Kubernetes

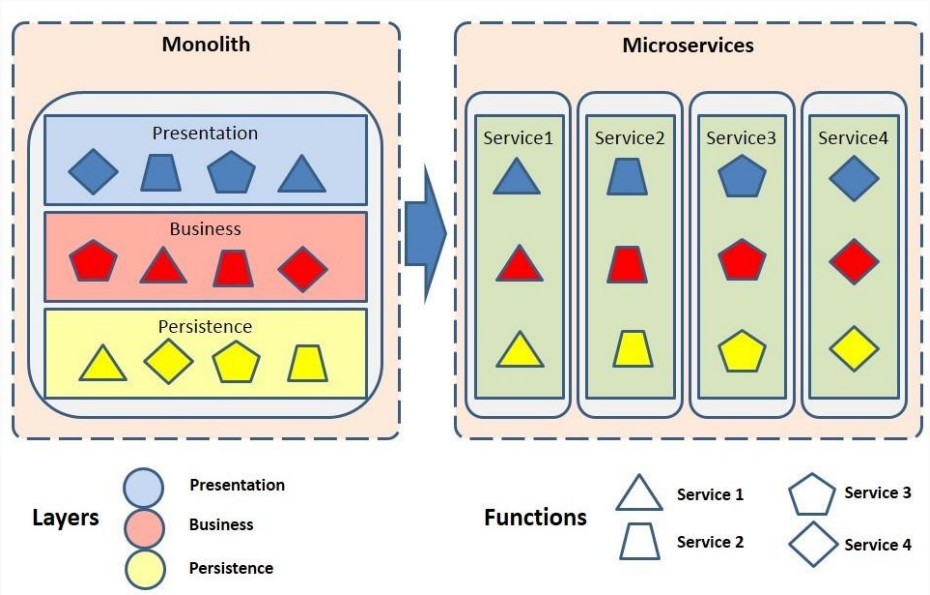
Agenda

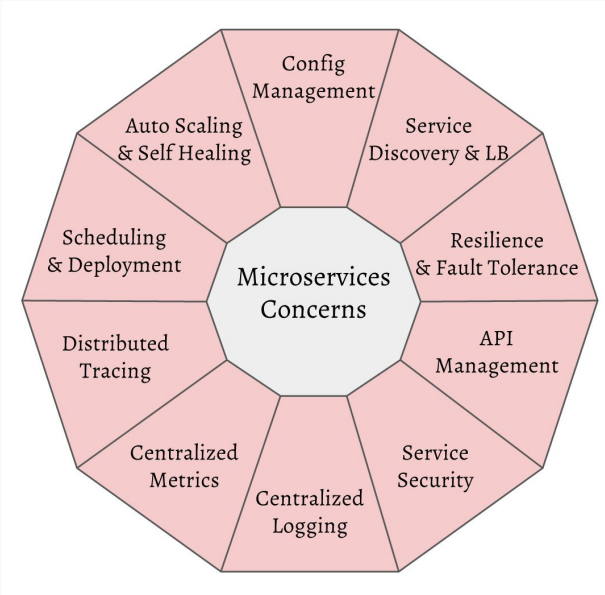
- Motywacja
- Wirtualizacja
- Kubernetes - charakterystyka
- Kubernetes - klaster
- Kubernetes API
- Migracja

Wildfly - problemy

- Zarządzanie zasobami - część maszyn ma zużycie 100%, inne się nudzą. Jak maksymalnie wykorzystać to co mamy?
- Uzależnienie od jednej technologii - Spring jeszcze śmignie na Wildfly, ale co jeśli zechcemy zrobić API composer w NodeJS? Jak zautomatyzować deployment?
- Brak wsparcia dla CI/CD - blue/green deployment, canary deployment
- Podatność na awarie - konieczność ręcznego restartowania Wildfly w przypadku krytycznej awarii
- Ścisłe powiązanie serwisów w ramach jednej grupy serwerów - Wildfly leży, to wszystkie usługi na nim leżą
- Skalowanie statyczne - poprzez definiowanie grupy serwerów, mikrozarządzanie

Architektura mikroserwisowa





Wirtualizacja - po co?

- Chcemy na jednym serwerze uruchomić aplikacje działające na różnych systemach operacyjnych
- Aplikacje powinny działać niezależnie od siebie
- Separacja zasobów

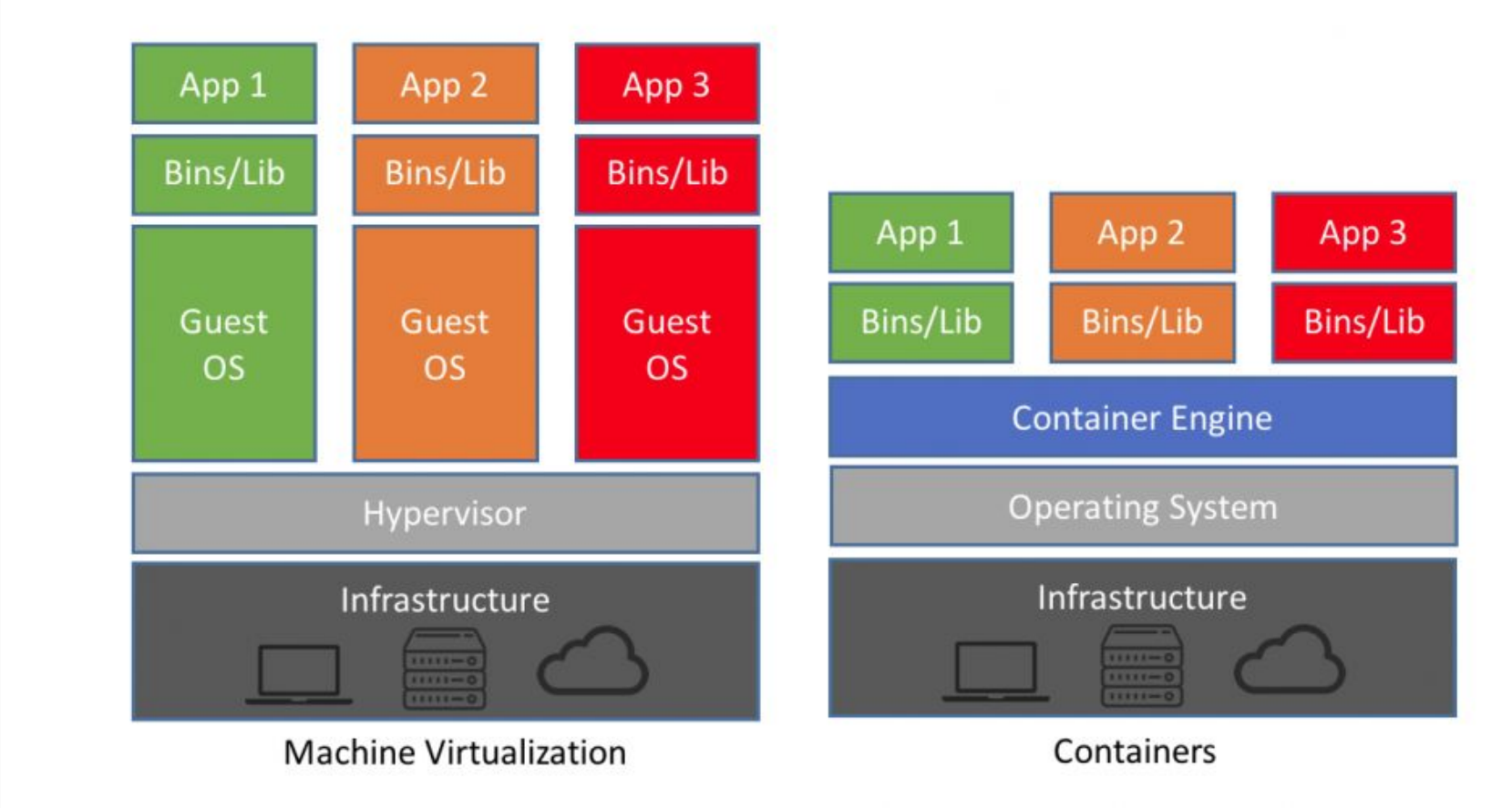
Wirtualizacja

- tworzenie symulowanego środowiska wygenerowanego komputerowo
- kilka maszyn wirtualnych może działać na jednym serwerze fizycznym
- maszyny wirtualne są niezależne od siebie, mogą mieć różne systemy itp.
- struktura warstwowa: host OS, hypervisor, guest OS, biblioteki/pliki binarne, aplikacja
- zasoby systemowe są przydzielane osobno dla każdej maszyny wirtualnej

Konteneryzacja

- wirtualizacja na poziomie systemu operacyjnego
- brak dodatkowych warstw: hypervisor, guest OS
- jednostka uruchomienia: kontener
- współdzielenie jądra systemu między kontenerami
- dla każdego kontenera tworzony jest osobny system plików
- zużywa mniej zasobów niż wirtualizacja
- do uruchamiania służy container engine, np. docker, containerd, cri-o

Konteneryzacja vs Wirtualizacja



Kubernetes

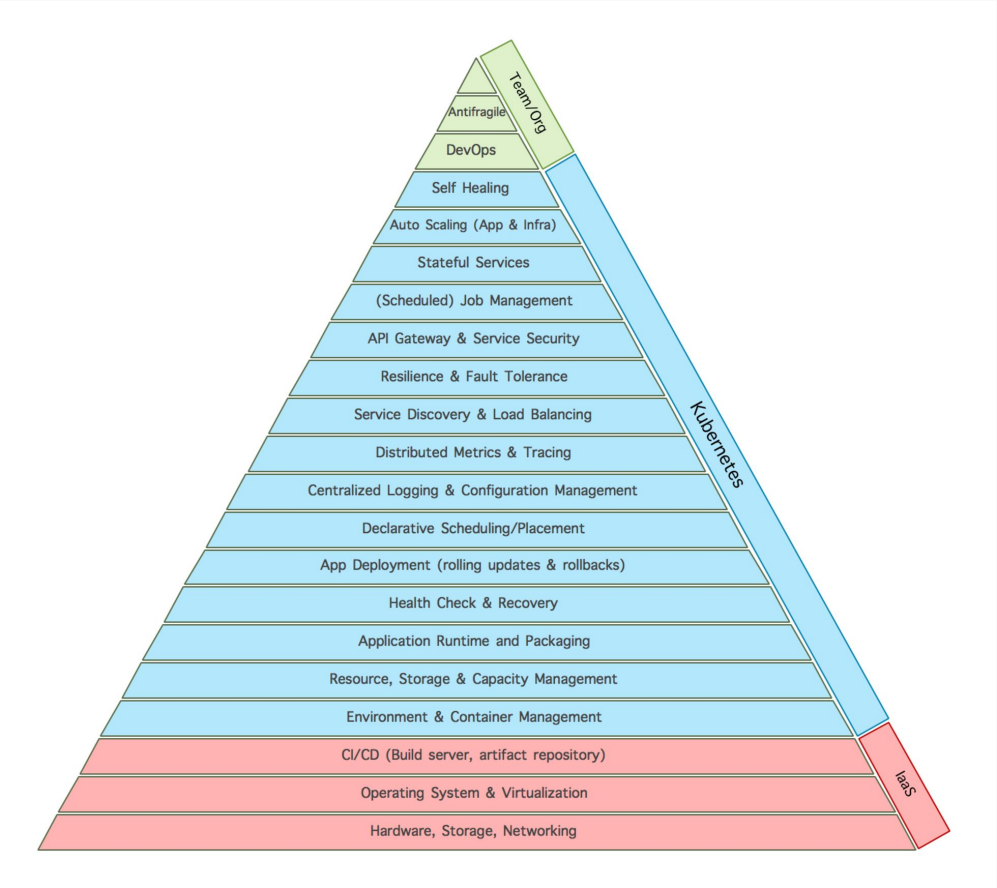
- system orkiestracji skonteneryzowanych usług
- platforma open source
- wspiera deklaratywną konfigurację i automatyzację
- pozwala uruchamiać aplikacje niezależnie od technologii
- różne możliwości wdrożenia: cloud, on premise, bare metal

Kubernetes - złote myśli Michała Juszczaka™

Nie myślimy o tym, że mamy 10 maszyn. Mamy jednego “kubernetesa”, który sam decyduje gdzie co zdeployować, ile razy replikować i jak komunikować usługi między sobą.

Nie interesuje nas infrastruktura maszyn, w dockerze piszemy, która wersja javy jest potrzebna i jak ma być skonfigurowane środowisko. Developerzy przejmują obowiązki zarządzania środowiskiem uruchomieniowym, ale w zamian za to dużo łatwiej np. zmigrować wersję java tylko w części usług.

Kubernetes - overview



Kubernetes - klaster

- węzły control plane - zawierają systemowe serwisy K8s
- węzły worker - zawierają uruchomione aplikacje
- w celu uzyskania wysokiej dostępności uruchamiane są dodatkowe węzły control plane (liczba nieparzysta >= 3)
- domyślnie usługi nie są związane z pojedynczym węzłem i mogą być dynamicznie skalowane
- na każdym węźle uruchamiany jest agent (kubelet) np. poprzez systemd
- kubectl - klient łączący się z masterem

Kubernetes - komponenty

- Komponenty control plane
- Komponenty węzła
- CNI - Container Network Interface
- control plane Load Balancer (HA)

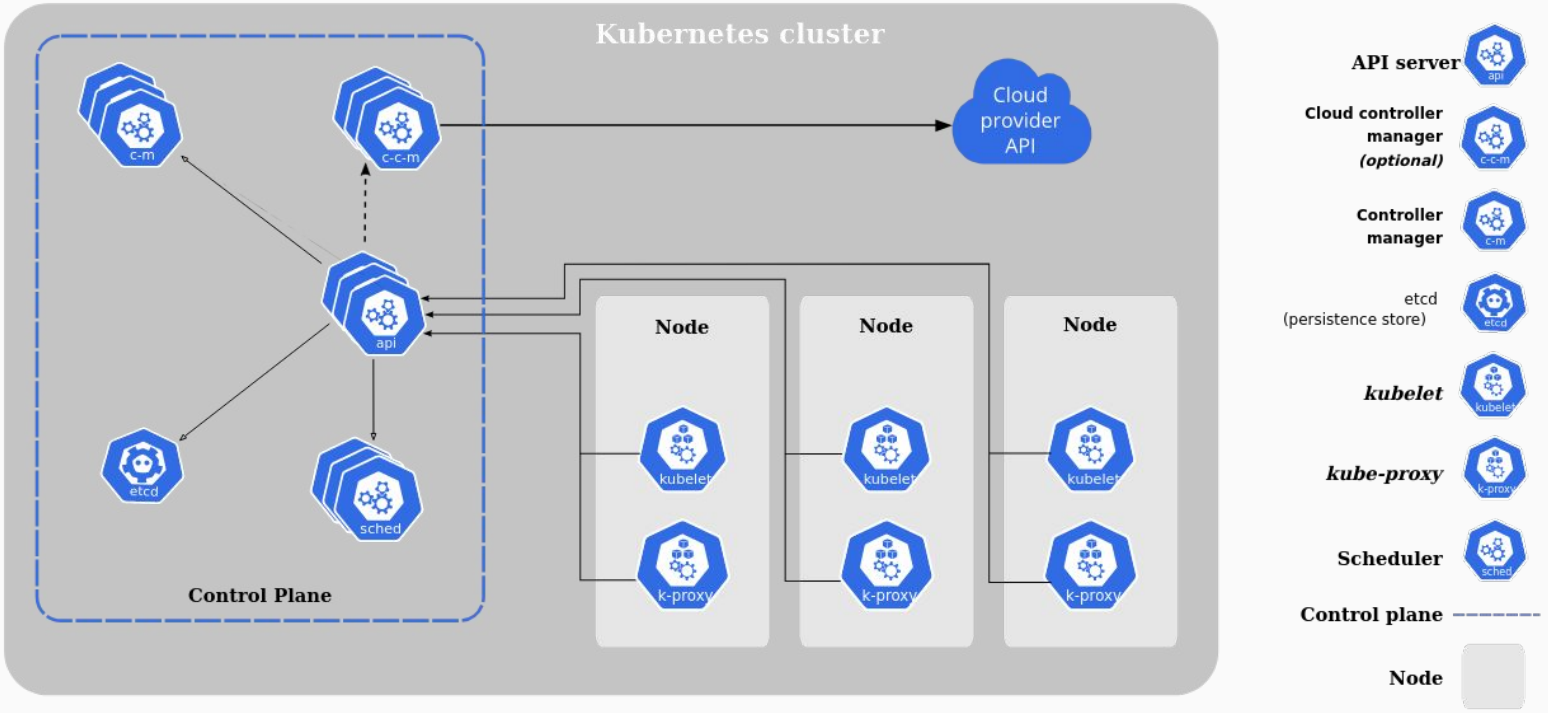
Kubernetes - komponenty control plane

- API server
- etcd
- Scheduler
- Controller Manager
- Cloud Controller Manager - opcjonalnie

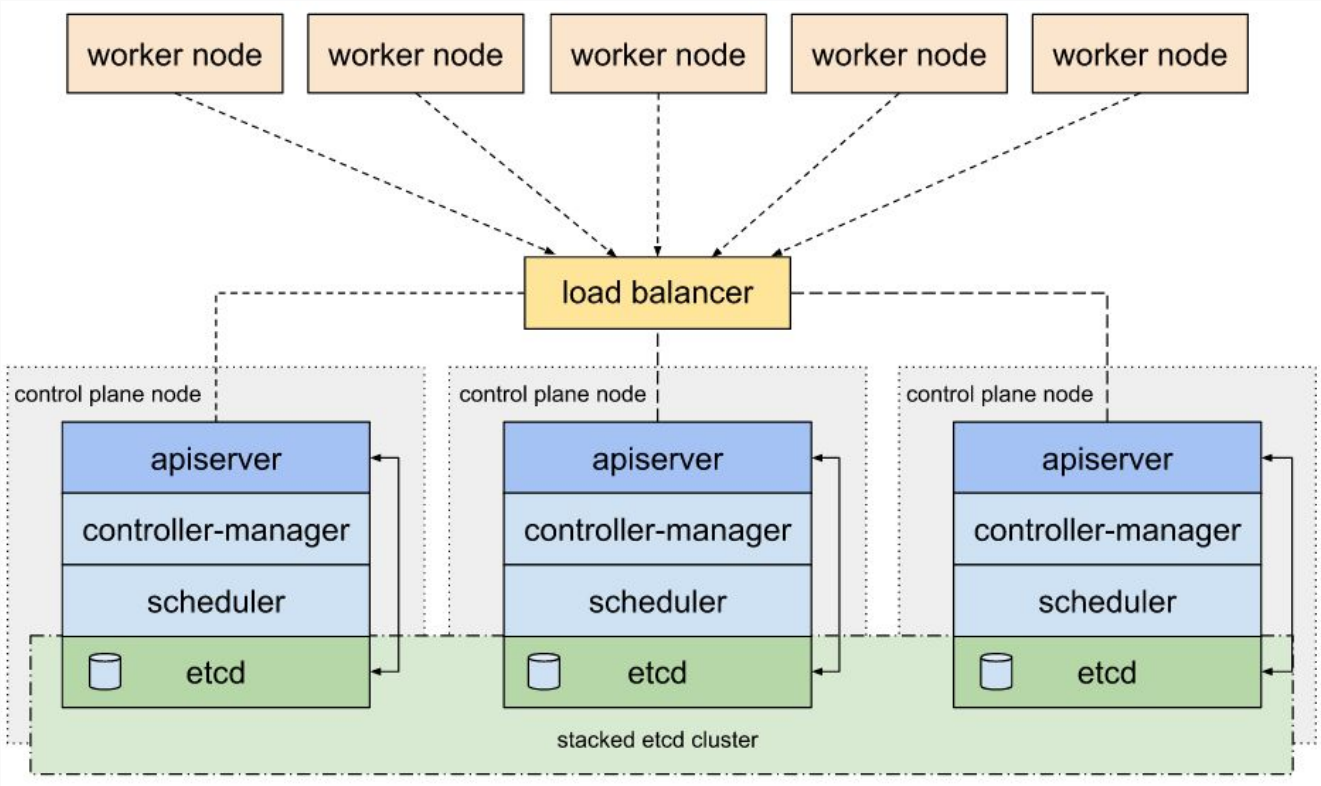
Kubernetes - komponenty węzła

- kubelet
- kube-proxy
- CRI - Container Runtime Interface

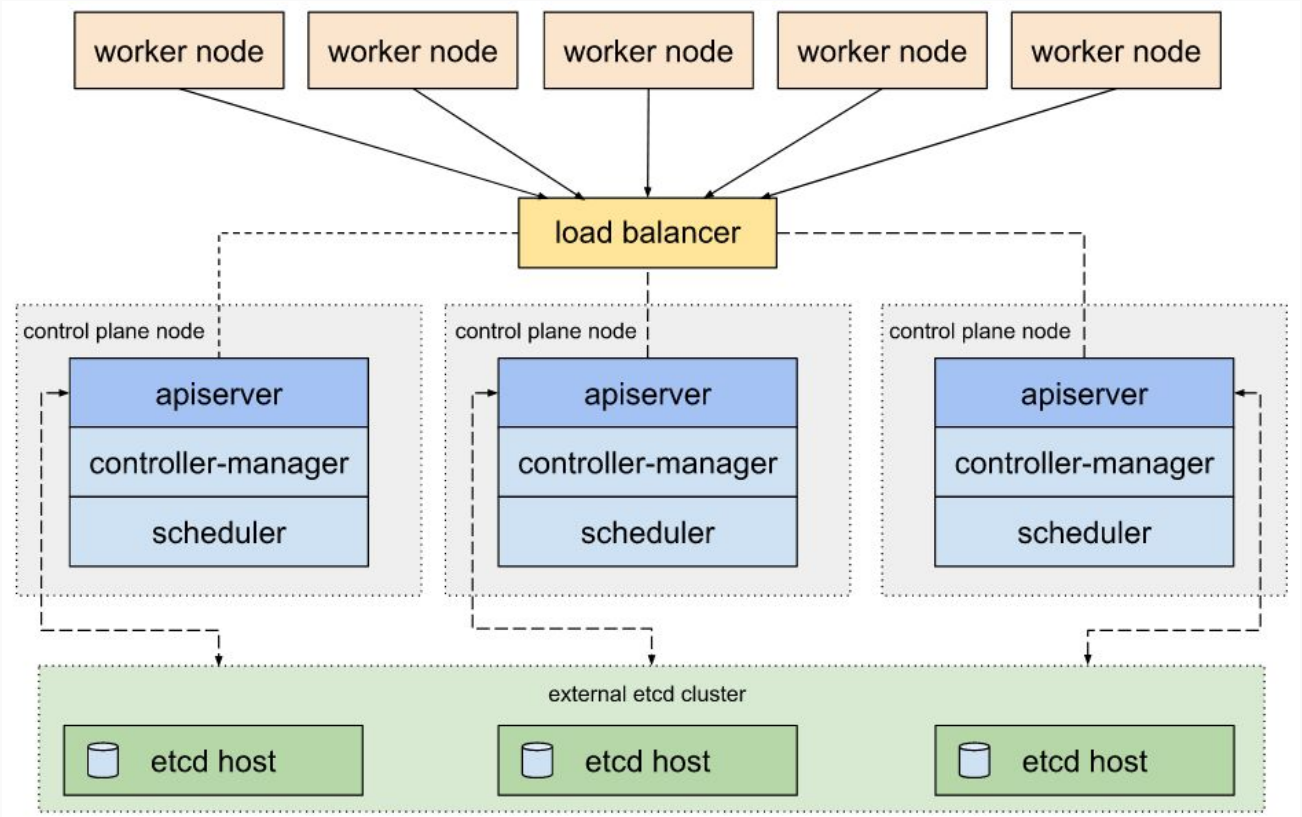
Kubernetes - klaster



Kubernetes HA - stacked etcd



Kubernetes HA - external etcd



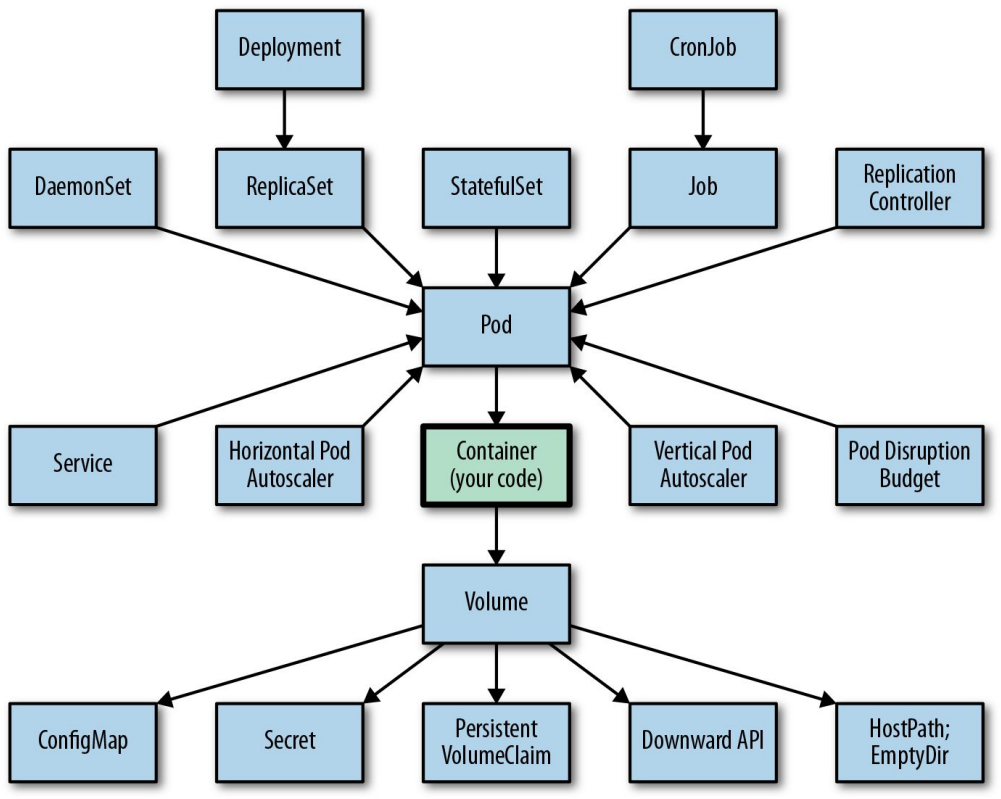
Kubernetes API

- struktura komponentów: zbiór obiektów
- stan obiektu to używane obrazy, ilość replik, zasoby fizyczne/sieciowe itp.
- obiekty są generowane poprzez deklaratywne określenie ich oczekiwanego stanu
- każdy obiekt ma swoją specyfikację zdefiniowaną w API - manifesty w formacie yml

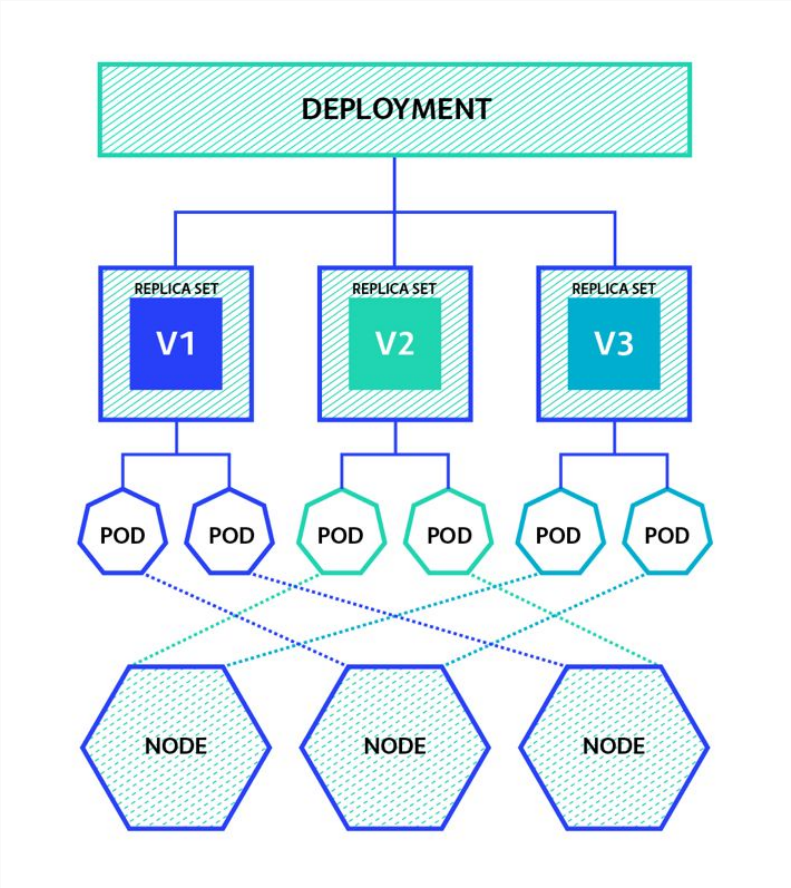
Kubernetes API - obiekty

- jednostka wdrożeniowa: Pod (enkapsulacja kontenera)
- kontrolery: Deployment, ReplicaSet, StatefulSet, DaemonSet, Job
- zarządzanie systemami plików: PersistentVolume, PersistentVolumeClaim
- selektor etykiet: Service (wykrywanie usług, definicja sposobu dostępu)
- obiekty konfiguracyjne: ConfigMap, Secret
- autoryzacja: ClusterRole, ClusterRoleBinding, ServiceAccount

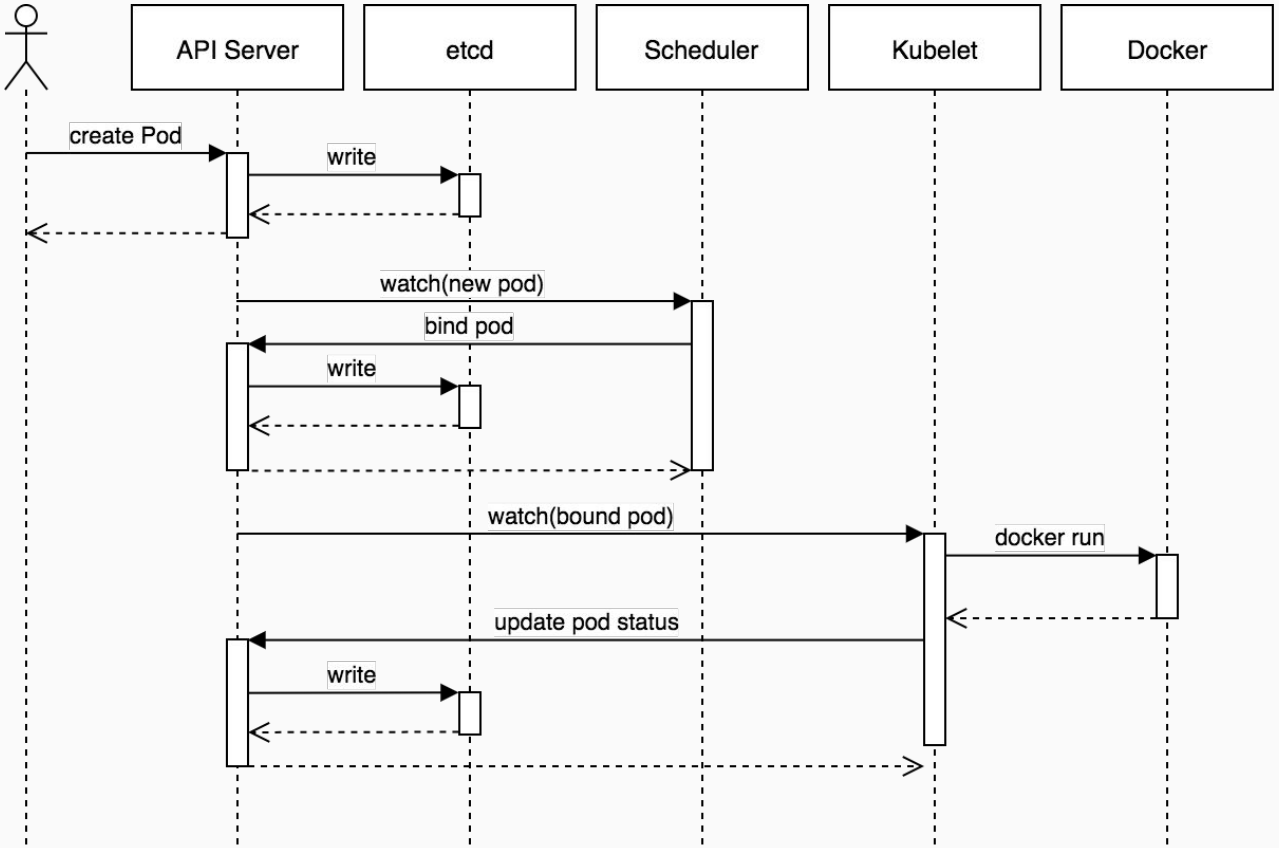
Kubernetes API - zależności



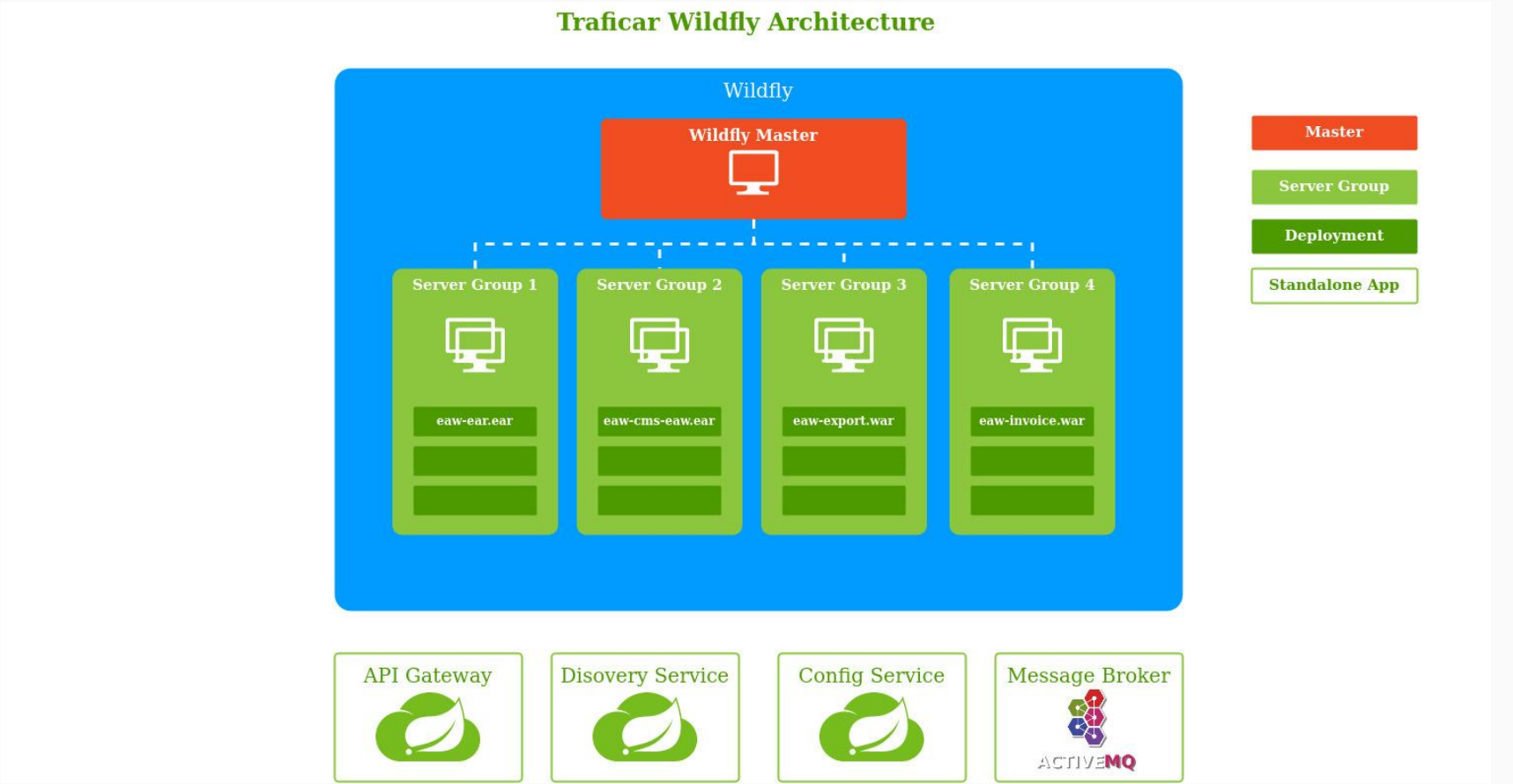
Kubernetes API - zależności pojedynczego wdrożenia



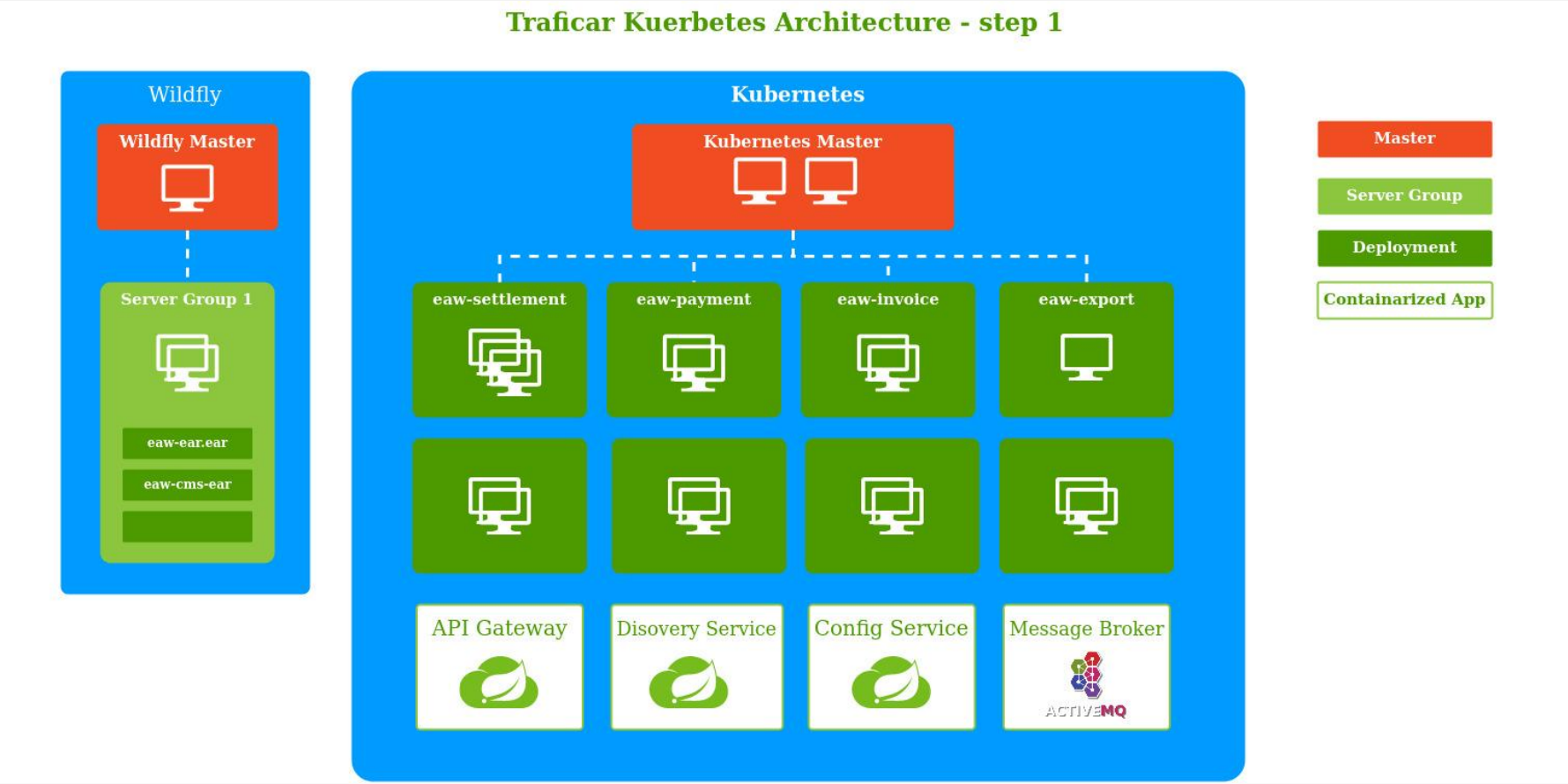
Przykład: uruchamianie Poda



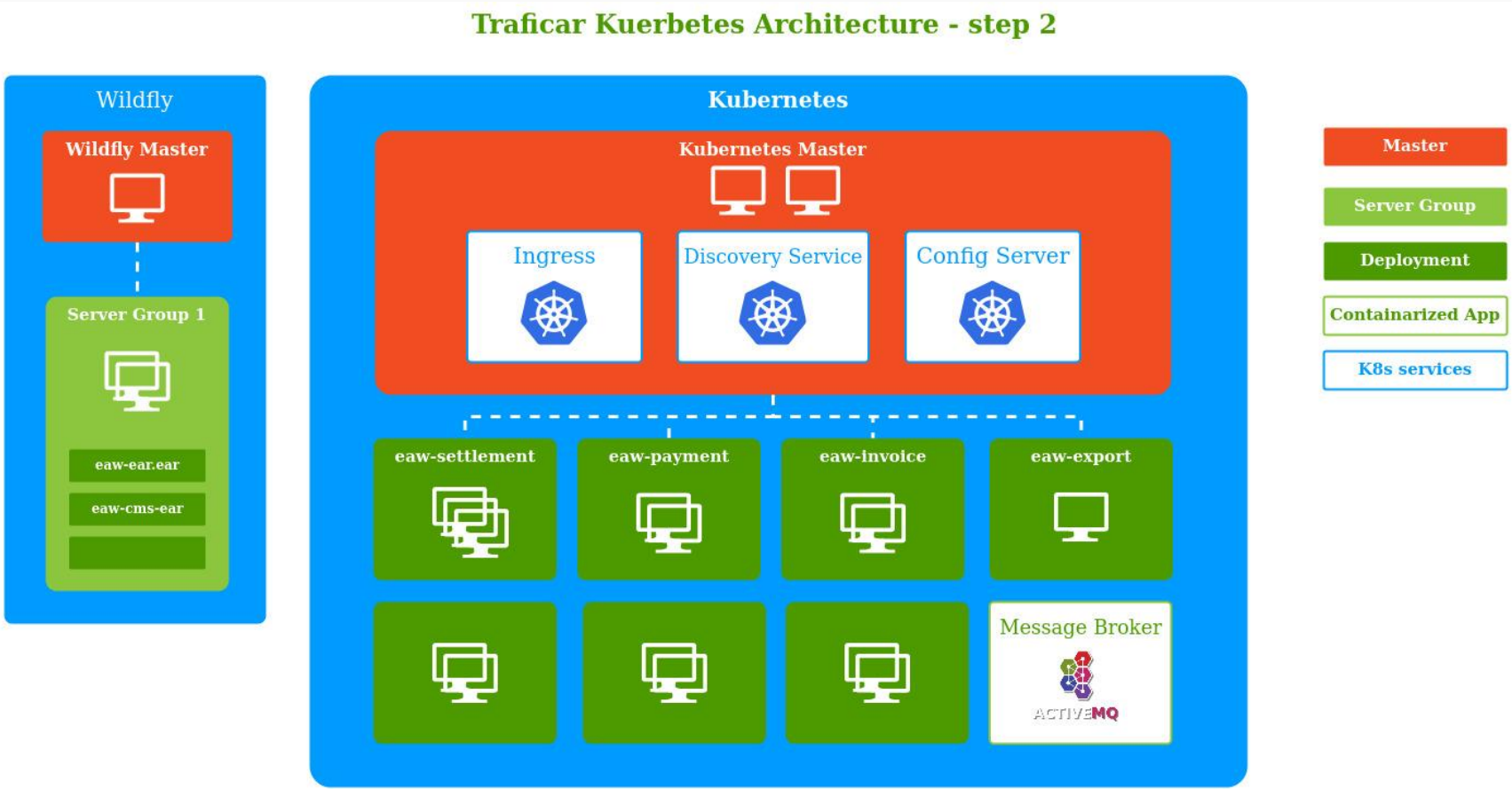
Migracja


Traficar Wildfly Architecture
Wildfly
Wildfly Master
Server Group 1
Server Group 2
Server Group 3
Server Group 4
eaw-ear.ear
eaw-cms-eaw.ear
eaw-export.war
eaw-invoice.war
Master
Server Group
Deployment
Standalone App
API Gateway
Disovery Service
Config Service
Message Broker
ACTIVEMQ

Migracja


Traficar Kuerbetes Architecture - step 1
Wildfly
Wildfly Master
Server Group 1
eaw-ear.ear
eaw-cms-ear
Kubernetes
Kubernetes Master
eaw-settlement
eaw-payment
eaw-invoice
eaw-export
API Gateway
Disovery Service
Config Service
Message Broker
ACTIVEMQ
Master
Server Group
Deployment
Containarized App

Migracja


Traficar Kuerbetes Architecture - step 2
Wildfly
Wildfly Master
Server Group 1
eaw-ear.ear
eaw-cms-ear
Kubernetes
Kubernetes Master
Ingress
Discovery Service
Config Server
eaw-settlement
eaw-payment
eaw-invoice
eaw-export
Message Broker
ACTIVEMQ
Master
Server Group
Deployment
Containarized App
K8s services

Migracja - możliwe problemy

- przygotowanie środowiska - instalacja Kubernetesa na każdym węźle, konfiguracja ruchu sieciowego, konfiguracja klastra itp.
- wykorzystanie zasobów - do sprawdzenia
- aplikacje stanowe - synchronizacja trwałych zasobów
- unrecognized risk - mogą pojawić się nowe problemy, które trudno teraz przewidzieć