Argo

Gabriel Kaźmierczak
Dariusz Piwowarski
Wojciech Przybytek
Przemysław Roman

2024

Spis treści

1 Wstęp

Celem projektu jest zaprezentowanie działania Argo, czyli zestawu narzędzi open source stworzonych do zarządzania zadaniami w kontenerach Kubernetes. Narzędzia te są szeroko stosowane do automatyzacji wdrażania aplikacji, zarządzania ich cyklem życia oraz koordynacji złożonych przepływów pracy. W ramach projektu skupimy się na dwóch kluczowych komponentach Argo: Argo CD, które umożliwia ciągłą dostawę aplikacji przy użyciu podejścia GitOps, oraz Argo Workflows, które pozwala na definiowanie i zarządzanie złożonymi przepływami pracy w środowisku Kubernetes.

Podział pracy w zespole

Ponieważ w ramach tego projektu omawiamy dwa zastosowania Argo zdecydowaliśmy na podział zespołu na dwa podzespoły, każdy omawiający jedno z zastosowań. Docelowo praca obu podzespołów ma zostać połączona, tworząc integralną całość.

Podzespół 1 - Argo CD

Osoby: Dariusz Piwowarski, Przemysław Roman
Cele: Stworzenie klastra Kubernetes na AWS oraz uruchomienie w nim Argo CD i zintegrowanie go z repozytorium aplikacji

Podzespół 2 - Argo Workflows

Osoby: Gabriel Kaźmierczak, Wojciech Przybytek
Cele: Zapoznanie się z API Argo Workflows, napisanie aplikacji wykorzystującej to API oraz udostępniającej własny interfejs

2 Podstawy teoretyczne

Kubernetes[1] to oprogramowanie open source służące do automatyzacji wdrażania, skalowania i zarządzania aplikacjami kontenerowymi. Zostało zaprojektowane przez Google, a obecnie jest utrzymywane przez Cloud Native Computing Foundation. Kubernetes dostarcza platformę do uruchamiania aplikacji w kontenerach na dużą skalę, zarządzając infrastrukturą i zapewniając funkcje, takie jak self-healing (automatyczne restartowanie kontenerów), odkrywanie usług i load balancing, przechowywanie danych, skalowanie i wiele innych.

Argo[2] jest projektem open source, który dostarcza narzędzia do tworzenia, uruchamiania i zarządzania pracami w kontenerach Kubernetes. Składa się z podprojektów, w których skład wchodzą Argo Workflows, Argo CD, Argo Rollouts oraz Argo Events.

Argo Workflows[3] to silnik do tworzenia i uruchamiania prac w kontenerach. Pozwala na definiowanie złożonych przepływów pracy (workflow), które mogą obejmować wiele zadań uruchamianych w różnych kontenerach, z możliwością kontroli przepływu, takiej jak równoległe lub sekwencyjne uruchamianie zadań, decyzje warunkowe, pętle i inne. Argo Workflows jest szczególnie przydatne w środowiskach takich jak przetwarzanie danych, czy uczenie maszynowe, gdzie istnieje potrzeba koordynacji wielu zadań.

Argo CD[4] to narzędzie do ciągłej dostawy (Continuous Delivery, CD), które implementuje podejście GitOps do zarządzania infrastrukturą. W modelu GitOps, żądane stany systemów są zapisywane w repozytorium Git, a narzędzia automatycznie aktualizują systemy, aby dopasować je do stanu zapisanego w Git. Argo CD monitoruje repozytorium i automatycznie wdraża zmiany na środowiska Kubernetes, gdy stan w Git się zmienia.

Oba omawiane podprojekty Argo są zintegrowane z Kubernetes i wykorzystują jego funkcje, takie jak API, schematy autoryzacji i mechanizmy skalowania. Dzięki temu są one naturalnym rozszerzeniem ekosystemu Kubernetes i mogą być łatwo zintegrowane z innymi narzędziami w tym ekosystemie.

3 Stos technologiczny

Amazon Elastic Kubernetes Service[5] zapewni infrastrukturę na której będzie działał klaster Kubernetes

GitHub[6] umożliwi przechowywanie repozytorium Git z kodem źródłowym aplikacji

Python 3[7] jako język implementacji aplikacji

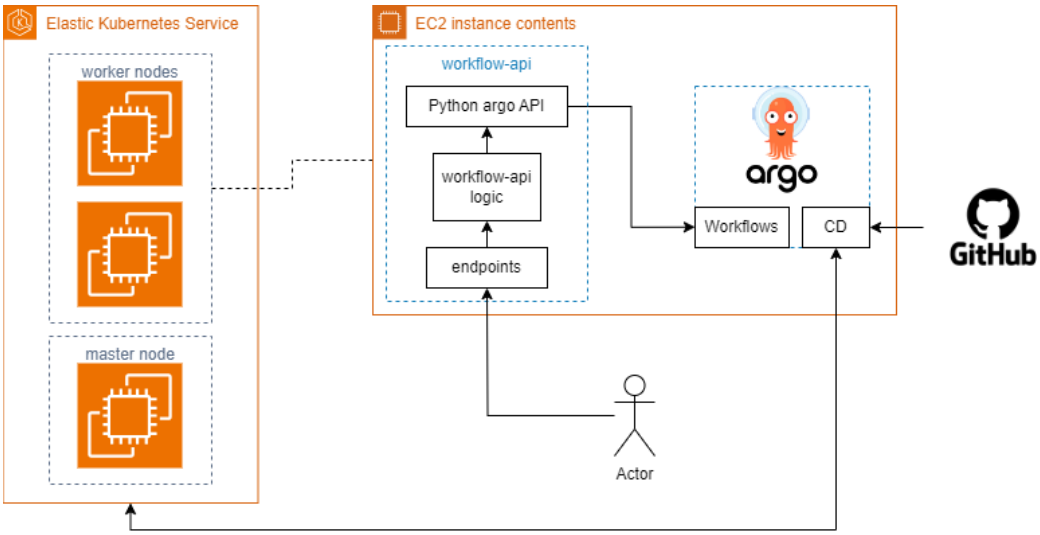
Hera[8] to biblioteka, która zostanie wykorzystane do komunikacji z API Argo

4 Koncepcja studium przypadku

Demo będzie zawierać aplikację działającą w klastrze Kubernetes, która zaprezentuje możliwości narzędzi Argo Workflows oraz Argo CD. Aplikacja będzie służyć jako punkt wejścia do uruchamiania zadań w Argo Workflows. Użytkownik będzie mógł wysyłać żądania HTTP do aplikacji uruchamiające odpowiednie workflowy za pomocą API do Argo Workflows. Każda zmiana w repozytorium przechowującym kod aplikacji będzie automatycznie wykrywana przez Argo CD, które następnie zaktualizuje działającą aplikację na klastrze.

5 Architektura rozwiązania

Architektura rozwiązania dla rozważanego studium przypadku widoczna jest na Rys. 1. Zarówno narzędzia Argo jak i nasza aplikacja (workflow-api) uruchomione są na klastrze Kubernetesa EKS. Argo CD integruje się z Githubem, z którego pobiera zmiany oraz z klastrem w celu aplikowania ich. Workflow-api udostępnia endpointy, na które użytkownicy mogą zlecać obliczenia.



Rysunek 1: Architektura rozwiązania

6 Instalacja narzędzi

Do zrealizowania projektu niezbędne jest zainstalowanie kilku narzędzi. W zależności od używanego systemu operacyjnego, proces instalacji może się różnić. Poniżej znajdują się linki do instrukcji instalacji dla poszczególnych narzędzi:

- **Terraform**[9]
- **kubectl**[10]
- **AWS CLI**[11]
- **argocd CLI**

7 Reprodukcja środowiska

7.1 Klaster Kubernetes

Bazowaliśmy na Provision an EKS cluster (AWS) oraz Learn Terraform - Provision an EKS Cluster.

1. `git clone https://github.com/SUU-Argo/infra`
2. Umieść konfigurację i dane logowania do AWS w `~/.aws/`
3. W związku z tym, że używamy AWS Academy Learner Lab konieczne jest wykorzystanie istniejącej roli LabRole i podanie jej arn jako zmiennej w terraformie. Pobranie arn:
   `aws iam get-role --role-name LabRole | grep Arn`
4. `terraform -chdir=infra/terraform init`
5. `terraform -chdir=infra/terraform apply -var="aws_iam_role=<your LabRole arn>"`
6. Sprawdź nazwę klastra: `aws eks list-clusters`
7. Wygeneruj kubeconfig: `aws eks update-kubeconfig --name <your cluster name>`
8. Zweryfikuj połączenie: `kubectl get nodes`

7.2 Argo CD

Przygotowane w oparciu o Argo - Getting Started

1. Utwórz namespace: `kubectl create namespace argocd`
2. Zainstaluj ArgoCD:
   ```
   kubectl -n argocd apply -f https://raw.githubusercontent.com/argoproj/argo-cd/
   stable/manifests/install.yaml
   ```
3. Przekieruj Argo na localhost (zajmuje terminal):
   `kubectl -n argocd port-forward svc/argocd-server 8080:443`
4. Uzyskaj dostęp do Argo za pomocą przeglądarki internetowej: https://localhost:8080
5. Uzyskaj hasło: `argocd admin initial-password -n argocd`
6. Zaloguj się: `argocd login localhost:8080 --username admin --password <password>`

7.3 Argo Workflows

1. Stwórz namespace: `kubectl create namespace argo`
2. Zainstaluj Argo Workflows:
   ```
   kubectl -n argo apply -f "https://github.com/argoproj/argo-
   workflows/releases/download/v3.5.7/quick-start-minimal.yaml"
   ```
3. Zaaplikuj łatkę do argo-server (repozytorium infra):
   ```
   kubectl -n argo patch deployment argo-server --type JSON
   --patch-file k8s/argo-server-patch.yaml
   ```
4. Przekieruj argo-server na localhost (zajmuje terminal):
   `kubectl -n argo port-forward svc/argo-server 2746:2746`
5. Uzyskaj dostęp do Argo za pomocą przeglądarki internetowej: https://localhost:2746

7.4 workflow-api

1. Ustaw namespace: `kubectl config set-context --current --namespace=argocd`
2. Utwórz aplikację z repozytorium:
   ```
   argocd app create workflow-api \
   --repo https://github.com/SUU-Argo/workflow-api.git \
   --path deploy \
   --dest-server https://kubernetes.default.svc \
   --dest-namespace argo
   ```
3. Wyświetl szczegóły instancji aplikacji: `argocd app get workflow-api` , początkowy status to `OutOfSync`
4. Włącz automatyczną synchronizację aplikacji z repozytorium:
   `argocd app set workflow-api --sync-policy automated`
5. Przekieruj aplikację na localhost (zajmuje terminal):
   `kubectl -n argo port-forward svc/workflow-api 8081:8080`
6. Zweryfikuj dostępność za pomocą przeglądarki internetowej: http://localhost:8081

8 Reprodukcja demo

9 Podsumowanie

Źródła

[1] *Kubernetes*. URL: https://kubernetes.io/ (visited on 06/05/2024).

[2] *Argo*. URL: https://argoproj.github.io/ (visited on 06/05/2024).

[3] *Argo Workflows*. URL: https://argoproj.github.io/workflows/ (visited on 06/05/2024).

[4] *Argo CD*. URL: https://argoproj.github.io/cd (visited on 06/05/2024).

[5] *AWS EKS*. URL: https://aws.amazon.com/eks/ (visited on 06/05/2024).

[6] *Github*. URL: https://github.com/ (visited on 06/05/2024).

[7] *Python*. URL: https://www.python.org// (visited on 06/05/2024).

[8] *Hera*. URL: https://github.com/argoproj-labs/herad (visited on 06/05/2024).

[9] *Terraform*. URL: https://www.terraform.io/ (visited on 06/05/2024).

[10] *kubectl*. URL: https://kubernetes.io/docs/reference/kubectl/ (visited on 06/05/2024).

[11] *AWS CLI*. URL: https://aws.amazon.com/cli/ (visited on 06/05/2024).

[12] *Terraform install*. URL: https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli (visited on 06/05/2024).