Nomad, Podman z Quadlet i zarządzany Kubernetes: AKS i EKS

Poziom: średnio

Dotyczy: HashiCorp Nomad 2.0.7 (licencja BSL), Podman 5.8 z Rocky Linux 9.8 (wydanie projektu 6.1.4), Quadlet, Pacemaker 2.1.10 z pcs 0.11.11, Azure Kubernetes Service, Amazon EKS, Azure CLI, eksctl, Kubernetes 1.34 do 1.37 u dostawców chmury

Czas: ~75 min

Kubernetes i Swarm nie są jedynymi drogami. Ta strona opisuje trzy inne: Nomada od HashiCorp, który planuje kontenery i zwykłe programy jednym plikiem binarnym; Podmana z Quadlet, w którym kontener jest usługą systemd, a dwie maszyny pilnuje Pacemaker; oraz Kubernetesa w chmurze, gdzie płaszczyznę sterowania utrzymuje Microsoft albo Amazon. Wracamy tu, gdy pada pytanie „czy na pewno potrzebujemy Kubernetesa”.

ZAŁOŻENIA SCENARIUSZA

laboratorium Nomada to trzy maszyny `nomad01` (10.20.10.110), `nomad02` (.111) i `nomad03` (.113) z Rocky Linux 9 i Dockerem, na dwóch hostach `hv01` i `hv02`, każda jednocześnie serwerem i klientem. Podman z Quadlet stoi na parze `pcmk01` i `pcmk02` z klastra Pacemakera opisanego w książce o wysokiej dostępności, z adresem wirtualnym 10.20.10.60 i trzecim głosem na `bkp01`. Klastry w chmurze są przykładowe: Azure w regionie `polandcentral`, AWS w `eu-central-1`, osobne subskrypcje i konta do prób, bez połączenia z siecią Zielonej.

Skróty użyte na tej stronie (17)

Skrót	Co znaczy
AKS	Azure Kubernetes Service: Kubernetes w chmurze Microsoftu, w którym płaszczyznę sterowania utrzymuje dostawca
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
AWS	Amazon Web Services: chmura Amazonu

Skrót	Co znaczy
BSL	Business Source License: licencja, na ktora HashiCorp zmienil w 2023 roku Terraform, ograniczajaca prawo do oferowania narzedzia jako konkurencyjnej platnej uslugi
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
CNI	Container Network Interface: wtyczka, która daje sieć podom w Kubernetesie
EKS	Elastic Kubernetes Service: Kubernetes w chmurze Amazon Web Services, w którym płaszczyznę sterowania utrzymuje dostawca
etcd	etc distributed: rozproszony magazyn par klucz-wartość, w którym Kubernetes trzyma cały stan klastra, a zmiany uzgadnia protokołem Raft
IBM	International Business Machines: firma, w której laboratorium powstał model relacyjny i pierwszy działający silnik SQL
QEMU	Quick Emulator: program, który uruchamia maszyny wirtualne na Linuksie, z KVM jako akceleracją sprzętową; Proxmox VE zarządza maszynami przez niego
Raft	protokół uzgadniania stanu w klastrze: zmiana jest przyjęta, gdy zgodzi się większość węzłów; na nim stoją menedżerowie Swarm i etcd
RPC	Remote Procedure Call: sposób, w jaki programy wywołują funkcje na innym komputerze w sieci; kontrolery domeny używają go między innymi do replikacji
SLA	Service Level Agreement: umowa o poziomie usług, np. 99,9% dostępności
SQLite	mała baza danych zapisana w jednym pliku; Home Assistant trzyma w niej historię w pliku home-assistant_v2.db
VPC	Virtual Private Cloud: prywatna sieć wirtualna klienta w chmurze AWS, w której stawiamy maszyny i zakładamy tablice tras
WMS	Warehouse Management System: system zarządzania magazynem
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

Fiszka

Komendy `nomad` wykonujemy na dowolnym węźle laboratorium Nomada. Komendy Podmana i `systemctl --user` jako zwykły użytkownik, a komendy z `sudo` i `pcs` na `pcmk01`. Komendy `az` po `az login` (Azure: podstawy operacyjne), komendy `eksctl` i `aws` z profilem AWS CLI (AWS CLI: instalacja, profile, poświadczenia). Nazwy `NazwaZadania`, `NazwaGrupyZasobow`, `NazwaKlastra`, `NazwaRegionu`, `NumerWersji` i `nazwa_usera` zastępujemy własnymi.

□□ ZMIENIA SYSTEM

Instalacja Nomada z repozytorium HashiCorp: yum-config-manager

```
# Komendy można wykonać całościowo jako blok tekstu, na każdym z trzech węzłów.
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo
https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo
sudo yum -y install nomad
```

Trzy komendy z dokumentacji HashiCorp [1]. Repozytorium podaje najnowszą wersję (2.0.7 w dniu odczytu), więc wersję zapisujemy w notatce wdrożenia i trzymamy tę samą na wszystkich serwerach.

yum = Yellowdog Updater Modified: starszy menedżer pakietów RHEL · **yum-config-manager** = narzędzie z pakietu yum-utils do dodawania i włączania repozytoriów pakietów

TYLKO ODCZYT

Lista serwerów Nomada i lidera: nomad server members

```
nomad server members
```

Pokazuje serwery regionu, ich stan i to, który jest liderem Raft.

Przykład:

```
Name              Address       Port  Status  Leader  Raft Version  Build  Datacenter
Region
nomad01.global    10.20.10.110  4648  alive   true    3             2.0.7  pamiatkowo
global
nomad02.global    10.20.10.111  4648  alive   false   3             2.0.7  pamiatkowo
global
nomad03.global    10.20.10.113  4648  alive   false   3             2.0.7  pamiatkowo
global
```

Raft = protokół uzgadniania stanu w klastrze: zmiana jest przyjęta, gdy zgodzi się większość węzłów; na nim stoją menedżerowie Swarm i etcd · **nomad** = program HashiCorp Nomad: ten sam plik działa jako serwer, klient i narzędzie do wysyłania zadań

TYLKO ODCZYT

Lista klientów Nomada: nomad node status

```
nomad node status
```

Pokazuje maszyny, które uruchamiają zadania, i to, czy są gotowe (`ready`).

nomad = program HashiCorp Nomad: ten sam plik działa jako serwer, klient i narzędzie do wysyłania zadań

TYLKO ODCZYT

Podgląd zmian przed wdrożeniem: nomad job plan

```
nomad job plan /etc/nomad.d/zadania/NazwaZadania.nomad.hcl
```

Pokazuje, co Nomad zmieni (ile alokacji powstanie, ile zniknie), bez wdrażania. Na końcu wypisuje komendę `nomad job run -check-index`, która wdroży dokładnie to, co zobaczyliśmy.

nomad = program HashiCorp Nomad: ten sam plik działa jako serwer, klient i narzędzie do wysyłania zadań

TYLKO ODCZYT

Wdrożenie zadania: nomad job run

```
nomad job run /etc/nomad.d/zadania/NazwaZadania.nomad.hcl
```

Wysyła opis zadania do serwerów i śledzi wdrożenie [2]. Kod wyjścia 2 oznacza problem z rozmieszczeniem (brak zasobów, niespełnione ograniczenia), a 1 każdy inny błąd.

nomad = program HashiCorp Nomad: ten sam plik działa jako serwer, klient i narzędzie do wysyłania zadań

TYLKO ODCZYT

Stan zadania i jego alokacji: nomad job status

```
nomad job status NazwaZadania
```

Pokazuje podsumowanie grup zadań, ostatnie wdrożenie i listę alokacji z węzłami, na których stoją [3].

nomad = program HashiCorp Nomad: ten sam plik działa jako serwer, klient i narzędzie do wysyłania zadań

TYLKO ODCZYT

Logi zadania bez szukania alokacji: nomad alloc logs -job

```
nomad alloc logs -job NazwaZadania nginx
```

> (gdzie `nginx` to nazwa zadania z bloku `task` w pliku; opcja `-job` wybiera losową, najchętniej działającą alokację tego zadania [4])

nomad = program HashiCorp Nomad: ten sam plik działa jako serwer, klient i narzędzie do wysyłania zadań

TYLKO ODCZYT

Powrót do poprzedniej wersji zadania: nomad job revert

```
nomad job revert NazwaZadania NumerWersji
```

Wdraża ponownie wskazaną wersję z historii zadania. Numery wersji pokazuje `nomad job history NazwaZadania`.

nomad = program HashiCorp Nomad: ten sam plik działa jako serwer, klient i narzędzie do wysyłania zadań

ZMIENIA SYSTEM

Plik zadania Nomada z kolorowym nginksem: kolory.nomad.hcl

```
# /etc/nomad.d/zadania/kolory.nomad.hcl
job "kolory" {
  datacenters = ["pamiatkowo"]
  type        = "service"

  group "www" {
    count = 3

    network {
      port "http" {
        to = 80
```

```
      }
    }

    service {
      name     = "kolory"
      port     = "http"
      provider = "nomad"
    }

    update {
      max_parallel = 1
      auto_revert  = true
    }

    task "nginx" {
      driver = "docker"

      config {
        image = "nginx:1.30.5"
        ports = ["http"]
      }

      resources {
        cpu    = 100
        memory = 64
      }
    }
  }
}
```

Trzy kopie nginksa, każda na losowym porcie hosta przekierowanym na port 80 kontenera, z rejestracją w wbudowanym katalogu usług Nomada (`provider = "nomad"`, bez Consula). Aktualizacja idzie po jednej kopii i cofa się sama, gdy nowa wersja nie przejdzie kontroli.

□□ **type** = pokaż, czym jest dane słowo dla powłoki: poleceniem, aliasem czy funkcją ·**config** = FortiOS: otwiera blok ustawień jednej tabeli konfiguracji, na przykład config system ha, i kończy się słowem end

□□ **TYLKO ODCZYT**

Plik konfiguracyjny serwera i klienta Nomada: nomad.hcl

```
# /etc/nomad.d/nomad.hcl na nomad01, nomad02 i nomad03
datacenter = "pamiatkowo"
data_dir   = "/opt/nomad/data"
bind_addr  = "0.0.0.0"

server {
  enabled          = true
  bootstrap_expect = 3

  server_join {
    retry_join = ["10.20.10.110", "10.20.10.111", "10.20.10.113"]
  }
}

client {
  enabled = true
}
```

`bootstrap_expect = 3` każe serwerom czekać na trzech uczestników przed wyborem lidera [5]. Blok `server_join` zastępuje starsze opcje `retry_join` na poziomie serwera, które według not wydań znikną w Nomadzie 2.1 [6].

PLIK KONFIGURACYJNY

Plik kontenera Quadlet dla użytkownika: kolory.container

```
# ~/.config/containers/systemd/kolory.container
[Unit]
Description=Kolorowy nginx w kontenerze Podmana

[Container]
Image=docker.io/library/nginx:1.30.5
PublishPort=8080:80
AutoUpdate=registry

[Service]
Restart=on-failure
```

```
[Install]
WantedBy=default.target
```

Po przeładowaniu systemd z pliku `kolory.container` powstaje usługa `kolory.service`, a kontener dostaje domyślnie nazwę `systemd-kolory` [7]. `AutoUpdate=registry` wymaga pełnej nazwy obrazu z rejestrem, dlatego piszemy `docker.io/library/nginx`, a nie samo `nginx` [8].

□□ TYLKO ODCZYT

Wczytanie plików Quadlet użytkownika: systemctl --user daemon-reload

```
systemctl --user daemon-reload
```

Generator Quadlet czyta pliki z `~/.config/containers/systemd/` i tworzy z nich jednostki systemd. Bez tego kroku nowy albo zmieniony plik nie istnieje dla systemd.

□□ **systemctl** = system control: sterowanie usługami systemd

□□ TYLKO ODCZYT

Uruchomienie kontenera jako usługi: systemctl --user start

```
systemctl --user start kolory.service
```

Usługi z Quadlet nie włączamy przez `systemctl enable`, bo systemd traktuje je jako generowane; o starcie z systemem decyduje sekcja `[Install]` w pliku `.container` [7].

□□ **systemctl** = system control: sterowanie usługami systemd

□□ ZMIENIA SYSTEM

Praca usług użytkownika bez zalogowania: loginctl enable-linger

```
sudo loginctl enable-linger nazwa_usera
```

Bez tego usługi użytkownika gasną, gdy użytkownik się wyloguje, i nie startują po restarcie serwera.

□□ **loginctl** = login control: zarządzanie sesjami użytkowników systemd, w tym lingering, dzięki któremu usługi użytkownika działają bez zalogowania

□□ ZMIENIA SYSTEM

Sprawdzenie dostępnych aktualizacji obrazów: podman auto-update --dry-run

```
podman auto-update --dry-run
```

Sprawdza, czy w rejestrze są nowsze obrazy dla kontenerów z `AutoUpdate`, bez pobierania i restartu; kolumna `UPDATED` pokazuje `pending` [8]. Same aktualizacje wykonuje codziennie o północy jednostka `podman-auto-update.timer`, którą włączamy przez `systemctl --user enable --now podman-auto-update.timer` (dla kontenerów systemowych bez `--user`). Po nieudanym restarcie Podman domyślnie wraca do poprzedniego obrazu.

□□ **podman** = pod manager: silnik kontenerów zgodny z Dockerem, bez stale działającego demona

□□ TYLKO ODCZYT

Podgląd tego, co wygeneruje Quadlet: podman-system-generator --dryrun

```
/usr/lib/systemd/system-generators/podman-system-generator --user --dryrun
```

Wypisuje jednostki, które powstałyby z plików Quadlet, i błędy składni. Dla plików systemowych z `/etc/containers/systemd/` pomijamy `--user` i uruchamiamy przez sudo [7].

□□ PLIK KONFIGURACYJNY

Plik kontenera Quadlet uruchamianego przez Pacemaker: etykiety-www.container

```
# /etc/containers/systemd/etykiety-www.container na pcmk01 i pcmk02
# Bez sekcji [Install]: kontener uruchamia tylko Pacemaker, nie rozruch systemu.
[Unit]
Description=Serwer WWW etykiet w kontenerze (zasób Pacemakera)

[Container]
```

```
Image=docker.io/library/nginx:1.30.5
ContainerName=etykiety-www
PublishPort=10.20.10.60:8080:80
Volume=/srv/etykiety/pliki:/usr/share/nginx/html:ro,Z
```

Ten sam plik leży na obu węzłach, a po `sudo systemctl daemon-reload` na każdym z nich istnieje jednostka `etykiety-www.service`. Kontener słucha tylko na adresie wirtualnym i czyta dane z systemu plików, który Pacemaker montuje na aktywnym węźle.

WWW = World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

ZMIENIA SYSTEM

Dodanie kontenera Quadlet do grupy Pacemakera: pcs resource create systemd

```
sudo pcs resource create www_etykiety systemd:etykiety-www op monitor interval=15s timeout=30s --group grp_etykiety
```

Dokłada usługę z kontenerem na koniec grupy `grp_etykiety`, po wolumenie, systemie plików i adresie wirtualnym, więc startuje dopiero wtedy, gdy dane i adres są na miejscu. Czy klasa `systemd` Pacemakera obsługuje jednostki generowane przez Quadlet bez zastrzeżeń, sprawdzamy testem w laboratorium; drugą drogą jest agent `ocf:heartbeat:podman` z pakietu resource-agents [9].

pcs = powłoka administracyjna klastra Pacemaker i Corosync: tworzy klaster, zasoby, fencing i ograniczenia z jednego miejsca

ZMIENIA SYSTEM

Utworzenie grupy zasobów Azure: az group create

```
az group create --name NazwaGrupyZasobow --location polandcentral
```

az = Azure CLI: narzędzie chmury Microsoftu

ZMIENIA SYSTEM

Utworzenie klastra AKS w trzech strefach: az aks create --zones

```
az aks create --resource-group NazwaGrupyZasobow --name NazwaKlastra --tier standard --node-count 3 --zones 1 2 3 --generate-ssh-keys
```

Tworzy klaster z pulą systemową po jednym węźle w każdej z trzech stref dostępności i z poziomem Standard, który ma umowę SLA [10][11]. Przed uruchomieniem sprawdzamy, czy wybrany region ma strefy dostępności i czy rozmiar maszyn je obsługuje (`az aks list-vm-skus`).

AKS = Azure Kubernetes Service: Kubernetes w chmurze Microsoftu, w którym płaszczyznę sterowania utrzymuje dostawca · **SLA** = Service Level Agreement: umowa o poziomie usług, np. 99,9% dostępności · **az** = Azure CLI: narzędzie chmury Microsoftu

TYLKO ODCZYT

Pobranie dostępu do klastra AKS: az aks get-credentials

```
az aks get-credentials --resource-group NazwaGrupyZasobow --name NazwaKlastra
```

Dopisuje klaster do `~/.kube/config`, po czym działają zwykłe komendy `kubectl` z Fiszki strony o porównaniu orkiestratorów.

AKS = Azure Kubernetes Service: Kubernetes w chmurze Microsoftu, w którym płaszczyznę sterowania utrzymuje dostawca · **az** = Azure CLI: narzędzie chmury Microsoftu

TYLKO ODCZYT

Lista wersji Kubernetesa w regionie Azure: az aks get-versions

```
az aks get-versions --location polandcentral --output table
```

Pokazuje wersje, które AKS oferuje w regionie, i ścieżki aktualizacji.

AKS = Azure Kubernetes Service: Kubernetes w chmurze Microsoftu, w którym płaszczyznę sterowania utrzymuje dostawca · **az** = Azure CLI: narzędzie chmury Microsoftu

ZMIENIA SYSTEM

Utworzenie klastra EKS z węzłami zarządzanymi: eksctl create cluster

```
eksctl create cluster --name NazwaKlastra --region eu-central-1
```

Tworzy klaster EKS z siecią VPC i grupą węzłów zarządzanych i od razu dopisuje go do `~/.kube/config` [12].

EKS = Elastic Kubernetes Service: Kubernetes w chmurze Amazon Web Services, w którym płaszczyznę sterowania utrzymuje dostawca · **VPC** = Virtual Private Cloud: prywatna sieć wirtualna klienta w chmurze AWS, w której stawiamy maszyny i zakładamy tablice tras · **eksctl** = EKS control: narzędzie do tworzenia i usuwania klastrów Amazon EKS jedną komendą

ZMIENIA SYSTEM

Utworzenie klastra EKS w trybie Auto Mode: eksctl create cluster --enable-auto-mode

```
eksctl create cluster --name=NazwaKlastra --enable-auto-mode
```

W trybie Auto Mode AWS zarządza także węzłami, magazynem i siecią klastra. Wymaga eksctl w wersji 0.195.0 lub nowszej [13].

EKS = Elastic Kubernetes Service: Kubernetes w chmurze Amazon Web Services, w którym płaszczyznę sterowania utrzymuje dostawca · **AWS** = Amazon Web Services: chmura Amazonu · **eksctl** = EKS control: narzędzie do tworzenia i usuwania klastrów Amazon EKS jedną komendą

TYLKO ODCZYT

Lista wersji Kubernetesa w EKS: aws eks describe-cluster-versions

```
aws eks describe-cluster-versions
```

Pokazuje wersje z ich stanem (`STANDARD_SUPPORT`, `EXTENDED_SUPPORT`) i datami końca wsparcia [14].

EKS = Elastic Kubernetes Service: Kubernetes w chmurze Amazon Web Services, w którym płaszczyznę sterowania utrzymuje dostawca · **aws** = AWS CLI: narzędzie chmury Amazonu

TYLKO ODCZYT

Usunięcie klastra AKS razem z grupą zasobów: az group delete

```
az group delete --name NazwaGrupyZasobow --yes --no-wait
```

Usuwa grupę zasobów ze wszystkim, co w niej jest, bez pytania. Do prób zakładamy osobną grupę, w której nie ma nic poza klastrem.

AKS = Azure Kubernetes Service: Kubernetes w chmurze Microsoftu, w którym płaszczyznę sterowania utrzymuje dostawca · **az** = Azure CLI: narzędzie chmury Microsoftu

TYLKO ODCZYT

Usunięcie klastra EKS: eksctl delete cluster

```
eksctl delete cluster --name NazwaKlastra --region eu-central-1
```

Usuwa klaster i węzły utworzone przez eksctl. Po zakończonych próbach zawsze sprawdzamy w konsoli, czy nie zostały wolumeny i równoważniki obciążenia, bo za nie płacimy dalej.

EKS = Elastic Kubernetes Service: Kubernetes w chmurze Amazon Web Services, w którym płaszczyznę sterowania utrzymuje dostawca · **eksctl** = EKS control: narzędzie do tworzenia i usuwania klastrów Amazon EKS jedną komendą

Z życia wzięte

Z ŻYCIA WZIĘTE

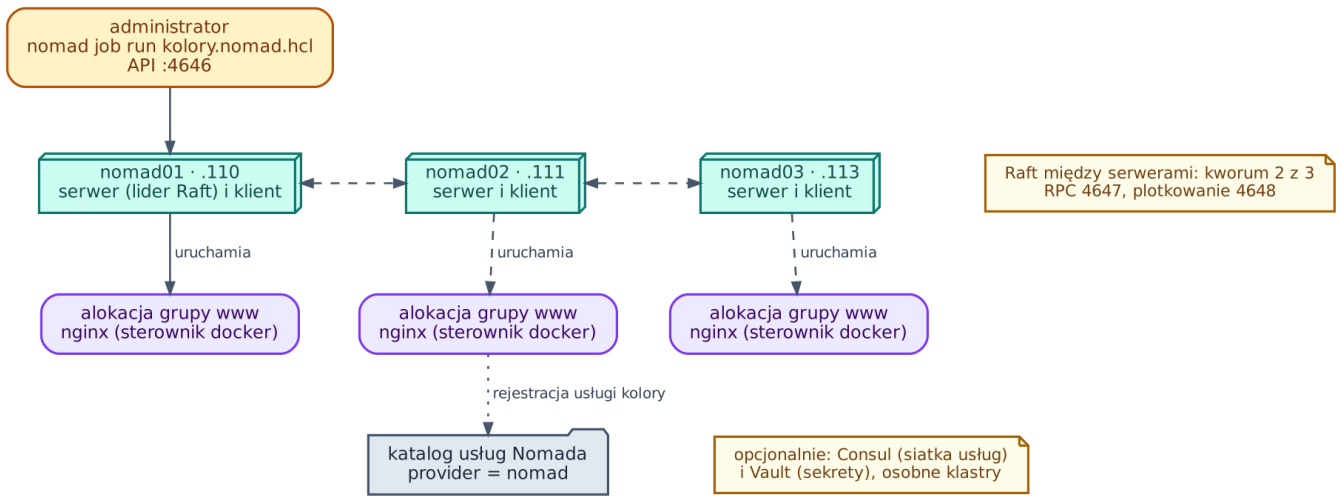
Piątek, 14:40. Tomek pokazuje Oli trzy maszyny z Nomadem, na których kolorowy nginx stanął po dwudziestu minutach, i mówi, że to „Kubernetes bez bólu”. Ola pyta o dwie rzeczy: kto nam pomoże, gdy Nomad 2.x zmieni zasady wsparcia, i czy pan Wojtek przyjmie licencję, której nie ma na liście licencji otwartych. W tym samym czasie pan Zbigniew przysyła link do AKS z dopiskiem „może po prostu w chmurze?”, a w poniedziałek okazuje się, że Optima i WMS i tak zostają w serwerowni. Trzy pomysły, trzy różne pytania, i dlatego ta strona opisuje każdy osobno.

Jak to działa

Rozwiń: jak to działa pod spodem

Nomad: jeden plik binarny, zadania i alokacje

> **Pojęcie: Nomad.** Orkiestrator HashiCorp, który planuje na klastrze maszyn nie tylko kontenery, ale też zwykłe programy, aplikacje Javy i maszyny wirtualne QEMU. Serwery i klienci to ten sam plik binarny uruchomiony z inną konfiguracją.



Rysunek 1. Laboratorium Nomada: trzy serwery w jednym zespole Raft z liderem na nomad01, te same maszyny jako klienci z Dockerem, zadanie kolory w trzech alokacjach i opcjonalne Consul i Vault obok (kliknięcie otwiera pełny rozmiar)

Opis pracy w Nomadzie ma trzy poziomy. Zadanie (job) to cały opis aplikacji, grupa (group) to zestaw zadań, które muszą stać razem na jednej maszynie, a task to jeden proces albo kontener. Każde umieszczenie grupy na kliencie to alokacja (allocation), czyli odpowiednik poda w Kubernetesie.

Nomad	Kubernetes	Swarm
job	Deployment z Service (całość aplikacji)	stos
group	Pod (kontenery, które stoją razem)	usługa
task	kontener w podzie	zadanie
allocation	uruchomiony Pod	uruchomione zadanie

Nomad	Kubernetes	Swarm
sterownik zadań (`docker`, `exec`, `java`, `qemu`)	zawsze środowisko uruchomieniowe kontenerów	zawsze Docker
`provider = "nomad"` albo Consul	Service i CoreDNS	DNS sieci overlay
Nomad Variables albo Vault	Secret	sekret Swarm

Do wykrywania usług Nomad od wersji 1.3 ma własny katalog usług, więc Consul nie jest już obowiązkowy [15]. Consul daje w zamian siatkę usług (service mesh) i bogatsze kontrole stanu, a Vault centralne sekrety. Każdy z nich to osobny klaster do utrzymania, więc w małym laboratorium zaczynamy bez nich.

Serwery Nomada uzgadniają stan protokołem Raft, tak jak menedżerowie Swarma. Dokumentacja zaleca trzy albo pięć serwerów w regionie, a wartość 1 w `bootstrap_expect` nie daje żadnej odporności [5][16]. Zasada kworum jest ta sama co w Swarmie i etcd: trzy serwery znoszą utratę jednego, szczegóły na stronie or2-01. Regiony Nomada nie dzielą się danymi, więc drugi region to osobny klaster, a nie kopia.

Porty: 4646/tcp (API i interfejs WWW), 4647/tcp (RPC między serwerami i klientami) i 4648 tcp i udp (protokół plotkowania między serwerami). Otwieramy je tylko między trzema węzłami i dla stacji administratora, tak jak porty Swarma w strefie firewalld na stronie o Swarmie.

Nomad: licencja i właściciel

Od 10 sierpnia 2023 roku nowe wydania Nomada są na licencji Business Source License, a dokumentacja HashiCorp mówi wprost, że Nomad Community Edition jest objęty BSL [17]. Licencja pozwala używać Nomada na własne potrzeby, zabrania oferowania go jako konkurencyjnej usługi. Od 27 lutego 2025 roku HashiCorp należy do IBM, a od Nomada 2.0.0 (21 kwietnia 2026) numeracja i cykl wsparcia idą według zasad IBM: nowa linia wsparcia w kwietniu, nowe funkcje w październiku, poprawki co miesiąc [6].

Linia	Koniec wsparcia podstawowego dla wersji społecznościowej
2.0.x	30 kwietnia 2028
1.11.x	31 października 2026
1.10.x (LTS)	30 kwietnia 2027

Dla Zielonej oznacza to dwie rzeczy. Licencja nie zabrania nam używania Nomada, ale pan Wojtek może zapytać, dlaczego wybraliśmy narzędzie z licencją nieotwartą, i odpowiedź musi być zapisana. Wsparcie dłuższe niż dwa lata jest płatne i dostępne tylko w wersji Enterprise, więc co dwa lata planujemy przejście na nową linię.

Podman z Quadlet: kontener jako usługa systemd

> **Pojęcie: Podman bez demona.** Podman uruchamia kontener jako zwykły proces potomny polecenia, bez stale działającego programu w tle, jakim jest `dockerd`. Dzięki temu kontener może należeć do zwykłego użytkownika i być pilnowany przez systemd jak każda inna usługa.

Quadlet łączy oba światy: plik `.container` opisuje obraz, porty i woluminy, a systemd zamienia go w usługę z restartem po awarii, zależnościami i dziennikiem. Pliki systemowe kładziemy w `/etc/containers/systemd/`, a pliki użytkownika w `~/.config/containers/systemd/` [7]. Oprócz `.container` są też pliki `.volume`, `.network` i `.kube`, ten ostatni uruchamia manifest Kubernetesa przez Podmana.

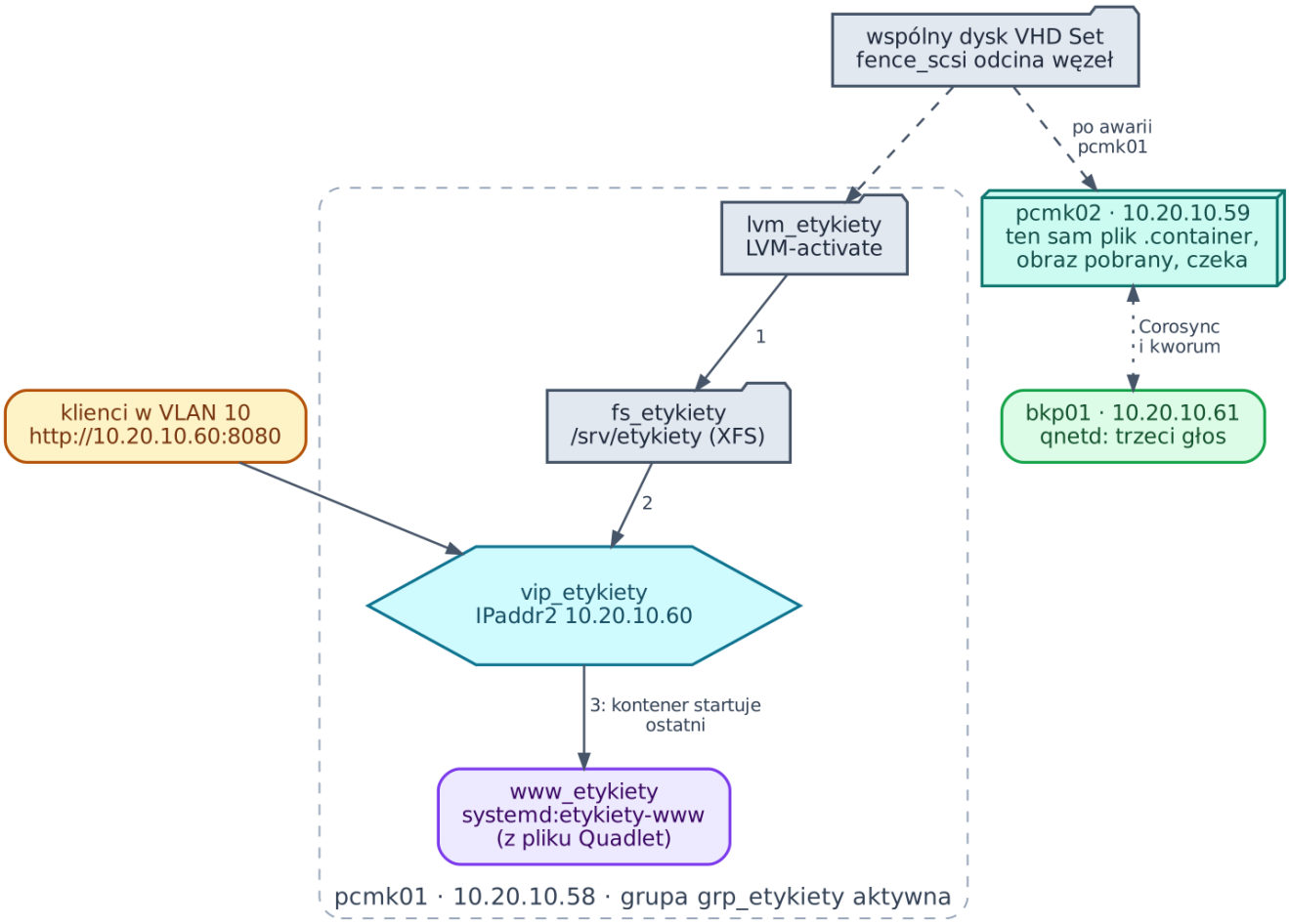
Sprawa	Podman z Quadlet	Docker z Compose na `docker01`
Kto pilnuje kontenera	systemd	demon Dockera z `restart:`
Konto	zwykły użytkownik albo root	w praktyce root (grupa `docker`)
Dzienniki	`journalctl -u kolory`	`docker logs`
Aktualizacja obrazu	`podman auto-update` z cofnięciem po nieudanym starcie	ręcznie albo narzędziem zewnętrznym
Wiele kontenerów razem	kilka plików `.container` z zależnościami albo plik `.kube`	jeden `compose.yaml`
Klaster	brak, jedna maszyna	brak, jedna maszyna

Rocky Linux 9.8 ma w repozytorium Podmana 5.8 [18]. Podman 6.0 z czerwca 2026 roku usunął obsługę cgroups v1, sieci CNI i iptables na rzecz Netavark i nftables [19]. Na Rocky Linux 9 nas to dziś nie dotyczy, ale przy przejściu na Rocky Linux 10 sprawdzamy, którą serię dostarczy dystrybucja.

Podman z Pacemakerem: wysoka dostępność bez orkiestratora

Quadlet pilnuje kontenera na jednej maszynie. Gdy maszyna pada, kontener pada razem z nią. Drugą maszynę dokłada Pacemaker, opisany w książce o wysokiej dostępności: zasób z kontenerem staje na końcu grupy, za wspólnym dyskiem, systemem plików i adresem

wirtualnym.



Rysunek 2. Podman z Quadlet w klastrze Pacemakera: grupa grp_etykiety na pcmk01 z wolumenem, systemem plików, adresem 10.20.10.60 i usługą etykiety-www z Quadlet; pcmk02 czeka, trzeci głos na bkp01, fencing przez fence_scsi (kliknięcie otwiera pełny rozmiar)

Ten układ ma sens dla jednej usługi stanowej, która musi działać na jednym węźle naraz i mieć dane na wspólnym dysku. Wszystkie zastrzeżenia ze stron Pacemaker na Linuksie: własna usługa z adresem wirtualnym i Pacemaker w utrzymaniu: zarządzanie, testy i aktualizacje zostają: bez fencingu nie wdrażamy, a przełączenie trwa dziesiątki sekund, bo kontener zatrzymuje się na jednym węźle i startuje na drugim.

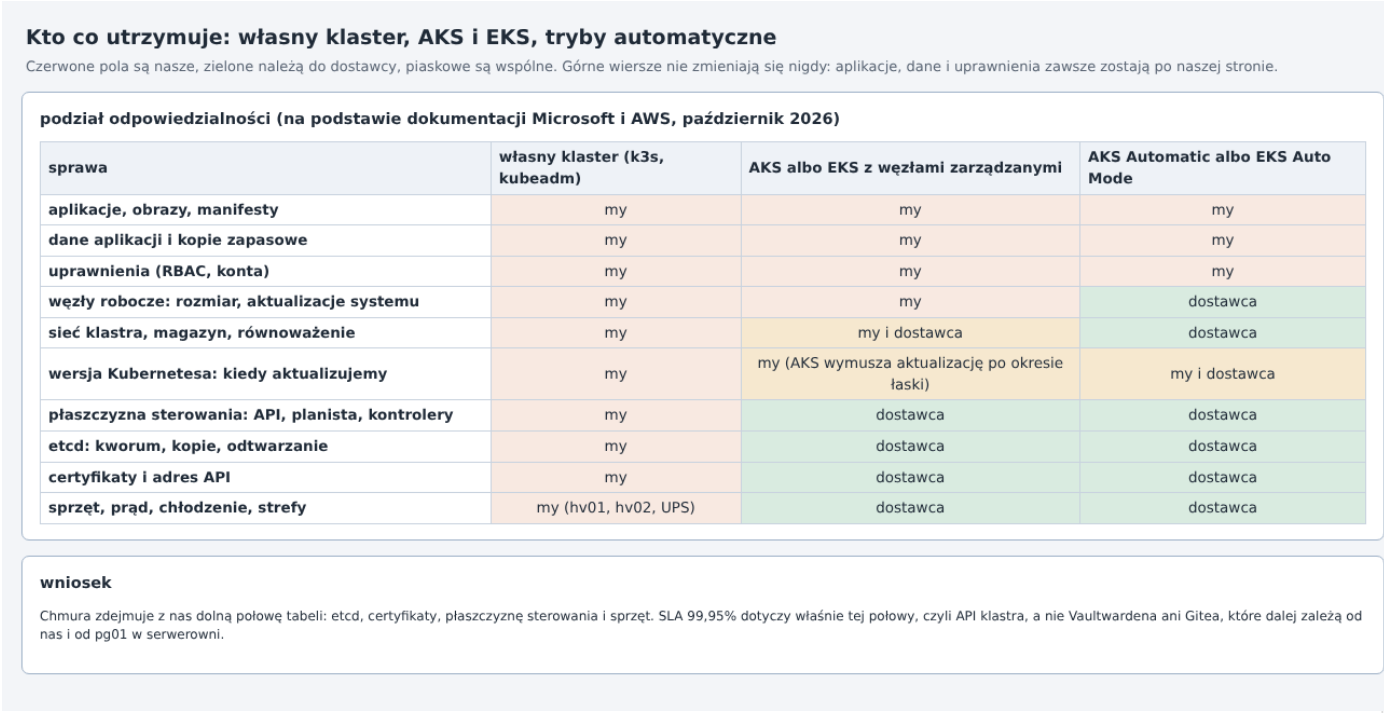
Sprawa	Podman z Pacemakerem	k3s z trzema serwerami
Minimum maszyn	dwie i trzeci głos (`qnetd` na `bkp01`)	trzy serwery

Sprawa	Podman z Pacemakerem	k3s z trzema serwerami
Dane	wspólny dysk albo DRBD, jeden węzeł zapisuje	Longhorn albo baza poza klastrem
Ile kopii usługi naraz	jedna	wiele
Aktualizacja bez przerwy	nie, przełączenie zawsze jest przerwą	tak, dla aplikacji bezstanowych
Czego trzeba się nauczyć	Pacemaker, pcs, fencing, Quadlet	Kubernetes, etcd, CNI, magazyn
Gdzie pasuje	pojedyncza usługa stanowa, którą znamy z systemd	wiele aplikacji pisanych pod kontenery

Uwaga praktyczna: Pacemaker uruchamia jednostki systemowe, nie jednostki użytkownika, więc plik `.container` dla klastra leży w `/etc/containers/systemd/`, a nie w katalogu domowym. Obraz pobieramy wcześniej na oba węzły (`sudo podman pull docker.io/library/nginx:1.30.5`), żeby przełączenie nie czekało na rejestr, a `AutoUpdate` w takim pliku wyłączamy, bo aktualizację przeprowadzamy świadomie, węzeł po węźle, według strony o utrzymaniu Pacemakera.

AKS i EKS: Kubernetes, którego płaszczyzny nie utrzymujemy

> **Pojęcie: zarządzany Kubernetes.** Usługa chmurowa, w której dostawca utrzymuje płaszczyznę sterowania (serwery API, etcd, planistę), jej kopie, certyfikaty i aktualizacje, a my dostajemy adres API i płacimy za węzły robocze. Azure nazywa ją AKS (Azure Kubernetes Service), AWS nazywa ją EKS (Elastic Kubernetes Service).



Rysunek 3. Kto co utrzymuje: we własnej serwerowni wszystko jest nasze, w AKS i EKS płaszczyzna sterowania należy do dostawcy, a w trybach AKS Automatic i EKS Auto Mode dostawca przejmuje także węzły; aplikacje, dane i uprawnienia zostają zawsze po naszej stronie (kliknięcie otwiera pełny rozmiar)

> **Pojęcie: strefa dostępności.** Oddzielne centrum danych (albo grupa centrów) w jednym regionie chmury, z własnym zasilaniem, chłodzeniem i siecią. Maszyny rozłożone na trzy strefy przeżywają awarię jednej z nich.

Obu dostawców opisujemy równorzędnie, bo różnią się szczegółami, a nie zasadą.

Sprawa	AKS (Azure)	EKS (AWS)
Płaszczyzna sterowania	utrzymuje Microsoft; w regionie ze strefami domyślnie rozłożona na strefy, nawet gdy węzły ich nie używają [10][11]	utrzymuje AWS; co najmniej dwa serwery API i trzy węzły etcd w trzech strefach, wymiana uszkodzonych instancji automatycznie [20]
Węzły robocze	pule węzłów (zestawy skalowania maszyn), strefy wybieramy przy tworzeniu puli i nie zmieniamy później [11]	grupy węzłów zarządzanych, węzły własne albo Fargate
Tryb, w którym dostawca bierze też węzły	AKS Automatic (domyślnie poziom Standard) [10]	EKS Auto Mode [13][20]

Sprawa	AKS (Azure)	EKS (AWS)
SLA płaszczyzny (informacyjnie)	poziom Free bez SLA; Standard i Premium 99,95% ze strefami dostępności i 99,9% bez nich [10]	99,95% dla standardowej płaszczyzny sterowania i 99,99% dla płaszczyzny typu Provisioned [21]
Wersje Kubernetesa (październik 2026)	1.34 do 1.37, 1.37 dostępna od października 2026 [22]	1.34 do 1.37, 1.37 w EKS od 1 października 2026 [14]
Wsparcie wersji dłużej niż społeczność	LTS na 24 miesiące w poziomie Premium [10]	wsparcie rozszerzone o 12 miesięcy po standardowym [14]
Wymuszona aktualizacja	po okresie łaski AKS sam podnosi klastry z wersji niewspieranych, a od 1 czerwca 2027 roku bez oglądania się na okna serwisowe [22]	zasady aktualizacji po końcu wsparcia rozszerzonego sprawdzamy w dokumentacji EKS przed wyborem wersji
Narzędzie w Fiszce	`az aks`	`eksctl` i `aws eks`

Co zarządzany Kubernetes zdejmuje z dwuosobowego zespołu: kworum etcd, kopie płaszczyzny sterowania, certyfikaty, adres wirtualny API i wysoką dostępność samego API. Czego nie zdejmuje: aplikacji, danych, sieci między chmurą a serwerownią, uprawnień, kosztów i obowiązku aktualizowania wersji co kilkanaście miesięcy. SLA dotyczy tylko dostępności API klastra, a nie naszych aplikacji, więc nawet 99,95% płaszczyzny nie mówi nic o tym, czy Vaultwarden będzie działał.

Dla Zielonej jest jeszcze trzeci koszt: odległość. Aplikacje na `docker01` korzystają z `pg01`, Active Directory na `dc01` i `dc02` i poczty w tej samej sieci, a klaster w Azure albo AWS rozmawiałby z nimi przez tunel. Dlatego chmurę traktujemy jako wariant dla nowych aplikacji, które nie zależą od serwerowni, a nie jako miejsce na przeprowadzkę istniejących. Cen nie podajemy; płaszczyzna sterowania w poziomie Standard AKS i w EKS jest płatna od godziny, a węzły i ruch sieciowy liczone osobno.

Częste błędy

Objaw	Przyczyna	Co robimy
Serwery Nomada nie wybierają lidera	`bootstrap_expect` większe niż liczba serwerów albo zablokowane porty 4647 i 4648	ta sama wartość 3 na trzech serwerach, porty w strefie firewalld
`nomad job run` kończy się kodem 2	brak zasobów albo sterownika na klientach	`nomad job status` i sekcja Placement Failure; sprawdzamy, czy na klientach działa Docker
Ostrzeżenia o przestarzałym `retry_join` po aktualizacji do 2.0	stare opcje na poziomie bloku `server`	przenosimy adresy do bloku `server_join` przed wersją 2.1

Objaw	Przyczyna	Co robimy
`systemctl enable kolory.service` zgłasza błąd	jednostki Quadlet są generowane	o starcie decyduje sekcja `[Install]` w pliku `.container`
Kontener użytkownika znika po wylogowaniu	brak lingeringu	`sudo loginctl enable-linger nazwa_usera`
`podman auto-update` nie aktualizuje kontenera	krótka nazwa obrazu bez rejestru	pełna nazwa, na przykład `docker.io/library/nginx:1.30.5`
Kontener w klastrze Pacemakera startuje sam po restarcie węzła	sekcja `[Install]` w pliku `.container`	usuwamy `[Install]`, start należy tylko do Pacemakera
Kontener nie startuje na adresie 10.20.10.60	adres wirtualny jeszcze nie istnieje na węźle	zasób z kontenerem na końcu grupy, za adresem
`az aks create --zones` zgłasza błąd rozmiaru maszyny	rozmiar nie obsługuje stref w regionie	`az aks list-vm-skus` i inny rozmiar
Po próbach w AWS przychodzi rachunek	zostały wolumeny i równoważniki po usuniętym klastrze	`eksctl delete cluster` i przegląd zasobów w konsoli

□□ Ćwiczenia

1. Nomad, Swarm i etcd w Kubernetesie używają tej samej zasady kworum. Ile serwerów Nomada trzeba, żeby przeżyć awarię dwóch naraz, i dlaczego nie cztery?

Odpowiedź

Pięć: większość z pięciu to trzy, więc dwa mogą paść. Cztery serwery potrzebują większości trzech, więc znoszą awarię tylko jednego, tak jak trzy, a dokładają jeszcze jedną maszynę, która może zawieść. Dlatego dokumentacja zaleca trzy albo pięć serwerów.

1. Dlaczego plik `.container` dla Pacemakera nie ma sekcji `[Install]`, a plik użytkownika ją ma?

Odpowiedź

Plik użytkownika opisuje usługę, która ma startować z systemem, więc `[Install]` z `WantedBy=default.target` jest potrzebny, a `systemctl enable` nie działa dla jednostek Quadlet. W klastrze o tym, gdzie i kiedy kontener działa, decyduje wyłącznie Pacemaker: gdyby kontener startował z systemem, mógłby wstać na węźle pasywnym bez danych i adresu, obok kopii na węźle aktywnym.

1. Pan Zbigniew czyta, że AKS ma SLA 99,95%. Co odpowiadamy na pytanie, czy Vaultwarden w AKS będzie dostępny przez 99,95% czasu?

Odpowiedź

Nie wiadomo. SLA dotyczy dostępności API klastra (płaszczyzny sterowania) w poziomie Standard ze strefami, a nie naszej aplikacji. Dostępność Vaultwardena zależy od liczby kopii, magazynu danych, sieci do serwerowni, poczty i od nas. Do tego Vaultwarden trzyma dane w SQLite, więc w klastrze i tak działa w jednej kopii.

1. Czym różni się odpowiedzialność za węzły robocze w EKS z grupą węzłów zarządzanych i w EKS Auto Mode?

Odpowiedź

W obu przypadkach płaszczyznę sterowania utrzymuje AWS. Przy grupie węzłów zarządzanych AWS pomaga je tworzyć i wymieniać, ale wybór rozmiaru, liczby i aktualizacji węzłów zostaje po naszej stronie. W Auto Mode AWS przejmuje także zarządzanie węzłami, magazynem i siecią klastra, a my odpowiadamy za aplikacje, dane i uprawnienia.

Powiązane: Orkiestracja kontenerów: od Borga do wojny orkiestratorów i tego, co z niej zostało · Swarm, k3s i Kubernetes: porównanie trzech orkiestratorów · Dystrybucje Kubernetesa: kubeadm, k3s, RKE2, k0s, MicroK8s, Talos i OpenShift · Pacemaker na Linuksie: własna usługa z adresem wirtualnym · Pacemaker w utrzymaniu: zarządzanie, testy i aktualizacje · Chmura: cudzy komputer, za który płacimy za minutę · Azure: podstawy operacyjne · AWS CLI: instalacja, profile, poświadczenia

Źródła: (odczyt 14 do 18 października 2026)

1. developer.hashicorp.com: Install Nomad
2. developer.hashicorp.com: nomad job run
3. developer.hashicorp.com: nomad job status
4. developer.hashicorp.com: nomad alloc logs
5. developer.hashicorp.com: Nomad server block (`bootstrap_expect`, `server_join`)
6. developer.hashicorp.com: Nomad 2.0.x release notes (2.0.0 z 21 kwietnia 2026, 2.0.7 z 17 września 2026, opcje do usunięcia w 2.1)
7. docs.podman.io: podman-systemd.unit(5)
8. docs.podman.io: podman-auto-update(1)
9. mankier.com: ocf_heartbeat_podman(7)

10. learn.microsoft.com: AKS Free, Standard and Premium pricing tiers
11. learn.microsoft.com: Availability zones in AKS
12. docs.aws.amazon.com: Get started with Amazon EKS: eksctl
13. docs.aws.amazon.com: Create an EKS Auto Mode Cluster with the eksctl CLI
14. docs.aws.amazon.com: Amazon EKS Kubernetes versions
15. hashicorp.com: Nomad 1.3 adds native service discovery and edge workload support
16. developer.hashicorp.com: Nomad architecture
17. developer.hashicorp.com: Nomad CE license and support
18. git.almalinux.org: rpms/podman, import podman-5.8.2-1.el9_8
19. linuxiac.com: Podman 6.0 Lands with Breaking Changes; endoflife.ai: Podman 6.1 (6.1.4 z 12 października 2026)
20. docs.aws.amazon.com: Amazon EKS architecture
21. aws.amazon.com: Amazon EKS Service Level Agreement
22. learn.microsoft.com: Supported Kubernetes versions in AKS

Revision #5
Created 2026-10-18 08:52:53 UTC by Tomasz Konieczny
Updated 2026-10-26 14:26:06 UTC by Tomasz Konieczny