Kubernetes

Kurs DevOps – Wykład 4

Kuba Nowak

Uniwersytet Wrocławski

12 listopada 2025

Kubernetes

Z greki: $\kappa\upsilon\beta\epsilon\rho\nu\eta\tau\eta\varsigma$ – zarządca, sternik.

Definicja dla zarządu

Otwartoźródłowa platforma do zarządzania, automatyzacji i skalowania aplikacji kontenerowych. Udostępnia zestaw narzędzi, które zapewniają mechanizmy wdrażania, utrzymywania i skalowania programów w oparciu o procesor, pamięć lub niestandardowe parametry.
Na podstawie: `https://pl.wikipedia.org/wiki/Kubernetes`

Kubernetes

Z greki: $\kappa\upsilon\beta\epsilon\rho\nu\eta\tau\eta\varsigma$ – zarządca, sternik.

Definicja dla zarządu

Otwartoźródłowa platforma do zarządzania, automatyzacji i skalowania aplikacji kontenerowych. Udostępnia zestaw narzędzi, które zapewniają mechanizmy wdrażania, utrzymywania i skalowania programów w oparciu o procesor, pamięć lub niestandardowe parametry.
Na podstawie: `https://pl.wikipedia.org/wiki/Kubernetes`

Definicja nieoficjalna

Program do włączania i wyłączania aplikacji w jak najbardziej skomplikowany sposób.

Po co nam Kubernetes?

Kubernetes zapewnia:

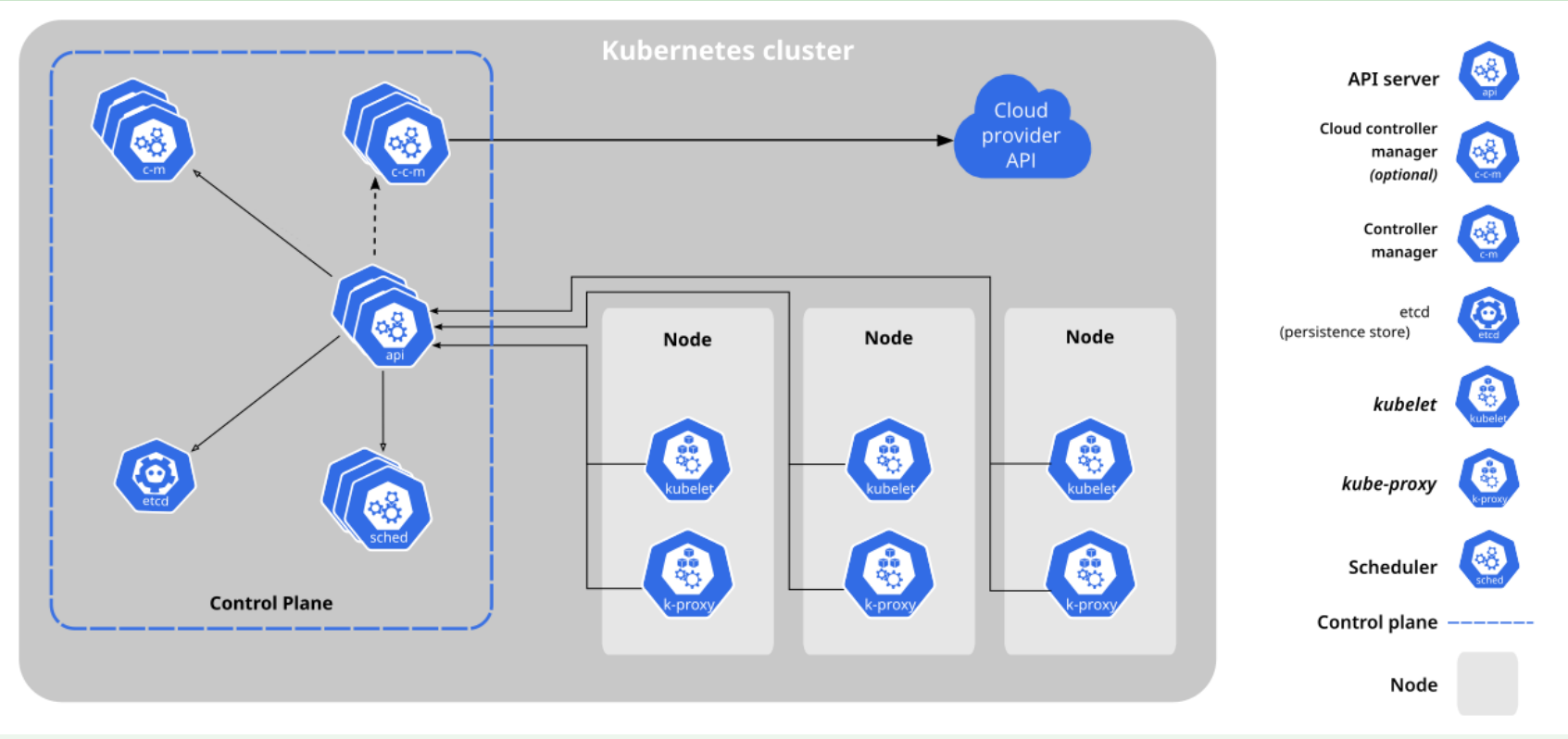
- Wykrywanie aplikacji
- Równoważenie obciążenia (ang. *load balancing*)
- Zarządzanie przestrzenią dyskową
- Automatyczne aktualizacje
- Automatyczne przydzielanie podów
- Akcje naprawcze
- Zarządzanie konfiguacją i sekretami
- Wykonanie zadań jednorazowych
- Skalowanie horyzontalne
- Rozszerzalność

Funkcjonalności

- Jeden byt uruchamialny (pod)
- Cztery sposoby uruchomienia: (*Deployment*, *StatefulSet*, *DaemonSet*, *Job*)
- Możliwość ograniczania zużycia cpu i pamięci RAM.
- Możliwość wystawienia portów do świata zewnętrznego.

Działanie kubernetesa

- Deklaratywny.
- Obiekty służą jako „record of intent”.
- Wykorzystuje istniejące implementacje kontenerów.
- Ograniczanie zasobów przy pomocy cgroup.
- Tworzony z mikroserwisów
- Projektowany do uruchamiania mikroserwisów



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

Budowa klastra

Klaster zbiór maszyn, na których można uruchamiać aplikacje użytkownika wraz z infrastrukturą do zarządzania nimi.

Control plane warstwa zarządzająca klastrem.

Node (czasami *worker* bądź *minion*) pojedyncza maszyna (fizyczna bądź wirtualna).

Źródło: Universal Pictures (materiały prasowe)

Budowa klastra

Klaster zbiór maszyn, na których można uruchamiać aplikacje użytkownika wraz z infrastrukturą do zarządzania nimi.

Control plane warstwa zarządzająca klastrem.

Node (czasami *worker* bądź *minion*) pojedyncza maszyna (fizyczna bądź wirtualna).

Biblioteka uruchomieniowa służy do uruchamiania kontenerów przykładowo: containerd, CRI-O, Docker Engine.

Etcd nieulotna, rozproszona baza przechowująca pary klucz-wartość, zawiera konfigurację i stan klastra Kubernetesa.

Kontroler odpowiada za zmienianie stanu klastra w kierunku stanu zadeklarowanego w konfiguracji. Odpowiada za zlecanie odpowiednich operacji.

Proxy przekazuje ruch sieciowy między podami i innymi węzłami/światem zewnętrznym.

Kubelet nadzoruje i konfiguruje węzeł wraz ze wszystkimi kontrolerami.

Kontroler chmury służy zintegrowaniu klastra z chmurą wybranego dostawcy.

Demonstracja 1

Uruchomienie klastra

Warstwa zarządzania

Składniki warstwy zarządzającej:

- kube-apiserver
- etcd
- kube-scheduler
- cloud-controller-manager
- kube-controller-manager
- kube-proxy
- network plugin

Komunikacja

Konfiguracja odbywa się poprzez REST API wysyłające JSON-y do serwera HTTP. Wewnętrznie Kubernetes też używa takich wiadomości.

Representational State Transfer (REST)

Zestaw reguł, które powinno się stosować podczas projektowania API:

- Podział klient/serwer wraz z dobrze określonym interfejsem.
- Bezstanowość
- Cacheable
- Warstwowość systemu
- Jednorodny interfejs
  - zasób i reprezentacja są niezależne
  - jeśli klient ma reprezentacje zasobu, to ma wystarczająco informacji do wykonania dowolnej akcji
  - wiadomości wysyłane do klienta powinny mieć wystarczająco informacji by wiedzieć jak je przetworzyć (samoopisujące się)
  - odkrywalność zasobów zależnych
- (Opcjonalnie) Wykonanie kodu po stronie klienta

REST API

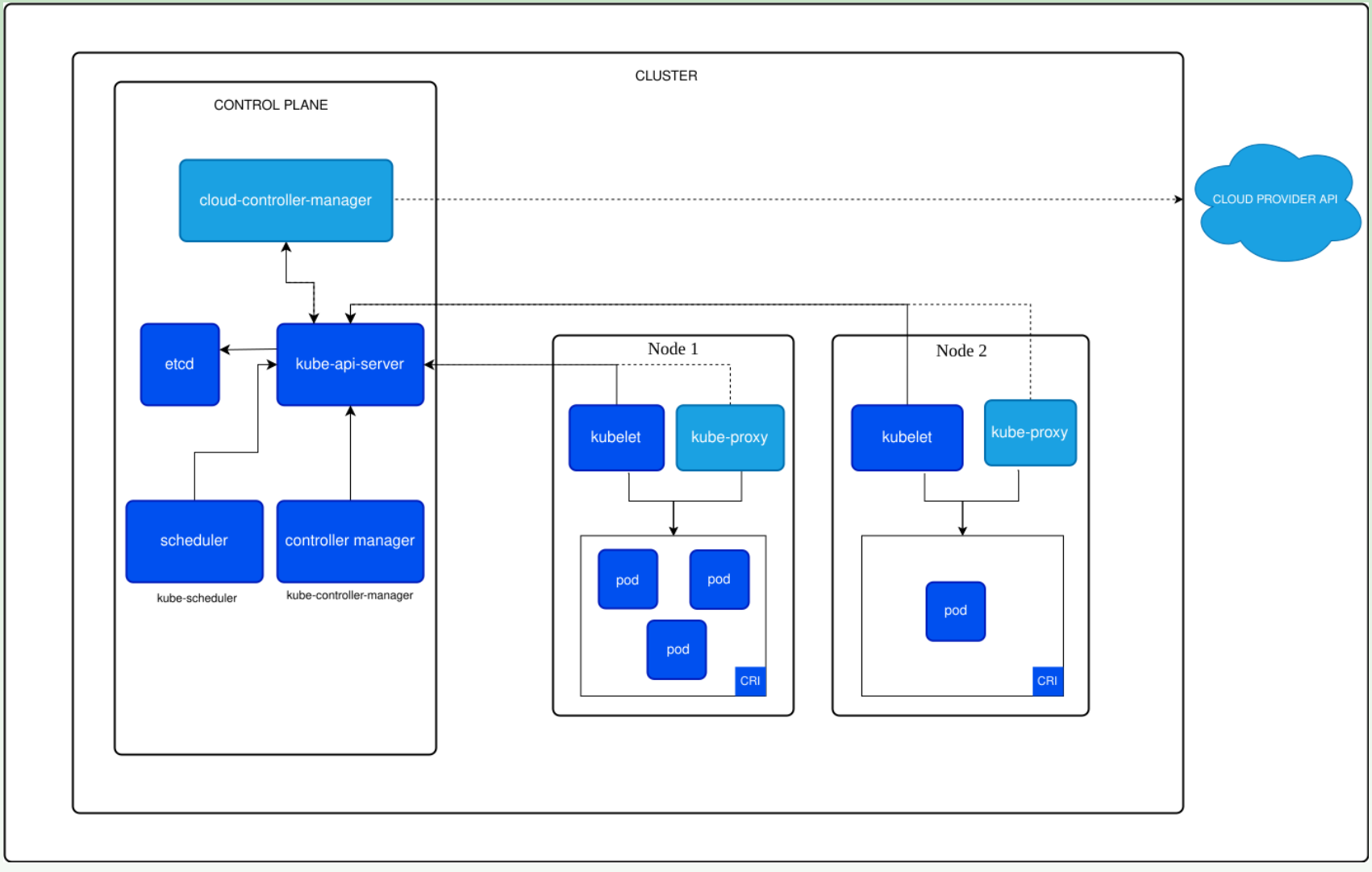
Najczęściej:

- Zapytania z użyciem HTTP.
- JSON używany do reprezentacji danych.
- Zasoby i akcje określone poprzez URL.
- Klucz/token w nagłówku.

Przykład użycia:

```
curl --request POST \
--url "https://api.github.com/repos/octocat/Spoon-Knife/issues" \
--header "Accept: application/vnd.github+json" \
--header "X-GitHub-Api-Version: 2022-11-28" \
--header "Authorization: Bearer YOUR-TOKEN" \
--data '{ "title": "Created with the REST API",
          "body": "This is a test issue created by the REST API" }'
```

Źródło: https://docs.github.com/en/rest/using-the-rest-api/getting-started-with-the-rest-api



Źródło: `https://kubernetes.io/docs/concepts/architecture/`

Obiekty

Klaster kontroluje zasoby obliczeniowe i pamięciowe, definiując je jako obiekty, którymi następnie można zarządzać. Kluczowymi obiektami są:

Pod najmniejsza jednostka atomowo uruchamiana w Kubernetesie. Składa się z jednego bądź więcej kontenerów.

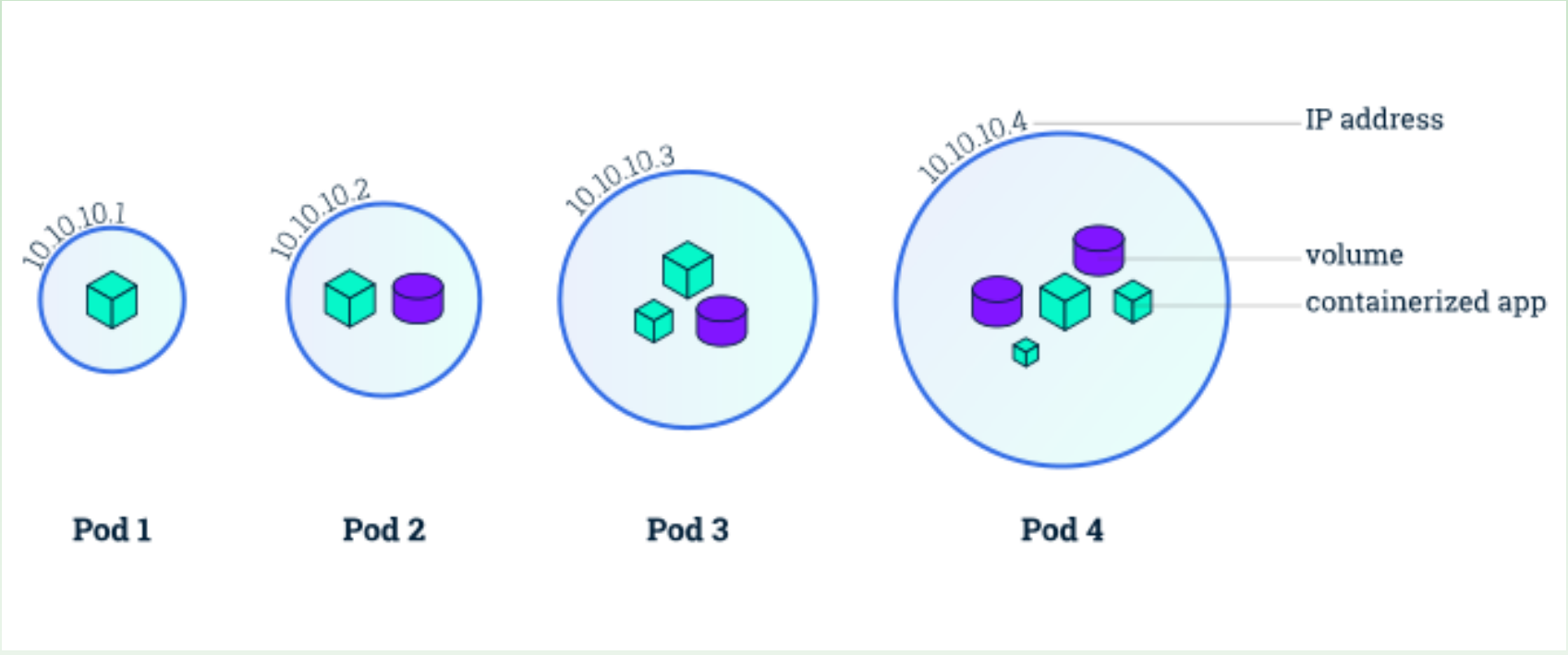
Service to abstrakcja pozwalająca ukryć zbiór podów i przedstawiać się na zewnątrz jako jedna aplikacja.

Wolumen pamięć masowa niezależna od kontenera. Może być powiązana z podem (ulotna) bądź stanowić odrębny byt.

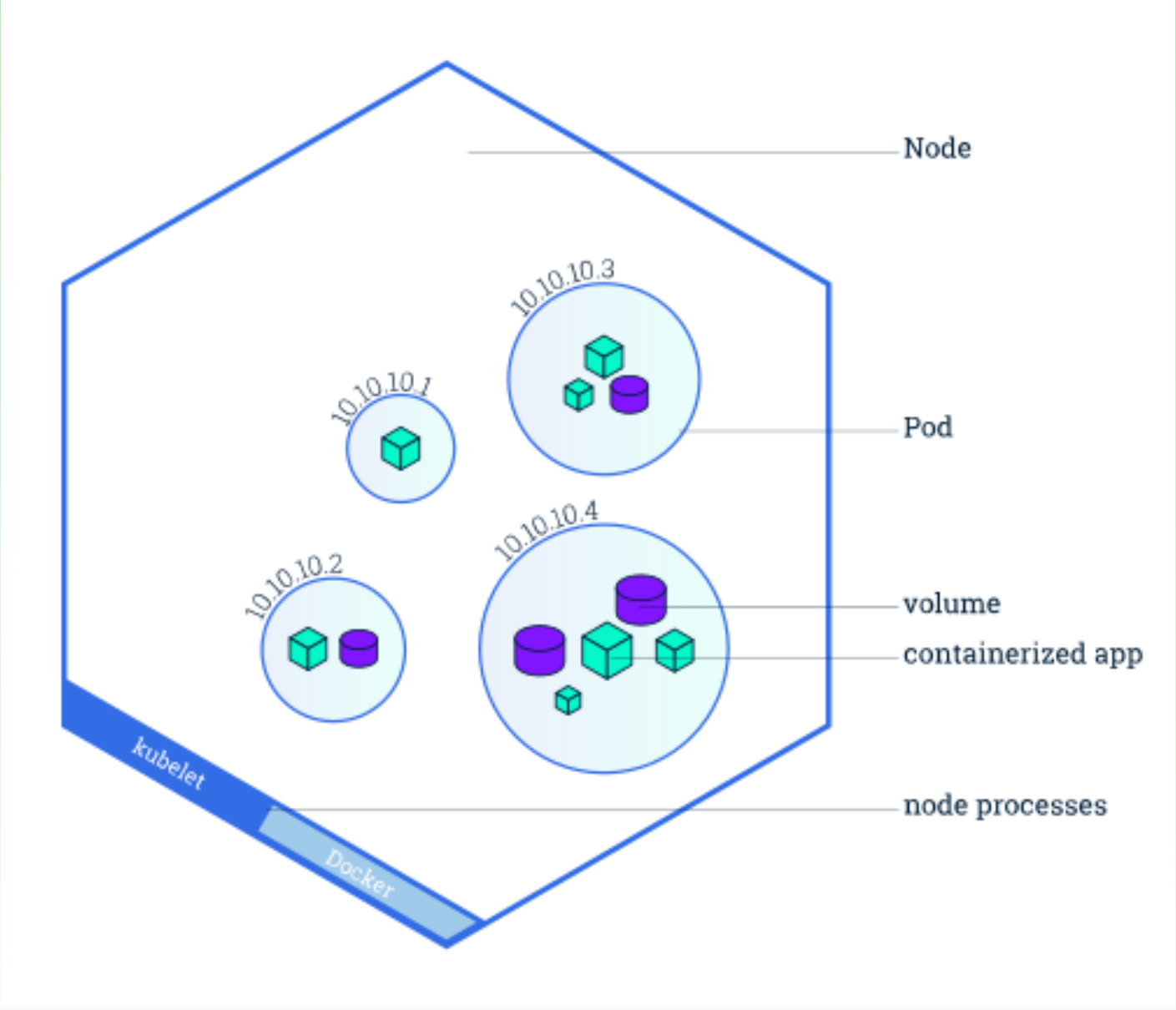
ConfigMap konfiguracja aplikacji, przechowywana niezależnie od niej, jako cześć stanu klastra.

Sekret podobny jest do ConfigMap, ale przeznaczony do przechowywania danych wrażliwych.

Zasoby są zgrupowane w *przestrzenie nazw* w celu wydzielenia niezależnych środowisk (np. dla różnych projektów, użytkowników, etapów życia dostarczenia).



Źródło:
https://kubernetes.io/pl/docs/tutorials/kubernetes-basics/explore/explore-intro/



Źródło:
https://kubernetes.io/pl/docs/tutorials/kubernetes-basics/explore/explore-intro/

Sposoby uruchamiania

Deployment to domyślny sposób uruchomienia aplikacji, która nie zależy od stanu i ma działać bez końca (np. serwer HTTP).

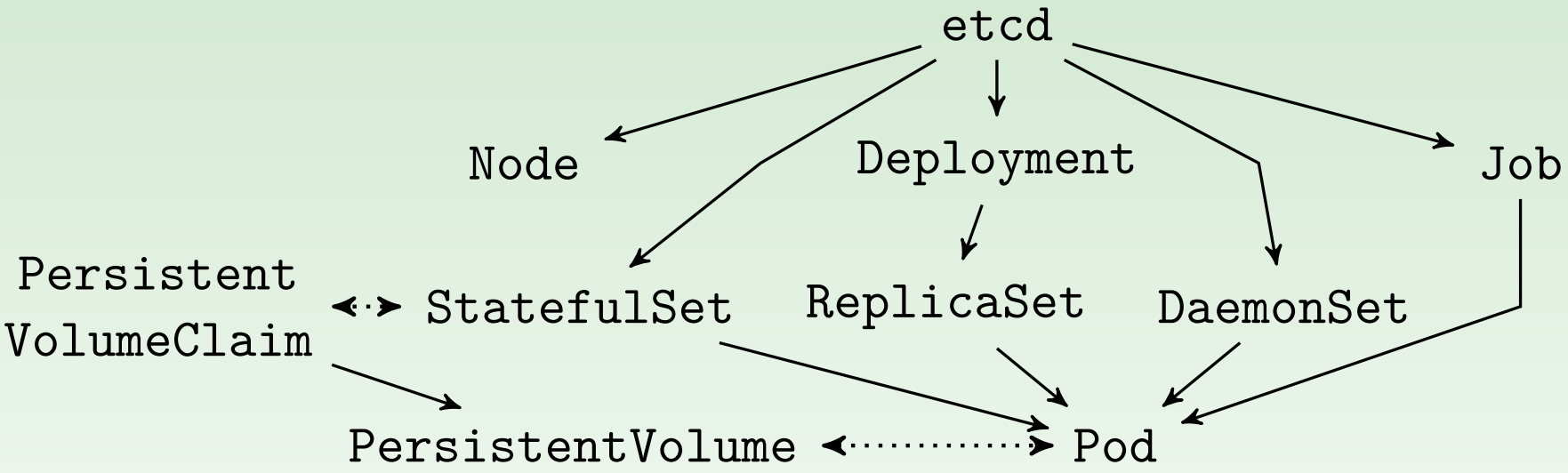
ReplicaSet jest generowana przez Deployment, w celu nadzorowania liczby uruchomionych podów.

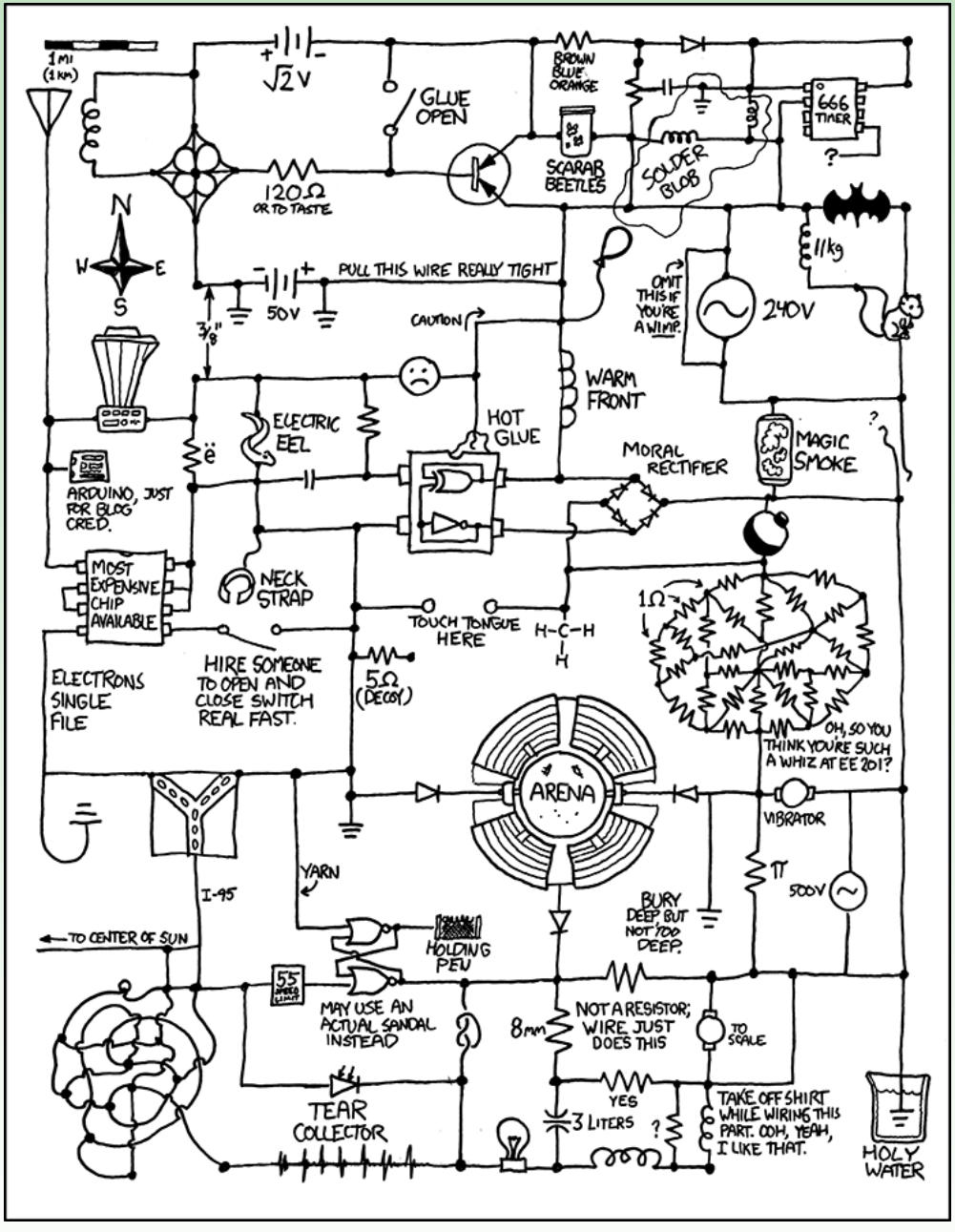
StatefulSet kontroluje liczbę uruchomionów podów, ale zapewnia także ich unikalność i uporządkowanie w celu zagwarantowania spójności stanu.

DaemonSet zawiera serwisy, które muszą występować w dokładnie jednej kopii na każdym węźle (np. sterownik sieciowy Kubernetesa, demon zbierający logi).

Job to odpowiednik Deploymentu, ale uruchamiający aplikację, która po wykonaniu zadania kończy działanie.

Podstawowe obiekty Kubernetesa





Źródło: https://xkcd.com/730/

Demonstracja 2

Tworzenie podów i deployment

Management technique	Operates on	Recommended environment	Supported writers	Learning curve
Imperative commands	Live objects	Development projects	1+	Lowest
Imperative object configuration	Individual files	Production projects	1	Moderate
Declarative object configuration	Directories of files	Production projects	1+	Highest

Źródło: `https://kubernetes.io/docs/concepts/overview/working-with-objects/object-management/`

Struktura obiektu

spec opisuje, w jakim stanie dany zasób ma się znaleźć. Może być kontrolowany przez użytkownika bądź kontrolery;

status zawiera aktualny stan zasobu;

apiVersion w celu zachowania kompatybilności (aktualnie `v1`);

name unikalna nazwa obiektu;

kind typ[1] obiektu jaki tworzymy;

ownerReference to wskaźnik na obiekt-twórcę (np. z `ReplicaSet` na `Deployment`).

[1] W tym momencie jestem zjadany przez osoby zajmujące się językami programowania.

Konrolery

- Osobne procesy z punktu logicznego. Jedna binarka i proces SO dla prostoty.
- Są pętlami sterowania.
- Obserwują co najmniej jeden obiekt.
- Dążą do osiągnięcia przez klaster stanu zadeklarowanego.

Pętla sterowania

W robotyce i automatyzacji, nieskończona pętla sterująca stanem systemu.

Kontrolery

Przykładowe kontrolery:

Węzłów odpowiedzialny za wykrywanie problemów z węzłami i reagowanie na nie (np. węzeł się wyłączył).

Jobów nadzoruje zadania jednorazowe i tworzy Pody do ich wykonania.

EndpointSlice wypełnia EndpointSlice (wpisuje mapowanie Podów na Service).

ServiceAccount Tworzy nowe domyślne ServiceAccounts dla nowych przestrzeni nazw.

Prezentowanie podów jako jedność

Service

Abstrakcja udostępniająca zbiór podów jako jeden byt.

EndpointSlice

Lista sprawnych podów należących do wybranego serwisu i wystawiających ten sam zbiór portów.

Typy serwisów:

- `ClusterIP` – widoczność tylko wewnątrz klastra.
- `NodePort` – statyczne mapowanie do portu noda.
- `LoadBalancer` – dostęp ze świata zewnętrznego poprzez serwer równoważący ruch.
- `ExternalName` – mapowanie poprzez zewnętrzny wpis DNS.
- *Headless* – podtyp `ClusterIP`, który nie ma przypisanego IP.

Etykiet (ang. labels)

Label

Etykieta pozwalająca zidentyfikować/wybrać obiekt

Annotation

Informacja dodatkowa nieidentyfikująca obiektu (komentarze, dane specyficzne dla implementacji).

Zastosowania:

- Wybór obiektów do kontrolowania (np. pody składające się na serwis)
- Zmiany w runtime poprzez zmiany etykiet (np. produkcja vs kwarantanna)
- Node/pod affinity
- Taint

Demonstracja 3

Serwisy

Pody

Pod (ang. stado, strąk)

Grupa jednego bądź więcej kontenerów współdzielących przestrzeń dyskową, zasoby sieciowe i konfigurację.

- Ulotne, łatwo zastępowalne
- Tworzone pośrednio (Deployment, Job)
- Ograniczona modyfikowalność (zmiana szablonu powoduje wyłączenie starych i włączenie nowych)
- Punkty monitorowania:
  - Żywotności
  - Gotowości
  - Uruchomienia

Pody

Typy kontenerów:

- Zwykły/Aplikacji
- Init
- Sidecar
- Ulotny (ang. ephemeral)

Idempotentność

Kontenery (w tym init i sidecar) mogą być restartowane. Muszą więc być przygotowane na sytuację, w której wykonały już swoją akcję i zmieniły stan poda (np. `emptyDir` nie jest puste).

Wolumeny

- Ulotne
  - emptyDir
  - ConfigMap
  - Secret
- Trwałe
  - NFS
  - hostPath
  - local
- Projected – mapują istniejące wolumeny do wspólnego katalogu.

PersistentVolume

Nieulotna przestrzeń dyskowa tworzona przez administratora lub dynamicznie poprzez StorageClass z zasobów chmurowych. Jest to zasób w klastrze – podobnie jak węzeł.

PersistentVolumeClaim

Alokacja pozwalająca używać PersistentVolume.

Obsługa stanu

Zastosowania StatefulSet:

- Stabilne i unikalne identyfikatory sieciowe.
- Pamięć nieulotna.
- Uporządkowane wdrażanie i skalowanie.
- Uporządkowane aktualizacje.

Działanie StatefulSet:

- Każdy pod w StatefulSet ma unikalny indeks z zakresu od 0 do $N-1$.
- Indeks jest zapisany w etykiecie `apps.kubernetes.io/pod-index`
- Wdrażanie i skalowanie:
  - Pody są tworzone sekwencyjnie zgodnie z numerami indeksów
  - Pody są usuwane zawsze od największego indeksu
  - Nie może powstać dziura w indeksach (a jeśli powstanie, to operacja jest zatrzymywana do czasu jej załatania)

Obsługa stanu

Ograniczenia:

- Redukcja liczby podów używających stanu, nie usuwa wolumenów ze stanem.
- StatefulSet wymaga Headless Service w celu zapewnienia dostępów do podów (zarządzanie domeną sieciową).
- StatefulSets nie usuwa/wyłącza podów gdy sam zostanie usunięty. Najpierw należy przestawić liczbę replik na 0.
- Istnieje ryzyko zakleszczenia aktualizacji, wymagające ręcznej interwencji.

Job

Definicja

Obiekt Job umożliwia uruchomienia jednokrotnego zadania w klastrze, po którego wykonaniu kontener automatycznie kończy działalność.

`spec.completions` liczba instancji Joba, którą trzeba wykonać

- NonIndexed różne instancje są nieodróżnialne
- Indexed instancje indeksowane, każdy indeks musi się choć raz zakończyć sukcesem

`spec.parallelism` liczba instancji, które mogą się wykonywać równolegle

Nieoczywiste scenariusze w Kubernetes

- Nieoczekiwana liczba kontenerów:
  - więcej niż zadeklarowano
  - mniej niż zadeklarowano
- Pod zduplikowany dla tego samego indeksu Joba:
  - przy awariach węzła, restarcie kubeleta, wywłaszczaniu;
  - tylko pierwszy zakończony pod liczy się do końcowego stanu Joba;
  - pozostałe pody są usuwane przez kontroler.
- Zduplikowany pod może powstać nawet gdy: `.spec.parallelism=1`; `.spec.completions=1` i `.spec.template.spec.restartPolicy = "Never"`
- Cały pod może się zepsuć np.
  - wycofanie węzła z użycia (upgrade, reboot, delete)
  - awaria kontenera przy restartPolicy=Never

  Aplikacja musi być gotowa na działanie po zrestartowaniu w nowym podzie z tym samym indeksem.

W skrócie

Wysypać się może wszystko, zawsze i wszędzie, a aplikacja musi być w stanie to obsłużyć.

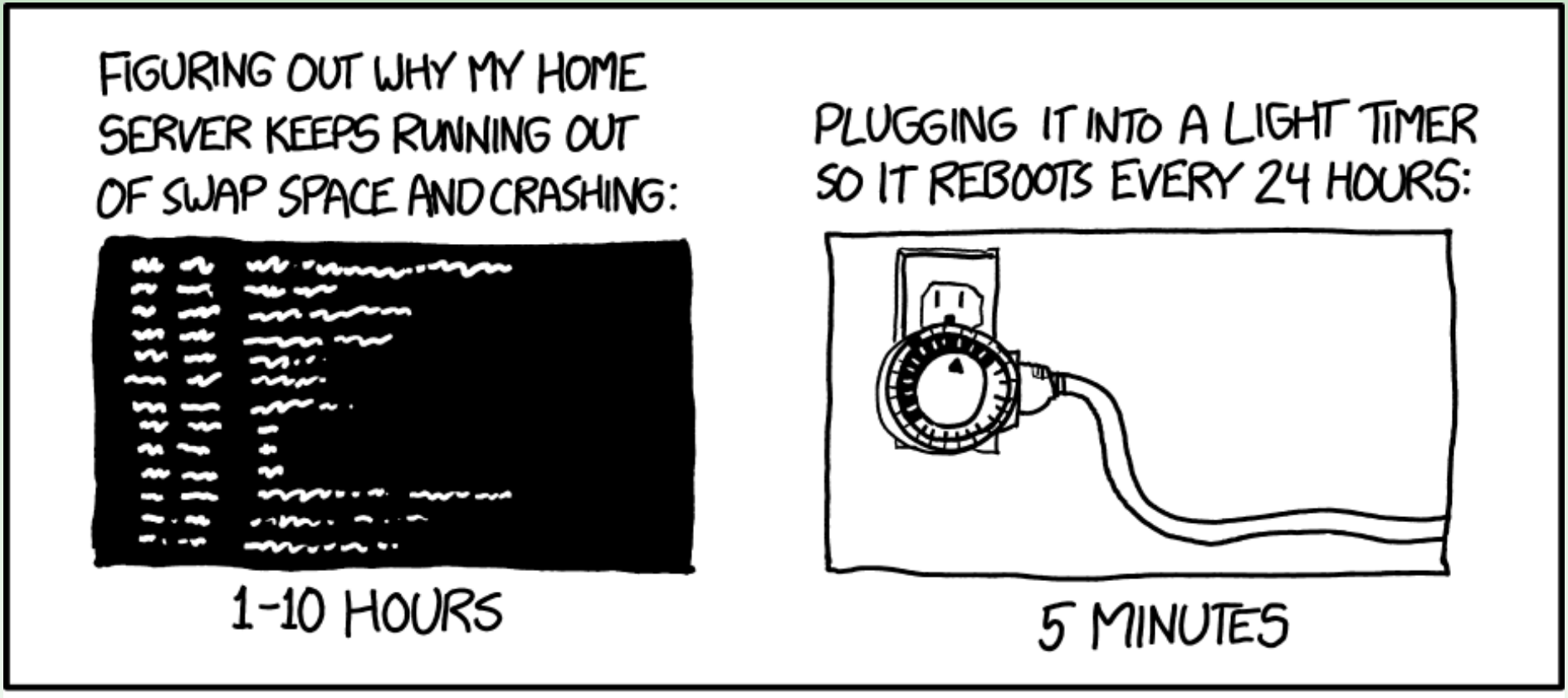
Demonstracja 4

Wolumeny i zadania

Akcje naprawcze

Akcje naprawcze udostępniane przez Kubernetes:

- Restart kontenera
- Zastąpienie poda
- Odłączenie poda od serwera równoważącego ruch
- Przepięcie pamięci nieulotnej



Źródło: `https://xkcd.com/1495/`

Scheduler

- Odpowiada za sparowanie tworzonych podów z węzłami, na których mają być uruchomione.
- Może zdecydować o wywłaszczeniu poda niższego priorytetu.
- Schedulowanie odbywa się w dwóch etapach:
  - filtrowanie,
  - ocena.

Rezerwacja zasobów

Limit Maksymalne możliwe użycie zasobu. Powyżej jego pod może zostać zabity.

Request Użycie oczekiwane brane pod uwagę podczas schedulowania poda.

Rezerwacja zasobów

Zasady rezerwacji zasobów:

- Zasoby rezerwowane przez fazę init, to maksimum z zasobów rezerwowanych przez każdy kontener init.
- Zasoby rezerwowane przez pod, to maksimum z zasobów rezerwowanych przez fazę init i sumy zasobów rezerwowanych przez aplikacje.
- Quality of Service określane jest na poziomie całego poda.
- Aplikacja może zarezerwować więcej zasobów niż używa, ale nie zostaną one przydzielone innym.

Skalowanie

Dwa typy skalowania:

horyzontalne to zwiększenie liczby podów/nodów

wertykalne to zwiększenie zasobów dostępnych do wykorzystania przez dany pod

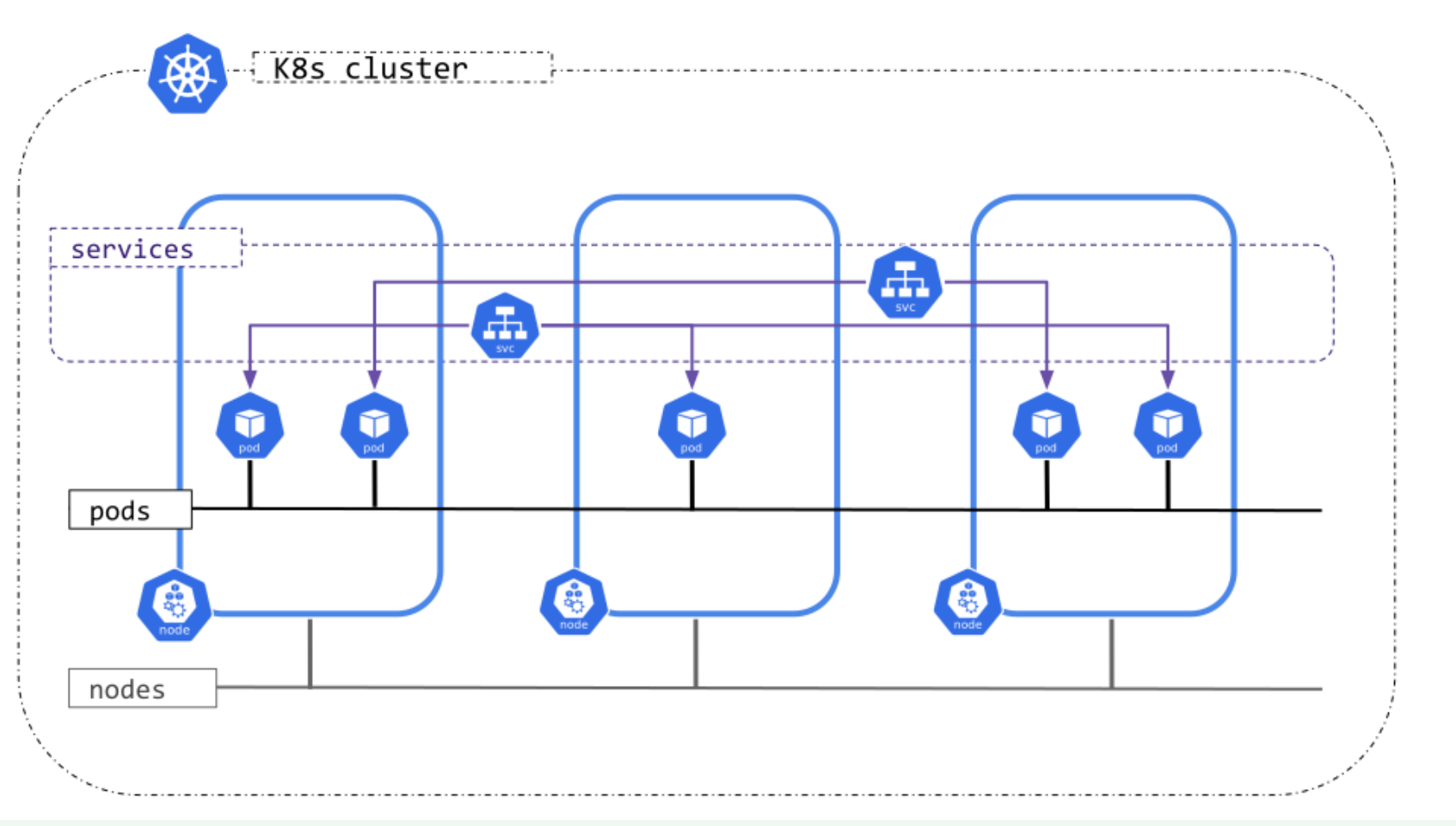
Demonstracja 5

Skalowanie zasobów

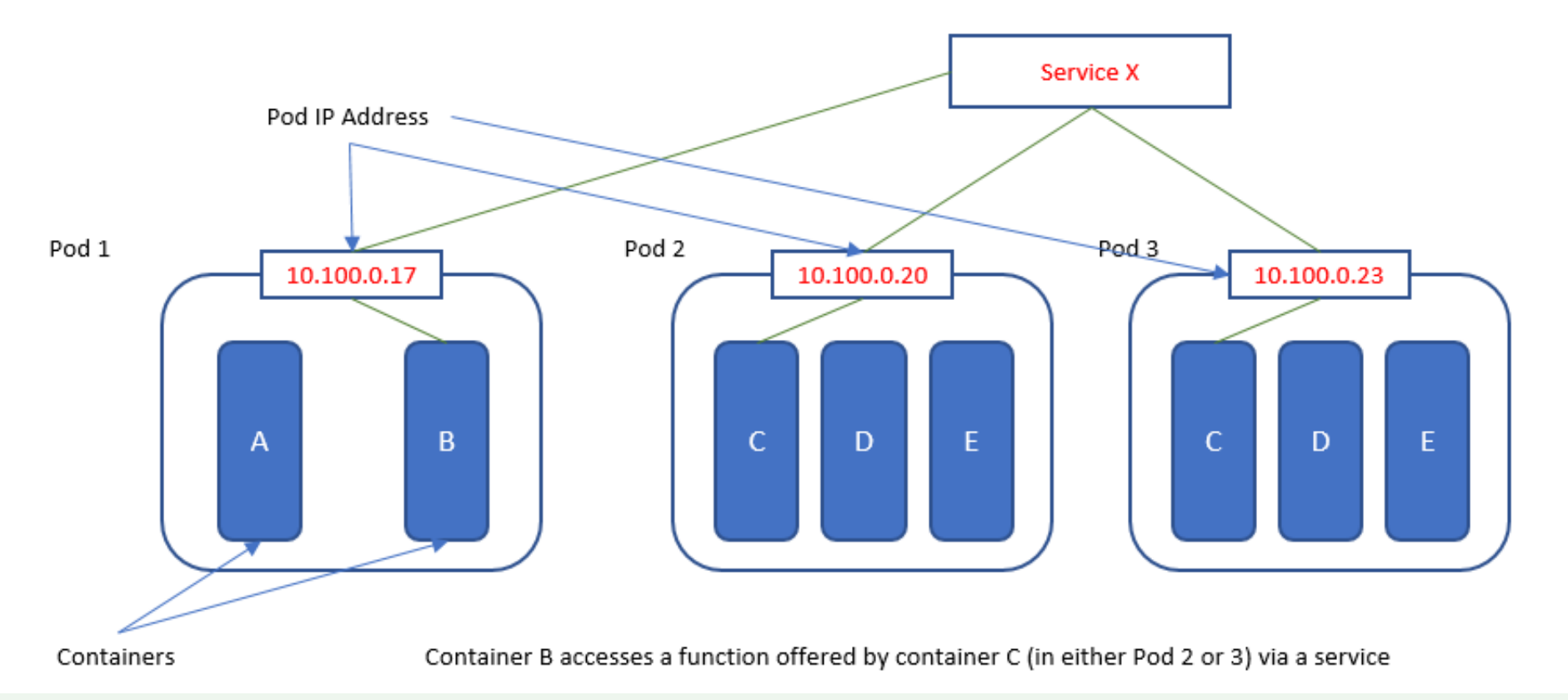
Model sieciowy

Elementy modelu sieciowego Kubernetesa:

- Każdy pod ma unikalny, wewnętrzny, adres IP.
- Pody znajdują się w innych sieciowych przestrzeniach nazw Linuksa.
- Kontenery w ramach jednego poda mogą komunikować się poprzez localhost.
- Procesy z węzła mogą komunikować się ze wszystkimi podami na tym podzie.
- Service umożliwia wystawienie stabilnego adresu IP.
- Domyślnie pody nie mogą komunikować się między sobą – wymagany jest network plugin.
- Gateway i Ingress umożliwiają wystawienie Service na świat zewnętrzny.



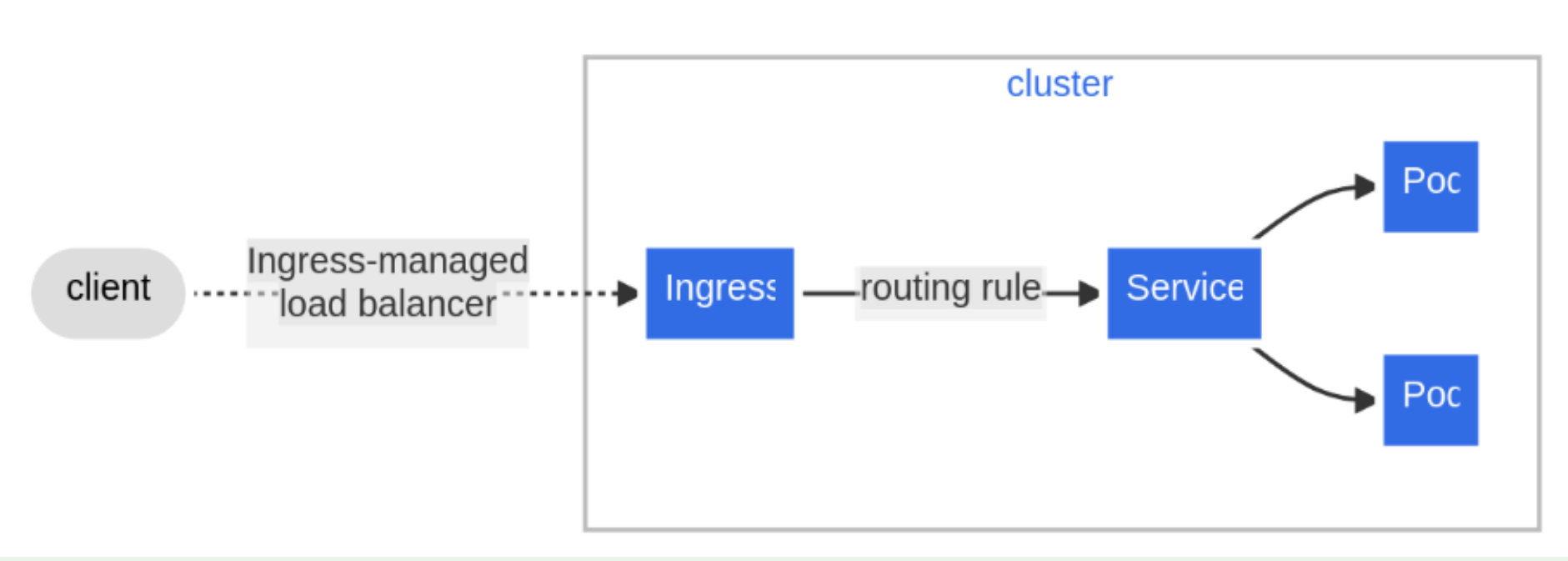
Źródło: https://kubernetes.io/docs/concepts/cluster-administration/networking/



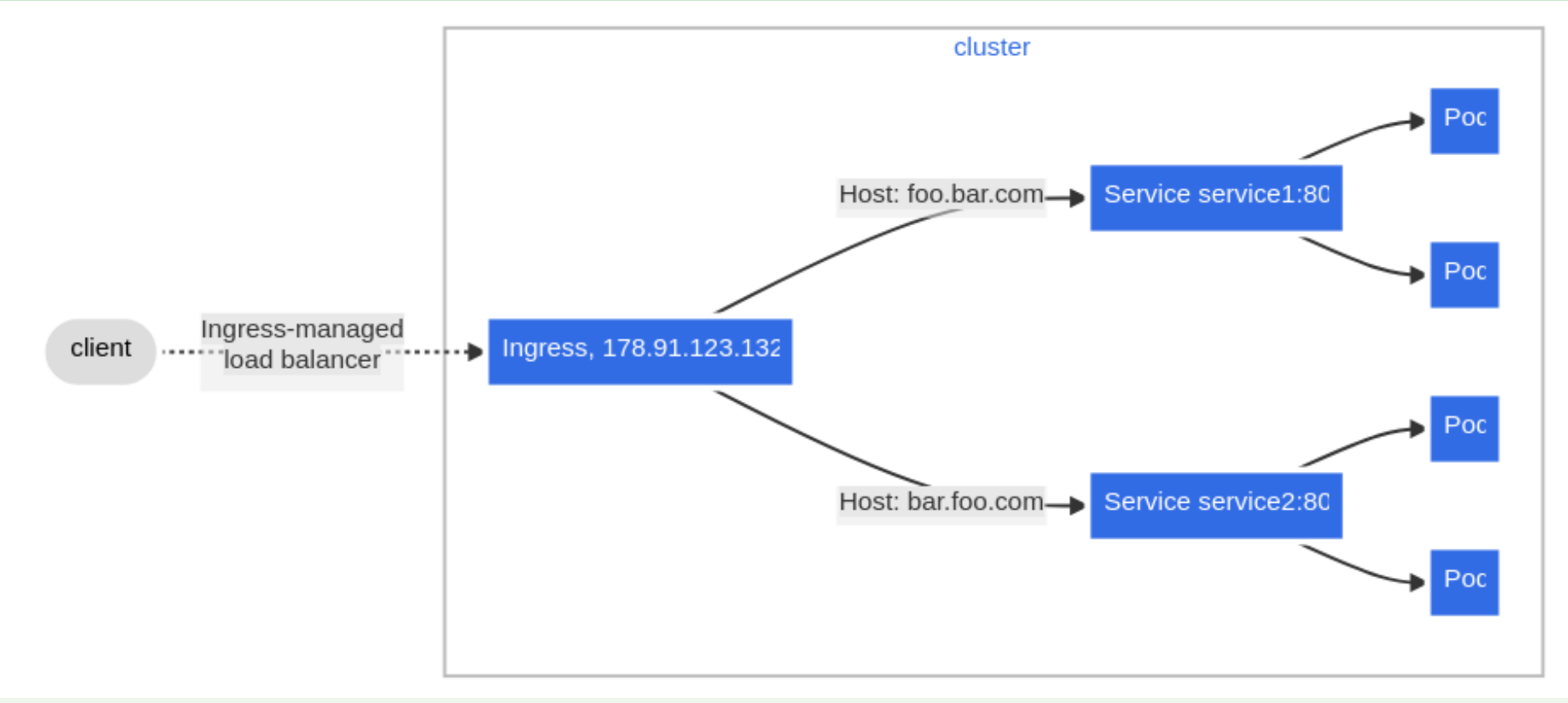
Źródło: `https://pl.wikipedia.org/wiki/Plik:Pod-networking.png`

Ingress

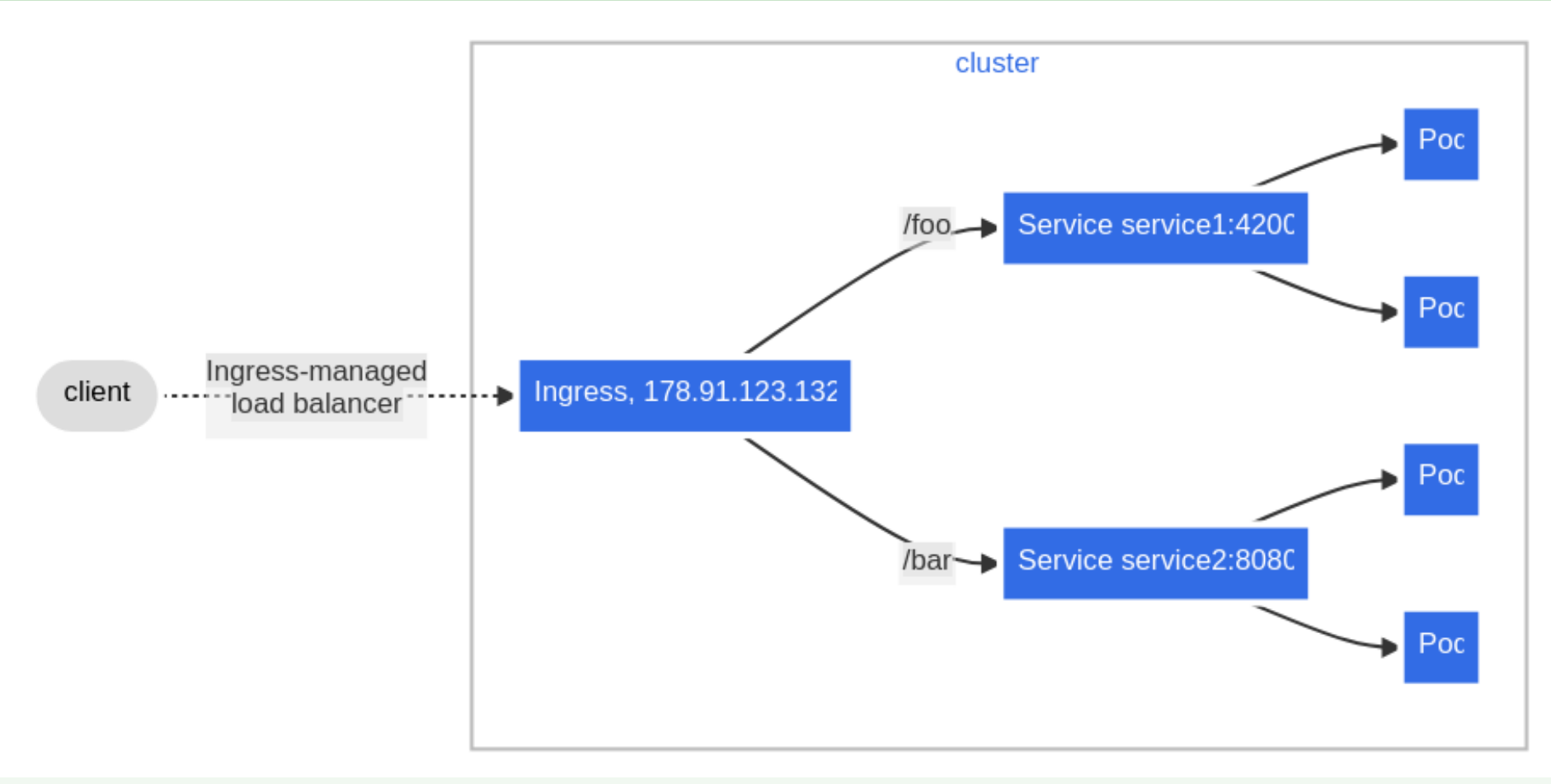
- Pozwala udostępnić Service do świata zewnętrznego.
- Obsługuje protokół HTTP/S.
- Może zapewniać równoważenie obciążenia i routowania na podstawie adresu.
- Wymaga stworzenia kontrolera Ingress (`IngressClass`)
- Pozwala tworzyć reguły HTTP/S na podstawie:
  - hosta (np. foo.bar.com),
  - ścieżki zasobu (np. /testpath),
- Backend to para Service oraz port.



Źródło: `https://kubernetes.io/docs/concepts/services-networking/ingress/`



Źródło: `https://kubernetes.io/docs/concepts/services-networking/ingress/`



Źródło: `https://kubernetes.io/docs/concepts/services-networking/ingress/`

Ingress Controller

Przykładowe kontrolery:

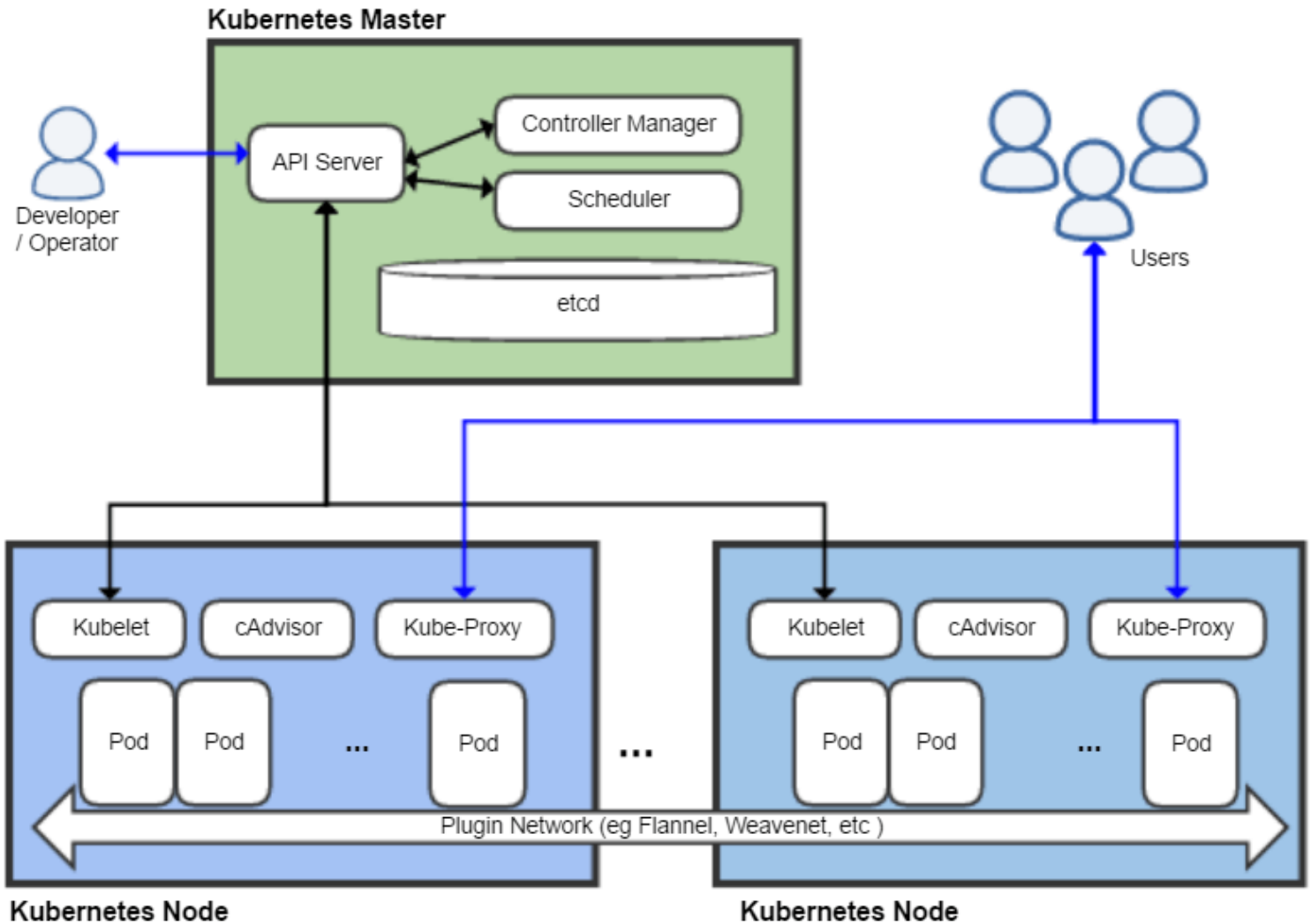
- HAProxy Ingress `https://haproxy-ingress.github.io/`
- NGINX `https://www.nginx.com/products/nginx-ingress-controller/`

Pełna lista: `https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/`

Demonstracja 6

Ingress

Podsumowując



Źródło: `https://en.wikipedia.org/wiki/File:Kubernetes.png`

Pułapki

YAML

YAML 1.2 jest niekompatybilny wstecz z YAML 1.1; gdyż rozpoznaje "yes"i "no"jako stringi, a wcześniej były to boole. Kubernetes używa *"mostly"* YAML 1.1.

Sekrety

Sekrety nie są tak sekretne jak mogło by się wydawać, bowiem może je przeczytać każdy:

- z dostępem do API;
- z dostępem do etcd;
- kto może tworzyć pody (w tym poprzez Deployment).

Dodatkowo Sekrety są przechowywane w sposób niezaszyfrowany.

Tematy do dalszego dokształcania się

- Helm – menadżer pakietów dla Kubernetesa
- Ephemeral containers
- Namespace
- Integracja z chmurą
- Bezpieczeństwo
- Pod Topology Spread Constraints
- Node autoscaler
- ServiceAccount
- Node affinity
- Lease
- Finalizers

„Informatycy to ludzie, którzy tworzą problemy, by później bohatersko je rozwiązać.”