Jurnal: Elektrika Borneo (JEB)
Vol. 9, No. 1, April 2023, hlm. 8-14
p-ISSN 2443-0986
e-ISSN 2685-001X

AUTOMATE DEPLOYMENT APLIKASI WEB MENGGUNAKAN METODE GITOPS PADA KUBERNETES CLUSTER (Studi kasus: Web Riset Informatika Universitas Muhammadiyah Malang)

Fachry Fathurahman[1], Ir. Agus Eko Minarno, S.Kom. M.Kom.IPM.[2], Mahar Faiqurahman, S.Kom., MT.[3]

[1,2,3]Jurusan Informatika, Fakultas Teknik, Universitas Muhammadiyah Malang, Malang, Jawa Timur, Indonesia

[1]fachryfathurahman@webmail.umm.ac.id
[2]aguseko@umm.ac.id
[3]mahar@umm.ac.id

***Abstract* - The increase in internet users requires web services to have High Availability, which is the ability of web services to serve users and reduce downtime in the shortest possible time. One of the factors that can increase downtime is during the deployment process. Repetitive and manual deployment processes will be very vulnerable to human errors which will have an impact on downtime. On Informatics research website of the Muhammadiyah University of Malang is still implementing web services without using automation in deployment and also not using web containerization. In this study, the researcher designed a server environment that supports the two previously mentioned, by implementing Jenkins server as a CI/CD tool, cluster kubernetes as an orchestration container, and Github as a repository that supports the GitOps method, as a source of truth. This study tested 4 Informatics research website of the Muhammadiyah University of Malangg in automate deployment and got an average time of 126.74 seconds to build, push, and deploy.**

*Keywords— **GitOps, Kubernetes, Automasi, CI/CD***

Intisari— Meningkatnya pengguna internet mengharuskan web service memiliki High Availability, yaitu kemampuan web service dalam melayani pengguna dan mengurangi waktu downtime sesingkat-singkatnya. Salah satu faktor yang dapat meningkatkan downtime adalah ketika proses deployment. Proses deployment yang repetitif dan manual akan sangat rentan terhadap human error yang akan berakibat meningkatnya waktu downtime. Pada web riset informatika universitas Muhammadiyah Malang masih menerapkan web service tanpa menggunakan automasi dalam deployment dan juga belum menggunakan kontainerisasi web. Pada penelitian ini peneliti merancang lingkungan server yang mendukung kedua hal yang disebutkan sebelumnya, dengan mengimplementasikan Jenkins server sebagai tools CI/CD, kubernetes kluster sebagai kontainer orchestration, dan Github sebagai repositori yang mendukung metode GitOps, sebagai source of truth. Penelitian ini menguji coba 4 produk web riset informatika Universitas Muhammadiyah Malang dalam melakukan automate deployment dan mendapatkan waktu rata-rata 126.74 detik dalam melakukan build, push, dan deployment

Kata Kunci **– GitOps, Kubernetes, Automasi, CI/CD**

I. PENDAHULUAN

Perkembangan pengguna internet semakin lama semakin meningkat. Hal ini juga akan berdampak pada meningkatnya traffic pada sebuah web service. Salah satu ciri dari web service yang baik adalah web service yang memiliki ketersediaan yang tinggi atau high availability [1]. high availability tercapai jika suatu sistem tersedia setidaknya 99.999% sepanjang waktu, atau sekitar 5 menit pemadaman dengan rentang waktu satu tahun [2]. Hal tersebut dapat dicapai dengan menerapkan sistem terdistribusi dari kubernetes kluster [3]. Kubernetes merupakan salah satu platform open-source yang digunakan selain untuk meningkatkan availability juga dapat digunakan untuk melakukan manajemen workloads aplikasi yang dikontainerisasi. Pembuatan kubernetes kluster dapat melalui berbagai metode, seperti eksctl, kops, [4] kubeadm, ataupun terraform [5]. Dalam skala yang besar, sangat penting untuk memiliki sebuah platform orkestrator untuk menangani aplikasi yang dikontainerisasi [6].

Selain dari high availability, waktu yang dibutuhkan untuk melakukan deployment sebuah aplikasi stateful ataupun stateless [7] juga sangat penting. Deployment tanpa sebuah automasi akan memakan waktu yang yang lebih banyak dan setiap pengembang aplikasi melakukan perubahan pada code, harus dilakukan deployment kembali [8]. Oleh karena itu, dibutuhkan alat automasi seperti Jenkins untuk melakukan automasi proses deployment pada kubernetes kluster.

Menurut stakeholder terkait, aplikasi web riset informatika Universitas Muhammadiyah Malang saat ini menggunakan infrastruktur tradisional IT menggunakan virtualisasi dengan platform Proxmox. Untuk menyediakan satu web service, dibutuhkan setidaknya satu linux container (LXC) [9] atau satu virtual machine (VM) [10]. Hal ini akan berdampak buruk jika lalu lintas suatu aplikasi meningkat dan dapat menyebabkan single point of failure (SPOF), yaitu kondisi dimana server tidak dapat atau gagal dalam merespon request client. Hal ini dapat di atasi dengan menggunakan teknik horizontal pod

autoscaler(HPA) yang merupakan fitur dari Kubernetes [11] dan fitur ini dapat ditingkatkan dengan model ARIMA [12] . Pada penelitian [13] telah melakukan pengujian performa web server pada kluster kubernetes yang menerapkan HPA. Web server yang digunakan pada penelitian tersebut adalah web server nginx. Kluster kubernetes pada penelitian tersebut berada diatas virtual machine dengan 1 master node dan 2 worker node. Penelitian tersebut membuktikan bahwa server yang menerapkan virtualisasi dan autoscaling lebih baik daripada server yang tidak menerapkan virtualisasi dan autoscaling dengan perbandingan parameter throughput sebesar 4494.32 KB/s dan parameter response time sebesar 14.46 request/detik serta parameter CPU usage sebesar 13.01%.

Web riset informatika Universitas Muhammadiyah Malang kedepannya akan menjadi kumpulan dari web-web service dari hasil riset para peneliti. Setiap web service memiliki kebutuhan lingkungan dan prasyarat yang berbeda-beda. Oleh karena itu, teknologi kontainerisasi sangat di butuhkan dalam mengatasi masalah tersebut. Selain itu, dari sisi administrator server juga harus menyiapkan LXC atau sebuah VM pada proxmox VE [14] untuk masing-masing web service secara manual. Penelitian tersebut [14] menunjukkan rata rata waktu downtime 15,7s dan waktu migrasi sebesar 20s untuk VM pertama dan waktu downtime sebesar 15,3s dan waktu migrasi sebesar 18.0s pada VM kedua.

Pada penelitian [15] telah dilakukan automasi kluster kubernetes menggunakan jenkins dan ansible. Penelitian tersebut menggunakan Amazon Web service sebagai tempat kluster kubernetes, Docker Hub sebagai tempat registry image Docker, dan Github repositori sebagai tempat sumber kode aplikasi dan juga konfigurasi file untuk melakukan deployment aplikasi. Sesuai dengan penelitian [16], menggunakan private registry untuk menyimpan hasil dari build image Docker mendapatkan prioritas should have di mana private registry tersebut menggunakan firewall untuk membatasi network access-nya.

GitOps [17]atau git operations merupakan salah satu metode dalam continuous deployment. Inti dari GitOps adalah menggunakan Git repositori sebagai source of truth yang berisi detail rincian yang diperlukan dalam melakukan deployment sebuah aplikasi ke sebuah lingkungan produksi. Metode yang akan digunakan oleh peneliti adalah metode push-base model dimana deployment akan diterapkan jika terjadi event push ke git repositori.

Pada penelitian dibangun sebuah sistem yang akan melakukan automasi proses deployment produk dari web riset informatika Universitas Muhammadiyah Malang. Sistem ini membantu para peneliti untuk menerapkan proses Agile-Devops [18], [19] dalam pengembangan produk riset. Rancangan peneliti akan menggunakan tool Jenkins sebagai tool automasi dan kubernetes kluster sebagai platform orchestrator. Peneliti akan menggunakan 1 master node dengan 2 worker node berupa virtual machine yang berjalan di atas proxmox. Peneliti akan menggunakan Traefik sebagai load balancer yang akan mendistribusikan request ke setiap pod yang identik. Peneliti juga akan menggunakan private registry yang berada di luar kubernetes kluster sebagai pengganti registry Dockerhub.

II. LANDASAN TEORI

Pada penelitian yang dilakukan oleh Artur Cepuc dkk.[15] dalam penelitiannya yang berjudul ”Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon web services Using Jenkins, Ansible and Kubernetes” mengimplementasikan Continuous Integration and Deployment (CI/CD) menggunakan jenkins dan ansible. Penelitian tersebut menggunakan Amazon Web service sebagai cloud provider. Hasil penelitian tersebut berhasil membuat sistem CI/CD dari membuat aplikasi menjadi sebuah image sampai dengan deployment aplikasi ke kluster kubernetes dengan waktu 37.6 detik. Namun penelitian tersebut menggunakan Docker hub registry untuk menyimpan image yang telah dibuat. Sesuai dengan penelitian dari Arief Indriarto Haris dkk. [16] bahwa lebih baik menggunakan private Docker registry untuk menyimpan image yang telah di buat.

Pada penelitian yang dilakukan oleh Satvik Garg [17] mengatakan bahwa Ide inti dari GitOps adalah cara untuk memanfaatkan repositori Git yang berisi rincian infrastruktur yang diperlukan dalam lingkungan produksi. GitOps memiliki proses otomatis untuk menangani langkah-langkah yang diperlukan yang diperlukan untuk deployment. Misalnya, dapat melakukan deployment aplikasi baru atau memperbarui aplikasi yang ada hanya dengan memperbarui repositori.

Penelitian yang dilakukan oleh Imron Rosyadi dkk.[13], telah mengimplementasikan horizontal pod autoscaler pada kubernetes kluster Universitas Darussalam Gontor. Pada penelitian tersebut juga menggunakan Proxmox sebagai platform virtualisasi server. Penelitian performa web server dilakukan dengan dua kondisi, yaitu sebelum menerapkan horizontal pod autoscaler, dengan setelah menerapkan horizontal pod autoscaler. Adapun parameter yang di uji adalah throughput, response time, dan juga CPU usage. Pada penelitian tersebut masih menggunakan service dengan tipe NodePort untuk mengakses web aplikasi yang di uji.

Penelitian yang dilakukan untuk mengetahui perbandingan response time antara kubernetes kluster dengan Docker swarm yang dilakukan oleh Stefanus Eko Prasetyo[1]menggunakan load 300 user dan 2000 user. Dari percobaan tersebut di dapatkan hasil response time kubernetes kluster lebih unggul karena response time yang lebih rendah di bandingkan dengan Docker swarm.

Dalam Penelitian untuk melakukan Continuous Integration and Deployment (CI/CD) yang dilakukan oleh Sriniketan Mysari [8] menggunakan Jenkins dan Ansible, dengan perbandingan dengan tools integration lain yaitu Chef dan Puppet, di dapatkan hasil bahwa tools Jenkins lebih baik dengan agentless dari Jenkins. Selain itu juga, Sriniketan Mysari mengatakan bahwa Jenkins mudah dikonfigurasi, open-source, user-friendly, platform independent, fleksibel, menghemat banyak waktu dan

juga membantu pengembang untuk membangun dan menguji aplikasi secara terus menerus.

III. METODE PENELITIAN

A. *Lingkungan Penelitian*

GPenelitian ini berada pada *server* Laboratorium Informatika Universitas Muhammadiyah Malang menggunakan Proxmox VE. Adapun spesifikasi masing masing server yang di gunakan adalah

Tabel I
MASTER NODE KUBERNETES (VM):

No.	Item	Spesifikasi
1.	*Operating System*	Ubuntu 18.04.5 LTS
2.	*Memory*	8.00 GiB
3.	*Processors*	4 (1 sockets, 4 cores)[host]
4.	*Hard Disk (scsi0)*	50G
5.	*IP address*	10.10.11.232

Tabel II
WORKER NODE 1 KUBERNETES (VM):

No.	Item	Spesifikasi
1.	*Operating System*	Ubuntu 18.04.5 LTS
2.	*Memory*	8.00 GiB
3.	*Processors*	2 (1 sockets, 2 cores)[host]
4.	*Hard Disk (scsi0)*	40G
5.	*IP address*	10.10.11.233

Tabel III
WORKER NODE 2 KUBERNETES (VM):

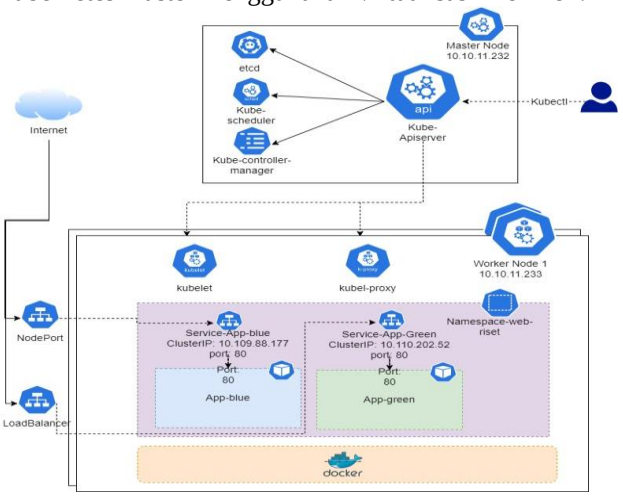
No.	Item	Spesifikasi
1.	*Operating System*	Ubuntu 18.04.5 LTS
2.	*Memory*	4.00 GiB
3.	*Processors*	2 (1 sockets, 2 cores)[host]
4.	*Hard Disk (scsi0)*	32G
5.	*IP address*	10.10.11.234

Tabel IV
JENKINS SERVER (LXC)):

No.	Item	Spesifikasi
1.	*Operating System*	Debian GNU/Linux 10 (buster)
2.	*Memory*	4.00 GiB
3.	*Processors*	2
4.	*Hard Disk (scsi0)*	32G
5.	*IP address*	10.10.11.235

B. *Arsitektur Kubernetes*

Arsitektur Kubernetes kluster terdiri dari setidaknya satu master node dan beberapa worker node [23]. Namun, pada suatu kluster juga dapat terdiri dari beberapa master node untuk mencegah single point of failure [3]. Pada penelitian ini, peneliti menggunakan satu server fisik dan membagi resource VM yang dibutuhkan untuk membuat kubernetes kluster menggunakan virtualisasi Proxmox.

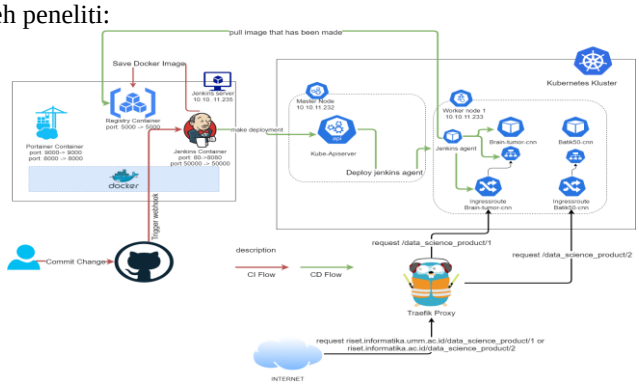


Gambar 1. Aristektur Kluster Kubernetes.

Resource paling dasar dari kubernetes kluster adalah sebuah pod. Pod dapat berisi sebuah kontainer atau lebih dengan instruksi bagaimana kontainer-kontainer tersebut dapat beroperasi satu sama lain. Pada gambar 1. terdapat dua pod, yaitu App-blue dan App-green. Satu pod dapat mewakili satu aplikasi dan pod tersebut berada pada sebuah namespace.

Ketika pod di buat, untuk mengaksesnya, kita harus menyediakan sebuah resource service bernama ClusterIP agar setiap pod dapat berkomunikasi satu sama lain secara internal. Pemisahan resource service dengan pod agar kubernetes dapat melakukan scale aplikasi secara horizontal. Misalnya, ketika pod App-blue membutuhkan resource komputational lebih, kubernetes dapat melakukan scale menggunakan horizontal pod autoscaler [11] dan membuat pod App-blue baru yang di tetapkan dengan ClusterIP yang sama dengan pod App-blue yang lama

Peneliti akan mengambil studi kasus web riset informatika Universitas Muhammadiyah Malang yang menggunakan single server. Untuk pengimplementasian akan di persiapkan sebuah kluster kubernetes sebagai container orchestration. Selain kluster kubernetes, peneliti juga akan membuat sebuah server terpisah sebagai Jenkins server dan private Docker registry untuk membuat sebuah image dan menyimpannya ke registry. Berikut adalah gambar rancangan yang akan di gunakan oleh peneliti:

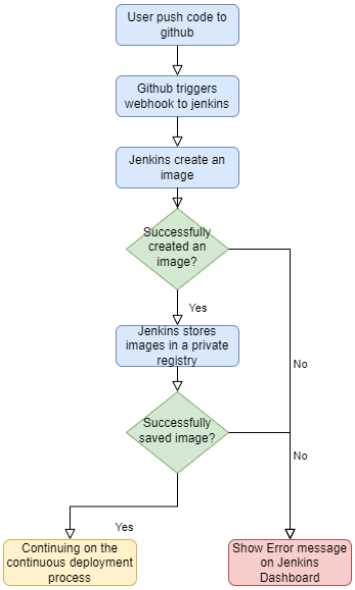


Gambar 2. Rancangan automate deployment.

Pada gambar 2, terdapat rancangan penelitian beserta alur dari automate deployment yang di ajukan oleh peneliti. Pada gambar tersebut, terdapat dua bagian, yaitu Jenkins server berupa LXC pada Proxmox VE, dan kluster kubernetes. Untuk mengatur request yang datang dari pengguna berupa alamat domain dan subfolder, peneliti menggunakan Traefik Proxy yang akan menjadi auto service discovery masing masing kontainer. Sesuai Gambar 2, jika terdapat request dari client berupa riset.informatika.umm.ac.id/data_science_product/1 maka Traefik akan mengarahkan request tersebut ke kontainer Brain tumor CNN melalui IngressRoute milik Traefik Proxy. Proses automate deployment terbagi menjadi dua proses utama, yaitu continuous integration flow dan continuous deployment flow

C. *Continuous integration flow*

Pada alur ini bertugas untuk membuat sebuah Docker image container dari aplikasi yang tersimpan di github dan menyimpan Docker image container ke dalam private registry. Adapun alur dari continuous integration dapat dilihat di bawah ini:

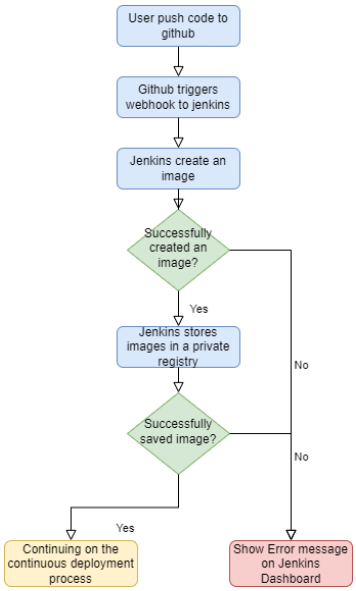


Gambar 3. Continuous Integration Flow.

Pada alur tersebut, dimulai dari user yang melakukan perubahan pada sumber kode dan melakukan push pada repositori github. Event push tersebut akan memicu webhook dan memberitahu Jenkins bahwa terjadi perubahan kode pada github. Jenkins akan membuat Docker image versi terbaru dengan menggunakan Dockerfile [25] yang tersedia pada repositori github. Ketika Jenkins berhasil membuat Docker image, Docker image tersebut akan disimpan ke private container registry tempat semua Docker image berada.

D. *Continuous deployment flow*

Pada proses continuous deployment, Jenkins berinteraksi dengan kluster kubernetes menggunakan service account kubernetes yang telah di konfigurasikan sebelumnya. service account kubernetes akan digunakan untuk keperluan deployment pod, service, inggresroute, middleware, serta Jenkins agent. Ketika proses continous integration telah selesai, Jenkins akan membuat Jenkins agent berupa pod yang berisi jnlp container dan kubectl container. Adapun alur dari continuous deployment dapat dilihat di bawah ini



Gambar 4. Continuous Deployment Flow.

E. *Rancangan Pengujian*

Peneliti menguji CI/CD pipeline yang telah dibuat menggunakan beberapa web service dari product web riset Informatika Universitas Muhammadiyah Malang. Penguji melakukan perubahan kode pada masing-masing web service sebanyak 10 kali iterasi dan mengamati apakah produk dapat di build dan di deploy sesuai dengan hasil yang di inginkan

IV. HASIL DAN PEMBAHASAN

A. *Impelentasi Kubernetes Kluster*

Sebelum membuat kubernetes kluster, peneliti terlebih dahulu menginstall Docker sebagai container runtime untuk kubernetes kluster. Untuk membuat kubernetes kluster, peneliti menggunakan tools kubeadm yang akan digunakan untuk menggabungkan 3 komputer menjadi 1 master dan 2 worker. Untuk membuat kubernetes kluster dengan tools kubeadm, pertama-tama peneliti melakukan inisiasi pada node yang akan dijadikan sebagai master atau control-plane. Setelah berhasil menjalankan perintah untuk inisiasi dan menggabungkan kubernetes kluster, hal selanjutnya yang akan dilakukan peneliti adalah menginstall add-on jaringan pod berbasis Container Network interface (CNI) yang bertujuan agar masing masing pod dapat berkomunikasi satu sama lain. Jika tidak di install, maka Cluster DNS (CoreDNS) tidak akan running. Add-on jaringan yang dipilih oleh peneliti adalah Weave Net. Setelah CNI berhasil di install, maka pod CoreDNS akan menjadi status running dan kubernetes cluster siap untuk digunakan.

Agar sebuah service dengan tipe loadbalancer dapat digunakan, dibutuhkan sebuah load balancer yang biasanya ditemukan di cloud plovider. Karena lingkungan penelitian berada di sebuah bare-metal server, maka

peneliti perlu untuk menginstall metallb manifest dengan address pool 10.10.11.243-10.10.11.245. Ketika sebuah service bertipe loadbalancer telah di deploy, maka metallb akan memberikan external ip address dari address pool dimulai dari yang pertama, yaitu 10.10.11.243.

Web riset informatika beserta produk-produk nya merupakan web service yang dikembangkan secara terpisah dan dapat berdiri sendiri. Agar dapat berjalan dengan satu domain dan dapat diarahkan dengan berbagai subdirektori, peneliti menggunakan Traefik proxy untuk melakukan hal tersebut. Traefik dirancang agar dapat mudah untuk di operasikan, tetapi mampu menangani deployment yang besar dan sangat kompleks di berbagai lingkungan dan protokol di public cloud, private, dan hybrid.

B. Implementasi Jenkins server

Jenkins merupakan tool open-source automation server yang akan digunakan untuk continous integration dan continous deployment oleh peneliti. Untuk membuat Jenkins server, peneliti membuat lingkungan terpisah dari kubernetes kluster. Peneliti menyiapkan Jenkins server dari LXC proxmox dengan basis image Debian GNU/Linux 10 (buster) yang di dalamnya akan di install Docker. Setelah melakukan instalasi Docker, peneliti kemudian menginstall Portainer yang akan digunakan untuk memanajemen container Docker. Di dalam fitur Portainer juga terdapat fitur App Template dimana pengguna dapat mendeploy container berdasarkan image yang sering digunakan seperti nginx, redis, MySQL ataupun Jenkins. Aplikasi Jenkins berupa container dengan nama Jenkins-Docker dengan menggunakan network bridge. Deployment tersebut akan meneruskan port 80 host ke port 8080 yang ada di container, dan port 50000 host ke port 50000 yang ada di container. Selanjutnya peneliti mengakses host 10.10.11.235 dengan port 80 dan melakukan konfigurasi untuk mengatur user admin.

Agar masing-masing produk dapat melakukan automasi deployment pada server, dibutuhkan Jenkins pipeline untuk setiap produk. Pipeline tersebut membutuhkan Jenkinsfile yang berisi kumpulan perintah yang akan dilakukan Jenkins untuk melakukan deployment suatu aplikasi. Peneliti membagi Jenkinsfile menjadi 3 bagian perintah, yaitu membuat image Docker, melakukan push ke registry, dan melakukan deployment di kubernetes kluster. Untuk stage pertama dan kedua menggunakan Docker plugin untuk membuat (build) dan menyimpan (push) Docker image ke private registry yang berada bersama-sama dengan Jenkins server. Peneliti menggunakan BUILD_NUMBER yang merupakan environment variable yang digunakan untuk membuat versioning untuk masing-masing image. Untuk stage ketiga, Jenkins melakukan ssh connection ke control-plane kubernetes kluster dan melakukan deployment menggunakan manifest yang tersedia di repositori masing-masing produk web service. Jenkins akan mengupdate manifest tersebut agar melakukan deployment menggunakan image Docker yang telah di push di registry.

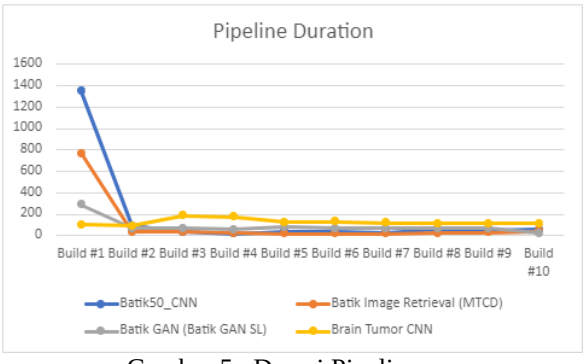
C. Kontainerisasi Web service

Produk-produk web riset informatika menggunakan bahasa pemrograman python dengan framework Flask. Agar lebih mudah dalam melakukan delivery produk web, peneliti melakukan kontainerisasi menggunakan Docker image dengan dasar image python. Tahapan pembuatan Docker image produk web riset informatika dibagi menjadi tiga bagian. Bagian pertama adalah membuat image dasar dari python dan menginstall package-package yang dibutuhkan agar web service dapat berjalan. Bagian kedua adalah menyalin sumber kode ke dalam image agar web service dapat mengetahui apa yang akan dijalankan ketika kontainer dibuat menggunakan image yang telah di build. Bagian terakhir adalah mengekspos port yang akan digunakan gunicorn (python web server) dan menjalankan gunicorn dengan port tersebut.

Di dalam repositori masing-masing produk juga terdapat sebuah kubernetes manifest untuk masing masing produk. Manifest ini akan digunakan oleh Jenkins untuk mendeploy produk web riset informatika. Manifest tersebut terdiri dari 3 kubernetes object, yaitu Deployment, service, dan IngressRoute. Deployment object akan mendeploy pod yang berisi container dari produk tersebut. Selain itu terdapat service, cara abstrak untuk mengekspos aplikasi yang berjalan pada satu set Pod sebagai sebuah network service. Setiap service akan mendapatkan ip internal dan single dns name yang digunakan untuk berkomunikasi di dalam kubernetes kluster. Bagian terakhir adalah IngressRoute, yang merupakan Custom Resource Definition (CRD) dari Traefik yang merupakan objek untuk mengarahkan request dengan subdirectory tertentu ke object service kubernetes.

D. Pengujian Sistem

Pada tahap pengujian sistem ini dilakukan peneliti untuk mengetahui apakah sistem yang telah di desain dapat berjalan dengan baik atau tidak. Pengujian dilakukan dengan mengubah varsi sumber kode dari masing-masing produk web riset informatika dan mengamati apakah terjadi proses build, push, dan deploy pada Jenkins beserta mencatat durasi dari masing masing tahapan. Berikut hasil dari pengujian sistem tersebut:



Gambar 5. Durasi Pipeline.

E. Analisis Hasil Pengujian

Dari pengukuran durasi masing-masing stage dari setiap produk web riset informatika di atas, Peneliti menemukan bahwa setiap produk yang di build pertama kali akan memakan waktu yang lebih lama dibandingkan

dengan build selanjutnya. Ini disebabkan oleh Docker yang melakukan build image dengan mengunduh package yang diperlukan masing-masing produk untuk ditambahkan ke dalam image Docker. Setelah build kedua dan seterusnya, Docker menggunakan teknologi Docker Layer Caching (DLC) sehingga tidak perlu mengunduh kembali package yang telah di install. Pada produk Batik-GAN (Batik GAN SL) telah terjadi kegagalan pipeline dimana pipeline tersebut berhenti pada stage build. Peneliti memeriksa server Jenkins dan mendapatkan bahwa registry container server Jenkins telah penuh dengan image dari build sebelumnya. Peneliti membersihkan image dari produk yang telah di uji coba yaitu Batik50_CNN dan Batik Image Retrieval (MTCD) dan menjalankan pipeline Batik-GAN (Batik GAN SL) kembali dan mendapatkan status success.

Untuk waktu rata-rata stage deployment produk web riset di kluster kubernetes memiliki waktu paling rendah, yaitu 4.87 detik dibandingkan dengan stage push yaitu selama 39.20 detik. Hal ini disebabkan karena stage deployment hanya berisi script untuk mengupdate versi dari image dari pod yang digunakan di kubernetes kluster. Untuk stage build memiliki waktu rata-rata paling lama diantara stage lain yaitu sebesar 90.02 detik atau sekitar 1 menit 30.02 detik. Untuk waktu rata-rata pipeline melakukan delivery produk web riset informatika adalah sebesar 126.74 detik atau sekitar 2 menit 6.74 detik

V. KESIMPULAN

Pada penelitian ini, peneliti melakukan implementasi automasi dari continous integration dan continous deployment menggunakan tools Jenkins sebagai tools automatisasi dengan kluster kubernetes. Penelitian ini berhasil mengimplementasikan kubernetes kluster sebagai tempat deployment produk web riset informatika Universitas Muhammadiyah Malang. Penelitian ini juga menggunakan metode GitOps sebagai metode dalam automasi build dan deployment aplikasi web yang sebelumnya belum di implementasikan di server laboratorium informatika Universitas Muhamadiyah Malang. Metode GitOps menggunakan repositori github sebagai source of truth deployment, yang artinya ketika ada perubahan kode pada branch master, akan menjalankan pipeline di Jenkins. Pengujian performa dilakukan dengan menguji coba 4 aplikasi produk web riset Informatika Universitas Muhammadiyah Malang sebanyak 10 iterasi atau pipeline. Disetiap iterasi tersebut dibagi menjadi 3 stage, yaitu build, push, dan deploy Dari hasil penelitian yang dilakukan peneliti, stage build memiliki waktu rata rata yang paling lama diantara stage yang lain. Hal ini disebabkan package yang harus di unduh terlebih dahulu saat build pada iterasi pertama. Setelah iterasi pertama, stage build akan menjadi lebih cepat karena teknologi Docker Layer Caching (DLC). Pada stage deployment memiliki waktu yang paling sedikit, karena penggunaan kubernetes kluster hanya perlu mengupdate versi image terbaru dari pod produk web riset.

REFERENSI

[1] S. E. Prasetyo and Y. Salimin, “Analisis Perbandingan Performa Web Server Docker Swarm dengan Kubernetes Cluster,” 2021. [Online]. Available: https://journal.uib.ac.id/index.php/combines.

[2] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, “Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned,” in *IEEE International Conference on Cloud Computing, CLOUD*, Sep. 2018, vol. 2018-July, pp. 970–973. doi: 10.1109/CLOUD.2018.00148.

[3] L. Widyawati, H. Santoso, and H. Budiman, “Analisa Penerapan server deployment Menggunakan Kubernetes Untuk Menghindari single of failure,” Jurnal Informatika Teknologi dan Sains, vol. 3, no. 1, pp. 267–271, 2021.

[4] A. Poniszewska-Marańda and E. Czechowska, “Kubernetes cluster for automating software production environment,” *Sensors*, vol. 21, no. 5, pp. 1–24, 2021, doi: 10.3390/s21051910.

[5] Y. Hidayat and B. Arifwidodo, “Implementasi Web Server Menggunakan Infrastructure As Code Terraform Berbasis Cloud Computing.”

[6] N. D. Nguyen and T. Kim, “Balanced leader distribution algorithm in kubernetes clusters,” *Sensors (Switzerland)*, vol. 21, no. 3, pp. 1–15, Feb. 2021, doi: 10.3390/s21030869.

[7] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, “Microservice Based Architecture: Towards High-Availability for Stateful Applications with Kubernetes,” in *Proceedings - 19th IEEE International Conference on Software Quality, Reliability and Security, QRS 2019*, Jul. 2019, pp. 176–185. doi: 10.1109/QRS.2019.00034.

[8] S. Mysari and V. Bejgam, “Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible,” Feb. 2020. doi: 10.1109/ic-ETITE47903.2020.239

[9] D. Wijayanto, A. Firdonsyah, F. Dharma Adhinata, and A. Jayadi, “Rancang Bangung Private Server Menggunakan Platform Proxmox dengan Studi Kasus: PT.MKNT.”

[10] N. K. Surbakti, M. Arif, and F. Ridha, “Implementasi Kubernetes Cluster Menggunakan KVM.”

[11] T. T. Nguyen, Y. J. Yeom, T. Kim, D. H. Park, and S. Kim, “Horizontal pod autoscaling in kubernetes for elastic container orchestration,” *Sensors (Switzerland)*, vol. 20, no. 16, pp. 1–18, 2020, doi: 10.3390/s20164621.

[12] A. Zhao, Q. Huang, Y. Huang, L. Zou, Z. Chen, and J. Song, “Research on Resource Prediction Model Based on Kubernetes Container Auto-scaling Technology,” in *IOP Conference Series: Materials Science and Engineering*, Aug. 2019, vol. 569, no. 5. doi: 10.1088/1757-899X/569/5/052092.

[13] Julianti, M., Ramdhan, S. and Mulyana, A, ”Perancangan Server Cloud Computing Model Infrastructure As A Service Berbasis Proxmox pada PT Fortuna Mediatama”. Academic Journal of Computer Science Research. Vol. 1 No. 1, July, 2019

[14] I. Rosyadi, S. N. Utama, and O. V. Putra, “Implementation Autoscaling Container Web Server using Kubernetes Promox-Based on Server University of Darussalam Gontor Implementation Autoscaling Container Web Server using Kubernetes Promox-Based on Server University of Darussalam Gontor,” *Jurnal Rekayasa Sistem Dan Industri*, vol. 6, no. June, 2019.

[15] A. Cepuc, R. Botez, O. Craciun, I. A. Ivanciu, and V. Dobrota, “Implementation of a continuous integration and deployment pipeline for containerized applications in

amazon web services using jenkins, ansible and kubernetes," *Proceedings - RoEduNet IEEE International Conference*, vol. 2020-Decem, 2020, doi: 10.1109/RoEduNet51892.2020.9324857.

[16] A. I. Haris, Rd. A. Ferianda, B. Riyanto, F. I. Nugraha, and J. Abadi, "Pengamanan Container Orchestration Berbasis Kubernetes Di Lembaga Penerbangan dan Antariksa Nasional (LAPAN)," *Jurnal Teknoinfo*, vol. 14, no. 1, p. 1, Jan. 2020, doi: 10.33365/jti.v14i1.501.

[17] S. Garg, P. Pundir, G. Rathee, P. K. Gupta, S. Garg, and S. Ahlawat, "On Continuous Integration / Continuous Delivery for Automated Deployment of Machine Learning Models using MLOps," no. Ci, pp. 25–28, 2022, doi: 10.1109/aike52691.2021.00010.

[18] K. Maroukian and S. R. Gulliver, "Exploring the Link Between Leadership and Devops Practice and Principle Adoption," *Advanced Computing: An International Journal*, vol. 11, no. 4, pp. 1–18, Jul. 2020, doi: 10.5121/acij.2020.11401.

[19] G. B. Ghantous and A. Q. Gill, "An agile-devops reference architecture for teaching enterprise agile," *International Journal of Learning, Teaching and Educational Research*, vol. 18, no. 7, pp. 128–144, 2019, doi: 10.26803/ijlter.18.7.9.

[20] O. Koshedran and V. Tkachov, "Methods for Automating Development Processes and Deployment of Microservice Applications," 2021, doi: 10.30837/csitic52021231722.

[21] S. R. Doddaguni, S. Asif S, M. MN, and R. R, "Understanding SDLC using CI/CD Pipeline," *International Journal of Soft Computing and Engineering*, vol. 9, no. 6, pp. 22–25, May 2020, doi: 10.35940/ijsce.F3405.059620.

[22] J. Onyarin Ogala, 2022. A Complete Guide to DevOps Best Practices. International Journal of Computer Science and Information Security (IJCSIS), 20(2).

[23] L. H. Phuc, L. A. Phan, and T. Kim, "Traffic-Aware Horizontal Pod Autoscaler in Kubernetes-Based Edge Computing Infrastructure," *IEEE Access*, vol. 10, pp. 18966–18977, 2022, doi: 10.1109/ACCESS.2022.3150867.

[24] A. Jeffery, H. Howard, and R. Mortier, "Rearchitecting Kubernetes for the Edge," in *EdgeSys 2021 - Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking, Part of EuroSys 2021*, Apr. 2021, pp. 7–12. doi: 10.1145/3434770.3459730.

[25] Shri Sant Gajanan Maharaj College of Engineering, Institute of Electrical and Electronics Engineers. Bombay Section, and Institute of Electrical and Electronics Engineers, *1st International Conference on Innovative Trends and Advances in Engineering & Technology : ICITAET-2019 : conference proceeding : 27th and 28th Dec. 2019*. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.