Лабораторная работа #6

Образ виртуальной машины для выполнения заданий: https://app.vagrantup.com/ubuntu/boxes/jammy64

Инструкция по импорту образа: https://howtoprogram.xyz/2016/07/06/add-vagrant-box-local-remote/

Оформление результатов лабораторной работы

Необходимо выполнить задание по инструкции и написать отчет о проделанной работе. Каждый этап выполнения необходимо сопроводить наглядными скриншотами экрана и краткими пояснениями.

- Оформление произвольное.
- Формат файла pdf.
- Размер файла до 50 МБ.

Задание

Доп.информация: https://docs.docker.com/engine/swarm/swarm-tutorial/

Необходимо развернуть три виртуальные машины с именами `manger1`, `worker1` и `worker2` с общей приватной сетью.

Инициализация кластера

1. Подключитесь по SSH к виртуальной машине `manager1`.
2. Выполните команду для инициализации роя:

```
$ sudo docker swarm init --advertise-addr <IP машины в приватной сети>
```

3. Появиться информация о добавлении нод, которая понадобиться на следующем этапе.

Добавление нод

1. Подключитесь по SSH к виртуальной машине `worker1`.
2. Запустите команду, полученную в результате выполнения пункта `2` предыдущего этапа. Для повторного получения информации необходимо выполнить следующую команду на узле `manager1`

```
$ sudo docker swarm join-token worker
```

3. Далее подключитесь по SSH к виртуальной машине `worker2` и осуществите подключение к кластеру в роли "worker".
4. Подключитесь на машину `manager1` и выведите перечень узлов:

```
$ sudo docker node ls
```

Развертывание сервиса в кластере

1. Переключитесь к виртуальной машине `manager1`.

2. Запустите следующую команду:

```
$ sudo docker service create --replicas 1 --name helloworld alpine ping docker.com
```

- Команда `docker service create` создает службу.
- Флаг `--name` называет службу helloworld.
- Флаг `--replicas` указывает желаемое состояние одного работающего экземпляра.
- Аргументы `alpine ping` docker.com определяют службу как контейнер Alpine Linux, который выполняет команду `ping docker.com`.

3. Запустите команду чтобы посмотреть перечень запущенных сервисов:

```
$ sudo docker service ls
```

Инспектирование сервиса в кластере

1. Для просмотра информации о сервисе запустите команду на машине `manger1`:

```
$ sudo docker service inspect --pretty helloworld
```

2. Чтобы вывести в формате JSON запустите команду без параметра `--pretty`:

```
$ sudo docker service inspect  helloworld
```

3. Узнайте на каких нодах запущен сервис запустив команду:

```
$ sudo docker service ps helloworld
```

Масштабирование сервиса в кластере

1. Выполните следующую команду, чтобы изменить желаемое состояние сервиса, работающего в кластере:

```
$ sudo docker service scale helloworld=5
```

2. Посмотрите состояние сервиса:

```
$ sudo docker service ps helloworld
```

Удаление сервиса

1. Выполните следующую команду, чтобы удалить сервис:

```
$ sudo docker service rm helloworld
```

2. Для просмотра информации о сервисе запустите команду. Команда должна вернуть сообщение о том что сервис не найден

```
$ sudo docker service inspect helloworld
```

3. Несмотря на то, что служба больше не существует, очистка контейнеров занимает несколько секунд. Запустите `docker ps` на всех машинах, чтобы проверить, когда задачи были удалены:

```
$ sudo docker ps
```

Обновление сервиса

1. Выполните следующую команду на машине `manager1`, чтобы развернуть сервис `Redis` в группе с 10-секундной задержкой обновления:

```
$ sudo docker service create \
  --replicas 3 \
  --name redis \
  --update-delay 10s \
  redis:3.0.6
```

2. Просмотрите информацию о сервисе:

```
$ sudo docker service inspect --pretty redis
```

3. Теперь вы можете обновить образ контейнера `Redis`. Менеджер кластера применяет обновление к узлам в соответствии с политикой `UpdateConfig`:

```
$ sudo docker service update --image redis:3.0.7 redis
```

По умолчанию планировщик применяет чередующиеся обновления следующим образом:

- Остановка первого `task`.
- Планирование обновления для остановленной задачи(`task`).
- Запуск контейнера обновленной задачи.
    - Если обновление задачи возвращает `RUNNING`, ожидание установленной задержки, а затем запуск задачи.
    - Если в процессе обновления задача возвращает `FAILED`, остановка обновления.

4. Просмотрите состояние сервиса:

```
$ sudo docker service inspect --pretty redis
```

5. Для перезапуска обновления необходимо запустить следующую команду:

```
$ sudo docker service update redis
```

6. Запустите следующую команду, чтобы посмотреть на процесс обновления:

```
$ sudo docker service ps redis
```

Обслуживание нод кластера

На предыдущих этапах руководства все узлы работали в `ACTIVE`режиме. Менеджер кластера может назначать задачи любому `ACTIVE` узлу, поэтому до сих пор все узлы были доступны для получения задач.

Иногда, например, во время планового обслуживания, вам необходимо установить узел в режим `DRAIN`. `DRAIN` режим не позволяет узлу получать новые задачи от менеджера кластера. Это также означает, что менеджер останавливает задачи, выполняемые на узле, и запускает задачи реплики на `ACTIVE` доступном узле.

1. Запустите команду на машине `manager1`, чтобы убедиться что все узлы доступны:

```
$ sudo docker node ls
```

2. Запустите сервис `redis` , если он еще не запущен.

```
$ sudo docker service create --replicas 3 --name redis --update-delay 10s
redis:3.0.6
```

3. Запустите команду для перевода узла `worker1` в режим `DRAIN`:

```
$ sudo docker node update --availability drain worker1
```

4. Осмотрите узел, чтобы проверить его доступность:

```
$ sudo docker node inspect --pretty worker1
```

5. Запустите команду, чтобы увидеть, как Swarm Manager обновил назначения задач для `redis` сервиса:

```
$ sudo docker service ps redis
```

6. Теперь запустите команду для перевода узла `worker1` в режим `ACTIVE`:

```
$ sudo docker node update --availability active worker1
```

7. Осмотрите узел, чтобы проверить его доступность:

```
$ sudo docker node inspect --pretty worker1
```