Zarządzanie Systemami Rozproszonymi
Laboratoria z ArgoCD i GitOps

mgr inż. Jakub Woźniak

1 Wprowadzenie do GitOps i ArgoCD

1.1 Czym jest GitOps?

GitOps to podejście do zarządzania infrastrukturą i aplikacjami w Kubernetes z wykorzystaniem repozytorium Git jako jedynego źródła prawdy (*single source of truth*). W GitOps wszystkie zmiany infrastruktury i aplikacji są definiowane jako kod, przechowywane w repozytorium Git i automatycznie synchronizowane z klastrem Kubernetes.
Kluczowe założenia GitOps:

- **Deklaratywność:** Stan infrastruktury i aplikacji jest definiowany w sposób deklaratywny w plikach konfiguracyjnych YAML.
- **Automatyczna synchronizacja:** Narzędzia GitOps automatycznie synchronizują stan klastra Kubernetes z repozytorium Git.
- **Zarządzanie przez pull requesty:** Zmiany są wprowadzane poprzez pull requesty, co zapewnia pełną historię zmian i mechanizm zatwierdzania (*code review*).
- **Obserwowalność:** GitOps zapewnia pełną widoczność stanu klastra i aplikacji dzięki porównaniu stanu bieżącego z wersją w repozytorium.

1.2 Dlaczego GitOps?

Korzyści z GitOps:

- **Spójność:** Repozytorium Git jest jedynym źródłem prawdy, co zapewnia spójność stanu środowiska.
- **Automatyzacja:** Proces wdrożenia i synchronizacji jest w pełni zautomatyzowany, co redukuje ryzyko błędów ludzkich.
- **Wersjonowanie:** Wszystkie zmiany są zapisywane w historii Git, co umożliwia łatwe śledzenie i przywracanie poprzednich wersji.

- **Bezpieczeństwo:** Zmiany są wprowadzane w sposób kontrolowany poprzez pull requesty, co pozwala na ich przegląd i zatwierdzenie przed wdrożeniem.
- **Obserwowalność:** Narzędzia GitOps, takie jak ArgoCD, umożliwiają monitorowanie stanu klastra i szybkie wykrywanie odchyleń od pożądanego stanu.

1.3 ArgoCD: Wprowadzenie

ArgoCD to jedno z najpopularniejszych narzędzi GitOps, stworzone specjalnie do zarządzania aplikacjami Kubernetes. Umożliwia automatyczną synchronizację stanu klastra z konfiguracją przechowywaną w repozytorium Git.
Najważniejsze funkcje ArgoCD:

- **Automatyczna synchronizacja:** ArgoCD monitoruje repozytorium Git i automatycznie synchronizuje stan klastra z deklaratywną konfiguracją.
- **Obsługa wielu klastrów:** ArgoCD pozwala zarządzać wieloma klastrami Kubernetes z jednego interfejsu.
- **Wsparcie dla Helm i Kustomize:** Oprócz plików YAML, ArgoCD wspiera konfiguracje Helm Charts oraz Kustomize.
- **Historia synchronizacji:** Rejestruje wszystkie operacje synchronizacji i zmiany w stanie klastra.
- **Przyjazny interfejs:** Graficzny interfejs użytkownika (GUI) umożliwia łatwe zarządzanie aplikacjami i obserwowanie ich stanu.

1.4 Architektura ArgoCD

ArgoCD działa w klastrze Kubernetes jako zestaw kontrolerów, które stale monitorują repozytorium Git i synchronizują stan klastra z jego zawartością.
Główne komponenty ArgoCD:

- **API Server:** Serwer API zarządzający komunikacją z interfejsem użytkownika i CLI.
- **Repository Server:** Obsługuje połączenia z repozytorium Git i pobiera konfiguracje aplikacji.
- **Application Controller:** Monitoruje stan aplikacji w klastrze i synchronizuje go z repozytorium Git.
- **User Interface (UI):** Przyjazny interfejs graficzny do zarządzania aplikacjami i klastrami.

ArgoCD umożliwia szybkie i niezawodne wdrożenia aplikacji oraz ich automatyczne synchronizowanie, co czyni je idealnym narzędziem do realizacji filozofii GitOps.

2 Przygotowanie środowiska

W tej sekcji studenci przygotują środowisko do pracy z GitOps za pomocą ArgoCD. Proces obejmuje instalację ArgoCD na klastrze Minikube, konfigurację narzędzi CLI oraz utworzenie repozytorium Git w GitLab.

2.1 Wymagane narzędzia

Do wykonania ćwiczeń wymagane są następujące narzędzia:

- **Minikube:** Klaster Kubernetes działający na lokalnej maszynie. Instalację Minikube można przeprowadzić zgodnie z instrukcjami dostępnymi na stronie `https://minikube.sigs.k8s.io/docs/start/`.
- **kubectl:** Narzędzie CLI do zarządzania klastrem Kubernetes. Można je pobrać ze strony `https://kubernetes.io/docs/tasks/tools/install-kubectl/`.
- **ArgoCD CLI:** Narzędzie do interakcji z ArgoCD z poziomu terminala. Plik binarny możesz pobrać stąd: `https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64`.
- **Git:** Do klonowania i zarządzania repozytoriami Git. Większość systemów Linux ma preinstalowane Git, ale można je zainstalować przez `apt`, `yum` lub `brew`.

2.2 Dlaczego GitLab?

W GitOps repozytorium Git pełni rolę jedynego źródła prawdy (*single source of truth*). GitLab jest jednym z najpopularniejszych narzędzi tego typu, zapewniającym:

- Wersjonowanie plików konfiguracyjnych aplikacji Kubernetes.
- Łatwy mechanizm przeglądu zmian (*code review*) przez pull requesty.
- Publiczne repozytoria bez konieczności konfiguracji serwera Git.

GitLab umożliwia studentom szybki dostęp do gotowych repozytoriów oraz łatwe zarządzanie konfiguracjami aplikacji Kubernetes.

2.3 Instalacja ArgoCD na Minikube

ArgoCD musi zostać zainstalowane jako zestaw komponentów w klastrze Kubernetes. Proces instalacji składa się z następujących kroków:

1. **Utwórz namespace dla ArgoCD:**

```
kubectl create namespace argocd
```

2. **Zainstaluj ArgoCD za pomocą oficjalnych manifestów:** Oficjalny manifest: `https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml`

```
kubectl apply -n argocd -f <adres pliku>
```

Powyższe polecenie pobiera oficjalne manifesty YAML i instaluje wszystkie komponenty ArgoCD w namespace `argocd`.

3. **Sprawdź status komponentów ArgoCD:**

```
kubectl get pods -n argocd
```

Upewnij się, że wszystkie Pody są w stanie `Running`.

4. **Uzyskaj hasło administratora ArgoCD:** Domyślne hasło znajduje się w `argocd-initial-admin-secret`. Odczytaj je poleceniem:

```
kubectl -n argocd get secret
    argocd-initial-admin-secret \
  -o jsonpath="{.data.password}" | base64 -d;
     echo
```

2.4 Utworzenie repozytorium Git

Repozytorium Git będzie przechowywać deklaratywną konfigurację aplikacji Kubernetes.

1. **Zaloguj się na platformie GitLab i utwórz nowe repozytorium:** Przejdź na `https://gitlab.com` i utwórz nowe repozytorium o nazwie `argo-gitops-demo`. Ustaw widoczność na `publiczne`.

2. **Sklonuj repozytorium na swój komputer lokalny:**

```
git clone
    https://gitlab.com/<twoja-nazwa-żuytkownika>/argo-gitops-demo.git
```

3. **Utwórz foldery i pliki dla aplikacji Kubernetes:**

```
mkdir -p apps/nginx
touch apps/nginx/deployment.yaml
touch apps/nginx/service.yaml
```

4. **Dodaj pliki YAML dla aplikacji: Deployment aplikacji NGINX:**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
```

```
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

Service dla aplikacji NGINX:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: NodePort
```

5. **Zatwierdź zmiany i wypchnij je do repozytorium:**

```
git add .
git commit -m "Add initial Kubernetes manifests
    for NGINX app"
git push
```

Ważne: Nie wykonuj `kubectl apply` na tych plikach!
Po zakończeniu tego kroku środowisko jest gotowe do użycia z ArgoCD i dalszych ćwiczeń związanych z GitOps.

3 Automatyzacja wdrożeń aplikacji Kubernetes

W tej sekcji studenci skonfigurują ArgoCD do automatycznego zarządzania wdrożeniami aplikacji w klastrze Kubernetes. Wykorzystamy wcześniej utwor-

zone repozytorium Git oraz aplikację NGINX.

3.1 Podłączenie ArgoCD do repozytorium Git

Aby ArgoCD mogło monitorować zmiany w repozytorium i synchronizować stan aplikacji, należy podłączyć repozytorium do ArgoCD.

1. **Dodaj repozytorium Git w ArgoCD:**
   - Zaloguj się do interfejsu ArgoCD pod adresem `https://localhost:8080`.
   - Przejdź do zakładki `Settings > Repositories`.
   - Kliknij `Connect Repo` i wprowadź:
     - **URL:** `https://gitlab.com/<twoja-nazwa-użytkownika>/argo-gitops-demo.git`
     - **Typ:** Publiczne repozytorium.
2. **Zweryfikuj połączenie:** Repozytorium powinno być widoczne na liście podłączonych repozytoriów.

3.2 Tworzenie aplikacji w ArgoCD

Teraz skonfigurujemy aplikację w ArgoCD, która będzie monitorować stan aplikacji NGINX w repozytorium Git.

1. **Utwórz aplikację w ArgoCD:**
   - Przejdź do zakładki `Applications` i kliknij `Create Application`.
   - Wypełnij formularz:
     - **Application Name:** `nginx-app`
     - **Project:** `default`
     - **Sync Policy:** `Manual` (na początek).
     - **Repository URL:** `https://gitlab.com/<twoja-nazwa-użytkownika>/argo-gitops-demo.git`
     - **Revision:** `HEAD`
     - **Path:** `apps/nginx`
     - **Cluster:** `https://kubernetes.default.svc`
     - **Namespace:** `default`
2. **Zatwierdź konfigurację:** Kliknij przycisk `Create`.

3.3 Synchronizacja aplikacji

Synchronizacja aplikacji w ArgoCD polega na porównaniu bieżącego stanu klastra z konfiguracją w repozytorium Git i zastosowaniu brakujących zmian.

1. **Ręczna synchronizacja:**
   - W interfejsie aplikacji kliknij przycisk `Sync`.
   - Wybierz opcję `Prune resources` i kliknij `Synchronize`.
2. **Obserwacja wdrożenia:** Sprawdź, czy aplikacja została wdrożona:

   ```
   kubectl get pods
   kubectl get svc
   ```

   Sprawdź dostępność usługi NGINX za pomocą adresu `NodePort` wyświetlonego w wyniku komendy.
3. **Automatyczna synchronizacja:**
   - Zmień **Sync Policy** aplikacji na `Automatic`.
   - Wprowadź zmiany w repozytorium Git, np. zmień liczbę replik w pliku `deployment.yaml`:

     ```
     spec:
       replicas: 3
     ```

   - Wypchnij zmiany do repozytorium:

     ```
     git add .
     git commit -m "Update replicas to 3"
     git push
     ```

   - Obserwuj automatyczne zastosowanie zmian w ArgoCD i klastrze Kubernetes.

3.4 Zadania samodzielne

Po skonfigurowaniu podstawowej aplikacji studenci mogą wykonać poniższe zadania samodzielnie:

- **Dodanie drugiej aplikacji:** Utwórz nową aplikację, np. Redis, w tym samym repozytorium i dodaj ją do ArgoCD.
- **Testowanie zmian:** Zmień wersję obrazu Dockera (np. `nginx:alpine`) i zaobserwuj proces wdrożenia.
- **Symulacja awarii:** Usuń Pod aplikacji NGINX i sprawdź, jak ArgoCD synchronizuje zmiany i przywraca stan aplikacji.

Dzięki automatyzacji wdrożeń z ArgoCD, studenci mogą zobaczyć w praktyce, jak zmiany w repozytorium Git wpływają na stan klastra Kubernetes w czasie rzeczywistym.

4 Zadania samodzielne

W tej sekcji studenci wykonają zadania samodzielne, polegające na wykorzystaniu zdobytej wiedzy do wdrożenia aplikacji Redmine za pomocą ArgoCD.

4.1 Wdrożenie aplikacji Redmine

Redmine to popularne narzędzie do zarządzania projektami, które wymaga bazy danych jako backendu. Zadanie polega na przygotowaniu konfiguracji aplikacji i jej wdrożeniu za pomocą GitOps.

4.1.1 Wymagania

- Użyj publicznego obrazu `redmine` z Docker Hub (`docker.io/library/redmine`).
- Wykorzystaj PostgreSQL jako bazę danych. Użyj obrazu `postgres` z Docker Hub (`docker.io/library/postgres`).
- Skonfiguruj aplikację tak, aby była dostępna przez Service typu `NodePort`.
- Utwórz PersistentVolumeClaim (PVC) dla danych PostgreSQL, aby zapewnić trwałość danych.

4.1.2 Kroki do wykonania

1. **Przygotuj repozytorium Git:**
   - Utwórz nowy folder `apps/redmine` w repozytorium Git.
   - Dodaj pliki konfiguracyjne YAML dla Deployment, Service oraz PersistentVolumeClaim aplikacji Redmine i PostgreSQL.
2. **Skonfiguruj aplikację PostgreSQL:**
   - Utwórz Deployment dla PostgreSQL, definiując:
     - Nazwę użytkownika (`POSTGRES_USER`).
     - Hasło (`POSTGRES_PASSWORD`).
     - Bazę danych (`POSTGRES_DB`).
   - Powiąż Deployment z PVC w celu przechowywania danych.
3. **Skonfiguruj aplikację Redmine:**
   - Utwórz Deployment dla Redmine, ustawiając zmienne środowiskowe umożliwiające połączenie z bazą PostgreSQL (`DB_HOST`, `DB_USER`, `DB_PASSWORD`, `DB_NAME`).
   - Powiąż aplikację z Service typu `NodePort`, aby była dostępna w klastrze.
4. **Dodaj aplikację do ArgoCD:**

- W interfejsie ArgoCD utwórz nową aplikację:
  - Repozytorium: adres Twojego repozytorium Git.
  - Path: `apps/redmine`.
  - Namespace: `default`.

5. **Sprawdź wdrożenie:**
   - Upewnij się, że aplikacja Redmine i baza PostgreSQL są uruchomione.
   - Zaloguj się do aplikacji Redmine przez przeglądarkę, używając adresu uzyskanego z polecenia `kubectl get svc`.

4.1.3 Kryteria zaliczenia

- Redmine jest dostępny przez Service typu `NodePort`.
- Dane PostgreSQL są przechowywane na PersistentVolume.
- Aplikacja automatycznie synchronizuje się z repozytorium Git po wprowadzeniu zmian.

4.2 Dodatkowe zadanie (opcjonalne)

- Zmodyfikuj Deployment Redmine, aby dodać zmienną środowiskową `REDMINE_LANG` ustawioną na `pl`.
- Obserwuj proces synchronizacji i sprawdź, czy zmiana została poprawnie wdrożona.

5 Podsumowanie i wnioski

Laboratoria z GitOps z wykorzystaniem ArgoCD pozwoliły studentom na praktyczne zapoznanie się z nowoczesnym podejściem do zarządzania aplikacjami i infrastrukturą w Kubernetes. W trakcie zajęć studenci mieli okazję:

- Zrozumieć podstawowe koncepcje GitOps, w tym rolę repozytorium Git jako jedynego źródła prawdy (*single source of truth*).
- Poznać architekturę i funkcje ArgoCD jako narzędzia wspierającego automatyczne wdrożenia.
- Samodzielnie skonfigurować i wdrożyć aplikacje Kubernetes, takie jak Redmine, przy użyciu podejścia GitOps.
- Zobaczyć w praktyce, jak zmiany w repozytorium Git są automatycznie synchronizowane z klastrem Kubernetes.
- Zrozumieć kluczowe korzyści GitOps, takie jak automatyzacja, spójność, audytowalność i łatwość przywracania zmian.

5.1 Korzyści z GitOps

Podejście GitOps oferuje liczne zalety w zarządzaniu infrastrukturą i aplikacjami:

- **Automatyzacja:** Eliminacja ręcznych procesów wdrożeniowych zmniejsza ryzyko błędów i przyspiesza dostarczanie zmian.
- **Przejrzystość:** Repozytorium Git pozwala na pełne śledzenie historii zmian oraz łatwe identyfikowanie i rozwiązywanie problemów.
- **Zwiększona wydajność zespołu:** Wprowadzenie zmian za pomocą pull requestów umożliwia zespołom współpracę w kontrolowany sposób.
- **Szybkie przywracanie środowiska:** Możliwość łatwego odtworzenia poprzedniego stanu systemu na podstawie historii Git.
- **Skalowalność:** Wsparcie dla dużych i dynamicznych środowisk z wieloma aplikacjami i klastrami Kubernetes.

5.2 Wyzwania i potencjalne problemy

Choć GitOps i ArgoCD oferują znaczące korzyści, warto pamiętać o potencjalnych wyzwaniach:

- **Złożoność konfiguracji:** Początkowe wdrożenie i konfiguracja GitOps może być czasochłonne i wymagać odpowiednich umiejętności.
- **Zarządzanie konfliktami:** W przypadku zmian wprowadzanych bezpośrednio w klastrze może dojść do konfliktu z repozytorium Git.
- **Bezpieczeństwo:** Publiczne repozytoria wymagają szczególnej uwagi w kwestii przechowywania danych wrażliwych, takich jak klucze i hasła.
- **Wydajność przy dużych klastrach:** Synchronizacja dużych ilości danych lub złożonych aplikacji może wymagać optymalizacji.

5.3 Praktyczne zastosowania

GitOps z ArgoCD znajduje zastosowanie w wielu scenariuszach:

- Zarządzanie mikroserwisami w dużych środowiskach Kubernetes.
- Automatyzacja procesów wdrożeniowych i integracja z pipeline'ami CI/CD.
- Utrzymanie spójności między środowiskami (np. dev, staging, prod).
- Ułatwienie migracji aplikacji między różnymi platformami chmurowymi.

5.4 Wnioski końcowe

GitOps z ArgoCD to podejście, które upraszcza zarządzanie aplikacjami w Kubernetes poprzez pełną automatyzację i integrację z Git. Studenci zyskali praktyczną wiedzę na temat:

- Tworzenia i wdrażania aplikacji w modelu GitOps.
- Konfigurowania i zarządzania ArgoCD.
- Automatycznego synchronizowania stanu aplikacji z repozytorium Git.

Podejście GitOps jest kluczowym elementem nowoczesnych procesów DevOps, szczególnie w środowiskach rozproszonych, i stanowi ważny krok w kierunku większej automatyzacji oraz niezawodności zarządzania aplikacjami i infrastrukturą.