

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

О г л а в л е н и е

1. Требования к системе	3
2. Основные компоненты	3
3. Ход установки	4
3.1. Установка Kubernetes-кластера	4
3.2. Установка инфраструктурных компонентов кластера	5
3.3. Установка прикладного стека	7

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: 8 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: ~ 768 GB;
- Диск: ~ 18 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 kubectl (Apache License 2.0, <https://github.com/kubernetes/kubectl/blob/main/LICENSE>, <https://github.com/kubernetes/kubectl>). 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, реж 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 обеспечения tls-протокола для ингресс-эндпоинтов. 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/getops/sops>), установка:

```
curl -LO https://github.com/getops/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 публич-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) потребуются 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") - копируем (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
```

```
5 / 5 + [ ] 1: devel@pat-m1: ~
devel@pat-m1:~$ flux get all
NAME                                REVISION                SUSPENDED    READY    MESSAGE
gitrepository/flux-system           master@sha1:dea743e2    False       True     stored artifact for revision 'master@sha1:dea743e2'
gitrepository/piraeus-operator      v2.7.1@sha1:014bf002    False       True     stored artifact for revision 'v2.7.1@sha1:014bf002'
gitrepository/stack                 master@sha1:39094f3b     False       True     stored artifact for revision 'master@sha1:39094f3b'

NAME                                REVISION                SUSPENDED    READY    MESSAGE
kustomization/cert-manager          master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/cilium-hubble         master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/cluster-issuers       master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/cluster-policies      master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/coredns               master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/dex                   master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/fluent                master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/flux-system           master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/ingress-nginx         master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/ingress-nginx-sm      master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/local-path-provisioner master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/metallb               master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/metallb-conf          master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/metrics-server        master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/mongodb-conf          master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/mongodb-operator      master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/nats                  master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/oauth2-proxy          master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/opensearch            master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/openshift-console     master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/piraeus               master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/piraeus-conf          master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/prometheus-stack      master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/redis-conf            master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/redis-operator        master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/stack-entry           master@sha1:39094f3b     False       True     Applied revision: master@sha1:39094f3b
kustomization/stack-main            master@sha1:39094f3b     False       True     Applied revision: master@sha1:39094f3b
kustomization/vars                  master@sha1:dea743e2    False       True     Applied revision: master@sha1:dea743e2
kustomization/vars-stack            master@sha1:39094f3b     False       True     Applied revision: master@sha1:39094f3b

devel@pat-m1:~$
```

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
```



```

1: devel@pat-m1: ~
devel@pat-m1:~$ k get pod -A

```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
cert-manager	cert-manager-84884777c-cc6n1	1/1	Running	23 (6h29m ago)	93d
cert-manager	cert-manager-cainjector-58b478b58c-rpbx9	1/1	Running	27 (6h31m ago)	93d
cert-manager	cert-manager-webhook-7db9c94796-hsmr7	1/1	Running	1 (6h31m ago)	93d
dex	dex-6ccfdbb794-7jg5f	1/1	Running	1 (6h31m ago)	43d
fluent	fluent-bit-72922	1/1	Running	1 (6h31m ago)	93d
fluent	fluent-bit-crdtd	1/1	Running	1 (6h31m ago)	93d
fluent	fluent-bit-dx44z	1/1	Running	2 (6h31m ago)	93d
fluent	fluent-bit-g229m	1/1	Running	1 (6h31m ago)	93d
fluent	fluent-bit-hfbg5	1/1	Running	1 (6h31m ago)	93d
fluent	fluent-bit-pbk76	1/1	Running	1 (6h31m ago)	93d
fluent	fluent-bit-qd2x4	1/1	Running	2 (6h31m ago)	93d
fluent	fluent-bit-vx7hv	1/1	Running	1 (6h31m ago)	93d
fluent	fluent-bit-vmfx	1/1	Running	1 (6h31m ago)	93d
flux-system	helm-controller-5c898f4887-tmrr5	1/1	Running	5 (6h31m ago)	43d
flux-system	kustomize-controller-7bcf986f97-7fwlq	1/1	Running	36 (6h31m ago)	94d
flux-system	notification-controller-5f66f99d4d-x9ch2	1/1	Running	5 (6h31m ago)	43d
flux-system	source-controller-54bc45dc6-b7gjm	1/1	Running	35 (6h31m ago)	94d
ingress-nginx	ingress-nginx-controller-6fqkt	1/1	Running	0	6h26m
ingress-nginx	ingress-nginx-controller-qpf4b	1/1	Running	0	6h24m
ingress-nginx	ingress-nginx-controller-zwqf8	1/1	Running	0	6h27m
kube-system	cilium-4gcxs	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-6htbx	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-7qzcg	1/1	Running	2 (6h31m ago)	96d
kube-system	cilium-c7xfp	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-ct8tr	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-envoy-5qb69	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-envoy-8clw8	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-envoy-f9pdc	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-envoy-jsh2b	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-envoy-jsj55	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-envoy-pzg9r	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-envoy-rkwfd	1/1	Running	2 (6h31m ago)	96d
kube-system	cilium-envoy-v5zlh	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-envoy-vcfsl	1/1	Running	2 (6h31m ago)	96d
kube-system	cilium-l85fp	1/1	Running	1 (6h31m ago)	96d
kube-system	cilium-mdmk5	1/1	Running	1 (6h31m ago)	96d
prometheus-stack	prom-operator-prometheus-node-exporter-k7hqk	1/1	Running	1 (6h31m ago)	92d
prometheus-stack	prom-operator-prometheus-node-exporter-n4p5z	1/1	Running	1 (6h31m ago)	92d
prometheus-stack	prom-operator-prometheus-node-exporter-p6wqm	1/1	Running	1 (6h31m ago)	92d
prometheus-stack	prom-operator-prometheus-node-exporter-ppmd4	1/1	Running	2 (6h31m ago)	92d
prometheus-stack	prom-operator-prometheus-node-exporter-qxs9x	1/1	Running	1 (6h31m ago)	92d
prometheus-stack	prom-operator-prometheus-node-exporter-rg4c2	1/1	Running	1 (6h31m ago)	92d
prometheus-stack	prometheus-prom-operator-kube-prometh-prometheus-0	2/2	Running	7 (6h31m ago)	92d
redis	redis-operator-86d789b956-nbn24	1/1	Running	4 (6h31m ago)	43d
redis	rfr-redis-0	1/1	Running	1 (6h31m ago)	38d
redis	rfr-redis-1	1/1	Running	4 (6h31m ago)	93d
redis	rfr-redis-2	1/1	Running	3 (6h31m ago)	93d
redis	rfs-redis-64fd74b4fd-2nh6s	1/1	Running	2 (6h31m ago)	43d
redis	rfs-redis-64fd74b4fd-9xkn6	1/1	Running	5 (6h31m ago)	93d
redis	rfs-redis-64fd74b4fd-lxrp4	1/1	Running	6 (6h31m ago)	93d
retaild	app-acctranscard-85c5659c4f-rkbbb	1/1	Running	0	100m
retaild	app-acctransloan-6c67b87f75-qm8bx	1/1	Running	0	100m
retaild	app-acctranstest-5d5bb75d58-sbfq2	1/1	Running	0	100m
retaild	app-auth-5c4bd46d58-7pv8f	1/1	Running	0	9m16s
retaild	app-cardcalc-7b6b6dd48-pjplt	1/1	Running	0	100m
retaild	app-conf-7c45dd7c75-7nt44	1/1	Running	0	100m
retaild	app-etl-86bd4dd4f45-xb5mk	1/1	Running	0	9m16s
retaild	app-feecommon-5b4f984648-qbdmn	1/1	Running	0	100m
retaild	app-fs-8c556b457-k5bdw	1/1	Running	0	100m
retaild	app-http-7fd685477b-5bm9	1/1	Running	0	100m
retaild	app-interest-54df6cd9fc-zjnwv	1/1	Running	0	100m
retaild	app-msg-7fdf4b4855-cjlvl	1/1	Running	0	100m
retaild	app-opercenter-7c658775d9-fs6dm	1/1	Running	0	9m16s
retaild	app-opercentertest-7b7d6d4946-np98x	1/1	Running	0	100m
retaild	app-postgres-5c476b86f4-r2pxz	1/1	Running	0	100m
retaild	app-xatcalc-9b4777954-697vx	1/1	Running	0	16m
retaild	app-redis-66977448b8-6z8d7	1/1	Running	0	100m
retaild	app-report-79d6b85797-jjt8b	1/1	Running	0	100m
retaild	app-sftp-75f76499c-644hj	1/1	Running	0	100m
retaild	app-trans-55bff9956b-kklnl	1/1	Running	0	100m
retaild	app-v4app-77f5886d68-72kfn	1/1	Running	0	100m
retaild	front-main-5c4484b968-5swbq	2/2	Running	0	100m
retaild	front-main-5c4484b968-tg2w8	2/2	Running	0	100m

```

devel@pat-m1:~$

```

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.

The image shows a simple authorization window titled 'Авторизация'. It contains two input fields: the first is labeled 'Логин' (Login) and the second is labeled 'Пароль' (Password). Below these fields is a single orange button with the text 'Войти' (Login/Enter).

Авторизация

Логин

Пароль

Войти

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