**Разработка инфраструктуры для тестирования децентрализованной системы управления порталом сайтов**

Актуальность заявленной темы заключается в том, что по данным Global Digital с каждым годом увеличивается количество пользователей в сети Интернет, на 2023 год этот показатель составил 64,4% от мирового населения. С ростом популярности веб-технологий увеличивается и число посетителей веб-порталов, в следствии чего повышается вероятность выхода из строя серверов, обрабатывающих запросы пользователей к сайтам. Последствиями такой ситуации могут быть как экономические, так и репутационные потери. Кроме того, необходимость обеспечения отказоустойчивости обусловлена ГОСТ 28806-90, где одной из основных характеристик качества программного продукта является надежность, которая включает в себя отказоустойчивость как подхарактеристику.

На основании положений, выносимых С. И Заикиным в статье «Концепция обеспечения отказоустойчивости ИТ-инфраструктуры», а также утверждений И. Б Шубинского в труде «Надежные отказоустойчивые информационные системы» можно утверждать, что отказоустойчивость системы обеспечивается за счет избыточности (резервирования) и балансировки нагрузки. И балансировка нагрузки, и резервное копирование – сложные процессы, которыми необходимо каким-то образом управлять для обеспечения нормальной работы портала сайтов. Одним из методов управления в таких системах является применение систем оркестрации контейнеров, в том числе к таким относится Kubernetes.

Таким образом, цель исследования: развернуть кластер Kubernetes из нескольких узлов

Задачи:

1) развернуть две виртуальные машины;

2) настроить кластер Kubernetes;

3) запустить в кластере веб-приложение.

1. Развертывание виртуальных машин

Для развертывания виртуальных машин скачиваем и устанавливаем Virtual Box, а также установочный образ Ubuntu Server 20.04.4 (если на рабочей машине включен Hyper-V, предварительно его необходимо отключить).

Для начала настроим одну шаблонную виртуальную машину. Для этого выделим 25 ГБ для виртуального жесткого диска, 2 ГБ оперативной памяти, 2 процессора и включим два сетевых адаптера с режимами NAT и Host-only. Затем устанавливаем Ubuntu Server (при установке необходимо проверить корректность ip-адресов сетевых адаптеров).

Командой ping проверим сетевое подключение и доступ в Интернет. Скачиваем mc и отключаем swap. Для более удобной работы настраиваем общую папку.

Для установки Docker и Kubernetes напишем инструкции .sh по установке, которые передадим с основной ОС на ВМ через общую папку. Важно, чтобы и Docker и Kubernetes использовали один и тот же драйвер, для этого в /etc/docker создадим daemon.json, где укажем использование systemd, а в /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf добавим --cgroup-driver=systemd. После внесения этих изменений необходимо перезагрузить ВМ и в /etc/containerd/config.toml заменить disabled\_plugin на enabled\_plugin.

Клонируем виртуальную машину, затем настраиваем hostname и ip-адрес. Результат приведен на рисунках 1 и 2.

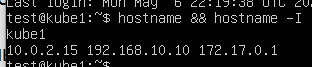


Рисунок 1 - Hostname и ip-адрес первой ВМ



Рисунок 2 - Hostname и ip-адрес второй ВМ

**2. Настройка кластер Kubernetes**

Когда Kubernetes (kubelet) будет запускаться на узле, он в качестве ip узла будет использовать ip адрес сетевого адаптера по умолчанию, а это NAT с динамическими адресами, который нам не подходит. Для настройки статического ip-адреса отредактируем файл /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf, добавив туда --node-ip=192.168.10.10 и --node-ip=192.168.10.11 для первой и второй ВМ соответственно.

Для инициализации кластера вызовем kubeadm init с параметрами --apiserver-advertise-address=192.168.10.10 и --pod-network-cidr=10.10.0.0/16. Весь вывод запишем в файл kubeadm-join.sh.

Для нормальной работы кластера необходимо на каждом из узлов от имени обычного пользователя создать папку /.kube и поместить туда конфигурационных файл. Теперь можно добавить вторую ВМ в кластер, выполнив команду kubeadm join. Токен для этой команды был записан в файл kubeadm-join.sh.

Kubernetes имеет абстрактную модель работы с сетью, а конкретная её реализация определяется сетевыми плагинами, такими как Flannel или Calico. Следовательно, установим Flannel скачав kube-flannel.yml и заменив ip-адрес на то значение, которое передавали в --pod-network-cidr при инициализации. Затем применим этот файл.

Результат развертывания и настройки кластера представлен на рисунке 3.



Рисунок 3 – Под и узлы кластера

**3. Запуск веб-приложения**

Для того, чтобы развернуть свое контейнеризированное приложение в Kubernetes необходимо создать Deployment, который определяет, как создавать и обновлять экземпляры приложения. При создании развертывания нужно указать образ контейнера приложения и количество запущенных реплик. Впоследствии эти параметры можно изменить. Создадим файл deploument1.yml и servise1.yml и внедрим их. Результат представлен на рисунке 4.



Рисунок 5 – Сервисы Kubernetes

Таким образом, развернуты две виртуальные машины, настроен кластер Kubernetes и развернуто приложение, следовательно поставленная цель достигнута. Добавляя новые узлы и поды, а также настроив балансировщик нагрузки можно все больше повышать управляемость и отказоустойчивость портала сайтов, что является поводом для дальнейших исследований.