

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

СИСТЕМА РЕДАКТИРОВАНИЯ И СОВМЕСТНОЙ РАБОТЫ

СИСТЕМА ХРАНЕНИЯ ДАННЫХ

3.2

СЕРВИСНО-РЕСУРСНАЯ МОДЕЛЬ

Версия 1

На 33 листах

Дата публикации: 17.12.2024

**Москва
2024**

МойОфис

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

1	Общие сведения	7
1.1	Назначение	7
1.2	О компонентах	7
1.3	Перечень технической документации	8
1.4	Градации и определение степени влияния Сервисов на Подсистемы	9
2	Диаграммы	10
2.1	Общая структура взаимодействия	10
2.2	Сервисно-ресурсная модель Редакторы	11
2.3	Сервисно-ресурсная модель Хранилище	12
3	Редакторы	13
3.1	Подсистема авторизации и веб-приложений	13
3.1.1	Методика контроля AuthAPI	13
3.1.2	Методика контроля TLS GOST Proxy	14
3.1.3	Методика контроля параметров SSO Application	14
3.1.4	Методика контроля параметров WFM Application	14
3.1.5	Методика контроля параметров WTE Application	14
3.2	Подсистема файлового менеджера	15
3.2.1	Методика контроля FM Service	15
3.3	Подсистема конвертации документов	15
3.3.1	Методика контроля CVM Service	15
3.3.2	Методика контроля DCM Service	16
3.3.3	Методика контроля параметров CU Pool of Containers	16
3.3.4	Методика контроля параметров JOD Conversion Service	16
3.4	Подсистема просмотра и печати документов	16
3.4.1	Методика контроля Pregen Service	17
3.5	Подсистема редактирования документов	17
3.5.1	Методика контроля параметров DU Pool of Containers	17
3.6	Подсистема уведомлений	17
3.6.1	Методика контроля параметров NM Service	17

3.7 Подсистема аудита	18
3.7.1 Методика контроля параметров Audit Service	18
3.8 Подсистема управления кластером	18
3.8.1 Методика контроля Redis Cluster	19
3.8.2 Методика контроля RabbitMQ Cluster	19
3.8.3 Методика контроля ETCD Cluster	19
3.8.4 Методика контроля HAProxy Service	20
3.8.5 Методика контроля Manage API Service	20
3.8.6 Методика контроля параметров CDN Service	20
3.8.7 Методика контроля параметров Chatbot Service	20
3.9 Подсистема логирования	21
3.9.1 Методика контроля параметров Fluent Server	21
3.9.2 Методика контроля параметров Fluent Agents	21
3.9.3 Методика контроля параметров Elasticsearch	21
3.9.4 Методика контроля параметров Kibana	22
4 Хранилище	23
4.1 Описание сервисов без отказоустойчивости	23
4.2 Описание сервисов с поддержкой отказоустойчивости	23
4.3 Подсистема Ядро API	24
4.3.1 Методика контроля Aristoteles	24
4.3.2 Методика контроля Euclid	24
4.3.3 Методика контроля Epicure	24
4.3.4 Методика контроля Dionis	25
4.3.5 Методика контроля Pheidippides	25
4.3.6 Методика контроля Heraclitus	25
4.4 Подсистема обмена данными	25
4.4.1 Методика контроля Redis	25
4.4.2 Методика контроля RabbitMQ	26
4.5 Подсистема управления доступом	26
4.5.1 Методика контроля Keycloak Service	26
4.6 Подсистема поиска	27

4.6.1	Методика контроля Elasticsearch Service	27
4.6.2	Методика контроля RabbitMQ Service	27
4.6.3	Методика контроля SisyphusSearch Service	27
4.6.4	Методика контроля SisyphusWorker Service	28
4.7	Подсистема хранения пользователей, метаданных файлов и прав доступа	28
4.7.1	Методика контроля ArangoDB Service	28
4.7.2	Методика контроля Postgres Service	28
4.8	Подсистема администрирования	29
4.8.1	Методика контроля Polemon Service	29
4.9	Подсистема конфигурирования	29
4.9.1	Методика контроля ETCD Service	29
4.10	Подсистема хранения файлов	30
4.10.1	Методика контроля MinIO	30
4.10.1.1	Методика контроля старта сервиса	30
4.10.1.2	Методика контроля работы сервиса	30
4.10.2	Методика контроля GlusterFS	31
4.10.2.1	Методика контроля работы сервиса	31
4.10.2.2	Методика контроля работы Gluster Volume	31
Приложение. Параметры сервисов редакторов и хранилища		33

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, ТЕРМИНОВ И ОПРЕДЕЛЕНИЙ

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

Таблица 1 — Сокращения и обозначения

Сокращение, термин	Расшифровка и определение
API	Application Programming Interface, интерфейс программирования приложений
Auth SSO	Single Sign-On, подсистема единого входа (аутентификации и авторизации)
CA	Certificate Authority, удостоверяющий центр для подтверждения подлинности ключей шифрования
CDN	Content Delivery Network, сеть доставки содержимого (в данном случае одна из ролей Auth SSO сервера)
CO	Система редактирования и совместной работы
CU	Converter Unit, сервис конвертирования разных форматов файлов
DU	Document Unit, синоним DCS
FCM	Firebase Cloud Messaging, сервис уведомлений мобильных приложений Google, ранее назывался GCM
FQDN	Fully Qualified Domain Name, полностью определенное имя домена
GCM	Google Cloud Messaging, сервис нотификаций мобильных приложений Google, заменен сервисом FCM
PGS	Система хранения данных
PSN	Почтовый сервер
ОС	Операционная система
ПК	Персональный компьютер
ПО	Программное обеспечение

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Назначение

Сервисно-ресурсная модель (SRM) — это логическая модель сервиса, описывающая состав и взаимосвязи компонентов (ресурсов), которые совместно обеспечивают предоставление сервиса. SRM может быть представлена в виде иерархического графа, узлами которого являются компоненты, а ребрами — связи между ними. SRM 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 SRM приведено в разделе «Диаграммы», а методики проверки значений параметров сервиса — в соответствующих разделах.

1.2 О компонентах

Система хранения данных — компонент, предназначенный для безопасного хранения корпоративных файлов и обеспечения возможностей авторизации, аутентификации и разграничения прав доступа пользователей.

Система редактирования и совместной работы — компонент, предназначенный для индивидуального и совместного редактирования текстовых и табличных документов, а также просмотра и демонстрации презентаций.

Почтовая система — компонент, предназначенный для обработки входящих и исходящих сообщений электронной почты, совместной работы с календарями и задачами, а также ведения адресных книг.

Представленные компоненты входят в состав следующих продуктов:

- «МойОфис Частное Облако 3»;
- «МойОфис Профессиональный 2»;
- «МойОфис Профессиональный 3»;
- «МойОфис Схема»;
- Squadus PRO.

Подробное описание возможностей продукта приведено в документе «Функциональные возможности».

1.3 Перечень технической документации

С помощью технической документации, представленной в таблице 2, осуществляется развертывание серверной части, настройка и администрирование продукта.

Комплект документации распространяется на компоненты продукта:

- Система редактирования и совместной работы (CO);
- Система хранения данных (PGS).





Таблица 2 — Перечень технической документации

Наименование документа	Используемые компоненты	Содержание документа
«Системные требования»	CO, PGS	Системные и программные требования к продукту
«Архитектура»	CO, PGS	Описание архитектуры продукта для выбора типа установки и выделения ресурсов для серверов
«Система редактирования и совместной работы. Руководство по установке»	CO	Порядок установки системы редактирования и совместной работы
«Система хранения данных. Руководство по установке»	PGS	Порядок установки системы хранения данных
«Руководство по настройке»	CO, PGS	Настройка серверов продукта после установки и в ходе эксплуатации системы, а также процессов мониторинга и логирования
«Руководство по администрированию»	CO, PGS	Функции управления тенантом в ходе эксплуатации системы
«Руководство по резервному копированию»	PGS	Порядок резервного копирования баз данных, расположенных в системе хранения данных
«Сервисно-ресурсная модель»	CO, PGS	Логическая модель сервиса, описывающая состав и взаимосвязи компонентов (ресурсов), которые совместно обеспечивают предоставление сервиса
«Руководство по мониторингу»	CO, PGS	Функции отображения текущего состояния системы и отдельных сервисов
«Руководство по работе с API»	CO, PGS	Набор методов для автоматизированного управления пользователями, группами, общими папками, доменами и тенантами

1.4 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 3.

Таблица 3 — 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 Общая структура взаимодействия

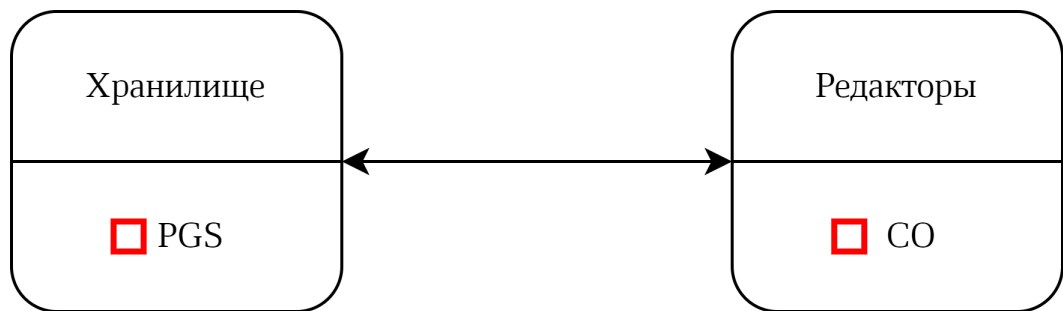


Рисунок 1 — Сервисно-ресурсная модель продукта

2.2 Сервисно-ресурсная модель Редакторы

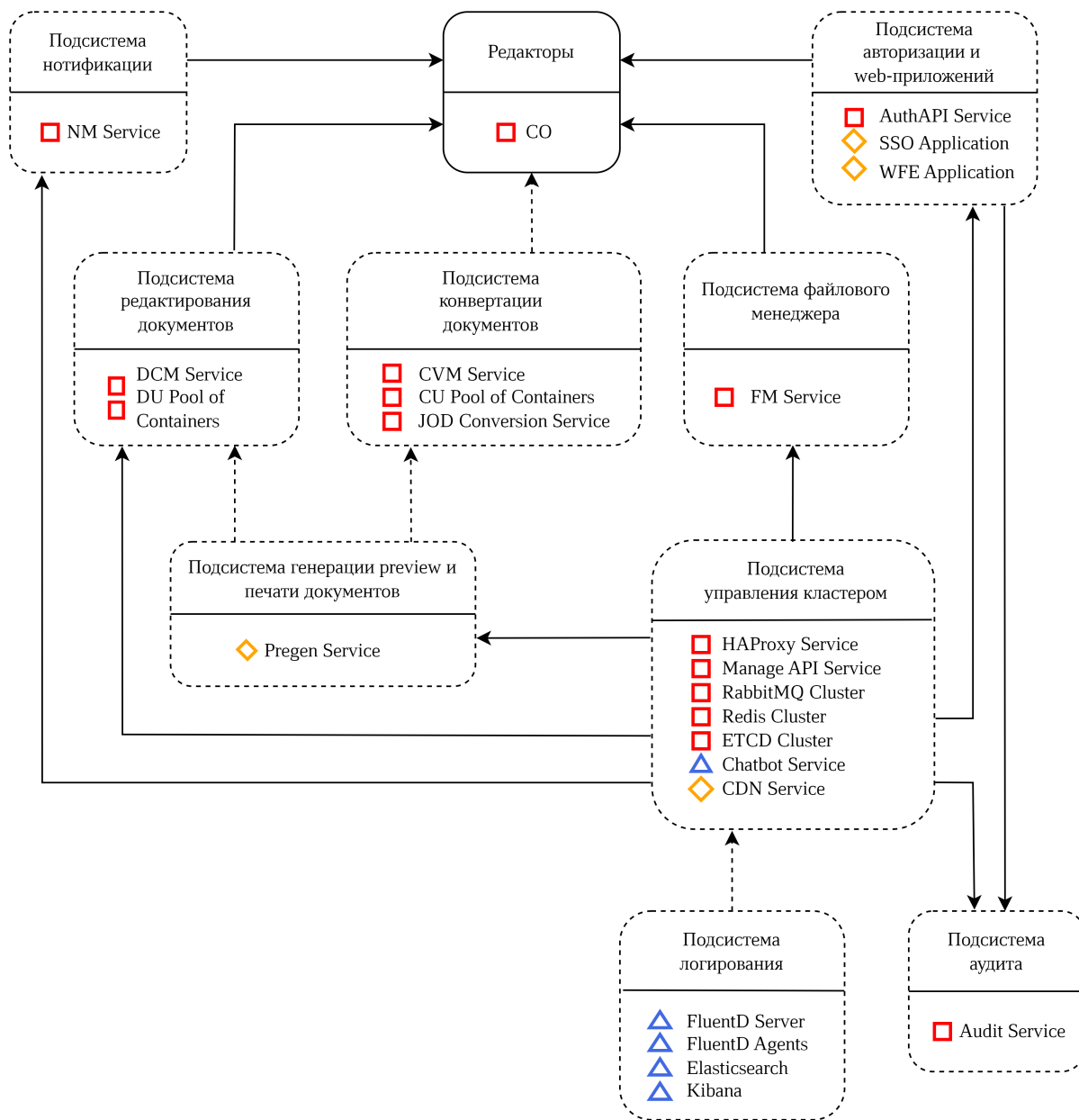


Рисунок 2 — Сервисно-ресурсная модель Редакторы (CO)

2.3 Сервисно-ресурсная модель Хранилище

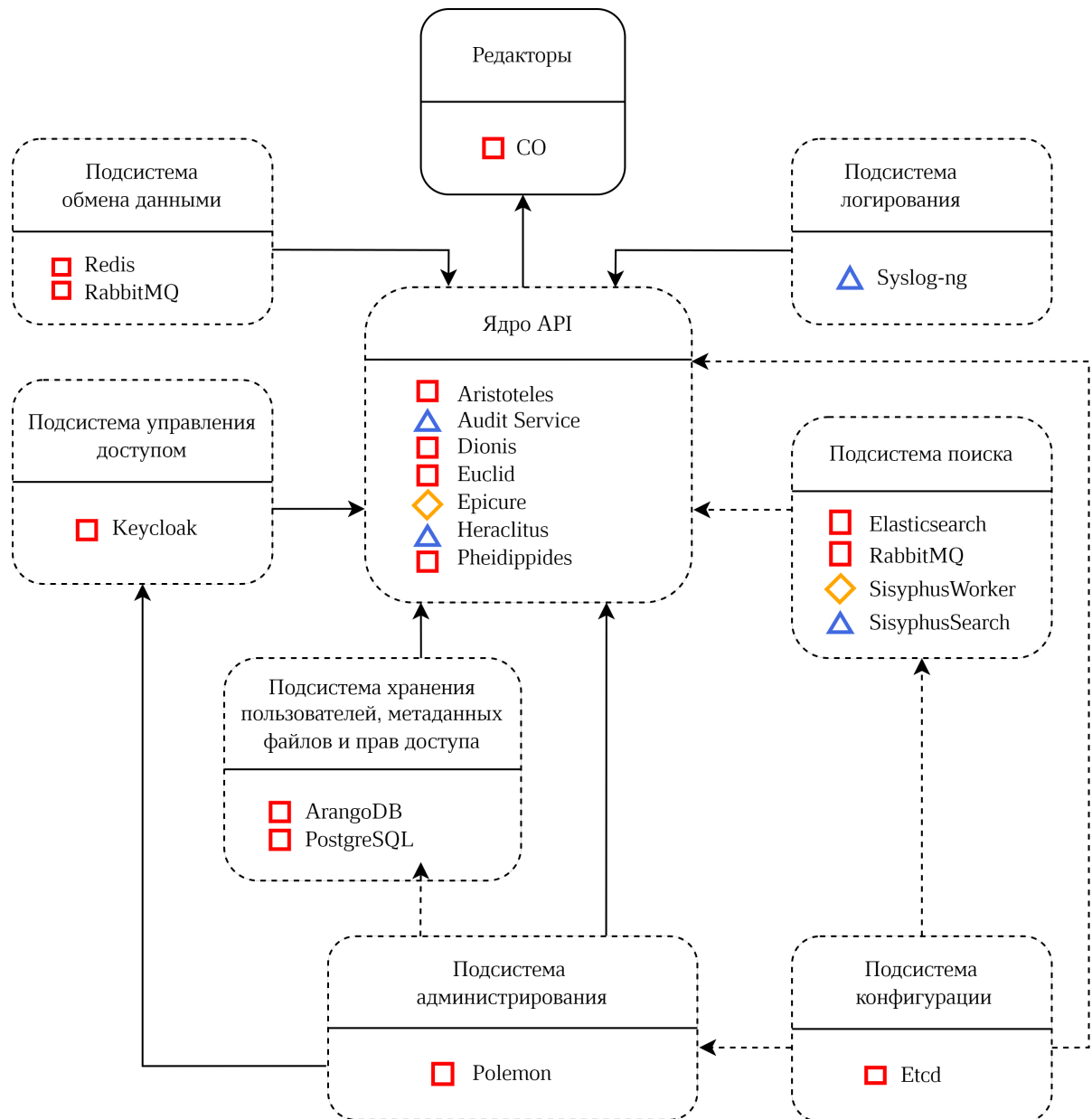


Рисунок 3 — Сервисно-ресурсная модель Хранилище (PGS)

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 `ansible_openresty_api_password`.

3.1 Подсистема авторизации и веб-приложений

Параметры подсистемы авторизации и веб-приложений представлены в таблице 4.

Таблица 4 — 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 5.

Таблица 5 — 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.

Пример ответа:

```
"UP"
```

3.3 Подсистема конвертации документов

Контролируемые параметры подсистемы конвертации документов представлены в таблице 6.

Таблица 6 — 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 7.

Таблица 7 — Параметр подсистемы просмотра и печати документов

Контролируемый параметр	Корректное значение	Критичность
Pregen Service	OK	Major

3.4.1 Методика контроля Pregon Service

Проверка статуса работы подсистемы Pregon 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 8.

Таблица 8 — 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 9.

Таблица 9 — 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 10.

Таблица 10 — 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 11.

Таблица 11 — 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 12.

Таблица 12 — 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.

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-redis_redis                 1/1
pgs-sisyphus_sisyphussearch     1/1
pgs-sisyphus_sisyphusworker     1/1
pgs_aristoteles                 1/1
pgs_dionis                      1/1
pgs_epicure                     1/1
pgs_euclid                      1/1
pgs_heraclitus                  1/1
pgs_pheidippides                1/1
pgs_polemon                     1/1
```

4.2 Описание сервисов с поддержкой отказоустойчивости

Посмотреть все сервисы и их статус можно командой:

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

В кластерной установке количество реплик соответствует необходимому уровню отказоустойчивости сервиса:

```
pgs-arangodb_agent              3/3
pgs-arangodb_coordinator        2/2
pgs-arangodb_dbserver           2/2
pgs-elasticsearch_elasticsearch 3/3
pgs-elasticsearch_es-node1     1/1
pgs-elasticsearch_es-node2     1/1
pgs-elasticsearch_es-node3     1/1
pgs-etcd_etcd                   3/3
pgs-etcd_etcd1                  1/1
pgs-etcd_etcd2                  1/1
pgs-etcd_etcd3                  1/1
pgs-keycloak_keycloak           3/3
pgs-nginx_nginx                 2/2
pgs-postgres_haproxy            2/2
pgs-postgres_postgres0          1/1
pgs-postgres_postgres1          1/1
pgs-rabbitmq_rabbitmq           3/3
pgs-redis_redis-master          1/1
pgs-redis_redis-sentinel        3/3
pgs-redis_redis-slave           3/3
```

pgs-sisyphus_sisyphussearch	3/3
pgs-sisyphus_sisyphusworker	2/2
pgs_aristoteles	2/2
pgs_dionis	1/1
pgs_epicure	2/2
pgs_euclid	2/2
pgs_heraclitus	1/1
pgs_pheidippides	1/1
pgs_polemon	2/2

4.3 Подсистема Ядро API

Контролируемые параметры подсистемы ядра API представлены в таблице 13.

Таблица 13 — Параметры подсистемы Ядро API

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

4.3.1 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"
```

4.3.2 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"
```

4.3.3 Методика контроля Epicure

Проверка статуса работы Epicure осуществляется с помощью команды:

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

Пример ответа:

```
"running"
```


4.3.4 Методика контроля Dionis

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

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

Пример ответа:

```
"true"
```

4.3.5 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"
```

4.3.6 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
```

4.4 Подсистема обмена данными

Контролируемые параметры подсистемы обмена данными представлены в таблице 14.

Таблица 14 — Параметры подсистемы обмена данными

Контролируемый параметр	Корректное значение	Критичность
Redis	PONG	Critical
RabbitMQ	Running	Critical

4.4.1 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
```

4.4.2 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
```

4.5 Подсистема управления доступом

Контролируемый параметр подсистемы управления доступом представлен в таблице 15.

Таблица 15 — Параметр подсистемы управления

Контролируемый параметр	Корректное значение	Критичность
Keycloak	Running	Critical

4.5.1 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
```

4.6 Подсистема поиска

Контролируемые параметры подсистемы поиска представлены в таблице 16.

Таблица 16 — Параметры подсистемы поиска

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

4.6.1 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
```

4.6.2 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
```

4.6.3 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
```

4.6.4 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
```

4.7 Подсистема хранения пользователей, метаданных файлов и прав доступа

Контролируемые параметры подсистемы хранения пользователей, метаданных файлов и прав доступа представлены в таблице 17.

Таблица 17 — Параметры подсистемы хранения метаданных файлов и прав доступа

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

4.7.1 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
```

4.7.2 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
```

4.8 Подсистема администрирования

Контролируемые параметры подсистемы администрирования представлены в таблице 18.

Таблица 18 — Параметры подсистемы администрирования

Контролируемый параметр	Корректное значение	Критичность
Euclid	Running	Critical
Polemon	Running	Major

4.8.1 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
```

4.9 Подсистема конфигурирования

Контролируемый параметр подсистемы конфигурирования представлен в таблице 19.

Таблица 19 — Параметр подсистемы конфигурирования

Контролируемый параметр	Корректное значение	Критичность
etcd	cluster is healthy	Critical

4.9.1 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 d282ac2ce600clce is healthy: got healthy result from http://etcd2:2379
cluster is healthy
```

4.10 Подсистема хранения файлов

Контролируемые параметры подсистемы хранения файлов представлены в таблице 20.

Таблица 20 — Параметры подсистемы хранения файлов

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

4.10.1 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 «Система хранения данных. Руководство по установке».

4.10.1.1 Методика контроля старта сервиса

Проверка старта сервиса осуществляется на сервере с ролью storage с помощью команды:

```
docker ps --format '{{.Names}}\t{{.Status}}' -f name=minio
```

Пример ответа:

```
minio Up 12 days
```

4.10.1.2 Методика контроля работы сервиса

Для проверки статуса необходимо использовать утилиту mc, которая установлена в контейнере MinIO.

Порядок действий:

1. Подключиться к контейнеру MinIO с помощью команды:

```
docker exec -it minio bash
```

2. Создать alias с помощью команды:

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

3. 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
```

4.10.2 Методика контроля GlusterFS

4.10.2.1 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
```

4.10.2.2 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
```

МойОфис

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

ПРИЛОЖЕНИЕ

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