# Кластеризация в Proxmox VE - Академия Selectel

selectel.ru/blog/proxmox-cluster

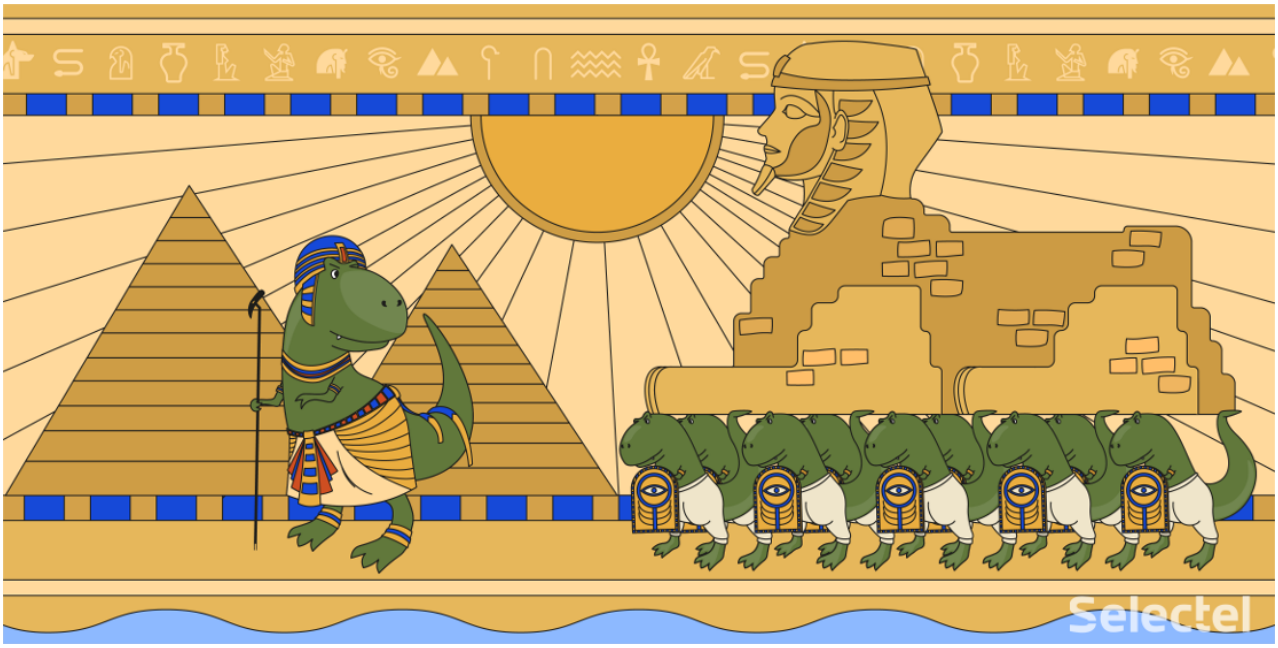
19 декабря 2019 г.

## Кластеризация в Proxmox VE

Николай Рубанов Старший технический писатель

19 декабря 2019

В прошлых статьях мы начали рассказывать о том, что такое Proxmox VE и как он работает. Сегодня мы расскажем о том, как можно использовать возможность кластеризации и покажем какие преимущества это дает. Что же такое кластер и зачем он нужен? Кластер (от англ. cluster) — это группа серверов, объединенных скоростными каналами связи, работающая и представляющаяся […]




В прошлых статьях мы начали рассказывать о том, что такое Proxmox VE и как он работает. Сегодня мы расскажем о том, как можно использовать возможность кластеризации и покажем какие преимущества это дает.

Что же такое кластер и зачем он нужен? Кластер (от англ. cluster) — это группа серверов, объединенных скоростными каналами связи, работающая и представляющаяся пользователю как единое целое. Существует несколько основных сценариев использования кластера:

- **Обеспечение отказоустойчивости** (High-availability).
- **Балансировка нагрузки** (Load Balancing).
- **Увеличение производительности** (High Performance).
- **Выполнение распределенных вычислений** (Distributed computing).

Каждый сценарий предъявляет свои собственные требования к составляющим кластера. Например, для кластера, выполняющего распределенные вычисления, основным требованием является высокая скорость выполнения операций с плавающей запятой и низкая латентность сети. Подобные кластеры часто используются в научно-исследовательских целях.

Раз уж мы коснулись темы распределенных вычислений, то хочется отметить, что существует еще такое понятие как **грид-система** (от англ. grid — решетка, сеть). Несмотря на общую схожесть, не стоит путать грид-систему и кластер. Грид не является кластером в привычном понимании. В отличие от кластера, входящие в грид узлы чаще всего разнородны и отличаются низкой доступностью. Такой подход упрощает решение задач распределенных вычислений, однако не позволяет создать из узлов единое целое.

Яркий пример грид-системы — популярная вычислительная платформа BOINC (Berkeley Open Infrastructure for Network Computing). Эта платформа изначально создавалась для проекта SETI@home (Search for Extra-Terrestrial Intelligence at Home), занимающегося проблемой поиска внеземного разума путем анализа радиосигналов.

## Как это работает

Огромный массив данных, полученных с радиотелескопов, разбивается на множество небольших кусочков, и они отправляются на узлы грид-системы (в проекте SETI@home роль подобных узлов играют компьютеры добровольцев). Данные обрабатываются на узлах и после завершения обработки отправляются на центральный сервер проекта SETI. Таким образом проект решает сложнейшую глобальную задачу, не имея в своем распоряжении требуемых вычислительных мощностей.

Теперь, когда у нас есть четкое понимание того, что такое кластер, предлагаем рассмотреть каким образом его можно создать и задействовать. Будем использовать систему виртуализации с открытым исходным кодом Proxmox VE.

Особенно важно перед тем, как приступать к созданию кластера четко понимать ограничения и системные требования Proxmox, а именно:

- максимальное количество нод в кластере — **32**;
- все ноды должны иметь **одинаковую версию Proxmox** (есть исключения, но для production они не рекомендуются);
- если в дальнейшем планируется задействовать функционал High Availability, то в кластере должно быть как минимум **3** ноды;
- для взаимодействия нод друг с другом должны быть открыты порты **UDP/5404**, **UDP/5405** для corosync и **TCP/22** для SSH;
- задержка в сети между нодами не должна превышать **2** мс.

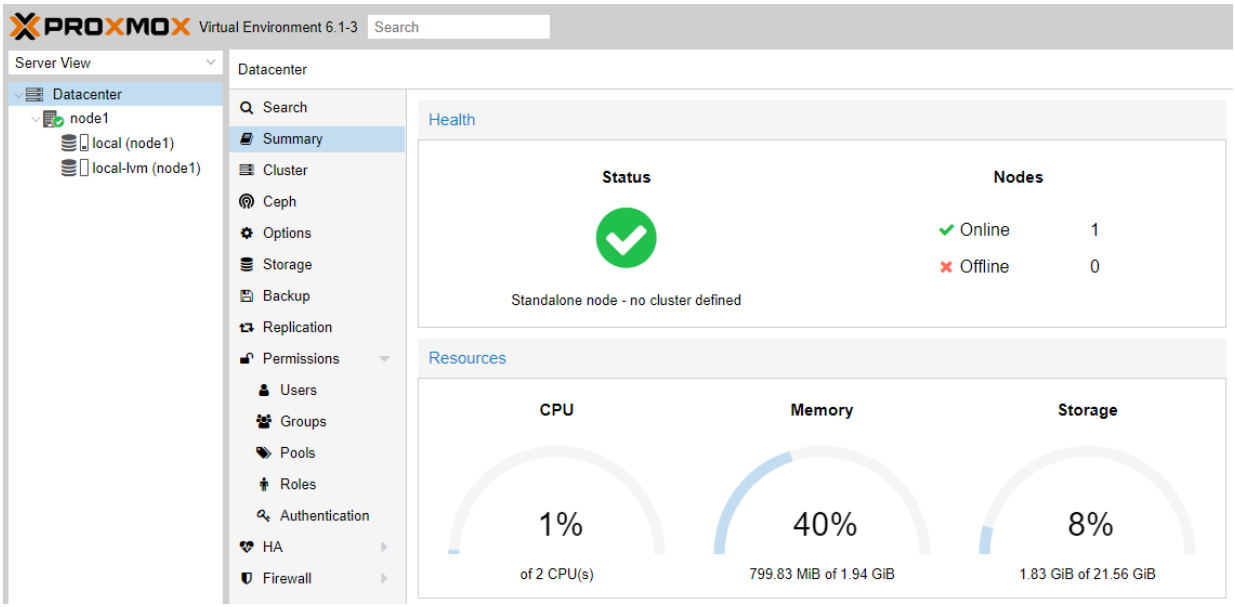
## Создание кластера

**Важно! Нижеприведенная конфигурация является тестовой. Не забудьте свериться с официальной документацией Proxmox VE.**

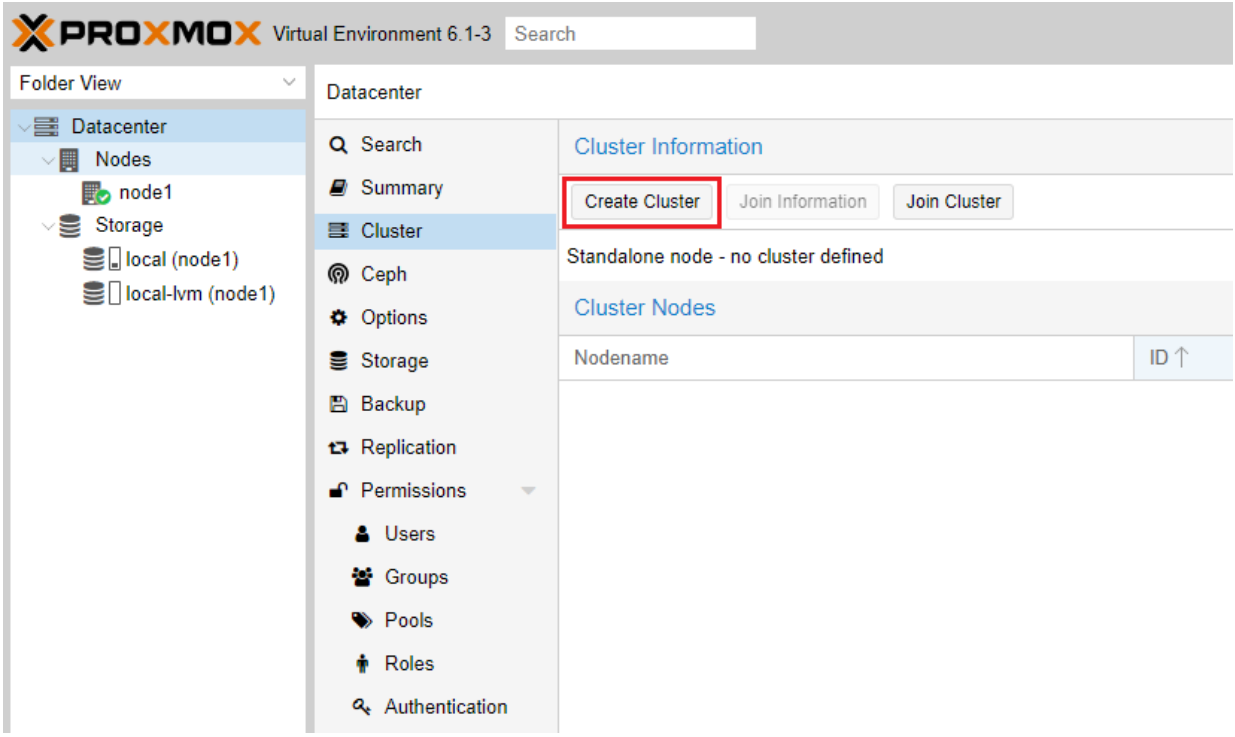
Для того, чтобы запустить тестовый кластер, мы взяли три сервера с установленным гипервизором Proxmox одинаковой конфигурации (2 ядра, 2 Гб оперативной памяти).

Если вы хотите узнать каким образом можно установить Proxmox, то рекомендуем прочитать нашу предыдущую статью — Магия виртуализации: вводный курс в Proxmox VE

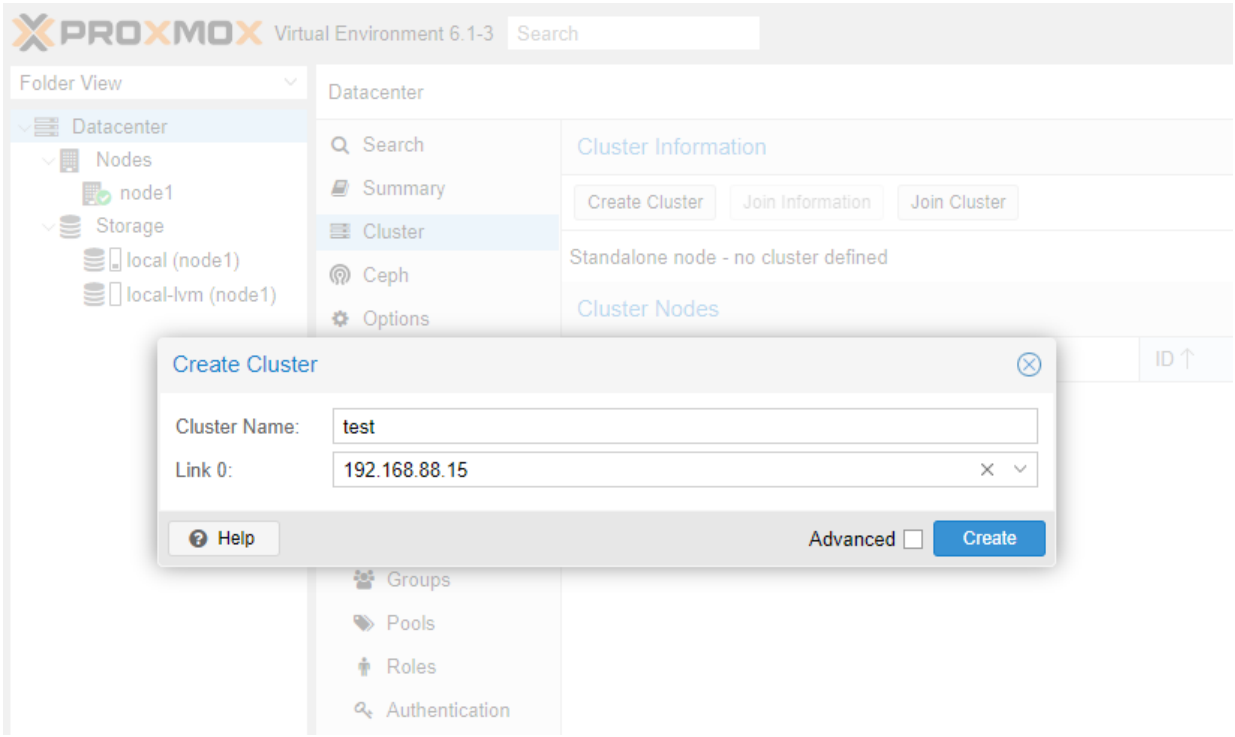
Изначально, после установки ОС, единичный сервер работает в **Standalone-mode**.



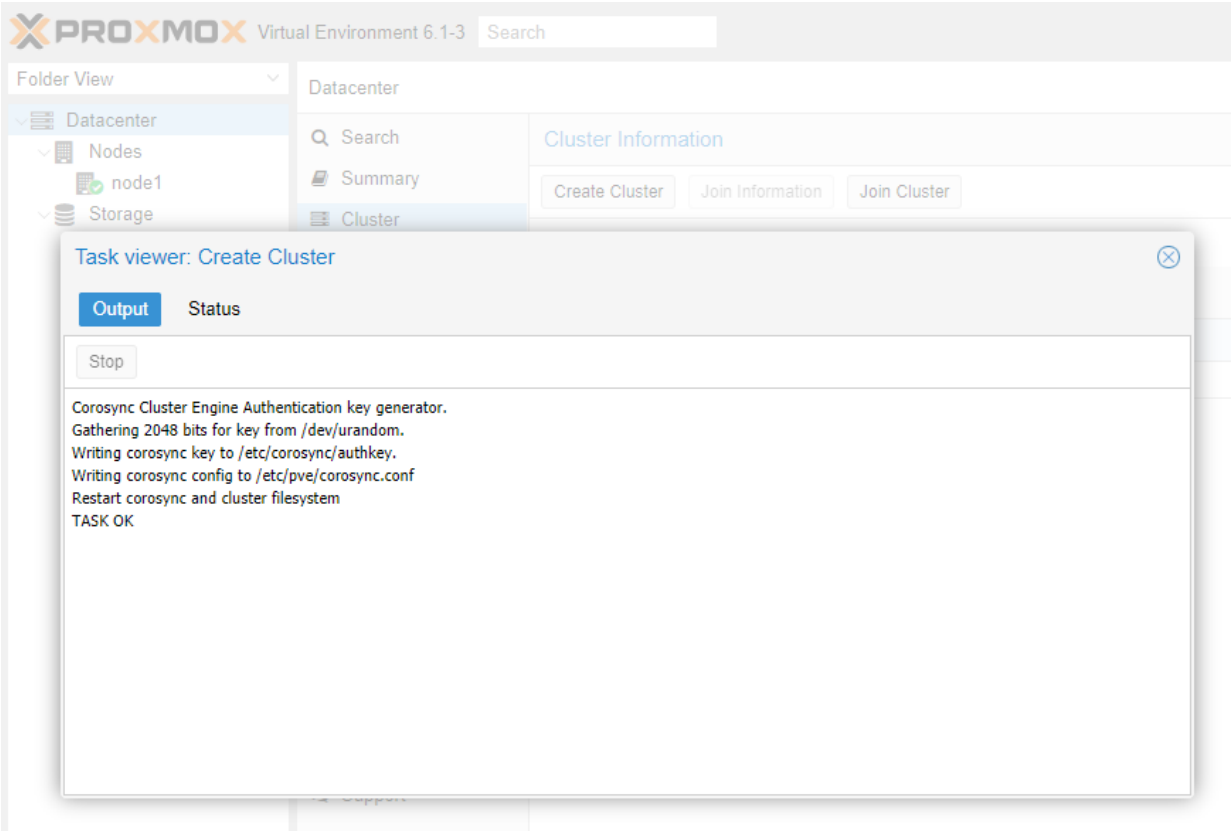

Создадим кластер, нажав кнопку **Create Cluster** в соответствующем разделе.



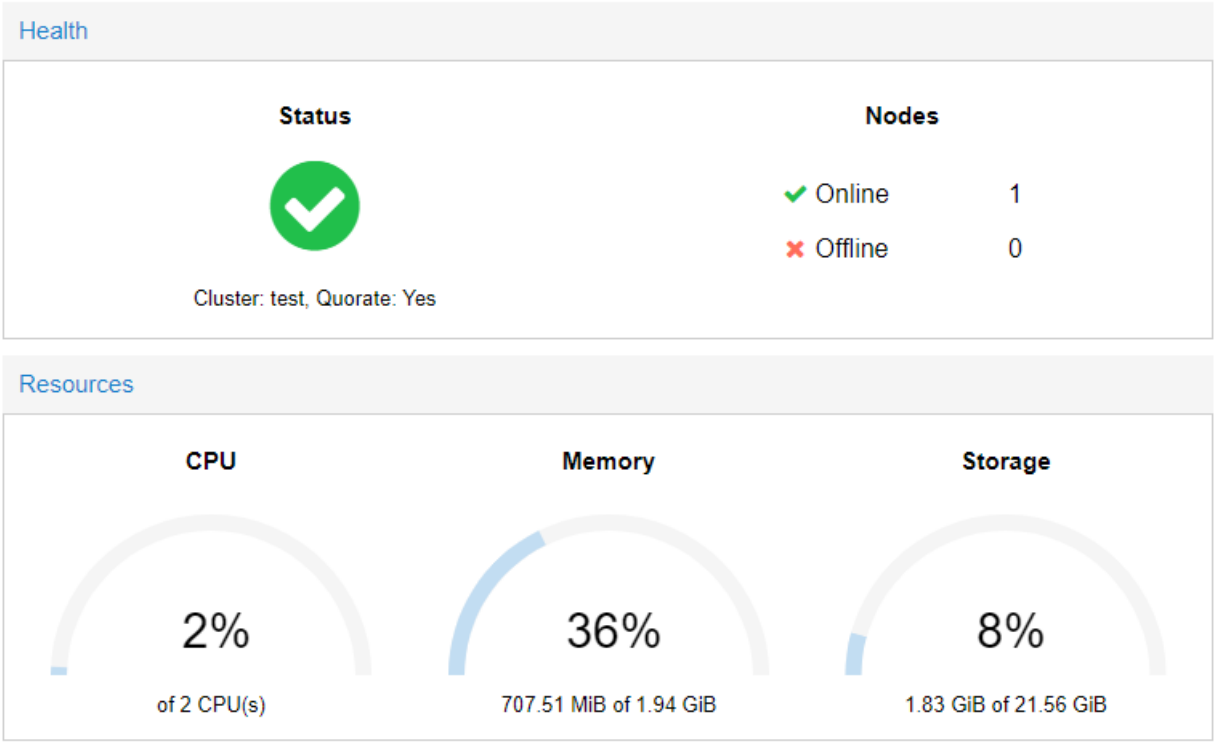

Задаем имя будущему кластеру и выбираем активное сетевое соединение.




Нажимаем кнопку **Create**. Сервер сгенерирует 2048-битный ключ и запишет его вместе с параметрами нового кластера в конфигурационные файлы.

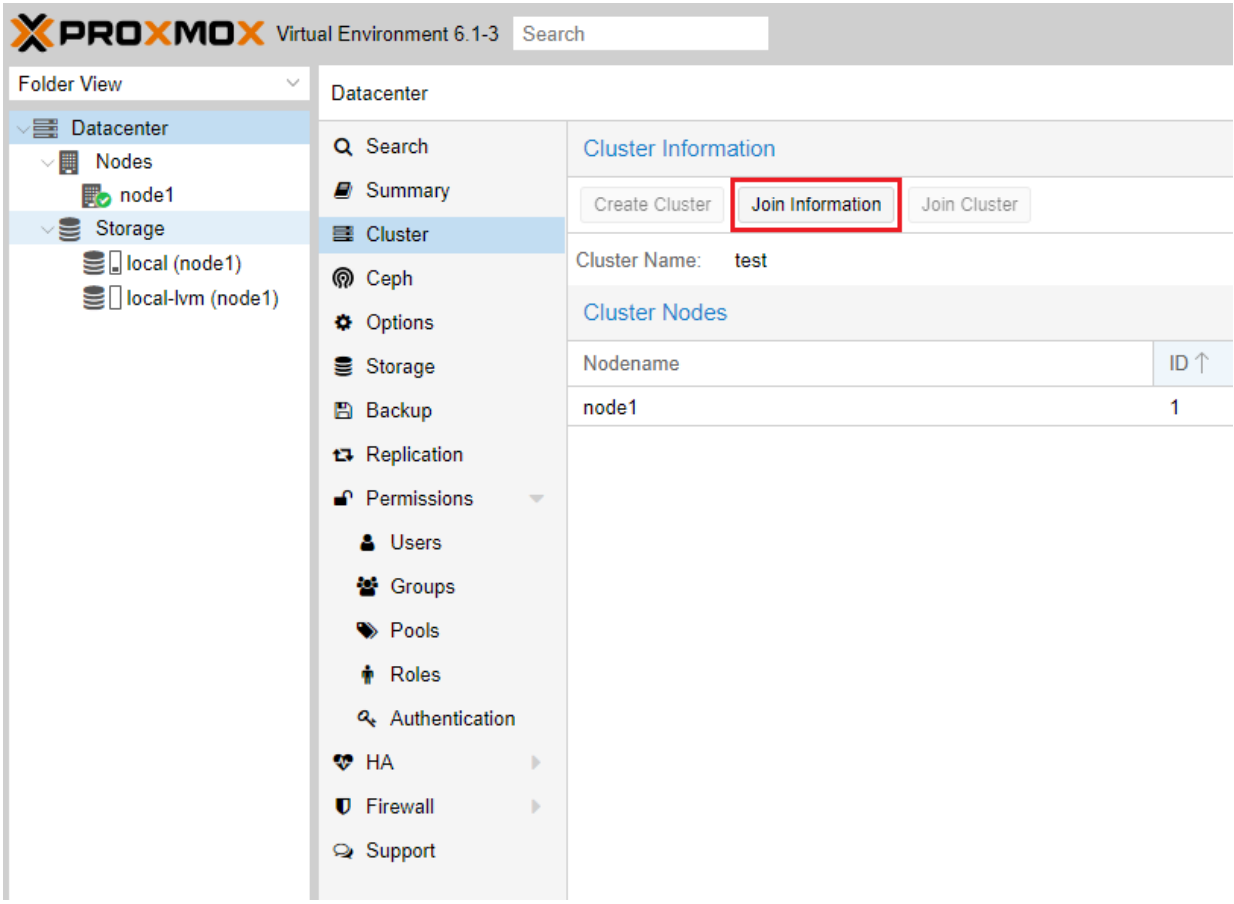



Надпись **TASK OK** свидетельствует об успешном выполнении операции. Теперь, взглянув на общую информацию о системе видно, что сервер перешел в режим кластера. Пока что кластер состоит всего лишь из одной ноды, то есть пока у него нет тех возможностей, ради которых необходим кластер.

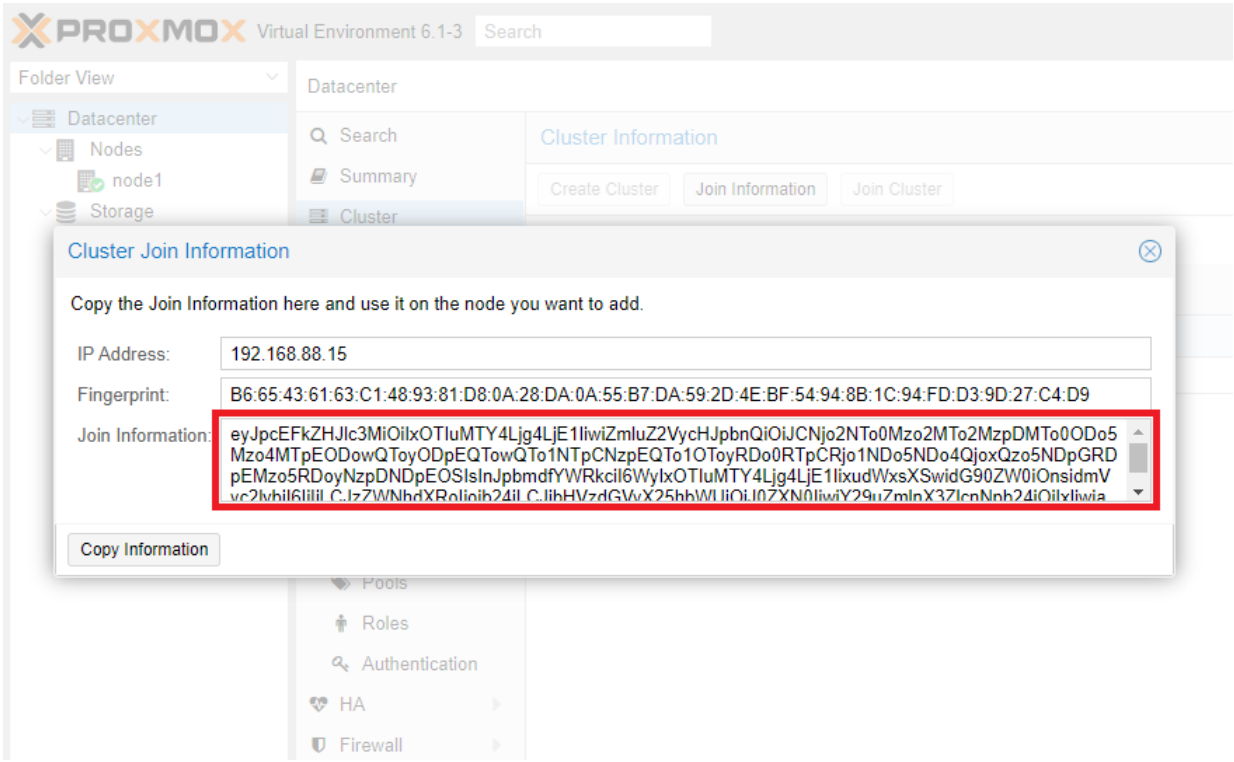



## Присоединение к кластеру

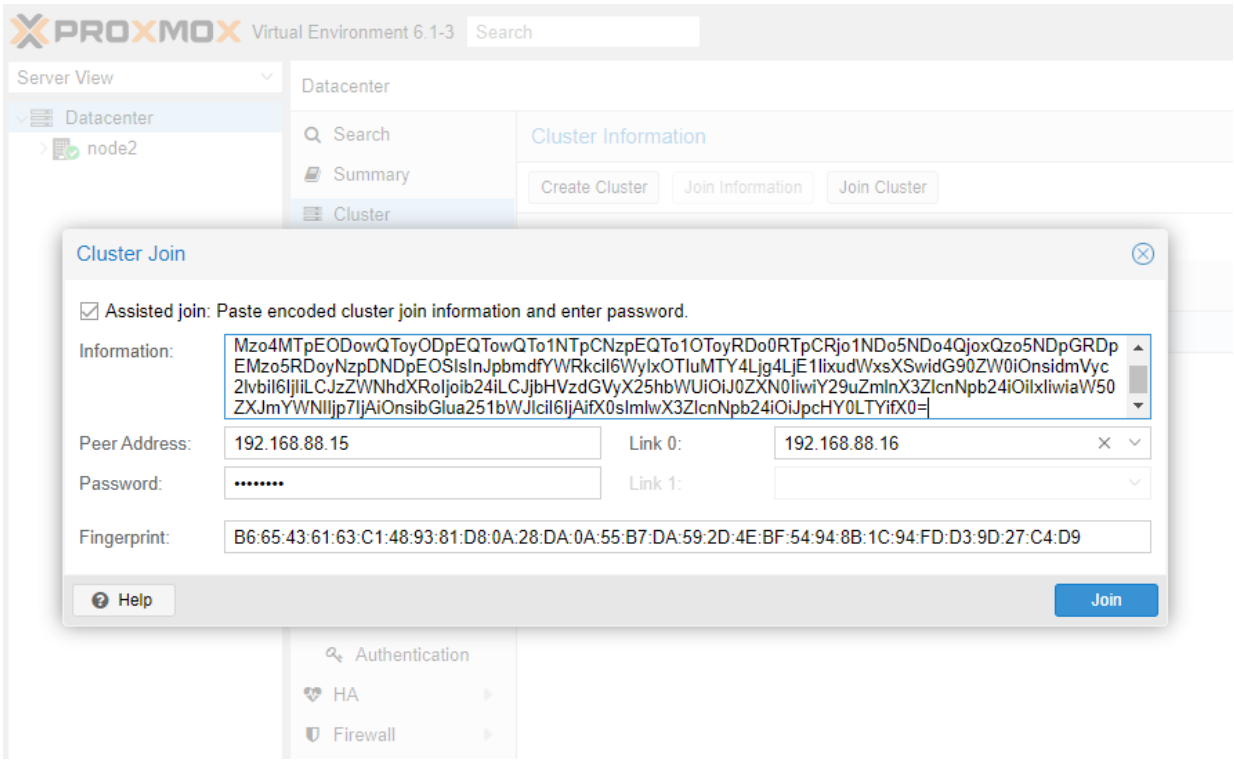
Прежде чем подключаться к созданному кластеру нам необходимо получить информацию для выполнения подключения. Для этого заходим в раздел **Cluster** и нажимаем кнопку **Join Information**.



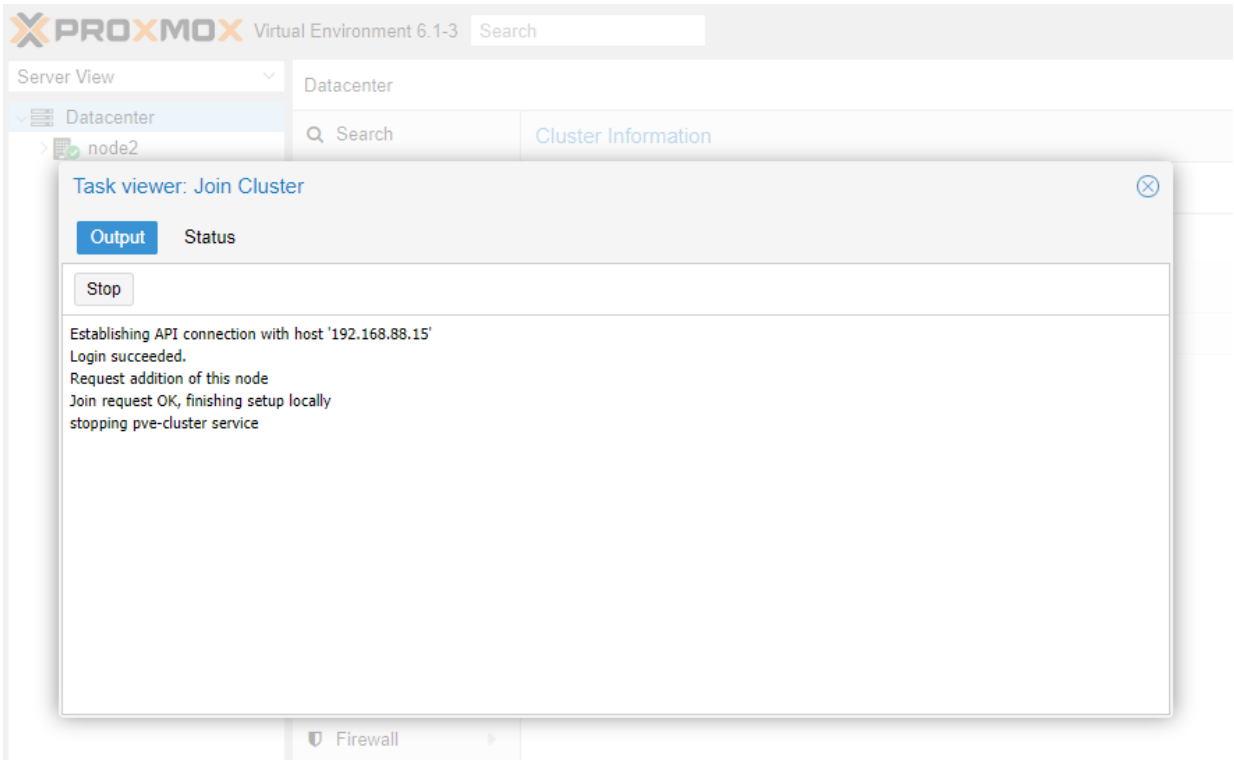

В открывшемся окне нас интересует содержимое одноименного поля. Его необходимо будет скопировать.



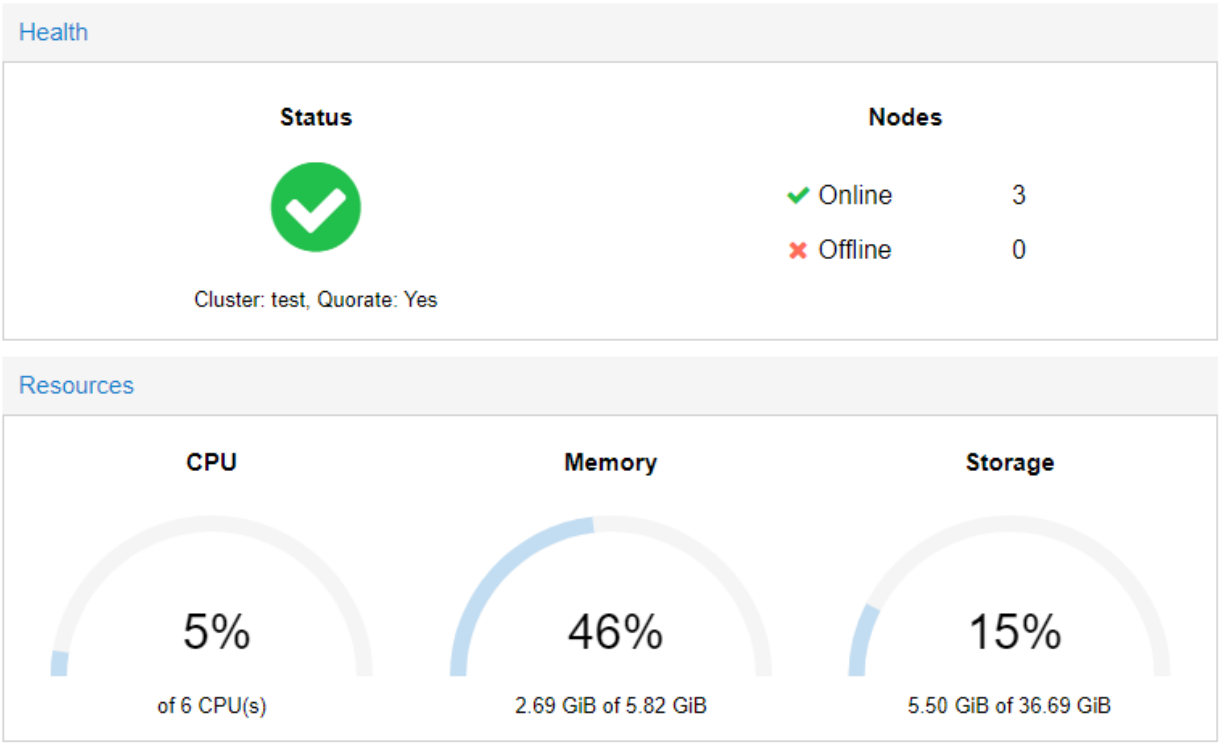
Здесь закодированы все необходимые параметры подключения: адрес сервера для подключения и цифровой отпечаток. Переходим на сервер, который необходимо включить в кластер. Нажимаем кнопку **Join Cluster** и в открывшемся окне вставляем скопированное содержимое.



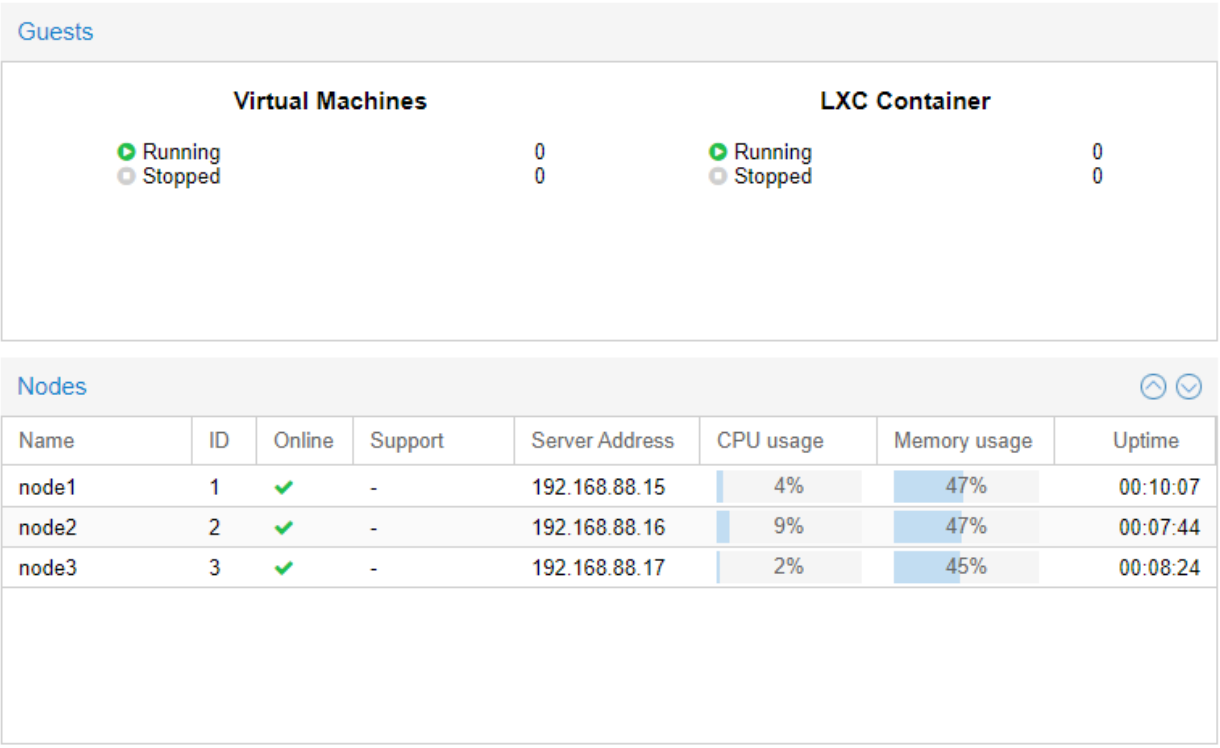

Поля **Peer Address** и **Fingerprint** будут заполнены автоматически. Вводим пароль root от ноды номер 1, выбираем сетевое подключение и нажимаем кнопку **Join**.



В процессе присоединения к кластеру веб-страница GUI может перестать обновляться. Это нормально, просто перезагружаем страницу. Точно таким же образом добавляем еще одну ноду и в итоге получаем полноценный кластер из 3-х работающих узлов.




Теперь мы можем контролировать все узлы кластера из одного GUI.




## Организация High Availability

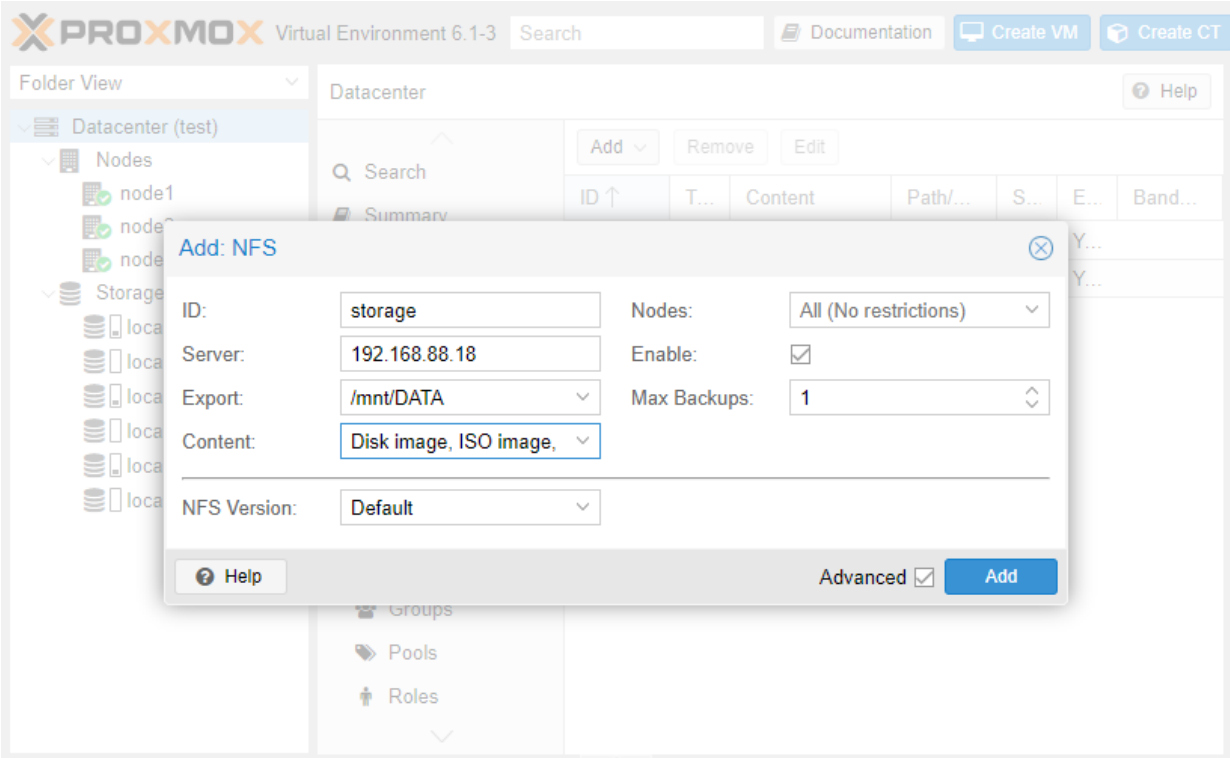
Proxmox «из коробки» поддерживает функционал организации HA как для виртуальных машин, так и для LXC-контейнеров. Утилита **ha-manager** обнаруживает и отрабатывает ошибки и отказы, выполняя аварийное переключение с отказавшей ноды на рабочую. Чтобы механизм работал корректно необходимо, чтобы виртуальные машины и контейнеры имели общее файловое хранилище.

После активации функционала High Availability, программный стек ha-manager начнет непрерывно отслеживать состояние работы виртуальной машины или контейнера и асинхронно взаимодействовать с другими нодами кластера.

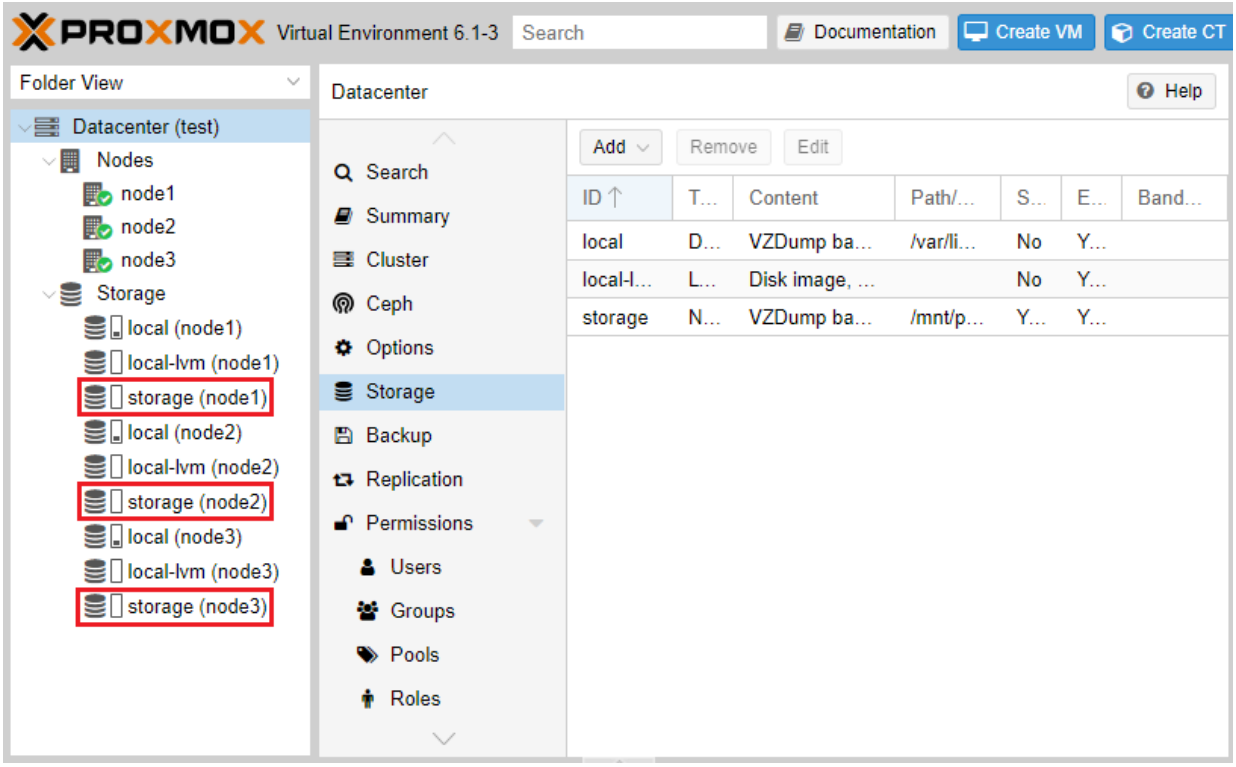
## Присоединяем общее хранилище

Для примера мы развернули небольшое файловое хранилище NFS по адресу 192.168.88.18. Чтобы все ноды кластера могли его использовать нужно проделать следующие манипуляции.

Выбираем в меню веб-интерфейса **Datacenter** — **Storage** — **Add** — **NFS**.




Заполняем поля **ID** и **Server**. В выпадающем списке **Export** выбираем нужную директорию из доступных и в списке **Content** — необходимые типы данных. После нажатия кнопки **Add** хранилище будет подключено ко всем нодам кластера.

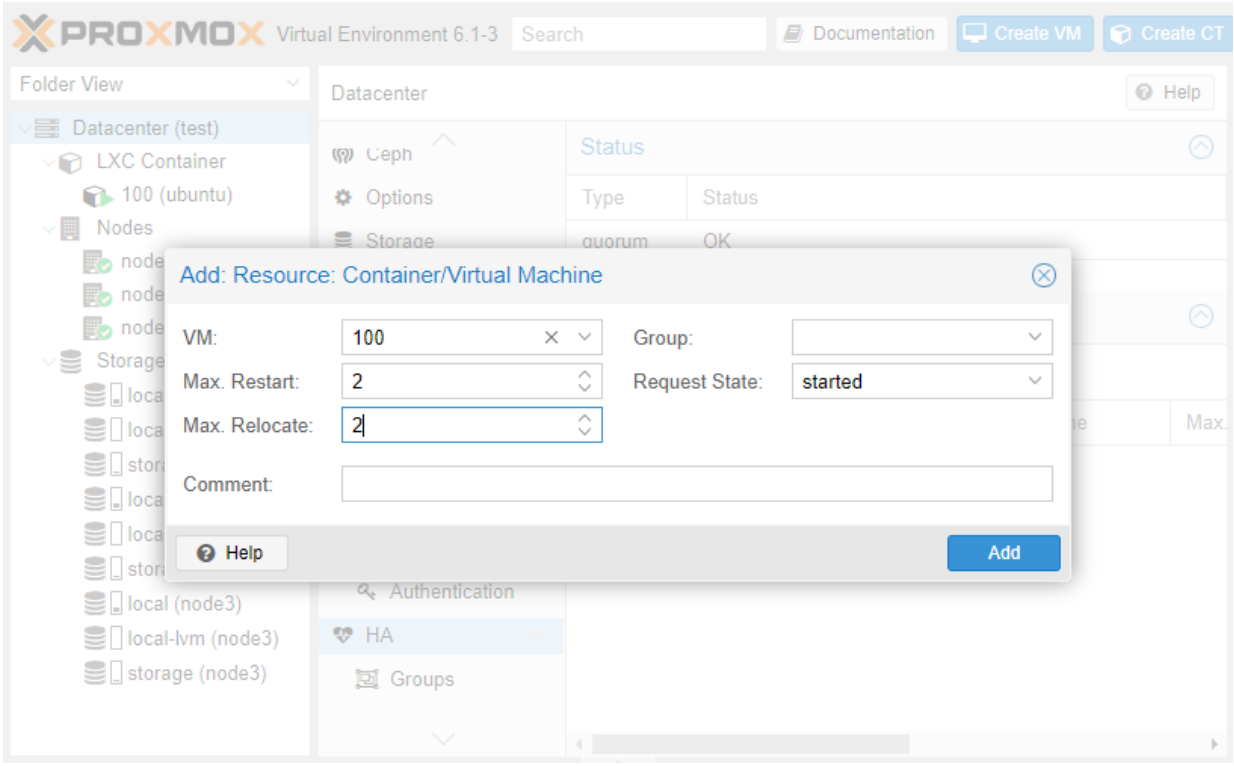



При создании виртуальных машин и контейнеров на любом из узлов указываем наш **storage** в качестве хранилища.

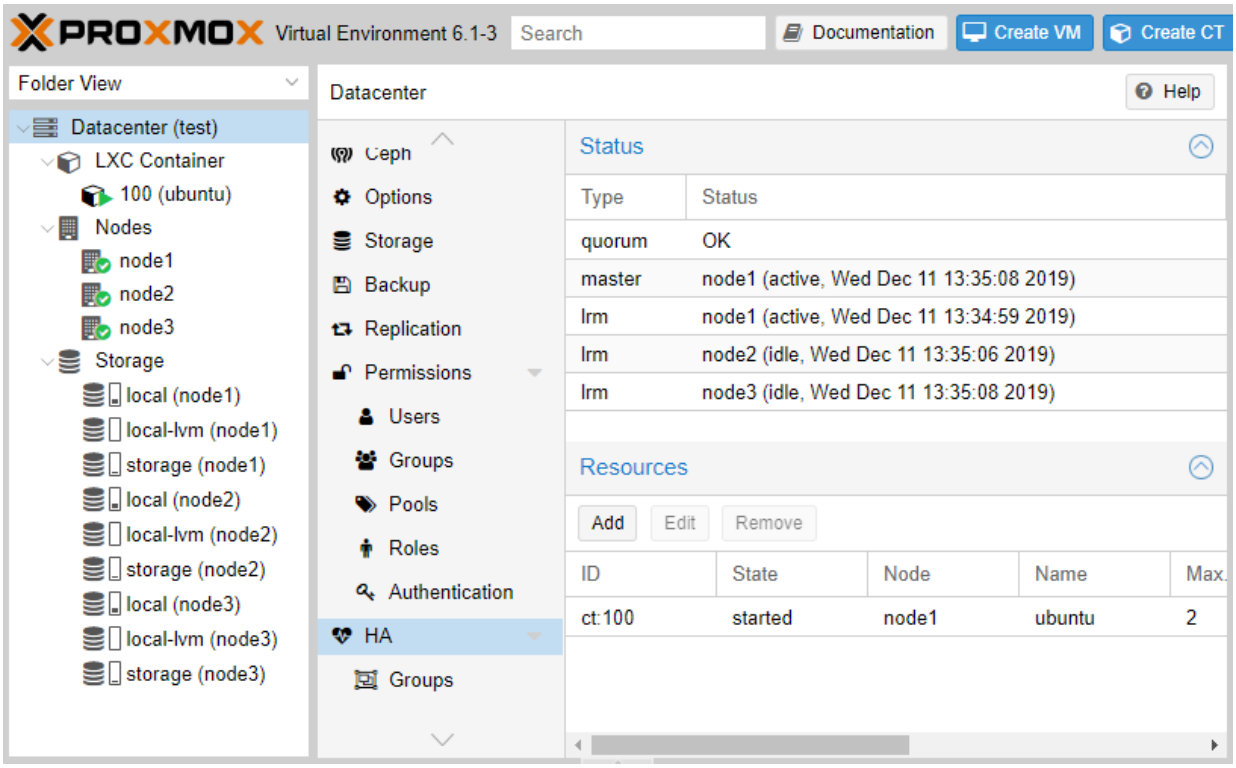
## Настраиваем HA

Для примера создадим контейнер с Ubuntu 18.04 и настроим для него High Availability. После создания и запуска контейнера заходим в раздел **Datacenter** — **HA** — **Add**. В открывшемся поле указываем ID виртуальной машины/контейнера и максимальное количество попыток рестарта и перемещения между нодами.

Если это количество будет превышено, гипервизор пометит VM как сбойную и переведет в состояние Error, после чего прекратит выполнять с ней какие-либо действия.

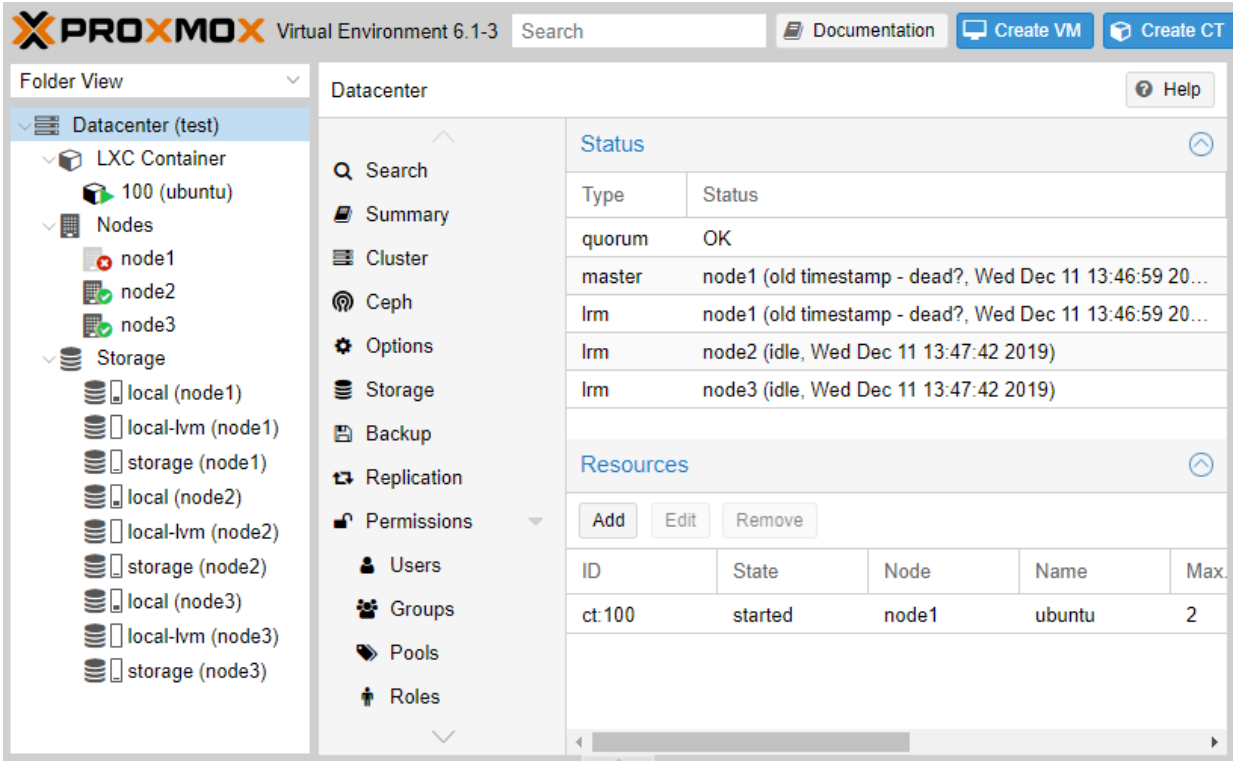



После нажатия кнопки **Add** утилита **ha-manager** оповестит все ноды кластера о том, что теперь VM с указанным ID контролируется и в случае падения ее необходимо перезапустить на другой ноде.



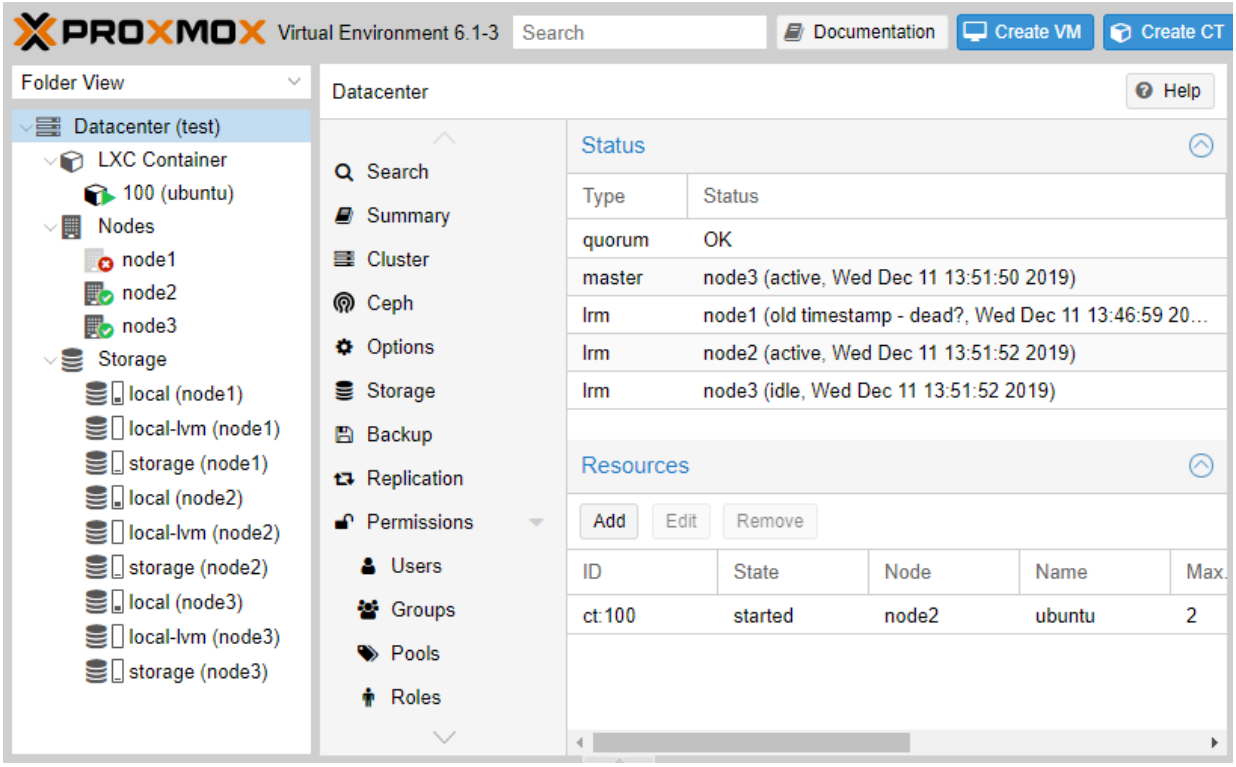

## Устроим сбой

Чтобы посмотреть, как именно отрабатывает механизм переключения, погасим нештатно node1 по питанию. Смотрим с другой ноды, что происходит с кластером. Видим, что система зафиксировала сбой.

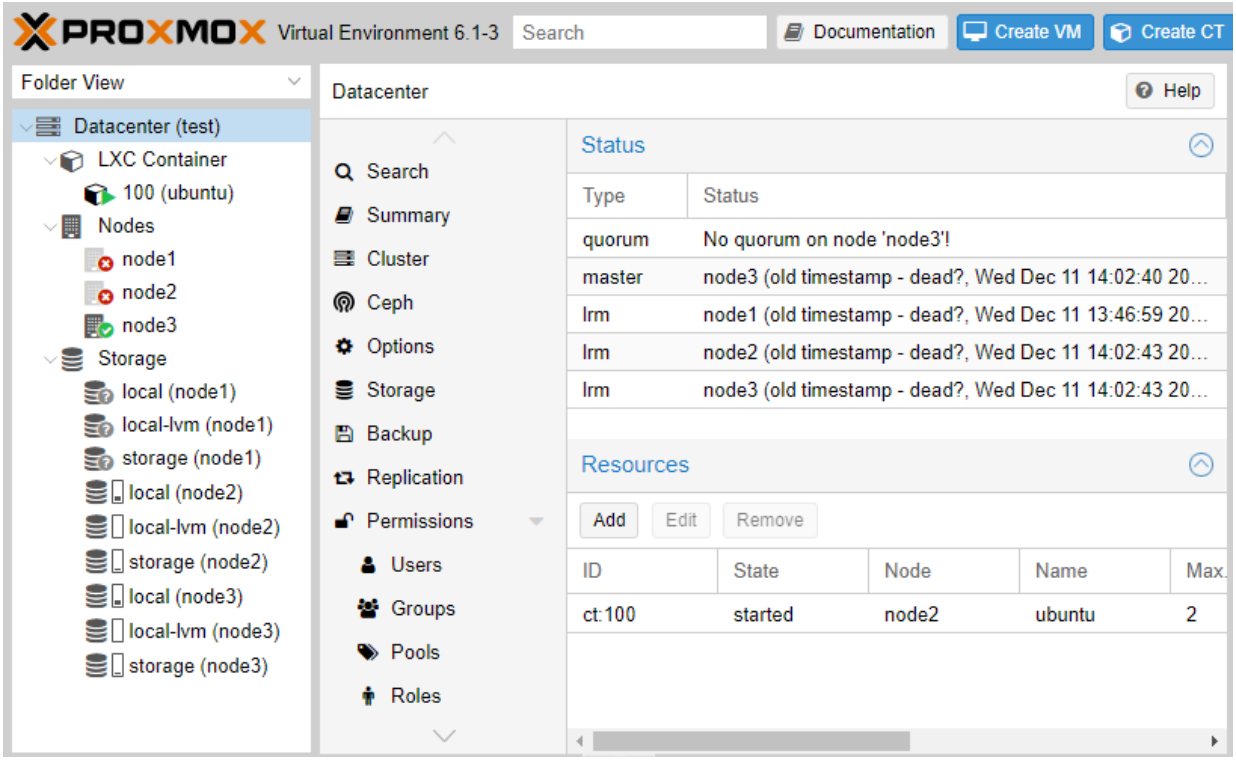



Работа механизма HA не означает непрерывность работы VM. Как только нода «упала», работа VM временно останавливается до момента автоматического перезапуска на другой ноде.

И вот тут начинается «магия» — кластер автоматически переназначил ноду для выполнения нашей VM и в течение 120 секунд работа была автоматически восстановлена.

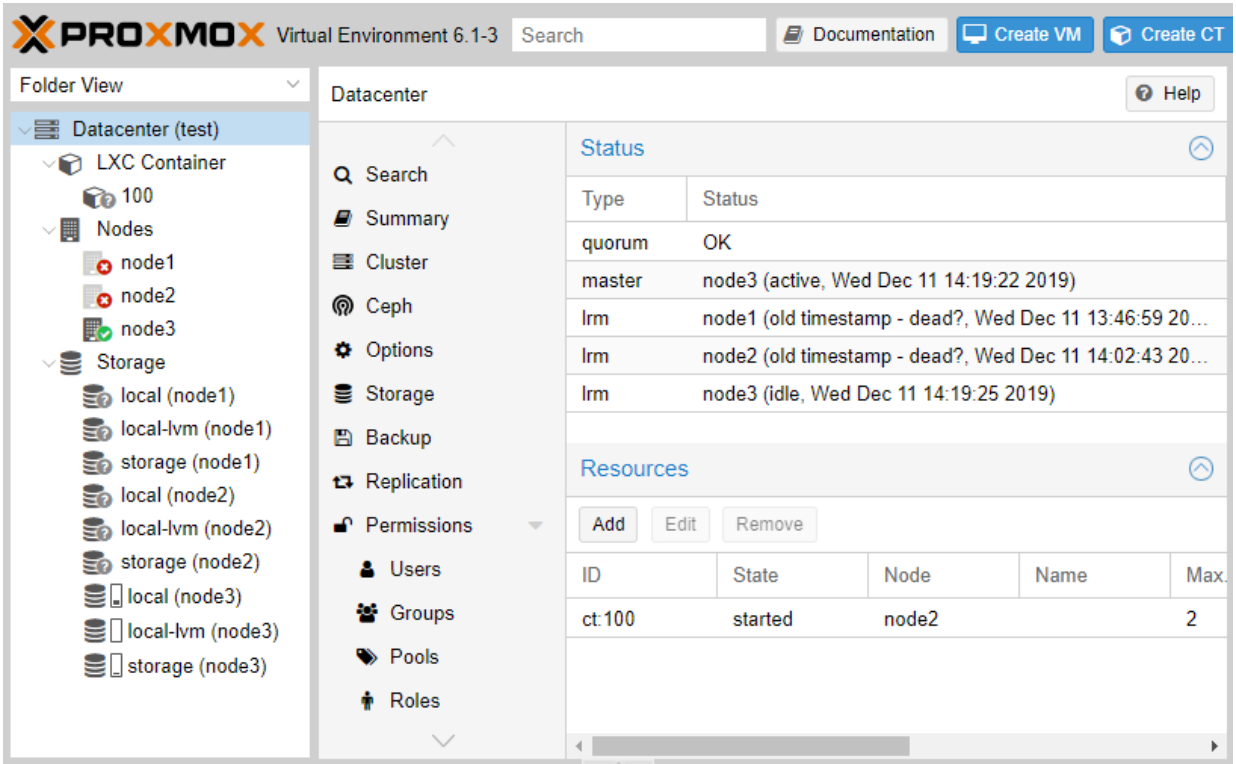



Гасим node2 по питанию. Посмотрим, выдержит ли кластер и вернется ли VM в рабочее состояние автоматически.

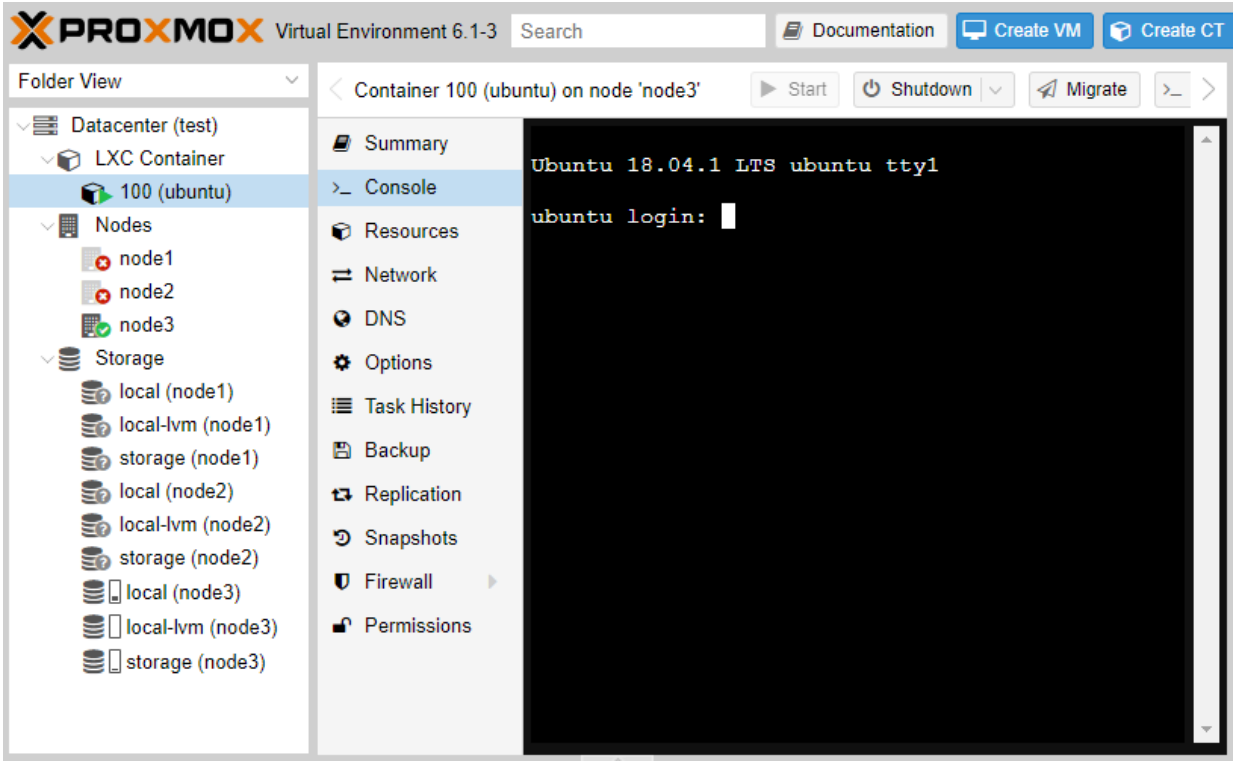



Увы, как видим, у нас возникла проблема с тем, что на единственной, оставшейся в живых, ноде больше нет кворума, что автоматически отключает работу HA. Даем в консоли команду принудительной установки кворума.

```
pvecm expected 1
```




Спустя 2 минуты механизм HA отработал корректно и не найдя node2 запустил нашу VM на node3.




Как только мы включили обратно node1 и node2, работа кластера была полностью восстановлена. Обратите внимание, что обратно на node1 VM самостоятельно не мигрирует, но это можно сделать вручную.

## Подводим итоги

Мы рассказали вам про то, как устроен механизм кластеризации в Proxmox, а также показали, как настраивается HA для виртуальных машин и контейнеров. Грамотное использование кластеризации и HA значительно повышает надежность инфраструктуры, а также обеспечивает восстановление после сбоев.

Перед тем как создавать кластер, нужно сразу планировать для каких целей он будет использоваться и насколько его потребуется масштабировать в будущем. Также нужно проверить сетевую инфраструктуру на готовность работы с минимальными задержками, чтобы будущий кластер работал без сбоев.

Расскажите нам — используете ли вы возможности кластеризации в Proxmox? Ждем вас в комментариях.

*Предыдущие статьи на тему гипервизора Proxmox VE:*

Выделенные серверы и оборудование