    spec:
      ports:
        - {name: http-8080, port: 8080, protocol: TCP, targetPort: 8080}
        - {name: https-5443, port: 5443, protocol: TCP, targetPort: 5443}
      selector:
        istio: crtx-egress-mon
```

crtx-egress использует порт 5443 (mtls + OTT) в качестве выходного шлюза. Все остальные порты используются для маппинга портов.

```
  apiVersion: v1
kind: Service
metadata:
  name: crtx-egress
  labels:
    app.kubernetes.io/component: istio-discovery
    app.kubernetes.io/managed-by: maistra-istio-operator
    app.kubernetes.io/name: istio-discovery
    app.kubernetes.io/part-of: istio
    istio.io/rev: basic
spec:
  ports:
    # ott+mtls
    - name: https-5443,
      port: 5443
```

```
    protocol: TCP
    targetPort: 5443
  # db curate mapping port
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_db_currate_main_mapping_port }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_db_currate_main_mapping_port }}
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_db_currate_main_mapping_port }}
  # db curate failover mapping port
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_db_currate_main_failover_mapping_port }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_db_currate_main_failover_mapping_port }}
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_db_currate_main_failover_mapping_port }}
  # db curate si mapping port
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_db_currate_si_mapping_port }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_db_currate_si_mapping_port }}
    protocol: TCP
    targetPort: tcp-{{ istio_ose_istio_egress_svc_spec_ports_db_currate_si_mapping_port }}
  # db curate si failover port
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_db_currate_si_failover_mapping_port }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_db_currate_si_failover_mapping_port }}
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_db_currate_si_failover_mapping_port }}
  # db ekpit main mapping port
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_main_mapping_port }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_main_mapping_port }}
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_main_mapping_port }}
  # db ekpit main failover port
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_main_failover_mapping_port }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_main_failover_mapping_port }}
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_main_failover_mapping_port }}
  # db ekpit si mapping port
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_si_mapping_port }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_si_mapping_port }}
```

```
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_si_mapping_port }}
  # db ekpit si failover port
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_si_failover_mapping_port }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_si_failover_mapping_port }}
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_db_ekpit_si_failover_mapping_port }}
  # aj mapping port 1
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_1 }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_1 }}
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_1 }}
  # aj mapping port 2
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_2 }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_2 }}
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_2 }}
    # aj mapping port 3
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_3 }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_3 }}
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_3 }}
    # aj mapping port 4
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_4 }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_4 }}
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_4 }}
    # aj mapping port 5
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_5 }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_5 }}
    protocol: TCP
    targetPort: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_5 }}
  # aj mapping port 6
  - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_6 }}
    port: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_6 }}
```

```
      protocol: TCP
      targetPort: {{ istio_ose_istio_egress_svc_spec_ports_aj_kafka_mapping_port_6 }}
    # messenger kafak port 1
    - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_1 }}
      port: {{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_1 }}
      protocol: TCP
      targetPort: {{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_1 }}
    # messenger kafak port 2
    - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_2 }}
      port: {{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_2 }}
      protocol: TCP
      targetPort: {{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_2 }}
    # messenger kafak port 3
    - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_3 }}
      port: {{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_3 }}
      protocol: TCP
      targetPort: {{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_3 }}
    # messenger kafak port 4
    - name: tcp-{{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_4 }}
      port: {{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_4 }}
      protocol: TCP
      targetPort: {{ istio_ose_istio_egress_svc_spec_ports_messenger_kafka_mapping_port_4 }}
    # nsi facade
    - name: tcp-{{ istio_ose_istio_egress_nsi_mapping_port }}
      port: {{ istio_ose_istio_egress_nsi_mapping_port }}
      protocol: TCP
      targetPort: {{ istio_ose_istio_egress_nsi_mapping_port }}
```

```
  selector:
    istio: crtx-egress
type: ClusterIP
```

НСИ ФАСАД – фабрика вызываемая продуктом Platform V Exchange Rates, (MTLS+OTT), является опциональной интеграцией.

Взаимодействие с Фабрикой осуществляется через MTLS+OTT.

```
- apiVersion: networking.istio.io/v1alpha3
```

```
    kind: ServiceEntry
    metadata:
      labels:
        external: integration-nsi-facade
      name: crtx-se-nsi-facade-service
    spec:
      exportTo:
        - .
      hosts:
        - ${ istio_ose_istio_egress_nsi_host }
      location: MESH_EXTERNAL
      ports:
        - name: http-80
          number: 80
          protocol: HTTP
        - name: https-443
          number: 443
          protocol: TLS
      resolution: DNS
```

```
  - apiVersion: networking.istio.io/v1alpha3
    kind: Gateway
    metadata:
      labels:
        external: integration-nsi-facade
      name: crtx-gw-nsi-facade
    spec:
      selector:
        istio: crtx-egress
      servers:
        - hosts:
        - - {{ istio_ose_istio_egress_nsi_host }}
          port:
            name: http-5443
            number: 5443 # PORT EGRESS ДЛЯ MTLS+OTT
```

```
          protocol: HTTP

- apiVersion: networking.istio.io/v1alpha3
  kind: VirtualService
  metadata:
    labels:
      external: integration-nsi-facade
    name: crtx-vs-nsi-facade
  spec:
    exportTo:
      - .
    gateways:
      - crtx-gw-nsi-facade
      - mesh
    hosts:
      - {{ istio_ose_istio_egress_nsi_host }}
    http:
      - match:
          - gateways:
              - mesh
            port: 80
        rewrite:
          authority: {{ istio_ose_istio_egress_nsi_host  }}
        route:
          - destination:
              host: crtx-svc-egress
              port:
                number: 5443
            weight: 100
        retries:
          attempts: 2
          retryOn: gateway-error,connect-failure,refused-stream
      - match:
          - gateways:
              - gw-cr-egress-nsi-facade
```

```
            port: 5443
        route:
          - destination:
              host: ${NSI_FACADE_HOST}
              port:
                number: 443
            weight: 100
        retries:
          attempts: 2
          retryOn: gateway-error,connect-failure,refused-stream
```

```
- apiVersion: networking.istio.io/v1alpha3
  kind: DestinationRule
  metadata:
    labels:
      external: integration-nsi-facade
    name: crtx-dr--nsi-facade
  spec:
    exportTo:
      - .
    host: {{ istio_ose_istio_egress_nsi_host }}
    trafficPolicy:
      loadBalancer:
        simple: ROUND_ROBIN
      portLevelSettings:
        - port:
            number: 443
          tls:
            caCertificates: /etc/istio/gateway-ca-certs/ca.pem
            clientCertificate: /etc/istio/gateway-certs/tls.crt
            mode: MUTUAL
            privateKey: /etc/istio/gateway-certs/tls.key
            sni: {{ istio_ose_istio_egress_nsi_host }}
```

Компонент Журналирование (LOGA) продукта Platform V Monitor, взаимодействие MTLS + OTT

```
- apiVersion: networking.istio.io/v1alpha3
    kind: ServiceEntry
    metadata:
      labels:
        external: logger
      name: crtx-se-logger-service
    spec:
      exportTo:
        - .
      hosts:
        - ${LOGGER_HOST}
      location: MESH_EXTERNAL
      ports:
        - name: http-80
          number: 80
          protocol: HTTP
        - name: https-443
          number: 443
          protocol: TLS
      resolution: DNS
```

```
- apiVersion: networking.istio.io/v1alpha3
    kind: Gateway
    metadata:
      labels:
        external: logger
      name: crtx-gw-logger
    spec:
      selector:
        istio: crtx-egress-mon
      servers:
        - hosts:
            - ${LOGGER_HOST}
          port:
            name: http-5443 # Порт на egress MTLS+OTT
```

```
          number: 5443
          protocol: HTTP

- apiVersion: networking.istio.io/v1alpha3
  kind: VirtualService
  metadata:
    labels:
      external: logger
    name: crtx-vs-logger
  spec:
    exportTo:
      - .
    gateways:
      - crtx-gw-logger
      - mesh
    hosts:
      - {{ istio_ose_istio_egress_logger_host }}
    http:
      - match:
          - gateways:
              - mesh
            port: 80
        rewrite:
          authority: {{ istio_ose_istio_egress_logger_host }}
        route:
          - destination:
              host: crtx-svc-egress-mon
              port:
                number: 5443
            weight: 100
        retries:
          attempts: 2
          retryOn: gateway-error,connect-failure,refused-stream
      - match:
          - gateways:
```

```
              - gw-cr-egress-logger
            port: 5443
        route:
          - destination:
              host: ${LOGGER_HOST}
              port:
                number: 443
            weight: 100
        retries:
          attempts: 2

          retryOn: gateway-error,connect-failure,refused-stream

- apiVersion: networking.istio.io/v1alpha3
  kind: DestinationRule
  metadata:
    labels:
      external: logger
    name: dr-cr-egress-logger
  spec:
    exportTo:
      - .
    host: ${LOGGER_HOST}
    trafficPolicy:
      loadBalancer:
        simple: ROUND_ROBIN
      portLevelSettings:
        - port:
            number: 443
          tls:
            caCertificates: /etc/istio/gateway-ca-certs/ca.pem
            clientCertificate: /etc/istio/gateway-certs/tls.crt
            mode: MUTUAL
            privateKey: /etc/istio/gateway-certs/tls.key
            sni: ${LOGGER_HOST}
```

Компонент AUDT продукта Platform V Audit SE, взаимодействие MTLS + OTT

```
- apiVersion: networking.istio.io/v1alpha3
    kind: ServiceEntry
    metadata:
      labels:
        external: integration-audit
      name: se-cr-audit-service
    spec:
      exportTo:
        - .
      hosts:
        - ${AUDIT_HOST}
      location: MESH_EXTERNAL
      ports:
        - name: http-80
          number: 80
          protocol: HTTP
        - name: https-443
          number: 443
          protocol: TLS
      resolution: DNS
```

```
- apiVersion: networking.istio.io/v1alpha3
    kind: Gateway
    metadata:
      labels:
        external: integration-audit
      name: gw-cr-egress-audit
    spec:
      selector:
        istio: ${EGRESS_MON_GATEWAY_ISTIO_LABEL}
      servers:
        - hosts:
            - ${AUDIT_HOST}
          port:
            name: http-5443
```

```
          number: 5443
          protocol: HTTP

- apiVersion: networking.istio.io/v1alpha3
  kind: VirtualService
  metadata:
    labels:
      external: integration-audit
    name: vs-cr-egress-audit
  spec:
    exportTo:
      - .
    gateways:
      - gw-cr-egress-audit
      - mesh
    hosts:
      - ${AUDIT_HOST}
    http:
      - match:
          - gateways:
              - mesh
            port: 80
        rewrite:
          authority: ${AUDIT_HOST}
        route:
          - destination:
              host: svc-cr-egress-for-mon
              port:
                number: 5443
            weight: 100
        retries:
          attempts: 2
          perTryTimeout: 1s
          retryOn: gateway-error,connect-failure,refused-stream
      - match:
```

```
      - gateways:
          - gw-cr-egress-audit
        port: 5443
      route:
        - destination:
            host: ${AUDIT_HOST}
            port:
              number: 443
          weight: 100
      retries:
        attempts: 2
        perTryTimeout: 1s
        retryOn: gateway-error,connect-failure,refused-stream
```

```
- apiVersion: networking.istio.io/v1alpha3
  kind: DestinationRule
  metadata:
    labels:
      external: integration-audit
    name: dr-cr-egress-audit
  spec:
    exportTo:
      - .
    host: ${AUDIT_HOST}
    trafficPolicy:
      loadBalancer:
        simple: ROUND_ROBIN
      portLevelSettings:
        - port:
            number: 443
          tls:
            caCertificates: /etc/istio/gateway-ca-certs/ca.pem
            clientCertificate: /etc/istio/gateway-certs/tls.crt
            mode: MUTUAL
            privateKey: /etc/istio/gateway-certs/tls.key
```

```
            sni: ${AUDIT_HOST}
```

Установка с помощью компонента Deploy tools (CDJE) продукта Platform V DevOps Tools

Подготовка к развертыванию

Для подготовки к развертыванию продукта выполните следующие шаги:

1. Создание секрета в jenkins для доступа в среду контейнеризации.
2. Создание задачи в jenkins по установке дистрибутива.
3. Создание секрета.
4. Создание файла _passwords.conf в репозитории common, применяется только при опциональной автоматизированной установке с помощью DevOps Tools (Смотри пункт разворачивание джобы компонента Deploy tools (CDJE) продукта Platform V DevOps Tools).
5. Создание файла зависимых параметров установки, применяется только при опциональной автоматизированной установке с помощью DevOps Tools (Смотри пункт разворачивание джобы компонента Deploy tools (CDJE) продукта Platform V DevOps Tools.

Создание секрета в jenkins для доступа в среду контейнеризации

Секрет для доступа к проекту среды контейнеризации представляет собой секрет с типом "Service Account Tokens".

Поле **ID** – имя секрета.

Description – описание назначения секрета.

Token – токен сервис аккаунта jenkins из среды контейнеризации.

Получение токена Получение токена выполняется при создании сервисного аккаунта

Для того чтобы создать сервисный аккаунт (в случае его отсутствия):

1. Выполните команду.

```
kubectl create serviceaccount jenkins
```

2. В результате получите ответ.

```
serviceaccount "jenkins" created
```

3. Получите список секретов, привязанных к сервисному аккаунту.

```
kubectl get serviceaccounts jenkins -o yaml
```

4. В полученном ответе найдите имя секрета, который содержит токен формата: <имя сервисного аккаунта>-token-*.
5. Выполните команду.

```
kubectl get secret <имя сервисного аккаунта>-token-*  -o yaml
```

6. Пример ответа на вышеуказанную команду.

```
apiVersion: v1
data:
  ca.crt: (APISERVER'S CA BASE64 ENCODED)
------------
  token: (BEARER TOKEN BASE64 ENCODED)
kind: Secret
metadata:
  # ...
type: kubernetes.io/service-account-token
```

Значение поля токена необходимо декодировать из base64.

Разворачивание джобы компонента Deploy tools (CDJE) продукта Platform V DevOps Tools.

Пререквизиты для установки: Перед установкой убедитесь, что выполнены все следующие условия:

1. Наличие у пользователя, производящего установку возможности создания репозиториев в GitLab, Nexus Public и получения прав на чтение/запись, включая права на чтение/запись в ветке **master**.
2. Наличие технической учетной записи для работы с репозиториями.
3. Должны быть созданы следующие репозитории GitLab в проектной области:

- Репозитории **pipeline**, например:
    - crtx pipeline — репозиторий с кодовой базой релизов pipeline (на каждый стенд свой репозиторий, например crtx_pipeline_ift / crtx_pipeline_lt);
    - crtx_common — репозиторий с глобальными (одинаковыми для всех компонент в рамках одного стенда) переменными среды (на каждый стенд свой репозиторий, например crtx_common_ift / crtx_common_lt).
- Репозитории с конфигурациями функциональных подсистем, например:
    - crtx_crtx (на каждый стенд свой репозиторий).

4. Должны быть прописаны доступы в репозитории по протоколам SSH + HTTP на чтение/запись во все ветки, в т.ч. в ветку **master**.
5. В Nexus Public должны быть размещены дистрибутивы разворачиваемых функциональных подсистем, к данным репозиториям должен быть доступ с правами на чтение для технической учетной записи.
6. В Nexus Public должны быть созданы следующие репозитории, в которые загружены дистрибутивы всех компонентов **pipeline** (Installer.Base, Installer.Migration, Installer.Common):

    -
        - *AS_EFS_Installer.Base* — глобальные настройки и утилиты для **pipeline**;
    -
        -
            - AS_EFS_Installer.Migration* — дистрибутив утилиты миграции;
    -
        -
            - AS_EFS_Installer.Common* — базовый дистрибутив **common** — это глобальные параметры по умолчанию для pipeline. К перечисленным в данном пункте репозиториям должен быть доступ с правами на чтение для ТУЗ.

7. В GitLab должны быть созданы репозитории, в которые загружены дистрибутивы всех компонентов **pipeline**.

Подготовка инструментов развертывания

Предусмотрены следующие инструменты развёртывания:

- JOB Service** — job обновления (первичная загрузка кода JOB Deploy на полигон, обновление кода JOB Deploy при выпуске новых релизов JOB Deploy);
- JOB Deploy** — job развертывания дистрибутивов.

Далее перейдите к созданию секретов и последующих параметров касающихся сервиса.

Пример. Создание секрета входного шлюза

Секрет входного шлюза состоит из двух ресурсов типа secret. Создание ресурса выполняется с помощью утилиты командной строки – OC client с локальной машины.

Перед началом создания секретов получите следующие файлы:

- приватный ключ;
- серверный сертификат, подписанный удостоверяющим центром;
- цепочка сертификатов.

Выпуск сертификата для создания секрета входного шлюза

Выдача сертификатов осуществляется удостоверяющим центром.

Выполните следующие шаги:

1. Создайте конфигурационный файл запроса на сертификат **request.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           = SBRF
commonName                 = currency-rates.dev-apps.ocp-geo.ca.sbrf.ru

[ req_ext ]
subjectAltName = @alt_names

[alt_names]
DNS.1   = currency-rates-token.<your_domen>
DNS.2   = currency-rates-token.<your_domen>
DNS.3   = currency-rates-token.<your_domen>
DNS.4   = currency-rates.<your_domen>
DNS.5   = currency-rates.<your_domen>
DNS.6   = currency-rates.<your_domen>
DNS.7   = currency-rates-https.<your_domen>
```

   где в commonName указывается один из полных адресов хостов сервиса, или произвольная строка, в DNS.1, DNS.2 и далее, указываются полные имена хостов сервиса. Все адреса приведены в качестве примера и для реальной инсталляции могут отличаться.

   Адреса задаются в параметрах, перечень которых приведен ниже. В сертификате должны быть указаны следующие адреса:

- ${appFQDN_https}.${NAMESPACE_CDNS};
- ${appFQDN_https_ott}.${NAMESPACE_CDNS};
- ${appFQDN_all_in_on_ott}.${NAMESPACE_CDNS};
- ${appFQDN_all_in_on_https}.${NAMESPACE_CDNS};
- ${appFQDN_https}.${NAMESPACE_CDNS_GEO};
- ${appFQDN_https_ott}.${NAMESPACE_CDNS_GEO};
- ${appFQDN_all_in_on_ott}.${NAMESPACE_CDNS_GEO};

- ${appFQDN_all_in_on_https}.${NAMESPACE_CDNS_GEO};
- ${appFQDN_https}.${NAMESPACE_CDNS_SMART_GEO};
- ${appFQDN_https_ott}.${NAMESPACE_CDNS_SMART_GEO};
- ${appFQDN_all_in_on_ott}.${NAMESPACE_CDNS_SMART_GEO};
- ${appFQDN_all_in_on_https}.${NAMESPACE_CDNS_SMART_GEO}.

Выше указан набор адресов одного кластера, существует возможность сделать общий сертификат на оба кластера:

request.csr.

```
openssl req -out request.csr -newkey rsa:2048 -nodes -keyout tls.key -config request.cnf
```

1. Подпишите сертификат в вашем УЦ.
2. Получите файл request.cer (подписанный сертификат в формате DER) и цепочку сертификатов, которым он подписан.
3. Преобразуйте полученный сертификат в формат **PEM**.
   ```
   openssl x509 -inform DER -in request.cer -out tls.crt
   ```
4. Преобразуйте полученный корневой сертификат в формат **PEM**.
   ```
   openssl x509 -inform DER -in "Test Root CA 2.cer" -out root.pem
   ```
5. Преобразуйте полученный промежуточный сертификат в формат **PEM**.
   ```
   openssl x509 -inform DER -in "Test Issuing CA 2.cer" -out issuing.pem
   ```
6. Объедините корневой и промежуточный сертификат в один файл.
   ```
   cat root.pem issuing.pem > ca.pem
   ```
7. Получите три файла:

- tls.key – приватный ключ;
- tls.crt – серверный сертификат;
- ca.pem – доверенная цепочка.

Создание секретов Дальнейшие действия проводятся в проекте, где у пользователя есть права на создание секретов. Сами файлы секретов хранятся в common репозитории в зашифрованном виде.

1. Для создания секретов в среде контейнеризации изучите файл distrib.yaml находящийся в дистрибутиве. Пример части файла с описанием секретов в файле distrib.yaml

```
# -------------------------------------------------
# Файл описывающий состав дистрибутива
# Secrets которые возьмутся из _passwords.conf (Common репозиторий)
secrets: [
  # База данных CR MAIN
  "DB_CURRATE_MAIN_LOGIN",
  "DB_CURRATE_MAIN_PASSWORD",
  # База данных CR SI
  "DB_CURRATE_STANDIN_LOGIN",
```

```
  "DB_CURRATE_STANDIN_PASSWORD",
  #
  "DB_EKPIT_MAIN_LOGIN",
  "DB_EKPIT_MAIN_PASSWORD",
  "DB_EKPIT_STANDIN_LOGIN",
  "DB_EKPIT_STANDIN_PASSWORD",
  #
  "MESSENGER_KAFKA_SSL_KEY_STORE_PASSWORD",
  "MESSENGER_KAFKA_SSL_TRUST_STORE_PASSWORD",
  # пароль от закрытого ключа
  "MESSENGER_KAFKA_SSL_PRIVATE_KEY_PASSWORD",
  #
  "OTT_CERTSTORE_PWD",
  "OTT_TRUST_STORE_PWD",
  # пароль от закрытого ключа
  "OTT_CERTSTORE_PRIVATE_KEY_PWD",
  #
  "AJ_KEYSTORE_PASSWORD",
  "AJ_TRUSTORE_PASSWORD",
  # пароль от закрытого ключа
  "AJ_KEY_PRIVATE_KEY_PASSWORD"
]
```

```
extra-secrets: [
  "secret-mq",
  "secret-aj",
  "secret-ott-certs",
  "secret-ingress-certs",
  "secret-ingress-ca-certs",
  "secret-egress-certs",
  "secret-egress-ca-certs"
]
```

2. В случае использование компонента Deploy tools (CDJE) продукта Platform V DevOps Tools для установки, в репозитории common заполните файл _password.conf. Название параметров представлено в блоке secrets в файле

distrib.yaml. **Внимание!** Помните, что название и количество параметров может изменяться от релиза к релизу.

Имя параметра секрета	**Значение**
DB_CURRATE_MAIN_LOGIN	Логин (имя пользователя) под которым продукт будет обращаться к базе данных приложения
DB_CURRATE_MAIN_PASSWORD	Пароль пользователя под которым продукт обращаться к базе данных приложения
DB_CURRATE_STANDIN_LOGIN	Логин (имя пользователя) под которым продукт обращаться к базе данных приложения stand-in (в случае отсутствия базы stand-in указать значение dummy)
DB_CURRATE_STANDIN_PASSWORD	Пароль пользователя под которым продукт обращаться к базе данных приложения stand-in (в случае отсутствия базы stand-in указать значение dummy)
DB_EKPIT_MAIN_LOGIN	Логин (имя пользователя) под которым продукт будет обращаться к базе данных ekpit
DB_EKPIT_MAIN_PASSWORD	Пароль пользователя под которым продукт обращаться к базе данных ekpit
DB_EKPIT_STANDIN_LOGIN	Логин (имя пользователя) под которым продукт обращаться к базе данных

Имя параметра секрета	**Значение**
	ekpit stand-in (в случае отсутствия базы stand-in указать значение dummy)
DB_EKPIT_STANDIN_PASSWORD	Пароль пользователя под которым продукт обращаться к базе данных ekpit stand-in (в случае отсутствия базы stand-in указать значение dummy)
MESSENGER_KAFKA_SSL_KEY_STORE_PASSWORD	В состав дистрибутива входит сервис messanger. Для работы текущего сервиса применяться брокер компонента Kafka Sber Edition (KFKA) продукта Platform V Corax . В случае использования Kafka с включённым ACL и протоколом SASL вам необходимо создать секрет с и менем secret-mq который будет содержать хранилище сертификатов JSK. Далее вам необходимо заполнить эти параметры передав пароли от хранилища Jks. SSL_KEY_STORE пароль от хранилища, т.д(в случае не использования сервиса указать значение dummy)
MESSENGER_KAFKA_SSL_TRUST_STORE_PASSWORD	В состав дистрибутива входит сервис messanger. Для работы текущего сервиса применяется брокер сообщений компонента Kafka Sber Edition (KFKA) продукта Platform V CoraxPlatform V Corax. В случае использования Kafka с включённым ACL и протоколом SASL вам необходимо создать секрет с и менем secret-mq который будет содержать хранилище сертификатов JSK. Далее вам необходимо заполнить эти параметры передав пароли от хранилища Jks. SSL_KEY_STORE пароль от хранилища, т.д (в случае не использования сервиса указать значение dummy)

Имя параметра секрета	**Значение**
MESSENGER_KAFKA_SSL_PRIVATE_KEY_PASSWORD	В состав дистрибутива входит сервис messanger. Для работы текущего сервиса применяться брокер сообщений компонент Kafka Sber Edition (KFKA) продукта Platform V Corax. В случае использования Kafka с включённым ACL и протоколом SASL вам необходимо создать секрет с и менем secret-mq который будет содержать хранилище сертификатов JSK. Далее вам необходимо заполнить эти параметры передав пароли от хранилища Jks. SSL_KEY_STORE пароль от хранилища, т.д (в случае не использования сервиса указать значение dummy)
OTT_CERTSTORE_PWD	По аналогии с секретами от messanger текущие параметры несут в себе пароли от JKS для сервиса ott (в случае не использования сервиса указать значение dummy)
OTT_TRUST_STORE_PWD	По аналогии с секретами от messanger текущие параметры несут в себе пароли от JKS для сервиса ott (в случае не использования сервиса указать значение dummy)
OTT_CERTSTORE_PRIVATE_KEY_PWD	По аналогии с секретами от messanger текущие параметры несут в себе пароли от JKS для сервиса ott (в случае не использования сервиса указать значение dummy)
AJ_KEYSTORE_PASSWORD	По аналогии с секретами от messanger текущие параметры несут в себе пароли от JKS для сервиса ПЖ (в случае

Имя параметра секрета	**Значение**
	неиспользования сервиса указать значение dummy)
AJ_TRUSTORE_PASSWORD	По аналогии с секретами от messanger текущие параметры несут в себе пароли от JKS для сервиса ПЖ (в случае неиспользования сервиса указать значение dummy)
AJ_KEY_PRIVATE_KEY_PASSWORD	По аналогии с секретами от messanger текущие параметры несут в себе пароли от JKS для сервиса ПЖ (в случае неиспользования сервиса указать значение dummy)

3. Далее зашифруйте файл _password.conf согласно инструкции компонента Deploy tools (CDJE) продукта Platform V DevOps Tools. Для простоты шифрования существует возможность воспользоваться скриптом шифрования на powerShell. Создайте файл с тем же именем, что и файл который должен быть зашифрован, добавьте к имени окончание “_d”. Далее скрипт найдет все файлы с окончанием “_d” и зашифрует, создав файл без окончания “_d”.
4. В файл .gitignore добавьте *_d что бы нешифрованные файлы не попали в репозиторий.

```
$files = (Get-ChildItem *_d).FullName
$vault_cred_openshift_ops='<your pass encrypt>'
ForEach ($file in $files){
    echo $file
    openssl enc -aes-256-cbc -k $vault_cred_openshift_ops -in $file -out $file.TrimEnd('_d')
}
```

5. Далее создайте файлы в директории openshift_extra_secrets с именами, описанными в блоке extra-secrets. Имя секрета должно совпадать с именем файла. **secret-aj**:

```
kind: Secret
apiVersion: v1
metadata:
```

```
  name: secret-aj
type: Opaque
data:
  keystore.jks: <base64 enc jks store >
  truststore.jks: <base64 enc jks store >
```

secret-egress-ca-certs:

```
kind: Secret
apiVersion: v1
metadata:
  name: secret-egress-ca-certs
type: Opaque
data:
  ca.pem: <base 64 cert pem >
```

secret-egress-certs:

```
kind: Secret
apiVersion: v1
metadata:
  name: secret-egress-certs
  labels:
    app: crtx
type: Opaque
data:
  tls.crt: <base64 enc jks store >
  tls.key: <base64 enc jks store >
```

secret-ingress-ca-certs:

```
kind: Secret
apiVersion: v1
metadata:
  name: secret-ingress-ca-certs
type: Opaque
data:
  ca.pem:<base 64 cert pem>
```

secret-ingress-certs:

```
kind: Secret
apiVersion: v1
metadata:
  name: secret-ingress-certs
  labels:
    app: crtx
type: Opaque
data:
  tls.crt: <base64 enc jks store >
  tls.key: <base64 enc jks store >
```

secret-mq:

```
kind: Secret
apiVersion: v1
metadata:
  name: secret-mq
  labels:
    app: crtx
type: Opaque
data:
  keystore.jks: <base64 enc jks store >
  truststore.jks: <base64 enc jks store >
```

secret-ott-certs:

```
kind: Secret
apiVersion: v1
metadata:
  name: secret-ott-certs
type: Opaque
data:
  keystore.jks: <base64 enc jks store >
  truststore.jks: <base64 enc jks store >
```

Имя секрета	Содержимое
secret-ott-certs	Секрет содержит jks хранилище keystote, truststore. Текущий секрет необходим для функционирование ott (может быть опционален)
secret-mq	Секрет содержит jks хранилище keystote, truststore. Текущий секрет необходим для функционирование сервиса messanger (может быть опционален)
secret-ingress-certs	Секрет содержит сертификат в формате pem необходим для реализации mtls, монтируется в pod ingress istio/servicemesh
secret-ingress-ca-certs	Секрет содержит root сертификат центра сертификации в формате pem необходим для реализации mtls, монтируеться в pod ingress istio/servicemesh
secret-egress-certs	Секрет содержит сертификат в формате pem необходим для реализации mtls, монтируется в pod egress istio/servicemesh
secret-egress-ca-certs	Секрет содержит root сертификат центра сертификации в формате pem необходим для реализации mtls, монтируеться в pod egress istio/servicemesh
secret-aj	Секрет содержит jks хранилище keystote, truststore. Текущий секрет необходим для функционирования ПЖ (может быть опционален)

Администратор самостоятельно выбирает как реализовать kind secret для k8s, использовать data или datastring, и какой тип секрета ему применить. Важно сохранить имя секрета и ключ значения параметра. Тип хранилища может быть как jks, так и PKCS12.

Пример создание хранилище сертификата УЦ Процесс создания хранилища сертификата УЦ продемонстрирован на примере OTT.

Для того чтобы клиенты OTT при установке SLT(TLS) соединения доверяли SSL−сертификатам серверов OTT:

1. Включите в trust store сертификат УЦ, который выпустил SSL−сертификат серверов OTT.
2. Скачайте файл УЦ посредствами браузера, подключившись к серверу OTT (https://<ott−server>:8443/).
3. Для импорта полученного сертификата в trust store используйте команду:

shell

```
keytool −importcert −file ./root_ca.cer −keystore ./ott_truststore.p12 −alias "root_ca"
```

Настройка клиента OTT

Чтобы настроить клиента OTT на использование полученного сертификата OTT Service, необходимо задать параметры:

Параметр	**Описание**	**Пример**
ott.trust.store.path	Путь к хранилищу, содержащему сертификат OTT Service	/etc/pprb/ott/ott_truststore.p12
ott.trust.store.pwd	Пароль от хранилища, содержащего сертификат OTT Service. При получении сертификата задавался в параметре <password>	<здесь ваш пароль от хранилища ключей>
ott.service.cert.alias	Alias сертификата OTT Service в хранилище	ott−service
ott.certstore.type=PKCS12	Тип хранилища (PKCS12 или JKS)	PKCS12

Для получения приватной части сертификата:

1. Администратор прикладного модуля: выполняет генерацию ключевой пары в p12−контейнере.

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

   Рекомендуется использовать версию OpenJDK не ниже 1.8
2. Администратор прикладного модуля: формирует запрос на сертификат.

```
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 создастся файл CSR– **${module_id}_cert_req.pem**, который необходимо передать Администратору OTT в заявке на генерацию сертификата для модуля. В заявке указать, что сертификат должен быть сгенерирован из приложенного запроса по данной инструкции.

3. Администратор OTT: выписывает сертификат.

```
./ejbcawsracli.sh certreq %{module_id} "CN=%{module_id}" NULL PlatformCA_EC
PPRBModule %{module_id}_cert_req.pem PKCS10 PEM NONE
```

В результате выполнения данной команды будет создан файл с сертификатом– **%{module_id}.pem**, который необходимо передать Администратору прикладного модуля.

4. Администратор OTT: вместе с сертификатом модуля передает сертификат УЦ OTT **PlatformCA_EC.pem.** CN=PlatformCA_EC
CA certificate: **Download as PEM**
5. Администратор прикладного модуля: импортирует сертификат УЦ OTT и сертификат модуля в keystore, созданный на 1 шаге (строго в указанном порядке).

```
keytool –import –keystore ${module_id}.p12 –storetype PKCS12 –file
PlatformCA_EC.pem –alias PlatformCA_EC
keytool –import –keystore ${module_id}.p12 –storetype PKCS12 –file
${module_id}.pem –alias ${module_id}
```

6. После выполнения шагов должно быть два файла (необязательно с такими именами):
    0. ott_service_truststore.p12 – публичная часть (truststore);
    1. pricing–service.p12 – приватная часть (keystore).

Подготовка файла конфигурации стенда зависимых параметров Это текстовый файл в формате: имя_параметра: значение_параметра. Цифровые значения задаются в одинарных кавычках. **Пример файла параметров Expand source**

```
CR_NAMESPACE: ci00642380–idevgen2–dynamic–pricing–ift
```

При изменении строки подключения к БД (имена, ip адреса, порты), списка брокеров Kafka (имена, ip адреса, порты) необходимо изменение необязательных параметров по умолчанию в части маршрутизации трафика. Без этих изменений не произойдет соединение с хостом. Если количество хостов брокеров Kafka увеличиться потребуется новый релиз с поддержкой новой конфигурации. Если один из портов преобразования будет использоваться в строке подключения к БД и/или как порт подключения к брокеру Kafka, его значение придется переназначить на другой порт. При этом нужно учесть, что эти порты преобразования указаны в сервисе выходного шлюза. Нужно будет выбрать незанятый порт среди списка портов сервиса выходного шлюза.

В репозитории common можно добавить параметры представленные в таблице ниже. В файле /b1/installer/system/efs/config/parameters/openShift.conf

Имя параметра	**Описание**	**Пример**
global_docker_registry	Имя вашего приватного хранилища докер контейнеров	dzo.sw.sbc.space
crtx_cb_cloud_app_docker_part	Имя тега докер контейнера в разрезе сервисов без указания доменной части приватного хранилища docker и версии	sbt_dev/ci90000071_crtx_dev/crtx-cb-cloud-app
crtx_financial_quotes_cloud_app_docker_part	Имя тега докер контейнера в разрезе сервисов без указания доменной части приватного хранилища docker и версии	sbt_dev/ci90000071_crtx_dev/crtx-financial-quotes-cloud-app

Имя параметра	Описание	Пример
crtx_individual_cloud_app_docker_part	Имя тега докер контейнера в разрезе сервисов без указания доменной части приватного хранилища docker и версии	sbt_dev/ci90000071_crtx_dev/crtx-individual-cloud-app
crtx_interbank_rates_cloud_app_docker_part	Имя тега докер контейнера в разрезе сервисов без указания доменной части приватного хранилища docker и версии	sbt_dev/ci90000071_crtx_dev/crtx-interbank-rates-cloud-app
crtx_legal_cloud_app_docker_part	Имя тега докер контейнера в разрезе сервисов без указания доменно	sbt_dev/ci90000071_crtx_dev/crtx-messenger-cloud-app

Имя параметра	Описание	Пример
	й части приватного хранилища docker и версии.	
crtx_messenger_cloud_app_docker_part	Имя тега докер контейнера в разрезе сервисов без указания доменной части приватного хранилища docker и версии	sbt_dev/ci90000071_crtx_dev/crtx-supplementary-cloud-app
crtx_supplementary_cloud_app_docker_part	Имя тега докер контейнера в разрезе сервисов без указания доменной части приватного хранилища docker и версии	sbt_dev/ci90000071_crtx_dev/crtx-legal-cloud-app

Имя параметра	**Описание**	**Пример**
global_docker_secret	Имя секрета для доступа к docker хранилищу	default
NAMESPACE_CDNS	Доменные имена балансировшика на которых будут созданы url для текущей инсталляции	apps.stands-vdc01.solution.sbt
NAMESPACE_CDNS_GEO	Доменные имена балансировшика на которых будут созданы url для текущей инсталляции	geo.apps.stands-vdc01.solution.sbt
NAMESPACE_CDNS_SMART_GEO	Доменные имена балансировшика на	geo-smart.apps.stands-vdc01.solution.sbt

Имя параметра	Описание	Пример
	которых будут созданы url для текущей инсталляции	
mq_kafka_topic_person_distant	Имя топика для для обмена сообщения	CRTX.CURRENCYRATESFORSBBOLCREATEDEVENT.V1
mq_kafka_ssl_endpoint_algorithm	Алгоритм взаимодействия с брокерами kafka	PLAINTEXT или SASL
mq_kafka_brokers	Список брокеров в формате host:port	Kafka_brocker1:9093,kafka_brocker2:9093
mq_kafka_topic_person_metal_distant	Имя топика для для обмена сообщения	CRTX.CURRENCYRATESPERSONMETALDISTANTCREATEDEVENT.V1

Имя параметра	**Описание**	**Пример**
mq_kafka_topic_legal_distant	Имя топика для для обмена сообщения	CRTX.CURRENCYRATESFORSBBOLCREATEDEVENT.V1
ott_image	Полная ссылка на образ ott	registry..sbrf.ru/sigma/pprb/ci00641491/ci01125613_ott/ott-client-api@sha256:dcbb541e856497a0d1587bbea79d2625f4d7ed2cb00a5dabb2c6fea0e8340046
ott_servers_urls	Список хостов для ott в формате host:port	'str-vat-app2141.delta.sbrf.ru:8443,str-vat-app2142.delta.sbrf.ru:8443'
fluent_bit_image	Полная ссылка на образ fluent_bit	registry.delta.sbrf.ru/docker-release/ci00734898/ci00685811_synapse/fluent-bit@sha256:cafd572be2350573fd76a04f12a46b7d56f399289c88b46a3783c01302aa6473
fluent_bit_loggerHost	Хост elk	Elk.logger.ru
fluent_bit_loggerPort	Порт elk	Если предполагается использовать для вывода трафика из k8s/openshift egress. То необходимо указать порт 80. В случае отказа от istio необходимо указать целевой порт elk
fluent_bit_standId	Имя стенда	Любая строка для идентификации логов пример dev/prom

Имя параметра	Описание	Пример
fluent_bit_zoneId	Имя Зоны	Пример CV
image_istio_image	Полная ссылка на образ ingress (envoy-proxy) istio/servicemesh	registry.redhat.io/openshift-service-mesh/proxyv2-rhel8@sha256:320f5bd35c208e00005c01ce0e83c8f05276119f273e9f881da950fdfff59a13

Параметры базы данных

Имя параметра	Описание	Пример
jdbc_db_currate_main_url	Строка подключение к базе приложения в формате jdbc	jdbc:postgresql://10.23.20.18:5432/crtxdev?currentSchema=currency_rates
jdbc_dbcurrate_si_url	Строка подключение к базе приложения stand-in в формате jdbc	jdbc:postgresql://10.23.20.19:5432/crtxdev?currentSchema=currency_rates
jdbc_db_currate_driver	Тип используемого	org.postgresql.Driver

Имя параметра	Описание	Пример
	драйвера jdbc	
jdbs_db_curate_owner	Указание владельца схемы базы данных приложения	currency_rates
jdbc_db_ekpit_main_url	Строка подключение к базе ekpit в формате jdbc	jdbc:postgresql://10.23.130.16:6544/ekpitift01?currentSchema=ekpit
jdbc_db_ekpit_si_url	Строка подключение к базе ekpit stand-in в формате jdbc	jdbc:postgresql://10.23.130.17:6545/ekpitift01?currentSchema=ekit
jdbc_db_ekpit_owner_schema	Указание владельца схемы базы данных ekpit	ekpit
jdbc_db_ekpit_driver	Тип используемого	org.postgresql.Driver

Имя параметра	Описание	Пример
	драйвера jdbc	

Параметры интеграционных сервисов

Имя параметра	Описание	Пример
audit_url_host	url для отправки метрик аудита	audit.sberbank.ru
audit_disable	Параметр отключения Метрик аудита	false
nsi_facade_host	url для подключения к nsi facad	nsi.sberbank.ru
nsi_facade_port	Порт для подключения к nsi façade	Если предполагается использовать для вывода трафика из k8s/openshift egress. То необходимо указать порт 80. В случае отказа от istio необходимо указать целевой порт nsi
nsi_facade_use	Параметр отключает обращение к nsi façade и переключает приложение на использование базы ekpit	true
aj_kafka_brokers	Список брокеров для ПЖ	aj.sberbansk.ru:9092,aj.sberbank.ru:9092

Имя параметра	Описание	Пример
aj_client_stub	Отключение прикладной репликации	false
aj_zone_id	Зона репликации ПЖ	CV

Для обеспечения безопасности передаваемых данных используется mTLS + OTT. Все входящие и исходящие межсервисные соединения проходят через ingress/egress proxy. Для входящих/исходящих взаимодействий используются mTLS.

Стендозависимые параметры ISTIO/ Platform V **Synapse Service Mesh**

Имя параметра	Описание	Пример
global.multiClusters.openshiftControlPlaneIstiodService	Параметр берётся из контекста описания среды в файле multiClusters.json в common репозитории. Содержит ссылку на control plane istio/servicemesh	istiod-basic.ci01994970-idevgen-control-panel-synapse.svc:15012
NAMESPACE_CDNS	Доменная часть url создаваемого route/ingress В промышленной среде это балансировщики	apps.stands-vdc01.solution.sbt

Имя параметра	**Описание**	**Пример**
NAMESPACE_CDNS_GEO	Доменная часть url создаваемого route/ingress В промышленной среде это балансировщик и	apps.stands-vdc01.solution.sbt
NAMESPACE_CDNS_SMART_GEO	Доменная часть url создаваемого route/ingress В промышленной среде это балансировщик и	apps.stands-vdc01.solution.sbt
appFQDN_https	Хостовая часть урл.	currency-rates-kb , currency-rates-https-kb currency-rates , currency-rates-https
appFQDN_https_ott	Хостовая часть урл.	currency-rates-kb , currency-rates-https-kb currency-rates , currency-rates-https
appFQDN_all_in_on_ott	Хостовая часть урл.	currency-rates-kb , currency-rates-https-kb currency-rates , currency-rates-https

Имя параметра	Описание	Пример
appFQDN_all_in_on_https	Хостовая часть урл.	currency-rates-kb , currency-rates-https-kb currency-rates , currency-rates-https

В результате развертывания ingress/router будут созданы 12 ingress. Такое количество связано с циклом жизни приложения.

Параметры «тонкой» настройки istio

Текущие параметры настраивают сущности Gateway, Virtual Service, Service Entry, Envoy Filter. Также, текущие параметры настраивают проксирование и выполняют роль firewall.

Пример: При указании параметров для подключения к базе данных приложения:

```
jdbc_db_currate_main_url= jdbc:postgresql://10.23.20.18:5432/crtxdev?currentSchema=currency_rates
```

Текущий параметр отвечает за настройку сервиса. Далее настройте среду чтобы текущий трафик мог быть разрешён в сервисной сети. Для этого заполните параметры. **Пример для базы данных**

```
jdbc_db_currate_main_host = 10.23.20.18
jdbc_db_currate_main_ip = 10.23.20.18
jdbc_db_currate_main_port = 5432\
```

Эти параметры переобозначаются в репозитории ФП В файле crtx.istio.all.conf

```
…
#currate db egrees
# const mapping port
istio_ose_istio_egress_svc_spec_ports_db_currate_main_mapping_port=5555
istio_ose_istio_egress_svc_spec_ports_db_currate_main_failover_mapping_port=5556
istio_ose_istio_egress_svc_spec_ports_db_currate_si_mapping_port=5556
istio_ose_istio_egress_svc_spec_ports_db_currate_si_failover_mapping_port=5557
#host
istio_ose_istio_egress_db_currate_main_host=${jdbc_db_currate_main_host}
istio_ose_istio_egress_db_currate_main_failover_host=${jdbc_db_currate_main_failover_host}
istio_ose_istio_egress_db_currate_si_host=${jdbc_db_currate_si_host}
istio_ose_istio_egress_db_currate_si_failover_host=${jdbc_db_currate_si_failover_host}
# ip
```

```
istio_ose_istio_egress_db_currate_main_ip=${jdbc_db_currate_main_ip}
istio_ose_istio_egress_db_currate_main_failover_ip=${jdbc_db_currate_main_failover_ip}
istio_ose_istio_egress_db_currate_si_ip=${jdbc_db_currate_si_ip}
istio_ose_istio_egress_db_currate_si_failover_ip=${jdbc_db_currate_si_failover_ip}
# port
istio_ose_istio_egress_db_currate_main_port=${jdbc_db_currate_main_port}
istio_ose_istio_egress_db_currate_main_failover_port=${jdbc_db_currate_main_failover_port}
istio_ose_istio_egress_db_currate_si_port=${jdbc_db_currate_si_port}
istio_ose_istio_egress_db_currate_si_failover_port=${jdbc_db_currate_si_failover_port}
…
```

Пример конфигурации istio/servicemesh, который будет сформирован из дистрибутива:

```
apiVersion: networking.istio.io/v1alpha3
    kind: ServiceEntry
    metadata:
      name: se-db-currency-main
    spec:
      addresses:
        - {{istio_ose_istio_egress_db_currate_main_ip }}/32
      endpoints:
        - address: {{ istio_ose_istio_egress_db_currate_main_ip }}
      exportTo:
        - .
      hosts:
        - {{ istio_ose_istio_egress_db_currate_main_host }}
      location: MESH_EXTERNAL
      ports:
        - name: tcp-{{ istio_ose_istio_egress_db_currate_main_port }}
          number: {{ istio_ose_istio_egress_db_currate_main_port }}
          protocol: TCP
      resolution: STATIC
```

```
    apiVersion: networking.istio.io/v1alpha3
    kind: Gateway
    metadata:
      name: gw-cr-egress-db-currency-main
```

```
  spec:
    selector:
      istio: crtx-egress
    servers:
      - hosts:
          - {{ istio_ose_istio_egress_db_currate_main_host }}
        port:
          name: tcp-{{ istio_ose_istio_egress_db_currate_main_port }}
          number: {{ istio_ose_istio_egress_db_currate_main_port }}
          protocol: TCP

---
    apiVersion: networking.istio.io/v1alpha3
  kind: VirtualService
  metadata:
    name: vs-cr-egress-db-currency-main
  spec:
    exportTo:
      - .
    gateways:
      - gw-cr-egress-db-currency-main
      - mesh
    hosts:
      - {{ istio_ose_istio_egress_svc_spec_ports_db_currate_main_host }}
    tcp:
      - match:
          - gateways:
              - mesh
            port: {{ istio_ose_istio_egress_svc_spec_ports_db_currate_main_port }}
        route:
          - destination:
              host: svc-cr-egress
              port:
                number: {{ istio_ose_istio_egress_svc_spec_ports_db_currate_main_mapping_port }}
            weight: 100
      - match:
```

```
          - gateways:
              - gw-cr-egress-db-currency-main
            port: {{ istio_ose_istio_egress_svc_spec_ports_db_currate_main_mapping_port }}
        route:
          - destination:
              host: {{ istio_ose_istio_egress_db_currate_main_host }}
              port:
                number: {{ istio_ose_istio_egress_db_currate_main_port }}
            weight: 100
```

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
 labels:
   external: database
name: dr-dp-egress-${DB_CURRENCY_MAIN_HOST}
spec:
  exportTo:
    - .
  host: {{ istio_ose_istio_egress_db_currate_main_host }}
  trafficPolicy:
    loadBalancer:
      simple: ROUND_ROBIN
    portLevelSettings:
      - port:
       number: {{ istio_ose_istio_egress_db_currate_main_mapping_port }}
      connectionPool:
        tcp:
          connectTimeout: {{ istio_ose_istio_egress_db_connectionPool_tcp_connectTimeout }}
```

```
          maxConnections:{{ istio_ose_istio_egress_db_connectionPool_tcp_maxConnections }}
          tcpKeepalive:
            time: {{ istio_ose_istio_egress_db_connectionPool_tcpKeepalive_time }}
            interval: {{ istio_ose_istio_egress_db_connectionPool_tcpKeepalive_interval }}
```

Параметры базы данных приложения

Имя параметра	Описание	Пример
jdbc_db_currate_main_host	В текущем параметре передается host name или ip адресс базы данных (если у базы нет FQDN)	db.curren-rate.sberban.ru или 10.21.1.21
jdbc_db_currate_main_failover_host	В текущем параметре передается host name или ip адресс базы данных (если у базы нет FQDN)	db.curren-rate.sberban.ru или 10.21.1.21
jdbc_db_currate_si_host	В текущем параметре передается host name или ip адресс базы данных (если у базы нет FQDN)	db.curren-rate.sberban.ru или 10.21.1.21
jdbc_db_currate_si_failover_host	В текущем параметре передается host name или ip адресс базы данных (если у базы нет FQDN)	db.curren-rate.sberban.ru или 10.21.1.21
jdbc_db_currate_main_ip	В текущих параметрах передается ip адресс базы данных приложения	10.21.1.21

Имя параметра	**Описание**	**Пример**
jdbc_db_currate_main_failover_ip	В текущих параметрах передается ip адресс базы данных приложения	10.21.1.21
jdbc_db_currate_si_ip	В текущих параметрах передается ip адресс базы данных приложения	10.21.1.21
jdbc_db_currate_si_failover_ip	В текущих параметрах передается ip адресс базы данных приложения	10.21.1.21
istio_ose_istio_egress_db_currate_main_port	В текущих параметрах передается порт базы данных приложения	5432
istio_ose_istio_egress_db_currate_main_failover_port	В текущих параметрах передается порт базы данных приложения	5432
istio_ose_istio_egress_db_currate_si_port	В текущих параметрах передается порт базы данных приложения	5432

Имя параметра	Описание	Пример
istio_ose_istio_egress_db_currate_si_failover_port	В текущих параметрах передается порт базы данных приложения	5432

Параметры базы данных ekpit

Имя параметра	Описание	Пример
jdbc_db_ekpit_main_host	В текущем параметре передается host name или ip адресс базы данных ekpit (если у базы нет FQDN)	db.ekpit.sberban.ru
jdbc_db_ekpit_main_failover_host	В текущем параметре передается host name или ip адресс базы данных ekpit (если у базы нет FQDN)	db.ekpit.sberban.ru
jdbc_db_ekpit_si_host	В текущем параметре передается host name или ip адресс базы данных ekpit (если у базы нет FQDN)	db.ekpit.sberban.ru
jdbc_db_ekpit_si_failover_host	В текущем параметре передается host name или ip адресс базы данных ekpit (если у базы нет FQDN)	db.ekpit.sberban.ru
jdbc_db_ekpit_main_failover_ip	В текущих параметрах передается ip адресс базы данных ekpit	10.21.1.21
jdbc_db_ekpit_si_ip	В текущих параметрах передается ip адресс базы данных ekpit	10.21.1.21

Имя параметра	Описание	Пример
jdbc_db_ekpit_si_failover_ip	В текущих параметрах передается ip адресс базы данных ekpit	10.21.1.21
jdbc_db_ekpit_main_ip	В текущих параметрах передается ip адресс базы данных ekpit	10.21.1.21
jdbc_db_ekpit_main_port	В текущих параметрах передается порт базы данных ekpit	5432
jdbc_db_ekpit_main_failover_port	В текущих параметрах передается порт базы данных ekpit	5432
jdbc_db_ekpit_si_port	В текущих параметрах передается порт базы данных ekpit	5432
jdbc_db_ekpit_si_failover_port	В текущих параметрах передается порт базы данных ekpit	5432

Параметры ПЖ

Имя параметра	Описание	Пример
aj_host_1	В текущем параметре передается Fqdn или ip адресс брокеров ПЖ (если у базы нет FQDN)	aj-1.sberban.ru
aj_host_2	В текущем параметре передается Fqdn или ip адресс брокеров ПЖ (если у базы нет FQDN)	aj-1.sberban.ru

Имя параметра	**Описание**	**Пример**
aj_host_3	В текущем параметре передается Fqdn или ip адресс брокеров ПЖ (если у базы нет FQDN)	aj-1.sberban.ru
aj_host_4	В текущем параметре передается Fqdn или ip адресс брокеров ПЖ (если у базы нет FQDN)	aj-1.sberban.ru
aj_host_5	В текущем параметре передается Fqdn или ip адресс брокеров ПЖ (если у базы нет FQDN)	aj-1.sberban.ru
aj_host_6	В текущем параметре передается Fqdn или ip адресс брокеров ПЖ (если у базы нет FQDN)	aj-1.sberban.ru
aj_ip_1	В текущем параметре передается ip адресс брокеров ПЖ	10.21.1.21
aj_ip_2	В текущем параметре передается ip адресс брокеров ПЖ	10.21.1.21
aj_ip_3	В текущем параметре передается ip адресс брокеров ПЖ	10.21.1.21
aj_ip_4	В текущем параметре передается ip адресс брокеров ПЖ	10.21.1.21
aj_ip_5	В текущем параметре передается ip адресс брокеров ПЖ	10.21.1.21
aj_ip_6	В текущем параметре передается ip адресс брокеров ПЖ	10.21.1.21

Имя параметра	Описание	Пример
aj_port_1	В текущих параметрах передается порт endpoint broker ПЖ	9092
aj_port_2	В текущих параметрах передается порт endpoint broker ПЖ	9092
aj_port_3	В текущих параметрах передается порт endpoint broker ПЖ	9092
aj_port_4	В текущих параметрах передается порт endpoint broker ПЖ	9092
aj_port_5	В текущих параметрах передается порт endpoint broker ПЖ	9092
aj_port_6	В текущих параметрах передается порт endpoint broker ПЖ	9092

Параметры messanger

Имя параметра	Описание	Пример
messenger_host_1	В текущем параметре передается Fqdn или ip адресс(если у базы нет FQDN) брокеров kafka SE для сервиса messenger	Kafka-1.sberbank.ru
messenger_host_2	В текущем параметре передается Fqdn или ip адресс(если у базы нет FQDN) брокеров kafka SE для сервиса messenger	Kafka-1.sberbank.ru

Имя параметра	Описание	Пример
messenger_host_3	В текущем параметре передается Fqdn или ip адресс(если у базы нет FQDN) брокеров kafka SE для сервиса messenger	Kafka-1.sberbank.ru
messenger_host_4	В текущем параметре передается Fqdn или ip адресс(если у базы нет FQDN) брокеров kafka SE для сервиса messenger	Kafka-1.sberbank.ru
messenger_ip_1	В текущем параметре передается ip адресс брокеров kafka SE для сервиса messenger	10.21.1.22
messenger_ip_2	В текущем параметре передается ip адресс брокеров kafka SE для сервиса messenger	10.21.1.22
messenger_ip_3	В текущем параметре передается ip адресс брокеров kafka SE для сервиса messenger	10.21.1.22
messenger_ip_4	В текущем параметре передается ip адресс брокеров kafka SE для сервиса messenger	10.21.1.22
messenger_port_1	В текущих параметрах передается порт endpoint брокеров kafka SE	9092
messenger_port_2	В текущих параметрах передается порт endpoint брокеров kafka SE	9092
messenger_port_3	В текущих параметрах передается порт endpoint брокеров kafka SE	9092

Имя параметра	Описание	Пример
messenger_port_4	В текущих параметрах передается порт endpoint брокеров kafka SE	9092

После заполнения всех перечисленных параметров и подготовки ufc pipline можно приступать к установке.

Обновление продукта

Обновление осуществляется так же как первоначальная установка: запускается задача установки с параметром updateMode равным fullInstall. В процедуру установки включен процесс обновления базы данных с помощью утилиты liquibase. Установку новой версии продукта необходимо осуществлять согласно описанным шагам в разделе **Установка** текущего документа. Дополнительных настроек не требуется.

Проверка работоспособности

Для проверки корректности установки:

1. Задача установки завершилась со статусом SUCCESS (успешно).
2. Зайдите в web панель администратора среда контейнеризациии убедитесь, что все pod'ы обновлены и работоспособны (имеют статус ready у всех контейнеров, входящих в pod).
   Пример экрана web панели администратора в разделе pod'ов:

   Признаки:
   - все приложения находятся в статусе "Running";
   - нет pod'ов в статусе "Error";
   - в колонке "Ready" все контейнеры работают: 2/2, 3/3 и т.д.;
   - в колонке "Created" должны появиться pod'ы с текущей датой установки.

     В основном, это касается pod'ов приложения (cb-rates, individual-rates, legal-rates), т.к. именно он чаще всего меняется от дистрибутива к дистрибутиву. Ingress, egress, egress-for-mon, prometheus меняются реже и не всегда будут обновляться. Таким образом не обновившаяся дата создания будет нормальным признаком.
   - Точную проверку успешности обновления проведите через web-интерфейс:
     0. для Deployments;
     1. выберите нужны deployment;

2. перейдите в Replica Sets;

   Активной должна быть та реплика, которая появилась позже. Активная, значит в колонке Status стоят не нули.
3. для Deployment Configs;
4. выберите нужный Deployment Config;
5. перейдите Replication Controllers.

   Активной должна быть та реплика, которая появилась позже. Активная, значит в колонке Status стоят не нули.

6. Запустите задачу по тестированию инсталляции.

Откат

Автоматической процедуры отката нет.

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

Часто встречающиеся проблемы и пути их устранения

В данном разделе собраны наиболее частые проблемы и описаны пути их устранения.

Проблема	**Причина**	**Способ устранения**
Осталась старая версия приложения	Не успела развернуться за таймаут в 30 секунд	Выполнить команду rollout
	Не успела обновиться сетевая конфигурация istio	Повторить команду rollout, через некоторое время. Время ожидания зависит от многих факторов (может составлять до 1,5 часов. Ведется решение данного вопроса).
	Произошли какие−то ошибки	Следует собрать логи с приложения, на котором возникает ошибка и обратиться к технической поддержке. Это могут быть как ошибки в конфигурации, параметрах, сетевой

Проблема	Причина	Способ устранения
	при старте приложения	маршрутизации, настройки баз данных, не верные сертификаты доступа и т.д.

Чек лист валидации установки

После установки необходимо проверить результат выполнения следующих playbook:

- Проверить, что для проекта в среде контейнеризации создались объекты приложений на основании файлов конфигурации: DeploymentConfig, ConfigMap, Service, Route, Pod.
- Проверить, что для проекта в среде контейнеризации создались объекты istio: ingress/egress на основании файлов конфигурации: Deployment, VirtualService, ServiceEntry, DestinationRule, Gateway, ConfigMap.
- Проверить логи на отсутствие ошибок.