Вашему вниманию представляется дипломная работа на тему «Развертывание kubernetes кластера на AWS с пакетом офисных приложений ONLYOFFICE».

Основной целью работы ставилось:

создать полный цикл развертывания приложения с использованием kubernetes кластера.

За основу был принят кластер AWS EKS (Amazon Elastic Kubernetes Service), который позволяет управлять запуском и масштабированием приложений Kubernetes в облаке и локальной среде.

Причины выбора EKS AWS:

управляемый control plane: поддержка высокодоступного control plane (или плоскости управления) для приложений в промышленной эксплуатации.

безопасность и соответствие требованиям.

стоимость.

масштабируемость.

различные виды вычислительных мощностей.

надёжность и доступность.

Для создания кластера нам нужно поднять свою VPC, создать сети, настроить роутинг, и добавить IAM роль, которая будет использоваться кластером для авторизации.

IAM роль

Переходим в IAM, создаём новую роль, тип EKS:

Permissions AWS выберет сам.

VPC

Далее потребуется создать VPC и подсети – публичную, для Load Balancer, и приватную – для Worker-нод.

Переходим в SecurityGroups, создаём SG для кластера.

Internet Gateway

Создаём IGW, через который будет ходить трафик из публичных подсетей:

Подключаем его к VPC:

Subnets

Pod-ы будут получать IP из подсетей

Создаём первую публичную сеть с блоком 10.0.0.0/18 (всего-то 16384 адресов):

Вторую публичную – 10.0.64.0/18:

В публичных адреса – включаем добавление публичного IP инстансам:

И аналогично добавляем две приватные подсети:

NAT Gateway

В публичной подсети создаём NAT Gateway – через него будет ходить трафик из приватных подсетей

Сразу настраиваем маршрутизацию::

Route tables

Тут требуется создать две таблицы маршрутизации – одна для публичных подсетей, и вторую – для приватных.

Public route table

Создаём для публичной:

Редактируем маршруты – задаём 0.0.0.0/0 через IGW:

Переключаемся во вкладку Subnet association – подключаем две публичные подсети:

Аналогично создаём вторую таблицу, для приватных подсетей:  
Добавляем маршрут к 0.0.0.0/0 через NAT GW:

Возвращаемся к подсетям, подключаем к ним созданные таблицы маршрутизации – Edit route table association:

К приватным подсетям подключаем, соответственно, приватную RTB, с маршрутами через NAT:

К публичным – RTB с маршрутом через IGW

Конфигурация Terraform ссылается на переменную cluster-name ( var.cluster-name ), которая используется для согласованности.

EKS требует использования виртуального частного облака в качестве основы для сетевой конфигурации. Использование указанных ниже тегов ресурсов kubernetes.io/cluster/\* необходимо для EKS и Kubernetes для обнаружения сетевых ресурсов и управления ими.

Именно здесь в игру вступает сервис EKS. Для этого требуется несколько ресурсов, управляемых оператором, чтобы Kubernetes мог правильно управлять другими сервисами AWS, а также разрешать входящие сетевые соединения с вашей локальной рабочей станции (при желании) и рабочих узлов.

Ниже приведен пример роли и политики IAM, позволяющих сервису EKS управлять или получать данные из других сервисов AWS. Также возможно создать эти политики с aws\_iam\_policy\_document данных aws\_iam\_policy\_document

Эта группа безопасности контролирует сетевой доступ к мастерам Kubernetes. Позже мы настроим его с помощью правила входа, чтобы разрешить трафик с рабочих узлов.

Данный ресурс является собственно мастер-кластером Kubernetes. Его предоставление в AWS может занять несколько минут.

Получение конфигурации кубекта с терраформы

Если вы планируете использовать kubectl для управления кластером Kubernetes, возможно, сейчас самое подходящее время для настройки вашего клиента. Приведенный ниже вывод Terraform генерирует образец конфигурации kubectl для подключения к вашему кластеру.

Вы можете проверить доступ к кластеру с помощью kubectl version, отображающего информацию о версии сервера в дополнение к информации о версии локального клиента.

установить Prometheus на кластер KubernetesРазверните диаграмму Prometheus Stack Helm

Скачать prometheus-values.yamlфайл для Прометея

helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>

helm repo update

helm search repo kube-prometheus-stack --max-col-width 23

helm install monitoring prometheus-community/kube-prometheus-stack --values prometheus-values.yaml --version 32.2.1

kubectl get pods -n monitoring

kubectl get svc -n monitoring

kubectl port-forward \

svc/monitoring-kube-prometheus-prometheus 9090 \

-n monitoring

kubectl port-forward \

svc/monitoring-kube-prometheus-prometheus 9090 \

-n monitoring

Go to [http://localhost:9090](http://localhost:9090/) and select targets

kubectl get cm kube-proxy-config -n kube-system -o yaml

kubectl -n kube-system get cm kube-proxy-config -o yaml |sed 's/metricsBindAddress: 127.0.0.1:10249/metricsBindAddress: 0.0.0.0:10249/' | kubectl apply -f -

watch -n 1 -t kubectl get pods -n kube-system

kubectl -n kube-system patch ds kube-proxy -p "{\"spec\":{\"template\":{\"metadata\":{\"labels\":{\"updateTime\":\"`date +'%s'`\"}}}}}"

Go back to [http://localhost:9090](http://localhost:9090/) under targets and alerts

Wait 20-30 min to get more data in Prometheus

kubectl port-forward \

svc/monitoring-grafana 3000:80 \

-n monitoring

Open following dashboards

Kubernetes / Compute Resources / Cluster

Kubernetes / Kubelet

USE Method / Cluster

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.1/deploy/static/provider/aws/deploy.yaml>