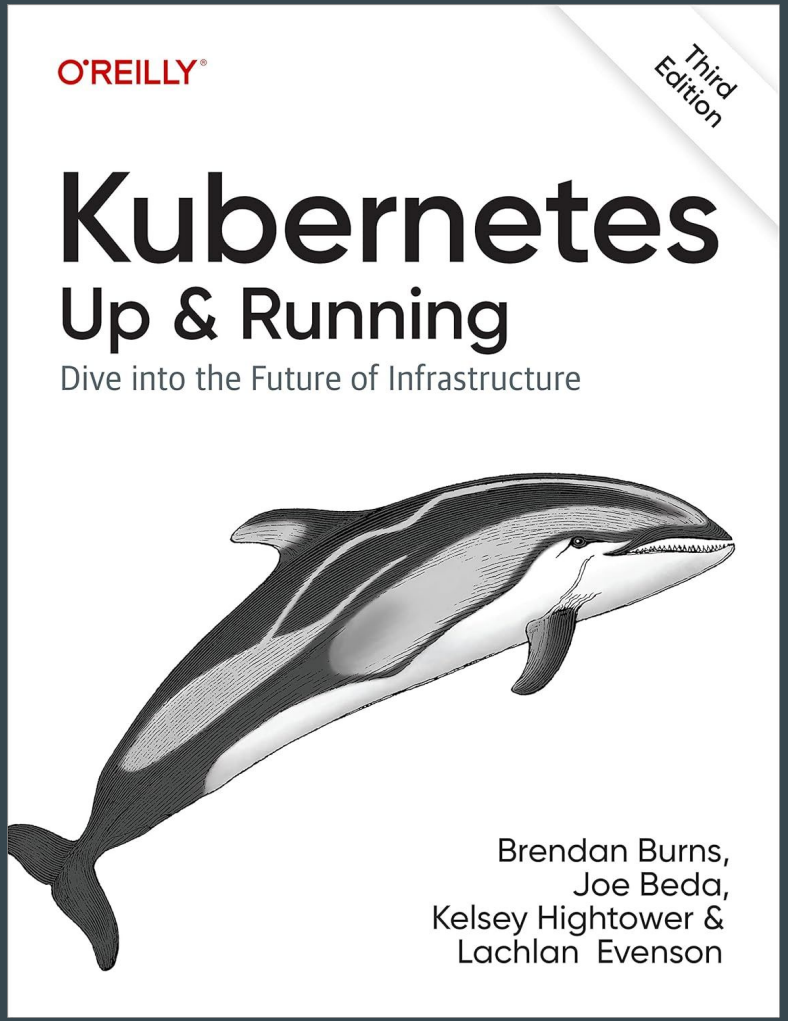
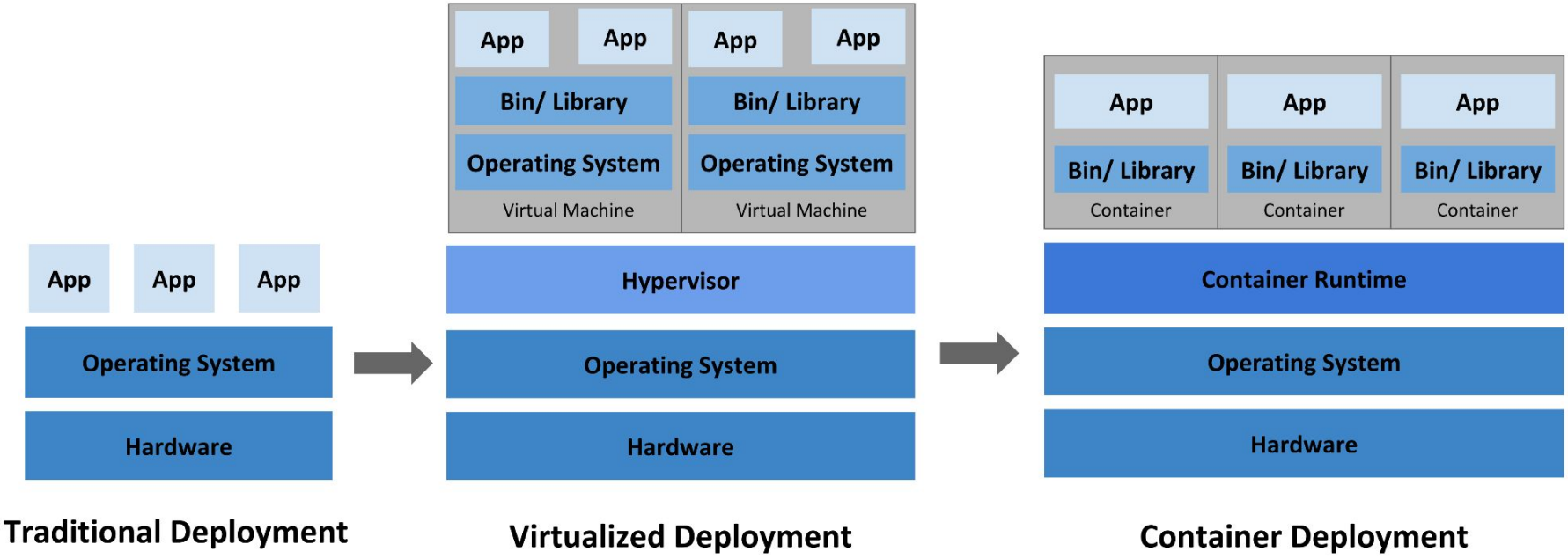
Kubernetes

Bartosz Szczeciński


O'REILLY
Third Edition
Kubernetes
Up & Running
Dive into the Future of Infrastructure
Brendan Burns,
Joe Beda,
Kelsey Hightower &
Lachlan Evenson

Czym jest kontener?



https://kubernetes.io/docs/concepts/overview/

Problem - zarządzanie zbiorem kontenerów (mikroserwisów)

- Restartowanie kontenerów w przypadku crasha / deadlocka
- Przeniesienie kontenera na inną maszynę (VM) w przypadku awarii sprzętu
- Skalowanie zależne od natężenia ruchu (duplikacja kontenerów)
- Konfiguracja sieci
- Dostęp do konfiguracji całego klastra w jednym miejscu
- Rollout/rollback wszystkich kontenerów

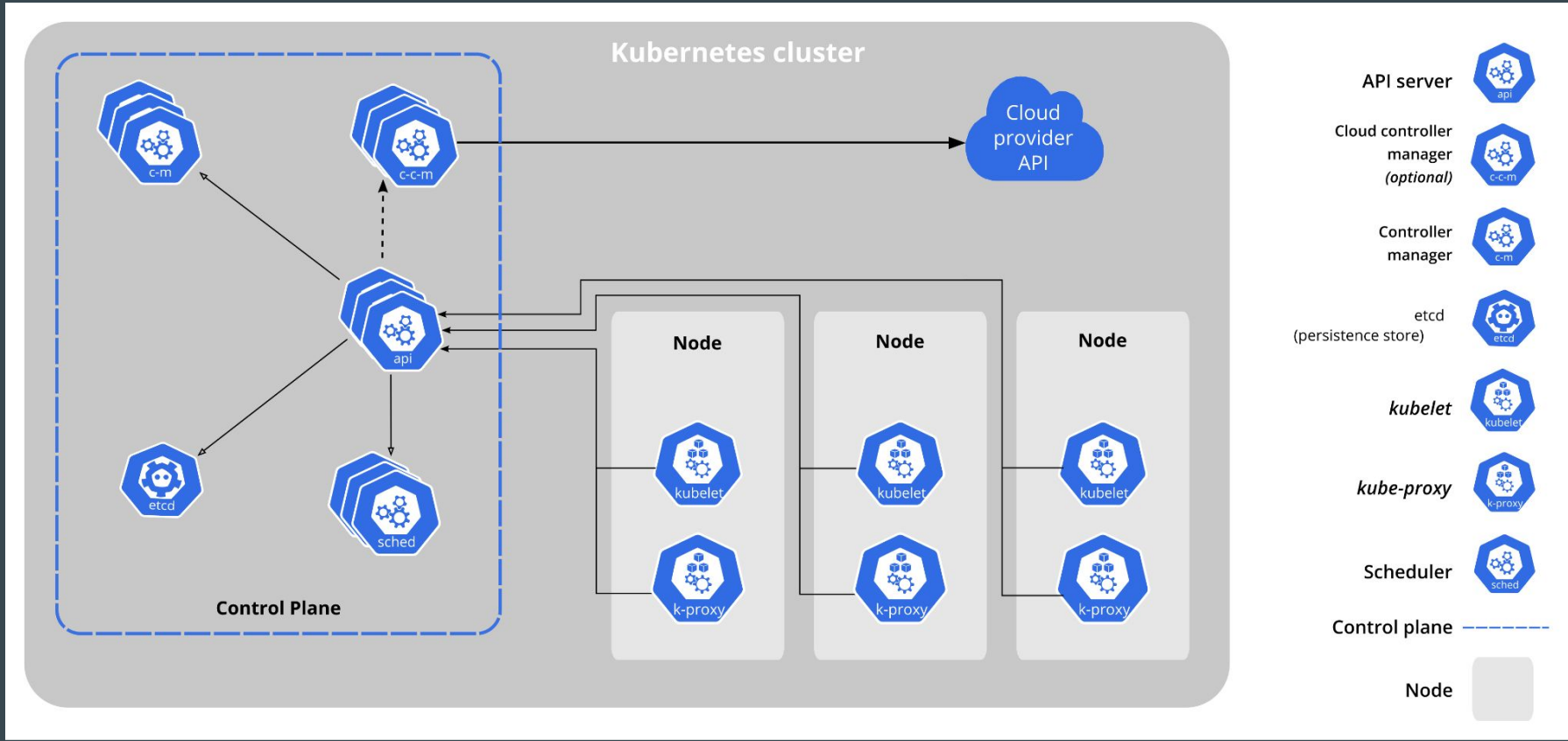
Historia

- 2004: Borg - Rozwiązanie closed source od Google'a do zarządzania klastrami
- 2015: Kubernetes 1.0 - Stworzone przez Google, od początku open source

    Powstanie Cloud Native Computing Foundation, które zarządza projektem
- 2016: Helm - package manager dla Kubernetes
- 2017: Wsparcie od VMWare, Docker, Microsoft Azure, AWS

Filozofia Kubernetes

Deklaratywny kod specyfikuje pożądany stan systemu, a kubernetes robi wszystko, żeby go osiągnąć



https://kubernetes.io/docs/concepts/overview/components/

Pod

- Zbiór kontenerów i woluminów (volumes, np. baza danych) działających na jednej maszynie (node)
- Najmniejsza jednostka w Kubernetes
- Każdy pod ma osobne IP
- Każdy kontener wewnątrz poda jest w osobnej cgroupie, ale dzieli niektóre namespace'y z innymi (Domyślnie ipc, net, uts są dzielone)

1-kuard-pod-simple.yaml

Health checks

- Liveness probe - restartuje kontener po kilku porażkach
- Readiness probe - po kilku porażkach load balancer nie przekierowywuje do poda
- Możemy sami zdefiniować jak Kubernetes ma sprawdzać zdrowie poszczególnych kontenerów w podach. Np. wywołanie endpointa REST API, uruchomienie jakiegoś skryptu w kontenerze, wywołanie endpointa RPC...

Zarządzanie zasobami

- Możemy zażądać minimalnego, gwarantowanego procenta CPU i pamięci
- Możemy określić maksymalne zużycie CPU i pamięci - dany pod nie dostanie więcej

2-kuard-pod-full.yaml

ReplicaSet

- Tworzy kopie danego poda
- Zarządza wieloma pod'ami

3-kuard-rs.yaml

Deployment

- Rolling update
- Podgląd historii i łatwy powrót do poprzednich wersji
- Zarządza między innymi wieloma ReplicaSet'ami

4-kuard-deployment.yaml

Service discovery

- Problem: które procesy słuchają na jakich adresach pod jakimi portami?
- Rozwiązanie: Stwórz serwis, w którym każdy pod automatycznie dostaje label identyfikujący serwis (np. app=my-app)
- Żeby dostać się do konkretnego serwisu, wystarczy znać jego port i dostać się do dowolnego node'a, a ten przekieruje nas do poda (być może na innym node), gdzie znajduje się dany serwis. To wykorzystuje kube-proxy oraz wewnętrzny serwis dns.
- Alternatywnie osobny load balancer (tylko w chmurze)

Service discovery demo

DaemonSet

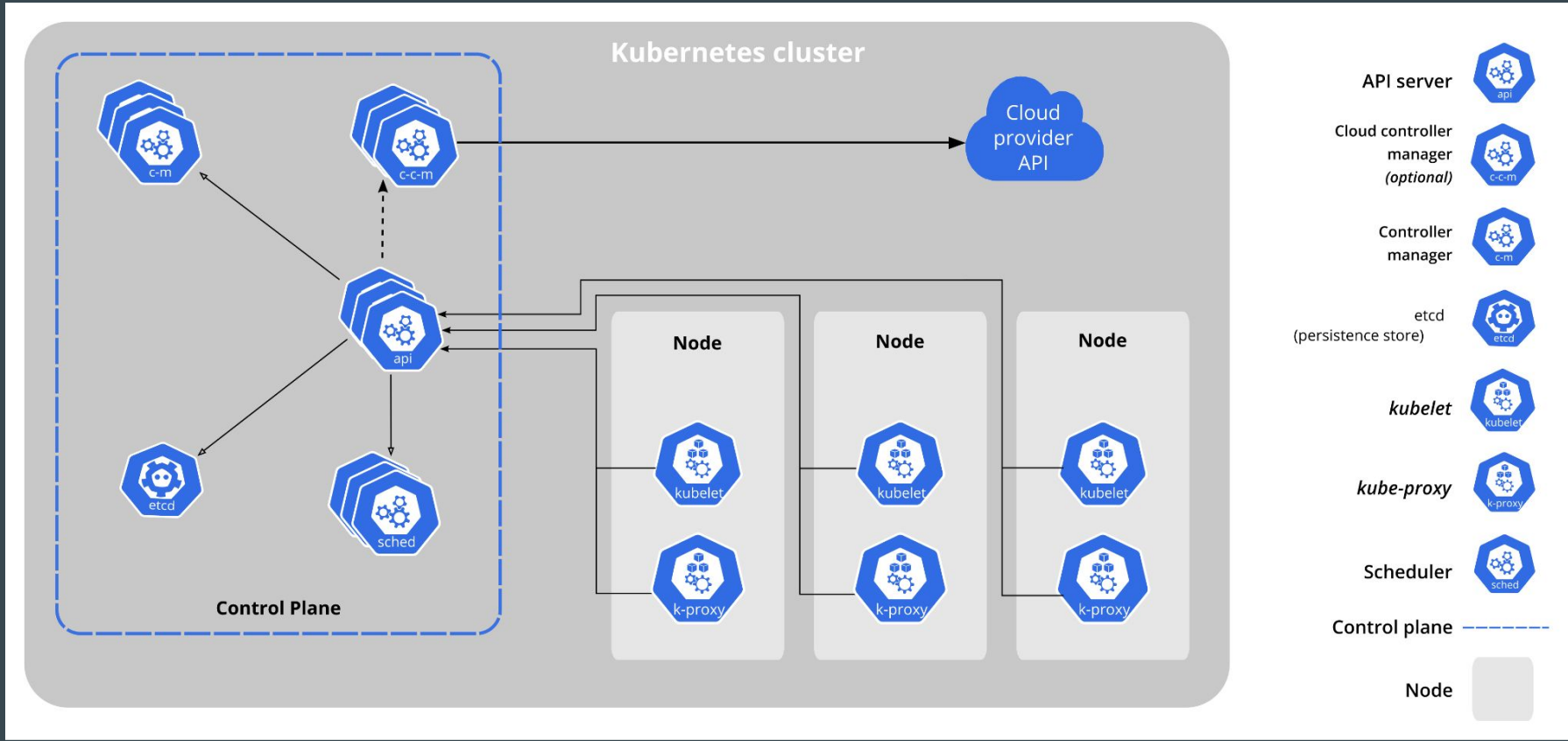
- Automatycznie umieszcza po jednej kopii danego pod'a na każdym node
- Dobre np. do zbierania logów, rozbudowanego monitorowania każdego node'a

5-fluentd-daemonset.yaml

Jobs

- Pojedyncze wykonanie
- Wykonanie zadań z kolejki i wyjście
- CronJob

6-job-oneshot.yaml



https://kubernetes.io/docs/concepts/overview/components/

Co ma Kubernetes czego nie ma Docker Swarm?

- Wsparcie dla innych typów kontenerów niż docker (Containerd, CRI-O, ...)
- Automatyczne skalowanie (np. na podstawie zużycia procesora)
- Automatyczne zarządzanie sekretami
- RBAC (Role-based access control)
- Automatyczny self-healing
- Duże wsparcie społeczności (pluginy)
- Dostępne jako gotowy serwis u popularnych dostawców chmurowych (GCP, AWS, Azure)
- Wbudowany monitoring i dashboard

Zalety Docker Swarm

- Prostszy, szybszy do nauki
- Wbudowany w docker engine, zintegrowany z narzędziami dockerowymi
- Automatyczny load balancing
- Idealny do mniejszych projektów