E-ISSN: 2776-2343

Monitoring Kubernetes Cluster MenggunakanPrometheus dan Grafana

Salma Rachman Dira [1], Muhammad Arif Fadhly Ridha [2]
[1]*Program Studi Teknik Informatika, Politeknik Caltex Riau, Pekanbaru, Indonesia*
[2]*Program Studi Teknik Informatika, Politeknik Caltex Riau, Pekanbaru, Indonesia*

E-mail: [1)] salmadira@alumni.pcr.ac.id [2)] fadhly@pcr.ac.id

Abstract:*The large number of client requests can cause excessive workload on the server so that it can make the server down. Because it uses cloud computing services that can increase the number of client requests. With the cluster technology in cloud computing, the server workload will be divided evenly or balanced to each server. Clustering can be combined with containers where each process or application running on each container has the same kernel. Therefore, we need an application capable of managing containers, one of which is Kubernetes. Kubernetes has several components, namely clusters, pods, services and nodes that need to be monitored. To be able to carry out monitoring, an application that can help is used, namely Prometheus and Grafana. Prometheus will retrieve the data in Kubernetes, then the data that has been obtained by Prometheus can be visualized with Grafana. Grafana can convert metric data into graphs that are easy to understand and interactive. Based on the results of functionality testing that has been carried out, Prometheus has succeeded in reading data from the target server and Grafana has succeeded in displaying it in graphical form. In the test, from 5 trials of 20-100 user access, the monitoring system can show the amount of CPU usage, memory, and traffic load that can continue to increase according to the user's access load.*
Keywords: *Cloud Computing, Cluster, Container, Kubernetes, Prometheus, Grafana, CPU, Memory, Traffic.*

Abstrak: Banyaknya jumlah *request client* dapat menyebabkan beban kerja berlebih pada server sehingga dapat membuat server *down.* Karna hal itu digunakan layanan cloud computing yang dapat meningkatkan jumlah *request client*. Dengan adanya teknologi *cluster* pada cloud computing maka beban kerja server akan dibagi secara merata atau seimbang kepada masing-masing server. *Clustering* bisa dikombinasikan dengan container dimana tiap proses atau aplikasi yang dijalankan tiap container memiliki kernel yang sama. Oleh karna itu dibutuhkan sebuah aplikasi yang mampu melakukan management terhadap container, salah satu aplikasinya adalah Kubernetes. Kubernetes memiliki beberapa komponen yaitu *cluster, pod, service* dan *node* yang perlu di *monitoring*. Untuk dapat melakukan *monitoring* digunakan sebuah aplikasi yang dapat membantu yaitu Prometheus dan Grafana. Prometheus akan mengambil data yang ada pada Kubernetes, lalu data yang telah didapat oleh Prometheus dapat divisualisasikan dengan Grafana. Grafana dapat merubah data metrik ke dalam bentuk grafik yang mudah dipahami dan interaktif. Berdasarkan hasil pengujian fungsionalitas yang telah dilakukan, Prometheus berhasil membaca data dari server target dan Grafana berhasil menampilkannya ke dalam bentuk grafik. Pada pengujiannya dari 5 percobaan akses 20-100 user, sistem monitoring dapat menampilkan besar penggunaan *CPU*, *memory*, dan beban *traffic* yang dapat terus meningkat sesuai dengan beban akses *user*.
Kata Kunci: *Cloud Computing, Cluster, Container, Kubernetes, Prometheus, Grafana, CPU, Memory, Traffic*

1. PENDAHULUAN

Dengan banyaknya tuntutan layanan yang dapat diakses setiap saat serta meningkatnya jumlah *request client* dapat menyebabkan beban kerja yang berlebih pada server hingga dapat mengakibatkan server *down*. Hal ini disebabkan karena server masih menggunakan komputasi konvensional sehingga kinerja server menjadi berat. Salah satu jenis dari komputasi modern adalah *cloud computing*. Cloud computing merupakan teknologi yang menjadikan internet sebagai pusat server untuk mengelola data dan juga aplikasi pengguna [1]. Selain itu, *cloud computing* memudahkan penggunanya untuk menjalankan program tanpa harus menginstall aplikasi terlebih dahulu dan mengakses data dan informasi melalui internet. Penggunaan teknologi *cluster* dapat mengurangi beban kerja server. *Cluster computing* adalah teknologi yang memanfaatkan beberapa sumber daya komputer untuk bekerja secara bersamaan, sehingga terlihat seperti satu sistem yang saling terintegrasi [2]. *Clustering computing* dapat dikombinasikan dengan *container*, *container* adalah virtualisasi pada level sistem operasi dimana tiap proses atau aplikasi yang dijalankan tiap *container* memiliki kernel yang sama. Oleh karena itu dibutuhkan sebuah aplikasi yang mampu melakukan management terhadap *container*, salah satu aplikasi kubernetes. Kubernetes adalah *platform* open *source* untuk mengelola kumpulan kontainer dalam suatu *cluster* server [3].

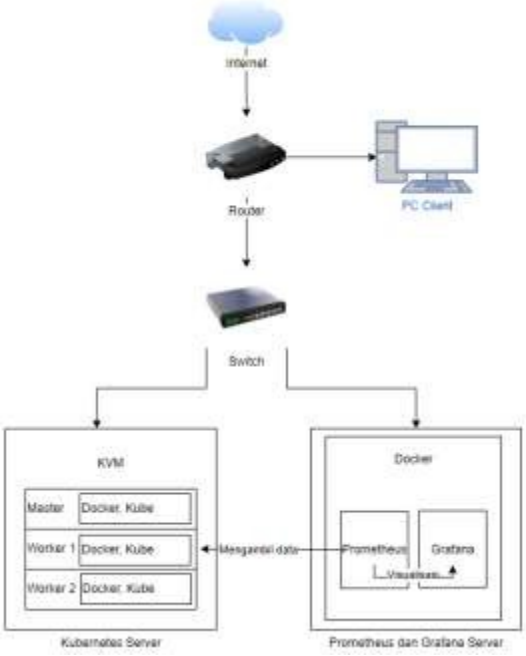
Kubernetes dapat menyederhanakan penerapan aplikasi dalam wadah dan di seluruh cloud, tetapi kubernetes memiliki kompleksitas dimana terdapat banyaknya komponen yang dianalisis, dan kebutuhan untuk mempertahankan beban pemeliharaan yang cukup rendah oleh seorang administrator. Kubernetes juga bisa mengalami masalah seperti beban *traffic* yang besar, kelebihan dalam penggunaan *CPU*, dan kelebihan penggunaan memory yang dapat memperlambat kinerja kubernetes dalam memenuhi permintaan *client* dan bahkan dapat mematikan layanan dari kubernetes tersebut. Untuk itu, kubernetes harus selalu di-monitoring untuk memantau status dari kondisi kubernetes dan memastikan sumber daya kubernetes dalam keadaan stabil. Dalam melakukan *monitoring* kinerja aplikasi yang ada pada cluster Kubernetes, penting untuk memeriksa kinerja container, pod, dan layanan, serta karakteristik cluster secara keseluruhan. Dengan memberikan informasi tentang penggunaan sumber daya aplikasi, *monitoring* kubernetes memungkinkan untuk mengukur kinerja aplikasi untuk mendeteksi dan memantau layanan. Selain itu, administrator dapat mengetahui apakah semua node berfungsi dengan baik dan pada kapasitas apa serta berapa banyak aplikasi yang berjalan pada setiap node dan pemanfaatan sumber daya seluruh *cluster*. Kubernetes tidak memiliki fitur seperti *monitoring* kondisi *Cluster*, *Pod, Service, Node* yang berjalan di Kubernetes yang dapat mudah dibaca atau dimengerti oleh seorang Administrator [4]. Salah satu solusi untuk mengatasi hal tersebut adalah dengan menggunakan sebuah aplikasi yang membantu *monitoring* Kubernetes yang disebut Prometheus dan Grafana.

Prometheus merupakan aplikasi *open-source* yang berguna sebagai pemantauan ketersediaan dan performa jaringan komputer. Prometheus dapat menghasilkan data metrik berupa data *resource* server. Data yang telah didapat oleh Prometheus dapat divisualisasikan dengan aplikasi tambahan. Salah satu aplikasi yang dapat digunakan sebagai visualisasi data metrik dari pemantau jaringan adalah Grafana. Grafana adalah sebuah software *open source* yang membaca sebuah data metrics untuk dibuat menjadi sebuah grafik atau sebuah data tertulis [5]. Grafana sering digunakan untuk melakukan analisis data dan *monitoring*. Dengan menggunakan grafana dapat memudahkan untuk meminta, memvisualisasikan, mengatur peringatan dan memahami data dengan bantuan metric. Grafana juga dapat menampilkan *traffic* jaringan secara *real time* sehingga dapat dipastikan bahwa data yang didapat adalah kondisi server saat ini. Prometheus dan Grafana dapat membantu membuat *system monitoring* jaringan yang mudah dimengerti dan dapat membantu seorang *system* administrator. Salah satu keunggulan Grafana adalah grafana sangat cocok untuk membuat dashboard yang dinamis dengan berbagai menu bawaan. Grafana juga memiliki dashboard template yang bisa digunakan untuk mengumpulkan variabel data yang digunakan [6].

Berdasarkan uraian diatas, maka dilakukan penelitian "*Monitoring* Kubernetes *Cluster* Menggunakan Prometheus Dan Grafana".

1. METODE PENGUJIAN

Adapun dalam melakukan implementasi pada penelitian ini, menggunakan 2 buah server yaitu server Kubernetes dan server Prometheus dan Grafana. Ada pun topologi yang digunakan seperti Gambar 1



Gambar 1 Topologi Jaringan

Dari topology diatas, Prometheus akan mengambil data metrik dari server kubernetes dan data metrik tersebut akan dikirim ke Grafana agar dapat divisualisasikan ke dalam bentuk grafik.

Metode pengujian pada proyek akhir ini dilakukan dengan beberapa cara yakni sebagai berikut:

A. Pengujian Fungsional

Pengujian fungsional bertujuan untuk menguji sistem, dengan memberikan masukan yang sesuai, memverifikasi keluaran terhadap persyaratan fungsional. Pengujian ini memeriksa antarmuka pengguna, API, Basis Data, dan komunikasi klien/server [7]. Pengujian fungsional pada prometheus dan grafana dilakukan dengan cara:

1) Pengujian pertama dilakukan dengan mengakses dashboard prometheus untuk memastikan apakah prometheus sudah berhasil di install dengan menggunakan ip dari server prometheus.
2) Pengecekan target server untuk melihat apakah sudah terkoneksi dengan server monitoring yang sebelumnya sudah di buat di dalam file konfigurasi prometheus yang bertipe yml.
3) Pengecekan data metrik yang nanti gunanya untuk melihat sumber daya komputer server apakah sudah berhasil didapatkan dari target server dengan cara mengekspose matrik target server.
4) Melakukan query sederhana pada dashboard prometheus untuk melihat grafik penggunaan *CPU* dan *Memory*.
5) Pada grafana dilakukan pengecekan dengan membuka panel grafik yang dibutuhkan untuk pengujian.

B. Pengujian *Performance*

Pengujian fungsional bertujuan untuk menguji sistem, dengan memberikan masukan yang sesuai, memverifikasi keluaran terhadap persyaratan fungsional. Pengujian ini memeriksa antarmuka pengguna, API, Basis Data, dan komunikasi klien/server[8]. Pengujian fungsional pada prometheus dan grafana dilakukan dengan cara:

1) Pengujian pertama dilakukan dengan mengakses dashboard prometheus untuk memastikan apakah prometheus sudah berhasil di install dengan menggunakan ip dari server prometheus.
2) Pengecekan target server untuk melihat apakah sudah terkoneksi dengan server monitoring yang sebelumnya sudah di buat di dalam file konfigurasi prometheus yang bertipe yml.
3) Pengecekan data metrik yang nanti gunanya untuk melihat sumber daya komputer server apakah sudah berhasil didapatkan dari target server dengan cara mengekspose matrik target server.

4) Melakukan query sederhana pada dashboard prometheus untuk melihat grafik penggunaan *CPU* dan *Memory*.
5) Pada grafana dilakukan pengecekan dengan membuka panel grafik yang dibutuhkan untuk pengujian.

2. HASIL DAN PEMBAHASAN

3.1 Hasil dan Analisis Pengujian Fungsionalitas

Setelah melakukan pengujian fungsional terhadap masing-masing fungsi dari sistem monitoring. Hasil pengujian dapat dilihat pada Tabel 1

Tabel 1 Hasil pengujian fungsional

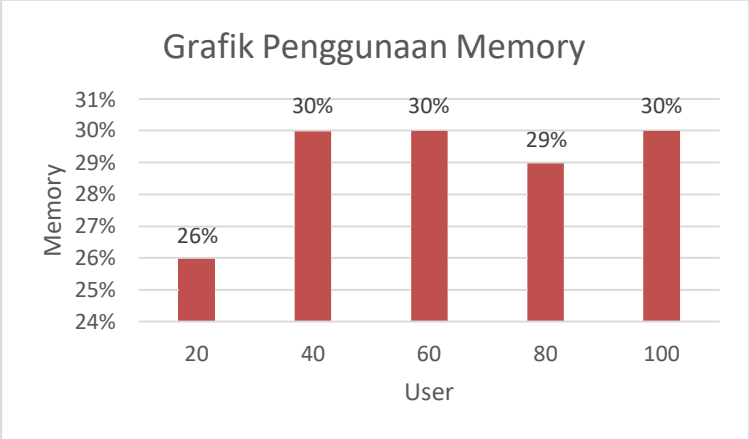
Pengujian	Antarmuka	API	Basis Data	Komunikasi klien/server
Akses Dashboard	□	□	□	□
Server Target	□	□	□	□
Data Metrik	□	□	□	□
Query	□	□	□	□
Panel Grafik	□	□	□	□

Berdasarkan pada tabel dapat disimpulkan bahwa pengujian pada masing-masing fungsi berjalan dengan baik sistem dapat menampilkan dashboard saat diakses, api, basis data dan komunikasi juga berjalan baik ini dibuktikan dengan dapat ditemukannya server target dan dapat mengambil data metrik dari target serta sistem dapat menampilkan grafik baik melalui query sederhana maupun panel grafik yang diimport pada grafana.

3.2 Hasil dan Analisis Pengujian *Performance*

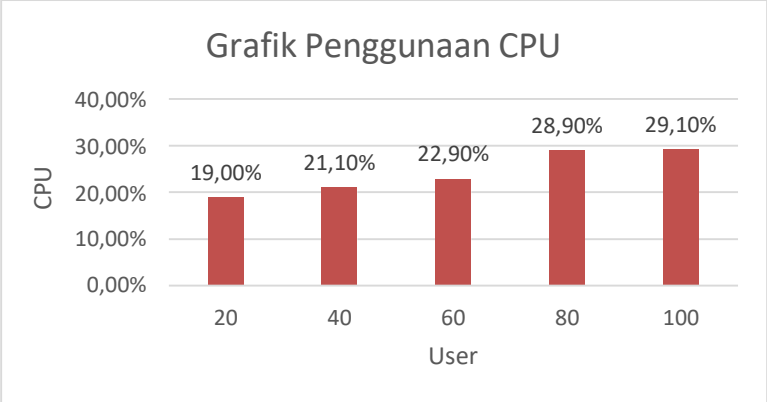
Pengujian yang pertama yaitu melakukan simulasi akses beberapa *user* ke *Web Server* selama 15 detik yang bertujuan untuk melihat penggunaan CPU dan *memory* berdasarkan jumlah *user* yang akses pada *Web Server* menggunakan aplikasi JMeter. Hasil dari pengujian tersebut dapat disimpulkan beberapa hal yaitu penggunaan CPU, *memory*, ketersediaan memory, dan *traffic* jaringan dapat meningkat atau menurun berdasarkan jumlah *user* yang akses.

Terlihat pada Gambar 2 bahwa rata-rata penggunaan *memory* berdasarkan 5 tahap pengujian adalah 29%. Pada akses 20 *user* menunjukkan penggunaan *memory* sebesar 26% dan meningkat hingga akses 100 *user* menjadi 30%. Dengan ini dapat disimpulkan bahwa sistem *monitoring* dapat menampilkan penggunaan *memory* yang dapat meningkat dan menurun berdasarkan beban akses *user*



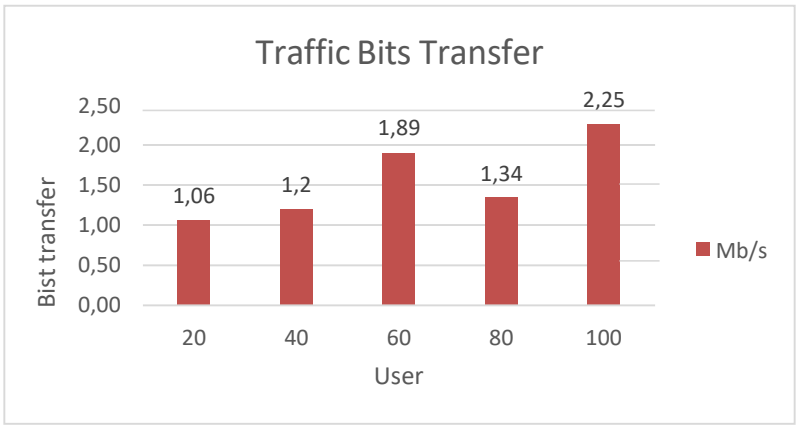
Gambar 2 Grafik penggunaan memory

Analisa selanjutnya adalah besar penggunaan CPU. Pada Gambar 4.34 terlihat bahwa 5 pengujian menunjukkan tingkat kenaikan penggunaan CPU yang signifikan berdasarkan beban akses *user*. Pada akses 20 *user*, penggunaan CPU adalah sebesar 19.0% dan meningkat menjadi 29.1% pada akses 100 *user*. Rata-rata penggunaan CPU pada 5 percobaan adalah 24,2%. Maka berdasarkan hasil dari pengujian penggunaan CPU ini dapat disimpulkan bahwa sistem *monitoring server* berhasil menampilkan penggunaan CPU yang dapat meningkat sesuai dengan beban akses pada *Web Server*.



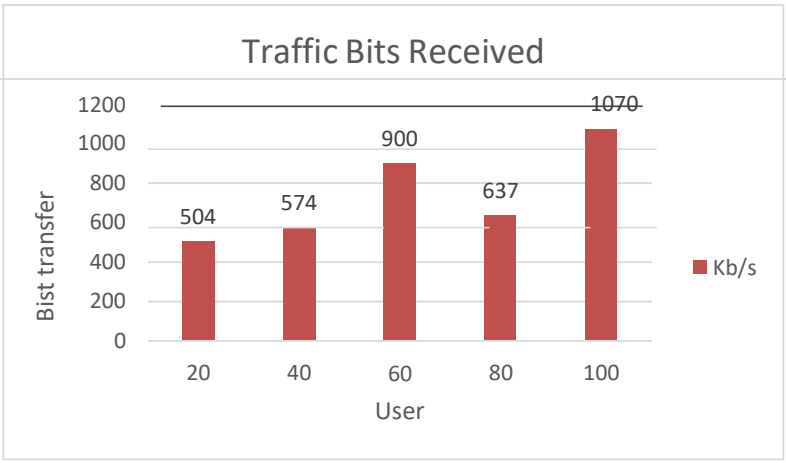
Gambar 3 Grafik penggunaan CPU

Analisa selanjutnya adalah besar paket yang dikirim oleh *user*. pada Gambar 3 terlihat bahwa 5 pengujian menunjukkan tingkat kenaikan paket yang dikirim berdasarkan aktivitas *user* pada *Web Server* selama 15 detik. Grafik pada *bits transfer* terus meningkat sesuai dengan beban yang diberikan yaitu dari 1,06Mb/s pada akses 20 *user* menjadi 2,25Mb/s pada akses 100 *user*. Maka dapat disimpulkan bahwa sistem *monitoring server* berhasil menampilkan paket yang terkirim berdasarkan aktivitas *user* dan dapat meningkat dan menurun sesuai dengan beban yang diberikan.



Gambar 4 Grafik bits transfer

Analisa selanjutnya adalah besar paket yang diterima. Terlihat pada Gambar 4 bahwa jumlah paket yang diterima meningkat sesuai dengan aktivitas *user* selama 15 detik. Pada awalnya akses 20 *user* hanya menerima paket sebesar 504Kb/s dan meningkat hingga 1070Kb/s pada akses 100 *user*. Berdasarkan dari data yang dapat maka dapat disimpulkan bahwa sistem *monitoring server* dapat menampilkan besar paket yang diterima berdasarkan banyaknya aktivitas *user* dan dapat meningkat serta menurun sesuai dengan beban yang diberikan.



Gambar 5 Grafik bits received

Pengujian yang kedua adalah dengan melakukan simulasi akses *user* sebanyak 5 kali yaitu 200 *user*, 400 *user*, 600 *user*, 800 *user*, dan 1000 *user*. Pengujian ini bertujuan untuk menentukan jumlah maksimum *user* yang dapat akses pada *Web Server* dan seberapa besar *error* yang terjadi berdasarkan *sample* aktivitas *user*. Pada setiap simulasi, *user* mengakses *Web Server* selama 2 menit. Hasil yang didapat dari pengujian kedua ini dapat dilihat pada Tabel 2

Tabel 2 Hasil pengujian eror

User	**200**	**400**	**600**	**800**	1000
Memory	26%	27%	28%	28%	29%
CPU	43,4%	47,3%	49,5%	45,5%	47,0%
Samples	345268	290904	281644	265872	255679
Error	0%	0%	0,29%	0,59%	1,09%
User	**200**	**400**	**600**	**800**	1000

Berdasarkan Tabel 4.13, sistem *monitoring server* menunjukkan penggunaan *memory* pada *Web Server* terus meningkat seiring dengan bertambahnya jumlah *user*. Dari 200 *user* hingga 1000 *user*, penggunaan *memory* bertambah sebanyak 3%. Rata-rata penggunaan *memory* dari 5 percobaan adalah 27.6%. Sedangkan penggunaan CPU dapat dikatakan tidak stabil yaitu turun sebesar 4% pada akses 800 *user* dan naik kembali sebesar 1.5% pada akses 1000 *user*. Rata-rata penggunaan CPU dari akses 200-1000 user adalah 46.54%.

Selanjutnya adalah tingkat *error* yang diterima oleh *Web Server* selama percobaan akses *user*. Pada akses 200 user, jumlah permintaan adalah 345268 *samples* dan tingkat *error* adalah sebesar 0%. Sedangkan pada akses 400 user, jumlah permintaan turun drastis yaitu sebesar 290904 *samples* dan tingkat *error* sebesar 0%. Penurunan jumlah *samples* dapat terjadi akibat dari koneksi internet yang tidak stabil. Pada percobaan selanjutnya, tingkat *error* pada akses 600 user meningkat menjadi 0.29%. Namun tingkat *error* pada akses 600-1000 *user* dapat dikatakan stabil yaitu *error* meningkat sebanyak 0,8%. Dari hasil analisa ini dapat disimpulkan *Web Server* dapat berjalan dengan baik saat akses 200 *user* dan 400 user selama 2 menit karena tidak terdapat error, namun dapat menangani lebih dari 200 dan 400 *user* meski kinerjanya menjadi lambat dan tingkat *error* yang semakin naik. Tingkat eror tersebut dapat disebabkan oleh koneksi internet yang kurang stabil.

3. Kesimpulan

Dari hasil pengujian dan analisis yang telah dilakukan, maka dapat diambil beberapa kesimpulan sebagai berikut:.

1) Sistem *monitoring server* Kubernetes berhasil diimplementasikan menggunakan Prometheus sebagai pembaca data metrik dan Grafana sebagai visualisasi data ke dalam bentuk grafik.
2) Sistem *monitoring server* Kubernetes dapat diakses pada *dashboard monitoring* Grafana

3) Penggunaan monitoring sesuai dengan yang di harapkan setelah dilakukan pengujian, sistem mendapatkan output yaitu perubahan grafik pada panel monitoring seperti grafik penggunaan CPU, Memory dan bandwith jaringan.
4) Pada pengujian *performance* kenaikan dan penurunan CPU, memory dan traffic terjadi seiring dengan dinaikannya user yang melakukan request. Koneksi yang stabil merupakan salah satu faktor terpenting dalam melakukan pengujian.

Ucapan Terima Kasih

Pada kesempatan ini, penulis ingin mengucapkan terima kasih kepada pihak yang telah banyak memberikan bantuan dan dukungan yang tiada terhingga baik secara langsung maupun tidak langsung. Ucapan terima kasih tersebut penulis tujukan kepada:

1. Allah Azza Wa Jalla atas rahmat, kemudahan dan berkah yang telah dilimpahkan-Nya, sehingga penulis bisa menyelesaikan tugas akhir ini tepat waktu
2. Kedua orang tua penulis, ibu dan ayah yang selalu mendo'akan serta dukungan dan kasih sayang nya, sehingga penulis bisa menyelesaikan tugas akhir tepat waktu.
3. Dr. Dadang Syarif Sihabudin Sahid, S.Si., M.Sc. selaku Direktur Politeknik Caltex Riau yang telah memberikan dukungan moral dalam menyelesaikan proyek akhir ini.
4. Ibu Kartina Diah Kusuma Wardhani, S.T., M.T. selaku Ketua Program Studi Teknik Informatika yang telah memberikan izin untuk menyelesaikan proyek akhir.
5. Ibu Puja Hanifah, S.S.T., M.MSI. selaku Koordinator Proyek Akhir yang telah membantu dalam menyelesaikan proyek akhir ini.
6. Bapak Muhammad Arif Fadhly Ridha, S.Kom., M.T selaku dosen pembimbing yang telah memberikan ilmu dan bimbingan dengan penuh kesabaran kepada penulis dalam menyelesaikan proyek akhir.
7. Bapak Erzi Hidayat, S.T., M.Kom. dan Bapak Ibnu Surya, S.T., M.T selaku dosen penguji, yang telah memberikan masukan dan saran dalam menyelesaikan proyek akhir.
8. Seluruh dosen khususnya Program Studi Teknik Informatika dan seluruh dosen di Politeknik Caltex Riau pada umumnya yang telah memberikan bekal ilmu kepada penulis dalam menyelesaikan proyek akhir.

Rujukan

[1] appdynamics. (n.d.). *Kubernetes monitoring: how to monitor using best practices*. Retrieved from appdynamics: https://www.appdynamics.com/topics/how-to-monitor-kubernetes-best-practices
[2] apridayanti, s., isnawaty, & saputra, r. A. (2018). Desain dan implementasi viirtualisasi berbasis docker untuk deployment aplikasi web.
[3] armipajasa, g. (2021). Implementasi monitoring server dengan prometheus dan grafana.
[4] dwiyatno, s., rakhmat, e., & gustiawan, o. (2020). Implementasi virtualisasi server berbasis docker container. *Jurnal prosisko*.
[5] fachrizal, a. A. (2017, april 16). *Grafana, software opensource yang powerful untuk analisis data dan monitoring*. Retrieved from tncdigitalmedia: https://tncdigitalmedia.com/perangkat-lunak/open- source/grafana-software-opensource-yang-powerful-untuk-analisis-data-dan-monitoring
[6] febriana, r. M. (n.d.). Implementasi sistem monitoring menggunakan prometheus dan grafana. 1-5.
[7] mell, p., & grance, t. (2011). *The nist definition of cloud computing recommendations of the national institute of standards and technology*. Retrieved from https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf