SCIĄGA LARAVEL · DEPLOYMENT

Laravel na k3s

dominik-dev.pl

seria (4 części) →

Workload, kontrakt kontenera, rollout, CI z migracją-Job, operacje dnia drugiego

TL;DR

Zacznij od **małego, ale poprawnego** workloadu i ścieżki awarii, zanim dołożysz funkcje. Zdrowy deployment to pierwsza obserwowalna jednostka — **nie** dowód disaster recovery ani bezpieczeństwa. Migracje to nazwany krok release'u, nigdy start poda.

1. Bazowy workload (Deployment + Service + Ingress)

```
kind: Deployment            # namespace: storefront
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
    rollingUpdate: { maxUnavailable: 0, maxSurge: 1 }   # na 1 VPS rollout raczej poczeka niż ubije
poda
  revisionHistoryLimit: 5
  template:
    spec:
      terminationGracePeriodSeconds: 30
      containers:
        - name: web
          image: registry.example.com/acme/store:3f18d7c   # tag = SHA commita
          ports: [{ containerPort: 8080, name: http }]
          envFrom: [{ secretRef: { name: storefront-runtime } }]  # referencja, nie manifest sekretu
          resources:
            requests: { cpu: 100m, memory: 256Mi }
            limits:   { cpu: 500m, memory: 512Mi }
          readinessProbe: { httpGet: { path: /up, port: http }, periodSeconds: 10 }
          livenessProbe:  { httpGet: { path: /up, port: http }, initialDelaySeconds: 20 }
          lifecycle: { preStop: { exec: { command: ["/bin/sh","-c","sleep 5"] } } }
```

PROBE Readiness = pod może przyjąć ruch. Liveness = proces nie utknął. **Nie** odpytuj bazy z liveness — awaria bazy ma odebrać ruch przez readiness, nie restartować zdrowe procesy web. Sekrety k8s są tylko base64 — twórz wartości poza Gitem (external-secrets / chroniony krok).

2. Kontener jest częścią kontraktu

```
FROM php:8.5-cli-alpine AS application
WORKDIR /var/www/html
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer
COPY composer.json composer.lock ./
RUN composer install --no-dev --prefer-dist --no-interaction --no-scripts
COPY . .
RUN composer dump-autoload --classmap-authoritative --no-dev \
    && addgroup -S laravel && adduser -S laravel -G laravel \
```

```
    && chown -R laravel:laravel storage bootstrap/cache
USER laravel
EXPOSE 8080
CMD ["php","artisan","octane:start","--server=frankenphp","--host=0.0.0.0","--port=8080"]
```

Assety + zależności Composera w obrazie, konfiguracja ze środowiska. Jeden proces na pierwszym planie, udokumentowany port, non-root, obraz uruchamialny lokalnie pod tymi samymi nazwami zmiennych. Octane → sprawdź singletony i stan statyczny przed skalowaniem (PHP-FPM za nginx jest równie poprawne).

3. CI/CD — buduj raz, promuj ten sam artefakt

```
stages: [test, build, migrate, deploy, verify]
variables: { IMAGE_TAG: "$CI_COMMIT_SHA" }

build:
  script:
    - docker build --pull -t "$CI_REGISTRY_IMAGE:$IMAGE_TAG" .
    - docker push "$CI_REGISTRY_IMAGE:$IMAGE_TAG"
    - printf 'DEPLOY_IMAGE=%s\n' "$CI_REGISTRY_IMAGE:$IMAGE_TAG" > image.env
  artifacts: { reports: { dotenv: image.env } }   # kolejne joby nie wybiorą innej wartości

deploy-production:
  when: manual                                    # wąska akceptacja, nie zamiast review
  script:
    - kubectl -n storefront set image deployment/web web="$DEPLOY_IMAGE"
    - kubectl -n storefront rollout status deployment/web --timeout=180s
```

Nie buduj ponownie tego samego commita dla produkcji — obraz bazowy / mirror paczek / czas budowania mogą zmienić bajty. CI dostaje **tożsamość** (scoped ServiceAccount/RBAC), nie poświadczenia administratora.

4. Migracja = Job z nazwą, przed podami web

```
kind: Job
metadata: { name: migrate-3f18d7c }              # SHA w nazwie — zakończony Job nie użyty ponownie
spec:
  backoffLimit: 1
  ttlSecondsAfterFinished: 86400
  template:
    spec:
      restartPolicy: Never
      containers:
        - name: migrate
          image: registry.example.com/acme/store:3f18d7c   # ten sam niemutowalny obraz
          command: ["php","artisan","migrate","--force","--no-interaction"]
          envFrom: [{ secretRef: { name: storefront-runtime } }]
```

```
job="migrate-${CI_COMMIT_SHORT_SHA}"
kubectl -n storefront delete job "$job" --ignore-not-found
sed -e "s|IMAGE_PLACEHOLDER|$DEPLOY_IMAGE|g" -e "s|JOB_PLACEHOLDER|$job|g" k8s/migrate-job.yaml |
kubectl apply -f -
```

```
kubectl -n storefront wait --for=condition=complete "job/$job" --timeout=180s \
  || { kubectl -n storefront logs "job/$job" --all-containers=true; exit 1; }
```

> **EXPAND-CONTRACT** `migrate --force` potwierdza nieinteraktywny run, nie czyni zmian odwracalnymi. Najpierw nullable column / nowa tabela → kod działający z oboma kształtami → backfill async → dopiero później usuń starą ścieżkę. Nie ponawiaj ślepo nieudanej migracji — sprawdź, czy zapisała częściowe dane.

5. Weryfikacja i rollback

```
kubectl -n storefront rollout status deployment/web --timeout=180s
kubectl -n storefront get pods -l app=storefront-web \
  -o custom-
columns=NAME:.metadata.name,READY:.status.containerStatuses[0].ready,IMAGE:.spec.containers[0].image
curl --fail --show-error --retry 3 https://shop.example.com/up      # HTTP 200 ze starego poda ≠
udany deploy

# rollback szablonu poda — NIE odwraca Job bazy
kubectl -n storefront rollout undo deployment/web
```

Wycofaj Deployment **dopiero po** potwierdzeniu, że migracja pozostaje kompatybilna z poprzednim obrazem. Schemat niekompatybilny wstecz → świadomy plan odzyskania, nie optymistyczne polecenie k8s.

6. Operacje dnia drugiego

```
# sygnał, nie dashboard — odróżnij zły release od problemu węzła PRZED usuwaniem podów
kubectl -n storefront get events --sort-by=.lastTimestamp | tail -n 30
kubectl -n storefront top pods ; kubectl top nodes
kubectl -n storefront logs "$pod" --previous --all-containers=true --tail=200   # CrashLoopBackOff

# backup + DOWÓD odtworzenia (zero z polecenia nie dowodzi backupu)
mysqldump --single-transaction --routines --hex-blob --host="$DB_HOST" "$DB_DATABASE" | gzip >
"storefront-$(date +%F-%H%M).sql.gz"
# regularnie: odtwórz kopię do izolowanej bazy, uruchom zapytanie integralności, zapisz czas
(RPO/RTO)
```

Dla workerów kolejki sprawdzaj głębokość i błędy *kolejki*, nie readiness poda web. Przekaż `RELEASE_SHA` jako niesekretną zmienną → logi strukturalne łączą „klienci widzą 500" z rewizją. Uploady testuj osobno (lifecycle object storage, wersjonowanie bucketu).

Usługi wspierające (część 4) — krótko

MinIO / GlitchTip / Uptime Kuma / dashboardy: każdy dodaje storage, aktualizacje, poświadczenia, własność alertów. Dodawaj **tylko** pod konkretną potrzebę operacyjną, we własnym namespace z requestami/limitami, z przetestowanym odtworzeniem, endpointy administracyjne prywatne (VPN / proxy tożsamości — nie sama allowlista IP). DSN/klucze S3 nigdy w manifeście w Gicie.

Kiedy to NIE jest właściwe pierwsze wdrożenie

Prosta aplikacja, którą jeden kontener + reverse proxy obsłuży taniej i czytelniej. k3s zarabia, gdy potrzebujesz realnie rolloutów, izolacji workloadów i deklaratywnej infrastruktury — a zespół uniesie platformę (backupy, upgrade'y, RBAC).