3.4 Твій Kubernetes кластер

З визначення **образ** містить інформацію, достатню для запуску програми на цільовій платформі: команда, аргументи, змінні оточення тощо. Ця специфікація визначає як створити OCI-образ. Результатом збірки є маніфест образу, серіалізація файлової системи та конфігурація образу. Маніфест образу містить метадані про вміст і залежності образу, включно з архівом файлової системи, який буде розпакований для створення фінальної файлової системи контейнера, який можна запустити.

Маніфест — це набір з одного або декількох архівів та інструкції як саме і в якому порядку їх розархівувати, щоб отримати файлову систему. Наприклад, ми починаємо з порожнього образу, далі копіюємо якийсь набір файлів (шар перший), змінюємо потрібним нам чином файлову систему (кожна зміна окремий шар), і, в підсумку, записуємо покрокову інструкцію як з порожнього образу створити те, що нам потрібно.

Створення Container images — це важливий етап у розробці та розгортанні додатків. Для досягнення максимальної ефективності та забезпечення безпеки додатків, слід дотримуватися наступних найкращих практик та порад.

Використовуйте офіційні базові образи
Офіційні базові образи контейнерів мають визначені правила версіонування, оновлення та безпеки, тому вони є надійними та безпечними. Використовуйте їх, коли це можливо, або користуйтеся образами, які базуються на офіційних базових образах.

Оновлюйте базові образи регулярно
Базові образи містять важливі пакети та бібліотеки, тому вони повинні бути оновлювані регулярно.

Не включайте зайві файли та пакети
Включайте у свій образ тільки необхідні файли та пакети, а не все, що є в вашій робочій директорії. Це допоможе зменшити розмір образу та зробити його більш ефективним.

Використовуйте кешування пакетів
Використовуйте кешування пакетів, щоб зменшити час збірки образу та зробити його більш ефективним.

Використовуйте аргументи збірки
Використовуйте аргументи збірки, щоб передавати значення змінних під час збірки образу. Це допоможе зробити ваш образ більш гнучким та налаштовуваним.

Використовуйте мітки образів
Використовуйте мітки образів, щоб ідентифікувати та відстежувати версії образів. Це допоможе зробити процес розгортання більш простим та ефективним.

Забезпечте безпеку образів
Перевірте свій образ на вразливості та дотримуйтеся найкращих практик безпеки. Використовуйте механізми контейнеризації, щоб забезпечити ізольованість та безпеку контейнерів.

Документуйте образи
Документуйте свої образи, щоб інші розробники могли легко розуміти, які пакети та бібліотеки входять до образу та як він працює.

Коли образ контейнеру готовий, можна приступити до його розгортання на Kubernetes.

Варіантів **встановлення Kubernetes** цілий список. Але якщо розібратися — по суті Kubernetes це ж процеси. На яких ресурсах і

архітектурі запускати процеси — не принципово, за винятком деяких нюансів.

Можливі варіанти:

Перший. Це може бути ваш локальний сетап для навчання, розробки або експериментів. **Сетап** — це процедура налаштування середовища для готовності до запуску необхідних програм. У цьому випадку вам знадобиться більш-менш потужний комп'ютер або ноутбук.

Здебільшого потрібна оперативна пам'ять і місце на диску. Нам знадобитися розміщувати контейнер іміджі як на диску так і в пам'яті. Це все з огляду на вашу операційну систему з графічним інтерфейсом, рантайм енжін і, звісно ж, браузер, щоб читати і дивитися навчальні матеріали.

Тут буде достатнім варіант Single Node Cluster. Роль мастера і воркера поєднані на одній ноді. Але використовуючи сучасні інструменти, ви не обмежені — з легкістю запускайте повноцінний multinode кластер навіть на своєму лептопі.

K3d - найпростіший, гнучкий і легкий варіант для старту. Ця система розроблена під docker. Тобто, ми запускаємо Kubernetes ноди як контейнери. Як можна помітити, вже на етапі вибору сетапу ми починаємо знайомитися з перевагами Kubernetes. Перша і головна з цих переваг — Kubernetes повний агностик. Ми не залежимо від інфраструктури та середовища виконання.

Другий варіант — це віддалені віртуальні або bare metal ресурси. Простіше кажучи, сервери, розташовані за межами вашого локального оточення. Залізний фізичний сервер, що стоїть поруч на столі, недорогий комп'ютер розміром із кредитну картку Raspberry Pi, орендовані сервери в місцевого провайдера в дата-центрі, віртуальні

сервери хмарного провайдера. Нам підходить для запуску Kubernetes буквально все, що має пам'ять, процесор, мережу і мінімальний розмір диска.

Варіант номер три. Hosted service Kubernetes — це Cloud Provider сервіс, тобто, керований вашим хмарним провайдером сервіс. Ви за нього платите. Як за управління местер нодами, так і за кожну воркер ноду. Переваги в повній автоматизації запуску, супроводу життєвого циклу кластера. Патчі безпеки, автоматичне оновлення версій, надання SLA, вертикальне масштабування, HA або режим високої доступності, резервування та багато іншого. Підходить як для тренувань і навчання, так і для повноцінного продакшн варіанту.

Хороший варіант познайомиться з хмарною інфраструктурою. Особисто моя рекомендація — спробуйте почати з GCP. Для вашого першого знайомства google надає всі умови і кредит на навчання, який не потрібно повертати. Просто зареєструйтеся і використовуйте безоплатно всі необхідні ресурси в рамках певного часу або на надану суму кредиту.

Ще варіант — кастомний Kubernetes кластер на хмарному провайдері. Як і хостед сервіс — очікувано, ніякої магії — все крутиться на серверах. Тільки в цьому випадку всі процеси контролюєте ви самостійно. Напевно, найбільш нетривіальний підхід, але й найефективніший для навчання. Отже, ми створюємо декілька віртуальних серверів і далі налаштовуємо, встановлюємо, конфігуруємо наш кластер самостійно. Тут для нас два шляхи: легкий — постав свій кластер за 60 секунд; і складний Kubernetes the Hard Way, до речі, за авторством Келсі Хайтауера, покроковий сценарій розгортання кластера з нуля.

Отже, мій сетап наступний. Операційна система macOS, 8 гігабайт оперативної пам'яті, дискового простору багато не передбачається, процесор intel i7 або M1. І, звичайно, доступ в інтернет — нам потрібні контейнер іміджі, тобто, доступ до одного з реєстрів, наприклад hub.docker.com.
На машині встановлений і запущений докер з індивідуальною ліцензією для особистого користування. Якщо використовувати google клауд сервер, все необхідне вже встановлено.

Локально я буду використовувати k3s від ранчер. k3s — це single binary production ready Kubernetes кластер. Як для майстра, так і воркер нод. Встановлення всього набору займе не більше двох хвилин.

Підготовка Dockerfile, тегування образу, push у GCR та запуск на Minikube в Github Codespace:

Клонуйте або відкрийте репозиторій у Github Codespace.

Створіть новий файл Dockerfile у кореневій директорії вашого репозиторію.

Відкрийте файл Dockerfile у редакторі та почніть додавати інструкції для збірки вашого Docker-імеджа. Наприклад, ви можете почати з базового імеджа та додати необхідні залежності та програми.

Після того, як ви завершили підготовку Dockerfile, переконайтеся, що він містить всі необхідні інструкції для збірки вашого імеджа.

Збережіть файл Dockerfile та відкрийте термінал.

У терміналі виконайте наступні команди для збірки вашого Docker-імеджа:

`docker build -t` <ім'я-імеджа>
де <ім'я-імеджа> - ім'я, яке ви хочете дати вашому імеджу.

Тегуйте свій Docker-імедж за допомогою команди:

`docker tag` <ім'я-імеджа> `gcr.io/`<проєкт>`/`<ім'я-імеджа>`:`<версія>
де <ім'я-імеджа> - ім'я вашого Docker-імеджа, <проєкт> - назва вашого проекту в GCR, <версія> - версія вашого Docker-імеджа.

Завантажте ваш Docker-імедж до GCR за допомогою команди:

`docker push gcr.io/`<проєкт>`/`<ім'я-імеджа>`:`<версія>

Запустіть Minikube у Github Codespace за допомогою наступної команди:
`minikube start`

Переконайтеся, що Minikube успішно запущений, виконавши наступну команду:
`minikube status`

Запустіть ваш Docker-імедж в Minikube за допомогою команди:
`kubectl create deploy` <ім'я-деплойменту> `--image=`<ім'я-імеджа>