

Engineer

Kubernetes

⁂ система, що підтримує роботу безлічі контейнерів



Що це таке

Kubernetes, який часто скорочують до K8s, — це програмне забезпечення з відкритим кодом, що запускає контейнеризовані застосунки на групі машин, яку називають кластером, і постійно тримає їх відповідно до заданого опису: який образ запускає кожен застосунок, скільки копій має існувати, скільки пам'яті й процесорного часу може брати кожна копія і як до неї звертаються інші програми. Власна документація проєкту називає його переносною й розширюваною платформою для керування контейнеризованими навантаженнями та сервісами. Назва походить із грецької й означає стерничого або лоцмана, вісімка в K8s рахує літери між K та s, а Google відкрив код проєкту 2014 року.

Кластер складається з двох частин. Площина керування зберігає опис: API-сервер — єдиний HTTP-інтерфейс, через який проходить кожна зміна, etcd зберігає все, що йому передали, планувальник призначає кожен новий под на відповідний вузол, а менеджер контролерів запускає цикли, які діють згідно з описом. На кожному робочому вузлі kubelet стежить, щоб призначені туди поди працювали, kube-proxy підтримує мережеві правила, на яких тримаються об'єкти Service, а середовище виконання контейнерів, як-от containerd чи CRI-O, запускає самі контейнери. Docker Engine перестав бути вбудованим середовищем виконання в Kubernetes 1.24, проте образи, зібрані за допомогою Docker, працюють без змін, бо це звичайні образи OCI.

Опис складається з об'єктів, які зазвичай пишуть у YAML. Под (Pod) — найменша одиниця, яку розміщує Kubernetes: один чи кілька контейнерів, що ділять мережеву адресу й сховище на одній машині. Deployment задає, скільки однакових подів має існувати, і поступово замінює їх, коли змінюється їхній шаблон. Service дає мінливій групі подів одне стале ім'я та адресу й добирає учасників за мітками. Невеликий вебзастосунок виглядає так:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - name: web
          image: registry.example.com/web:1.4.2
          resources:
            requests:
              cpu: 250m
              memory: 256Mi
            limits:
              memory: 256Mi
          readinessProbe:
            httpGet:
              path: /healthz
              port: 8080
```

Kubernetes зазвичай порівнюють із трьома меншими варіантами. Docker Compose описує контейнери на одній машині, і разом із машиною зникають і вони. Менеджер служб на кшталт systemd може перезапустити контейнер після збою й запустити його під час завантаження системи; цього вистачає для дуже багатьох потреб за значно меншого обсягу механізмів, але переносити роботу на іншу машину він не вміє. Хмарні платформи на кшталт Google Cloud Run чи Azure Container Apps запускають контейнери й не передають вам площину керування в експлуатацію, натомість налаштувань там менше. Документація також прямо каже, що Kubernetes відрізняється від традиційної всеосяжної PaaS: він не збирає ваш застосунок і не постачає власних баз даних чи черг повідомлень.

Застосуйте поняття до рішення

Застосуйте «Що таке Kubernetes?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Запакована так, що поводиться однаково на ноутбуці, на тестовому сервері й у продакшні, і масштабується, коли настає напружений тиждень.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Вела 20 застосунків Docker у Google Cloud і AWS — 20 контейнеризованих застосунків розгорнуто й поставлено під моніторинг

Вела 20 застосунків Docker у Google Cloud і AWS Зв'язатися з нами →

Чому це важливо

Головна ідея в тому, що кластер постійно рухається до записаного опису, а не виконує список команд один раз. Застосування файлу вище напряму нічого не запускає; воно фіксує бажаний стан. Контролери — це цикли керування, які стежать за кластером і діють так, щоб наблизити поточний стан до бажаного. Якщо машина виходить з ладу і з трьох копій лишаються дві, контролер Deployment помічає різницю, створюється новий под, а планувальник розміщує його на машині, де є місце. А оскільки опис — це файли, він може лежати в репозиторії й проходити той самий перегляд, що й код, який він запускає.

Релізи вбудовано в саму платформу. Типова стратегія Deployment — поступове оновлення: нові поди стартують, а старі зупиняються в міру того, як нові стають готовими, у межах, що за замовчуванням становлять 25% понад потрібну кількість і 25% нестачі водночас. Кожна зміна шаблону пода зберігається як ревізія, за замовчуванням десять останніх, а `kubectl rollout undo` повертає попередню. Скрипт, який колись виводив сервер з роботи, міняв бінарний файл і повертав сервер у стрій, стає вбудованою поведінкою.

Планувальник змінює й економіку парку машин. Він розміщує кожен под за процесорним часом і пам’яттю, які той запитує, тож багато дрібних сервісів ділять спільний набір машин, а не отримують кожен окремий сервер, розрахований на найгірший день.

Ціна цілком реальна. Кластер сам є розподіленою системою з версіями, які треба оновлювати за графіком, мережевим рівнем, рівнем зберігання, сертифікатами й контролем доступу, і хтось мусить розуміти все це, коли воно поводиться дивно. Керовані сервіси на кшталт Google Kubernetes Engine, Amazon EKS та Azure Kubernetes Service обслуговують площину керування за вас, але кожен маніфест, а зазвичай і робочі вузли, лишаються вашими. Для кількох сервісів на одній чи двох машинах звичайний менеджер служб часто справляється з меншою кількістю рухомих частин.

Як це виглядає на практиці

Сторінка технологій компанії проводить чітку межу: Docker і Kubernetes беруть для навантажень, яким вони справді потрібні, а для решти вистачає звичайних юнітів systemd. Kubernetes обирають під розмір і форму задачі, а не за звичкою.

Навантаження першого типу видно в одному проєкті. Двадцять Docker-застосунків працювали в Google Cloud та AWS під постійним супроводом, серед них аналітичні застосунки на R, з однаковою конфігурацією й моніторингом, щоб кожен застосунок жив за тими самими правилами. Несправності розбирали через Podman, а проблеми оркестрації — через Kubernetes. Сторінка повідомляє про швидшу діагностику й про розгортання, які лишалися стабільними й щоразу давали той самий результат.

Друга половина правила — на власній платформі компанії. Контейнерна площина на Podman і Quadlet запускає кожен контейнер як службу під власним наглядачем операційної системи, описану в тій самій декларативній формі, що й решта хоста. Контейнери користуються мережею хоста й не публікують портів, тож правила брандмауера, які встановлює роль посилення захисту, і далі мають останнє слово. Непривілейований користувач випускає нову версію, запускаючи один фіксований скрипт, що збирає образ і перезапускає службу. На одному хості це дає кожному контейнерові записаний опис і наглядача, якому хост уже довіряє, без другого менеджера процесів. Поза цією схемою лишаються розміщення на кількох машинах і заміна вузла, що вийшов з ладу, і саме там Kubernetes починає виправдовувати свою вагу.

Де виникають проблеми

Більшість проблем із Kubernetes виникає в проміжку між тим, що просить маніфест, і тим, що повідомляє кластер.

Розгортання застрягає на півдорозі, а CI-завдання звітує про успіх. Коли нові поди так і не стають готовими — через хибний тег образу, збій під час запуску, readiness-пробу, що не проходить, чи вичерпану квоту, — Deployment перестає просуватися. Після граничного терміну, за замовчуванням 600 секунд, він записує причину `ProgressDeadlineExceeded`, і, за документацією, більше Kubernetes нічого не робить: старі поди, що ще

працюють, і далі обслуговують запити, а сам собою відкат не відбувається. `kubectl rollout status` завершується з ненульовим кодом, щойно термін минув, тож крок розгортання може дочекатися його й відкотити реліз:

```
kubectl apply -f web.yaml
kubectl rollout status deployment/web --timeout=10m \
  || kubectl rollout undo deployment/web
```

Контейнер перезапускається з `OOMKilled` або сповільнюється на вузлі з вільним процесором. Ліміти забезпечує ядро, і два ресурси поводяться по-різному. Контейнер, що перевищив ліміт пам'яті, ядро може завершити, коли відчуває нестачу пам'яті, після чого його перезапускають; контейнер, що вперся в ліміт процесора, пригальмовують, і він працює повільніше, не падаючи, що схоже на ваду самого застосунку. Планувальник розміщує поди за тим, що вони запитують, а не за тим, що споживають, тож низькі запити переповнюють вузол, а високі лишають машини без діла. Ліміт без запиту сам стає запитом. Пам'ять варто задавати з виміряного споживання, а ліміти процесора ставити свідомо.

Застосунок, що довго стартує, вбивають раніше, ніж він встигає піднятися. Коли liveness-проба не проходить, kubelet вбиває контейнер і запускає його знову, тож застосунок, якому на старт треба хвилина, а перевіряють його через десять секунд, перезапускається без кінця. Startup-проба стримує інші перевірки, доки застосунок не підніметься. Readiness-проба виводить под зі складу Service, не перезапускаючи його, і це пасує поду, який зайнятий чи прогріває кеш. Liveness-проба, яка звертається до бази даних, перезапускає всі репліки одночасно, щойно база сповільнюється, і повільна залежність перетворюється на збій.

Под, запущений вручну, зникає, а ручне виправлення скасовується. Под, створений сам по собі, лишається на своєму вузлі, доки не завершиться, не буде видалений чи витіснений або доки не відмовить вузол, і жоден контролер його не відтворить; документація радить Deployment чи Job навіть для однієї копії. Зміна, зроблена через `kubectl edit`, існує лише в кластері, і наступний apply з репозиторію повертає старе значення, тож аварійне виправлення має потрапити у файли, щойно аварія минула.

Що почитати далі

- Kubernetes overview — власний опис проєкту: що таке Kubernetes, що він пропонує і чого не бере на себе.
- Pods — найменша одиниця, яку розміщує Kubernetes, її життєвий цикл і чому поди рідко створюють вручну.
- Deployments — поступові оновлення, відкати, історія ревізій і граничний термін просування.
- Service — сталі імена й адреси перед мінливою групою подів.
- Resource Management for Pods and Containers — запити, ліміти і як ядро забезпечує кожен із них.
- Configure Liveness, Readiness and Startup Probes — три проби і як захистити контейнер, що довго стартує.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке Kubernetes?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/kubernetes/