KUB 04: Запуск Kubernetes в облаке с помощью terraform

Описание:

Как уже говорилось в предыдущих заданиях, данный практикум делает основной упор на локальную установку Kubernetes, то есть self-hosted версию. Но все же мы решили показать вам, как быстро и просто развернуть облачный Kubernetes кластер в облаке Yandex. Данный пример очень похож на развертывание кластера и в других публичных облаках - AWS, GCP, Azure. Мы сделали это задание опциональным, поскольку вам придется самостоятельно зарегистрироваться в яндекс облаке, а так же установить клиентскую утилиту yc для управления облачными ресурсами. При регистрации Яндекс выдает стартовый грант на 4 000 рублей - этого с головой хватит на первоначальный запуск Kubernetes. Итак, поехали!

В случае, если поддержка своего кластера не кажется интересной задачей и хочется иметь возможность расширять свой кластер потенциально бесконечно, есть смысл посмотреть в сторону облачных поставщиков, которые, как правило, берут на себя управление Control Plane, то есть за Scheduler, Controller Manager, Kubernetes API и etcd. А заодно предоставляют большее количество возможностей за счет интеграции с другими облачными сервисами, такими как балансировщики нагрузки, облачные хранилища и многими другими.

Поставщиков Kubernetes as a Service, на самом деле, очень много — у каждого уважающего себя IT-гиганта есть такая услуга: Google, AWS, Microsoft, Oracle, IBM, Red Hat (ну не совсем, но все же). Российские гиганты тоже не остались в стороне — подобную услугу предлагают и Yandex.Cloud (Yandex Managed Service for Kubernetes), и Mail.ru Cloud Services.

Условно их можно разбить на 2 категории — Kubernetes as a Service и Kubernetes в облачной инфраструктуре. Разница состоит в том, что в первом случае мы получаем кластер, управление которым берет кто-то другой, а во втором — автоматизированная система облачного провайдера берет все на себя.

Kubernetes в облачной инфраструктуре

Вариант использования: развертывание в облаке.

Плюсы: скорость, масштабируемость, управляемость.

Минусы: блокировки, безопасность.

Если размещение всех данных и рабочих нагрузок в облаке является приемлемым, проще всего развернуть и использовать Kubernetes с помощью услуги, предоставляемой крупным облачным провайдером.

Основные варианты сегодня — это Managed Kubernetes от Yandex Cloud, Google Container Engine (сокращенно — GKE, чтобы отличить его от Google Compute Engine), Amazon Elastic Kubernetes Service (EKS) и Azure Kubernetes Service (AKS).

Разворачивание Kubernetes в Yandex Managed Service for Kubernetes

Чтобы развернуть кластер Kubernetes в облаке, можно использовать как веб-интерфейс, так и API. Поскольку все стараются идти по пути лучших девопс-практик и использовать Infrastructure as a Code в своей работе, то второй вариант выглядит предпочтительным. Но создавать что-либо через API в лоб достаточно тяжело. Для исправления этой ситуации Hashicorp предложил инструмент, который позволяет управлять облачными ресурсами, — Terraform. А Yandex.Cloud выпустили свой провайдер, который позволит вам развернуть кластер Kubernetes в несколько строчек кода.
Давайте попробуем развернуть кластер Kubernetes в Yandex Managed Service for Kubernetes. Для этого нам нужно две основные утилиты — yandex cli и Terraform.
Для начала нам будет необходимо установить yandex cli для управления облаком через API. Для Linux достаточно набрать одну команду:

```
curl https://storage.yandexcloud.net/yandexcloud-yc/install.sh | bash
```

После чего можно переходить к настройке аккаунта:

```
yc init
```

Yandex Cli попросит открыть URL, который создаст для вас oAuth token, он и будет использован утилитой для доступа к облаку. Далее утилита попросит указать id облака, для которого необходимо настроить данный профиль, а также дефолтную зону доступности. После этого настройку можно считать оконченной!
После установки вы можете вывести все переменные конфигурации с помощью команды:

```
yc config list
```

Эти параметры нам будут нужны при настройке terraform. Итак, давайте создадим директорию yc-kub и перейдем в нее:

```
mkdir yc-kub
cd yc-kub
```

После чего создадим несколько файлов для terraform:
main.tf — в этом файле указываются основные параметры для инициализации провайдера Yandex.Cloud.

```
## Yandex.Cloud
variable "yc_token" {
  type = string
  description = "Yandex Cloud API key"
}
variable "yc_region" {
  type = string
  description = "Yandex Cloud Region (i.e. ru-central1-a)"
}
variable "yc_cloud_id" {
  type = string
```

```
  description = "Yandex Cloud id"
}
variable "yc_folder_id" {
  type = string
  description = "Yandex Cloud folder id"
}

#-----

# Provider
terraform {
  required_providers {
    yandex = {
      source = "yandex-cloud/yandex"
      version = "0.48.0" # Актуально на момент составления
данной статьи, актуальную версию необходимо проверять в
документации
    }
  }
}

provider "yandex" {
  token = var.yc_token
  cloud_id  = var.yc_cloud_id
  folder_id = var.yc_folder_id
  zone      = var.yc_region
}
```

vars.auto.tfvars — в этом файле указываем публично доступные параметры, в нашем случае — регион в облаке, который будет использоваться по умолчанию:

```
yc_region = "ru-central1-c"
```

private.auto.tfvars — файл с приватными параметрами, который не должен попасть в git-репозиторий, поэтому не забудьте добавить его в .gitignore. Здесь нужно указать ваши параметры, которые можно получить через yc config list.

```
yc_token = "..."
yc_cloud_id = "..."
yc_folder_id = "..."
```

network.tf — файл, в котором мы описываем настройки virtual private cloud. Мы создаем сеть internal, а также — по одной подсети в каждой зоне доступности — ru-central1-a, ru-central1-b и ru-central1-c:

```
resource "yandex_vpc_network" "internal" {
  name = "internal"
}

resource "yandex_vpc_subnet" "internal-a" {
  name           = "internal-a"
  zone           = "ru-central1-a"
  network_id     = yandex_vpc_network.internal.id
  v4_cidr_blocks = ["10.0.0.0/16"]
}

resource "yandex_vpc_subnet" "internal-b" {
  name           = "internal-b"
  zone           = "ru-central1-b"
  network_id     = yandex_vpc_network.internal.id
  v4_cidr_blocks = ["10.1.0.0/16"]
}

resource "yandex_vpc_subnet" "internal-c" {
  name           = "internal-c"
  zone           = "ru-central1-c"
  network_id     = yandex_vpc_network.internal.id
  v4_cidr_blocks = ["10.2.0.0/16"]
}
```

service-accounts.tf — файл для настройки сервисных аккаунтов. Мы создаем два аккаунта — один нужен для управления группами узлов — для их автоматического масштабирования, а второй — для авторизации в container registry — чтобы мы могли скачивать и заливать наши образы в yandex container registry.
Сервисный аккаунт — аккаунт, от имени которого программы способны управлять ресурсами в Yandex.Cloud. То есть мы можем написать приложение, которое, используя сервисный аккаунт, будет обращаться к API Yandex Cloud и управлять ресурсами — добавлять ноды в кластер, удалять или создавать базы и так далее. В данном случае нам требуется как раз аккаунт для управления виртуальными машинами в группе узлов — чтобы работало автоматическое масштабирование кластера, а также сервисный аккаунт, который сможет обращаться к container registry.
Container registry — это репозиторий, который хранит образы контейнеров. Практически тот же docker hub, только приватный в Yandex.Cloud.

```
resource "yandex_iam_service_account" "docker-registry" {
  name        = "docker-registry"
  description = "service account to use container registry"
}
```

```
resource "yandex_iam_service_account" "instances-editor" {
  name        = "instances-editor"
  description = "service account to manage VMs"
}

resource "yandex_resourcemanager_folder_iam_binding" "editor" {
  folder_id = var.yc_folder_id

  role = "editor"

  members = [

"serviceAccount:${yandex_iam_service_account.instances-editor.id
}",
  ]

  depends_on = [
    "yandex_iam_service_account.instances-editor"
  ]
}

resource "yandex_resourcemanager_folder_iam_binding" "pusher" {
  folder_id = var.yc_folder_id

  role = "container-registry.images.pusher"

  members = [

"serviceAccount:${yandex_iam_service_account.docker-registry.id}
",
  ]

  depends_on = [
    "yandex_iam_service_account.docker-registry"
  ]
}
```

Обращаю ваше внимание на параметр depends_on, который указан при создании yandex_resourcemanager_folder_iam_binding. Эти директивы необходимы для указания явных зависимостей между объектами. В большинстве случаев terraform самостоятельно

строит дерево зависимостей, и нет необходимости в дополнительных директивах. Это здорово видно в структуре предыдущего файла — network.tf. Ресурсы yandex_vpc_subnet будут созданы после его создания и удалены перед его удалением. Но в данном случае при создании сервисных аккаунтов прямых зависимостей нет, что способно привести к некорректному процессу создания или удаления кластера.
kubernetes.tf — основной файл, в котором мы описываем кластер, а также группу узлов, которая будет к нему подключаться. Вообще кластеры бывают двух типов — зональный и региональный. Региональный дает большую отказоустойчивость, за счет создания трех машин для api server, а также размещения их в трех разных зонах доступности. Таким образом, при падении одного из серверов доступность нарушена не будет. Такой режим рекомендуется использовать в production окружениях.

```
# Создаем кластер кубернетес
resource "yandex_kubernetes_cluster" "kuber-cluster" {
  # Указываем его имя
  name        = "kuber-cluster"

  # Указываем, к какой сети он будет подключен
  network_id = yandex_vpc_network.internal.id

  # Указываем, что мастера располагаются в регионе ru-central и
какие subnets использовать для каждой зоны
  master {
    regional {
      region = "ru-central1"

      location {
        zone      = yandex_vpc_subnet.internal-a.zone
        subnet_id = yandex_vpc_subnet.internal-a.id
      }

      location {
        zone      = yandex_vpc_subnet.internal-b.zone
        subnet_id = yandex_vpc_subnet.internal-b.id
      }

      location {
        zone      = yandex_vpc_subnet.internal-c.zone
        subnet_id = yandex_vpc_subnet.internal-c.id
      }
    }

    # Указываем версию Kubernetes
```

```
    version   = "1.18"
    # Назначаем внешний ip master нодам, чтобы мы могли
подключаться к ним извне
    public_ip = true
  }

  # Указываем канал обновлений
  release_channel = "RAPID"

  # Указываем сервисный аккаунт, который будут использовать
ноды, и кластер для управления нодами
  node_service_account_id =
yandex_iam_service_account.docker-registry.id
  service_account_id      =
yandex_iam_service_account.instances-editor.id
}

# Создаем группу узлов
resource "yandex_kubernetes_node_group" "node-group-0" {
  # Указываем, к какому кластеру они принадлежат
  cluster_id  = yandex_kubernetes_cluster.kuber-cluster.id
  # Указываем название группы узлов
  name        = "node-group-0"
  # И версию
  version     = "1.18"

  # Настраиваем шаблон виртуальной машины
  instance_template {
    platform_id = "standard-v2"

    network_interface {
      nat = true
    }

    resources {
      core_fraction = 20 # Данный параметр позволяет уменьшить
производительность CPU и сильно уменьшить затраты на
инфраструктуру
      memory        = 2
      cores         = 2
    }
```

```
    boot_disk {
      type = "network-hdd"
      size = 64
    }

    scheduling_policy {
      preemptible = false
    }
  }

  # Настраиваем политику масштабирования — в данном случае у нас группа фиксирована и в ней находятся 2 узла
  scale_policy {
    fixed_scale {
      size = 2
    }
  }

  # В каких зонах можно создавать машинки — указываем все зоны
  allocation_policy {
    location {
      zone = "ru-central1-a"
    }

    location {
      zone = "ru-central1-b"
    }

    location {
      zone = "ru-central1-c"
    }
  }

  # Отключаем автоматический апгрейд
  maintenance_policy {
    auto_upgrade = false
    auto_repair  = true
  }
}
```

Больше параметров и их описание можно найти в официальной документации.
Теперь мы можем инициализировать terraform и создать наш кластер:

```
# init скачает и настроит провайдер Yandex.Cloud
terraform init
# запускаем создание ресурсов
terraform apply
```

После создания можем посмотреть на наш кластер, используя утилиту yandex cli:

```
$ yc container cluster list
+----------------------+--------------+---------------------+--
------+---------+-----------------------+-------------------+
|          ID          |     NAME     |     CREATED AT      |
HEALTH  | STATUS  |   EXTERNAL ENDPOINT    | INTERNAL ENDPOINT |
+----------------------+--------------+---------------------+--
------+---------+-----------------------+-------------------+
| catcn9mkdf475bj54h3f | kuber-cluster | 2020-12-02 12:15:35 |
HEALTHY | RUNNING | https://84.201.133.155 | https://10.0.0.7  |
+----------------------+--------------+---------------------+--
------+---------+-----------------------+-------------------+
```

И также, если вы поставили локально утилиту kubectl, вы сможете добавить новую конфигурацию кластера в kubeconfig:

```
$ yc container cluster get-credentials kuber-cluster --external

Context 'yc-kuber-cluster' was added as default to kubeconfig
'/Users/vozerov/.kube/config'.
Check connection to cluster using 'kubectl cluster-info
--kubeconfig /Users/vozerov/.kube/config'.

Note, that authentication depends on 'yc' and its config profile
'default'.
To access clusters using the Kubernetes API, please use
Kubernetes Service Account.
```

Можем проверить состояние кластера:

```
$ kubectl cluster-info
Kubernetes master is running at https://84.201.133.155
CoreDNS is running at
https://84.201.133.155/api/v1/namespaces/kube-system/services/ku
be-dns:dns/proxy
```

```
Metrics-server is running at
https://84.201.133.155/api/v1/namespaces/kube-system/services/ht
tps:metrics-server:/proxy

To further debug and diagnose cluster problems, use 'kubectl
cluster-info dump'.
$ kubectl get nodes
NAME                        STATUS   ROLES    AGE   VERSION
cl128p75cket5bspv91v-oles   Ready    <none>   10m   v1.18.9
cl128p75cket5bspv91v-oxuq   Ready    <none>   10m   v1.18.9
```

Google Kubernetes Engine (GKE) — это просто еще одна услуга в большом наборе, предлагаемом в рамках облачной платформы Google.
Если вы уже используете GCP или GCE (часть платформы IaaS), GKE — это всего лишь щелчок мыши в веб-консоли Google cloud, и произойдет интеграция с существующим Identity and Access Management. При развертывании нового кластера необходимо указать операционную систему (традиционную Linux или ОС с высокой оптимизацией для контейнеров), размер инстанса и размер кластера (количество рабочих нод). GKE автоматически создает и управляет мастер нодами для кластера, поэтому получить к ним доступ не представляется возможным. Однако вы можете получить доступ (через SSH) — и оплачивать — рабочие ноды, которые являются обычными инстансами GCE. Кластер GKE так же можно легко расширить, добавив рабочие ноды в пул или дополнительные пулы нод.
GKE выделяется своей способностью изящно обрабатывать обновления: обновления master node (API) происходят автоматически и прозрачно через несколько недель после каждого нового выпуска Kubernetes, поэтому версия API вашего кластера поддерживается в актуальном состоянии с новыми функциями и исправлениями ошибок. В то время как обновление рабочих нод исторически требовало ручного действия пользователя, GKE недавно добавила флаг, позволяющий автоматически обновлять рабочие узлы.
Еще одним примечательным недавним дополнением является поддержка федеративности, позволяющая нескольким кластерам взаимодействовать по всему миру, обеспечивая высокую доступность и низкую задержку ответов веб-приложений.
Azure Kubernetes Service (AKS) — это гораздо более молодое предложение от Microsoft. По сравнению с GKE, качество предложения AKS может вызывать вопросы.
С другой стороны, AKS может оказаться интересным вариантом для пользователей, заинтересованных в запуске приложений .NET на ОС Windows Server.
Поддержка Docker и Kubernetes для Windows молода и развивается, но есть компания, которая собирается бороться и вкладывать в нее ресурсы — это Microsoft.
Поэтому неудивительно, что AKS превратилась в отличный (если не единственный) выбор для запуска рабочих нагрузок Windows на Kubernetes.
Amazon Elastic Kubernetes Service (Amazon EKS) — это полностью управляемый сервис Kubernetes от Amazon.

Такие клиенты, как Intel, Sap, Intuit, GoDaddy и Autodesk запускают наиболее критически важные приложения на этом решении из-за его безопасности, надежности и масштабируемости.
EKS работает в тесном сотрудничестве с командой Kubernetes и сертифицирован Kubernetes conformant, поэтому вы можете использовать все преимущества инструментов с открытым исходным кодом от сообщества.
Вы также можете легко перенести любое стандартное приложение Kubernetes в EKS без необходимости рефакторинга кода.
Kubernetes as a Service
Пример использования: гибридные и мультиоблачные развертывания.
Плюсы: скорость, масштабируемость, управляемость.
Минусы: стоимость.
Kubernetes может использоваться как услуга пользователями, ищущими более быстрое и простое решение, которое позволило бы им сосредоточиться на создании программного обеспечения, а не на управлении контейнерами. Хорошо известные примеры управляемых сервисов Kubernetes включают Kube2Go.io, стек Point.io и Platform9 Managed Kubernetes.
Platform9 Managed Kubernetes (PMK)
PMK предлагает действительно корпоративный управляемый сервис Kubernetes, который работает в любой базовой инфраструктуре: включая физическую серверную инфраструктуру, виртуализированные среды или облака, такие как AWS, Azure и GCP.
Platform9 Managed Kubernetes предоставляется как SaaS-управляемое решение, с развертыванием, мониторингом, устранением неполадок и обновлениями, о которых заботится Platform9.
Таким образом, операционное SLA для управления Kubernetes обеспечивается платформой 9.
Помимо того, что служба полностью управляется и работает на любом сервере или облачной инфраструктуре, Platform9 Managed Kubernetes имеет несколько общих корпоративных интеграций:

1. Единое представление нескольких кластеров.
2. Высокодоступные кластеры Kubernetes с несколькими мастерами, которые автоматически масштабируются в зависимости от рабочих нагрузок.
3. Общие промышленные интеграционные решения, такие как SSO/изолированные пространства имен; и возможность развертывания приложений с помощью Helm.
4. Кластерная федеративность, обеспечивающая действительно бесшовную гибридную среду в нескольких облаках или центрах обработки данных.

Чтобы начать работу с Platform9, вы можете развернуть бесплатную песочницу с установленным Kubernetes.
Песочница включает в себя пошаговое руководство для управляемых SaaS Kubernetes.
Чтобы создать локальный кластер, установите поддерживаемую операционную систему Linux на хосты с доступом в интернет, загрузите установщик для агента Platform9 и примените его на хостах.
Кроме того, вы можете развернуть кластер Kubernetes в облаке, например, Amazon Web Services, предоставив учетные данные пользователя для своей облачной среды.

Полезные ссылки:

- AWS Fargate
- Getting started (official docs)
- Kubernetes Cloud Services
- The Ultimate Kubernetes Cost Guide: AWS vs GCP vs Azure vs Digital Ocean
- Yandex Cloud Terraform Quick Start
- Terraform Yandex Cloud Provider Documentation w/ Examples

Задание:

1. Зарегистрируйтесь в Яндекс Облаке (console.cloud.yandex.ru), подтвердите свои данные и укажите свою карту - это необходимо для получение гранта в 4 000 рублей. Только не забудьте удалить кластер после выполнения задания, чтобы у вас не списывались деньги (сделать это можно в веб интерфейсе console.cloud.yandex.ru в сервисе managed kubernetes либо с помощью команды terraform destroy).
2. Установите terraform, yandex cli и kubectl на подготовленную виртуальную машину под пользователем user
3. Настройте yandex cli для работы с вашим облаком под пользователем user
4. Напишите terraform файл, который будет создавать следующие ресурсы:

- сервисный аккаунт docker с правами push в container registry;
- сервисный аккаунт instances с правами editor для создания виртуальных машин;
- сеть internal с адресацией 10.200.0.0/16, 10.201.0.0/16 и 10.202.0.0/16 в разных зонах доступности;
- Kubernetes-кластер с именем kub-test;
- Kubernetes-кластер должен быть зональный (1 мастер в зоне ru-central1-a, в остальных зонах мастеров нет)
- группа узлов с именем test-group-auto;
- группа узлов должна масштабироваться автоматически с максимальным количеством хостов 3 штуки и минимальным — 2 штуки;
- группа узлов должна находиться в одной зоне доступности (allocation_policy - ru-central1-a)
- Kubernetes-кластер должен поддерживать сетевые политики безопасности (Включается при создании кластера в managed service for kubernetes. Сетевые политики должны поддерживаться сетевым драйвером - например, calico. При включении данной опции в Yandex Cloud - Yandex установит сетевой плагин calico в kubernetes кластер.)

5. Запустите terraform и создайте кластер.
6. Импортируйте конфигурацию для kubectl, используя yandex cli для пользователя user

7. Выполните команду kubectl label nodes --all status=done, для того чтобы добавить всем узлам метку status=done.
8. Отправьте задание на проверку.