Automation and improvement in CMSWEB Kubernetes clusters

Project Report
CERN, Geneva

Submitted by

Rukaiah Badran
CERN Summer Student, 2023

Supervised by

Aroosha Pervaiz

Co-Supervised by

Muhammad Imran
Andreas Pfeiffer

CERN-THESIS-2023-123
//2023

Abstract

The recent migration of CMSWEBs VM clusters to Kubernetes clusters (k8s) improved performance remarkably, making it more sustainable than previous versions, cost-efficient operations and shorter upgrade cycles. Therefore, all of this affirms the value and necessity of CMSWEB to the Compact Muon Solenoid (CMS) experiment. Implementing Kustomize and ArgoCD into CMSWEB's k8s clusters is to be discussed further in this report, as the objective is the enhance automation and improve upon CMSWEB k8s clusters. Kustomize is a configuration management tool integrated with kubectl to modify YAML manifests indirectly, allowing complete and more straightforward customizing and managing Kubernetes manifests. Additionally, it supports reusable components maintaining separate sets of YAML files for each environment, making it a lightweight addition to CMSWEB's k8s clusters. Moreover, ArgoCD is an open-source continuous delivery tool for Kubernetes that automates application deployment and management. It also offers a web UI, CLI, and API for easy management and supports rollbacks and multi-cluster environments. This approach improves application reliability, reduces manual intervention, and enables easy rollbacks in case of any issues occurring. Altogether, these features allow seamless deployment and management of CMSWEB services and make them even more reliable.

Table of contents

I. Introduction

The CMS experiment [1], located at the CERN Large Hadron Collider (LHC), is a monumental undertaking in particle physics. It aims to uncover the universe's fundamental building blocks by colliding protons at unparalleled energies, generating vast amounts of data that hold the key to unravelling the mysteries of our cosmos. To manage and analyze this colossal data stream, the CMS experiment relies on CMSWEB, a sophisticated data management system that enables global collaboration and real-time data processing. CMSWEB's role in facilitating seamless

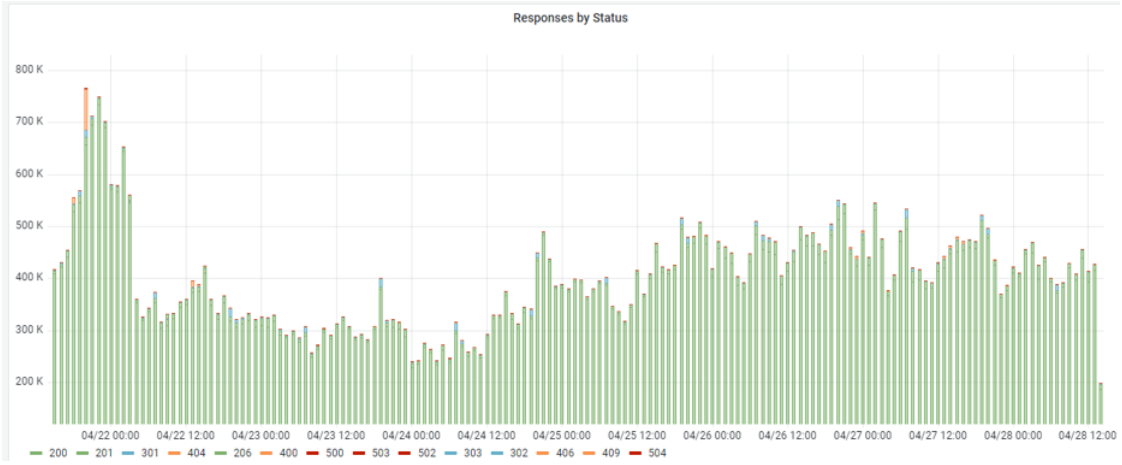

Figure I.1: CMSWEB cluster receives 0.5 million to 1 million requests per hour

data sharing and analysis cannot be understated as demonstarted in Figure I.1. Its distributed architecture leverages grid computing to provide scientists worldwide with access to the data generated by the CMS detector. This distributed approach fosters international collaboration and ensures rapid data processing and analysis, accelerating scientific discoveries within the particle physics community.

Integrating Kustomize and ArgoCD into the CMSWEB infrastructure holds immense potential to enhance its efficiency and scalability. Kustomize, a configuration customization tool, empowers CMSWEB to manage complex configurations efficiently. With Kustomize, CMSWEB can tailor its configurations for various environments without duplicating YAML files, simplifying maintenance and reducing the risk of errors. ArgoCD, a declarative, GitOps continuous delivery tool, further elevates CMSWEB's capabilities. By automating the deployment and management of applications, ArgoCD ensures consistent and reliable releases across different environments. This streamlines the deployment process, reduces human intervention, and enhances system stability.

The combination of Kustomize and ArgoCD brings several benefits to the CMSWEB ecosystem. Firstly, it enhances the flexibility of managing configuration, enabling swift adaptation to changes in requirements or infrastructure. This is crucial in the

ever-evolving landscape of particle physics research. Secondly, the GitOps approach of ArgoCD improves traceability and accountability, enabling transparent tracking of changes and simplifying troubleshooting. Moreover, the automation enabled by ArgoCD reduces the risk of manual errors during deployment and ensures consistent configurations across different stages of the system's lifecycle. This is particularly critical for a project of the scale and complexity of CMSWEB, where reliability and accuracy are paramount.

In this report, Section II overviews Kustomize and describes how these customizations have been implemented. Section III introduces ArgoCD and explains how it is used to implement applications for test clusters. Section IV briefly introduces CircleCI and how it can be used for services for the CMSWEB cluster in the future. Section V provides quick detail about documentation and support. Finally, the report closes in Section VI.

II. Kustomize

Kustomize was primarily conceptualized and developed by the engineering team at Google and is the brainchild of Brendan Burns, a renowned software engineer known for his significant contributions to the Kubernetes ecosystem. Kustomize can be best described as a configuration management tool that simplifies the customization of Kubernetes manifests. It separates the configuration from the resource definitions, enabling developers to modify the configuration without directly editing the resource files. This approach is based on a series of overlay files that contain changes to be applied to the base resource definitions. These overlays can be managed in version control systems, promoting better collaboration and tracking of changes.

II.1 Key Features and Benefits

1. **Declarative Approach:** Kustomize follows a declarative approach to configuration management. This means that developers specify what they want the final configuration to look like, and Kustomize applies the necessary changes to achieve that state.

2. **Seamless Environment Customization:** With overlays, developers can easily customize configurations for different environments (such as development, staging, and production) without duplicating entire resource definitions.

3. **Version Control Integration:** Kustomize's use of overlay files lends itself well to version control. Changes to configurations can be tracked, reviewed, and managed using tools like Git, enhancing collaboration and auditability.

4. **Modularity:** Kustomize promotes modularity by allowing the creation of reusable components that can be shared across different projects. This reduces duplication and encourages best practices.

5. **Automation and CI/CD Integration:** Kustomize can be seamlessly integrated into continuous integration and continuous deployment (CI/CD) pipelines, enabling the automated and consistent deployment of applications.

Kustomize was officially integrated into the kubectl CLI [2] as a standalone sub-command in Kubernetes version 1.14, released in March 2019 [3]. This integration aimed to provide users with a more streamlined and seamless experience when working with customization's of Kubernetes resources. By integrating it directly into kubectl, Kubernetes made it easier for users to manage configurations and apply customizations to resources, thereby enhancing the overall Kubernetes experience.

II.2 Implementation

In this project, to learn how to implement the tools required, a repository was created [4] to host the work done without disrupting the already-made repository [5]. One can fetch these repositories, push, and update their locally cached repositories to match the changes in the repositories, as mentioned earlier.

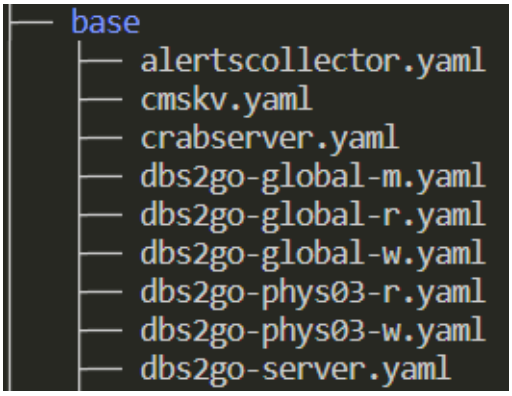

Figure II.1: Structure of base in this project

To implement Kustomize, the initial requisite is the installation of the tool, which can be achieved through the download of the binary or the utilization of the kubectl kustomize command. The foundational step involves crafting a base configuration, functioning as the fundamental scaffold for subsequent customization endeavours as referenced in Figure II.1.

Figure II.2: Overlay of cmskv.yaml

Subsequently, creating overlays becomes paramount, each encapsulating distinct modifications or customization's meticulously organized within hierarchical directory structures. Within each overlay, meticulous modifications to YAML files are undertaken, encompassing the precise alterations desired as can be seen in Figure II.2, whether the addition, modification, or removal of fields. Each overlay is associated with a dedicated kustomization.yaml file is a nexus linking the base configuration and overlay files.

Figure II.3: Final structure of the repository

After implementing the configurations seen in Figure II.3, execution of the kubectl kustomize command yields the final, refined Kubernetes resource configurations tailored to the specific overlay in consideration. The subsequent deployment of these configurations to the cluster is orchestrated through the judicious employment of the kubectl apply command

The exploration of advanced Kustomize features, inclusive of variables, patches, and strategic merge patches, serves as a logical progression, enabling the handling of intricate customizations while fostering modularity and reusability in configurations. The ultimate step of embedding Kustomize within the organization's CI/CD pipelines, such as ArgoCD (look in section III), assumes paramount importance, as it automates the deployment process, thereby amplifying efficiency in managing myriad environments and developmental stages. Adhering to these systematically structured steps culminates optimizing Kustomize's capabilities, furnishing an elevated trajectory for one's Kubernetes deployment workflows.

III. ArgoCD

ArgoCD [6], at its essence, is an application delivery platform meticulously tailored for Kubernetes. It empowers users to declaratively define the desired state of their applications, utilizing Git repositories as the sacrosanct source of truth. This methodology harmoniously aligns with the principles of GitOps, whereby all application configurations and deployment definitions are diligently versioned within a Git repository. ArgoCD perpetually monitors this prescribed desired state, diligently ensuring that the actual state of the deployed applications in the Kubernetes cluster remains congruent with this ordained objective.

ArgoCD originated as part of the Argo project [7], a consortium of open-source tools meticulously designed to simplify various aspects of Kubernetes orchestration and the automation of workflows. The Argo project was conceived under the stewardship of Applatix [8], a company at the forefront of container-native workflow automation. ArgoCD, a distinctive component within the Argo suite, was tailored to address the complexities entailed in managing Kubernetes resources and deploying applications in a declarative and automated paradigm.

Applatix is the bedrock upon which ArgoCD's initial development took place, providing the foundation for a resilient and flexible tool. However, akin to the success stories of numerous thriving open-source projects, ArgoCD's evolution has been profoundly influenced by the contributions of the extensive Kubernetes and cloud-native community. The collaborative endeavours of diverse developers, organizations, and ardent Kubernetes aficionados have enriched ArgoCD, enhancing its versatility and ensuring its adaptability to diverse use cases.

III.1 Key Features and Benefits

1. **GitOps Workflow:** ArgoCD implements a GitOps-centric workflow, allowing teams to manage application configurations effectively, meticulously track changes, and robustly enforce uniformity by leveraging the capabilities of version control.

2. **Automated Synchronization:** ArgoCD's intrinsic strength lies in continuously monitoring the designated Git repository. It adeptly detects alterations and initiates automatic synchronization of Kubernetes resources, ensuring impeccable alignment with the desired state.

3. **Facilitating Application Rollouts:** ArgoCD extends its prowess to support automated rollouts, simplifying the management of updates and facilitating swift rollbacks, ensuring the Kubernetes cluster remains harmonious and consistent.

4. **Declarative Configuration:** A hallmark of ArgoCD is its steadfast commitment to a declarative approach. This simplifies the intricate landscape of application deployment, systematically mitigates the need for manual intervention, and significantly enhances the reproducibility of deployments.

5. **Efficient Multi-environment Management:** ArgoCD's efficacy transcends the confines of single-environment management. It thrives in multi-environment scenarios, adeptly handling diverse environments such as development, staging, and production, all while guaranteeing unwavering uniformity across configurations.

ArgoCD's triad of tools—Web UI, CLI, and API—form a comprehensive ecosystem that caters to a diverse range of users, from those who prefer visual interactions and easy configuration to those who seek the power of command-line control and the extensibility of programmable interfaces. This versatility ensures that ArgoCD remains adaptable to various workflows, promotes collaboration, and meets the demands of modern application management in Kubernetes environments. By leveraging the ArgoCD Web UI, CLI, and API in concert, organizations can optimize their application deployment pipelines, enforce GitOps best practices, and achieve higher operational efficiency.

III.2 Implementation

In this project, to learn how to implement the tools required and manage the application, a web UI has been created to access the project [9].

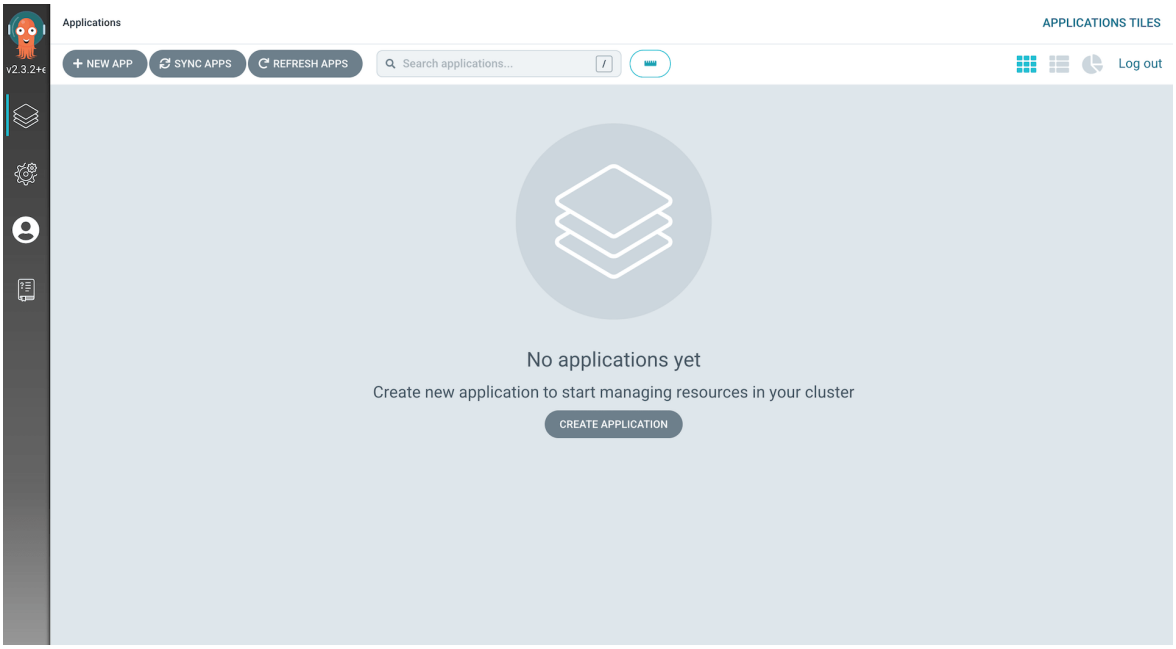

Figure III.1: ArgoCD Web UI

Initiate the process by installing ArgoCD into a Kubernetes cluster, meticulously adhering to the clear and concise installation guidelines in the official documentation. Access the ArgoCD web UI as seen above in Figure III.1, a user-friendly interface that facilitates an intuitive and interactive engagement with the platform. Within this interface, one shall manage applications and configurations and diligently monitor the deployment status.

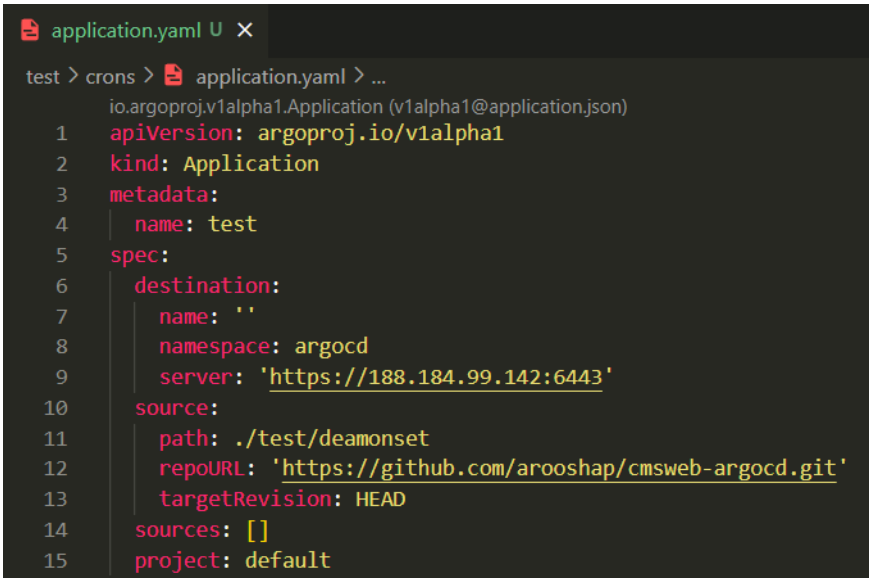

Figure III.2: CronsJob application manifest in test environment

The application.yaml file within the context of ArgoCD holds paramount significance, representing a pivotal cornerstone in the orchestration of application deployment. This YAML file functions as an authoritative declaration, encapsulating meticulous metadata, configuration specifications, and intricate synchronization details pertinent to the associated application, an example of which is Figure III.2. It emerges as a strategic directive, impeccably guiding ArgoCD in managing the application's deployment lifecycle. Within this file, discerning parameters, including the precise source of the application's manifests, the designated destination namespace within the Kubernetes cluster, the meticulous synchronization policy governing the deployment cadence (whether manual or automatic), and an array of other critical customization directives, are diligently delineated.

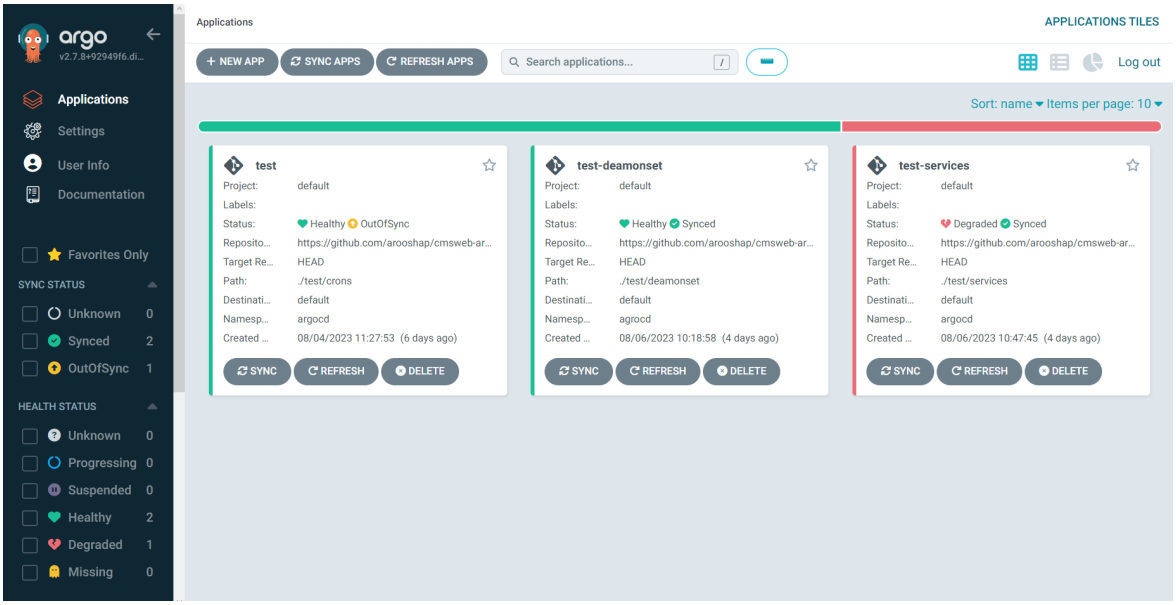

Figure III.3: Applications and their states

As the synchronization process is initiated, meticulously scrutinize the rollout status presented in the ArgoCD UI. This detailed insight into the deployment's progression offers an invaluable vantage point, enabling an astute comprehension of how modifications are meticulously applied, all while maintaining vigilant oversight over the application's operational health observed in the aforementioned Figure III.3. ArgoCD's inherent capability for rollbacks presents an additional layer of control, empowering users to revert to previous application versions when exigencies warrant.

The nexus between ArgoCD and the cluster's CI/CD pipeline materializes as an embodiment of automation. Capitalize on industry-standard CI/CD tools such as CircleCI or GitLab CI/CD, adroitly orchestrating ArgoCD synchronization events upon merging changes into the repository. This synergistic integration seamlessly embeds application deployments within the larger context of the continuous delivery process, effectively harmonizing software delivery and management.

IV. Future Remarks

As we continue to refine and advance our deployment and continuous delivery practices, the implementation of CircleCI [10] is the next logical step in our journey towards optimal software delivery. By integrating CircleCI into our pipeline, we can further improve our automation abilities, ensuring quicker build and testing cycles while maintaining robust quality assurance. Moreover, it is keeping up to date with modern DevOps techniques, streamlining our workflows, and ultimately accelerating the delivery of high-quality software. Alongside pre-existing tools, CircleCI will empower us to accomplish more outstanding consistency, scalability, and agility in our development process. Overall, we are poised to embark on a path that promises to elevate our development and deployment capabilities to new peaks.

V. Documentation and Support

Documentation plays an important part in familiarizing others with your work and increases the usability of one's code fourfold. For this reason, reading the documentation of both tools is highly useful. This can be found at Kubernetes [11] official website for Kustomize [3], and for ArgoCD [6] the documentation they provide is well maintained.

VI. Conclusion

In conclusion, the seamless integration of Kustomize, ArgoCD, and CircleCI within the CMSWEB infrastructure represents a transformative stride forward in the operational capabilities of this foundational data management system for the CMS experiment. This strategic amalgamation of robust tools brings forth a comprehensive array of advantages that hold the potential to revolutionize the entire operational landscape.

Kustomize, the configuration customization tool, affords CMSWEB the vital agility to manage configurations with a heightened efficiency level. This adaptive capability, vital in a dynamic scientific endeavour such as particle physics, simplifies maintenance while simultaneously curbing the risk of errors stemming from duplicated YAML files, ensuring a streamlined operational environment.

ArgoCD contributes a pivotal layer of automation, culminating in a deployment process characterized by consistency and reliability throughout the system's intricate lifecycle. This profound level of automation is not solely pivotal in error reduction but also the optimization of deployment procedures, culminating in heightened system stability and an elevated degree of traceability that fundamentally simplifies troubleshooting efforts.

The pivotal inclusion of CircleCI propels this transformative endeavour to an even higher echelon. CircleCI, by automating build, test, and deployment processes, engenders a seamless and continuous integration and delivery pipeline. This meticulously structured pipeline ensures comprehensive testing and integration before any code alterations attain the production environment. Such meticulous testing enhances overall reliability and the quality of the software, elevating confidence in system operations.

As a collective entity, this multifaceted integration imparts CMSWEB with the capacity to effectively address the ever-evolving exigencies inherent in managing and analyzing the colossal data streams emerging from the CMS detector. With this integration in place, CMSWEB secures a framework characterized by streamlined operations, heightened efficiency, and unwavering robustness. The unification of technology and scientific endeavour, as exemplified in this integrated approach, steadfastly pushes the boundaries of human knowledge.

Acknowledgments

I am very thankful to CERN for providing me with this invaluable opportunity to not only gain technical knowledge but also gain insight into how things work at a professional level. I would like to express my gratitude to all my supervisors namely, Aroosha Pervaiz, Muhammad Imran, and Andreas Pfeiffer. However, I wish to extend my special thanks to Aroosha Pervaiz for always guiding me through everything and teaching me invaluable skills; it would not have been possible without her help and support.

Bibliography

[1] C. Collaboration, S. Chatrchyan, G. Hmayakyan, V. Khachatryan, A. Sirunyan, W. Adam, T. Bauer, T. Bergauer, H. Bergauer, M. Dragicevic, *et al.*, "The cms experiment at the cern lhc," *JInst*, vol. 3, p. S08004, 2008.

[2] "kubectl command line tool." https://kubernetes.io/docs/reference/kubectl/.

[3] "Kustomize: Declarative management of kubernetes objects using kustomize." https://kubernetes.io/docs/tasks/manage-kubernetes-objects/kustomization/.

[4] "arooshap/cmsweb-argocd." https://github.com/arooshap/cmsweb-argocd.

[5] "dmwm/cmskubernetes." https://github.com/dmwm/CMSKubernetes.

[6] "Argocd." https://argo-cd.readthedocs.io/en/stable/.

[7] "Argo project." https://argoproj.github.io.

[8] "Applatix." https://applatix.com.

[9] "Argocd web ui." https://137.138.226.175/.

[10] "Circleci." https://circleci.com/.

[11] "Kubernetes." https://kubernetes.io.