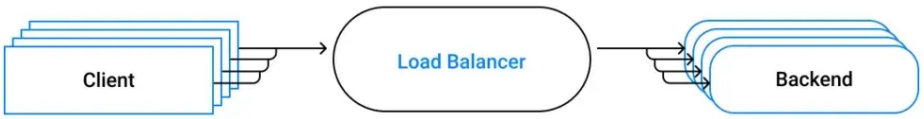
Теория.

Балансировка нагрузки

Вы разрабатываете сайт интернет-магазина и хотите, чтобы он был готов к неожиданному наплыву посетителей и никогда не падал. Абсолютной стабильности, конечно, не добиться. Но в ваших силах сделать сайт устойчивым ко многим проблемам.
Пережить одновременный визит множества пользователей поможет развёртывание копий сайта на нескольких ВМ. В таком случае нагрузка равномерно распределится между ними. Если машин пять — то каждой достанется 20% запросов. Такой подход называется **сетевой балансировкой**.
Работает это так. Перед ВМ с сайтом ставят **балансировщик** — приложение, которое принимает запросы от пользователей и распределяет их по ВМ, а затем получает от ВМ ответы и передаёт (проксирует) их пользователям. При этом сервису, передающему трафик, не нужно знать адреса и названия ВМ: процедура разрешения имён делегируется балансировщику. Это называется **абстрактностью имён**.
Балансировка не только помогает распределить нагрузку между серверами, но и защищает веб-приложение от выхода ВМ из строя. Предположим, на одном из серверов возникли неполадки и он не может обрабатывать запросы. В этом случае балансировщик перераспределит нагрузку между другими серверами, и с этого момента недоступность сервиса для конечных пользователей прекратится.



Кроме того, балансировщик позволяет незаметно для пользователей обновлять код сайта или веб-приложения на серверах: вы просто поочередно убираете ВМ из-под балансировки, обновляете софт, после чего возвращаете их под балансировку.
Добиться максимальной доступности сайта или приложения можно, разместив ВМ в разных зонах доступности. Если в одной зоне произойдёт авария и все ВМ выйдут из строя — балансировщик начнёт распределять трафик по ВМ в других зонах, пока работа отключённой зоны не восстановится.

Yandex Network Load Balancer

Чтобы настроить сетевую балансировку в Yandex Network Load Balancer, разберёмся с двумя базовыми понятиями.
Первое — это **целевая группа**, т. е. набор серверов или других облачных ресурсов, по которым распределяются запросы пользователей. Целевая группа выглядит как список внутренних IP-адресов и подсетей, к которым эти IP-адреса относятся.
Допустим, вам нужно распределить трафик по пяти виртуальным машинам (ВМ). В этом случае целевая группа балансировщика может выглядеть так:

```
10.10.10.15, e9b7a3k9rqq3j0j36m9u
10.10.10.20, e9b7a3k9rqq3j0j36m9u
10.10.20.31, e2lgvksek5io187a48q5
10.10.20.10, e2lgvksek5io187a48q5
10.10.30.20, b0cnsvg8jfoe938ktqp4
```

Здесь перечислены пять внутренних IP-адресов, причём для каждого адреса указан идентификатор его подсети. Все адреса целевых ресурсов должны принадлежать одной облачной сети.
Чтобы посмотреть список подсетей и их идентификаторов, откройте в консоли управления раздел **Virtual Private Cloud** и перейдите на вкладку **Облачные сети**.
Второе базовое понятие — **обработчик**. Это приложение принимает соединения от пользователей, распределяет их между IP-адресами целевой группы, а затем передаёт обратный трафик клиентам.
Адресация трафика строится по принципу 5-tuple: учитывается адрес и порт отправителя, адрес и порт целевого (принимающего) облачного

ресурса, а также протокол передачи информации. Для приёма трафика обработчик использует порты от `1` до `32767`.
При создании сетевого балансировщика необязательно сразу настраивать обработчик. Если хотите, добавьте его позднее.
Кроме того, вы можете задать несколько обработчиков. Это пригодится, если запущенный на ВМ сервис предполагает использование нескольких портов сразу. К примеру, вы используете надстройку над Git наподобие GitLab. Значит, одновременно должны быть доступны и веб-интерфейс, и сервер Git, работающие на разных портах.
Целевую группу можно подключить к нескольким балансировщикам — например, чтобы балансировщики на портах `80` и `443` смогли обрабатывать и HTTP-, и HTTPS-запросы. Однако в этом случае вам придётся использовать разные целевые порты. Если группа подключена к одному балансировщику на порту `8080`, то к другому балансировщику вам придётся подключить её на порту `8081`.
После подключения целевой группы балансировщик начнёт проверять состояние целевых ресурсов и сможет распределять нагрузку между ними.

Проверка состояния

Вы создали сетевой балансировщик, настроили обработчики и указали целевую группу из пяти ВМ, на каждой из которых работает копия веб-сайта. Одна ВМ вышла из строя. Чтобы балансировщик узнал о неполадке и перестал проксировать трафик на проблемную ВМ, настройте проверку состояния: специальный запрос от балансировщика по протоколу TCP или HTTP.
Например, балансировщик раз в 10 секунд запрашивает у каждой ВМ страницу по HTTP. Если все ВМ за отведённое время отдают код `200`, их состояние — `Healthy` (*англ.* здорова). Значит, ВМ готовы принимать трафик. Но если ВМ не успевает ответить, её состояние меняется на `Unhealthy` (*англ*. нездорова). Балансировщик обрабатывает результат проверки и затем перестаёт отправлять трафик на ВМ.
Когда работа ВМ восстанавливается и ВМ успевает за отведённое время отдать код `200` — её статус меняется на `Healthy`.

Как правильно использовать балансировщики

Чтобы построить эффективную инфраструктуру с высокой отказоустойчивостью:

- **Создавайте ресурсы в разных зонах доступности**
  Размещайте копии виртуальных машин (ВМ) в нескольких зонах доступности. Так приложения останутся доступны, даже если одна из зон выйдет из строя.
  В каждой зоне доступности разместите одинаковое количество облачных ресурсов. Если в `ru-central1-a` находится три ВМ — поместите по три ВМ и в `ru-central1-b`, и в `ru-central1-c`.
- **Создавайте облачные ресурсы с запасом**
  Если одна ВМ в зоне доступности выйдет из строя, трафик продолжит поступать в зону в том же объёме, а нагрузка на оставшиеся машины увеличится. Чтобы все ВМ не вышли из строя — помимо ресурсов, необходимых для обслуживания расчётной нагрузки, добавьте в каждой зоне дополнительные вычислительные ресурсы (vCPU, RAM).
- **Используйте разные балансировщики для разных приложений**
  Если вы разворачиваете в Yandex Compute Cloud несколько приложений — настройте для их обслуживания отдельные балансировщики. Это поможет эффективнее управлять нагрузкой.
- **Организуйте два уровня балансировщиков**
  Балансировщики Yandex Cloud работают с протоколами TCP и UDP — так называемыми транспортными протоколами или протоколами четвёртого уровня сетевой модели OSI. Они называются так потому, что предназначены для обеспечения надёжной передачи данных от отправителя к получателю. Кроме того, бывают балансировщики протоколов седьмого уровня — на этом уровне работает, например, протокол HTTP.

Поскольку балансировщики седьмого уровня, например веб-сервер NGINX, выполняют более сложную работу с IP-пакетами (сборка, анализ, журналирование), они выиграют от предварительного

распределения нагрузки на четвертом уровне, особенно при DDoS-атаках.

Организуйте двухуровневую архитектуру с балансировщиками на четвертом (транспортном) уровне OSI и седьмом уровне приложения (например HTTP). Балансировщик четвертого уровня будет принимать трафик и передавать его целевой группе балансировщиков седьмого уровня, а те распределят трафик по ВМ с приложениями. В качестве балансировщиков седьмого уровня вы можете использовать ВМ, на которые установили ПО для балансировки (например NGINX).

Практическая работа. Знакомство с Yandex Cloud CLI

На этом практическом занятии вы научитесь быстрее создавать веб-серверы, чтобы затем помещать их за сетевой балансировщик.
В качестве веб-серверов выступят две ВМ, на которых доступна информационная страница запущенного веб-сервера NGINX. Для реальных проектов с высокой нагрузкой вам, скорее всего, придётся создавать гораздо больше ВМ, поэтому вам пригодится умение делать это автоматически: с помощью консольного интерфейса Yandex Cloud CLI. Подробнее о работе с CLI поговорим в рамках другого курса. Здесь же остановимся на простом примере применения этого инструмента.

1. Установите и настройте Yandex Cloud CLI, следуя инструкциям в документации.
2. Создайте файл `startup.sh` командой touch startup.sh, который будет запускаться на ВМ после её создания, со следующим содержимым:

Скопировать код

```
#!/bin/bash
apt-get update
apt-get install -y nginx
service nginx start
sed -i -- "s/nginx/Yandex Cloud - ${HOSTNAME}/"
/var/www/html/index.nginx-debian.html
EOF
```

Скрипт получает список актуальных пакетов с софтом, устанавливает и запускает NGINX, а затем меняет информационную страницу работающего сервера.

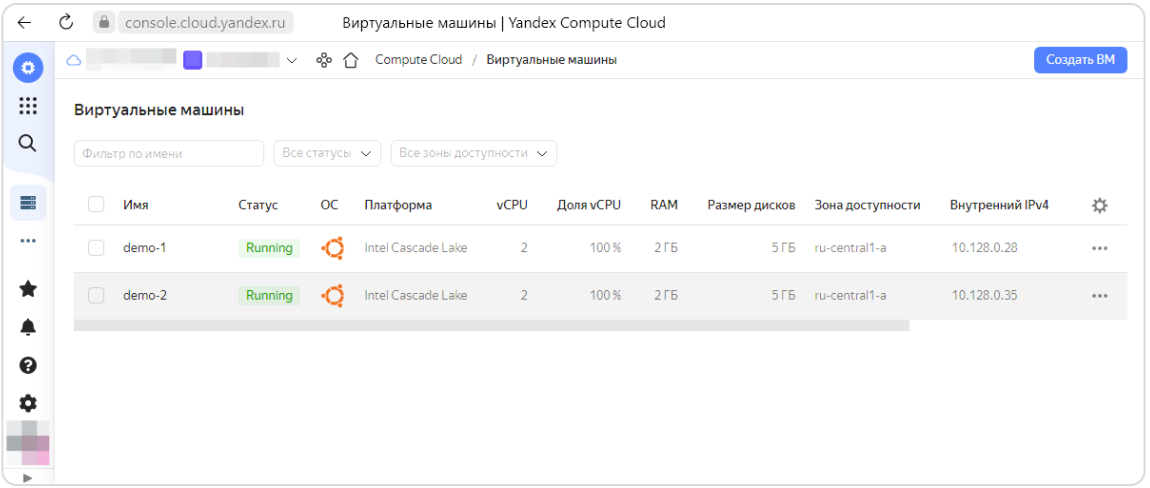
3. Создайте первую ВМ с помощью команды `yc compute instance create`. Чтобы указать, какую именно ВМ мы хотим создать, в команде используются параметры. В данном случае — имя ВМ (`--name demo-1`), откуда взять метаданные (из файла скрипта `startup.sh`, созданного ранее. Вставьте полный путь к файлу), из какого образа создать загрузочный диск (`ubuntu-2004-lts`), в какой зоне создать машину (`--zone ru-central1-a`) и в какую подсеть подключить сетевой интерфейс с IPv4-адресом (`--network-interface ...`).

Скопировать код

```
yc compute instance create \
--name demo-1 \
--metadata-from-file user-data=startup.sh \
--create-boot-disk
image-folder-id=standard-images,image-family=ubuntu-2004-lts \
--zone ru-central1-a \
--network-interface

subnet-name=default-ru-central1-a,nat-ip-version=ipv4
```

4. Самостоятельно измените и запустите еще раз эту команду, чтобы создать такую же ВМ с именем `demo-2`.
5. Теперь проверим, что обе ВМ успешно созданы. Это можно сделать, заглянув в консоль управления:



6. А можно и из командной строки с помощью команды `yc compute instance list`. Попробуйте этот способ. У вас должен получиться примерно такой результат:

Скопировать код

```
+----------------------+--------+---------------+---------+-------------
----+-------------+
|          ID          |  NAME  |    ZONE ID    | STATUS  |  EXTERNAL
IP   | INTERNAL IP |
+----------------------+--------+---------------+---------+-------------
----+-------------+
| fhmnc7nireh12e25ctk0 | demo-1 | ru-central1-a | RUNNING |
84.201.174.97  | 10.130.0.18 |
| fhmrbmchmu3ganplskp2 | demo-2 | ru-central1-a | RUNNING |
84.252.129.231 | 10.130.0.31 |
+----------------------+--------+---------------+---------+-------------

----+-------------+
```

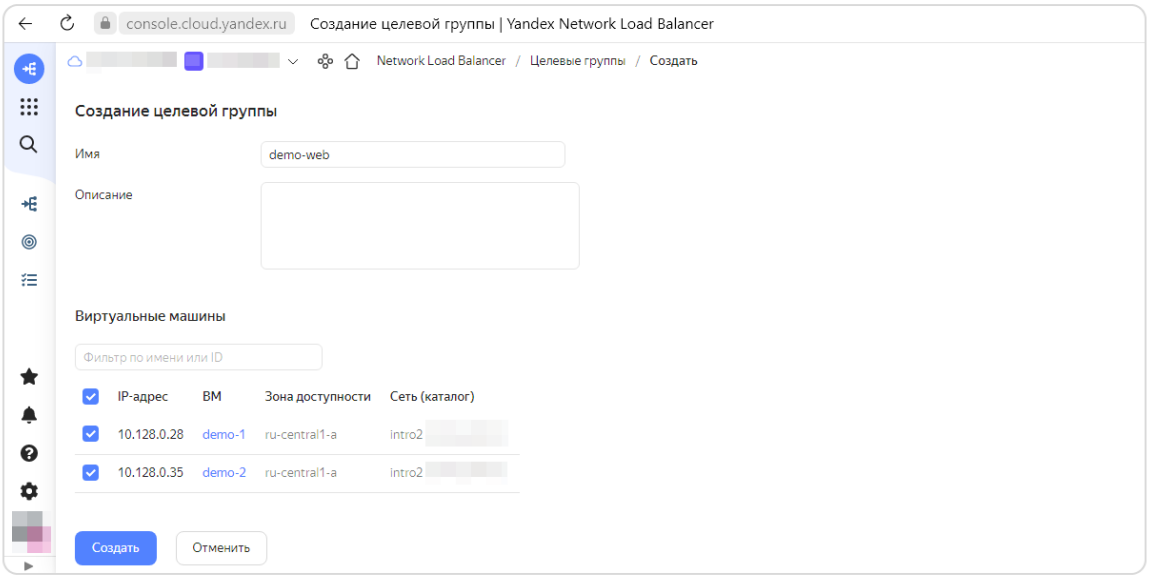
7. Убедитесь, что статус машин сменился на **Running**, а скрипт выполнился (иногда нужно подождать до одной минуты, пока обновится список пакетов и установится NGINX).
8. Введите публичные IP-адреса ВМ в браузере и проверьте, что главные страницы веб-серверов доступны:


51.250.93.95
Welcome to Yandex Cloud - demo-1!
Welcome to Yandex Cloud - demo-1!
If you see this page, the Yandex Cloud - demo-1 web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using Yandex Cloud - demo-1.


51.250.9.14
Welcome to Yandex Cloud - demo-2!
Welcome to Yandex Cloud - demo-2!
If you see this page, the Yandex Cloud - demo-2 web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using Yandex Cloud - demo-2.

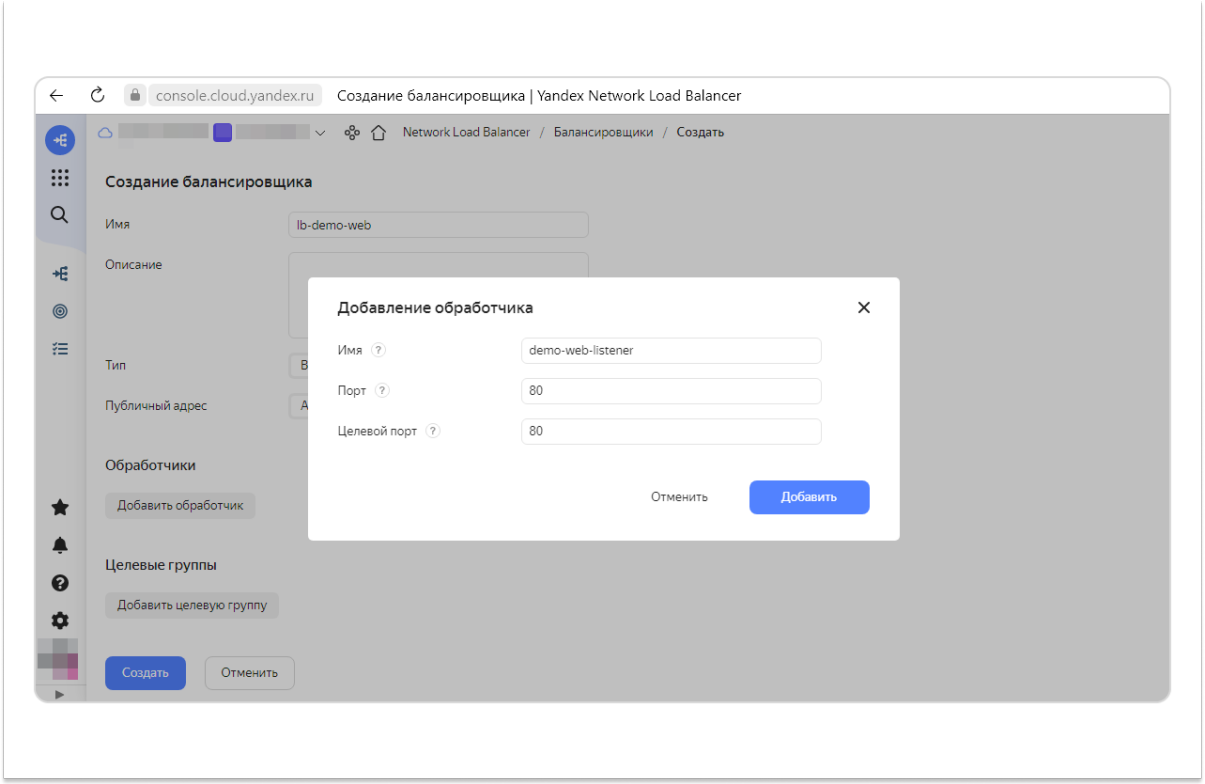
Практическая работа. Создание балансировщика

Итак, у вас есть виртуальные машины. Можно сразу создать и балансировщик, и целевую группу, но мы поступим иначе: сначала создадим целевую группу, затем подключим её к балансировщику.
В консоли управления откройте раздел **Network Load Balancer,** на вкладке **Целевые группы** нажмите кнопку **Создать целевую группу**.
На открывшейся странице введите имя целевой группы (например `demo-web`), выберите обе ВМ, созданные на предыдущем уроке, и нажмите кнопку **Создать**.

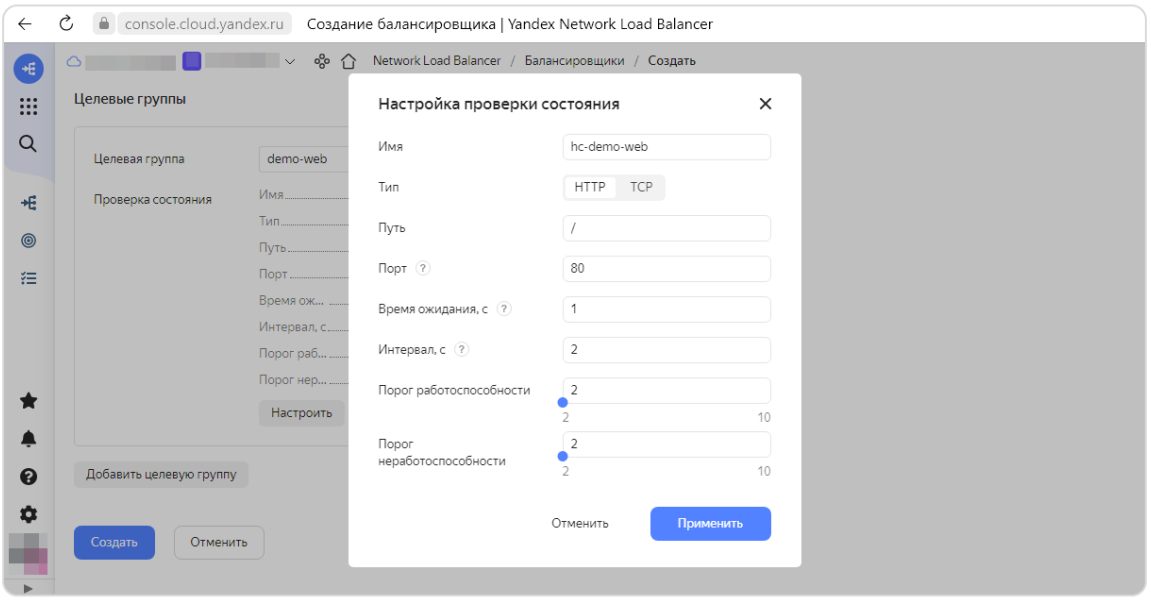


Остаётся создать балансировщик. Для этого сначала создайте обработчик и настройте проверку состояния ресурсов в целевой группе:

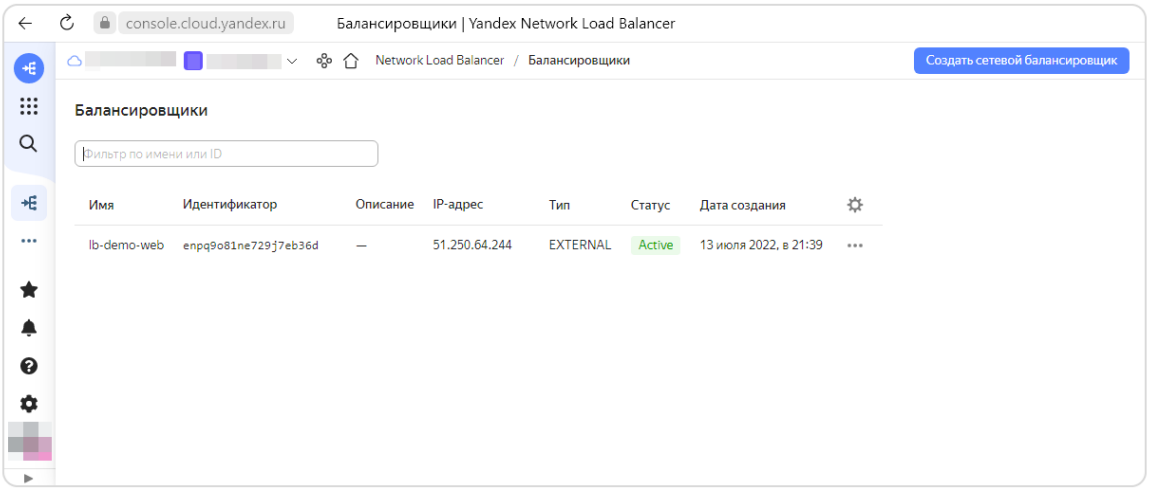
1. На вкладке **Балансировщики** нажмите кнопку **Создать сетевой балансировщик**.
2. Заполните имя балансировщика (например `lb-demo-web`) и нажмите кнопку **Добавить обработчик**.
3. В открывшемся окне введите имя обработчика (например `demo-web-listener`). В качестве портов укажите `80`и нажмите кнопку **Добавить**.



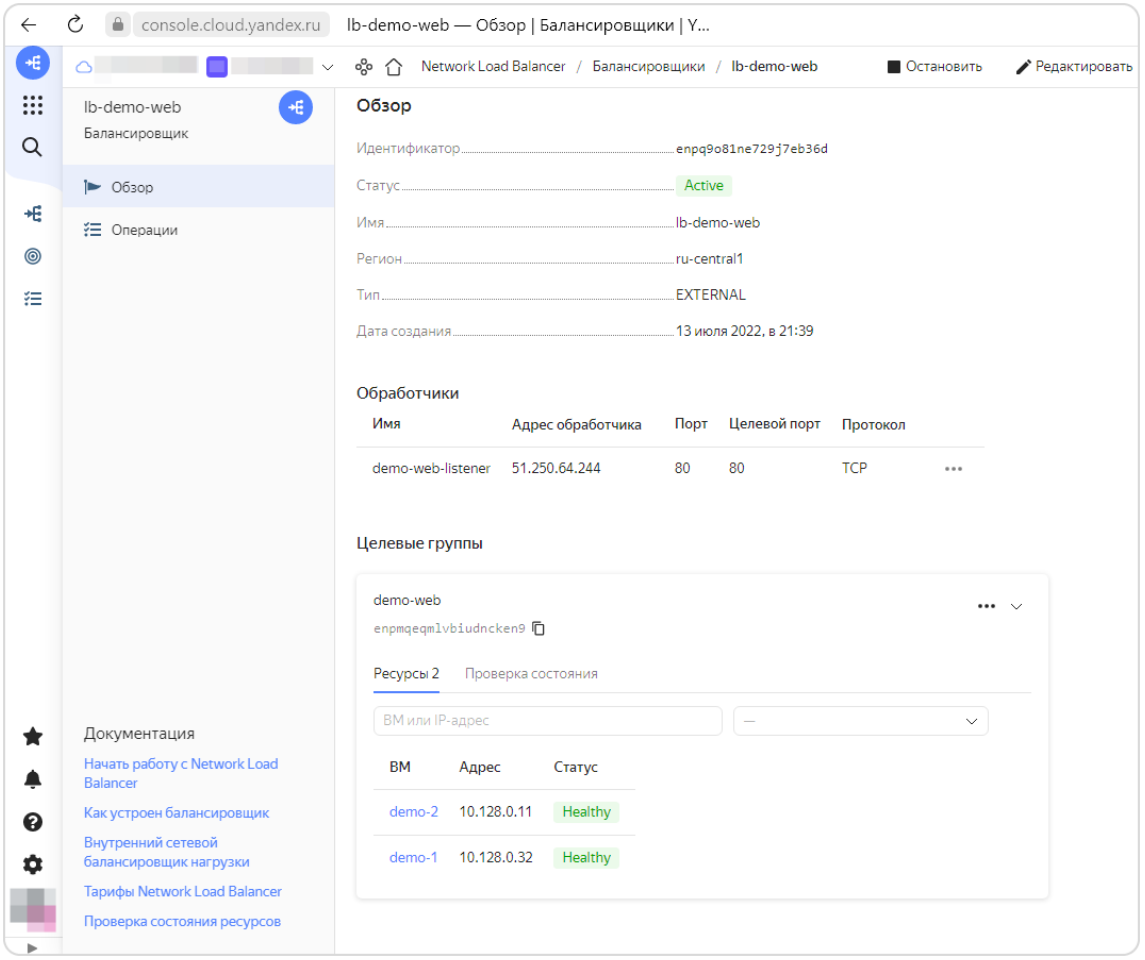
4. После создания обработчика нажмите кнопку **Добавить целевую группу**. Укажите имя проверки состояния (например `hc-demo-web`), тип проверки (`HTTP`), порт (`80`), интервал отправки проверок состояния в секундах, порог работоспособности и порог неработоспособности. Оставьте указанный по умолчанию путь для проверок, используйте значения по умолчанию и для других параметров. Нажмите кнопку **Применить**, а затем кнопку **Создать**.



5. После создания балансировщика проверьте состояние ресурсов: в консоли управления откройте страницу балансировщика и убедитесь, что его статус — **Active**. Значит, балансировщик готов передавать трафик целевым ресурсам.

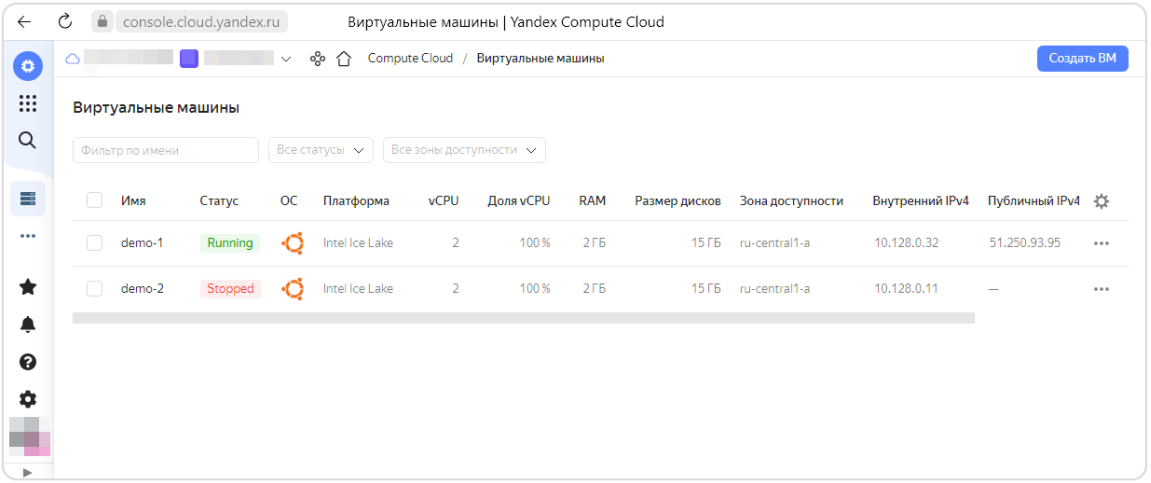


6. Перейдите на страницу балансировщика и посмотрите на блок **Целевые группы**. У запущенных ВМ, готовых принимать трафик, будет статус **Healthy**.

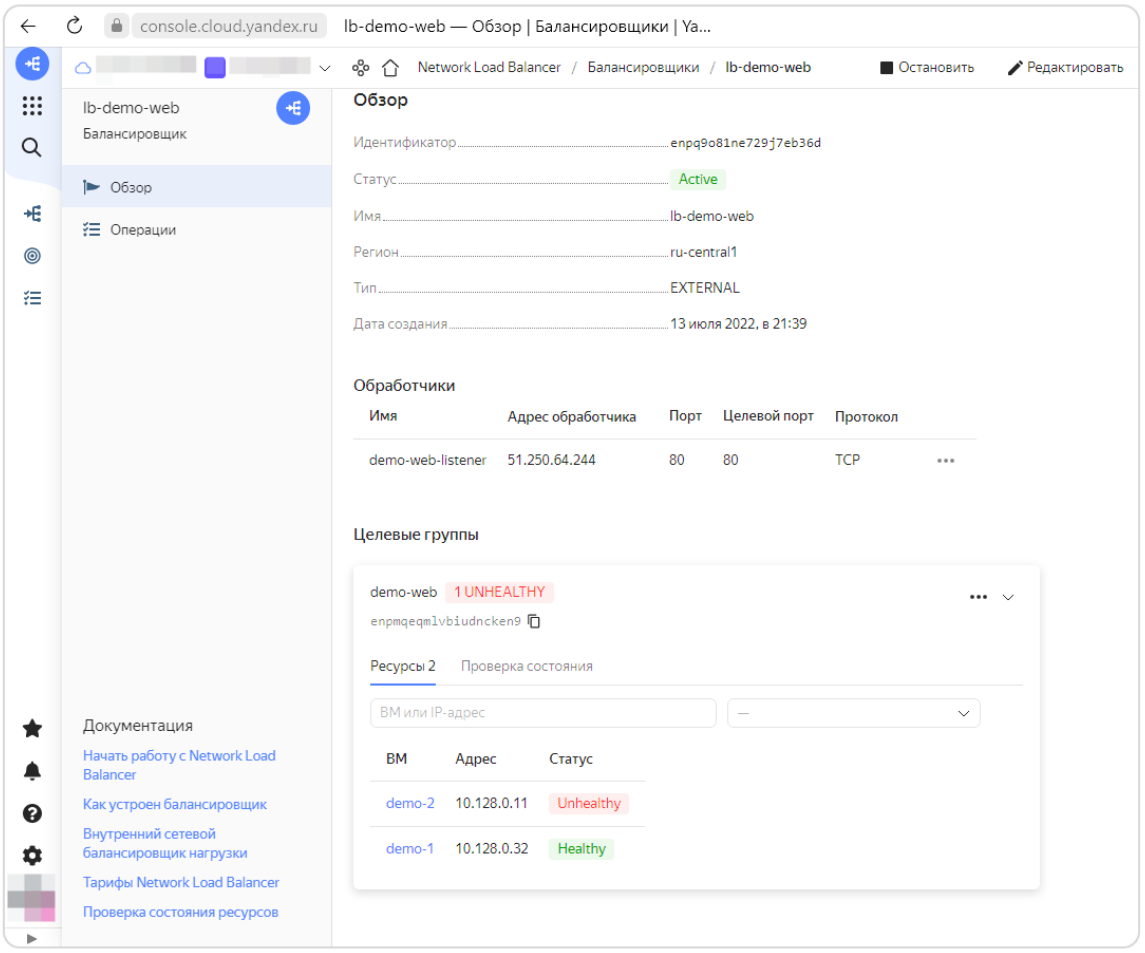


7. Введите внешний IP-адрес балансировщика в адресную строку браузера — и балансировщик перенаправит вас на одну из машин целевой группы. Обратите внимание на имя ВМ, указанное во второй строке веб-страницы.

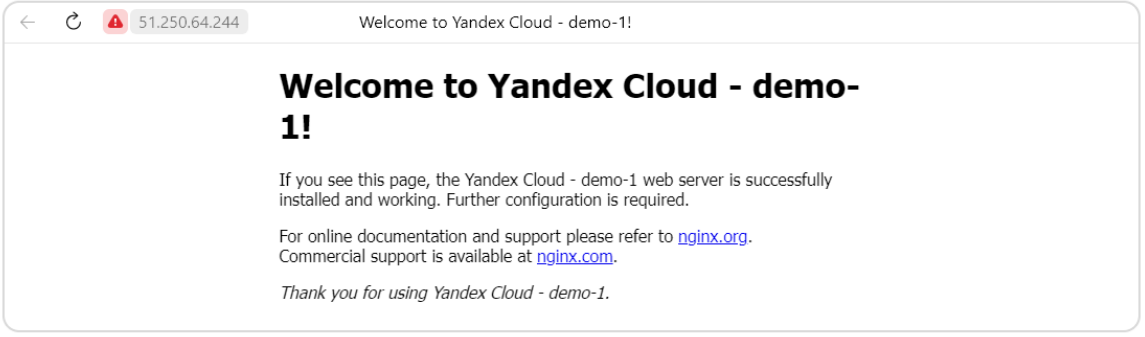
8. Чтобы протестировать отказоустойчивость, в консоли управления перейдите в раздел **Compute Cloud** и остановите одну из ВМ целевой группы.



9. Вернитесь на страницу балансировщика и убедитесь, что статус остановленной ВМ изменился на **Unhealthy**. Это означает, что целевой ресурс группы не прошёл проверку состояния и не готов принимать трафик.



10. Обновите страницу с IP-адресом балансировщика, и вы увидите, что трафик перенаправлен на другую ВМ (изменилось имя ВМ, указанное во второй строке веб-страницы).



После завершения работы не забудьте удалить использованные ресурсы: две ВМ и балансировщик.