cloudware

Jak budować DevOps na OpenShift i IBM Power

IBM Common Polska 2026

Konrad Rzentarzewski

Dziekońskiego 1
00-728 Warszawa

Agenda

1. Po co Power w OpenShift i czym jest klaster multiarch
2. Instalacja: payload Multi, dołączanie workerów ppc64le, pułapki
3. **DEMO 1:** aplikacja multiarch — Kustomize + CLI
4. **DEMO 2:** GitOps z ArgoCD — zmiana architektury commitem
5. Wnioski i dobre praktyki

Cloudware 2026. Wszelkie prawa zastrzeżone.

Dlaczego Power w OpenShift?

Praca bliżej danych
Workloady obok IBM i, AIX i Db2 na tej samej platformie sprzętowej

Gęstość i wydajność na rdzeń
Mniej węzłów (i licencji) dla tych samych zadań

PowerVS
Power jako usługa w IBM Cloud: LPAR-y na żądanie, bez własnego sprzętu

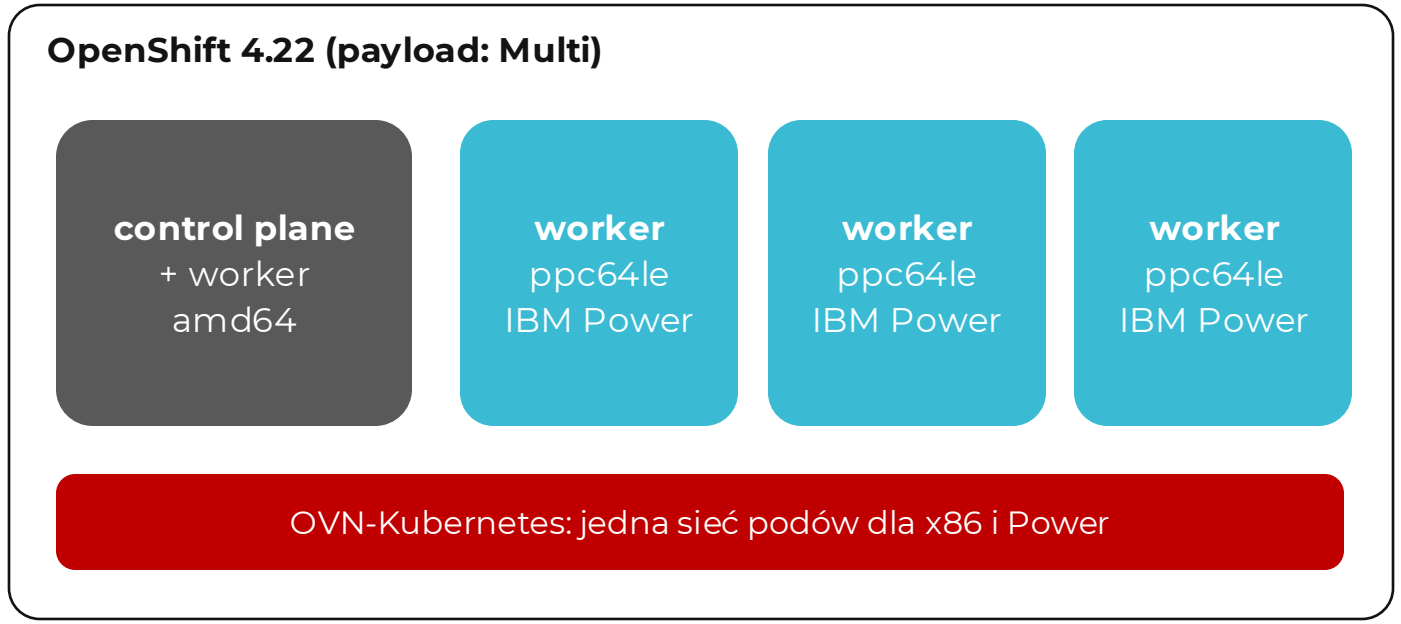
Jedna platforma aplikacyjna
Ten sam OpenShift, oc, GitOps i Pipelines — niezależnie od CPU

Cel to nie „przenieść wszystko na Power”, tylko ***„każdy komponent uruchomić tam, gdzie ma to sens”***

Cloudware 2026. Wszelkie prawa zastrzeżone.

Klaster multiarch — o co chodzi

OpenShift 4.22 • payload Multi • OVN-Kubernetes



- `kubernetes.io/arch` — etykieta każdego węzła: amd64, ppc64le, s390x, arm64
- **Obraz wieloarchitekturowy** (manifest list): jeden tag, osobny wariant dla każdej architektury; CRI-O pobiera pasujący
- **Scheduler nie wie**, czy obraz ma wariant dla danej architektury

Klaster demo: 1× x86 (SNO: control plane + worker) + 3× IBM Power (ppc64le)

Cloudware 2026. Wszelkie prawa zastrzeżone.

Część 1 — Instalacja

payload Multi · ignition · RHCOS ppc64le · pułapki z labu

Dziekońskiego 1
00-728 Warszawa

Ścieżki instalacji z Power

Klaster w całości na PowerVS	Klaster x86 + workery Power	On-prem PowerVM / platforma None
instalator IPI `platform: powervs`	**multi-arch compute** `payload Multi + węzły ppc64le` DZIŚ	**UPI** `ignition + boot RHCOS ppc64le, ręczne CSR`

Dzisiejsze demo: multi-arch compute — istniejący klaster x86, do którego dołączamy trzy workery ppc64le

cloudware
Dziekońskiego 1
00-728 Warszawa

Cloudware 2026. Wszelkie prawa zastrzeżone.

Krok 1: payload Multi

Bez niego węzeł ppc64le nie wstanie

1

- Payload single-arch: obrazy systemowe (MCD, OVN, multus, DNS…) mają tylko wariant amd64
- Operacja **jednokierunkowa** i dla całego klastra
- Bez ręcznych digestów, bez --force
- ConfigMap coreos-bootimages wymienia ppc64le także przy single-arch — to **nie jest** dowód multiarch

```
# jaki payload ma klaster?
oc get clusterversion version \
  -o jsonpath='{.status.conditions[?(@.type==
  "ReleaseAccepted")].message}'
#    ... architecture="amd64"   <- single-arch

# migracja w miejscu, na tej samej wersji
oc adm upgrade --to-multi-arch
#    ... architecture="Multi"
```


Cloudware 2026. Wszelkie prawa zastrzeżone.

Krok 2: ignition dla workera ppc64le

Stub wskazuje na Machine Config Server

2

```
{ "ignition": { "version": "3.4.0",
  "config": { "merge": [ { "source":
    "https://api-int.<cluster>:22623/config/worker" } ] },
  "security": { "tls": { "certificateAuthorities": [
    { "source": "data:...; base64,<mcs-ca>" } ] } } } }
```

```
oc get cm -n openshift-machine-config-operator \
  machine-config-server-ca \
  -o jsonpath='{.data.ca-bundle\.crt}' | base64
```

Serwowanie
mały httpd w klastrze + NodePort — instalator pobiera ignition, zanim jest dostępny DNS

Stabilność
CA MCS nie zrotowało przez 3 upgrade'y minor — nie trzeba było poprawiać pliku *stub*

Cloudware 2026. Wszelkie prawa zastrzeżone.

Krok 3: boot RHCOS ppc64le i dołączenie węzła

3 → GRUB / netboot → ignition → MCS: rendered-worker → CSR × 2 → Node Ready

```
coreos.live.rootfs_url=<rootfs ppc64le>
coreos.inst.install_dev=/dev/sda
coreos.inst.ignition_url=http://<ip>:30080/<ign>
ip=<ip>::<gw>:<maska>:<hostname>:env6:none
nameserver=<dns>
```

```
oc get csr -w
oc get csr -o name | \
  xargs oc adm \
  certificate approve
oc get nodes \
  -L kubernetes.io/arch
```

Na PowerVM / PowerVS praktyczniej bootować przez GRUB/netboot (kernel + initramfs + rootfs) niż z ISO

Cloudware 2026. Wszelkie prawa zastrzeżone.

Pułapki z labu

czyli jeśli coś może pójść nie tak, to...

1

Hostname PRZED dołączeniem

Nazwa węzła jest niezmienna przez cały czas życia klastra. Po zmianie: odrzucenia NodeRestriction i brak chassis OVN = brak ruchu pod↔pod;

W praktyce można zainstalować od nowa.

2

Pamięć LPAR-a

~1 GB RAM: firstboot-osupdate pada z OOM. Minimum dla workera to 8 GiB

3

Nazwa interfejsu

Nie jest jednolita (env6 vs env7) — czytać z konsoli konkretnego LPAR-a

4

„API działa” ≠ „sieć działa”

ClusterIP API idzie po sieci fizycznej, z pominięciem sieci podów

Cloudware 2026. Wszelkie prawa zastrzeżone.

Efekt

OpenShift 4.22.8 • payload Multi • wszystkie Cluster Operators zdrowe

```
$ oc get nodes -L kubernetes.io/arch
NAME                 STATUS  ROLES                  VERSION  ARCH
00-50-56-3e-a2-38    Ready   control-plane,worker v1.35.6  amd64
56-27-80-8a-4b-07    Ready   worker               v1.35.6  ppc64le
56-27-84-65-50-06    Ready   worker               v1.35.6  ppc64le
56-27-84-cd-4c-06    Ready   worker               v1.35.6  ppc64le
```

1× amd64

3× ppc64le

Instancja OpenShift GitOps sama działa już na węzłach Power (multiarch)

Cloudware 2026. Wszelkie prawa zastrzeżone.

Część 2 — Aplikacja multiarch

Sock Shop · obrazy wieloarchitekturowe · nodeSelector · Kustomize

Dziekońskiego 1
00-728 Warszawa

Sock Shop — mikroserwisy w wersji multiarch

ocp-power-demos/sock-shop-demo • Apache 2.0 • Java, Go, Node.js + MongoDB, MariaDB, Redis, RabbitMQ

Rodzaj	Komponenty	Obraz
Multiarch	front-end, carts, catalogue, orders, payment, shipping, user, queue-master, session-db	wieloarchitekturowy: amd64, arm64, ppc64le, s390x
Przypięte	carts-db, orders-db, catalogue-db, user-db, rabbitmq	tylko amd64 (lub osobny tag :ppc64le)

9 komponentów multiarch

5 przypiętych do architektury

Cloudware 2026. Wszelkie prawa zastrzeżone.

Dwa wzorce rozmieszczenia

(1) Komponent multiarch

```
spec:
  containers:
    - image: quay.io/powercloud/
        sock-shop-front-end:latest
        # wieloarchitekturowy
  nodeSelector:
    node-role.kubernetes.io/worker: ""
```

(2) Komponent przypięty

```
spec:
  containers:
    - image: quay.io/mongodb/mongodb:
        org-4.4-standalone-ubuntu2204
        # tylko amd64
  nodeSelector:
    node-role.kubernetes.io/worker: ""
    kubernetes.io/arch: amd64
```

Bez przypięcia: ImagePullBackOff (no matching manifest for linux/ppc64le) albo *Exec format error*

Cloudware 2026. Wszelkie prawa zastrzeżone.

Jak sprawdzić obraz, zanim trafi do klastra

```
$ oc image info --show-multiarch \
    quay.io/powercloud/sock-shop-carts:latest | grep Arch
Arch:    amd64
Arch:    arm64
Arch:    ppc64le
Arch:    s390x

$ oc image info --show-multiarch \
    quay.io/mongodb/mongodb:org-4.4-standalone-ubuntu2204 \
    | grep Arch
Arch:    amd64
```

Bramka w CI
obraz bez ppc64le nie trafia do komponentu bez nodeSelectora

registry.redhat.io
dodaj -a <pull-secret.json>
(RH wymaga uwierzytelnienia kontem RHN)

Cloudware 2026. Wszelkie prawa zastrzeżone.

Kustomize: struktura repozytorium

```
sock-shop/
├── upstream/manifests/base  # projekt
├── components/
│   ├── roadrunner/    # poprawki pod mój klaster
│   └── db-on-power/   # przypięte -> Power
└── overlays/
    ├── dev/           # wdrażamy z CLI
    └── prod/          # wdrażamy przez ArgoCD
gitops/argocd/         # AppProject + Application
```

base
nie ruszamy cudzego kodu — łatwy update z upstream

component
wielokrotnego użytku „funkcja” włączana jednym przełącznikiem

overlay
różnice per środowisko: namespace, etykiety, limity, Route

Cloudware 2026. Wszelkie prawa zastrzeżone.

Component roadrunner

dopasowanie manifestów upstream do naszego klastra

```
kind: Component
resources: [namespace.yaml]    # zamiast Project
patches:
  - target: {kind: PersistentVolumeClaim}
    patch: |-
      - {op: replace, path: /spec/storageClassName,
         value: nfs}           # upstream: nfs-client
  - target: {kind: Service, name: front-end}
    patch: |-
      - {op: replace, path: /spec/type, value: ClusterIP}
      - {op: remove, path: /spec/ports/0/nodePort}
secretGenerator:
  - name: mongodb-creds
    literals: [username=root, password=admin]
```

Namespace
Zamiast project.openshift.io/Project — da się go zaaplikować przez apply i ArgoCD

ClusterIP
Ustawiony na sztywno NodePort 30001 kolidowałby między środowiskami

Secret
To tylko demo! Na produkcji: External Secrets (VSO) / Sealed Secrets (SOPS)

Cloudware 2026. Wszelkie prawa zastrzeżone.

Overlay dev

```
kind: Kustomization
namespace: sock-shop-dev
resources:
  - ../../upstream/manifests/base
components:
  - ../../components/roadrunner
labels:
  - pairs: {app.kubernetes.io/part-of: sock-shop, env: dev}
patches:
  - target: {kind: Route, name: sock-shop}
    patch: |-
      - {op: replace, path: /spec/subdomain,
         value: sock-shop-dev}
```

`namespace`
przemianowuje też obiekt Namespace z komponentu

`components`
włączenie poprawek klastra jedną linijką

`subdomain`
unikalny host Route per środowisko — inaczej będzie kolizja (HostAlreadyClaimed)

Cloudware 2026. Wszelkie prawa zastrzeżone.

DEMO 1
wdrożenie z CLI

DEMO

Kustomize + oc:
render, diff, apply, rozmieszczenie

```
# 1. co powstanie? (dry run; render lokalny)
oc kustomize sock-shop/overlays/dev | less

# 2. co się zmieni w klastrze?
oc diff -k sock-shop/overlays/dev

# 3. wdrożenie
oc apply -k sock-shop/overlays/dev

# 4. sprawdzamy gdzie wylądował każdy pod
./scripts/show-placement.sh sock-shop-dev
```

oc ma wbudowany Kustomize (-k) — osobny binarny kustomize nie jest konieczny (choć przydatny)

Cloudware 2026. Wszelkie prawa zastrzeżone.

DEMO 1 — wynik

./scripts/show-placement.sh sock-shop-dev

Pod	Węzeł	Przypięcie
carts	ppc64le	multiarch
carts-db	amd64	amd64
catalogue	ppc64le	multiarch
catalogue-db	amd64	amd64
front-end	ppc64le	multiarch
orders	ppc64le	multiarch
orders-db	amd64	amd64

Pod	Węzeł	Przypięcie
payment	ppc64le	multiarch
queue-master	ppc64le	multiarch
rabbitmq	amd64	amd64
session-db	ppc64le	multiarch
shipping	ppc64le	multiarch
user	ppc64le	multiarch
user-db	amd64	amd64

9
na IBM Power

5
na x86

Sklep działa jako całość: rejestracja → logowanie → koszyk — ruch płynie przez obie architektury

Cloudware 2026. Wszelkie prawa zastrzeżone.

Część 3 — GitOps z ArgoCD

AppProject · Application · managed-by · zmiana architektury commitem

Dziekońskiego 1
00-728 Warszawa

OpenShift GitOps: dwa obiekty

AppProject

```
kind: AppProject
metadata:
  name: ocp-power-demo
spec:
  sourceRepos:
    - http://gitlab.../ocp-power-demo.git
  destinations:
    - namespace: sock-shop-*
  clusterResourceWhitelist:
    - {group: "", kind: Namespace}
```

Application

```
kind: Application
spec:
  project: ocp-power-demo
  source:
    repoURL: http://gitlab.../ocp-power-demo.git
    targetRevision: main
    path: sock-shop/overlays/prod  # Kustomize
  destination: {namespace: sock-shop-prod}
  syncPolicy:
    automated: {prune: true, selfHeal: true}
```

ArgoCD sam wykrywa że istnieje plik *kustomization.yaml* w ścieżce i buduje overlay

Cloudware 2026. Wszelkie prawa zastrzeżone.

Haczyk: uprawnienia do namespace

domyślna instancja openshift-gitops nie ma uprawnień do zarządzania dowolnym namespacem

```
patches:
  - target: {kind: Namespace}
    patch: |-
      kind: Namespace
      metadata:
        name: sock-shop
        labels:
          argocd.argoproj.io/managed-by: openshift-gitops
        annotations:
          argocd.argoproj.io/sync-wave: "-1"
```

1. ArgoCD tworzy Namespace (fala -1)
2. operator widzi managed-by i nadaje role
3. sync reszty zasobów (retry)

retry w syncPolicy — pierwsza próba może wyprzedzić nadanie uprawnień przez operatora (asynchronicznie)

Cloudware 2026. Wszelkie prawa zastrzeżone.

DEMO 2

rejestracja aplikacji w ArgoCD

DEMO

ArgoCD:
ten sam Kustomize,
źródłem prawdy jest Git

```
# konfigurujemy repo z deploy tokenem
GL_USER=... GL_TOKEN=... \
  ./gitops/argocd/create-repo-secret.sh

# dodajemy AppProject + Application
oc apply -k gitops/argocd

# status
oc get application sock-shop-prod \
  -n openshift-gitops
./scripts/show-placement.sh sock-shop-prod
```


Cloudware 2026. Wszelkie prawa zastrzeżone.

DEMO 2 — zmiana architektury jednym commitem

przypięte komponenty z x86 na Power

```
  components:
    - ../../components/roadrunner
+   - ../../components/db-on-power
```

```
git commit -am "prod: bazy na IBM Power"
git push
oc annotate application sock-shop-prod \
  -n openshift-gitops \
  argocd.argoproj.io/refresh=normal
```

przed	po
carts-db · amd64	**carts-db** · ppc64le
orders-db · amd64	**orders-db** · ppc64le
catalogue-db · amd64	**catalogue-db** · ppc64le
user-db · amd64	**user-db** · ppc64le
rabbitmq · amd64	**rabbitmq** · ppc64le

Na demo nie czekaliśmy na polling Gita (co ~3 min) — wymuszamy odświeżenie adnotacją albo przyciskiem Refresh

Cloudware 2026. Wszelkie prawa zastrzeżone.

Component db-on-power

obraz jednoarchitekturowy wymaga dwóch zmian jednocześnie

1. nodeSelector: amd64 → ppc64le

```
patches:
  - target:
      kind: Deployment
      name: (carts-db|orders-db|user-db|
             catalogue-db|rabbitmq)
    patch: |-
      - {op: replace, value: ppc64le, path:
         /spec/template/spec/nodeSelector/
           kubernetes.io~1arch}
```

2. obraz → wariant ppc64le

```
images:
  - name: quay.io/mongodb/mongodb
    newName: icr.io/ppc64le-oss/mongodb-ppc64le
    newTag: 4.4.24
  - name: quay.io/powercloud/sock-shop-user-db
    newTag: ppc64le
  # + catalogue-db, rabbitmq-exporter
```

Tylko nodeSelector → pod na Power z obrazem amd64: exec format error / no matching manifest

Tylko obraz → wariant ppc64le zostaje na x86, bo nodeSelector nadal mówi amd64

Cloudware 2026. Wszelkie prawa zastrzeżone.

DEMO 2 — self-heal i rollback

```
# ktoś na szybko „naprawia” produkcję...
oc scale deploy/front-end \
  -n sock-shop-prod --replicas=0
# ...ArgoCD przywraca stan z Gita (selfHeal)

# rollback = git revert
git revert HEAD && git push
# bazy wracają na x86
```

Kustomize CLI
szybka iteracja, oc diff, środowisko dev

ArgoCD
audyt w Git, wykrywanie dryfu, self-heal — ten sam kod

Zweryfikowane w labie: replicas=0 → przywrócone po ~4 sekundach

Cloudware 2026. Wszelkie prawa zastrzeżone.

Trzy poziomy dojrzałości rozmieszczenia



Multiarch Tuning Operator
affinity wyliczana automatycznie z obrazów — bez edycji manifestów



nodeAffinity + preferencje
obraz ma część architektur; wskazujemy dozwolone i preferujemy Power



nodeSelector: kubernetes.io/arch
obraz jednoarchitekturowy — prosto i jawnie

```
# srebro: nodeAffinity
affinity:
  nodeAffinity:
    required...:
      nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/arch
            operator: In
            values: [amd64, ppc64le]
    preferred...:
      - weight: 100  # preferuj Power
```

 **Złoto** zdejmuje z zespołów aplikacyjnych obowiązek pamiętania o architekturze

Cloudware 2026. Wszelkie prawa zastrzeżone.

Wnioski

1 Payload Multi

to warunek wstępny — oc adm upgrade --to-multi-arch, wystarczy to zrobić raz na klaster

2 Hostname

ustaw przed dołączeniem węzła, inaczej czekają Cię kłopoty

3 Sprawdzaj obrazy

oc image info --show-multiarch

4 Najsłabszy kontener

pod jest tak przenośny, jak jego najmniej przenośny kontener (sidecary!)

5 Kustomize components

zamiast kopii overlayów — zmiana architektury to jedna linia

6 GitOps

zmienia imperatywną migrację między architekturami w powtarzalny proces code review: diff, commit, revert

Cloudware 2026. Wszelkie prawa zastrzeżone.

Materiały

1. **Repo demo** — manifesty, skrypty, README z historią instalacji labu
   `napisz do mnie: /in/asquelt`

2. **Sock Shop multiarch** — aplikacja demo z overlayami Kustomize
   `github.com/ocp-power-demos/sock-shop-demo`

3. **Dokumentacja OpenShift** — multi-arch compute i Multiarch Tuning Operator
   `Configuring multi-architecture compute machines`

4. **Red Hat OpenShift GitOps** — zarządzanie namespace'ami przez instancję ArgoCD
   `argocd.argoproj.io/managed-by`

Cloudware 2026. Wszelkie prawa zastrzeżone.

cloudware

Dziękuję!

Pytania? Napisz do mnie: **/in/asquelt**

Dziekońskiego 1
00-728 Warszawa