Kubernetes, также известный как k8s — это портативная расширяемая платформа с открытым исходным кодом для управления контейнеризованными рабочими нагрузками и сервисами, которая облегчает как декларативную настройку, так и автоматизацию. Kubernetes можно настроить как в локальной, так и в облачной инфраструктуре. Однако, кластер Kubernetes немного сложен в настройке и требует значительных ресурсов, что затрудняет развертывание на локальной машине для обучения или разработки. Это приводит к тому, что не для каждого бизнеса перейти на полноценный k8s представляется возможным. К счастью, на этот случай есть более легкое решение в виде k3s.

K3s лучше всего описать как своего рода дистрибутив Kubernetes, который содержит дополнительные компоненты и сервисы, необходимые для полнофункционального кластера, выходящего за рамки стандартного Kubernetes. В этой статье вы узнаете об особенностях k3s, о том, чем он отличается от полноценного Kubernetes, а также в рамках данной статьи произведем его установку на виртуальной машине.

Немного истории о k3s

Во время создания Rio (одна из разработок для kubernetes от Rancher Labs), разработчик Даррен Шепард был разочарован тем, что для тестирования данного продукта, постоянно требовалось запускать новый кластер Kubernetes. Он знал, что это занимает всегда много времени и если он сможет запускать кластер быстрее, он сможет выделять больше времени на написание кода и быстрее выпускать новые фичи. Так как Даррен был хорошо знаком с исходным кодом Kubernetes, он начал его разбирать, удаляя все ненужные компоненты, чтоб обеспечить быстрый запуск.

Вот таким образом в 2019 году компания Rancher Labs объявила о запуске своего нового проекта с открытым исходным кодом k3s, который представлял собой легкий и простой в установке дистрибутив k8s, предназначенный для разработчиков и операторов, ищущих способ запуска Kubernetes на платформах с ограниченными ресурсами.

Чтоб адаптировать kubernetes к слабому железу Rancher Labs потребовалось  следующее:

* Удалить старый и неиспользуемый код. По словам разработчиков из кода было вырезано порядка 1000 строк.
* Вместо Docker использовать conteinerd.
* Добавить поддержку Kine, в качестве хранилища по умолчанию.
* Удалить драйвера облачных провайдеров.

Дистрибутив k3s поддерживает множество архитектур, включая AMD64, ARM64 и ARMv7-8. Благодаря последовательному процессу установки k3s может работать на Raspberry Pi Zero, NVIDIA Jetson Nano, Intel NUC или на экземпляре Amazon EC2 a1.4xlarge.

Отличия от k8s

Давайте для начала разберем, чем k3s отличается от полноценного k8s. Если сравнивать с традиционным кластером Kubernetes, то в k3s нет четкого различия между мастер и воркер нодами. Управление подами происходит на любой ноде, независимо от роли, которую они играют. Таким образом, понятия мастер и воркер нод не применимы к кластеру k3s.

Rancher Labs избавились от множества необязательных компонентов Kubernetes, например, от плагинов объема хранилища и API-интерфейсов облачных провайдеров, которые, как они посчитали, не будут критичны для запуска кластера k3s.

Затем были добавлены некоторые важные элементы, включая containerd, Flannel, CoreDNS, CNI, ingress controller Traefik, локальное хранилище данных, встроенный сервис Load Balancer. Все эти элементы упакованы в один бинарный файл и выполняются в рамках одного процесса. Помимо этого, дистрибутив также поддерживает Helm-charts из коробки. K3s не имеет внешних зависимостей. Это означает, что вам не нужно ничего устанавливать, чтобы запустить его.

Другое ключевое отличие k3s от k8s заключается в способе управления состоянием кластера. Kubernetes полагается на распределенную базу данных «ключ-значение» etcd для хранения всего состояния кластера. Изначально разработчики k3s заменили etcd облегченной базой данных под названием SQLite. Однако позже команда создала (они называют его революционным) проект под названием Kine, что означает «Kine is not etcd». Kine предоставляет прокладку API, которая позволяет k3s поддерживать различные серверные части баз данных, включая MySQL, Postgres и SQLite. Он принимает запросы etcd v3 от Kubernetes, преобразовывает их в SQL-запросы и отправляет в серверную часть вашей базы данных.

Также выделяется еще одна отличительная особенность от старшего брата — автоматическое развертывание. Оно позволяет развертывать манифесты Kubernetes и Helm-чарты, помещая их в определенный каталог. То есть k3s отслеживает изменения и заботится об их применении без дальнейшего взаимодействия. Просто создайте/обновите свою конфигурацию, и k3s обеспечит актуальность ваших deployments .

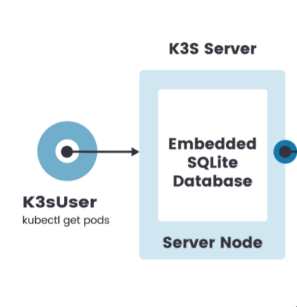
Архитектура

На приведенном выше рисунке показана разница между сервером k3s и узлами агента k3s.

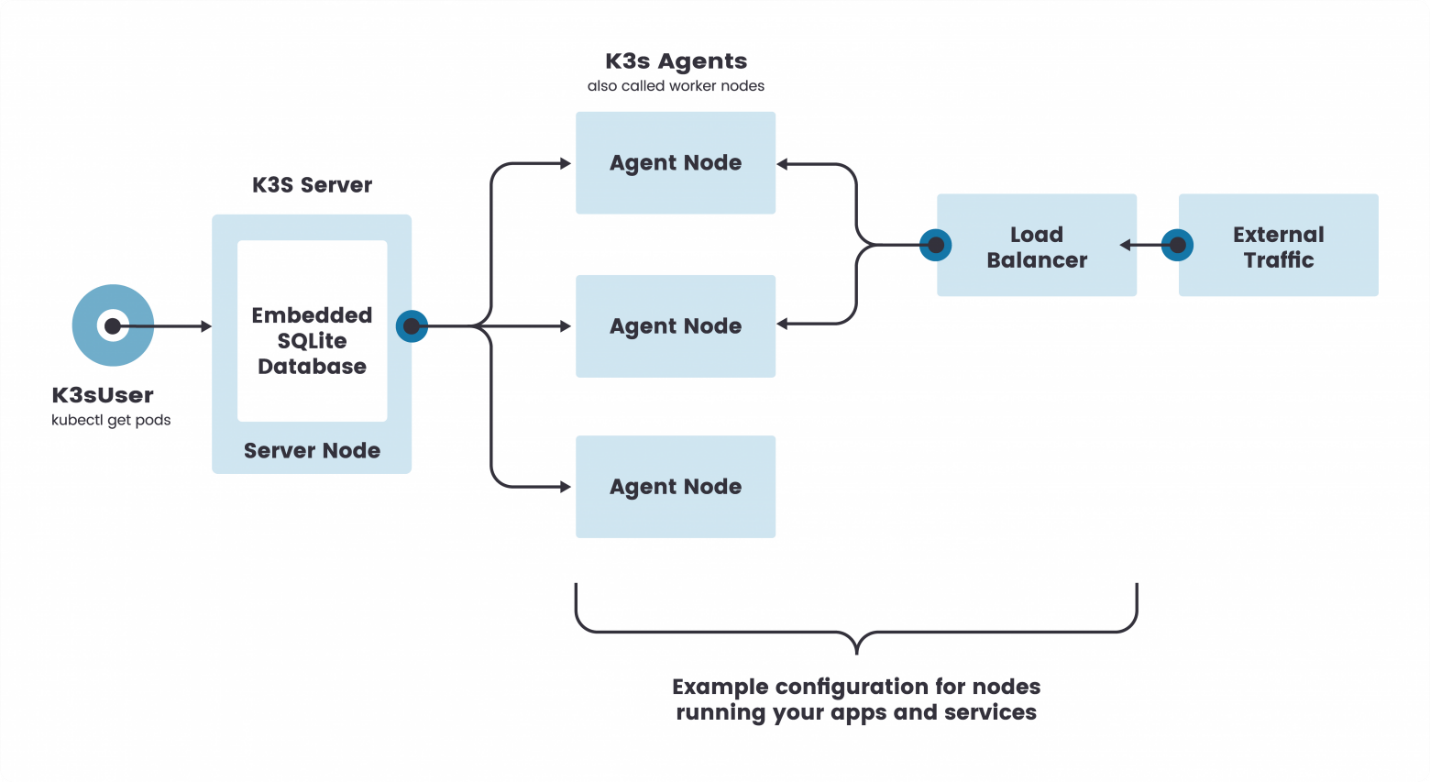
Если копнуть глубже, то k3s упакован и развернут в виде одного бинарного файла размером приблизительно 60 МБ, представляющего из себя самодостаточный инкапсулированный объект, который запускает почти все компоненты кластера Kubernetes.

На рисунке мы наблюдаем две ноды, на одной из которых работает k3s service, а на второй агент k3s. API server проверяет и настраивает данные для объектов API — например pods, services и т.д. Controller Manager наблюдает за общим состоянием кластера через API-сервер, вносит в него изменения и переводит из текущего состояния в желаемое. Scheduler назначает модули узлам в соответствии с их текущим состоянием, ограничениями и доступными ресурсами. Kubelet как основной агент узла в кластере, работающий на каждой ноде. Kine, где в качестве хранилища по умолчанию используется SQLite. Containerd как исполняемая среда для запуска контейнеров. Связь между агентом и сервером осуществляется через туннельный прокси.

Теперь, когда мы разобрались с архитектурой, давайте рассмотрим, как на самом деле можно развернуть k3s. Для k3s существует несколько архитектур развертывания. Давайте рассмотрим основные(картинки взяты из [официальной документации k3s](https://rancher.com/docs/k3s/latest/en/architecture/)):

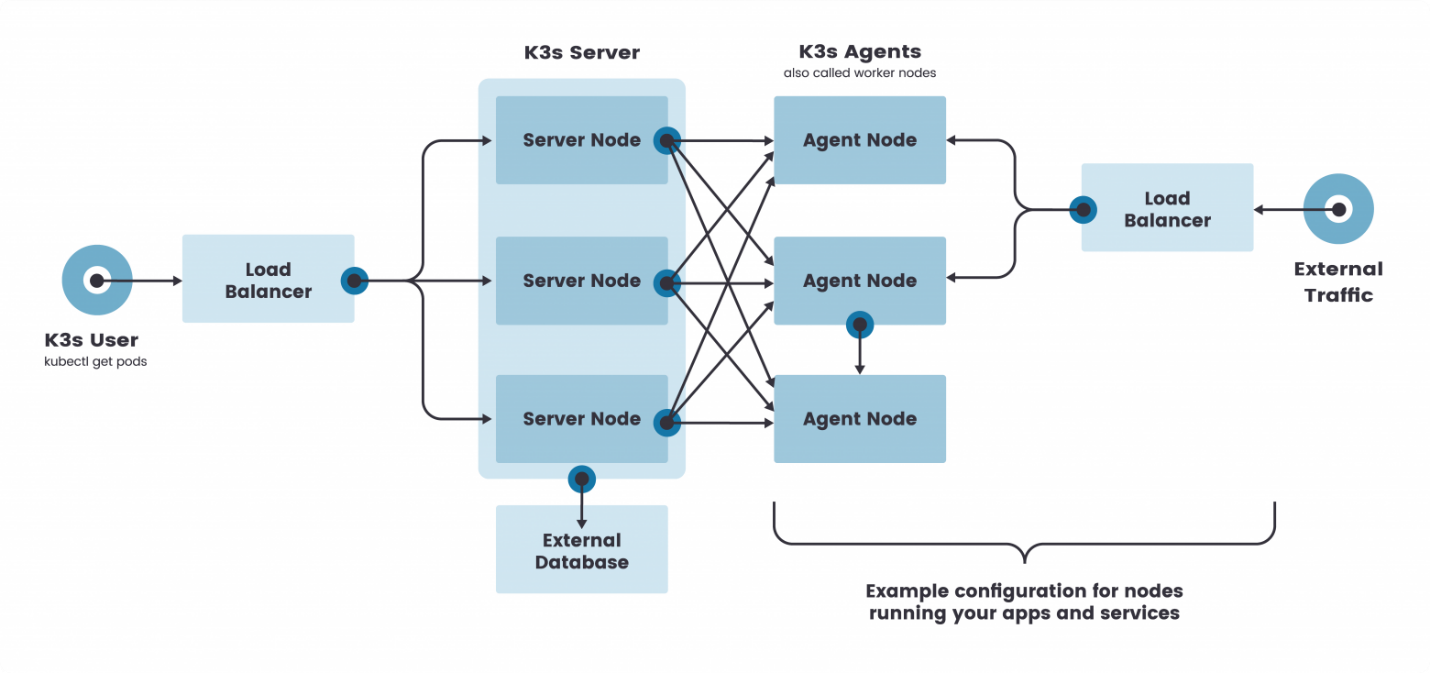
* **Кластер из одной ноды:**  
  

Этот самый упрощенный метод для разворачивания кластера k3s, который оптимизирован для пограничных устройств. Вы просто запускаете команду установки при помощи команды curl, и она разворачивает полноценный кластер Kubernetes. Но стоит сразу обратить внимание, что данная архитектура развертывания не сможет обеспечить вам отказоустойчивость.

* **Один кластер с несколькими агентами:**  
  

Этот метод также использует один сервер к которому вы можете добавить агентов в кластер. Агенты безопасно подключаются с помощью токена, который сервер создает при запуске. В этой конфигурации каждый узел агента регистрируется на одном и том же узле сервера. Пользователь k3s может управлять ресурсами Kubernetes, вызывая API k3s на узле сервера.

* **Высокодоступный сервер k3s с внешней БД:**



Этот метод можно использовать, если вам важно настроить отказоустойчивый кластер. Данный кластер k3s состоит из:

* Двух или более серверных узлов, которые будут обслуживать API Kubernetes и запускать другие службы плоскости управления.
* Внешнего хранилища данных (вместо встроенного хранилища данных SQLite, которое используется в конфигурациях с одним сервером по умолчанию).

В данной статье мы на примере рассмотрим установку однонодового кластера и подключим агента к основному серверу.

Практика “Быстрый старт”

Наша цель быстро запустить кластер k3s на виртуальной машине для демонстрации.  
Рекомендуемые требования, заявленные разработчиками следующие:

* RAM: 512MB минимум (рекомендовано 1GB)
* CPU: минимум 1 ядро  
  Для нашей демонстрации мы подготовили 2 виртуальные машины со следующими характеристиками:
* k3s server с hostname k3s\_server: debian 10, cpu 1, ram 1gb
* k3s agent c hostname worker1: debian 10, cpu 1, ram 1gb

Для начала развернем кластер из одной ноды:

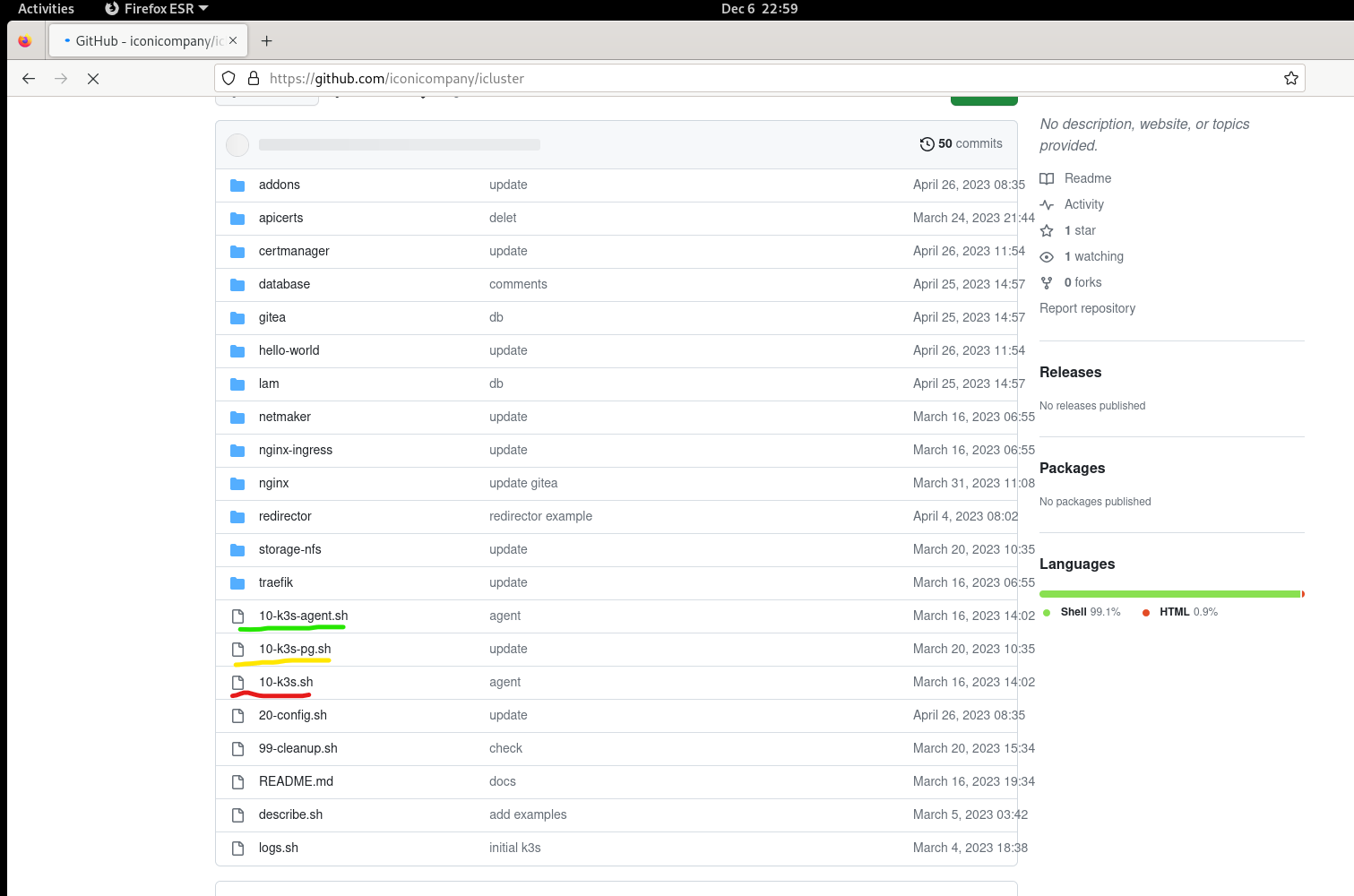
Установка K3S

У нас 3 варианта:

Зеленый – строчка для подключения агента к нашему кластеру

Жёлтый – установка k3s вместе с бд postgres (подходит для дальнейшего масштабирования)

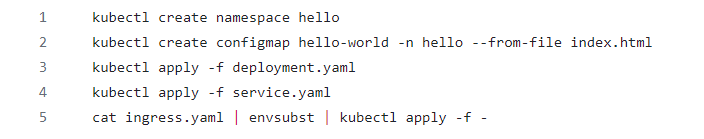
Красный – обычная установка k3s, ее и используем



\*Для подключения через OpenLens нужно запустить файл 20-config.sh\*

Давайте запустим простое приложение hello-world(находится в одноименном каталоге)

Самый простой вариант установки – hello-world.sh



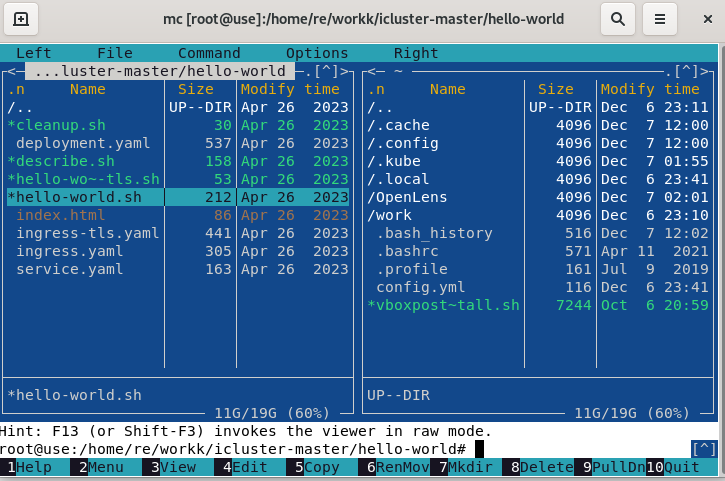
В нем находится описание приложения, и состоит оно из нескольких этапов

Создается namespace(все приложения должны хранится в собственных namespace)

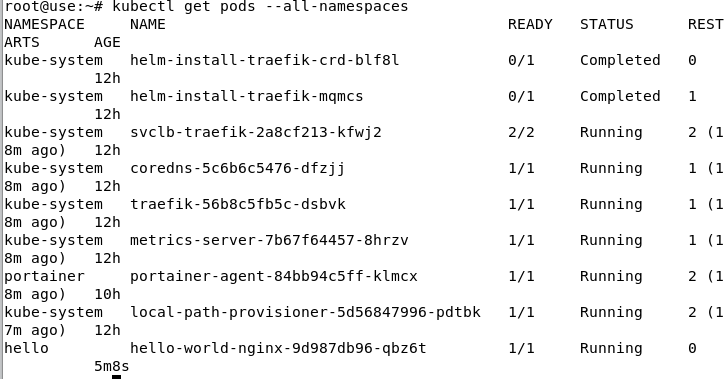
Далее мы создаем том с данными

Подключаем deployment (в файле можете увидеть описание приложения)

\*для удобной работы с .sh файлами установите mc\*



Запускаем и проверяем



Так же в браузере

