Теория.

Если вы разработали отличную программу или сервис — это только часть успеха. Важно, чтобы ваш сервис был надёжным и быстрым, выдерживал пиковые нагрузки и не падал при каждом обновлении. Лет 10–20 назад, чтобы запустить новый сервис иногда было достаточно скопировать пачку скриптов на сервер. Сегодня сервис нужно не только создать, но еще и правильно внедрить и поддерживать. На это повлияло несколько причин:

- Сами сервисы стали сложнее, они часто используют сторонние библиотеки, причем строго определенных версий. Чтобы избежать конфликтов, сервисы стали изолировать друг от друга — помещать в виртуальные машины или контейнеры.
- С появлением методологии Agile, когда развитие приложений стало практически непрерывным, пришлось создавать и новые практики IT-администрирования.
- Рост числа пользователей интернета привел к тому, что практически каждый публичный сервис приходится проектировать как высоконагруженный.
- Многие сервисы стали жизненно важными для миллионов пользователей. Такие сервисы должны оставаться доступными невзирая на пиковые нагрузки и стихийные бедствия.

Как правило, поддержка IT-инфраструктуры не входит в сферу ответственности разработчиков. Раньше инфраструктурой занимались системные администраторы. Сегодня, когда эта область знаний стала гораздо шире, ее принято называть **Operations**. На стыке разработки и администрирования возникла новая специализация — **DevOps**. Область интересов DevOps — выстраивание эффективных процессов внедрения приложений и сервисов, управления серверами и сетями. Специально для DevOps создаются инструменты для поддержки инфраструктуры, в том числе облачной.
Чем выше нагрузка на сервис, тем важнее становится роль DevOps. И поскольку сегодня одни из самых нагруженных сервисов — это поисковики, именно эти компании стали «гуру» в сфере DevOps. Так, Google собрал ключевые знания в книге SRE Book (SRE — Site Reliability Engineering).

Практики DevOps ориентированы на то, чтобы найти баланс между простотой внедрения, скоростью внесения изменений и надёжностью сервисов.
Вот основные направления современного DevOps, которые мы рассмотрим в рамках данного курса:

- **Автоматизация**. Чем меньше операций выполняется вручную, тем меньше вероятность ошибок. Кроме того, автоматизация позволяет внедрить в практику регулярно выполняемые операции (например резервное копирование).
- **Infrastructure as Code** (**IaC**). Суть этого подхода заключается в том, что инфраструктура описывается в текстовых файлах специального формата — как настройки отдельных объектов и взаимосвязи между этими объектами. Такие файлы обычно называют спецификациями или манифестами. С этими файлами можно работать так же, как с программным кодом — отслеживать изменения в них, отдавать на ревью, хранить историю версий и обмениваться.
- **Простота настройки окружения**. Как мы уже говорили выше, современные сервисы принято изолировать друг от друга и упаковывать в контейнеры или отдельные виртуальные машины.
- **Простота масштабирования**. Поскольку нагрузка на сервисы меняется, нужно уметь выделять необходимые ресурсы и сворачивать их, когда они не нужны. Это особенно актуально в облачной инфраструктуре.
- **Отказоустойчивость**. Чтобы сервис работал бесперебойно, нужно правильно накатывать обновления, не допускать отказов и выявлять потенциальные проблемы до того, как их заметят пользователи.

Как пользоваться CLI Yandex Cloud

Что такое интерфейс командной строки Yandex Cloud и для чего он нужен

Предположим, вы пишете скрипт, который автоматизирует рутинные задачи в облаке: выгружает аналитику об аккаунте или пересоздаёт виртуальные машины (ВМ) при обновлении операционной системы. Или, возможно, вы гик и писать команды в консоли вам куда приятнее, чем кликать мышкой в веб-интерфейсе. В обоих случаях вам понравится утилита CLI Yandex Cloud (`yc`).
Эта утилита предоставляет интерфейс командной строки (command-line interface, CLI) для управления Yandex Cloud. На курсе мы станем использовать термины `yc` и CLI как синонимы, потому что других консольных интерфейсов у нас не будет. Утилита `yc` также подходит для взаимодействия с облаком из систем без графического интерфейса, например прямо с ВМ на Linux.
Работа в командной строке — это всегда последовательный запуск команд. В нашем CLI команды выглядят примерно так:

```
yc compute instance list
```

Эта команда выводит список ваших ВМ:

```
+----------------------+------+---------------+---------+-------------+
|          ID          | NAME |    ZONE ID    | STATUS  | INTERNAL IP |
+----------------------+------+---------------+---------+-------------+
| epdnq5kcsgs8tg31c2id | api1 | ru-central1-b | RUNNING | 172.17.0.23 |
| ef3jbvule3hhd05fd8s6 | api2 | ru-central1-c | RUNNING | 172.18.0.5  |
| fhm90d339g7vhu2971f9 | api3 | ru-central1-a | RUNNING | 172.16.0.5  |
+----------------------+------+---------------+---------+-------------+
```

Важно! Из-за особенностей синтаксиса командных оболочек `cmd` и `PowerShell` в ОС Windows некоторые команды `yc` могут выполняться некорректно. Рекомендуем использовать альтернативные варианты:

- сторонняя командная оболочка, например Git BASH;
- подсистемы Windows для Linux (WSL);
- терминал на компьютере или виртуальной машине с ОС Linux / macOS.

CLI и сервисы Yandex Cloud

Команды CLI разделены на группы, каждая из которых соответствует сервису или компоненту Yandex Cloud. Например:

- `yc resource-manager …` — управление облаками и каталогами;
- `yc compute …` — управление ВМ;
- `yc load-balancer …` — управление балансировщиками нагрузки.

Управление кластерами баз данных:

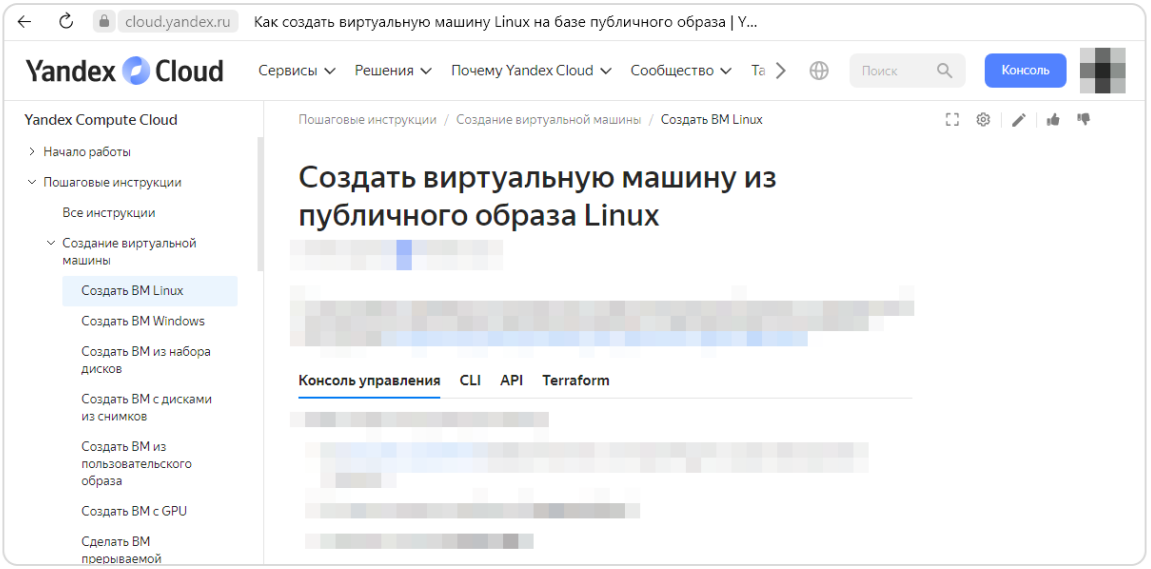
- `yc managed-mysql …` — MySQL;
- `yc managed-postgresql …` — PostgreSQL;
- `yc managed-clickhouse …` — ClickHouse.

Есть ещё небольшая группа служебных команд:

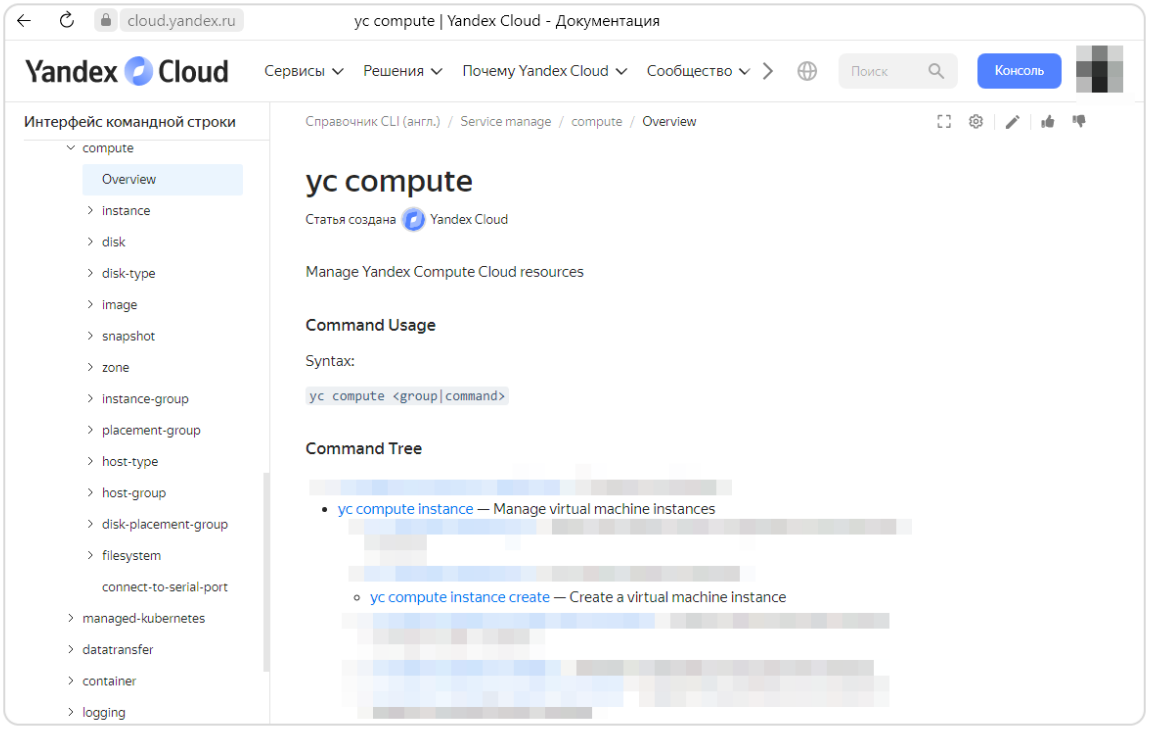
- `yc init` — первоначальная настройка CLI;
- `yc version` — показывает версию CLI;
- `yc help` — выводит описание всех команд или справку о команде.

Документация о CLI

В первом разделе (Виртуальные машины) вы пользовались консолью управления облаком и документацией сервисов Yandex Cloud. Так вот: большинство действий в облаке можно выполнить не только из консоли управления, но и из командной строки — с помощью утилиты `yc`.
На странице документации, где описывается операция, можно посмотреть, как выполнить её в консоли управления и в CLI. Это понадобится, например, если вы захотите автоматизировать действия и написать скрипт.



Иногда проще найти команду в документации о CLI: там есть краткий хорошо структурированный справочник команд.



Кроме того, в документации о CLI описано, как установить `yc` и что делать, если возникают ошибки.

Структура команды CLI

Большинство команд yc строится по следующему принципу:

- Сначала указывается сервис Yandex Cloud (или группа служебных команд). Например, `compute`.
- Затем ресурс, которым управляет сервис. Например, `instance`.
- Потом действие, которое необходимо выполнить над ресурсом. Например, `get`.
- И в конце — параметры и флаги, меняющие поведение команды, а также (если нужно) идентификатор ресурса.

Например:

	Сервис	Ресурс	Действие	Параметры и/или идентификатор
yc	resource-manager	folder	create	--name project-1
yc	compute	instance	list	--folder-name project1
yc	compute	instance	get	epdnq5kcsgs8tg31c2id

В первой команде мы обращаемся к сервису `resource-manager` и просим его создать (`create`) каталог (`folder`), указав имя каталога в параметре `--name`. Создадим каталог с именем `project-1`:

```
yc resource-manager folder create --name project-1
```

Результат:

```
id: b1g4i1fr007e94ut1kc7
cloud_id: b1gttd235imdk2fdud9p
created_at: "2021-03-27T06:26:39.656888Z"
name: project-1
status: ACTIVE
```

Где:

- `id` — идентификатор созданного каталога.
- `cloud_id` — идентификатор облака, в котором размещён каталог.
- `created_at` — дата и время создания каталога.
- `name` — имя каталога.
- `status` — текущий статус каталога.

Идентификаторы и параметры ресурсов уникальны. Идентификаторы и параметры ваших ресурсов будут отличаться от тех, что приведены в

примерах. Чтобы выделить часть команды, которую нужно заменить, мы будем использовать `<плейсхолдеры>`. При подстановке своих значений воспроизводить символы `<>` не нужно.
Некоторые команды принимают сразу несколько параметров. Например, давайте поменяем описание только что созданного каталога:

```
yc resource-manager folder update \
  --name project-1 \
  --description "Learning CLI"
```

Теперь в информации о каталоге выводится и его описание:

```
id: b1g4i1fr007e94ut1kc7
cloud_id: b1gttd235imdk2fdud9p
created_at: "2021-03-27T06:26:39Z"
name: project-1
description: Learning CLI
status: ACTIVE
```

Чтобы команды лучше читались, их разбивают на несколько строк с помощью символа `\` (благодаря ему командная строка поймёт, что вы ввели единую команду, а не три разных строчки).
В командных оболочках `cmd` и `PowerShell` вместо символа `\` для переноса строки используются `^` и `` ` `` соответственно. В наших примерах мы будем использовать `\`.

Использование CLI

Во всех вызовах `yc` есть что-то общее. Например, для любой команды можно указать, в каком облаке и каталоге её выполнить. Для некоторых команд можно указать, в каком формате вы хотите получить данные: в текстовом, JSON или YAML.
Параметры, которые можно использовать с любой командой, называются **глобальными**. Пример такого параметра — название каталога, в котором надо выполнить команды:

```
yc compute instance list --folder-name project-1
```

Некоторые глобальные параметры нельзя указывать вместе. Например, вы можете выбрать либо название каталога (`--folder-name …`), либо его идентификатор (`--folder-id …`), но не оба сразу.

Профили

Обычно для каждого проекта в Yandex Cloud создаётся отдельный каталог, работа ведётся в нём. Если у вас несколько каталогов — вам быстро надоест указывать `--folder-id` или `--folder-name` в каждой команде. Чтобы облегчить вашу жизнь, придуманы **профили**.
Профиль — это набор настроек, таких как имя облака и каталога, которые автоматически применяются к каждой команде. Чтобы переключиться на другой каталог, вы создаёте профиль (или переключаетесь на созданный ранее) и продолжаете работу уже в нём.
Представьте, что вы только начали проект. Создали для него каталог и теперь создаёте профиль `project-1`:

```
yc config profile create project-1
yc config set folder-name project-1 --profile project-1
```

Убедимся, что профиль запомнил имя каталога:

```
yc config list --profile project-1
```

Результат выполнения команды должен быть таким:

```
folder-name: project-1
```

В большинстве случаев при работе с облаком достаточно профиля по умолчанию — `default`. Он создаётся при первом использовании программы. Давайте снова переключимся из профиля `project-1` в профиль по умолчанию и продолжим работу в нём:

```
yc config profile activate default
```

Какие параметры можно задать в профиле, а какие — только для команд, смотрите в разделе Конфигурация CLI документации Yandex Cloud.

Практическая работа. Начало работы в CLI

На этой практической работе мы установим утилиту `yc`, познакомимся с режимом подсказок `--help` и выполним несколько простых команд.

Установка CLI

Первым делом скачайте и установите CLI (если вы ещё не сделали этого).
Теперь настройте программу для работы с вашим аккаунтом и облаком. Для этого запустите командную строку и введите команду:
Скопировать код
```
yc init
```

В консоли появится предложение перейти по ссылке, чтобы программа получила доступ к аккаунту и облаку. Перейдите по ссылке, согласитесь с условиями, затем скопируйте токен и вставьте его в окно командной строки.
Срок жизни токена — **один год**. Через год необходимо получить новый токен и повторить аутентификацию.
Если кто-то узнал ваш токен, отзовите его и запросите новый.
Продолжайте установку: выберите облако, каталог и зону доступности по умолчанию. Следуйте указаниям системы и завершите настройку.
Готово!
При установке автоматически создается профиль `default`, в него записываются выбранные настройки.
Список профилей можно посмотреть командой:
Скопировать код
```
yc config profile list
```

Флаг --help

Пожалуй, самый важный и частый флаг — `--help` (сокращенно `-h`), поэтому поговорим о нём подробнее.
Команд для управления ресурсами облака много. Вы запомните некоторые, но помнить всё невозможно. Пользуйтесь флагом `--help`, когда пишете команду. В этом случае она не выполняется, а в консоль выводится информация о ресурсах, которыми управляет команда, и параметрах запуска.

Важная особенность флага `--help` заключается в том, что его можно применять для каждого уровня этой структуры и писать команду постепенно.
В этой практической работе нам понадобится ВМ в облаке. Если у вас нет ВМ — создайте её в консоли управления, как мы это делали в предыдущих практических работах.
Допустим, вы хотите перезапустить эту ВМ, но не помните полный синтаксис команды.

1. Сначала вызовите описание сервиса Yandex Compute Cloud:

Скопировать код

```
yc compute --help
```

Вы увидите синтаксис команды (раздел Usage), список ресурсов, которыми можно управлять (Groups), а также действий (Commands) и флагов:

```
Usage:
  yc compute <group|command>

Groups:
  instance               Manage virtual machine instances
  disk                   Manage disks
  disk-type              Show available disk types
  image                  Manage images
  snapshot               Manage snapshots
  zone                   Show availability zones
  instance-group         Manage instance groups
  placement-group        Manage placement groups
  host-type              Show available host types
  host-group             Manage host groups
  disk-placement-group   Manage disk placement groups
  filesystem             Manage filesystems

Commands:
  connect-to-serial-port Connect to serial port

Global Flags:
      --profile string       Set the custom configuration file.
      --debug                Debug logging.
      --debug-grpc           Debug gRPC logging. Very verbose, used for
debugging connection problems.
      --no-user-output       Disable printing user intended output to
stderr.
      --retry int            Enable gRPC retries. By default, retries
are enabled with maximum 5 attempts.
```

```
                              Pass 0 to disable retries. Pass any
negative value for infinite retries.
                              Even infinite retries are capped with 2
minutes timeout.
      --cloud-id string       Set the ID of the cloud to use.
      --folder-id string      Set the ID of the folder to use.
      --folder-name string    Set the name of the folder to use (will be
resolved to id).
      --endpoint string       Set the Cloud API endpoint (host:port).
      --token string          Set the OAuth token to use.
      --format string         Set the output format: text (default),
yaml, json, json-rest.
  -h, --help                  Display help for the command.
```

2. Вам нужна ВМ — ресурс `instance`. Теперь узнайте, как получить список активных ВМ и какая команда отвечает за перезапуск:

Скопировать код

```
yc compute instance --help
```

Вы увидите список возможных действий (команд) над ВМ:

```
Manage virtual machine instances

Usage:
  yc compute instance <command>

Aliases:
  instances

Commands:
  get                       Show information about the specified virtual
machine instance
  list                      List virtual machine instances
  create                    Create a virtual machine instance
  create-with-container     Create a virtual machine instance running
Docker container
  update                    Update the specified virtual machine
instance
  update-container          Update the specified virtual machine
instance running Docker container
  add-metadata              Add or update metadata for the specified
virtual machine instance
  remove-metadata           Remove keys from metadata for the specified
virtual machine instance
  add-labels                Add labels to specified virtual machine
instance
```

```
  remove-labels            Remove labels from specified virtual machine
instance
  delete                   Delete the specified virtual machine
instance
  get-serial-port-output   Return the serial port output of the
specified virtual machine instance
  stop                     Stop the specified virtual machine instance
  start                    Start the specified virtual machine instance
  restart                  Restart the specified virtual machine
instance
  attach-disk              Attach existing disk to the the specified
virtual machine instance
  attach-new-disk          Attach new disk to the the specified virtual
machine instance
  detach-disk              Detach disk from the the specified virtual
machine instance
  attach-filesystem        Attach existing filesystem to the specified
virtual machine instance
  detach-filesystem        Detach filesystem from the specified virtual
machine instance
  update-network-interface Update the specified network interface of
virtual machine instance
  add-one-to-one-nat       Add one-to-one NAT to the the specified
network interface of virtual machine instance
  remove-one-to-one-nat    Remove one-to-one NAT to the the specified
network interface of virtual machine instance
  list-operations          List operations for the specified instance

Global Flags:
      --profile string       Set the custom configuration file.
      --debug                Debug logging.
      --debug-grpc           Debug gRPC logging. Very verbose, used for
debugging connection problems.
      --no-user-output       Disable printing user intended output to
stderr.
      --retry int            Enable gRPC retries. By default, retries
are enabled with maximum 5 attempts.
                             Pass 0 to disable retries. Pass any
negative value for infinite retries.
                             Even infinite retries are capped with 2
minutes timeout.
      --cloud-id string      Set the ID of the cloud to use.
      --folder-id string     Set the ID of the folder to use.
      --folder-name string   Set the name of the folder to use (will be
resolved to id).
      --endpoint string      Set the Cloud API endpoint (host:port).
      --token string         Set the OAuth token to use.
      --format string        Set the output format: text (default),
yaml, json, json-rest.
  -h, --help                 Display help for the command.
```

3. Сначала получите список ВМ, это команда `list`:

Скопировать код
```
yc compute instance list
```

Результатом выполнения команды будет список машин в том каталоге, где вы работаете, с именами и идентификаторами:

```
+----------------------+-------------+---------------+---------+-------
---------+-------------+
|          ID          |    NAME     |    ZONE ID    | STATUS  |
EXTERNAL IP   | INTERNAL IP |
+----------------------+-------------+---------------+---------+-------
---------+-------------+
| fhm2p20bifmg3k3voda7 | my-instance | ru-central1-a | RUNNING |
XXX.XXX.XXX.XX | XX.XXX.X.XX |
+----------------------+-------------+---------------+---------+-------

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

4. На шаге 2 вы нашли команду перезапуска — `restart`, теперь можно посмотреть её синтаксис:

Скопировать код
```
yc compute instance restart --help
```

Результат выполнения будет таким:

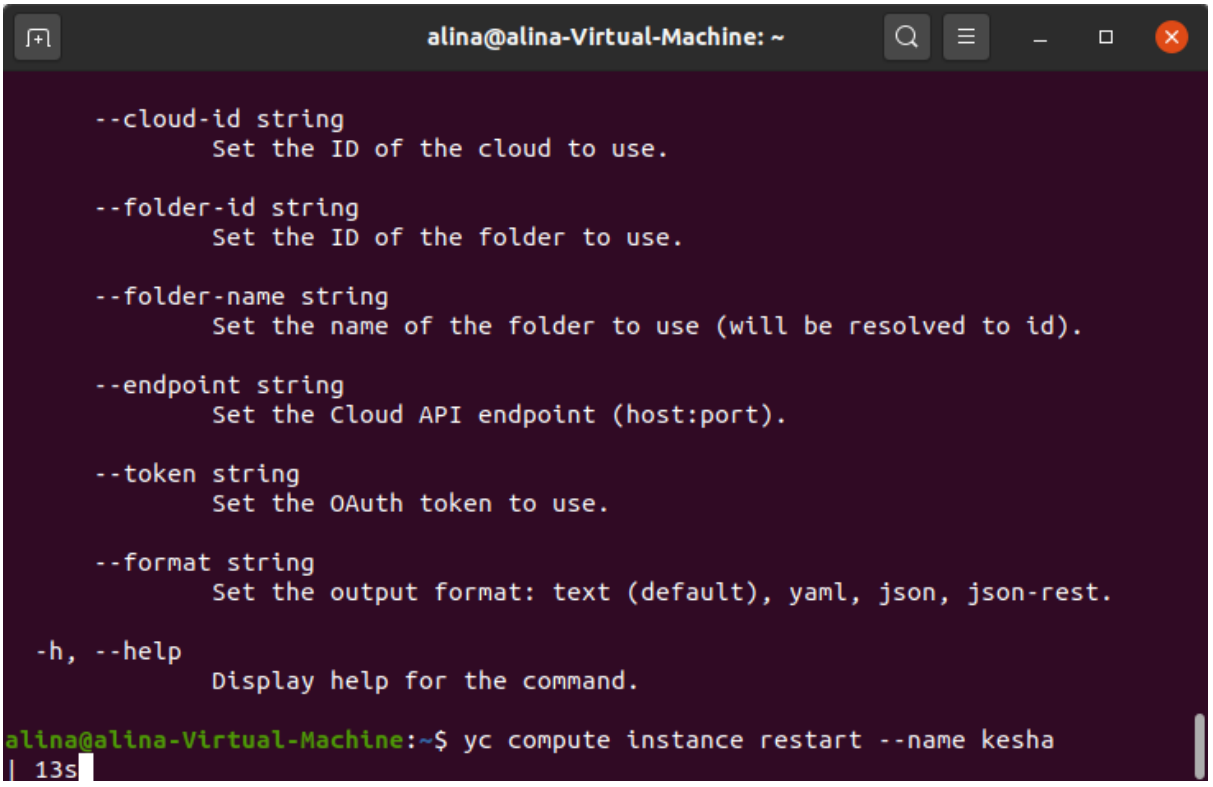
```
Restart the specified virtual machine instance

Usage:
  yc compute instance restart <INSTANCE-NAME>|<INSTANCE-ID> [Global
Flags...]

Flags:
      --id string     Instance id.
      --name string   Instance name.
      --async         Display information about the operation in
progress, without waiting for the operation to complete.

Global Flags:
      --profile string       Set the custom configuration file.
      --debug                Debug logging.
      --debug-grpc           Debug gRPC logging. Very verbose, used for
debugging connection problems.
      --no-user-output       Disable printing user intended output to
stderr.
```

```
      --retry int             Enable gRPC retries. By default, retries
are enabled with maximum 5 attempts.
                              Pass 0 to disable retries. Pass any
negative value for infinite retries.
                              Even infinite retries are capped with 2
minutes timeout.
      --cloud-id string       Set the ID of the cloud to use.
      --folder-id string      Set the ID of the folder to use.
      --folder-name string    Set the name of the folder to use (will be
resolved to id).
      --token string          Set the OAuth token to use.
      --format string         Set the output format: text (default),
yaml, json, json-rest.
  -h, --help                  Display help for the command.
```

5. Наконец, вы получили полный синтаксис команды перезапуска. Примените её к ВМ:

Скопировать код

```
yc compute instance restart --name <имя_ВМ>
```



По такому принципу вы можете сформировать в консоли любую команду.

Итак, подведём итоги. Чтобы уточнить синтаксис или порядок действий при работе с CLI, выберите один из трёх путей:

- найдите нужное действие на странице сервиса Yandex Cloud в документации и там переключитесь на вкладку CLI;
- найти команду в справочнике о `yc`;
- воспользуйтесь флагом `--help`, шаг за шагом уточняя возможности и синтаксис команды.

Синхронный и асинхронный режимы работы

Некоторые команды выполняются очень быстро. Например, создание каталога или просмотр настроек профиля. Такие команды можно выполнять подряд, одну за другой — без задержек.
Если при выполнении команды ресурс изменяет состояние, создается **операция**. Примеры операций — перезапуск ВМ после обновления или резервное копирование базы данных.
Если каждая следующая команда ждёт завершения предыдущей операции, такой режим выполнения называется **синхронным**. Когда какая-то операция в синхронном режиме выполняется долго, CLI выводит в консоли точки и время с начала операции, чтобы показать, что процесс не завис:

```
...1s...6s...12s...17s
```

Операция может выполняться довольно долго, а ожидание — затормозить процесс. В таких случаях важно оценить, нужен ли результат операции для выполнения следующих команд. Если нет — можно не ждать её завершения и сразу же переходить к следующей команде. Этот режим называется **асинхронным**.
Чтобы выполнить команду асинхронно, используйте флаг `--async`.
Перезапустите одну из ранее созданных ВМ в асинхронном режиме (в команде укажите имя этой ВМ):
Скопировать код

```
yc compute instance restart --name <имя_ВМ> --async
```

В ответ на асинхронный вызов CLI выводит идентификатор операции (в поле `id`) и информацию о ней:

```
id: fhm5k7iq03rm2s7enhdk
description: Restart instance
created_at: "2021-03-27T08:32:47.562595036Z"
created_by: aje9cb7k03512mrugcee
modified_at: "2021-03-27T08:32:47.562595036Z"
metadata:
  '@type':
type.googleapis.com/yandex.cloud.compute.v1.RestartInstanceMetadata
```

```
  instance_id: fhm2p20bifmg3k3voda7
```

С помощью идентификатора операции вы можете проверить результаты её выполнения:

Скопировать код

```
yc operation get <идентификатор_операции>
```

Когда операция завершится, вы увидите результат:

```
id: fhm5k7iq03rm2s7enhdk
...
done: true
```

Практическая работа. Создание виртуальных машин с помощью CLI

Создание ВМ — одна из самых сложных команд CLI, потому что в ней очень много параметров. Давайте потренируемся работать с ней. Но сначала подготовим окружение. Мы будем работать в каталоге по умолчанию. Создадим в нём сеть, в ней — три подсети, а затем по ВМ в каждой подсети.

1. Создайте сеть `my-network` в Virtual Private Cloud. Эта команда относится к группе vpc.

Скопировать код
```
yc vpc network create --name my-network
```

Выполнение операции займёт какое-то время. В итоге сеть появится в каталоге, который вы выбрали в предыдущей практической работе.

2. Теперь создадим три подсети в разных зонах доступности (`ru-central1-a`, `ru-central1-b`, `ru-central1-c`). Чтобы проще было выполнять задания дальше, назовите их `my-subnet-1`, `my-subnet-2` и `my-subnet-3`. А пространства IP-адресов для подсетей укажите, соответственно, как `192.168.1.0/24`, `192.168.2.0/24` и `192.168.3.0/24`.

Не забудьте указать, что вы создаёте подсети в новой сети, созданной на предыдущем шаге. Иначе они появятся в сети по умолчанию, которая есть в каждом каталоге.
Вот так выглядит команда создания подсети в зоне доступности `ru-central1-a`:
Скопировать код
```
yc vpc subnet create \
  --name my-subnet-1 \
  --zone ru-central1-a \
  --range 192.168.1.0/24 \
  --network-name my-network
```

Где:

- `name` — имя подсети.
- `zone` — зона доступности.
- `range` — адресное пространство подсети.
- `network-name` — имя сети, в которой создаётся подсеть.

Создайте две другие подсети сами.

3. Осталось создать три ВМ в нужных зонах доступности и привязать к ним подсети.

Пусть машины работают под управлением ОС Ubuntu 20.04 LTS, имеют диски объёмом 30 Гб, 4 Гб оперативной памяти и два виртуальных процессорных ядра.
ВМ могут создаваться долго, поэтому запускайте команды в асинхронном режиме.
Пример команды для первой ВМ:
Скопировать код
```
yc compute instance create \
  --name my-instance-1 \
```

```
  --hostname my-instance-1 \
  --zone ru-central1-a \
  --create-boot-disk
image-family=ubuntu-2004-lts,size=30,type=network-nvme \
  --image-folder-id standard-images \
  --memory 4 --cores 2 --core-fraction 100 \
  --network-interface subnet-name=my-subnet-1,nat-ip-version=ipv4 \
  --async
```

Посмотрите внимательно на все параметры. Подумайте, какой из них за что отвечает. Создайте две другие машины сами.

После выполнения каждой команды благодаря флагу `--async` вы получите идентификатор операции и ее описание в виде:

```
id: c9q9v4bsn1hs9api4b13
description: Create instance
created_at: "2021-03-01T03:23:00.079888Z"
created_by: aje8s4vd4pp7cduq2o4k
modified_at: "2022-07-29T09:14:53.567744154Z"
metadata:
  '@type':
type.googleapis.com/yandex.cloud.compute.v1.CreateInstanceMetadata
  instance_id: fhmepiaciq5l9slqid3k
```

4. Проследите за статусом одной из операций, используя ее идентификатор.

Скопировать код
```
yc operation get <идентификатор_операции>
```

5. Дождитесь, пока операция завершится. Используйте для этого команду `wait`.

Скопировать код
```
yc operation wait <идентификатор_операции>
```

6. Убедитесь, что ВМ созданы. Для этого выведите их список.

Скопировать код
```
yc compute instance list
```

По умолчанию список выдается в виде таблицы:

```
+----------------------+---------------+---------------+---------+-----
----------+--------------+
|          ID          |     NAME      |    ZONE ID    | STATUS  |
EXTERNAL IP  | INTERNAL IP  |
+----------------------+---------------+---------------+---------+-----
----------+--------------+
| ef34r4fs8dsva3qtsivs | my-instance-3 | ru-central1-c | RUNNING |
51.250.44.130 | 192.168.3.34 |
```

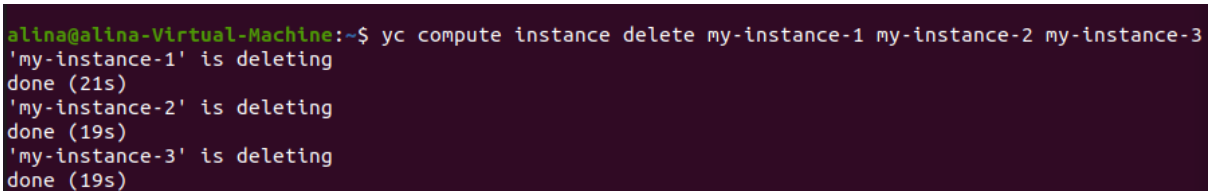
```
| epdj7u79isrolup3vfo8 | my-instance-2 | ru-central1-b | RUNNING |
158.160.6.249 | 192.168.2.28 |
| fhmepiaciq5l9slqid3k | my-instance-1 | ru-central1-a | RUNNING |
62.84.126.39  | 192.168.1.30 |
+----------------------+--------------+--------------+---------+-----

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

Список можно вывести в формате YAML или JSON (эта возможность пригодится вам на следующих уроках):
Скопировать код
```
yc compute instance list --format json
```

Список в формате JSON содержит больше информации, чем таблица.

7. Чтобы избежать ненужных расходов, удалите три созданные ВМ (в следующих практических работах они не понадобятся).

Скопировать код
```
yc compute instance delete my-instance-1 my-instance-2 my-instance-3
```



Практическая работа. Использование файлов спецификаций

В этой практической работе вы создадите, обновите и удалите группу ВМ.
Вы уже убедились, что создать даже одну ВМ через `yc` непросто: нужно установить много разных параметров. Создание группы ВМ требует ещё больше параметров. Чтобы не указывать их все в командной строке, конфигурацию описывают в файле, который используют при создании группы. Такой файл называется **спецификацией**. Использование спецификаций — это первый шаг в освоении подхода **Infrastructure as Code** (**IaC**), который мы будем применять на следующих уроках.
Спецификации пишутся в разных форматах. Для группы ВМ используется язык YAML. Если вы не знакомы с ним — ничего страшного. В документации есть шаблоны спецификаций, и на первых порах вам будет достаточно лишь немного их изменять. Ниже мы разберём, как составлять спецификации.

Часть 1. Создание Instance Group

1. Для разворачивания группы ВМ потребуется сеть. Если сети ещё нет, создайте её.
   Посмотрите информацию об имеющихся сетях.

   Скопировать код
```
yc vpc network list
```

2. Сохраните идентификатор сети, он понадобиться нам в дальнейшем.
3. По умолчанию все операции в Instance Groups выполняются от имени сервисного аккаунта с ролью `editor` на каталог. Если сервисного аккаунта нет, то тоже создайте его и назначьте эту роль.
   Посмотрите информацию об имеющихся сервисных аккаунтах.

   Скопировать код
```
yc iam service-account list
```

4. Сохраните идентификатор сервисного аккаунта, он понадобится нам в дальнейшем.
5. Для создания группы необходимо подготовить её спецификацию. Создайте в любом текстовом редакторе файл с расширением `yaml`, например `specification.yaml`.

Обратите внимание: в формате YAML важны отступы слева. Даже если текст правильный, но отступы не соблюдены, при выполнении спецификации возникнут ошибки.

4. Сначала внесите информацию о группе. Пусть группа называется `my-group`. Укажите идентификатор сервисного аккаунта, от имени которого будете работать (см. шаг 2).

Идентификаторы ресурсов уникальны. Копируя команды из текста урока, не забывайте подставлять свои идентификаторы.
Скопировать код
```
name: my-group
service_account_id: <идентификатор_сервисного_аккаунта>
```

5. Наша группа будет содержать три одинаковые ВМ. Машины создадим из публичного образа Ubuntu 18.04 LTS (возьмём не последнюю версию, чтобы потренироваться обновлять ВМ). Узнайте идентификатор образа с помощью команды:

Скопировать код

```
yc compute image list --folder-id standard-images
```

6. В столбце `FAMILY` найдите `ubuntu-1804-lts`, в столбце `ID` будет указан нужный идентификатор.

7. Опишите в спецификации ВМ. Это раздел `instance_template`.
   Пусть каждая машина использует платформу Intel Broadwell (посмотрите поддерживаемые платформы в документации Yandex Compute Cloud), имеет 2 Гб оперативной памяти и два процессорных ядра.

Скопировать код

```
instance_template:
     platform_id: standard-v1
     resources_spec:
         memory: 2g
         cores: 2
```

8. Добавьте описание загрузочного диска. Он будет использоваться на чтение и запись (режим `READ_WRITE`). Укажите идентификатор образа, который получили на шаге 5. Выделите сетевой HDD объёмом 32 Гб.

Скопировать код

```
    boot_disk_spec:
        mode: READ_WRITE
        disk_spec:
            image_id: <идентификатор_образа>
            type_id: network-hdd
            size: 32g
```

9. Теперь опишите сеть: идентификатор сети из каталога по умолчанию (см. шаг 1). Задайте публичный IP-адрес, чтобы к ВМ можно было обращаться извне.

Скопировать код

```
    network_interface_specs:
         - network_id: <идентификатор_сети>
         primary_v4_address_spec: { one_to_one_nat_spec: { ip_version:
IPV4 }}
```

10. В политике планирования укажите, что машина не прерываемая.

Скопировать код

```
scheduling_policy:
      preemptible: false
```

11. В политике развертывания (раздел `deploy policy`) укажите, что в каждый момент времени может быть неработоспособной только одна машина, не больше. Запретите увеличивать число ВМ, т. е. создавать больше трех машин одновременно. Мы чуть подробнее разберём эти настройки, когда будем обновлять ВМ в группе.

Скопировать код

```
    deploy_policy:
         max_unavailable: 1
         max_expansion: 0
```

12. Мы создаем группу фиксированного размера из трёх ВМ. Укажите это в политике масштабирования (раздел `scale_policy`):

Скопировать код

```
scale_policy:
    fixed_scale:
        size: 3
```

13. Наконец, в политике распределения машин по зонам (раздел `allocation_policy`) укажите, что будет использоваться зона `ru-central1-a`. Мы делаем это для простоты. Лучше распределять ВМ группы по разным зонам доступности — это позволит пережить краткие сбои или выход зоны из строя.

    Скопировать код

```
allocation_policy:
     zones:
         - zone_id: ru-central1-a
```

14. Для балансировщика нагрузки (раздел `load_balancer_spec`) укажите целевую группу, к которой он будет привязан (это мы рассмотрим чуть ниже).

Скопировать код

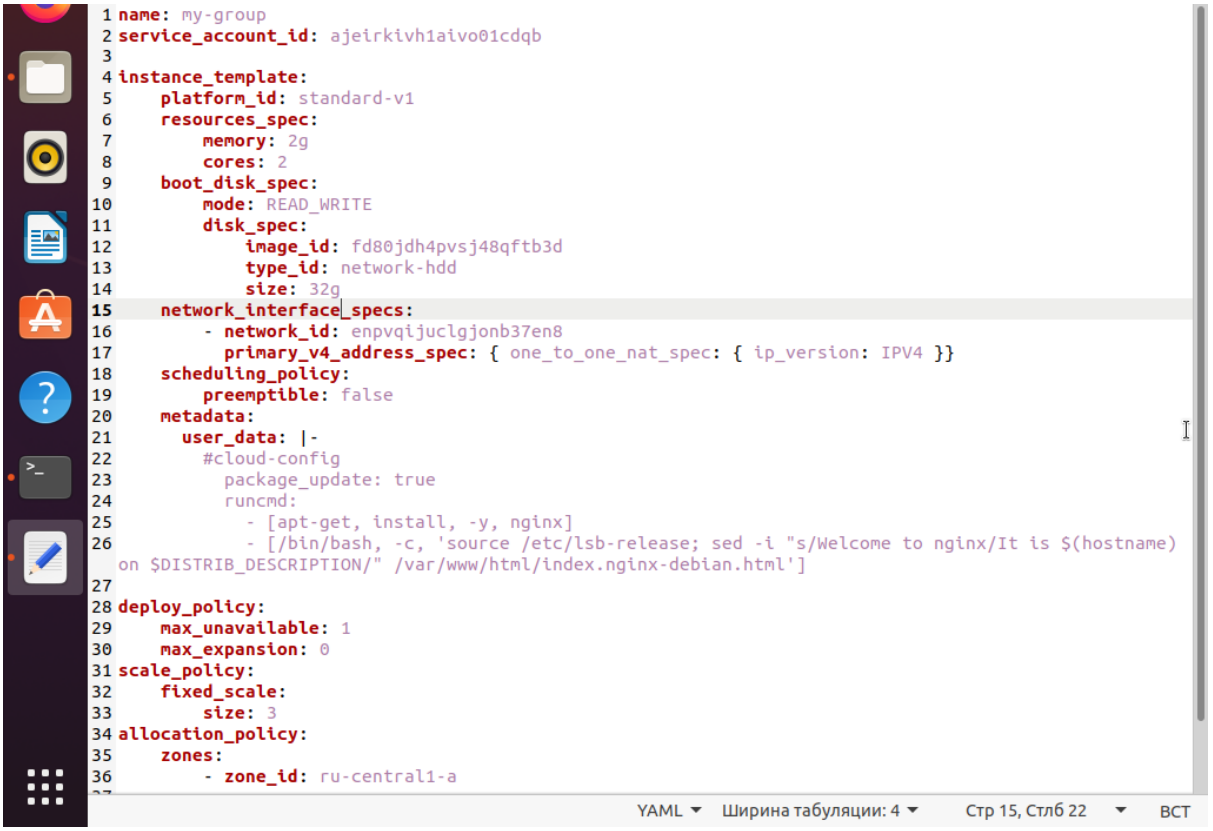
```
load_balancer_spec:
     target_group_spec:
         name: my-target-group
```

15. Нашей спецификации уже достаточно, чтобы создать группу ВМ. Но на эти машины не будет установлено никакого ПО, только операционная система из публичного образа. Если не менять конфигурацию, то после создания ВМ вам придётся устанавливать программы вручную.
    Чтобы сэкономить время и сократить число ошибок, давайте максимально автоматизируем создание ВМ, включая установку ПО. Для этого добавим в конфигурацию машины секцию, где будут вызываться команды установки программ. В этой же секции можно описать создание пользователей, но мы этого делать не будем, так как заходить на ВМ не планируем.
    Установим на машины веб-сервер NGINX и на веб-странице `index.nginx-debian.html`, которая создается по умолчанию и выводит приветственное сообщение «Welcome to nginx», заменим слово nginx идентификатором активной ВМ и версией ОС. Поскольку мы подключим балансировщик нагрузки, идентификатор активной ВМ будет различаться для разных пользователей. Это и позволит нам убедиться в том, что балансировщик работает.
    Для установки ПО используйте `cloud-init` — пакет, выполняющий команды на ВМ при первом запуске. Команды опишите в блоке конфигурации `#cloud-config`.
    Примеры команд смотрите в документации cloud-init.
    Содержимое `#cloud-config` описывается в разделе `instance_template` в секции `metadata`:

Скопировать код

```
metadata:
   user-data: |-
     #cloud-config
       package_update: true
       runcmd:
         - [apt-get, install, -y, nginx ]
              - [/bin/bash, -c, 'source /etc/lsb-release; sed -i
         "s/Welcome to nginx/It is $(hostname) on
         $DISTRIB_DESCRIPTION/"
         /var/www/html/index.nginx-debian.html']
```

16. Спецификация готова. Вот ее полный текст. Помните, что в формате YAML важно соблюдать отступы слева.



```
name: my-group
service_account_id: ajeirkivh1aivo01cdqb

instance_template:
    platform_id: standard-v1
    resources_spec:
        memory: 2g
        cores: 2
    boot_disk_spec:
        mode: READ_WRITE
        disk_spec:
            image_id: fd80jdh4pvsj48qftb3d
            type_id: network-hdd
            size: 32g
    network_interface_specs:
        - network_id: enpvqijuclgjonb37en8
          primary_v4_address_spec: { one_to_one_nat_spec: { ip_version: IPV4 }}
    scheduling_policy:
        preemptible: false
    metadata:
      user_data: |-
        #cloud-config
          package_update: true
          runcmd:
            - [apt-get, install, -y, nginx]
            - [/bin/bash, -c, 'source /etc/lsb-release; sed -i "s/Welcome to nginx/It is $(hostname)
on $DISTRIB_DESCRIPTION/" /var/www/html/index.nginx-debian.html']

deploy_policy:
    max_unavailable: 1
    max_expansion: 0
scale_policy:
    fixed_scale:
        size: 3
allocation_policy:
    zones:
        - zone_id: ru-central1-a
```

YAML ▾ Ширина табуляции: 4 ▾ Стр 15, Стлб 22 ▾ ВСТ

```
deploy_policy:
    max_unavailable: 1
    max_expansion: 0
scale_policy:
    fixed_scale:
        size: 3
allocation_policy:
    zones:
        - zone_id: ru-central1-a

load_balancer_spec:
    target_group_spec:
        name: my-target-group
```

YAML ▾ Ширина табуляции: 4 ▾ Стр 28, Стлб 15 ▾ ВСТ

15. Теперь создайте группу ВМ по подготовленной спецификации:

Скопировать код
```
yc compute instance-group create --file
```
```
<путь_к_файлу_specification.yaml>
```

Для тренировки можете вызвать эту команду в асинхронном режиме, а затем проверить её статус и дождаться завершения.

16. Убедитесь, что группа создана, в веб-консоли или выведя список групп с помощью `yc`.

Скопировать код
```
yc compute instance-group list
```

17. В списке вы должны увидеть свою группу машин `my-group`:

```
+----------------------+------------+------+
|          ID          |    NAME    | SIZE |
+----------------------+------------+------+
| amc65sbgfqeqf00m02sc | my-group   |    3
|+---------------------+------------+------+
```

Часть 2. Балансировщик

17. Создайте балансировщик `my-load-balancer`. Посмотрите, какие параметры должны быть у соответствующей команды:

Скопировать код
```
yc load-balancer network-load-balancer create --help
```

18. В выводе справки обратите внимание, что при создании балансировщика можно сразу создать и обработчик входящего трафика (параметр `--listener`).
Формат параметра `--listener` достаточно хитрый: в нём можно указать сразу несколько подпараметров через запятую:

```
...
--listener name=my-listener,external-ip-version=ipv4,port=80
...
```

19. Помимо имени обработчика, здесь указывается версия IP-протокола и порт, на котором балансировщик будет принимать трафик.

Скопировать код
```
yc load-balancer network-load-balancer create \
```

```
  --region-id ru-central1 \
  --name my-load-balancer \
  --listener name=my-listener,external-ip-version=ipv4,port=80
```

```
alina@alina-Virtual-Machine:~$ yc load-balancer network-load-balancer create --region-
id ru-central1 --name my-load-balancer --listener name=my-listener,external-ip-version
=ipv4,port=80
done (1s)
id: enpobk95v797qhu5r7rq
folder_id: b1grt4h2lvaalduceqa1
created_at: "2023-02-15T20:16:12Z"
name: my-load-balancer
region_id: ru-central1
status: INACTIVE
type: EXTERNAL
listeners:
  - name: my-listener
    address: 51.250.81.104
    port: "80"
    protocol: TCP
    target_port: "80"
    ip_version: IPV4
```

Затем подключите к балансировщику целевую группу (команда `attach-target-group`). Вам понадобится идентификатор целевой группы. Чтобы узнать его, запросите с помощью `yc` список доступных целевых групп и выберите ту, которую вы указали в спецификации `specification.yaml`.

Скопировать код

```
yc load-balancer target-group list
```

Целевая группа также подключается с помощью нескольких подпараметров, которые соответствуют настройкам в консоли управления. Для целевой группы укажите такие параметры:

- `target-group-id` — идентификатор группы;
- `healthcheck-name`, `healthcheck-interval`, `healthcheck-timeout`, `healthcheck-unhealthythreshold`, `healthcheck-healthythreshold`, `healthcheck-http-port` — параметры проверки состояния (см. документацию). Эти параметры аналогичны тем, что задаются в консоли управления при создании балансировщика.

Укажите 80-й порт, на котором запущен NGINX.

Скопировать код

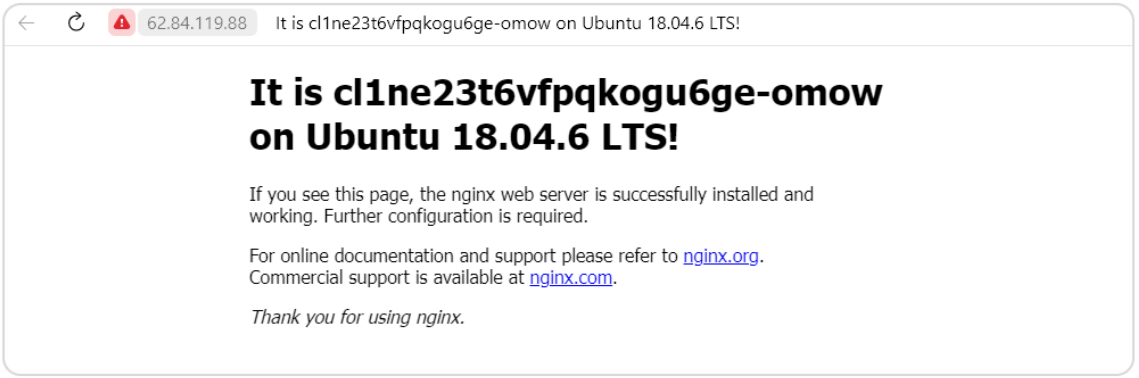
```
yc load-balancer network-load-balancer attach-target-group
my-load-balancer \
  --target-group target-group-id=<идентификатор целевой
группы>,healthcheck-name=test-health-check,healthcheck-interval=2s,heal
thcheck-timeout=1s,healthcheck-unhealthythreshold=2,healthcheck-healthy
threshold=2,healthcheck-http-port=80
```

Часть 3. Доступ к машинам группы

19. Проверьте состояние машин группы. Для этого запросите список машин и дождитесь статуса `HEALTHY`.

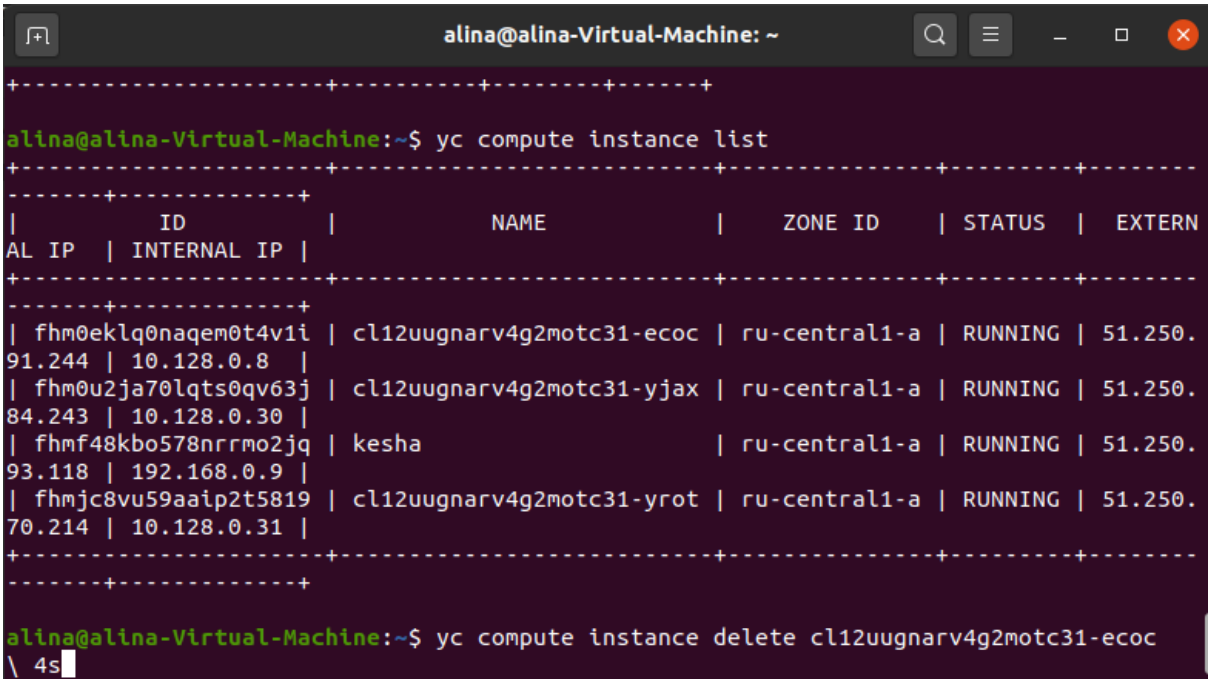
Скопировать код
```
yc load-balancer network-load-balancer target-states my-load-balancer \
--target-group-id <идентификатор_целевой_группы>
```

20. Теперь откройте в браузере страницу балансировщика. IP-адрес балансировщика вы можете узнать с помощью консоли управления или `yc`.
    На странице вы увидите приветственное сообщение и в нём идентификатор одной из машин.



Часть 4. Удаление машины из группы

На приветственной странице балансировщика посмотрите имя активной машины и попробуйте удалить ее. Убедитесь, что приветственная страница остаётся доступна всё время: балансировщик переключит трафик на другую машину группы. А Yandex Cloud тем временем пересоздаст удалённую машину.
Скопировать код
```
yc compute instance delete <имя_ВМ>
```



Часть 6. Удаление Instance Group

Теперь удалите группу и балансировщик командами `yc`.
Скопировать код

```
yc compute instance-group delete --name my-group
yc load-balancer network-load-balancer delete --name my-load-balancer
```

Кстати, ключевой параметр `--name` можно и не писать. Достаточно указать имя группы или балансировщика.
Убедитесь, что группы и балансировщика больше нет, через консоль управления или с помощью `yc`.