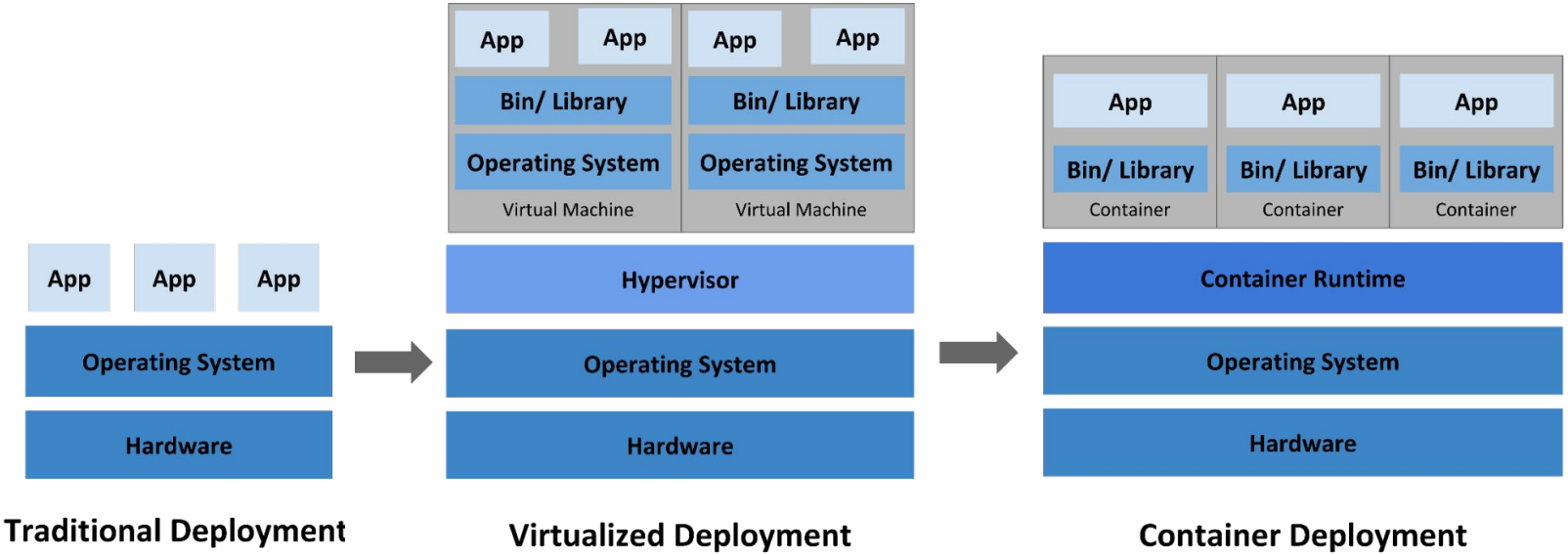
Orkiestracja kontenerów

Bartosz Wojciechowski

Plan szkolenia

- Konteneryzacja – przypomnienie
- Orkiestracja kontenerów
- Kubernetes – główne funkcje
- Dystrybucje K8s
- Zarządzanie klastrem K8s
- **Live demo – uruchomienie aplikacji na K8s**
- Rodzaje zadań w K8s
- Udostępnianie usług z K8s
- Wybrane przydatne funkcje i narzędzia
- Podsumowanie

Model tradycyjny vs maszyny wirtualne vs konteneryzacja



Źródło: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes

Konteneryzacja

- Lekka forma wirtualizacji
- Izolacja zadań w ramach tego samego systemu operacyjnego
- Kontener = gotowe środowisko uruchomieniowe aplikacji
  - kod
  - zależności (np. biblioteki, pakiety)
  - konfiguracja
- Oddzielenie danych od aplikacji
- Kontener tworzony jest z obrazu
- Definicja obrazu – plik Dockerfile

Przykładowy Dockerfile

```
FROM node:14-alpine
WORKDIR /usr/src/app

COPY package*.json ./
RUN npm ci

COPY index.js ./
COPY src/ ./src/

USER node
EXPOSE 3000

CMD ["npm", "start"]
```

Uproszczony cykl życia obrazu i kontenera

Demo?

Obraz

1. Przygotowanie konfiguracji obrazu – Dockerfile
2. Zbudowanie obrazu na podstawie Dockerfile
3. Wgranie obrazu do zewnętrznego repozytorium (opcjonalne)

Kontener

1. Pobranie obrazu z repozytorium lokalnego lub zewnętrznego
2. Utworzenie kontenera
3. Uruchomienie kontenera

Pets vs cattle (zwierzątka domowe vs bydło)

- **Kiedyś**: serwery, o które się dbało
- **Teraz**: zamiast naprawiać serwer łatwiej postawić nowy (automatyzacja, np. Ansible)
- Kontenery są z założenia efemeryczne
  - Nie należy wprowadzać zmian w kontenerze
  - Usunięcie kontenera nie powinno skutkować utratą danych

Trudności związane z konteneryzacją

- Stanowość (np. sesja)
- Persistent storage
- Bezpieczeństwo
- Zarządzanie wieloma kontenerami
- Zbieranie logów
- ...

Źródło: https://www.ceoutlook.com/wp-content/uploads/2020/11/Rena-ship.gif

Orkiestratory kontenerów

- Warstwa abstrakcji
- Zarządzanie kontenerami
  - uruchamianie, rozmieszczanie
  - monitorowanie
  - skalowanie poziome, load balancing
- Konfiguracja
  - sieci pomiędzy kontenerami
  - polityk sieciowych
  - przestrzeni dyskowej
  - przydział zasobów (RAM i CPU)
  - udostępniania usług
- Często możliwość wykorzystania wielu maszyn
- Kontrola dostępu

Przykłady orkiestratorów

- Docker Compose
- Docker Swarm
- Kubernetes (w skrócie K8s)
- HashiCorp Nomad

Kubernetes

Główne funkcje Kubernetesa (1)

- Zarządzanie kontenerami
- *Immutable infrastructure*, *infrastructure as code* (IaC)
- Wewnętrzny load balancing między instancje kontenerów
- Udostępnianie usług na zewnątrz
- Współpraca z *reverse proxy* / *application proxy*
- Możliwość wykorzystania wielu maszyn (node’ów)

Główne funkcje Kubernetesa (2)

- Przydział i ograniczenie zasobów CPU/RAM
- Polityki sieciowe
- Przydzielanie zasobów dyskowych z wykorzystaniem zewnętrznych rozwiązań
- Możliwość automatycznego skalowania poziomego (proste na podstawie wykorzystania CPU lub bardziej zaawansowane metryki)
- Wsparcie dla *high availability*, *fault tolerance*

Dystrybucje Kubernetesa

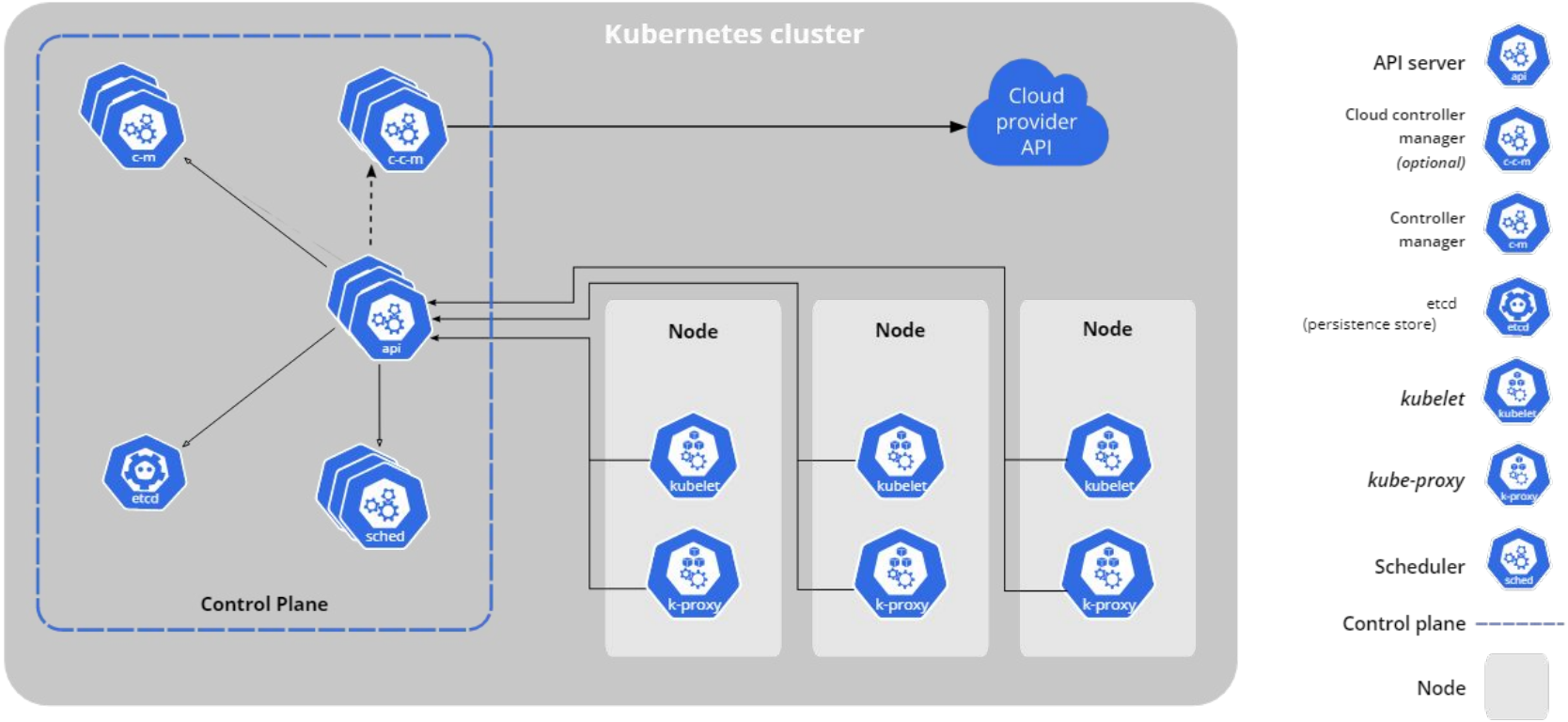
- Różne wersje Kubernetesa
- Organizacja CNCF certyfikuje dystrybucje – gwarancja zgodności
- Wybrane dystrybucje:
  - Kubernetes
  - Minikube (dev, automatycznie tworzy VM VirtualBox/HyperV/inne)
  - K3s (produkcyjny, lekki, łatwy do uruchomienia)
  - MicroK8s

K3S

CLOUD NATIVE
COMPUTING FOUNDATION

Podstawowe pojęcia

- Klaster
- Worker node
- Control plane
- Kontener
- Pod
- Replika, instancja (ang. *replica*)
- Service (w kontekście k8s)



Architektura Kubernetesa

Źródło: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes

Zarządzanie klastrem

- Koncepcja konfiguracji za pomocą zasobów (obiektów)
  - Pod
  - Deployment
  - Service
- Narzędzia
  - Kubectl
  - Web Dashboard
  - Lens
- Mechanizm RBAC

Live demo

na przykładzie aplikacji NodeJS

- Zbudowanie obrazu
- Wgranie obrazu do repozytorium
- Uruchomienie aplikacji na K8s
- Udostępnienie aplikacji na zewnątrz klastra
- Skalowanie poziome

Wybrane rodzaje zadań (ang. workloads)

- Deployment
  - aplikacje bezstanowe (stateless)
  - aplikacje dostosowane do częstego usuwanie/dodawanie instancji
- StatefulSet
  - podobny do Deploymentu
  - każdy pod jest identyfikowalny, ma tożsamość
  - aplikacje stanowe (stateful)
- DaemonSet
  - aplikacje przypisane do konkretnych node’ów
  - na przykład monitoring lub agent agregatora logów
- Job
  - jednorazowe zadanie
  - uruchamiane jako pod

Sposoby udostępniania usług

- Dostęp wyłącznie z wewnątrz klastra
  - Service typu *ClusterIP*
- Dostęp z zewnątrz
  - Service
    - typu *NodePort*
    - typu *LoadBalancer*
  - Ingress controller

Przydatne funkcje, narzędzia i dodatki

- Przechowywanie konfiguracji
  - ConfigMaps – zamiast pliku
  - Secrets
- Mechanizm namespace
- Node taints
- Helm
- Rozproszone systemy plików
  - MooseFS
  - Ceph

Potencjalne problemy, wyzwania

- Utrzymanie klastra
  - np. aktualizacje, monitorowanie
  - wymaga nakładów pracy
  - bardziej opłacalne mogą być gotowe rozwiązania
- Zarządzanie uprawnieniami dostępu do klastra
- Narzut wydajnościowy wykorzystania orkiestratora
- Więcej: Kubernetes Failure Stories https://k8s.af/

Podsumowanie

- Kontenery umożliwiają szybkie uruchamianie odizolowanych aplikacji
- Kubernetes ułatwia zarządzanie środowiskiem kontenerów i zwiększa dostępne możliwości
- Nie zawsze wykorzystanie konteneryzacji jest proste i nie zawsze stanowi najlepsze rozwiązanie
- Dostępne są usługi, które ułatwiają zarządzanie klastrem Kubernetes

Źródła i dodatkowe materiały

- https://sysdig.com/blog/dockerfile-best-practices/
- https://docs.docker.com/engine/reference/builder
- https://kubernetes.io/docs/home/
- https://kubernetes.io/docs/reference/glossary/
- https://www.cncf.io/blog/
- https://www.stackrox.com/post/2020/01/kubernetes-networking-demystified/
- https://rancher.com/docs/k3s/latest/en/

Dziękuję za uwagę!

Pytania?