Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)
Volume 23 ; Nomor 2 ; Agustus 2024 ; Page 290-296
E-ISSN : 2615-3475; P-ISSN : 1978-6603
https://ojs.trigunadharma.ac.id/index.php/jis/index

Implementasi Deployment Layanan Website Menggunakan Kubernetes Dengan Ci/Cd Jenkins

Irvan Maulana [1], Rusydi Umar [2], Anton Yudhana[3]

[1]Teknik Informatika, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[2]Teknik Informatika, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[3]Teknik Elektro, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
Email: [1]vanzoelmaulana@gmail.com, [2]rusydi.umar@gmail.com, [3]eyudhana@ee.uad.ac.id
Email Penulis Korespondensi: prodi@mti.uad.ac.id

Article History:
Received Jun 12th, 2024
Revised Jul 09th, 2024
Accepted Aug 08th, 2024

Abstrak

Saat ini, teknologi informasi bekembang sangat pesat, baik di bidang pendidikan, pemetintahan, perdagangan , dan lain-lain. *Website* adalah kumpulan halaman dalam suatu domain yang memuat tentang berbagai informasi agar dapat dibaca dan dilihat oleh pengguna internet melalui sebuah mesin pencari[13]. Informasi yang dapat dimuat dalam sebuah *website* umumnya berisi mengenai konten gambar, ilustrasi, video, dan teks untuk berbagai macam kepentingan. *Website* menjadi salah satu alat penyampai informasi paling popiler sat ini, mulai dari pemerintahan, media, berita, perusahaan maupun personal. Sehingga dibutuhkan *website* yang dapat terus berkembang dan pemeliharaan yang lebih sederhana. Penelitian ini berfokus pada pembangunan infrastruktur *Continous Integration/Continous Delivery/Deployment (CI/CD)* dengan manajemen *cluster* menggunakan kubernetes. Metode *deployment* aplikasi menggunakan CI/CD lebih efisien untuk perkembangan aplikasi yang berjalan terus menerus. Sedangkan kubernetes sangat membantu perkembangan aplikasi yang berbasis *container* dan *microservices*[1]. Selain itu, kubernetes juga memiliki beberapa kelebihan antara lain: *auto-scaling* dan *load balancing*. Penelitian ini menghasilkan sebuah produk infrastruktur CI/CD yang membuat proses *deployment* dan pengembangan aplikasi web dapat berjalan secara cepat, efisien dan efektif.
Kata Kunci : *Continous Integration/Continous Delivery/Deployment* (CI/CD), Docker, Git, Jenkins, Kubernetes,

Abstract

Currently, information technology is rapidly advancing across various sectors such as education, government, commerce, and others. A website is a collection of pages within a domain that contains various information to be read and viewed by internet users through a search engine. Information that can be included in a website generally consists of content such as images, illustrations, videos, and text for various purposes. Websites have become one of the most popular tools for disseminating information today, ranging from governmental, media, news, corporate, to personal purposes. Therefore, there is a need for websites that can continuously evolve and require simpler maintenance. This research focuses on the development of a Continuous Integration/Continuous Delivery/Deployment (CI/CD) infrastructure with cluster management using Kubernetes. The method of deploying applications using CI/CD is more efficient for continuously running application development. Meanwhile, Kubernetes greatly aids the development of container-based and microservices applications. Additionally, Kubernetes also offers several advantages, such as auto-scaling and load balancing[15]. *This research has resulted in a CI/CD infrastructure product that enables the deployment and development processes of web applications to run quickly, efficiently, and effectively*
Keyword : *Continous Integration/Continous Delivery/Deployment (CI/CD), Docker, Git, Jenkins, Kubernetes,*

Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)
Volume 23 ; Nomor 2 ; Agustus 2024 ; Page 290-296
E-ISSN : 2615-3475; P-ISSN : 1978-6603
https://ojs.trigunadharma.ac.id/index.php/jis/index

1. PENDAHULUAN

Saat ini, teknologi informasi bekembang sangat pesat, baik di bidang pendidikan, pemetintahan, perdagangan, dan lain-lain [14]. *Website* adalah kumpulan halaman dalam suatu domain yang memuat tentang berbagai informasi agar dapat dibaca dan dilihat oleh pengguna internet melalui sebuah mesin pencari. Informasi yang dapat dimuat dalam sebuah *website* umumnya berisi mengenai konten gambar, ilustrasi, video, dan teks untuk berbagai macam kepentingan. *Website* menjadi salah satu alat penyampai informasi paling popiler sat ini, mulai dari pemerintahan, media, berita, perusahaan maupun personal. Sehingga dibutuhkan *website* yang dapat terus berkembang dan pemeliharaan yang lebih sederhana[5].

Dalam pengembangan perangkat lunak memiliki beberapa tahapan proses yang cukup penting. Tahapan tersebut disebut dengan *System Development Life Cycle* (SDLC). *System Development Life Cycle* (SDLC) merupakan siklus dalam pengembangan perangkat lunak dengan tujuan untuk membuat masalah diselesaikan secara efektif sehingga dapat menghasilkan *website* yang berkualitas dan sesuai dengan tujuan *website* tersebut[7]. Sehingga dibutuhkan proses produksi yang efektif dan efisien. Saat ini pengembangan aplikasi web bisa dengan berbagai macam cara, namun kebanyakan masih menggunakan cara konvensional. Metode ini memerlukan lebih banyak waktu dan tenaga. Dalam perkembangannya, suatu organisasi atau perusahaan biasanya terdapat tim *developer* dan tim operasionalnya sendiri, sehingga dibutuhkan suatu infrastruktur yang dapat menjembatani tim developer dan tim operasional atau tim penguji.

Menurut Ahmad Farid dan Indra Gita Anugrah dalam jurnal Implementasi CI/CD *Pipeline* Pada *Framework* Androbase Menggunakan Jenkins (Studi Kasus: PT. Andromedia) (2021:522-527) menjelaskan bahwa CI/CD adalah : "*CI atau Continuous Integration* adalah prosses dimana aplikasi akan di buat dan di uji secara otomatis setelah repositori aplikasi terintegrasi pada server CI. Sedangkan CD atau *Continuous Delivery/Deployment* adalah prosses dimana aplikasi yang telah dibuat dan di uji akan dideploy pada server produksi secatomatis setelah repositori aplikasi terintegrasi pada server CI." [1] Jadi, CI/CD adalah salah satu praktik dalam bidang DevOps yang digunakan untuk mengembangkan perangkat lunak menjadi lebih terorganisir.

Dari latar belakang tersebut, penulis bertujuan untuk membangun desain infrastruktur CI/CD dengan Docker sebagai container, Jenkins sebagai pipeline CI/CD dan Kubernetes sebagai orkestrasi container[17]. Sehingga diharapkan infrastruktur ini dapat membantu proses pengembangan aplikasi web.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Penelitian ini menggunakan metode eksperimental dengan implementasi deployment layanan *website* pada infrastruktur *cloud*. Peneliti mengembangkan pipeline *Continuous Integration dan Continuous Deployment* (CI/CD) dengan menggunakan Jenkins, yang terintegrasi dengan Kubernetes sebagai orkestrator container[3][9].

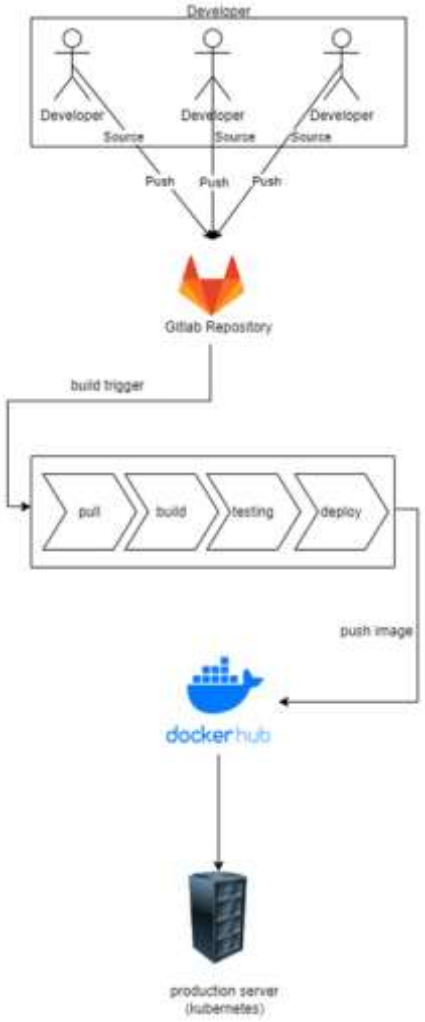
Secara keseluruhan, penelitian ini dibagi menjadi beberapa bagian, antara lain :

a. Desain Eksperimen
b. Eksperimen dilakukan dengan membangun sebuah pipeline CI/CD yang otomatis melakukan build, test, dan deployment kode sumber [16]. aplikasi *website* ke dalam lingkungan Kubernetes. Proses ini melibatkan penggunaan Docker sebagai platform containerisasi dan GitHub sebagai repositori kode sumber[4].
c. Pengumpulan Data:
d. Data dikumpulkan melalui log aktivitas Jenkins dan Kubernetes, termasuk waktu build, jumlah deployment yang berhasil, dan respons sistem terhadap beban kerja. Metrik performa sistem seperti latensi, throughput, dan utilitas sumber daya juga diukur untuk menilai efektivitas implementasi.
e. Analisis Data:
f. Analisis dilakukan dengan membandingkan performa deployment manual dengan deployment otomatis melalui pipeline CI/CD. Penelitian ini juga mengevaluasi keandalan dan skalabilitas layanan *website* yang di-deploy menggunakan Kubernetes [2].
g. Hasil:
h. Hasil eksperimen menunjukkan peningkatan efisiensi dalam proses deployment dan peningkatan kualitas layanan *website*. Implementasi Kubernetes dan Jenkins dalam pipeline CI/CD memberikan kemudahan dalam pengelolaan dan otomatisasi deployment, serta meningkatkan kecepatan dan keandalan layanan *website*.

2.2 Infrastruktur

Infrastruktur ini berjalan dari mulai tim *developer* melakukan *push* kode ke gitlab repository, kemudian pada server Jenkins akan memulai proses dari mulai build, test hingga *deployment* secara otomatis. Kemudian diteruskan ke dockerhub *registry* untuk menyimpan *image* dari Jenkins dan diteruskan dengan *deployment* ke server *production* dengan orkestrasi *cluster* oleh Kubernetes seperti pada gambar 1. Semua proses tersebut berjalan secara otomatis pada server Jenkins.

Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)
Volume 23 ; Nomor 2 ; Agustus 2024 ; Page 290-296
E-ISSN : 2615-3475; P-ISSN : 1978-6603
https://ojs.trigunadharma.ac.id/index.php/jis/index



Gambar 1. infrastruktur CI/CD

3. HASIL DAN PEMBAHASAN

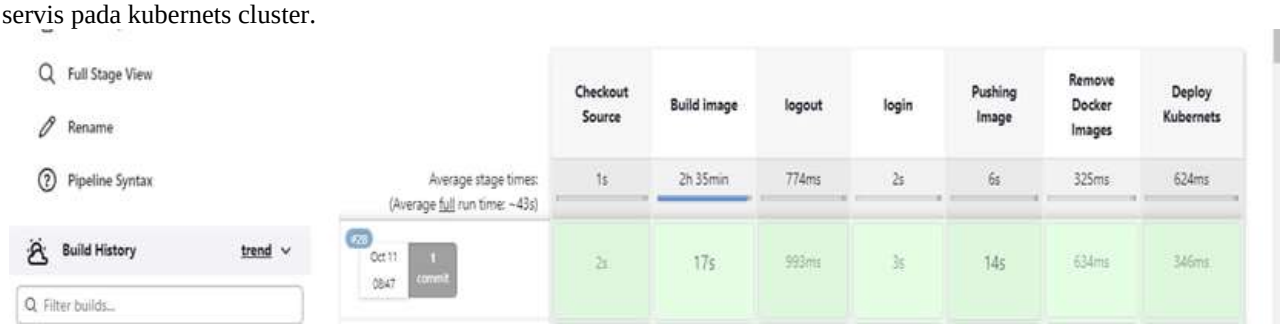
Berdasarkan penjelasan sebelumnya, berikut adalah hasil dari penelitian Implementasi Deployment Layanan *Website* Menggunakan Kubernetes Dengan Ci/Cd Jenkins.

3.1 Desain Infrastuktur

Dalam penelitian ini, diperlukan untuk menyiapkan beberapa kebutuhan, yaitu, server jenkins, dan Kubernetes cluster. Disini penulis menggunakan layanan cloud AWS sebagai penyedia server atau virtual machine.

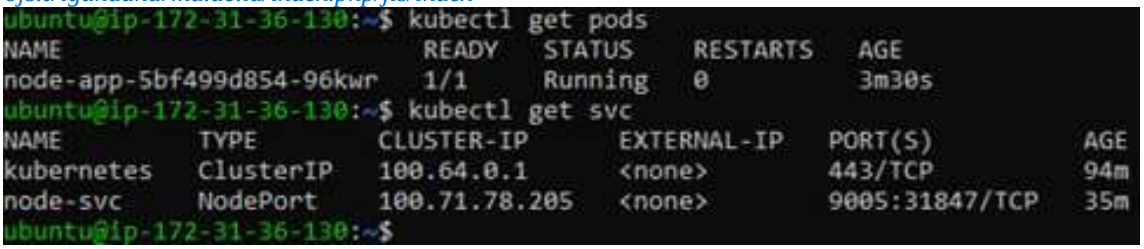
3.2 Deployment

Pada tahapan ini, aplikasi yang telah melewati proses pada server Jenkins. Dengan otomatis akan membuat pod dan servis pada kubernets cluster.



Gambar 2. Proses build hingga deployment dari Jenkins pipeline

Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)
Volume 23 ; Nomor 2 ; Agustus 2024 ; Page 290-296
E-ISSN : 2615-3475; P-ISSN : 1978-6603
https://ojs.trigunadharma.ac.id/index.php/jis/index



Gambar 3. pod dan service dari aplikasi

Gambar 4. Aplikasi berjalan secara native

Dengan mengintegrasikan aplikasi web dengan CI/CD pipeline dapat mempermudah proses deployment. Jika dibandingkan dengan metode deployment menggunakan cara manual, dengan menggunakan CI/CD pipeline proses deployment aplikasi berjalan secara otomatis dan lebih cepat seperti dilihat pada gambar 2 dan gambar 3, sedangkan proses *deployment* secara manual dan *native* seperti gambar 4.

3.3 Pengujian

Dalam proses pengujian, didapati proses deployment aplikasi menggunakan CI/CD untuk proses awalnya membutuhkan waktu 2 menit 50 detik, namun untuk proses selanjutnya dan seterusnya rata-rata waktu 42 detik dalam 35 kali percobaan. Proses deployment aplikasi menjadi lebih singkat dikarenakan jenkins sudah menyimpan riwayat proses sebelumnya dalam direktori temp job jenkins, sehingga ketika jenkins mengeksekusi pipeline yang sudah pernah dijalankan sebelumnya proses berikutnya tidak akan memakan waktu yang lebih lama daripada pertama kali mengeksekusi pipeline.

Sedangkan proses deployment menggunakan cara manual (tradisional) memakan waktu rata -rata 1 menit 50 detik dalam 5 kali percobaan. Metode manual memakan waktu yang lebih cepat pada awal proses deployment daripada metode CI/CD, namun memakan waktu yang lebih lama untuk proses selanjutnya. Ini terjadi karena beberapa faktor antara lain; proses testing secara manual, human error seperti kesalahan dalam proses mengunggah kode ke server, perbadaan versi perangkat lunak yang digunakan, dan lain-lain.

Dalam penelitian ini, menggunakan aplikasi web sederhana berbasis nodejs, sehingga waktu yang dibutuhkan dalam proses *deployment* bisa sangat cepat. Dalam perkembangannya, arsitektur aplikasi berbasis *microservices* mulai populer, sehingga waktu yang dibutuhkan dalam proses deployment bisa bertambah seiring dengan proses dan *environment* yang dibutuhkan. Jadi semakin banyak waktu yang dibutuhkan dalam proses *deployment* aplikasi, maka semakin besar pula selisih perbedaan waktu *deployment* aplikasi menggunakan CI/CD dengan metode tradisional (manual).

Berdasarkan hasil pengujian, didapatkan bahwa deployment aplikasi web secara continue menggunakan metode CI/CD lebih cepat dibandingkan dengan metode manual. Hal ini terjadi karena metode CI/CD mendefinisikan dan membuat standarisasi sebelum melakukan proses deployment aplikasi. Sehingga ketika proses deployment berjalan, hanya ada sedikit campur tangan manusia. Berbeda dengan metode tradisonal yang semua prosesnya berjalan dengan adanya campur tangan manusia dari awal sampai akhir.

3.4 Test

StressTesting aplikasi web yang telah berjalan, baik secara native maupun menggunakan kubernetes cluster. Test ini dilakukan sebanyak 10 kali dengan menggunakan sample 100, 200, 300, 400, 500, 600, 700, 800, 900, dan 1000 user dengan metode GET dari HTTP Request. Dari test tersebut diperoleh data sebagai berikut :

Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika dan Komputer)
Volume 23 ; Nomor 2 ; Agustus 2024 ; Page 290-296
E-ISSN : 2615-3475; P-ISSN : 1978-6603
https://ojs.trigunadharma.ac.id/index.php/jis/index

Tabel 1. Hasil Performance Test Native Menggunakan Jmeter

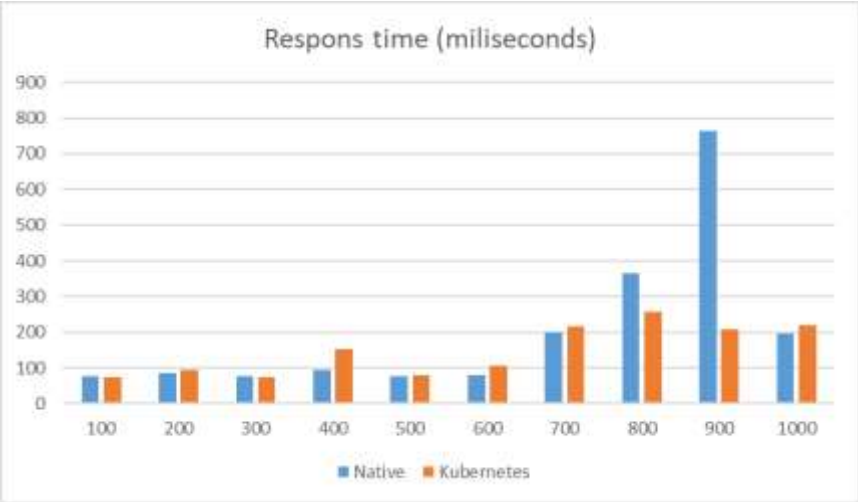
No	Sample	Native					
		Avrg	Min	Max	Std. dev	Error	THROUGHPUT
1	100	78	66	129	9,93	0,00%	94,3/SEC
2	200	84	66	383	43,94	0,00%	186,7/SEC
3	300	76	65	121	8,46	0,00%	304,9/SEC
4	400	93	65	1099	70,83	0,00%	2015,8/SEC
5	500	77	66	130	8,99	0,00%	463,8/SEC
6	600	79	66	151	10,5	0,00%	469,1/SEC
7	700	200	66	1082	76,68	0,00%	591,2/SEC
8	800	365	66	1252	222,82	0,00%	425,5/SEC
9	900	763	73	1889	357,32	0,00%	401,8/SEC
10	1000	197	67	1167	101,14	0,00%	503,0/SEC

Pada table 1 diatas, dilakukan pengujian sebanyak 10 kali secara berkala dari 100 *user* hingga 1000 user dengan *ramp-up* period 1sec pada aplikasi yang berjalan secara native. *Ramp-up* period adalah waktu yang dibutuhkan jmeter untuk menyelesaikan semua *request* yang dibuat. Pada tabel diatas, rata-rata *response* time tertinggi adalah 763ms (*miliseconds*) pada 900 *request* dengan *request* yang diproses setiap detiknya (*throughput*) adalah 401,8. Jadi waktu yang dibutuhkan aplikasi yang berjalan secara native untuk memproses 900 *reuquest* adalah 2,23 detik dengan rata-rata *response time* 763ms.

Tabel 1. Hasil *Performance Test* Kubernetes Menggunakan Jmeter.

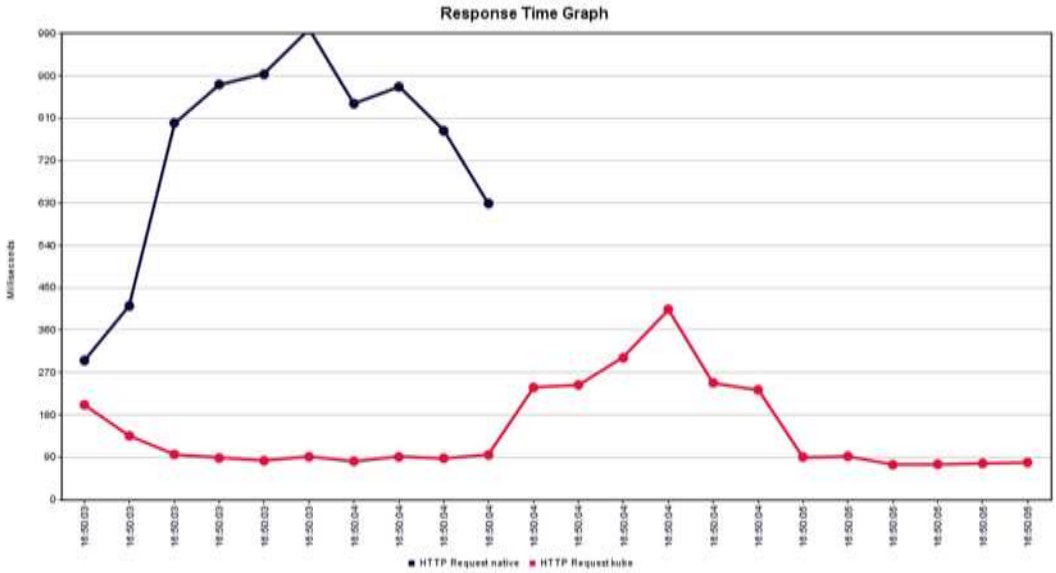
No	Sample	Kubernetes					
		Avrg	Min	Max	Std. dev	Error	Throughput
1	100	75	67	89	4,72	0,00%	92,3/sec
2	200	93	66	1086	91,29	0,00%	113,4/sec
3	300	75	65	86	4,30	0,00%	308,0/sec
4	400	152	66	352	92,51	0,00%	205,4/sec
5	500	79	67	107	6,59	0,00%	466,4/sec
6	600	106	67	233	45,42	0,00%	472,4/sec
7	700	217	77	1088	93,87	0,00%	441,1/sec
8	800	257	68	1209	184,15	0,00%	436,2/sec
9	900	207	66	1176	169,85	0,00%	339,1/sec
10	1000	218	72	1081	63,45	0,00%	502,0/sec

Pada tabel diatas, dilakukan pengujian sebanyak 10 kali secara berkala dari 100 user hingga 1000 user dengan ramp-up period 1sec pada aplikasi yang berjalan pada kubernetes *cluster*. Pada tabel diatas, rata-rata tertinggi response time adalah 257ms pada 800 request dengan dengan request yang diproses setiap detiknya (*throughput*) adalah 436,2. Jadi waktu yang dibutuhkan aplikasi yang berjalan pada kubernetes *cluster* untuk memproses 800 *reuquest* adalah 1,83 detik dengan rata-rata response time 257ms.



Gambar 5. Aplikasi berjalan secara native

Dari gambar 5 tersebut dapat disimpulkan bahwa response time tertinggi ada pada test dengan 900 user dengan rata-rata reponse time 763ms pada server aplikasi yang berjalan secara native, sedangkan pada kubernetes didapatkan reponse time 207ms. Namun secara durasi test , aplikasi yang berjalan secara native lebih cepat dibandingkan aplikasi yang menggunakan kubernetes cluster. Karena rata-rata request yang diproses setiap detiknya (*throughput*) pada aplikasi yang berjalan secara native lebih besar daripada aplikasi yang berjalan pada kubernetes *cluster*.



Gambar 6. Grafik Response Time 900 User

Bisa dilihat pada gambar 6 grafik diatas bahwa aplikasi yang berjalan secara native hanya membutuhkan waktu 2,23sec dengan response time 763miliseconds untuk menyelesaikan test 900 request dari user, sedangkan kubernetes memerlukan waktu 2,65sec dengan response time 207ms untuk menyelesaikan 900 request dari user.

4. KESIMPULAN

Berdasarkan penelitian yang telah diuraikan sebelumnya, dapat disimpulkan bahwa proses deployment aplikasi secara manual ternyata lebih cepat dibandingkan dengan menggunakan metode CI/CD dan Kubernetes. Namun, dalam hal pemeliharaan aplikasi web, metode CI/CD lebih memudahkan proses tersebut. Hasil uji performa menunjukkan bahwa aplikasi web yang berjalan secara native memberikan respons yang lebih cepat dibandingkan dengan yang berjalan di atas Kubernetes cluster. Meskipun demikian, aplikasi yang berjalan di Kubernetes cluster cenderung lebih stabil. Selain itu, infrastruktur ini dapat meminimalisir terjadinya kesalahan manusia karena prosesnya yang otomatis, serta membantu dalam perawatan aplikasi web. Dengan demikian, infrastruktur ini juga dapat dijadikan opsi yang baik dalam pengembangan perangkat lunak lainnya.

UCAPAN TERIMA KASIH

Ucapan syukur senantiasa tercurahkan kepa ALLAH SWT dan Ucapan terimakasih kepada: Kedua orang tua tercinta Bapak Nyaidi dan Ibu Napsiatun, Bapak M. Agung Nugroho, S.Kom. selaku dosen S1 Informatika, M.Kom, Bapak Rusydi Umar, S.T., M.T., Ph.D. selaku pembimbing Magister Teknik Informatika, dan teman-teman sekalian yang membantu berjalannya penelitian ini.

DAFTAR PUSTAKA

[1] I. G. A. Ahmad Farid, “Implementasi CI/CD Pipeline Pada Framework Androbase Menggunakan Jenkins (Studi Kasus: PT. Andromedia),” *Jurnal Nasional Komputasi dan Teknologi Informasi*, vol. 4, pp. 522-527, 2021.

[2] P. S. Rianto Hidayanto, “Performance Testing of e-Payment Website Using JMeter,” *International Research Journal of Advanced Engineering and Science*, pp. 350-352, 2019.

[3] M. A. Nugroho and S. Cuk, "ANALISIS CLUSTER CONTAINER PADA KUBERNETES DENGAN INFRASTRUKTUR GOOGLE CLOUD PLATFORM," *JIPI*, pp. 84-93, 2018.

[4] A. E. Y. H. W. B. B. Mohamad Septyan Asrofil, “DOCKER SALAH SATU PLATFORM YANG DIBANGUN BERDASARKAN TEKNOLOGI CONTAINER,” *journal.ittelkom-sby.ac.id*, pp. 145-153, 2020.

[5] K. A. S. Kamarudin, “Uji Kinerja Sistem Web Service Pembayaran Mahasiswa Menggunakan Apache JMeter (Studi Kasus: Universitas AMIKOM Yogyakarta),” *Jurnal Teknologi Informasi*, pp. 44-52, 2018.

[6] N. A. S. ,. A. D. L. Jaeni, "IMPLEMENTASI CONTINUOUS INTEGRATION/CONTINUOUS DELIVERY (CI/CD) PADA PERFORMANCE TESTING DEVOPS," *JURNAL OF INFORMATION SYSTEM MANAGEMENT*, pp. 62-66, 2022.

[7] M. A. A. Y. M. N. I. S. G. P. R. D. A. N. W. N. Desy Intan Permatasari, "Pengujian Aplikasi Menggunakan Metode Load Testing dengan Apache Jmeter pada Sistem Informasi Pertanian," *Jurnal Sistem dan Teknologi Informasi*, pp. 135-139, 2020.

[8] B. Bachina, "Dockerizing React App With NodeJS Backend," 13 Oktober 2022. [Online]. Available: https://medium.com/bb-tutorials-and-thoughts/dockerizing-react-app-with-nodejs-backend-26352561b0b7.

[9] E. C. Aneta Poniszewska-Mara´nda, "Kubernetes Cluster for Automating Software Production Environment," *sensors*, pp. 1-24, 2021.

[10] Creative Commons Attribution-ShareAlike 4.0, "User Handbook," 13 Oktober 2022. [Online]. Available: https://www.jenkins.io/doc/book/installing/linux/#debianubuntu.

[12] R. Syafi'i, D. Meilantika dan D. Herryanto, "MEMBANGUN WEBSITE DESA KARYA JAYA," *Jurnal Informatika dan Komputer (JIK)*, vol. 15, pp. 1-11, 2024.

[13] M. D. Sutarman, A. Fadli dan M. S. Aliim, "PERANCANGAN INFRASTRUKTUR DEVOPS SISTEM INFORMASI PUSAT HKI LPPM UNSOED," *TESLA*, vol. 25, pp. 59-71, 2023.

[14] H. H. R. M. Andrew Jeffery, "Rearchitecting Kubernetes for the Edge," *Association of Computing Machinery*, pp. 7-12, 2021.

[15] S. A. C. A. Ioannis Karamitsos, "Applying DevOps Practices of Continuous Automation for Machine Learning," *Journal Information*, vol. 11, no. 7, pp. 1-15, 2020.

[16] I. G. A. Ahmad Farid, "Implementasi CI/CD Pipeline Pada Framework Androbase Menggunakan Jenkins (Studi Kasus: PT. Andromedia)," *Jurnal Nasional Komputasi dan Teknologi Informasi*, vol. 4, pp. 522-527, 2021.

[17] L. A. Vayghan, M. . A. Saied, M. Toeroe dan F. Khendek, "A Kubernetes controller for managing the availability of elastic microservice based stateful applications," *The Journal of Systems & Software*, vol. 175, pp. 1-13, 2021.