**ИНСТРУКЦИЯ ПО УСТАНОВКЕ**

**RT.STREAMING**

2025

# 1 Введение

**Примечание:** Непосредственно операции по подготовке к созданию и созданию кластера описаны в документации к RT.ClusterManager. Данные о порядке создания компонент и по распределению сервисов компонент по нодам, описанные  в данном документе, относятся к операциям описанным в [п. 8.4.1 и п. 8.4.2 документа “RT.ClusterManager. Руководство администратора”](https://docs.data.rt.ru/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F/RT_ClusterManager/Vers_2/RT_ClusterManager_Admin_Guide_ver2#:~:text=8.4.1%20%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BE%D0%B2).

## 1.1 Конфигурация кластера из 3 нод

Предлагаемая конфигурация кластеров, создаваемых на основании плагина «RT.StreamingNiFi», состоит из **3 нод**, принадлежащих одному провайдеру, и отдельного сервера для установки RT.ClusterManager, который не входит в состав кластера:

m-1,

m-2,

m-3.

В данном описании ноды – сервера (хосты) на базе операционных систем семейства Linux, которые будут включены в кластер.

**Важно: При развёртывании стандартного кластера необходимо строго придерживаться порядка установки компонентов, указанного в таблицах ниже. В иных случаях существует возможность частично откорректировать распределение.**

**Примечание:** В RT.ClusterManager версии 2.Х-ХХ был изменен дизайн и терминология, поменялись местами термины “Компонент” и “Сервис”. Так в RT.ClusterManager версии 1.Х-ХХ: Zookeeper, HDFS … были сервисами, и входящие в них элементы программного обеспечения которые привязывались к нодам  назывались компонентами, а в RT.ClusterManager версии 2.Х-ХХ: Zookeeper, HDFS … стали компонентами, и входящие в них элементы программного обеспечения которые привязываются к нодам  называются сервисами. В данном документе используется терминология RT.ClusterManager версии 2.Х-ХХ.

# ****2 Распределение сервисов по нодам - плагин «****RT.StreamingNiFi****»****

В следующей таблице представлен порядок распределения по нодам компонентов и их сервисов на основании плагина «RT.StreamingNiFi»:

**Важно: При установке, придерживайтесь порядка компонент, указанных в таблицах.**

## ****2.1 Конфигурация из 3 нод****

| **№** | **Компонент** | **Назначение** | **Распределение сервисов по нодам** |
| --- | --- | --- | --- |
| 1 | **zookeeper** | Программная служба для координации распределенных систем.  Проверка статуса - см. п. 3.1 | **server**- m-1, m-2, m-3 |
| 2 | **nifi** | Автоматизация обмена данными между программными системам.  Проверка статус - см. п. 3.2 | **nifi**- m-1 (или любое количество) |

# 3 Проверка статуса компонентов

## 3.1 Проверка статуса zookeeper

Для проверки статуса zookeeper на каждом хосте, на котором установлен сервис **zookeeper server** можно воспользоваться командой:

/usr/lib/zookeeper/bin/zkServer.sh status

Copy

При корректно работающем сервисе в выводе должен быть упомянут путь до файла конфигурации, используемый сетевой порт, сетевой адрес/имя, статус SSL и режим работы сервиса (Mode: **leader**или **follower**. На одном хосте должен быть режим – **leader**, на остальных – **follower**) :

## 3.2 Проверка статуса nifi

???

# 4 TLS (SSL) сертификаты при использовании Kerberos

При установке с керберезированного кластера требется наличие TLS сертификатов. Понадобится:  
- CA ключ (1 экзмпляр)  
- CA сертификат (1 экзмпляр)  
- TLS ключ для сервера (1 \* кол. серверов nifi)  
- TLS сертификат для сервера (1 \* кол. серверов nifi)

Сертификаты можно генерировать и на стороне Windows server, так и на стороне Linux сервер

В продакшен среде возможно уже будет свой корневой сервер сертификации и корневой сертификат делать не нужно (нужно тогда связаться с его владельцем и получить дальнейшие инструкции от него. Далее описан метод изготовления самозаверенных сертификатов на Linux

**Корневой сертификат**

Создаём закрытый ключ длинной 2048

openssl genrsa -out ca.pem 2048

Copy

Создаём сертификат самоподписанного ключа

openssl req -x509 -sha256 -new -nodes -key ca.pem -days 3650 -out ca-cert.pem \

-subj "/C=RU/ST='Irkutskaya Oblast'/L=Irkutsk/O=ACLTelekom/OU=IT/CN=server\_name"

Copy

**Сертификаты нод (серверов nifi)**

# Создаём зашифрованный запрос на выпуск сертификата - CSR (Certificate Signing Request)

openssl req -newkey rsa:2048 -nodes -days 3650 \

-keyout server-key.pem \

-out server-req.pem \

-subj "/C=RU/ST='Irkutskaya Oblast'/L=Irkutsk/O=ACLTelekom/OU=IT/CN=server\_name"

Copy

Создаём из CSR, ключа сервера и CA сертификат ноды

openssl x509 -req -days 3650 -set\_serial 01 \

-in server-req.pem \

-out server-cert.pem \

-CA cacert.pem \

-CAkey ca.pem

# Введение

**Примечание:** Непосредственно операции по подготовке к созданию и созданию кластера описаны в документации к RT.ClusterManager. Данные о порядке создания компонент и по распределению сервисов компонент по нодам, описанные  в данном документе, относятся к операциям описанным в [п. 8.4.1 и п. 8.4.2 документа “RT.ClusterManager. Руководство администратора”](https://docs.data.rt.ru/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F/RT_ClusterManager/Vers_2/RT_ClusterManager_Admin_Guide_ver2#:~:text=8.4.1%20%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BE%D0%B2).

## 1.1 Конфигурация кластера из 3 нод

Предлагаемая конфигурация кластеров, создаваемых на основании плагина «RT.StreamingKafka», состоит из **3 нод**, принадлежащих одному провайдеру, и отдельного сервера для установки RT.ClusterManager, который не входит в состав кластера:

* m-1,
* m-2,
* m-3.

В данном описании ноды – сервера (хосты) на базе операционных систем семейства Linux, которые будут включены в кластер.

**Важно: При развёртывании стандартного кластера необходимо строго придерживаться порядка установки компонентов, указанного в таблицах ниже. В иных случаях существует возможность частично откорректировать распределение.**

**Примечание:** В RT.ClusterManager версии 2.Х-ХХ был изменен дизайн и терминология, поменялись местами термины “Компонент” и “Сервис”. Так в RT.ClusterManager версии 1.Х-ХХ: Zookeeper, HDFS … были сервисами, и входящие в них элементы программного обеспечения которые привязывались к нодам  назывались компонентами, а в RT.ClusterManager версии 2.Х-ХХ: Zookeeper, HDFS … стали компонентами, и входящие в них элементы программного обеспечения которые привязываются к нодам  называются сервисами. В данном документе используется терминология RT.ClusterManager версии 2.Х-ХХ.

# ****2 Распределение сервисов по нодам - плагин «****RT.StreamingKafka****»****

В следующей таблице представлен порядок распределения по нодам компонентов и их сервисов на основании плагина «RT.StreamingKafka»:

**Важно: При установке, придерживайтесь порядка компонент, указанных в таблицах.**

## ****2.1 Конфигурация из 3 нод****

| **№** | **Компонент** | **Назначение** | **Распределение сервисов по нодам** |
| --- | --- | --- | --- |
| 1 | **zookeeper** | Программная служба для координации распределенных систем.  Проверка статуса - см. п. 3.1 | **server**- m-1, m-2, m-3 |
| 2 | **kafka** | Распределённое хранилище событий и платформа потоковой обработки.  Проверка статус - см. п. 3.2 | **kafka**- m-1, m-2, m-3 |

**Важно:** Данный раздел актуален для Платформы данных On-Premise.

# 3 Проверка статуса компонентов

## 3.1 Проверка статуса zookeeper

Для проверки статуса zookeeper на каждом хосте, на котором установлен сервис **zookeeper server** можно воспользоваться командой:

/usr/lib/zookeeper/bin/zkServer.sh status

Copy

При корректно работающем сервисе в выводе должен быть упомянут путь до файла конфигурации, используемый сетевой порт, сетевой адрес/имя, статус SSL и режим работы сервиса (Mode: **leader**или **follower**. На одном хосте должен быть режим – **leader**, на остальных – **follower**) :

## 3.2 Проверка статуса kafka

Для проверки статуса kafka необходимо проверить, запущен ли прием на порту:

netstat -tlpn | grep 9092