KUB 01: Введение в Kubernetes

Описание:

Добро пожаловать на практикум Kubernetes by Rebrain!
Система оркестрации kubernetes плотно вошла в нашу ИТ-жизнь за последние несколько лет. Но предшествовало ей одно знаменательное событие — контейнерная виртуализация. До ее появления каждая команда сама решала, как ей стоит работать со своей инфраструктурой и приложениями. Кто-то писал скрипты автоматизации для накатки обновлений, кто-то по старинке разворачивал новую версию ПО из zip-архива. Кто-то использовал системы управления конфигурациями для настройки серверов, а кто-то настраивал их руками.
С появлением контейнерной виртуализации и, в частности, docker наша жизнь радикально изменилась. Теперь у нас есть универсальный инструмент для работы с приложениями. Мы добавили новый уровень абстракции — контейнеры. Они универсальны, и нас теперь не сильно интересует, что находится внутри них: nginx или ruby application, или python скрипт, или приложение на python. Мы можем работать с контейнерами везде одинаково — главное, чтобы на сервере стояла система запуска контейнеров, например, docker.
Окей, мы научились собирать и запускать контейнеры на сервере, следующим действием стала необходимость запуска нескольких контейнеров, связанных между собой. К примеру, мы хотим запустить наше приложение в одном контейнере, а базу данных, которую оно будет использовать, в соседнем. И тут на помощь пришел docker-compose, который позволил нам в рамках одной виртуальной (или физической) машины запускать множество контейнеров и связывать их между собой. На данном этапе, для развертывания сложного приложения, состоящего из десятков сервисов, требовалось всего лишь запустить одну команду — docker-compose up. Это ли не чудесно?
Но необходимо было двигаться дальше. Нужно было научиться запускать и связывать между собой контейнеры, которые могли запускаться на нескольких хостах. По сути, нам всем была нужна возможность создавать кластеры, которые позволили бы совсем перестать думать о низком уровне — виртуальных (или физических) хостах. Мы бы просто хотели собрать контейнер, отдать его мастер-серверу, и тот бы уже сам решил, на какой машине его запускать. Также было бы хорошо, если мастер-сервер позволил бы нам отслеживать состояние нашего контейнера и перезапускать его в случае необходимости. По сути, такой распределенный docker-compose в кластере. И мы пришли к этой технологии — системам оркестрации контейнеров.
За несколько лет появилось несколько систем оркестрации, ориентированных именно на работу с контейнеризированными приложениями:

- docker swarm,
- kubernetes,
- mesos,
- nomad.

Такие системы существенно упрощают работу с большим числом контейнеров.
21 июля 2015 года выпущена версия Kubernetes 1.0, после чего компания Google в партнерстве с Linux Foundation организовала специальный фонд Cloud Native Computing Foundation (CNCF), которому корпорация передала Kubernetes в качестве начального технологического вклада.
Kubernetes позволяет запускать приложения на тысячах узлов. Из-за того, что данные приложения работают в контейнерах, они не влияют на работу друг друга. Приложение становится отделенным от базовой инфраструктуры: сервер, сеть, хранилище. Это позволяет упростить и ускорить развертывание новых приложений. Процедура разворачивания всегда одинакова и не зависит от количества узлов в кластере.
Но если заглянуть чуть дальше, то можно сказать, что управление инфраструктурой вышло на новый уровень. Kubernetes, по сути, предоставляет нам API для доступа к инфраструктуре. Нужно запустить приложение? Отправляем запрос в API. Нужен диск на 20 Gb? Отправляем запрос в API. Нужно увеличить количество воркеров? Отправляем запрос в API. Это действительно очень удобно.
Итак, в данном практикуме вы познакомитесь с самой популярной системой оркестрации в мире — Kubernetes. Все задания будут состоять из теоретической и практической частей. Практическая часть позволит вам набить руку в работе с Kubernetes. Сразу хочется добавить, что в данной программе мы будем работать только с локальной установкой кубернетес на виртуальных машинах. Это решение было принято осознанно, поскольку kubernetes в разных облаках выглядит немного по-разному - отличаются настройки балансировщиков, добавляются различные типы внешних volumes, присутствуют или отсутствуют сетевые политики. И чтобы не записывать в одну программу 10 различных опций в зависимости от облака мы решили запустить несколько практикумов по kubernetes - свой для каждого облака. Но не переживайте - в данной программе вы познакомитесь со всеми основными ресурсами в kubernetes и сможете легко подключиться к любому облаку, всего лишь узнав про его конкретные особенности.
Как вы уже прекрасно понимаете мы начнем нашу программу без раскачки и сразу начнем запускать kubernetes. Поэтому в первом задании предлагаем вам размяться и установить утилиту kubectl локально на вашу систему. Эта утилита используется для общения с API Kubernetes и управления ресурсами. Ни одно задание в данном практикуме не обойдется без kubectl.
Установка хорошо описана в документации. Хотя по факту много описывать и не пришлось, ведь весь процесс заключается в выполнениии одной команды для линукс систем:

```
curl -LO
https://storage.googleapis.com/kubernetes-release/release/`curl
-s
https://storage.googleapis.com/kubernetes-release/release/stable
.txt`/bin/linux/amd64/kubectl && chmod +x kubectl && mv kubectl
/usr/local/bin
```

В OS X установка выглядит еще компактнее, если использовать менеджер пакетов brew:

```
brew install kubectl
```

После установки вы можете проверить установленную версию с помощью команды kubectl:

```
$ kubectl version --client
Client Version: version.Info{Major:"1", Minor:"20",
GitVersion:"v1.20.2",
GitCommit:"faecb196815e248d3ecfb03c680a4507229c2a56",
GitTreeState:"clean", BuildDate:"2021-01-13T13:28:09Z",
GoVersion:"go1.15.5", Compiler:"gc", Platform:"darwin/amd64"}
```

Таким образом мы получили установленную kubectl на нашу систему.
В нашем уроке задание будет по сути разминочным - чтобы вы познакомились с созданием окружения и автоматическими проверками, поэтому не пугайтесь - дальше сложность будет только возрастать :)

Полезные ссылки:

- Wikipedia - Kubernetes
- Kubernetes: что это?

Задание

1. Установите последнюю версию kubectl на виртуальную машину в директорию /usr/local/bin
2. Отправьте задание на проверку.