RABIT : Jurnal Teknologi dan Sistem Informasi Univrab
Volume 9 No. 1 | Januari 2024 : Hal : 154-161

ISSN CETAK : 2477-2062
ISSN ONLINE : 2502-891X

IMPLEMENTATION OF GITOPS IN CONTAINERIZED INFRASTRUCTURE

[1)] Rayhan Gusty Alif, [2)] Lulu Chaerani Munggaran
[1,2)]Graduate Program of Information System Management, Gunadarma University
[1,2)] Jl. Salemba No. 53 – Jakarta – Indonesia
E-mail : *rayhanga@student.gunadarma.ac.id*, *lulu@staff.gunadarma.ac.id*

ABSTRACT
IT Infrastructure is one of the core components of a business' scalability and reliability, thus having an efficient way on managing the IT Infrastructure would be one of the core decisions for a business. IT Infrastructure itself has evolved throughout the years, with the rise of virtualization technology, containerization became more relevant as ever, one such example is Containerization Infrastructure, an IT Infrastructure that uses Containerization as its backbone. With the push of the technology, a new way of managing Containerization Infrastructure efficiently is needed. There are multiple researches regarding the implementation of GitOps already, but none of them explained the connection between GitOps and Containerized Infrastructure, this paper is intended to discuss the connection by implementing GitOps in a Containerized infrastructure. This resulted in a quite steep learning curve and preparation time, but in the end all the changes and deployment of the application would be done automatically, this resulted in maximized focus on the development of the application rather than reflecting the changes later on. Other than that, GitOps itself is not limited to Containerized Infrastructure, although since GitOps is designed with virtualization in mind, theoretically, the efficiency would be reduced if it's implemented in other kind of IT Infrastructure.

Keyword: Containerized Infrastructure, GitOps, Implementation, IT Infrastructure Management.

INTRODUCTION

There are multiple researches regarding the implementation of GitOps already [1], [2]. Nevertheless, there are still no researches that explained the connection between Containerized Infrastructure and IT Infrastructure Management. This research is intended to discuss the connection between IT Infrastructure Management and the implementation of GitOps in a Containerized Infrastructure, this is done by implementing GitOps in a Containerized Infrastructure. IT Infrastructure itself is one of the core components of a business' scalability and reliability [3]. With Virtual Infrastructures, a change has been brought in on how we manage a business's IT Infrastructure, these changes increase the efficiency in managing a deployed software/application, which led to a much better virtualization technology called Containerization. This Containerization method helps us make our application more portable, scalable, and isolated [4]. However, due to the rise of Containerization technology and the popularity of microservices architecture, managing Containerized applications increases the management complexity, therefore using traditional methods of infrastructure management would involve manual operations to scale and even to deploy it for the first time [5]. Using Traditional methods is more prone to human errors, time-consuming, and reduced scalability, not only that, but the rise of Cloud Computing also increases the need to reduce cost and full utilization of IT resources, and shorten software deployment time [6]. Introduce GitOps, a modern approach to managing IT Infrastructure, and a potential solution to address these challenges. GitOps combines the principle of Infrastructure as Code (IaC) and Continuous Delivery/Deployment (CD) practices, it uses Git repositories as the main source of truth for infrastructure configurations [7], [8], [9], [10], [11], [12]. This paper will be covering the connection between GitOps and IT Infrastructure Management to see whether it's a viable IT Infrastructure Management method for a Containerized Infrastructure.

DOI : https://doi.org/10.36341/rabit.v9i1.3787

METHOD

This research will explore the connection between GitOps and IT Infrastructure Management by implementing GitOps in a Kubernetes Cluster (Containerized Infrastructure), but will not cover all the available variations. By implementing GitOps using ArgoCD, Gitlab CI/CD Pipeline, and Rancher Desktop (K3s) on a local home lab with an internet connection to connect with Gitlab CI/CD repository.

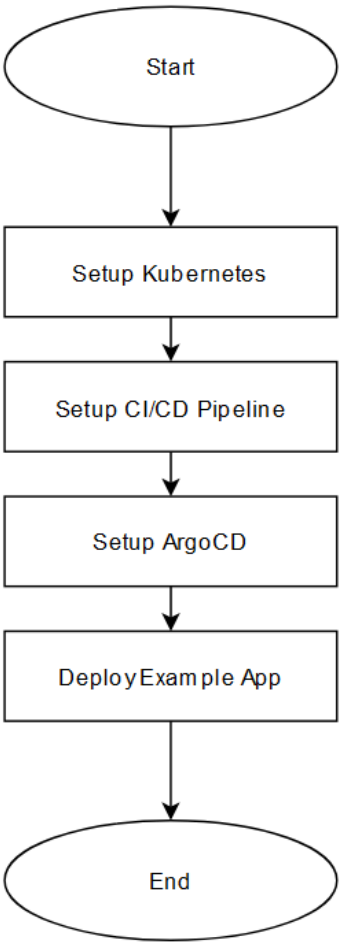


Figure 1. Research Framework

The whole research framework as shown in Figure 1 is a generic overview of GitOps implementation, since the research itself is to find a connection between IT Infrastructure Management with GitOps, this can be done by implementing GitOps in a local home lab. Each tool used in each stage of the implementation as shown in Figure 1 are interchangeable with any other tools that has the same purpose and either the steps of the implementation or the end result won't differ that much.

To **Setup a Kubernetes Cluster** on a local home lab we can use Rancher Desktop to provision Kubernetes Cluster, which in this case it's a K3s Cluster. To **Setup CI/CD Pipeline**, we can use Gitlab CI/CD, to do this we can create a YML file defining the build and deploy stages of the pipeline. The build stage will build all the docker images and update the Kubernetes manifests to use the recently built image. The deploy stage will send the docker images to Docker Hub (Image Registry) and commit the recently updated Kubernetes manifests. To **Setup ArgoCD** in the Kubernetes Cluster, we can apply the official ArgoCD Kubernetes Manifest directly into a new namespace called ArgoCD. Make sure to disable TLS since this will be done only in a local home lab which prioritizes giving ease of access without worrying too much about security, this can be done by applying a custom ConfigMap that will be discussed later on in the Implementation Chapter.

To **Deploy an Example App**, this research will use a slightly modified version of GoogleCloudPlatform/microservice-demo and ArgoCD to demonstrate the implementation, to deploy the example app using the ArgoCD, use the ArgoCD UI to register a new application in ArgoCD and then proceeds to sync up the manifests with the current deployment state from the Kubernetes.

DOI : https://doi.org/10.36341/rabit.v9i1.3787
Corresponding Author : Rayhan Gusty Alif

RESULT

There is no formally defined implementation of GitOps in a Containerized Infrastructure, although based on previous researches and industry standards, we can generalize the existing design of GitOps implementation to be:

1. Setup Kubernetes Cluster using K3s provisioned by Rancher Desktop.

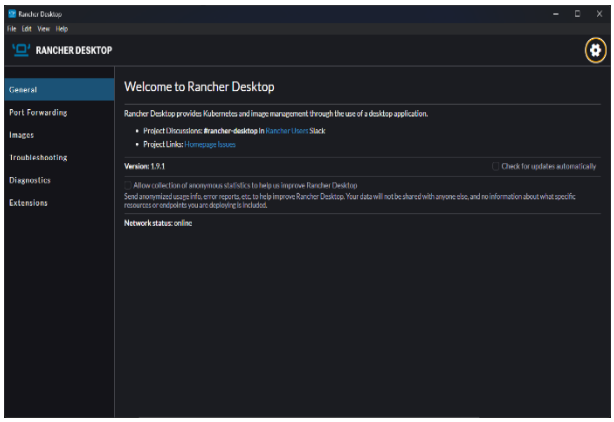


Figure 2. Rancher Desktop Main Page

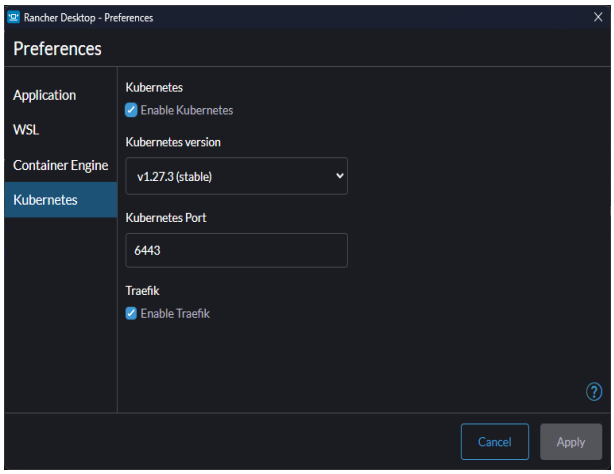


Figure 3. Rancher Desktop Preferences Page

Enable K3s Cluster inside the Rancher Desktop Settings as shown in Figure 2 and Figure 3.

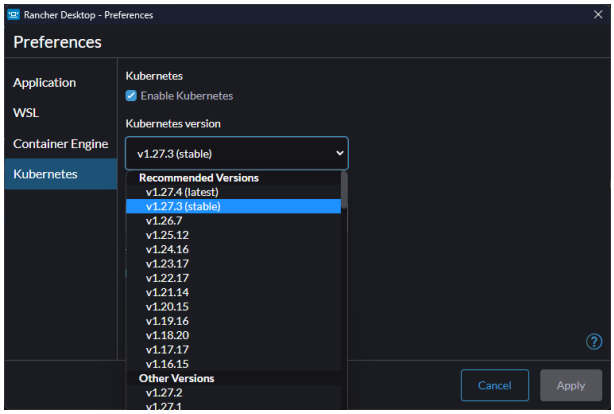


Figure 4. Rancher Desktop Kubernetes Version Setting

Pick the latest *stable* version of Kubernetes as shown in Figure 4.

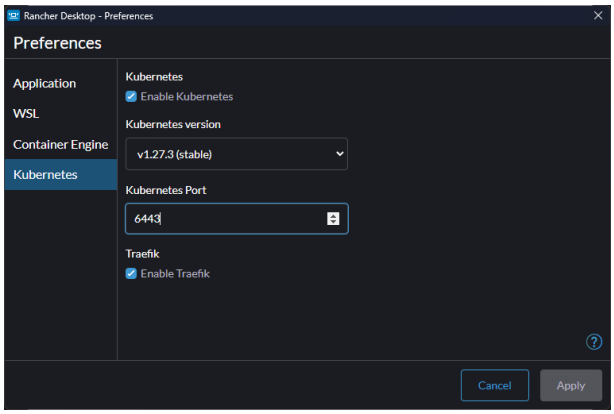


Figure 5. Rancher Desktop Kubernetes Port Setting

Pick any port for the Kubernetes Port as shown in Figure 5.

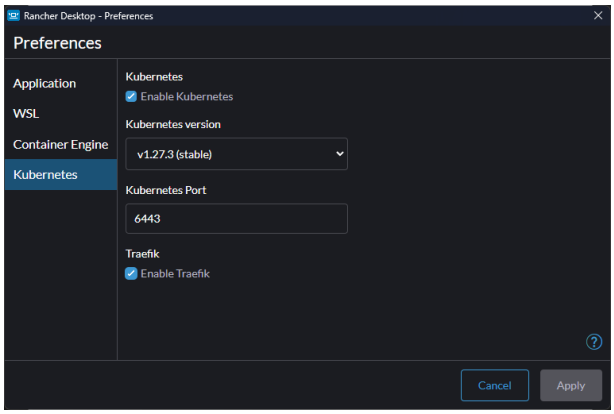


Figure 6. Rancher Desktop Kubernetes Traefik Setting

DOI : https://doi.org/10.36341/rabit.v9i1.3787
Corresponding Author : Rayhan Gusty Alif

Enable Traefik as the default network class for the Kubernetes Cluster as shown in Figure 6.

```
C:\Users\rayha\microservices-demo>kubectl create namespace git-ops
namespace/git-ops created
```

Figure 7. Creating Kubernetes Namespace called *git-ops*

```
C:\Users\rayha\microservices-demo>kubectl create namespace argocd
namespace/argocd created
```

Figure 8. Creating Kubernetes Namespace called *argocd*

Create new Kubernetes Namespaces as shown in Figure 7 and Figure 8.

2. Setup Gitlab CI/CD Pipeline

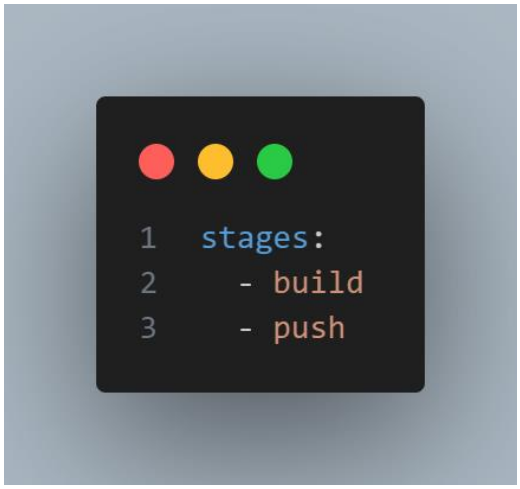


Figure 9. Defined stages in .gitlab-ci.yml

Define the stages for the CI/CD Pipeline as shown in Figure 9 which are build and push stages.

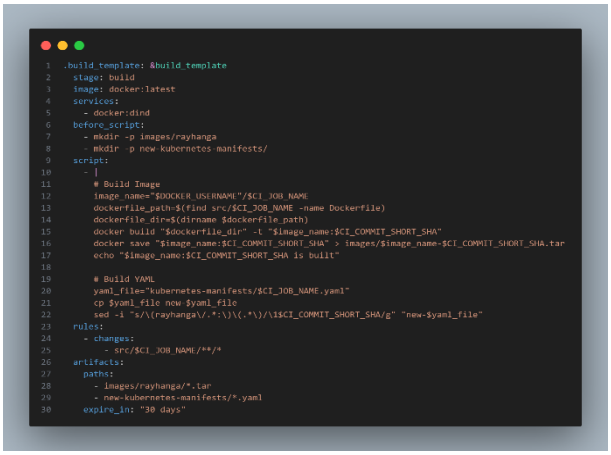


Figure 10. Definition of build_template

Define the build job template as shown in Figure 10 which will build said web app.

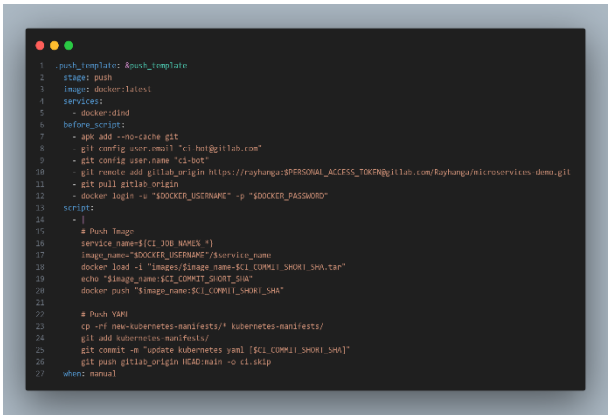


Figure 11. Definition of push_template

Define the push job template as shown in Figure 11.

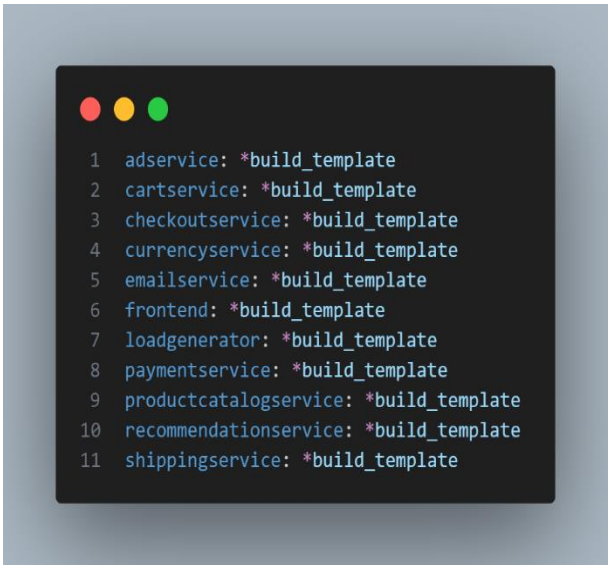


Figure 12. Defined jobs that uses build_template

Define each service build job as shown in Figure 12.

DOI : https://doi.org/10.36341/rabit.v9i1.3787
Corresponding Author : Rayhan Gusty Alif

```
adservice_push:
  <<: *push_template
  dependencies:
    - "adservice"
cartservice_push:
  <<: *push_template
  dependencies:
    - "cartservice"
checkoutservice_push:
  <<: *push_template
  dependencies:
    - "checkoutservice"
currencyservice_push:
  <<: *push_template
  dependencies:
    - "currencyservice"
emailservice_push:
  <<: *push_template
  dependencies:
    - "emailservice"
frontend_push:
  <<: *push_template
  dependencies:
    - "frontend"
loadgenerator_push:
  <<: *push_template
  dependencies:
    - "loadgenerator"
paymentservice_push:
  <<: *push_template
  dependencies:
    - "paymentservice"
productcatalogservice_push:
  <<: *push_template
  dependencies:
    - "productcatalogservice"
recommendationservice_push:
  <<: *push_template
  dependencies:
    - "recommendationservice"
shippingservice_push:
  <<: *push_template
  dependencies:
    - "shippingservice"
```

Figure 13. Defined jobs that uses push_template

Define each service push job as shown in Figure 13.

3. Setup ArgoCD

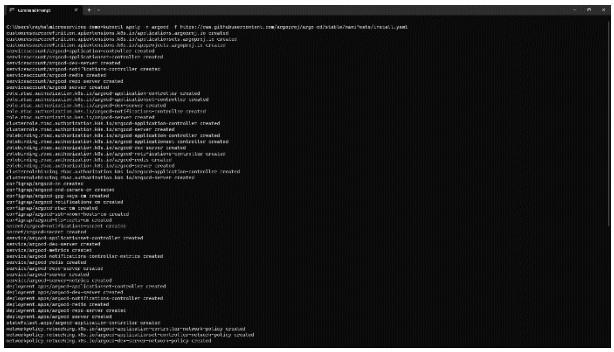


Figure 14. Applying ArgoCD Manifest

Apply the Official ArgoCD Kubernetes Manifest to the *argocd* namespace as shown in Figure 14.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: argocd-cmd-params-cm
  labels:
    app.kubernetes.io/name: argocd-cmd-params-cm
    app.kubernetes.io/part-of: argocd
data:
  server.insecure: "true"
```

Figure 15. ArgoCD ConfigMap Manifest

```
C:\Users\rayha\microservices-demo>kubectl apply -n argocd -f argocd-manifests\argocd-cm.yaml
configmap/argocd-cmd-params-cm configured

C:\Users\rayha\microservices-demo>kubectl -n argocd rollout restart deployment argocd-server
deployment.apps/argocd-server restarted
```

Figure 16. Applying ArgoCD ConfigMap and do a rollout-restart

Apply the custom ConfigMap (as shown in Figure 15) to *argocd* namespace and rollout-restart the ArgoCD deployment as shown in Figure 16.

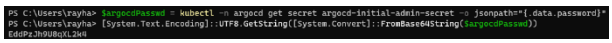
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: argocd-server-ingress
  namespace: argocd
  labels:
    name: argocd
  annotations:
    ingress.kubernetes.io/ssl-redirect: "false"
spec:
  rules:
    - host: argocd.localhost
      http:
        paths:
          - pathType: Prefix
            path: "/"
            backend:
              service:
                name: argocd-server
                port:
                  number: 80
```

Figure 17. ArgoCD Ingress Manifest

```
C:\Users\rayha\microservices-demo>kubectl -n argocd apply -f argocd-manifests\argocd-ingress.yaml
ingress.networking.k8s.io/argocd-server-ingress created
```

Figure 18. Applying ArgoCD Ingress

Apply the Ingress Manifest (as shown in Figure 17) to enable access through web browser to ArgoCD UI as shown in Figure 18.



```
PS C:\Users\rayha> $argocdPassed = kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}"
PS C:\Users\rayha> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($argocdPassed))
EddPzJh9U8qXL2H4
```

Figure 19. Getting the initial admin password for ArgoCD UI

Get the initial admin password using powershell command as shown in Figure 19.

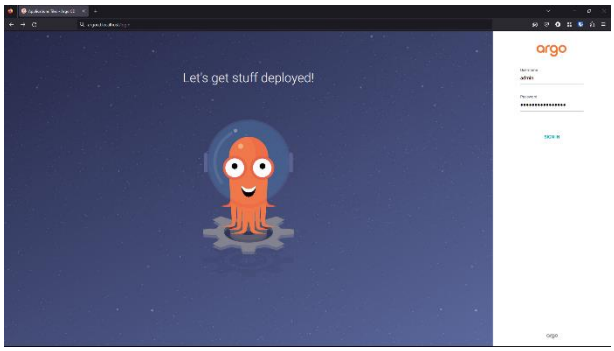


Figure 20. ArgoCD UI Login Page

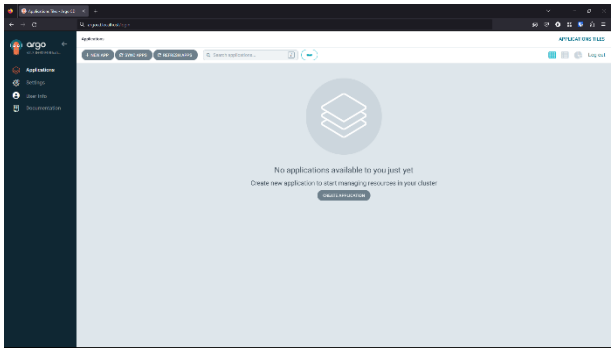


Figure 21. ArgoCD UI Homepage

Access the ArgoCD UI as shown In Figure 20 and Figure 21.

4. Deploy Example App

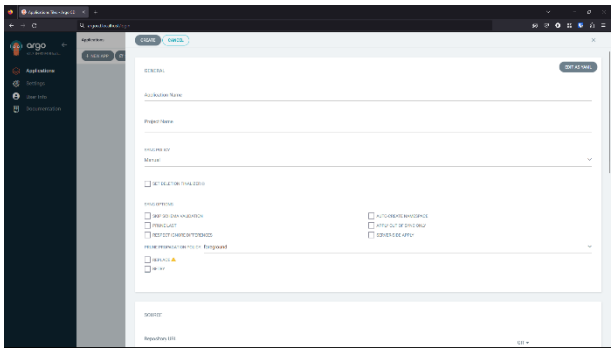


Figure 22. ArgoCD UI Create App Page

Create new App in ArgoCD UI as shown in Figure 22.

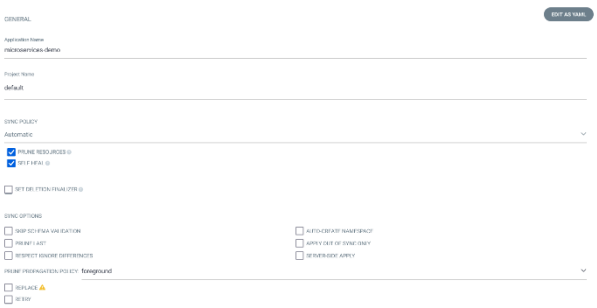


Figure 23. ArgoCD UI Create App GENERAL Section

Fill in the GENERAL section as shown in Figure 23.

DOI : https://doi.org/10.36341/rabit.v9i1.3787
Corresponding Author : Rayhan Gusty Alif

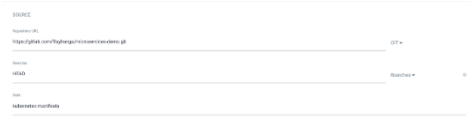


Figure 24. ArgoCD UI Create App SOURCE Section

Fill in the SOURCE section as shown in Figure 24

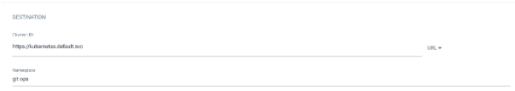


Figure 25. ArgoCD UI Create App DESTINATION Section

Fill in the DESTINATION section as shown in Figure 25.

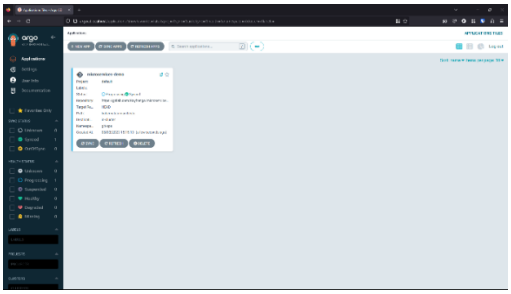


Figure 26. ArgoCD UI HomePage with the new Application

Create the application, and the created application will be listed in the ArgoCD UI Homepage as shown in Figure 26.

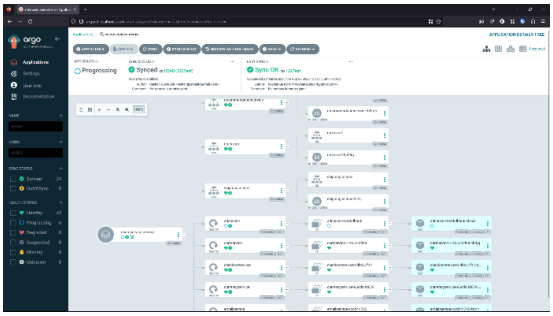


Figure 27. ArgoCD UI Example Application Services

Each services of the application can be seen by click the application from the ArgoCD UI Homepage, the list of services will be shown as in Figure 27.

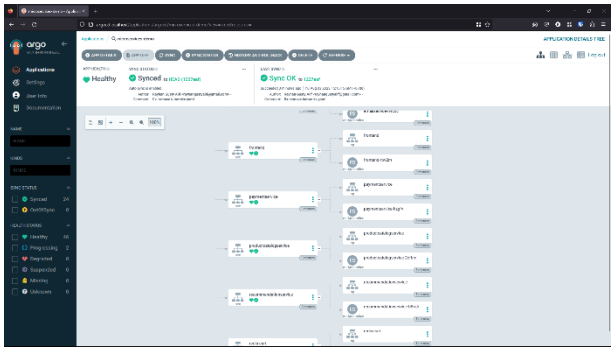


Figure 28. APP HEALTH is Healthy

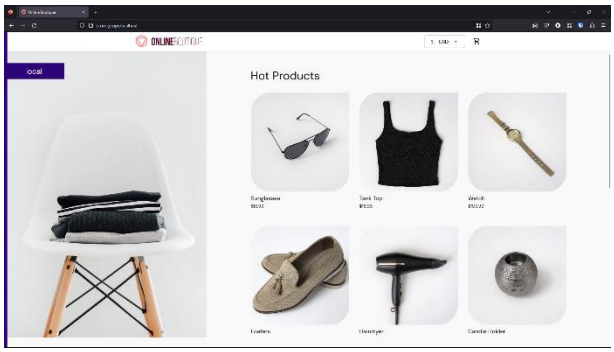


Figure 29. Working Web App at store.gitops.localhost

Once the APP HEALTH is Healthy as shown in Figure 28, proceed to open the deployed web app by accessing store.gitops.localhost as shown in Figure 29.

CONCLUSION

GitOps Implementation in a Containerized Infrastructure requires a lot of preparation in advance, but this preparation would pay off later on when changes are made to the application since all the changes would be automatically updated and all the needed IT Infrastructure would be provisioned

DOI : https://doi.org/10.36341/rabit.v9i1.3787
Corresponding Author : Rayhan Gusty Alif

automatically inside the Containerized Infrastructure.

GitOps is not limited to Containerized Infrastructure and can be implemented on any kind of IT infrastructure, but it would decrease the efficiency if it's being implemented on other than virtualized infrastructure since the GitOps method is designed with virtualization in mind, and since containerization itself is the successor of virtualization technology GitOps works perfectly with Containerized Infrastructure.

The main weakness that we can see in this approach is that it requires a quite steep learning curve and heavy initial setup, other than that there are no other weaknesses that could be given from GitOps, although to prevent the heavy setup in the future we can use some automation script either by writing our own or use some provisioning tools like Terraform or Ansible.

BIBLIOGRAPHY

[1] R. López-Viana, J. Díaz, and J. E. Pérez, "Continuous Deployment in IoT Edge Computing A GitOps implementation," in *Iberian Conference on Information Systems and Technologies, CISTI*, 2022. doi: 10.23919/CISTI54924.2022.9820108.

[2] Esa Paavola, "Managing Multiple Applications on Kubernetes Using GitOps Principles," Aug. 2021.

[3] I. Damyanov, "Corporate information infrastructure - Management aspects," *TEM Journal*, vol. 8, no. 1, 2019, doi: 10.18421/TEM81-14.

[4] O. Bentaleb, A. S. Z. Belloum, A. Sebaa, and A. El-Maouhab, "Containerization technologies: taxonomies, applications and challenges," *Journal of Supercomputing*, vol. 78, no. 1, 2022, doi: 10.1007/s11227-021-03914-1.

[5] J. P. Serrano and R. F. Pereira, "Improvement of IT Infrastructure Management by Using Configuration Management and Maturity Models: A Systematic Literature Review and a Critical Analysis," *Organizacija*, vol. 53, no. 1, 2020, doi: 10.2478/orga-2020-0001.

[6] Y. Vlasov, N. Khrystenko, and D. Uzun, "Analysis of Modern Continuous Integration/Deployment Workflows Based on Virtualization Tools and Containerization Techniques," in *Advances in Intelligent Systems and Computing*, 2020. doi: 10.1007/978-3-030-37618-5_46.

[7] T. A. Limoncelli, "GitOps: A path to more Self-service IT," *Queue*, vol. 16, no. 3, 2018, doi: 10.1145/3236386.3237207.

[8] B. Yuen, A. Matyushentsev, T. Ekenstam, and J. Suen, *GitOps and Kubernetes*. Manning Publications, 2022.

[9] F. Beetz and S. Harrer, "GitOps: The Evolution of DevOps?," *IEEE Softw*, vol. 39, no. 4, 2022, doi: 10.1109/MS.2021.3119106.

[10] S. Gupta, M. Bhatia, M. Memoria, and P. Manani, "Prevalence of GitOps, DevOps in Fast CI/CD Cycles," in *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing, COM-IT-CON 2022*, 2022. doi: 10.1109/COM-IT-CON54601.2022.9850786.

[11] G. Agarwal, *Modern DevOps Practices*. 2021.

[12] R. Salecha, "What Is GitOps?," in *Practical GitOps*, 2023. doi: 10.1007/978-1-4842-8673-9_1.