Инструкция по установке экземпляра ПО «Автоматизированная банковская система "АРНУВО"»

Оглавление

1. Требования к системе

В данной инструкции рассматривается один из вариантов развёртывания программы для ЭВМ «Автоматизированная банковская система "АРНУВО"» (далее также - АРНУВО, АБС «АРНУВО», микросервисная автоматизированная банковская система, микросервисная АБС «АРНУВО», АБС, система, ПО) на контейнерной инфраструктуре Kubernetes, используемой в качестве платформы для оркестрации микросервисов программы для ЭВМ «Автоматизированная банковская система "АРНУВО"».

Минимальная системная конфигурация:
Минимальная конфигурация предназначена для демонстрационной/тестовой установки с ограниченным набором компонентов и минимальным набором бизнес-модулей (например, SmART ESB + один-два функциональных микросервиса). Все инфраструктурные компоненты допускается размещать на одной виртуальной машине.

Минимальные требования:
- CPU: 6 vCPU;
- RAM: 16 GB;
- Disk: 100 GB;

Разворачиваемые компоненты в минимальной конфигурации:
- PostgreSQL (единый инстанс);
- NATS (1 нода);
- Redis/Mongo;
- OpenSearch (1 нода, базовая конфигурация);
- SmART ESB (NodeJS);
- 1–2 функциональных микросервиса (напр. Billing и Reporting) (Java/NodeJS)

Конфигурация обеспечивает техническую работоспособность системы с пустыми справочниками и минимальной тестовой нагрузкой.

Рекомендуемая системная конфигурация:
Рекомендуемая конфигурация предназначена для типового промышленного контура, обеспечивающего развёртывание всего набора бизнес-модулей и полного комплекта микросервисов. Расчёт приведён для нагрузки ~ 3 000 000 клиентов/договоров/счетов и ~ 1 500 000 транзакций в сутки.

Суммарные рекомендуемые ресурсы для кластера из 3х физических машин:
Кластер приложений (Kubernetes / VM)
СУБД PostgreSQL (primary + replica)
Инфраструктурные компоненты
Функциональные микросервисы
- CPU: ~ 96 CPU (96 Cores|192 Threads);
- RAM: ~ 512 GB;
- Диск: ~ 10 TB.

2. Основные компоненты

В таблице приведены основные компоненты (микросервисы) АБС «АРНУВО»:

Компоненты	Назначение
tools:acctranscard	Обработка операций по картам и иным договорам (формирование проводок);
tools:acctransloan	Обработка кредитных операций (формирование проводок);
auth	Микросервис авторизации пользователей;
card	Микросервис работы с договорами;
conf	Микросервис для взаимодействия с хранилищем логов, для получения информации о стеке, взаимодействия с git;
etl	Микросервис преобразования данных, обработки файлов различных форматов;
fee	Микросервис тарификации (биллинга);

fs	Микросервис работы с файлами (сохранение, извлечение в файловое хранилище);
http	Микросервис для работы с http сообщениями;
interest	Микросервис расчета процентов и графиков платежей;
msg	Микросервис рассылки сообщений;
postgres	Микросервис для работы с БД Postgres;
rate_calc	Микросервис конвертации;
redis	Микросервис для работы с БД redis;
report	Микросервис формирования отчетов;
sftp	Микросервис для работы с файловой системой (извлечение, сохранение файлов);
trans	Микросервис работы с транзакциями;
v4app	Микросервис сбора логов, для поиска процессов в mongo, для поиска элементов на которых произошла предусмотренная остановка процесса (usertask);
<конфигурационные приложения>	Микросервисы формируемые из настроек конкретного проекта.

Экземпляр ПО «Автоматизированная банковская система "АРНУВО"» предоставляется после приобретения ПО

3. Ход установки

3.1. Установка Kubernetes-кластера

Описан вариант установки Kubernetes-кластера на набор хостов (железных, либо виртуальных машин) с предустановленной операционной системой Linux (например, Debian, Ubuntu, РЕД ОС, Операционная система специального назначения «Astra Linux Special Edition», Операционная система общего назначения "Astra Linux Common Edition" и другие), версия ядра не ниже 5.15.

Настоящая инструкция предусматривает развёртывание отказоустойчивого кластера Kubernetes в пределах одного центра обработки данных. В целях обеспечения высокой доступности инфраструктура должна включать не менее трёх мастер-нод и трёх рабочих (worker) нод. Указанная конфигурация обеспечивает устойчивость к отказу любого отдельного узла и гарантирует непрерывность функционирования системы в соответствии с требованиями надёжности.

Для установки используется стандартная утилита kubeadm (Apache License 2.0, https://github.com/kubernetes/kubeadm/blob/main/LICENSE, https://github.com/kubernetes/kubeadm). Процесс установки автоматизирован с помощью средства автоматизации Ansible (GPL v3.0, https://github.com/ansible/ansible/blob/devel/COPYING, https://github.com/ansible/ansible) и состоит в последовательном исполнении ряда скриптов.

3.1.1. Подготовка

Инструкция включает процедуры развёртывания инфраструктурных сервисов, необходимых для работы приложений, таких как MongoDB, NATS-кластер, Elasticsearch/OpenSearch и Redis-кластер. Указанные компоненты не являются частью АБС «АРНУВО» и используются как внешние инфраструктурные зависимости.

Если перечисленные сервисы уже развёрнуты в целевом контуре, допускается использование существующей инфраструктуры с изменением адресов и параметров подключения к соответствующим эндпоинтам.

Перед установкой (применением ansible-скриптов) необходимо настроить inventory-файл, где указывается перечень целевых хостов - адреса, способ подключения, группы (роли) - файл hosts.ini. Все целевые хосты делятся на 2 основные группы:

- masters - хосты для организации control-plane (уровень управления); для отказоустойчивых инсталляций обычно используется 3 хоста, реже 5; возможен 1 для тестовых контуров;
- workers - остальные, для размещения рабочей нагрузки;

Последние, в свою очередь, могут выступать в ролях (включаться в соответствующие группы в инвентори):

- ingresses - используются для размещения подов ингресс-контроллера;
- logcollectors - используются для размещения (выделения) средств обработки/накопления логов, метрик и прочей служебной нагрузки;

Далее, требуется настройка параметров в group_vars/all.yml

```
cluster_name: "patent"
kubernetes_version_minor: "1.32"
crictl_version: "1.32.0"
cni_plugins_version: "1.7.1"
kubernetes_service_subnet: "100.99.0.0/16"
kube_dns_cluster_ip: "100.99.0.10"
kubernetes_pod_subnet: "100.98.0.0/16"
kubernetes_api_ep_domains:
  - "kubeapi.patent.art"
kubernetes_dns_domain: "cluster.local"
oidc_issuer_url: "https://dex.patent.art"
oidc_client_id: "dex-patent-auth"

kube_apiserver_port: 6443
loadbalancer_apiserver_port: 6443
kube_apiserver_endpoint: https://localhost:6443
nginx_image_repo: "nginx:1.19"
nginx_config_dir: "/etc/nginx"
kube_manifest_dir: /etc/kubernetes/manifests
loadbalancer_apiserver_keepalive_timeout: 5m
loadbalancer_apiserver_healthcheck_port: 8081
```

Особо обратить внимание на параметры:

- cluster_name
- kubernetes_api_ep_domains
- oidc_issuer_url
- oidc_client_id

Их требуется кастомизировать под конкретную инсталляцию и в соответствии со средой установки (доменные имена).

3.1.2. Выполнение

В репозитории лежат следующие ansible-playbooks:

- 1-unattended-upgrades-disable.yml
- 2-limits.yml
- 3-init-allhosts.yml
- 4-api-proxy.yml
- 5-k8s-install-part1.yml
- 5.1-set-labels.yml

Каждому соответствует свой шелл-скрипт .sh, в котором можно указать для ssh-agent ssh-ключ, применяемый для доступа к целевым хостам. Порядок исполнения соответствует нумерации в именах файлов.

После исполнения получаем преднастроенный установленный Kubernetes-кластер с CNI Cilium, core-dns, kubectl (на master-хостах). А также сгенерированным самоподписанным сертификатом, который далее будет использоваться в качестве CA-cert в cert-manager для обеспчения tls-протокола для ингресс-ендпоинтов. Этот CA-cert можно получить так:

```
kubectl get secret -n cert-manager local-ca-cert -o jsonpath='{.data.tls\.crt}' | base64 -d
```

3.2. Установка инфраструктурных компонентов кластера

3.2.1. Подготовка рабочей среды

В качестве рабочей среды может выступать компьютер с доступом к хостам кластера, либо одна из мастер-нод самого кластера. Если это не мастер-нода, то здесь должен быть установлен kubectl и присутствовать ~/.kube/config с действующими сертификатами для доступа к kube-api кластера.

Для установки набора инфраструктурных компонентов кластера используется GitOps-инструмент FluxCD. Установка FluxCD CLI:

```
curl -s https://fluxcd.io/install.sh | sudo bash
```

Установка компонент FluxCD на кластер:

```
flux install
```

Также, для доступа к чувствительным параметрам установки (секретам) понадобятся утилиты: age (https://github.com/FiloSottile/age), установка:

```
apt install age
```

и SOPS (https://github.com/getsops/sops), установка:

```
curl -LO https://github.com/getsops/sops/releases/download/v3.9.1/sops-v3.9.1.linux.amd64
mv sops-v3.9.1.linux.amd64 /usr/local/bin/sops
chmod +x /usr/local/bin/sops
```

3.2.2. Настройка параметров деплоя

Для внесения изменений этот репозиторий клонируется на локальную рабочую среду, подготовленную на предыдущем пункте.
Перед установкой нужно настроить параметры, размещённые в файлах:

- vars/vars.yaml
- vars/secrets.yaml

vars/secrets.yaml зашифрован SOPS. В данном демо-репозитории для шифрации использован ключ:

```
AGE-SECRET-KEY-1DTCAM5SJW0ZWLYMKD9R9ZH9JTXSTPQP4K4WC6X26SNYZJT9JRLZSVACKG4
```

(внимание: это демо-ключ; в реальной работе ключ должен содержаться приватно)
Для изменения vars/secrets.yaml сначала нужно его расшифровать:

```
sops -d -i --age 'AGE-SECRET-KEY-1DTCAM5SJW0ZWLYMKD9R9ZH9JTXSTPQP4K4WC6X26SNYZJT9JRLZSVACKG4'
vars/secrets.yaml
```

После изменения - зашифровать (при этом используются паблик-ключи из .sops.yaml, один из которых соответствует демо-ключу):

```
sops -e -i vars/secrets.yaml
```

Далее, изменения нужно закоммитить и отправить в репозиторий установки. Выданный аккаунт доступа к репозиторию (u-pat) не имеет прав на запись в данный репозиторий, поэтому, если изменения в параметрах требуется сохранить, то сначала потребуется создать форк репозитория установки (например, https://pat.git.art-fintech.ru/mabs-deploy/2-infra --> https://pat.git.art-fintech.ru/u-pat/2-infra), и далее, изменения коммитить уже в форк. Также, при инициализации (см. следующий раздел) потребутся изменить URL репозитория в команде "flux create source git" и параметр spec/url в манифесте GitRepository flux-system в файле cluster/flux-system/git-repo.yaml

3.2.3. Установка

Создать секрет с SOPS(age)-ключом, который будет использоваться FluxCD для доступа к секретам в репозитории установки:

```
kubectl create secret generic sops-age -n flux-system --from-literal=flux-infra-
patent.agekey=AGE-SECRET-KEY-1DTCAM5SJW0ZWLYMKD9R9ZH9JTXSTPQP4K4WC6X26SNYZJT9JRLZSVACKG4
```

Проинициализировать FluxCD из данного репозитория (либо из форка - см. предыдущий раздел), например:

```
flux create source git flux-system \
    --url=https://pat.git.art-fintech.ru/mabs-deploy/2-infra \
    --username=u-pat \
    --password=<password> \
    --branch=master \
    --interval=3m0s

flux create kustomization flux-system \
    --source=flux-system \
    --path="./cluster" \
    --prune=true \
    --interval=10m
```

в команде "flux create source git" заменить <password> на актуальный пароль пользователя u-pat, а также параметр --url, если должен использоваться форк (если потребовались изменения в репо установки).
Далее, автоматически начнется процесс установки. Статус выполнения можно проверить командой:

```
flux get all
```

В данной установке намеренно не используется вариант инициализации "flux bootstrap", т.к. в

одном из вариантов (без форка) описана возможность инсталляции из "read only" репозитория. "flux bootstrap" же предполагает полный доступ к репо установки, т.к. добавляет туда код инициализации самого FluxCD. Для реальных, не демо, установок рекомендуется использовать "flux bootstrap", чтобы код FluxCD поддерживался и обновлялся им же самим.

3.3. Установка прикладного стэка

Рабочая среда установки стэка совпадает со средой установки инфраструктурных компонентов кластера (см. соответствующий раздел "Подготовка рабочей среды"). Используются инструменты FluxCD, SOPS (+age).
Перед установкой требуется настроить параметры деплоя (секреты), расположенные в файле vars/secrets.yaml В частности, обратить внимание на параметры коннекта к БД. Порядок их изменения также совпадает с описанием из раздела установки инфраструктурных компонентов кластера ("2-infra") - клонируем (с форком, если требуется), расшифровываем, меняем, шифруем, коммитим и отправляем в репозиторий. См. Настройка параметров деплоя.

3.3.1. Установка

Проинициализировать FluxCD из данного репозитория (либо из форка), например:

```
flux create source git stack \
    --url=https://pat.git.art-fintech.ru/mabs-deploy/4-stack \
    --username=u-pat \
    --password=<password> \
    --branch=master \
    --interval=3m0s

flux create kustomization stack-entry \
    --source=stack \
    --path="./flux-entry" \
    --prune=true \
    --interval=10m
```

в команде "flux create source git" заменить на актуальный пароль пользователя u-pat, а также параметр --url, если должен использоваться форк (если потребовались изменения в репо установки). Далее, автоматически начнется процесс установки. Статус выполнения можно проверить командой (рисунок 1):

```
flux get all
```

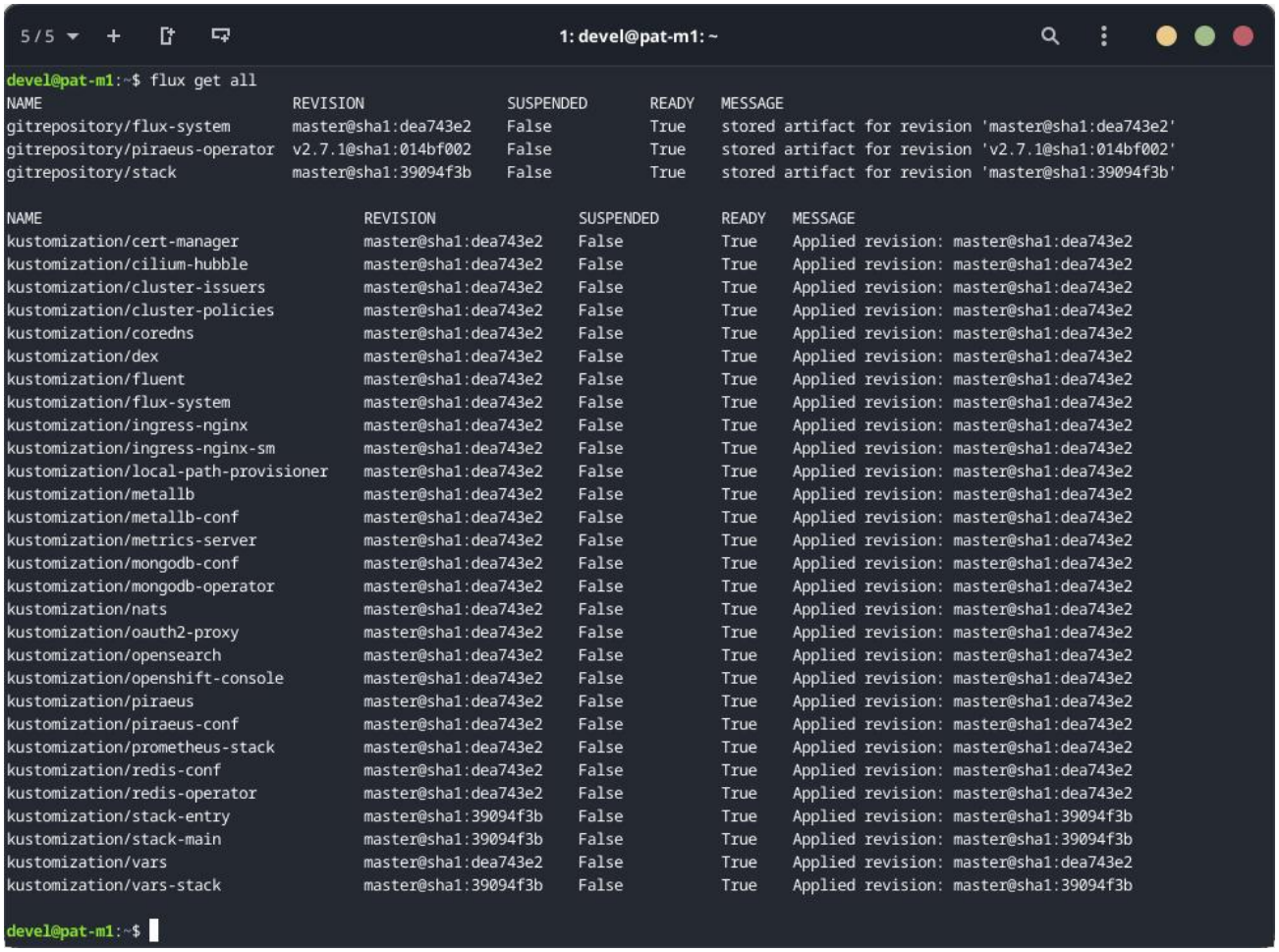

Рисунок 1

Чтобы убедиться в корректной работе кластера, необходимо проверить состояние всех Pod’ов. Все компоненты должны находиться в статусе Running, а показатель READY должен иметь вид n/n. Для просмотра состояния используйте команду (рисунок 2):

```
kubectl get pod -A
```

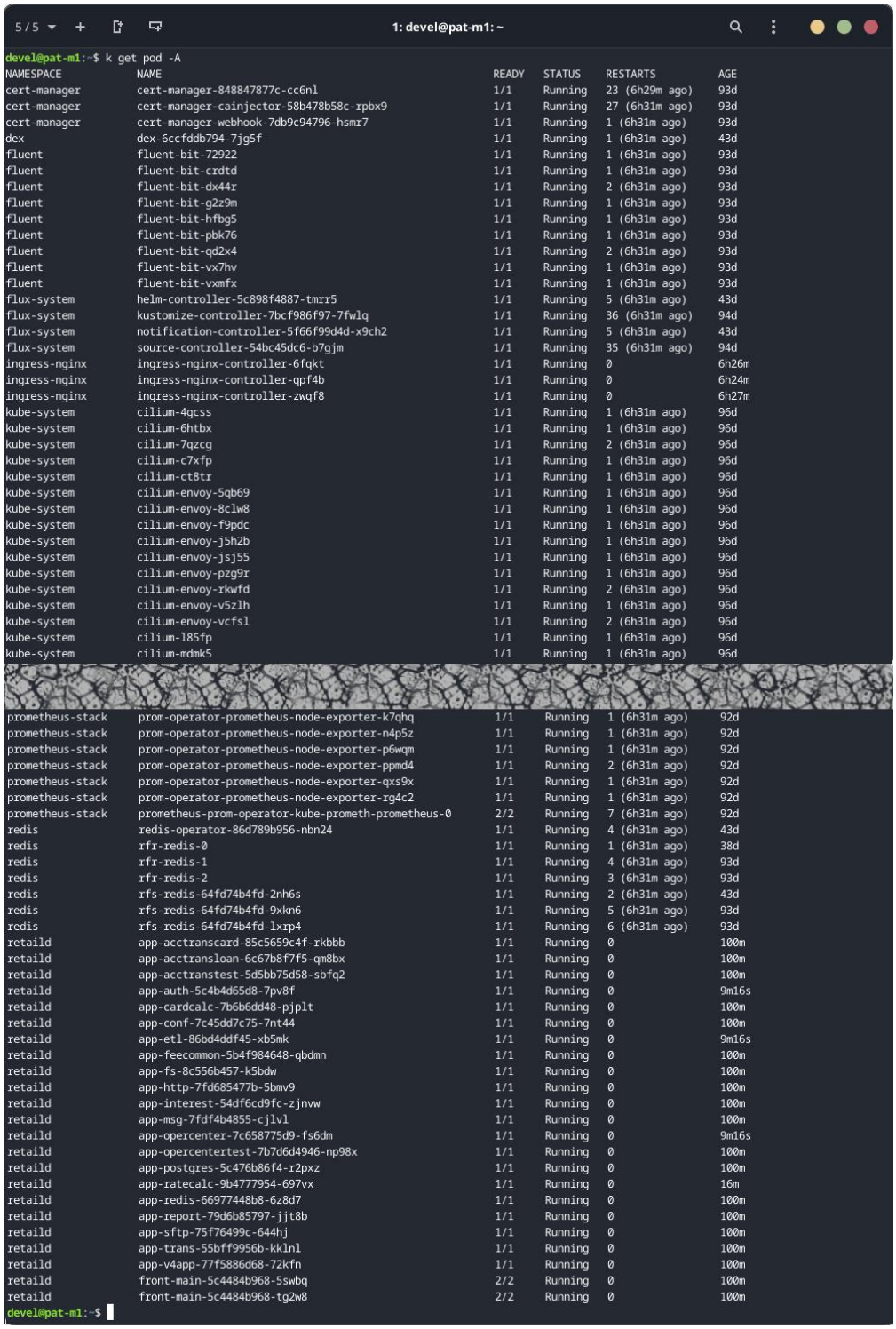

Рисунок 2

В ПО «Автоматизированная банковская система "АРНУВО"» реализовано большое количество модулей, каждый из которых отвечает за определённое направление работы — от клиентских операций и биллинга до отчётности, администрирования и интеграций.
В зависимости от выбранного модуля стартовое окно может иметь различный вид и состав элементов интерфейса, адаптируясь под функции и задачи конкретного пользователя.
Внешний вид стартового экрана может незначительно отличаться в разных модулях, но структура интерфейса остаётся единой и интуитивно понятной.
Пример типового стартового окна представлен на рисунке 3.

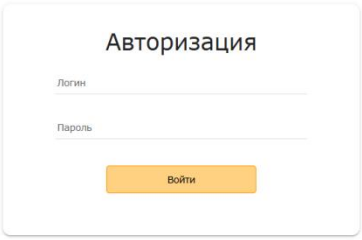

Рисунок 3 Стартовое окно ПО «Автоматизированная банковская система "АРНУВО"»