УДК 004.722
МРНТИ 50.39.15

ПРИМЕНЕНИЕ СИСТЕМЫ ОРКЕСТРАЦИИ КОНТЕЙНЕРОВ “KUBERNETES”

А.С. БАЙКЕНОВ, К.Д. БЕКБОСЫНОВ

Алматинский университет энергетики и связи

Аннотация: *Пока многие только присматриваются к Kubernetes, оценивая его возможности и зрелость, другие успевают продвинуться дальше, протестировать и запустить в production (полностью или частично), получив свой первый «взрослый» опыт эксплуатации [4].*
По определению некоторых архитекторов Kubernetes это: «Платформа, предоставляющая уровень абстракции, забирающий у вас какую-либо проблему, чтобы вы могли творить поверх неё (не думая о ней)». Примеры: платформа Linux даёт возможность исполнять системные вызовы вне зависимости от аппаратного обеспечения компьютера, а платформа Java – исполнять приложения вне зависимости от операционной системы. Какой же должна быть платформа для запуска приложений, созданных по принципам микросервисной архитектуры? Об этом и пойдет речь в этой статье, затрагивая обзорное представление о возможностях и целях использования Kubernetes как платформы для оркестрации контейнеров.

Ключевые слова: *Kubernetes, Docker, контейнеры, оркестрация контейнеров*

APPLYING OF CONTAINERS ORCHESTRATION SYSTEM “KUBERNETES”

Abstract: *While many of us are looking closely at Kubernetes and assessing its capabilities and maturity, others have time to move forward, test and run in production (fully or partially) and to receive their first “adult” operating experience.[4]*
Some architects define Kubernetes as “A platform providing a level of abstraction that takes away any problem from you, so that you can create on top of it (not thinking about it).” For example: The Linux platform allows you to make system calls regardless of the computer hardware, and the Java platform allows you to run applications regardless of the operating system. What the platform for launching applications which is based on microservice architecture should be like? This conversation will continue in the article, affecting the overview of the possibilities and goals of using Kubernetes as a platform for orchestrating containers.

Keywords: *Kubernetes, Docker, containers, container orchestration*

КОНТЕЙНЕРЛЕРДІ ОРКЕСТРЛЕУДЕ “KUBERNETES” ЖҮЙЕСІН ҚОЛДАНУ

Аңдатпа: *Көптеген адамдар Kubernetes-ке қарап, оның мүмкіндіктері мен жетілуін бағалағанымен, басқалары алға жылжып, өндірісте (толық немесе ішінара) алғашқы «ересек» операциялық тәжірибесін алып, сынақтан өтуге және іске асыруға уақыт алады.[4]*
Kubernetes-тің кейбір архитекторлары анықтағандай, «Кез келген мәселені шешетін және мәселе туралы сізді уайымнан үзіп абстракция деңгейін беретін платформа». Мысалдар: Linux платформасы компьютерлік жабдыққа қарамастан, жүйелік қоңыраулар жасауға және Java платформасы есептік жүйеге қарамастан, қосымшаларды іске қосуға мүмкіндік береді. Микросервис архитектурасына негізделген қосымшаларды іске қосу үшін қандай платформа болуы керек екендігі жөнінде мәселе көтеріледі. Осыған сәйкес Kubernetes контейнерлерді басқару платформасы ретінде пайдалану мүмкіндіктері мен мақсаттарына жан-жақты шолу жасалады.

Түйінді сөздер: *Kubernetes, Docker, контейнерлер, контейнерлерді оркестрлеу*

Kubernetes – движок оркестровки контейнеров, созданный для запуска контейнеризированных приложений на множестве узлов, которые обычно называют кластером.[1]

Ключевые характеристики таких платформ – портируемость и расширяемость. Каждая облачная платформа предлагает свои варианты для достижения этих целей. Однако облачные платформы создают жесткую привязку для своих клиентов и перенос какого-либо проекта с одной среды на другую (в публичном облаке, в своём дата-центре, на серверах клиента…) сопровождаются большими проблемами либо совершенно не решаемы.[5]

Суть Kubernetes как платформы, то есть по-настоящему универсальной системы для развёртывания приложений, физическое размещение которых может производиться где и как угодно: на голом железе, в публичных или частных облаках вне зависимости от их разработчиков и специфичных API. Но здорово в Kubernetes не только то, **где** запускать, но и **что**: ведь это могут быть приложения на разных языках и под разные ОС, они могут быть stateless и stateful (об этом я опишу ниже). Поддерживается принцип «если приложение может запускаться в контейнере, оно должно отлично запускаться в Kubernetes»[5]

О том, что Kubernetes не является традиционной PaaS,рассказываетсяв документации проекта, где поясняется, что авторы стремятся сохранить возможность пользовательского выбора в местах, где это важно. В частности:

- Kubernetes не предлагает никаких встроенных служб для обмена сообщениями, обработки данных, СУБД и т.п.
- Kubernetes не имеет своего магазина с готовыми сервисами для деплоя в один клик.
- Kubernetes не деплоит исходный код и не собирает приложения. Процессы непрерывной интеграции (CI) поддерживаются, но их реализация оставлена для других инструментов.
- Аналогично для систем журналирования и мониторинга.

Таким образом, если в PaaS обычно делается акцент на предоставление функциональных возможностей, то в Kubernetes первичен универсальный, абстрактный подход. Несмотря на то, что Kubernetes предлагает ряд функций, которые традиционно присущи PaaS: развёртывание приложений, масштабирование, балансировка нагрузок, журналирование и т.п., — платформа является модульной и предлагает пользователям самим выбирать конкретные решения для тех или иных задач. Такой подход сделал Kubernetes базой для такого PaaS, как OpenShift от RedHat.[5]

Работающий кластер Kubernetes(Рис. 1) включает в себя агента, запущенного на нодах (kubelet) и компоненты мастера (APIs, scheduler, etcd), поверх решения с распределённым хранилищем. При взгляде на архитектуру системы мы можем разбить его на сервисы, которые работают на каждой ноде и сервисы уровня управления кластера. На каждой нодеKubernetes запускаются сервисы, необходимые для управления нодой со стороны мастера и для запуска приложений. Конечно, на каждой ноде запускается Docker, который обеспечивает загрузку образов и запуск контейнеров. Kubelet управляет ресурсами pod-а, их контейнерами, образами, разделами, и т.д. Также на каждой ноде запускается простой proxy-балансировщик. Этот сервис запускается на каждой ноде и настраивается в Kubernetes API. Kube-Proxy может выполнять простейшее перенаправление потоков TCP и UDP (roundrobin) между набором бэкендов. Состояние мастера хранится в экземпляре etcd. Это обеспечивает надёжное хранение конфигурационных данных и своевременное оповещение прочих компонентов об изменении состояния. Kubernetes API обеспечивает работу api-сервера. Он предназначен для того, чтобы быть CRUD сервером со встроенной бизнес-логикой, реализованной в отдельных компонентах или в плагинах. Он, в основном, обрабатывает REST операции, проверяя их и обновляя соответствующие объекты в etcd (и событийно в других хранилищах). Scheduler привязывает незапущенные pod-ы к нодам через вызов /binding API (Рис. 2).

Pod-ы считаются базовыми строительными блоками Kubernetes, потому что все рабочие нагрузки в Kubernetes — например,

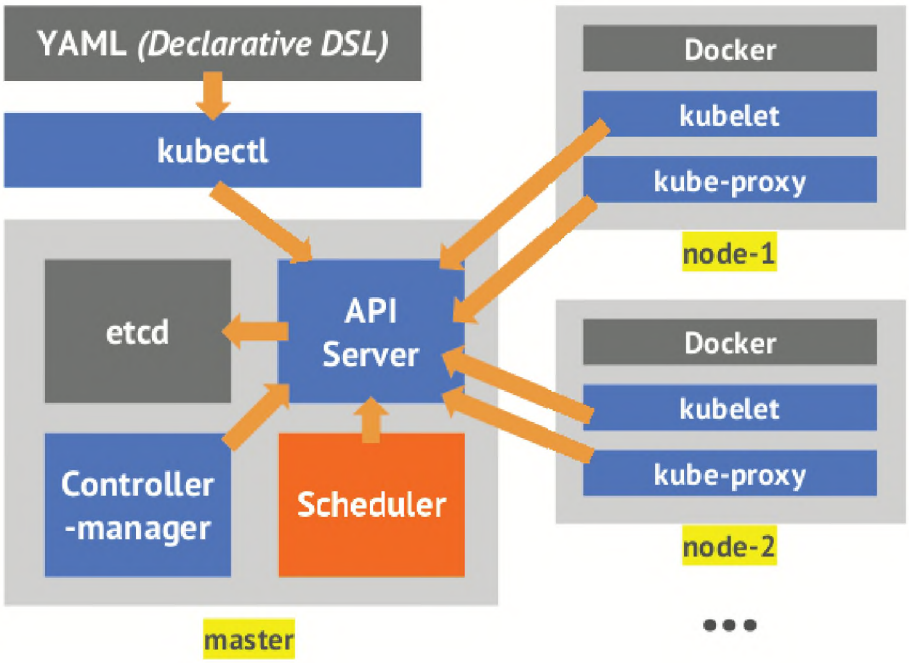

Рис. 1 – Архитектура Kubernetes

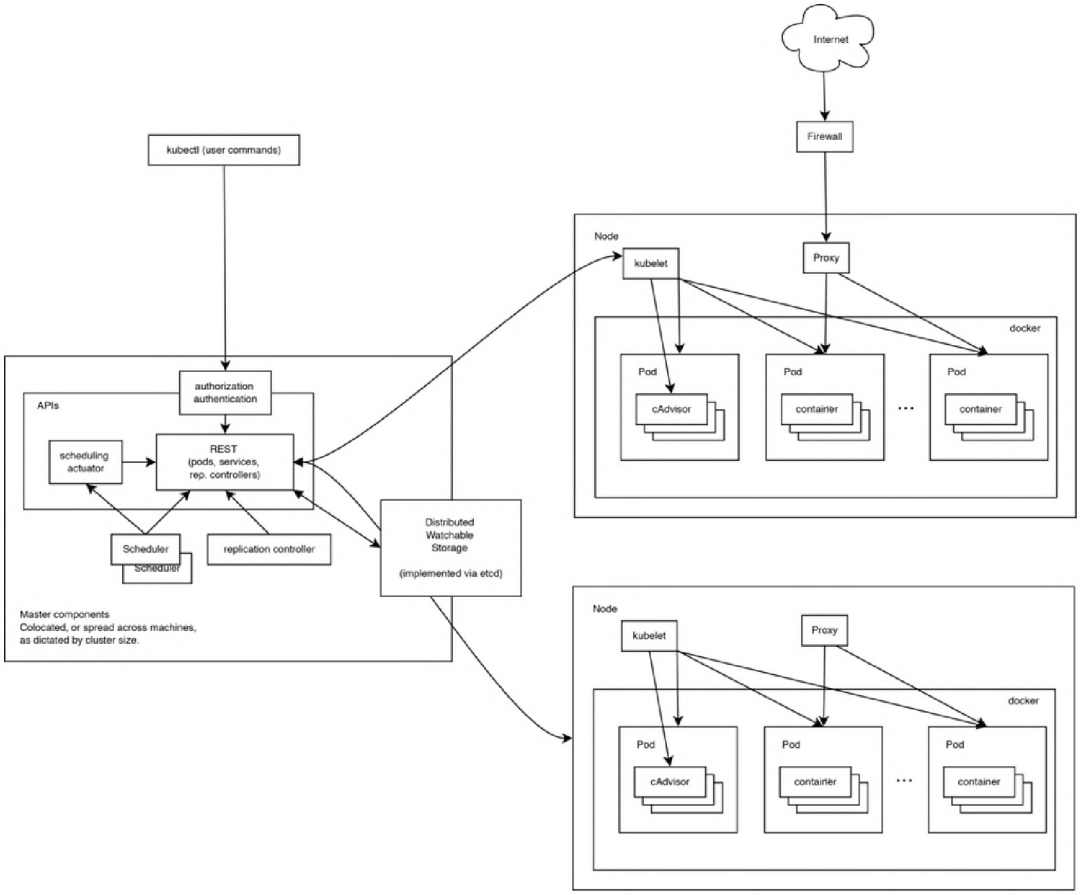

Рис. 2 – компоненты ноды

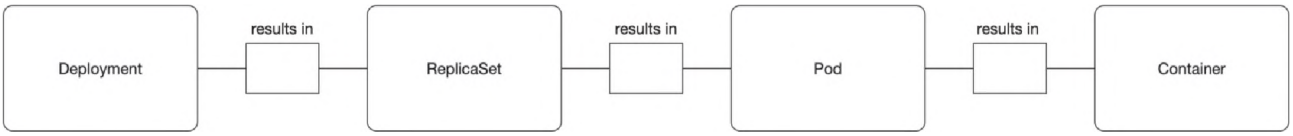

Рис. 3 – Deployment, ReplicaSet, pod и контейнеры

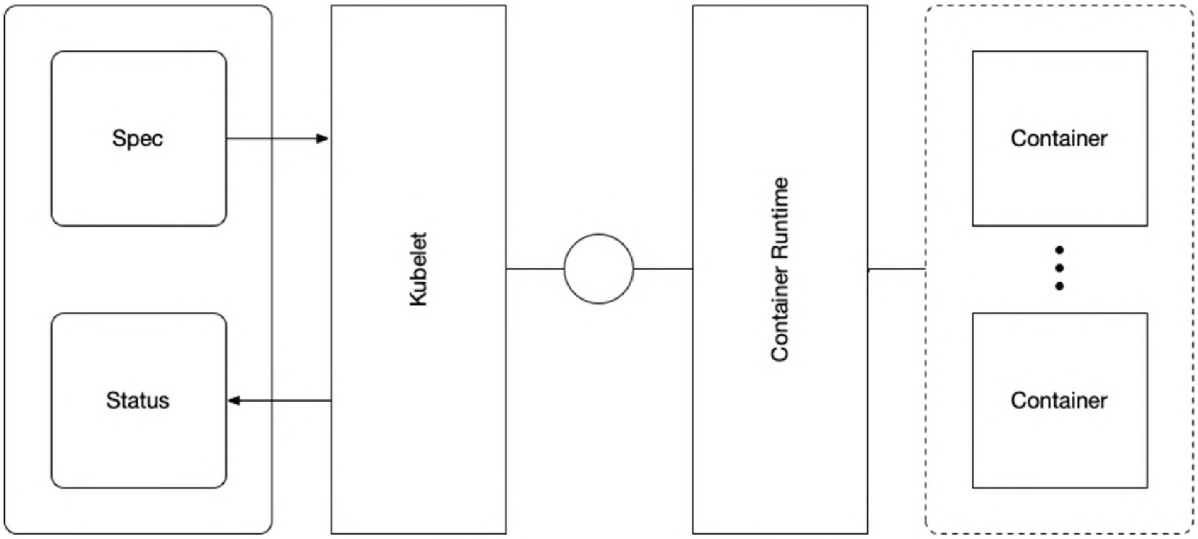

Схема 3 – Взаимодействие Kubelet с объектом pod-а и исполняемой средой контейнера (containerruntime)

Deployments, *ReplicaSets*и *Jobs*— могут быть выражены в виде pod-ов.[1]

Pod — это один и единственный объект в Kubernetes, который приводит к запуску контейнеров (Рис.3). Нет pod-а — нет контейнера![1]

Как говорилось ранее есть stateless и stateful, к первым относятся приложения, не зависящие от персистентности данных, например, веб- приложения и веб-сайты. Ко вторым относятся приложения, зависящие от персистентности данных, как например, базы данных и системы журналирования, которым важно не терять данные. Такое деление возникает по причине принципа работы pod-ов в Kubernetes. Pod-ы могут быть потеряны в случае аварии или по правилам политики масштабирования. И исходя от архитектуры приложения, выбирается необходимый с персистентным хранилищем или без. Высокая доступность достигается с помощью горизонтального масштабирования и возможности миграции pod-а на другие ноды. При пересоздании pod-а у него меняется IP адрес и имя (кроме statefulset). А распределение трафика происходит на виртуальный адрес, балансирующий на pod-ы в зависимости от меток, назначаемых на каждый pod. Такой компонент называется Service. По сути- это статический IP адрес перед группой pod-ов.

Описание компонентов в Kubernetes производится декларативно в манифест файле в yaml формате.

```
---
apiVersion: v1
kind: Pod
metadata:
  name: test-site
  labels:
    app: web
spec:
  containers:
    - name: front-end
      image: nginx
      ports:
        - containerPort: 80
    - name: test
      image: ealebed/test:v1
ports:
- containerPort: 88
```

Таким образом, можно описать архитектуру приложения в виде кода. Возникает следующий вопрос о том, как масштабировать и запускать контейнеры сразу на большом количестве хостов Docker, а также как выполнять их балансировку. Вся прелесть Kubernetes заключается в автоматическом управлении процессом разворачивания в кластер приложений посредством команды kubectl и API сервера. Создав файл манифест с необходимым описанием, kubectl посылает их на исполнение, и в зависимости от контейнера приложение может быть доступно от нескольких секунд до нескольких минут (при использовании локального registry сервера время скачивания значительно меньше). В случае отказа одного из нод кластеров pod-ы, находившиеся на нем, переезжают на другие ноды, при правильном масштабировании конечный пользователь не почувствует, что произошел отказ сервиса, так как помимо умершего pod-а доступны другие pod-*ы* из этой же группы.

Особый момент обретает система хранения в Kubernetes. Они разделяются на распределенные локальные и удаленные. Единственная возможность использовать statefulset в продуктиве – распределенная система хранения, например ceph. С использованием такого хранилища нодыKubernetes избавляются от необходимости хранить данные для работы приложений кластера локально, а при миграции stateful приложений на другую ноду, данные будут так же доступны, так как хранилище представляется на всех workerнодах.[6]

Выводы

Текущий подход эксплуатации инфраструктуры все активнее меняется в сторону контейнеров и самым популярным лидером среди систем оркестрации является Kubernetes. Новый инструмент нацелен на ускорение и упрощение развертывания приложений. А при использовании дополнительных наборов инструментов процесс становится автоматизированным и требует меньшего количества обслуживающего персонала. Достаточно крупные мировые компании, как CERN, Ebay, Huawei[3] и т. д. уже используют Kubernetes в продуктивной эксплуатации. [2] Для обучения этому инструменту доступны подробная документации и программы (minikube) для запуска мини кластера на своем компьютере. Проект бурно развивается, и все больше оно приобретает обширный набор функционала. А для разработки используется язык программирования go.

REFERENCES

1. https://habr.com/ru/company/flant/blog/427819/
2. https://itnan.ru/post.php?c=1&p=412571
3. https://habr.com/en/company/flant/blog/349940/
4. https://habr.com/en/company/flant/blog/334140/
5. https://habr.com/en/company/flant/blog/327338/
6. https://habr.com/ru/company/flant/blog/329666/