МойОфис
Частное Облако 3

Сервисно-ресурсная модель

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«МОЙОФИС ЧАСТНОЕ ОБЛАКО 3»

3.0

СЕРВИСНО-РЕСУРСНАЯ МОДЕЛЬ

Версия 1

На 32 листах

Дата публикации: 18.06.2024

Москва
2024

Все упомянутые в этом документе названия продуктов, логотипы, торговые марки и товарные знаки принадлежат их владельцам.

Товарные знаки «МойОфис» и «MyOffice» принадлежат ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ».

Ни при каких обстоятельствах нельзя истолковывать любое содержимое настоящего документа как прямое или косвенное предоставление лицензии или права на использование товарных знаков, логотипов или знаков обслуживания, приведенных в нем. Любое несанкционированное использование этих товарных знаков, логотипов или знаков обслуживания без письменного разрешения их правообладателя строго запрещено.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, ТЕРМИНОВ И ОПРЕДЕЛЕНИЙ

В настоящем документе применяют следующие сокращения с соответствующими расшифровками (см. Таблицу 1):

Таблица 1 — Сокращения и расшифровки

Сокращение, термин	Расшифровка и определение
ОС	Операционная система
СРМ	Сервисно-ресурсная модель. Логическая модель сервиса, описывающая состав и взаимосвязи компонентов
API	Application programming interface (англ.), программный интерфейс приложения — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.
CDN	Content Delivery Network или Content Distribution Network (англ.), сеть доставки (и дистрибуции) содержимого — распределенная сетевая инфраструктура, позволяющая оптимизировать доставку и дистрибуцию содержимого конечным пользователям в сети Интернет.
CO	Cloud Office (англ.), Облачный Офис, общее название продукта.
CVM	Conversion Manager (англ.), Сервис управления конвертированием файлов.
CU	Conversion Unit (англ.), экземпляр процесса конвертора различных форматов файлов.
DCM	Document Collaboration Manager (англ.), Сервис управления редактированием документов.
DU	Document Unit (англ.), экземпляр процесса редактирования и коллаборации документов.
FM	File Manager (англ.). Сервис файлового менеджера.
HA	High Availability (англ.), высокая доступность — характеристика технической системы, разработанной во избежание невыполненного обслуживания путем уменьшения или управления сбоями и минимизацией времени плановых простоев.
JOD	Java OpenDocument Converter (англ.). Сервис конвертирования документов стандарта OpenDocument.

Сокращение, термин	Расшифровка и определение
PGS	Pythagoras Storage (англ.). Компонент сервиса «МойОфис Частное Облако 3», представляет собой объектное хранилище.
WFE	Web Frontend (англ.). Общее название web приложений «МойОфис Частное Облако 3».
FM	File Manager (англ.). Сервис файлового менеджера.
HA	High Availability (англ.), высокая доступность — характеристика технической системы, разработанной во избежание невыполненного обслуживания путем уменьшения или управления сбоями и минимизацией времени плановых простоев.
JOD	Java OpenDocument Converter (англ.). Сервис конвертирования документов стандарта OpenDocument.
NM	Notification Manager (англ.). Сервис управления оповещениями.
SSO	Single Sign-On (англ.), технология единого входа — технология, при использовании которой пользователь переходит из одной системы в другую, не связанную с первой системой, без повторной аутентификации.
WFE	Web Frontend (англ.). Общее название web приложений «МойОфис Частное Облако 3».

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Введение

Сервисно-ресурсная модель (СРМ) — это логическая модель сервиса, описывающая состав и взаимосвязи компонентов (ресурсов), которые совместно обеспечивают предоставление сервиса. СРМ может быть представлена в виде иерархического графа, узлами которого являются компоненты, а ребрами — связи между ними. СРМ может быть использована для решения широкого круга задач, однако в первую очередь предназначена для мониторинга параметров качества сервиса, который может выполняться в рамках процесса управления доступностью. В данном документе графическое представление СРМ приведено в разделе «Диаграммы», а методики проверки значений параметров сервиса — в соответствующих разделах.

1.2 Градации и определение степени влияния Сервисов на Подсистемы

В данном документе используется несколько уровней влияния отдельного Сервиса на Подсистему и на Систему в целом. Уровни влияния определены в таблице 2.

Таблица 2 — Уровни влияния сервисов на Подсистему

Обозначение на диаграмме	Уровень влияния	Описание
	Critical	Отказ Сервиса приводит к отсутствию работоспособности Подсистемы
	Major	Отказ Сервиса приводит к отсутствию критичной функциональности при общей доступности Сервиса
	Minor	Отказ Сервиса приводит к отсутствию работоспособности редко используемой функциональности или к падению производительности
	Warning	Отказ Сервиса не приводит к потере функциональности (например, отказ одного из узлов кластера)

2 ДИАГРАММЫ

2.1 Сервис «МойОфис Частное Облако 3»

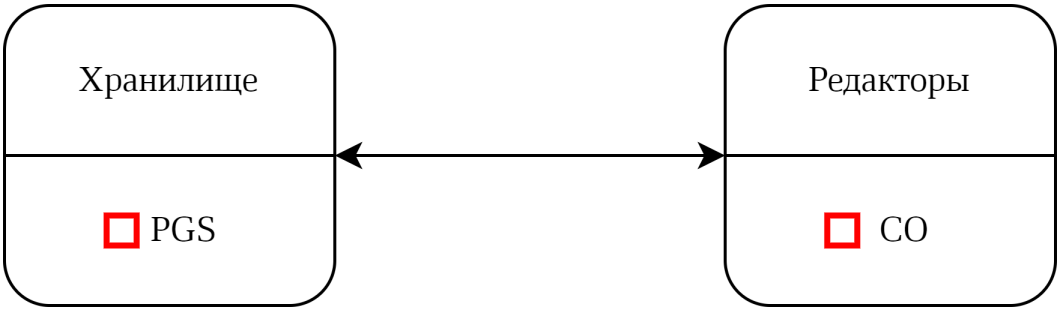


Рисунок 1 — Сервисно-ресурсная модель «МойОфис Частное Облако 3»

2.2 Сервисно-ресурсная модель Редакторы

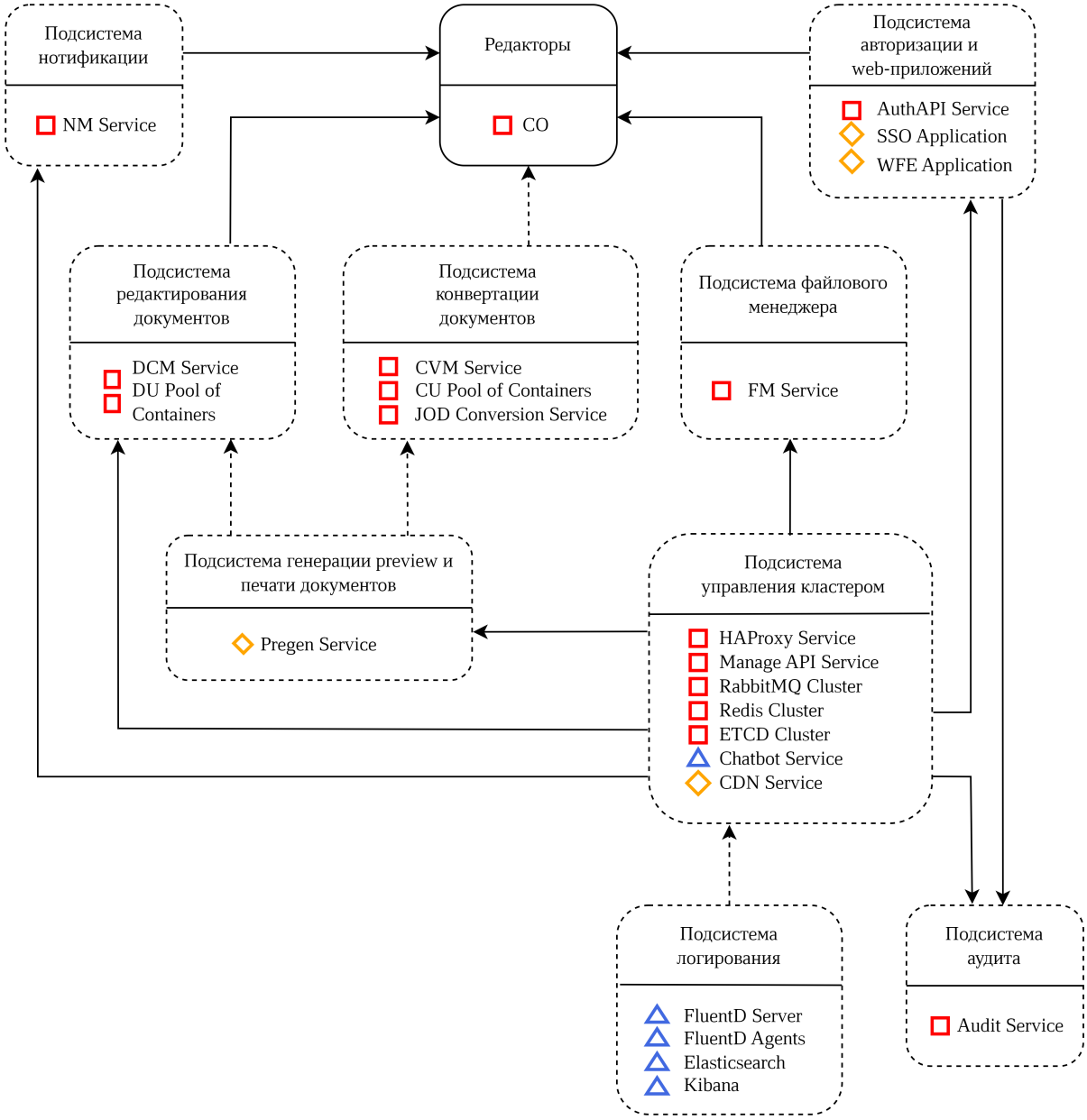


Рисунок 2 — Сервисно-ресурсная модель Редакторы (CO)

2.3 Сервисно-ресурсная модель Хранилище

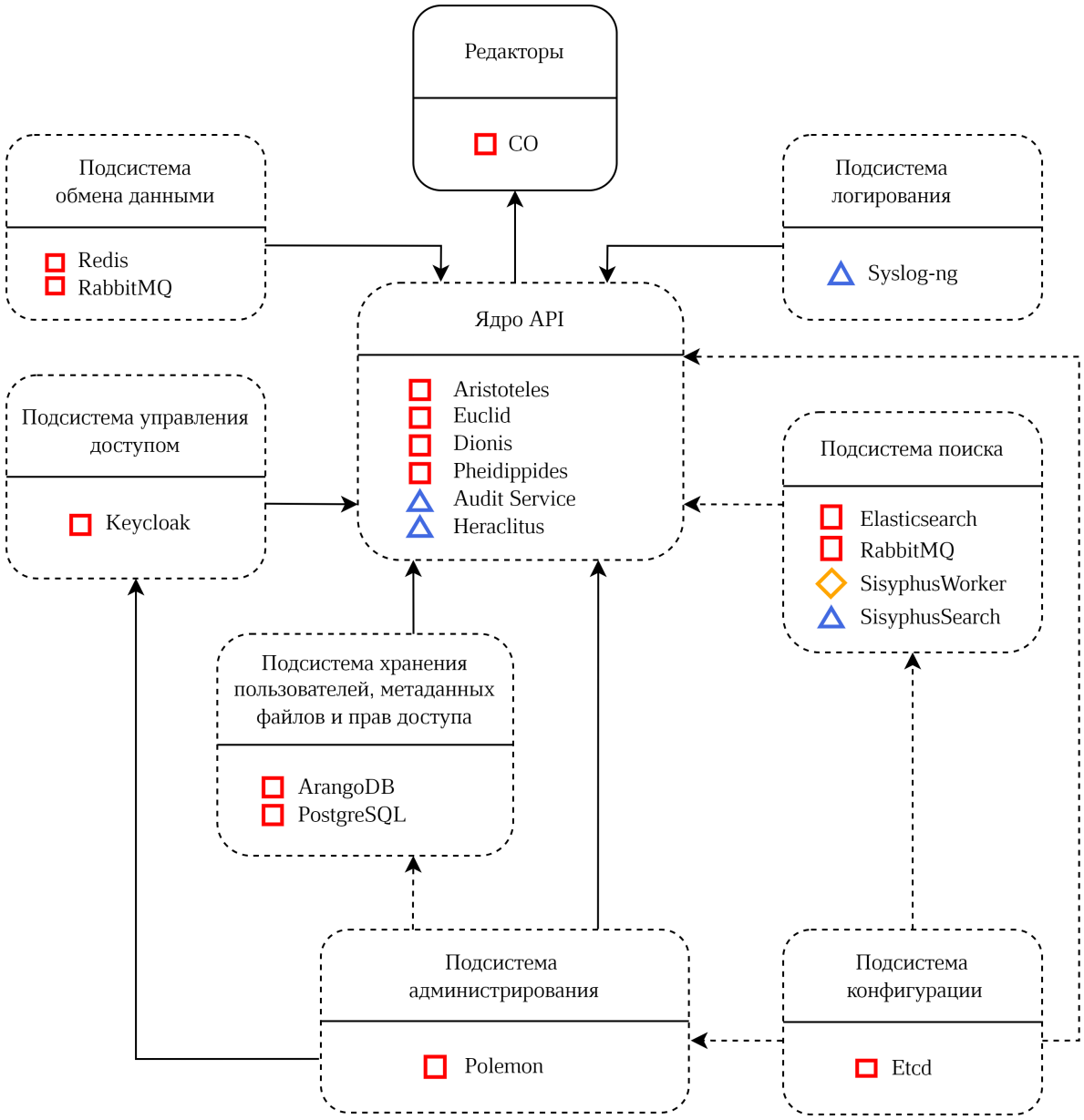


Рисунок 3 — Сервисно-ресурсная модель Хранилище (PGS)

2.4 Сервисно-ресурсная модель Почта

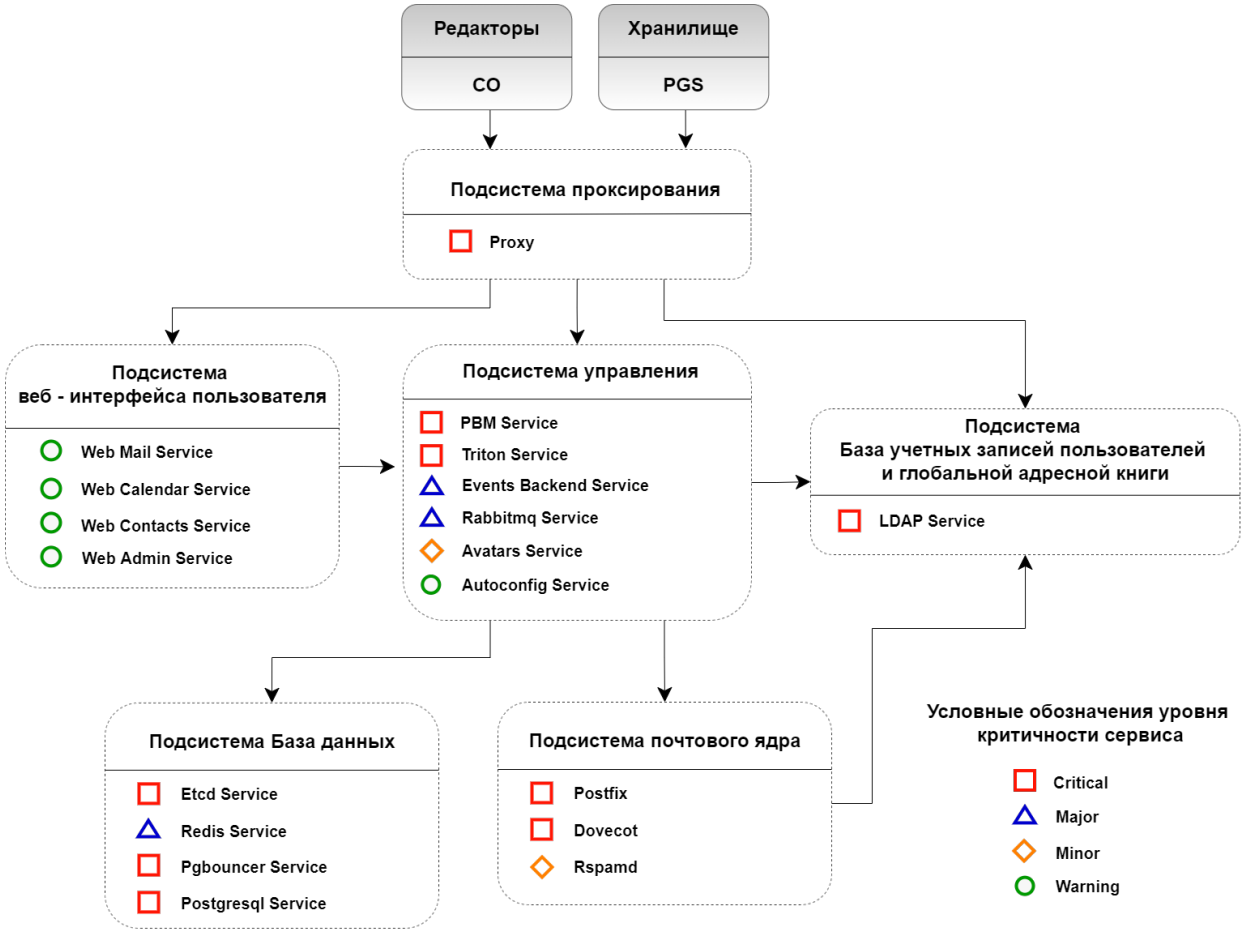


Рисунок 4 — Сервисно-ресурсная модель Почта (PSN)

3 РЕДАКТОРЫ

В описании методов контроля используются переменные вида `$PRIVATE_IPV4`, `$cvm_http_service_port` или `$dcm_check_path`, которые определяют параметры фактического окружения.

Для проверки хоста необходимо указать его IP-адрес в качестве значения переменной `$PRIVATE_IPV4`. Использование других переменных описано в методах контроля.

Если в описании переменной присутствует директория `/srv/docker`, стоит учитывать, что расположение файла может отличаться. При развертывании системы директория может быть изменена с помощью переменной `docker_volume_dir`.

Контроль функционирования сервисов выполняется запросами из командной строки, вызовами утилиты `curl`.

В примерах ниже используется авторизация в curl с логином `couser` и паролем `copass`. Реальные значения для текущей установки указаны в переменных `ansible openresty_api_username` и `openresty_api_password`.

3.1 Подсистема авторизации и веб-приложений

Параметры подсистемы авторизации и веб-приложений представлены в таблице 3.

Таблица 3 — Параметры подсистемы авторизации и веб-приложений

Контролируемый параметр	Корректное значение	Критичность
Auth API	OK	Critical
TLS GOST Proxy	running	Critical
SSO Application	running	Major
WFE Application	running	Major
WTE Application	running	Major

3.1.1 Методика контроля AuthAPI

Проверка статуса работы подсистемы `AuthAPI` осуществляется с помощью команды:

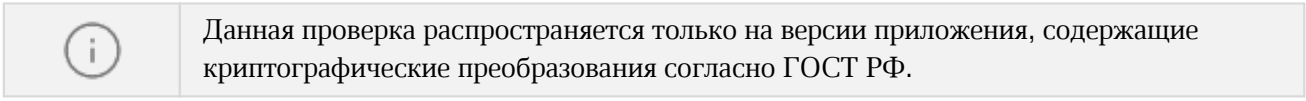
```
curl -s https://$PRIVATE_IPV4:$openresty_mng_port/api/manage/core/status \
--user couser:copass
```

Значение параметра `openresty_mng_port` приведено в приложении А.

Пример ответа:

```
"all":"OK"
```

3.1.2 Методика контроля TLS GOST Proxy



Данная проверка распространяется только на версии приложения, содержащие криптографические преобразования согласно ГОСТ РФ.

Для проверки статуса `TLS GOST` следует убедиться, что контейнер с приложением запущен. Для проверки в терминале от пользователя `root` необходимо выполнить команду:

```
curl -s unix-socket /var/run/docker.sock http://v1.38/containers/json | jq '.[] \
| select(.Names == ["/nginx-gost"]) | .State'
```

Пример ответа:

```
"running"
```

3.1.3 Методика контроля параметров SSO Application

Для проверки статуса `SSO Application` на всех хостах с ролью `co_lb_core_auth` следует убедиться, что контейнер с приложением запущен. Для проверки в терминале от пользователя `root` необходимо выполнить команду:

```
curl -s --unix-socket /var/run/docker.sock http:/v1.38/containers/json \
| jq '.[] | select(.Names == ["/web_sso"]) | .State'
```

Пример ответа:

```
"running"
```

3.1.4 Методика контроля параметров WFM Application

Для проверки статуса `WFM Application` на всех хостах с ролью `co_lb_core_auth` следует убедиться, что контейнер с приложением запущен. Для проверки в терминале от пользователя `root` необходимо выполнить команду:

```
curl -s --unix-socket /var/run/docker.sock http:/v1.38/containers/json \
| jq '.[] | select(.Names == ["/web_wfm"]) | .State'
```

Пример ответа:

```
"running"
```

3.1.5 Методика контроля параметров WTE Application

Для проверки статуса `WTE Application` на всех хостах с ролью `co_lb_core_auth` следует убедиться, что контейнер с приложением запущен. Для проверки в терминале от пользователя `root` необходимо выполнить команду:

```
curl -s --unix-socket /var/run/docker.sock http:/v1.38/containers/json \
| jq '.[] | select(.Names == ["/web_wte"]) | .State'
```

Пример ответа:

```
"running"
```

3.2 Подсистема файлового менеджера

Контролируемый параметр подсистемы файлового менеджера представлен в таблице 4.

Таблица 4 — Параметр подсистемы файлового менеджера

Контролируемый параметр	Корректное значение	Критичность
FM Service	UP	Critical

3.2.1 Методика контроля FM Service

Проверка статуса работы подсистемы `FM Service` осуществляется с помощью команды:

```
curl -s http://$PRIVATE_IPV4:$fm_service_port/$fm_check_path | jq '.status'
```

Значения параметров вызова приведены в приложении А.

Пример ответа:

```
"UP"
```

3.3 Подсистема конвертации документов

Контролируемые параметры подсистемы конвертации документов представлены в таблице 5.

Таблица 5 — Параметры подсистемы конвертации документов

Контролируемый параметр	Корректное значение	Критичность
CVM Service	UP	Critical
CU Pool of Containers	true	Critical
JOD Conversion Service	UP	Critical

3.3.1 Методика контроля CVM Service

Проверка статуса работы подсистемы `CVM Service` осуществляется с помощью команды:

```
curl -s http//$PRIVATE_IPV4:$cvm_http_service_port/$cvm_check_path | jq '.status'
```

Значения параметров вызова приведены в приложении А.

Пример ответа:

```
"UP"
```

3.3.2 Методика контроля DCM Service

Проверка статуса работы подсистемы `DCM Service` осуществляется с помощью команды:

```
curl –s http(s)://$PRIVATE_IPV4:$dcm_service_port/$dcm_check_path | jq '.status'
```

Значения параметров вызова приведены в приложении А.

Пример ответа:

```
"UP"
```

3.3.3 Методика контроля параметров CU Pool of Containers

Проверка статуса работы подсистемы `CU Containers on demand` осуществляется с помощью команды:

```
curl –s http://$PRIVATE_IPV4:$cvm_http_service_port/$cvm_check_path | jq
'.components.CVM.details.npsPool > 0'
```

Значения параметров вызова приведены в приложении А.

Пример ответа:

```
"true"
```

3.3.4 Методика контроля параметров JOD Conversion Service

Проверка статуса работы подсистемы `JOD Conversion Service` осуществляется с помощью команды:

```
curl –s http://$PRIVATE_IPV4:$jod_service_port/$jod_check_path | jq '.status'
```

Значения параметров вызова приведены в приложении А.

Пример ответа:

```
"UP"
```

3.4 Подсистема просмотра и печати документов

Контролируемый параметр подсистемы просмотра и печати документов представлен в таблице 6.

Таблица 6 — Параметр подсистемы просмотра и печати документов

Контролируемый параметр	Корректное значение	Критичность
Pregen Service	OK	Major

3.4.1 Методика контроля Pregen Service

Проверка статуса работы подсистемы `Pregen Service` осуществляется с помощью команды:

```
curl -sX HEAD -I http://$PRIVATE_IPV4:$pregen_service_port$pregen_check_path \
| grep HTTP | awk '{print $3 }'
```

Значения параметров вызова приведены в приложении А.

Пример ответа:

```
"OK"
```

3.5 Подсистема редактирования документов

Контролируемые параметры подсистемы редактирования документов представлены в таблице 7.

Таблица 7 — Параметры подсистемы редактирования документов

Контролируемый параметр	Корректное значение	Критичность
DCM Service	UP	Critical
DU Pool of Containers	true	Critical

3.5.1 Методика контроля параметров DU Pool of Containers

Проверка наличия контейнеров `DU` осуществляется с помощью команды:

```
curl -s http://$PRIVATE_IPV4:$dcm_service_port/manage/documents | jq 'length > 0'
```

Полученное значение должно соответствовать `"true"`.

3.6 Подсистема уведомлений

Контролируемый параметр подсистемы уведомлений представлен в таблице 8.

Таблица 8 — Параметр подсистемы уведомлений

Контролируемый параметр	Корректное значение	Критичность
NM Service	UP	Critical

3.6.1 Методика контроля параметров NM Service

Проверка статуса работы сервиса `NM Service` осуществляется с помощью команды:

```
curl -s http://$PRIVATE_IPV4:$nm_service_port/$nm_check_path | jq '.status'
```

Значения параметров вызова приведены в приложении А.

Пример ответа:

```
"UP"
```

3.7 Подсистема аудита

Контролируемый параметр подсистемы аудита представлен в таблице 9.

Таблица 9 — Параметр подсистемы аудита

Контролируемый параметр	Корректное значение	Критичность
Audit Service	UP	Critical

3.7.1 Методика контроля параметров Audit Service

Проверка статуса работы сервиса `Audit Service` осуществляется с помощью команды:

```
curl -s http://$PRIVATE_IPV4:$audit_service_port/$audit_check_path \
| jq '.status'
```

Значения параметров вызова приведены в приложении А.

Пример ответа:

```
"UP"
```

3.8 Подсистема управления кластером

Контролируемые параметры подсистемы управления кластером представлены в таблице 10.

Таблица 10 — Параметры подсистемы управления кластером

Контролируемый параметр	Корректное значение	Критичность
Redis Cluster	PONG	Critical
RabbitMQ Cluster	ok	Critical
ETCD Cluster	true	Critical
HA Proxy Service	200 OK Service ready	Critical
Manage API Service	OK	Critical
CDN Service	running	Major
Chatbot Service	OK	Minor

3.8.1 Методика контроля Redis Cluster

Для проверки сервиса `Redis Cluster` необходимо установить утилиту redis-cli.

Для rpm-based дистрибутивов утилита устанавливается с помощью команды:

```
yum install redis
```

Для deb-based дистрибутивов утилита устанавливается с помощью команды:

```
sudo apt install redis-tools
```

После установки утилиты следует выполнить проверку корректности работы сервиса `Redis Cluster` с помощью команды:

```
redis-cli -h $PRIVATE_IPV4 -p $redis_sentinel_port ping
```

Значение параметра `redis_sentinel_port` приведено в приложении А.

Пример ответа:

```
"PONG"
```

3.8.2 Методика контроля RabbitMQ Cluster

Проверка статуса работы сервиса `RabbitMQ Cluster` осуществляется с помощью команды:

```
curl -s http://root:$rabbitmq_password@$PRIVATE_IPV4:15672/api/aliveness-test/%2f
\
| jq '.status'
```

Значения параметра `rabbitmq_password` необходимо получить из переменной `rabbitmq_users.root.password`, используемой при установке.

Пример ответа:

```
"OK"
```

3.8.3 Методика контроля ETCD Cluster

Проверка статуса работы сервиса `ETCD Cluster` осуществляется с помощью команды:

```
curl -ks http://$etcd_browser_username:$etcd_browser_password@ \
$PRIVATE_IPV4:$etcd_browser_port/health | jq '.health'
```

Значения параметров вызова приведены в приложении А.

Пример ответа:

```
"true"
```

3.8.4 Методика контроля HAProxy Service

Проверка статуса работы сервиса `HAProxy` осуществляется с помощью команды:

```
curl -s http://$PRIVATE_IPV4:$haproxy_stats_port$api/manage/stats  \
| sed -e 's/<[^>]*>//g'
```

Значения параметров вызова приведены в приложении А.

Пример ответа:

```
200 OK
Service ready.
```

3.8.5 Методика контроля Manage API Service

Проверка статуса работы сервиса `Manage API Service` выполняется на серверах с ролью `co_lb_core_auth` с помощью команды:

```
curl -s http://$PRIVATE_IPV4:$openresty_mng_port/api/manage/config/version \
--user couser:copass
```

Значение параметра `openresty_mng_port` приведено в приложении А.

Ответ сервера с кодом `200 OK` может считаться признаком штатной работы сервиса.

3.8.6 Методика контроля параметров CDN Service

Для проверки статуса `CDN` на всех хостах с ролью `co_lb_core_auth` следует убедиться, что контейнер с приложением запущен. Для проверки в терминале от пользователя `root` необходимо выполнить команду:

```
curl -s --unix-socket /var/run/docker.sock http:/v1.38/containers/json \
| jq '.[] | select(.Names == ["/lsyncd"]) | .State'
```

Пример ответа:

```
"running"
```

3.8.7 Методика контроля параметров Chatbot Service

Проверка статуса работы сервиса `Chatbot` осуществляется с помощью команды:

```
curl -sX HEAD -I http://$PRIVATE_IPV4:$chatbot_service_port$chatbot_check_path \
| grep HTTP | awk '{print $3 }'
```

Значения параметров вызова приведены в приложении А.

Пример ответа:

```
OK
```


3.9 Подсистема логирования

Контролируемые параметры подсистемы логирования документов представлены в таблице 11.

Таблица 11 — Параметры подсистемы логирования

Контролируемый параметр	Корректное значение	Критичность
Fluent Server	OK	Minor
Fluent Agents	OK	Minor
Elasticsearch	green	Minor
Kibana	OK	Minor

3.9.1 Методика контроля параметров Fluent Server

Проверка статуса работы `Fluent Server` осуществляется на хостах с ролью `co_infra` с помощью команды:

```
docker exec -t fluentd_server curl -svo /dev/null \
http://127.0.0.1:$fluentd_server_port_monitor/api/plugins.json \
| grep '< HTTP' | awk '{print $4 }'
```

Значение параметра `fluentd_server_port_monitor` приведено в приложении А.

Пример корректного ответа:

```
OK
```

3.9.2 Методика контроля параметров Fluent Agents

Проверка статуса работы `Fluent Agents` осуществляется с помощью команды:

```
docker exec -t fluentd_agent curl -svo \
/dev/null http://127.0.0.1:$fluentd_agent_port_monitor/api/plugins.json \
| grep '< HTTP' | awk '{print $4 }'
```

Значение параметра `fluentd_agent_port_monitor` приведено в приложении А.

Пример корректного ответа:

```
OK
```

3.9.3 Методика контроля параметров Elasticsearch

Проверка статуса работы `Elasticsearch` осуществляется на хостах с ролью `co_infra` с помощью команды:

```
curl -sk \
"https://admin:$elasticsearch_admin_password@localhost:$elasticsearch_http_port/_
cluster/health" \
| jq '.status'
```

Значение параметров `elasticsearch_admin_password`, `elasticsearch_http_po` приведено в приложении А.

Ответ `green` будет признаком корректного функционирования Elasticsearch.

3.9.4 Методика контроля параметров Kibana

Проверка статуса работы `Kibana` осуществляется на хостах с ролью `co_infra` с помощью команды:

```
docker exec -t kibana curl -svko \
/dev/null https://$docker_bridge_ip:$kibana_http_port/app/login  \
| grep '< HTTP'  | awk '{print $4 }'
```

Значения параметров `docker_bridge_ip`, `kibana_http_port` приведены в приложении А.

Ответ `OK` будет признаком корректного функционирования Kibana.

4 ХРАНИЛИЩЕ

4.1 Описание сервисов без отказоустойчивости

Посмотреть все сервисы и их статус можно командой:

```
docker service ls --format 'table {{.Name}}\t{{.Replicas }}'
```

По умолчанию поддерживается установка с количеством реплик = 1, вывод должен соответствовать следующему примеру:

```
pgs-arangodb_arangodb            1/1
pgs-elasticsearch_elasticsearch  1/1
pgs-etcd_etcd                    1/1
pgs-keycloak_keycloak            1/1
pgs-nginx_nginx                  1/1
pgs-postgres_postgres            1/1
pgs-rabbitmq_rabbitmq            1/1
pgs-sisyphus_sisyphussearch      1/1
pgs-sisyphus_sisyphusworker      1/1
pgs_aristoteles                  1/1
pgs_dionis                       1/1
pgs_euclid                       1/1
pgs_heraclitus                   1/1
pgs_pheidippides                 1/1
pgs_polemon                      1/1
```

4.2 Подсистема Ядро API

Контролируемые параметры подсистемы ядра `API` представлены в таблице 12.

Таблица 12 — Параметры подсистемы Ядро API

Контролируемый параметр	Корректное значение	Критичность
Aristoteles	true	Critical
Euclid	true	Critical
Dionis	true	Critical
Pheidippides	true	Critical
Heraclitus	Running	Minor

4.2.1 Методика контроля Aristoteles

Проверка статуса работы `Aristoteles Service` осуществляется с помощью команды:

```
curl -k -XPOST  https://$PGS_URL/pgsapi/\?cmd\=api_version | jq .response.success
```

Пример ответа:

```
"true"
```

4.2.2 Методика контроля Euclid

Проверка статуса работы `Euclid Service` осуществляется с помощью команды:

```
curl -k -XPOST  https://$PGS_URL/adminapi/\?cmd\=api_version \
| jq .response.success
```

Пример ответа:

```
"true"
```

4.2.3 Методика контроля Dionis

Проверка статуса работы `Dionis Service` осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name\t{{.DesiredState}} pgs_dionis}}
```

Пример ответа:

```
"true"
```

4.2.4 Методика контроля Pheidippides

Проверка статуса работы `Pheidippides Service` осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name\t{{.DesiredState}} pgs_pheidippides}}
```

Пример ответа:

```
"true"
```

4.2.5 Методика контроля Heraclitus

Проверка статуса работы `Redis Service` осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' pgs_heraclitus
```

Пример ответа:

```
NAME                  DESIRED STATE
pgs_heraclitus.1      Running
```

4.3 Подсистема обмена данными

Контролируемые параметры подсистемы обмена данными представлены в таблице 13.

Таблица 13 — Параметры подсистемы обмена данными

Контролируемый параметр	Корректное значение	Критичность
Redis	PONG	Critical
RabbitMQ	Running	Critical

4.3.1 Методика контроля Redis

Проверка статуса работы сервиса `Redis` осуществляется с помощью команды:

```
docker exec -it $(docker ps -qf name=redis_redis-master) redis-cli \
--pass RjirbyfKfgf ping
```

Пример ответа:

```
PONG
```

4.3.2 Методика контроля RabbitMQ

Проверка статуса работы сервиса `RabbitMQ` осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' \
pgs-rabbitmq_rabbitmq
```

Пример ответа:

```
NAME                       DESIRED STATE
pgs-rabbitmq_rabbitmq.1    Running
```

4.4 Подсистема управления доступом

Контролируемый параметр подсистемы управления доступом представлен в таблице 14.

Таблица 14 — Параметр подсистемы управления

Контролируемый параметр	Корректное значение	Критичность
Keycloak	Running	Critical

4.4.1 Методика контроля Keycloak Service

Проверка статуса работы `Keycloak Service` осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' \
pgs-keycloak_keycloak
```

Пример ответа:

```
NAME                        DESIRED STATE
pgs-keycloak_keycloak.1     Running
```

4.5 Подсистема поиска

Контролируемые параметры подсистемы поиска представлены в таблице 15.

Таблица 15 — Параметры подсистемы поиска

Контролируемый параметр	Корректное значение	Критичность
Elasticsearch	green	Critical
RabbitMQ	Running	Critical
SisyphusSearch	Running	Critical
SisyphusWorker	Running	Major

4.5.1 Методика контроля Elasticsearch Service

Проверка статуса работы `Elasticsearch Service` осуществляется с помощью команды:

```
docker exec -it $(docker ps -qf name=elasticsearch_elasticsearch) \
curl localhost:9200/_cat/health | awk '{ print $4 }'
```

Пример ответа:

```
green
```

4.5.2 Методика контроля RabbitMQ Service

Проверка статуса работы `RabbitMQ Service` осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' \
pgs-rabbitmq_rabbitmq
```

Пример ответа:

```
NAME                      DESIRED STATE
pgs-rabbitmq_rabbitmq.1   Running
```

4.5.3 Методика контроля SisyphusSearch Service

Проверка статуса работы `SisyphusSearch Service` осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' \
pgs_sisyphussearch
```

Пример ответа:

```
NAME                    DESIRED STATE
pgs_sisyphussearch.1    Running
```

4.5.4 Методика контроля SisyphusWorker Service

Проверка статуса работы `SisyphusWorker Service` осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' \
pgs_sisyphusworker
```

Пример ответа:

```
NAME                     DESIRED STATE
pgs_sisyphusworker.1     Running
```

4.6 Подсистема хранения пользователей, метаданных файлов и прав доступа

Контролируемые параметры подсистемы хранения пользователей, метаданных файлов и прав доступа представлены в таблице 16.

Таблица 16 — Параметры подсистемы хранения метаданных файлов и прав доступа

Контролируемый параметр	Корректное значение	Критичность
ArangoDB	Running	Critical
PostgreSQL	accepting connections	Critical

4.6.1 Методика контроля ArangoDB Service

Проверка статуса работы `ArangoDB Service` осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' \
pgs-arangodb_arangodb
```

Пример ответа:

```
NAME                        DESIRED STATE
pgs-arangodb_arangodb.1     Running
```

4.6.2 Методика контроля Postgres Service

Проверка статуса работы `Postgres Service` осуществляется с помощью команды:

```
docker exec -it $(docker ps -qf name=postgres) pg_isready
```

Пример ответа:

```
/var/run/postgresql:5432 - accepting connections
```

4.7 Подсистема администрирования

Контролируемые параметры подсистемы администрирования представлены в таблице 17.

Таблица 17 — Параметры подсистемы администрирования

Контролируемый параметр	Корректное значение	Критичность
Euclid	Running	Critical
Polemon	Running	Major

4.7.1 Методика контроля Euclid Service

Проверка статуса работы `Euclid Service` осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' \
pgs_euclid
```

Пример ответа:

```
NAME           DESIRED STATE
pgs_euclid.1   Running
```

4.7.2 Методика контроля Polemon Service

Проверка статуса работы `Polemon Service` осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' \
pgs_polemon
```

Пример ответа:

```
NAME            DESIRED STATE
pgs_polemon.1   Running
```

4.8 Подсистема конфигурирования

Контролируемый параметр подсистемы конфигурирования представлен в таблице 18.

Таблица 18 — Параметр подсистемы конфигурирования

Контролируемый параметр	Корректное значение	Критичность
etcd	cluster is healthy	Critical

4.8.1 Методика контроля ETCD Service

Проверка статуса работы `ETCD Service` осуществляется с помощью команды:

```
docker exec -it $(docker ps -qf name=pgs-etcd_etcd) \
etcdctl cluster-health
```

Пример ответа:

```
member ade526d28b1f92f7 is healthy: got healthy result from http://etcd1:2379
member bd388e7810915853 is healthy: got healthy result from http://etcd3:2379
member d282ac2ce600c1ce is healthy: got healthy result from http://etcd2:2379
cluster is healthy
```

4.9 Подсистема хранения файлов

Контролируемые параметры подсистемы хранения файлов представлены в таблице 19.

Таблица 19 — Параметры подсистемы хранения файлов

Контролируемый параметр	Корректное значение	Критичность
MinIO	Running	Critical
GlusterFS	Active (running)	Critical

4.9.1 Методика контроля MinIO

Для контроля необходимы параметры, заданные при установке Хранилища (PGS):

– `ENV` — окружение установки (при наличии);

– `DEFAULT_DOMAIN` — домен установки;

– `S3.secret_key`;

– `S3.access_key`.

Описание переменных представлено в документе «"МойОфис Частное облако 3" Система хранения данных. Руководство по установке».

4.9.1.1 Методика контроля старта сервиса

Проверка старта сервиса осуществляется с помощью команды:

```
docker service ps --format 'table {{.Name}}\t{{.DesiredState}}' \
pgs-minio_minio$i
```

где `$i` — номер инстанса (например: 1,2,3).

Пример ответа:

```
NAME                    DESIRED STATE
pgs-minio_minio2.1      Running
```

4.9.1.2 Методика контроля работы сервиса

Для проверки статуса сервиса необходимо заранее установить дополнительное ПО Minio Client. Установка осуществляется с помощью команды:

```
docker run -it --entrypoint=/bin/sh minio/mc
```

После установки необходимо создать `alias` подключения с помощью команды:

```
mc alias set minio http://pgs-{{ ENV }}.{{ DEFAULT_DOMAIN}}:9000 \
{{ S3.access_key  }}  {{ S3.secret_key }}
```

После создания `alias` следует выполнить проверку статуса сервиса с помощью команды:

```
mc admin info minio
```

Пример ответа:

```
     minio1:9000
    Uptime: 2 days
    Version: 2021-02-01T22:56:52Z
    Network: 3/3 OK
    Drives: 2/2 OK

    minio3:9000
    Uptime: 2 days
    Version: 2021-02-01T22:56:52Z
    Network: 3/3 OK
    Drives: 2/2 OK

    minio2:9000
```



```
     Uptime: 2 days
     Version: 2021-02-01T22:56:52Z
     Network: 3/3 OK
     Drives: 2/2 OK
```

4.9.2 Методика контроля GlusterFS

4.9.2.1 Методика контроля работы сервиса

Проверка статуса работы сервиса осуществляется с помощью команды:

```
Systemctl status glusterd
```

Пример корректного ответа:

```
glusterd.service -GlusterFS, a clustered file-system server
Loaded: loaded (/usr/lib/systemd/system/glusterd.service; enabled; vendor preset:
disabled)
Active: active (running) since Tue 2021-01-26 22:33:19 MSK; 4 weeks 0 days ago
Docs: man:glusterd(8)
Main PID: 23706 (glusterd)
Tasks: 55
Memory: 2.5G
CGroup: /system.slice/glusterd.service
```

4.9.2.2 Методика контроля работы Gluster Volume

Проверка статуса осуществляется с помощью команды:

```
gluster volume status pgs-files-volume
```

Пример ответа:

```
Status of volume: pgs-files-volume
Gluster process                       TCP Port RDMA Port  Online    Pid
------------------------------------------------------------------------------
Brick 10.160.100.170:/data/glusterfs/ pgs-fil
t                                     49152     0          Y       10072
Brick 10.160.100.171:/data/glusterfs/ pgs-fil
t                                     49152     0          Y       23820
Self-heal Daemon on localhost         N/A       N/A        Y       23853
Self-heal Daemon on ldap.domain.ru    N/A       N/A        Y       10093
Task Status of Volume pgs-files-volume
```

Колонка `Online` во всех строках должна иметь значение `Y`. Количество `Brick(s)` приведено для примера и может отличаться (зависит от параметров установки).

Приложение А

Параметры сервисов редакторов и хранилища

Сервис	Параметр	Значение
AUDIT	$audit_service_port/$audit_check_path	9900/manage/health
KIBANA	$kibana_http_port	5601
ELASTIC SEARCH	$elasticsearch_http_port	9200,9300
OPENRESTY	$openresty_mng_port	8888
ETCD BROWSER	$etcd_browser_port	8001
HAPROXY	$haproxy_stats_port	20001-20002, 20004-20007
FLUENTD	$fluentd_server_port_monitor	23100, 23200
REDIS	$redis_sentinel_port	6379
PREGEN	$pregen_service_port$pregen_check_path	8002/health
CHATBOT	$chatbot_service_port$chatbot_check_path	8004/health
FM	$fm_service_port/$fm_check_path	9091/manage/health
NM	$nm_http_service_port/$nm_check_path	9092/manage/health
CVM	$cvm_http_service_port/$cvm_check_path	9094/manage/health
DCM	$dcm_service_port/$dcm_check_path	9095/manage/health
JOD	$jod_service_port/$jod_check_path	9096/manage/health