

**Инструкция по установке  
программного обеспечения  
"Программный комплекс  
«Онлайн Счет»"**

## ОГЛАВЛЕНИЕ

<b>1. ВВЕДЕНИЕ</b>	<b>3</b>
<b>2. СТРУКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ</b>	<b>4</b>
<b>3. ОПИСАНИЕ СЕРВИСОВ И ИХ НАЗНАЧЕНИЕ</b>	<b>5</b>
<b>4. ПРОЦЕСС УСТАНОВКИ ПО</b>	<b>8</b>

## 1. ВВЕДЕНИЕ

Программное обеспечение "Программный комплекс «Онлайн Счет»" предназначено для дистанционного оказания банковскими организациями услуг по резервированию счета своим Клиентам и их банковскому обслуживанию. Программное обеспечение обеспечивает хранение, учёт и систематизацию данных в процессе банковского обслуживания.

Программное обеспечение "Программный комплекс «Онлайн Счет»" позволяет Клиенту:

- дистанционно управлять своими счетами,
- получать выписку по счету,
- создавать, подтверждать и отправлять на исполнение документы по счету в Банк;
- позволяет просматривать всю информацию о состоянии счета, произведенных операциях по счетам и получать всю необходимую информацию и документы в онлайн режиме.

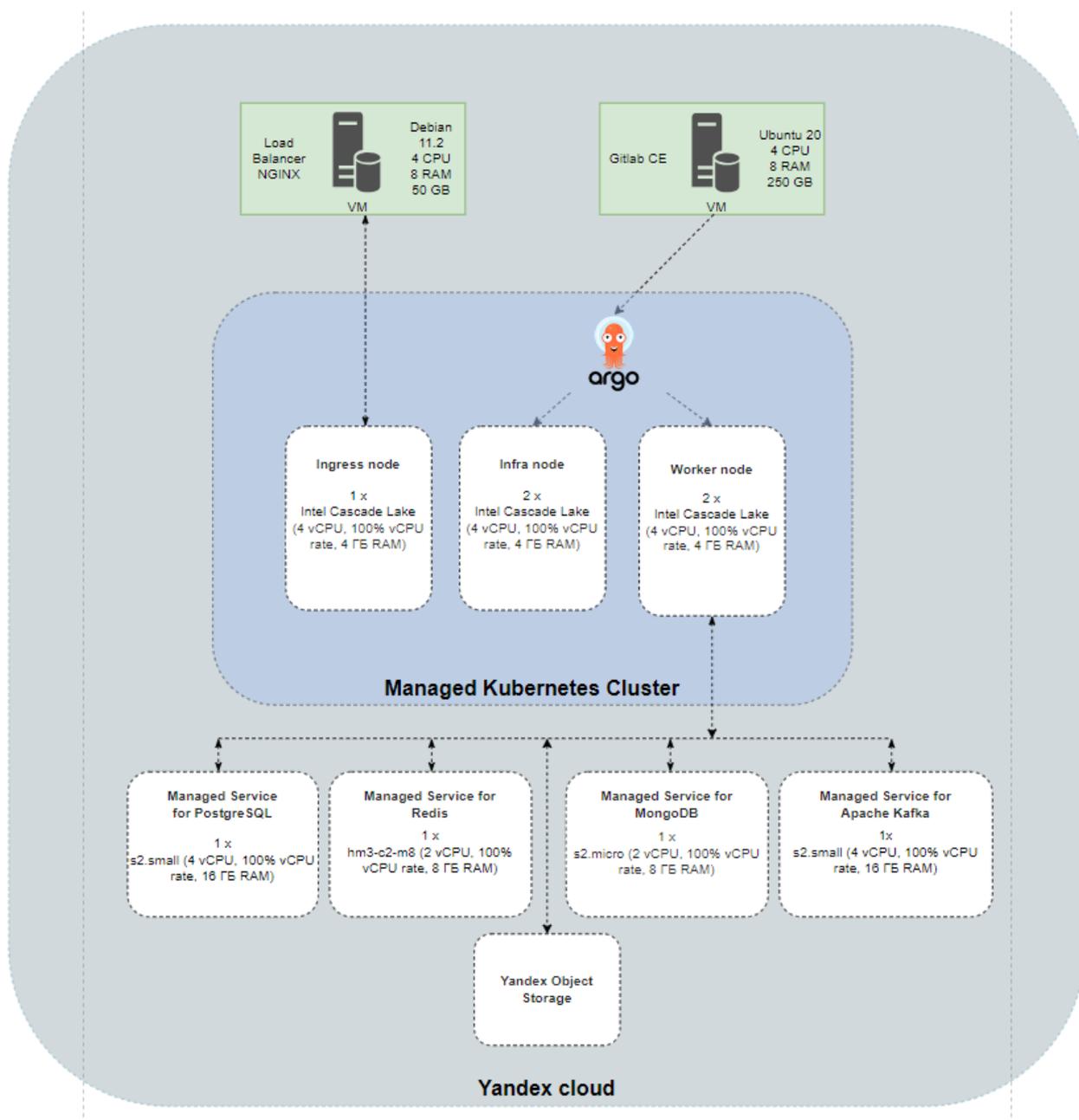
Программное обеспечение "Программный комплекс «Онлайн Счет»" позволяет банковским организациям:

- автоматизировать процесс резервирования счета;
- автоматизировать проверку клиента (скоринг);
- отслеживать этапы рассмотрения заявки при открытии счета;
- отслеживать временные затраты на всех этапах рассмотрения заявок.

Программное обеспечение "Программный комплекс «Онлайн Счет»" поставляется клиенту (заказчику, покупателю) в формате облачного решения – программа "Программный комплекс «Онлайн Счет»" и его данные размещаются на серверах клиента, а также могут размещаться на серверах компании, предоставляющей Дата-центр, где упомянутую компанию и Дата-центр (например, таких компаний как ООО «Селектел», ООО «ВК», ООО «Яндекс.Облако» и т.д.) может выбирать заказчик, а также программное обеспечение может размещаться на серверах правообладателя программного обеспечения "Программный комплекс «Онлайн Счет»".

## 2. СТРУКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

### Схема программного обеспечения



ПО представляет собой набор микросервисов, созданных на языках Java, JavaScript, HTML, CSS.

Функционал реализован в микросервисной архитектуре.

Все микросервисы размещаются в кластерах k8s (Managed Service for Kubernetes), например, развернутых как сервис на базе YandexCloud (<https://cloud.yandex.ru/ru/docs/managed-kubernetes/>)

### 3. ОПИСАНИЕ СЕРВИСОВ И ИХ НАЗНАЧЕНИЕ

#### **arm-spa**

Внутренний кабинет сотрудника. Здесь ведётся работа над Задачами, возникающими в рамках автоматизированного бизнес-процесса открытия расчётных счетов для Клиента.

#### **crm-spa**

Внутренний кабинет сотрудника сопровождения клиентов. Здесь ведётся работа над Обращениями Клиентов по вопросам, связанным расчётными счетами Клиента.

#### **arm-api**

Проверка доступов, проксирование запросов к остальным сервисам, преобразование данных остальных сервисов в формат, удобный для front части кабинета сотрудника (arm-spa).

#### **keycloak**

Сервис аутентификации и авторизации, реализует функционал Sing sign-on. Keycloak интегрирован с Active Directory для получения актуального списка пользователей и ролей.

#### **rko-pa**

Движок бизнес-процессов в формате bpmn 2.0, построенный на Camunda Community Edition. Сервис занимается оркестрацией бизнес-процесса открытия расчётных счетов для Клиента, координирует работу других сервисов. Здесь появляются Задачи, над которыми работают сотрудники.

#### **order-api**

Сервис Заявок предоставляет REST интерфейс для создания и управления Заявками на открытие расчётных счетов для Клиента. Данный сервис автоматически запускает бизнес-процесс в Оркестраторе rko-pa при создании новой Заявки.

#### **claim-api**

Сервис Обращений предоставляет REST интерфейс создания и управления Обращениями Клиентов. Обращения создаются сотрудниками сопровождения клиентов во Внутреннем кабинете сотрудника сопровождения клиентов. Также сервис интегрирован с системой управления проектами и отслеживания ошибок YouTrack. При создании типа Обращения в ИТ автоматически создаются тикеты в YouTrack.

#### **task-api**

Сервис предоставляет REST интерфейс управления Задачами, создающимися в рамках бизнес-процесса открытия расчётных счетов Клиента. Также сервис отслеживает SLA по работе над Задачами.

#### **external-api**

Сервис предоставляет REST интерфейс для автоматического создания Заявок на открытие расчётных счетов и отслеживания статуса созданных Заявок. Предоставляет базовые возможности интеграции с системой.

#### **notification-api**

Сервис предоставляет REST интерфейс для отправки уведомлений Клиентам и сотрудникам по таким каналам как СМС и Email. Сервис интегрирован с внешними сервисами партнеров:

- для рассылки СМС - SMSЦентр,
- рассылки Email - ExpertSender.

Рассылка Email сотрудникам производится через внутренний SMTP сервер.

**scoring-api**

Сервис предоставляет REST интерфейс для осуществления Скоринга Клиента по СТОП-факторам для открытия расчётных счетов. Результаты Скоринга хранятся в базе данных сервиса. Сервис интегрирован с системой Контур.Фокус через адаптер focus-api. Стоп-листы и "чёрные" списки для проведения Скоринга поставляются сервисом-загрузчиком risk-data-loader.

**risk-data-loader**

Сервис по расписанию загружает и актуализирует стоп-листы и "чёрные" списки для сервиса scoring-api. Сервис интегрирован с сайтами Минюста и ФНС. Также часть списков загружается с сетевых дисков.

**product-api**

Сервис предоставляет REST интерфейс для управления информацией о счетах Клиента - зарезервированных, а затем открытых в рамках Заявки Клиента.

**qiwi-gate-api**

Сервис предоставляет REST интерфейс для взаимодействия с qiwi-gate - шлюзом обмена xml сообщениями с АБС Банка. Сервис преобразует данные в формате JSON в формат XML, понятный шлюзу qiwi-gate. Реализована интеграция для автоматического создания карточек Клиентов и взаимосвязанных лиц в АБС Банка.

**smev-gate-api**

Сервис предоставляет REST интерфейс для взаимодействия с шлюзом подключения к СМЭВ на стороне Банка. Интеграция построена через промежуточное звено в виде 1С на стороне Банка.

**document-api**

Сервис предоставляет REST интерфейс для взаимодействия с S3 совместимым хранилищем Object Storage в Yandex.Cloud. Сервис хранит метаданные загружаемых документов, а сами файлы уже лежат в хранилище S3.

**report-api**

Сервис предоставляет REST интерфейс для генерации документов из готовых шаблонов. А также реализует непосредственный функционал генерации документов. Сервис генерирует как документы, необходимые для работы над Заявкой Клиента и открытия счетов, так и документы, запрашиваемые Клиентом в рамках работы с функционалом ДБО в системе.

**client-api**

Сервис предоставляет REST интерфейс для управления метаданными о Клиентах, подавших Заявку на открытие расчётных счетов. Здесь хранятся данные о Клиенте ЮЛ/ИП, а также данные физлиц, связанных с Клиентом. Для автоматического сбора публичных данных Сервис интегрирован с Контур.Фокус и [Dadata](#) через адаптеры.

**focus-api**

Сервис предоставляет REST интерфейс для подключения к Контур.Фокус. Реализует функционал адаптера. Также сервис интегрирован с сайтом ФНС для скачивания выписки ЕГРЮЛ/ЕГРИП с ЭЦП.

**dadata-api**

Сервис предоставляет REST интерфейс для подключения [Dadata](#). Реализует функционал адаптера.

**\*\*\*\*\*.com**

Единый Личный Кабинет Клиента (ЕЛКК). Предоставляет интерфейс доступа к функционалу ДБО для Клиента, позволяет просматривать список открытых счетов и операций по ним, совершать платежи, скачивать выписки и другие банковские документы. Также предоставляет канал связи со службой сопровождения Клиентов посредством встроенного чата.

**login.\*\*\*\*\*.com**

Сервис аутентификации и авторизации, реализует функционал Sing sign-on. Представляет из себя отдельную инсталляцию Keycloak для внешних пользователей – Клиентов. В сервисе реализована кастомизация интерфейса страниц аутентификации согласно дизайн-макетам системы. Также реализована кастомизация процесса аутентификации для возможности Клиенту использовать для аутентификации номер телефона и одноразовый OTP код.

**keycloak-api**

Предоставляет REST интерфейс для доступа к функционалу смены пароля в авторизованной зоне ЕЛКК, а именно на странице управления профилем.

**client-card-api**

Сервис предоставляет REST интерфейс для доступа и управления Карточками Клиентов и взаимосвязанных лиц. Сервис получает актуальные данные из Автоматизированной Банковской Системой (АБС) Банка через сервис-маршрутизатор abs-replication-api. Также Сервис отвечает за хранение данных о наличии и характеристиках доступа Клиента.

**account-api**

Сервис Счетов предоставляет REST интерфейс доступа к информации по счетам Клиента, актуальным остаткам денежных средств на счёте, а также наложенным блокировкам (при их наличии). Сервис получает актуальные данные из АБС Банка через сервис-маршрутизатор abs-replication-api.

**transaction-api**

Сервис Транзакции предоставляет REST интерфейс для доступа к совершенным операциям с денежными средствами в рамках расчетного счета Клиента. Сервис получает актуальные данные из АБС Банка через сервис-маршрутизатор abs-replication-api. А также присутствует интеграция сервисом Платежей payment-api для определения, какие именно транзакции были созданы через ЕЛКК.

**payment-api**

Сервис Платежей предоставляет REST интерфейс для создания, редактирования и подписания платежных поручений Клиентом в рамках счетов, открытых в системе ПК Онлайн Счет. Сервис реализует функционал проверки и валидации данных Платежа, также интегрирован с сервисом Комиссий (commission-api) для учета транзакционной комиссии. Платежные поручения направляются далее в Банк для фактического проведения и осуществления движения денежных средств. Механизм обработки Платежа асинхронный, построен на базе движка бизнес-процессов в формате bpmn 2.0 Camunda Community Edition (dbo-ra). Результаты обработки Платежей поступают из АБС Банка через сервис-маршрутизатор abs-replication-api.

**abs-replication-api**

Сервис является адаптером и маршрутизатором данных, поступающих из АБС Банка в систему ПК Онлайн Счет. Получение данных реализовано посредством подключения к распределенному логгу сообщений Apache Kafka. Сервис получает данные, сохраняет их в свою базу данных, а затем распределяет по целевым сервисам - client-card-api, account-api, transaction-api, payment-api.

### commission-api

Сервис Комиссий предоставляет REST интерфейс для доступа к функционалу расчёта и учёта транзакционной комиссии, а также следит за своевременным взиманием с Клиента ежемесячной комиссии.

### dbo-pa

Движок бизнес-процессов в формате bpmn 2.0, построенный с использованием Camunda Community Edition. Сервис занимается оркестрацией автоматических асинхронных процессов по обработке платежных поручей Клиентов, списания транзакционной комиссии и ежемесячной комиссии за пользование услугами системы ПК Онлайн Счет. Также в Сервисе реализован автоматический процесс создания и настройки учетной записи Клиента для предоставления доступа к системе ПК Онлайн Счет. Интегрирован с сервисом qiwi-gate-api для взаимодействия с АБС Банка, а также с другими сервисами.

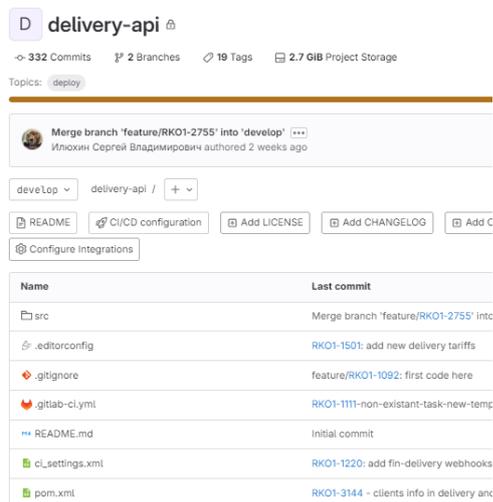
## 4. ПРОЦЕСС УСТАНОВКИ ПО

После приобретения заказчику предоставляется доступ к репозиторию с экземпляром ПО или предоставляется ссылка для загрузки файлов для установки (развертывания ПО) в том числе с использованием развернутого репозитория на мощностях заказчика.

Далее расписаны шаги сборки и “деплой” микросервисов в кластер k8s на примере - delivery-api, применимые к остальным частям ПО.

Сборка, сканирование и “деплой” осуществляются средствами ci-cd с использованием git-репозитория, например, git-репозитория «Gitlab Community Edition», на примере которого приведено дальнейшее описание.

Файлы, необходимые для установки (развертывания) ПО размещаются в упомянутом репозитории.



Следующим этапом следует сборка пакетов и самого кода средствами Gitlab. На рисунке ниже данные этапы выделены цветом.

### Merge branch 'feature/RKO1-2755' into 'develop'

Passed Илюхин Сергей Владимирович created pipeline for commit `be963e17` finished 2 weeks ago

For `develop`

latest 5 Jobs 4 minutes 17 seconds, queued for 0 seconds

Pipeline Needs Jobs 5 Tests 0 Security Licenses 13

Результатом сборки является docker-образ, который размещается в `container_registry` самого проекта.

После того, как готов docker- image, средствами Gitlab происходит коммит в проект с helm-чартами нужного проекта.

```

Code owners Assign users and groups as approvers for specific file changes. Learn more.
demo rko / delivery-api / Deployment.yml
Deployment.yml 1.59 KB
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: delivery-api
5   labels:
6     app.kubernetes.io/name: deDelivery-api
7 spec:
8   replicas: 1
9   revisionHistoryLimit: 0
10 strategy:
11   type: RollingUpdate
12   rollingUpdate:
13     maxUnavailable: 0
14     maxSurge: 1
15 selector:
16   matchLabels:
17     app.kubernetes.io/name: deDelivery-api
18 template:
19   metadata:
20     labels:
21       app.kubernetes.io/name: deDelivery-api
22   spec:
23     imagePullSecrets:
24       - name: registry-credentials
25     volumes:
26       - name: deDelivery-api
27         configMap:
28           name: deDelivery-api
29     containers:
30       - name: deDelivery-api
31         image: "gitlab.service.rowi.tech:80/rko/delivery-api:release-04-23-07"
32         imagePullPolicy: Always
33         volumeMounts:
34           - name: deDelivery-api
35             mountPath: /etc/config
36             readOnly: true
37         env:
38           - name: SPRING_CONFIG_IMPORT
39             value: file:/etc/config/application.yml
40           - name: JAVA_OPTS
41             value: -Xms256m -Xmx2024m
42         ports:
43           - name: http
44             containerPort: 8080
45             protocol: TCP
46         livenessProbe:
47           initialDelaySeconds: 20
48           httpGet:
49             path: /actuator/health/liveness
50             port: http
51         readinessProbe:
52           httpGet:
53             path: /actuator/health/readiness
54             port: http
55     resources:
56       limits:

```

Ниже выделен данный шаг процесса ci-cd.

RKO / delivery-api / Pipelines / #80096

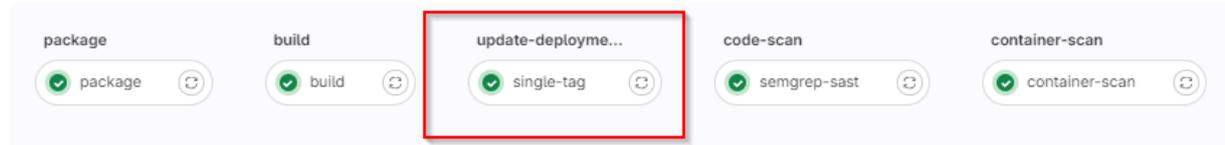
## Merge branch 'feature/RKO1-2755' into 'develop'

Passed Илюхин Сергей Владимирович created pipeline for commit `be963e17` finished 2 weeks ago

For `develop`

latest 5 Jobs 4 minutes 17 seconds, queued for 0 seconds

Pipeline Needs Jobs 5 Tests 0 Security Licenses 13



Следующим этапом после сборки docker image и обновления тэга в манифесте сервиса (deployment.yaml) – запускается шаг "code-scan", в котором выполняется сканирование исходного кода различными методами в зависимости от языка, на котором написаны исходный код и расширения манифестов кода.

RKO / delivery-api / Pipelines / #80096

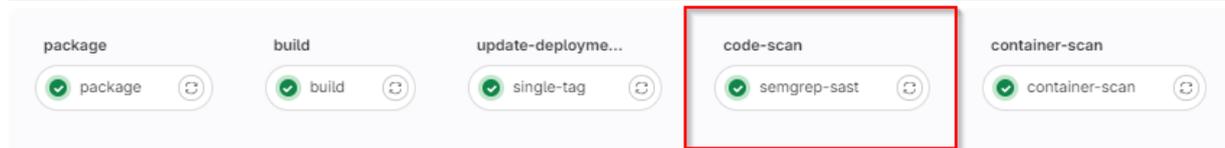
## Merge branch 'feature/RKO1-2755' into 'develop'

Passed Илюхин Сергей Владимирович created pipeline for commit `be963e17` finished 2 weeks ago

For `develop`

latest 5 Jobs 4 minutes 17 seconds, queued for 0 seconds

Pipeline Needs Jobs 5 Tests 0 Security Licenses 13



Container scan – это сканирование каждого слоя docker images на известные уязвимости. Уязвимости делятся на категории: critical, high, medium и low.

RKO / delivery-api / Pipelines / #80096

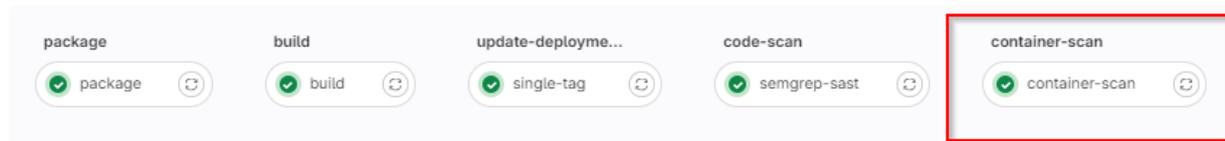
## Merge branch 'feature/RKO1-2755' into 'develop'

Passed Илюхин Сергей Владимирович created pipeline for commit `be963e17` finished 2 weeks ago

For `develop`

latest 5 Jobs 4 minutes 17 seconds, queued for 0 seconds

Pipeline Needs Jobs 5 Tests 0 Security Licenses 13



Финальным шагом "депоя" микросервиса в кластер k8s – является непосредственная синхронизация состояния сущностей (манифестов) с использованием GitOps инструмента ArgoCD.

The screenshot displays the ArgoCD web interface for an application named 'rko-demo-delivery-api'. At the top, there are navigation tabs: APP DETAILS, APP DIFF, SYNC, SYNC STATUS, HISTORY AND ROLLBACK, DELETE, and REFRESH. The main content area is divided into three sections: APP HEALTH, SYNC STATUS, and LAST SYNC. The APP HEALTH section shows a green heart icon and the word 'Healthy'. The SYNC STATUS section shows a green checkmark and the text 'Synced to demo (b3a25ba)', with a note that 'Auto sync is enabled' and a comment 'Update file ConfigMap.yml'. The LAST SYNC section shows a green checkmark and the text 'Sync OK to 8b649e7', with a timestamp 'Succeeded 10 days ago (Mon Jan 15 2024 12:46:38 GMT+0300)' and a comment 'Updated images tag for rko services'. Below these sections is a large diagram showing the application's structure. On the left, a central node 'rko-demo-delivery-api' is connected to four nodes: 'cm', 'svc', 'deploy', and 'ing'. The 'cm' node is connected to 'delivery-api' (cm). The 'svc' node is connected to 'delivery-api' (svc). The 'deploy' node is connected to 'delivery-api' (deploy) and 'rs'. The 'ing' node is connected to 'delivery-api' (ing). The 'rs' node is connected to 'delivery-api-57dbd4c456' (rs). The 'delivery-api-57dbd4c456' node is connected to 'delivery-api-57dbd4c456-dnm...' (pod). The diagram also shows other nodes like 'delivery-api-rjsqg' (endpointslice) and 'delivery-api' (ep). Each node has a status icon and a timestamp (e.g., '3 months', '10 days', '3 days').