Volume 5 No. 1 Mei 2024
ISSN 2747-1799

Implementasi Load Balancing pada Web Server Berbasis Container dalam Cluster Kubernetes pada PT Mandiri Utama Finance

Samson Sinaga[1], Irwan Agus Sobari[2]

[1,2]Universitas Nusa Mandiri
Jl. Kramat Raya No.18, Kota Jakarta Pusat, Indonesia
Email : [1]samsonsinaga30@gmail.com, [2]irwan.igb@nusamandiri.ac.id

Abstrak - PT Mandiri Utama Finance dalam operationalnya mengandalkan aplikasi berbasis web. Akan tetapi, masalah sering terjadi ketika diakses secara bersamaan dengan request yang besar terutama di akhir bulan sehingga kadang aplikasi menjadi lambat dan susah diakses bahkan bisa request timeout. Sedangkan sistem yang digunakan berbasis monolitik yang dimana aplikasi berada di dalam satu server ketika ada salah satu servis yang down aplikasi lain terdampak serta sumber daya yang dibutuhkan begitu besar. Dibutuhkan teknologi yang tepat untuk mewujudkan sistem web hosting yang high availability sehingga tidak mengganggu ketika diakses dan aplikasi berjalan dengan baik. Penggunaan web server berbasis container di dalam Kubernetes cluster merupakan arsitektur berbasis microservices. Teknologi ini relatif baru dan cepat perkembangannya. Ditambah dengan penggunaan load balancer external seperti HAProxy membuat sistem ini semakin reliable dalam pembagian request. Sistem container didalam cluster Kubernetes memiliki node masing-masing sehingga ketika ada service yang down aplikasi lain tidak terdampak dan pemanfaatan load balancer membuat pembagian request secara merata ke server web membuat aplikasi tetap stabil diakses ketika diakses secara Bersama -sama dengan request yang besar

Kata Kunci: ***High Availability, container, web server, load balancer***

Abstract *- PT Mandiri Utama Finance in its operations relies on web-based applications. However, problems often occur when accessed simultaneously with large requests especially at the end of the month so sometimes the application becomes slow and difficult to access even can request timeout. While the system used is based on monolithic which where the application is in one server when there is one of the services that the other application is affected and the resources required are so large. It takes the right technology to create a high availability web hosting system so that it does not interfere when accessed and the application runs well. The use of container-based web servers within the Kubernetes cluster is a microservices-based architecture. This technology is relatively new and rapidly developing. In addition to the use of external load balancers such as HAProxy, the system is more reliable in the division of requests. The container system within the Kubernetes cluter has their respective nodes so that when there is a service that other applications down are not affected and the use of load balancer makes the distribution of requests evenly to the web server making the application stable accessed when accessed together -as with a large request.*

Keywords: High Availability, container, web server, load balancer

PENDAHULUAN

Meningkatnya kebutuhan informasi pada internet menyebabkan beban traffic data pada server aplikasi semakin banyak akibat terlalu banyaknya request. Oleh karena itu dapat menyebabkan beban kerja pada suatu layanan baik secara web ataupun aplikasi mengalami kelebihan beban (request), sehingga server tersebut tidak bisa diakses atau down dikarenakan data yang overload. Permasalahan yang diakibatkan karena beban berlebih dalam suatu server dapat di tanggulangi dengan konsep clustering, yaitu sistem teknologi yang menggabungkan beberapa server untuk bekerja bersamaan yang merupakan seperti satu system tunggal. "Menurut Sumbogo dalam pengaplikasian clustering server perlu menggunakan metode load balancing agar beban traffic di setiap masing-masing server dapat berjalan secara optimal (Sumbogo, Data, & Siregar, 2018).

Adapun perhatian terhadap Cloud Computing juga semakin meningkat. Banyak dari perusahaan teknologi yang mengembangkan Xen, HyperV, VMware, vShepere, KVM, dan lain-lain yang dikenal sebagai teknologi virtualisasi.Teknik virtualisasi sendiri adalah teknik dimana mengisolasi suatu system sehingga tidak mengganggu system yang lainnya. "Salah satu teknik virtualisasi adalah berbasis container (Kusuma Hakim, Kun Riyanto, & Fauzan, 2019). "Virtualisasi berbasis container adalah teknik yang tidak menerapkan Hypervisor yang hanya mengisolasi proses tanpa mengisolasi perangkat keras, kernel dan operating system sehingga dapat mengurangi overhead pada perangkat keras juga performanya lebih baik dari virtualisasi mesin (Dwiyatno, Rakhmat, & Gustiawan, 2020). PT Mandiri Utama Finance (MUF) yang merupakan

perusahaan pembiayaan untuk memenuhi berbagai kebutuhan masyarakat, antara lain kepemilikan kendaraan bermotor baik baru dan bekas, pembiayaan multiguna, pembiayaan syariah dan pembiayaan fleet. Untuk mendukung kinerja perusahaan di era saat ini maka Mandiri Utama Finance menggunakan otomatisasi sistem. Sehingga dibutuhkan aplikasi yang bisa diakses untuk operasional. Disisi lain untuk menunjang agar aplikasi yang digunakan berjalan dengan baik maka, dibutuhkan juga server aplikasi yang berjalan dengan optimal sehingga Ketika aplikasi diakses tidak terjadi beban berlebih atau overload

METODE PENELITIAN

A. Teknik Pengumpulan Data

1. Observasi
   Cara ini dilakukan dengan mengamati secara langsung proses sistem di PT Mandiri Utama Finance guna mendapatkan data-data dan melihat aplikasi dengan cara mengakses aplikasi ataupun melihat kondisi aplikasi pada saat jam operasional ataupun closingan dan permasalahan lainnya
2 Wawancara
   Dalam pengumpulan data bersifat wawancara penulis mencari informasi dengan berbicara kepada staff dan kepala seksi bagian IT infrastruktur yang bertugas untuk menangani, memantau dan mengoptimalkan aplikasi yaitu kepada bapak Asdi Rosadillah dan Bapak Muhammad Iqbal Selaku Section Head dan juga Trio Alfianto selaku Staff Infrastruktur.
3. Studi Pustaka
   Untuk mendapatkan referensi yang sesuai dengan topik maka penulis mencari referensi dari jurnal-jurnal terkait yang ada sebagai kelengkapan dalam penulisan yang masih berhubungan dengan web server dan *load balancing*.

B. Analisa Penelitian
   Adapun setelah semua data terkumpul maka dilakukan Analisa bagaimana penelitian ataupun perancangan tersebut akan dibuat, diujikan dan diimplementasikan .

1. Analisa Kebutuhan
   Dalam Analisa ini dibutuhkan beberapa perangkat keras maupun perangkat lunak antara lain : Laptop ataupun komputer yang akan dijadikan virtualisasi, minikube dari Kubernetes.
2. Desain
   Melakukan perancangan sistem yang dibutuhkan berdasarkan hasil analisis. Merubah sistem dengan metode yang berbeda dari monolitik ke *microservices* dengan *load balancing*nya.
3. *Testing*
   Melakukan simulasi terhadap model sistem yang dibuat sebelumnya. Dimana pada metode ini akan dilakukan pengujian beban kerja pada web.
4. Implementasi
   Dalam pengimplementasian analisa ini dilakukan pada sebuah laptop yang nantinya jika sudah sesuai yang diharapkan akan di implementasikan ke PT. Mandiri Utama Finance.

C. Virtualisasi Server

Sederhananya, virtualisasi adalah lawan kata dari mesin fisik. Komputer fisik adalah server yang terdiri dari berbagai bagian, seperti *power supply*, *mainboard*, *RAM*, *disk*, dan lainnya. Meskipun aplikasi virtualisasi tampaknya berjalan sendirian pada satu mesin, sebenarnya virtualisasi berjalan bersama dengan aplikasi lain di mesin lain (Khalida, Muhajirin, & Setiawati, 2019). Adapun terdapat tiga jenis metode pendekatan virtualisasi dalam membangun *server* virtual yaitu (Zaida Efrizal, 2014).

1. *Partial Virtualization* adalah jenis virtualisasi yang terdapat pada beberapa komponen perangkat keras. Perangkat lunak pda virtualisasi parsial dibuat seakan - akan alat tersebut berada di perangkat kita.
2. *Full Virtualization* adalah virutalisasi yang berarti membuat seolah-olah terdapat komputer lain di dalam komputer. Dengan menginstall Linux dalam Windows Anda, demikian juga menginstall Windows dalam Linux.
3. *Hardware-assisted Virtualisation* adalah virtualisasi yang didukung oleh hardware, sehingga ada hardware khusus yang bermanfaat untuk meningkatkan kinerja proses virtualisasi. Virtualisasi yang dibantu hardware memiliki biaya yang tinggi, jadi untuk menjaga skalabilitas sistem operasi pengguna tidak terlalu rendah,, maka dibantu dengan hardware.

D. *Cloud Computing*

Definisi sederhana dari komputasi awan atau *cloud computing* yaitu menyimpan dan mengakses program dan data melalui Internet dari lokasi atau komputer yang jauh, bukan dari *hard drive* komputer kita. Lokasi jauh ini berbeda dari mesin jarak jauh biasa karena memiliki fitur seperti skalabilitas dan elastisitas. Jika kita menyimpan data atau menjalankan program dari *hard drive* komputer lokal itu disebut penyimpanan dan komputasi lokal. Awan hanyalah metafora untuk Internet. Kita perlu mengakses program atau data kita melalui Internet agar dapat dianggap komputasi awan. Komputasi awan dapat dilakukan di mana saja dan kapan saja dengan perangkat yang terhubung ke internet.
Cloud computing memiliki tiga jenis layanan yang berbeda yang tidak dapat diatur langsung oleh pengguna yaitu adalah (Dr Joseph Teguh Santoso S.Kom, 2023) :

1. *Infrastructure as a Service* (IaaS). Konsumen tidak dapat melakukan pengelolaan ataupun mengontrol infrastruktur cloud yang mendasar,

tetapi mereka memiliki kontrol atas sistem operasi, penyimpanan, dan aplikasi yang diterapkan. Mereka juga dapat menyediakan pemrosesan, penyimpanan, jaringan, dan sumber daya komputasi penting lainnya yang dibayar per penggunaan. Selain memiliki peralatan, penyedia layanan bertanggung jawab atas perumahan, pendinginan, dan pemeliharaan. Salah satu penyedia IaaS populer adalah Amazon Web Services (AWS).

2. *Platform as a Services* (PaaS). Konsumen dapat menyebarkan aplikasi yang dibuat atau diperoleh konsumen ke infrastruktur cloud menggunakan bahasa pemrograman, perpustakaan, layanan, dan alat yang didukung oleh penyedia. Konsumen tidak dapat mengelola atau mengontrol infrastruktur cloud yang mendasarinya, tetapi mereka bertanggung jawab atas aplikasi yang digunakan dan dapat mengatur konfigurasi lingkungan hosting aplikasi. Dengan kata lain, ini adalah kerangka kerja operasi atau pengembangan yang telah dikemas dan siap untuk digunakan. Jaringan, server, dan penyimpanan disertai dengan manajemen pemeliharaan dan skalabilitas oleh vendor PaaS. Klien biasanya membayar untuk layanan tersebut. *Google App Engine* dan *Microsoft Azure Services* adalah beberapa contoh penyedia PaaS.
3. *Software as a Service* (SaaS). Konsumen dapat menggunakan aplikasi yang berjalan di infrastruktur awan dari berbagai penyedia, seperti jaringan, server, sistem operasi, penyimpanan, dan bahkan kemampuan aplikasi individual namun pengguna mungkin tidak dapat mengatur konfigurasi aplikasi khusus. Aplikasi dapat diakses dari berbagai perangkat klien melalui antarmuka klien, seperti browser web atau antarmuka program. Infrastruktur cloud yang mendasarinya tidak dikelola atau dikontrol oleh konsumen. Manajemen hubungan pelanggan (CRM), analitik intelijen bisnis, dan perangkat lunak akuntansi online, *Gmail*, Office 365, *DropBox*, *Google Docs* dan *Salesforce*. adalah beberapa contoh layanan yang tersedia.

E. *Kubernetes*

Kubernetes adalah proyek open source yang sangat besar dengan banyak kode dan fitur. Orkestrasi kontainer adalah tugas utama Kubernetes, yang berarti memastikan bahwa setiap kontainer yang menjalankan berbagai tugas dijadwalkan menjalankan mesin fisik atau virtual. Kubernetes memiliki kemampuan tambahan untuk mengawasi semua kontainer yang berjalan dan mengganti kontainer yang mati, tidak responsif, atau tidak sehat (Sayfan, 2017).

Adapun beberapa komponen yang terdapat didalam Kubernetes menurut (Hideto Saito, Hui-Chuan Chloe Lee, & Ke-Jou Carol Hsu, 2016) yaitu :

1. *Etcd* adalah penyimpanan data nilai kunci terdistribusi. Dapat diakses melalui *RESTful API* untuk melakukan operasi *CRUD* melalui jaringan. Kubernetes menggunakan etcd sebagai penyimpanan data utama.
2. *API Server* menyediakan *RESTful API* berbasis HTTP atau HTTPS untuk menghubungkan komponen kubernetes seperti kubectl, scheduler, pengontrol replikasi, dan datastore lainnya, serta kubelet dan kube-proxy, yang berjalan pada node Kubernetes dan lainnya.
3. *Scheduler* membantu memilih wadah mana yang dijalankan oleh node mana. Ini adalah algoritma sederhana yang menentukan prioritas untuk mengirim dan mengikat kontainer ke node, misalnya CPU, Memory dan berapa banyak container yang berjalan.
4. *Controller Manager* melakukan operasi cluster. Misalnya :
   1) Mengelola node Kubernetes
   2) Membuat dan memperbarui informasi internal Kubernetes dan
   3) Upaya untuk mengubah status saat ini ke status yang diinginkan.

F. *Kontainer*

Kontainerisasi adalah merupakan pendekatan untuk pengembangan perangkat lunak di mana aplikasi atau layanannya dependensi, dan konfigurasinya dikemas bersama sebagai gambar wadah. Aplikasi kontainer dapat diuji sebagai satu unit dan disebarkan sebagai contoh *kontainer image* ke sistem operasi host (OS) (Cesar de la Torre, Bill Wagner, & Mike Rousos, 2023). Kontainer juga mampu menyediakan pengembangan, integrasi, dan fitur penyebaran. Hal ini memungkinkan pengembang untuk kehandalan menerima, membangun dan menyebarkan. Mereka juga mampu melakukan tugas mereka dengan cepat dan melakukan rollback sistem dengan mudah (Miles & Sheldon, 2020).

G. *Web Server*

Web Server adalah suatu sistem yang memberikan layanan yang diminta (request) kepada pengguna melalui internet. *Web server* terdiri dari server fisik, sistem operasi, dan aplikasi yang memungkinkan komunikasi HTTP. Selain itu, web server juga disebut sebagai internet server. Pada dasarnya, tugas *web server* adalah menerima permintaan dari klien dan membuat tanggapan terhadap permintaan tersebut. Untuk permintaan statis, *web server* menerima direktori dokumen dari klien, mengidentifikasi nama

file, dan kemudian mengirimkan file ke penyimpanan lokal kepada klien. Permintaan halaman dinamis akan diterjemahkan ke dalam sebuah program oleh web server, yang kemudian menjalankan permintaan tersebut dan mengirimkan output program tersebut kepada orang yang mengakses internet (Azi, Arifwidodo, & Wahyudi, 2023).

H. *HAProxy*

HAProxy atau *High Availability Proxy* merupakan load balancer yang populer saat ini, berbentuk *open Source* TCP digunakan sebagai HTTP *load balancer* dan juga *software proxy-server*. *Haproxy* dibuat dalam Bahasa C oleh Willy Tarreau dan didukung oleh SSL, format log custom keep-alive, dan penulisan header. *HAProxy* dapat berjalan dalam dua mode: lapisan TCP 4 dan lapisan HTTP 7. Pada lapisan TCP 4, *HAProxy* mengirimkan paket TCP RAW dari client ke application server. Pada lapisan HTTP 7, HAProxy melakukan parsing HTTP Header sebelum mengirimkannya ke application server. HAProxy sendiri telah digunakan oleh beberapa website terkenal diantaranya seperti Twitter Github, StackOverflow, Reddit, dan Tumblr (Dewaweb Ninja, 2023). Adapun 5 metode yang digunakan HAProxy sebagai *load balancer* yakni :

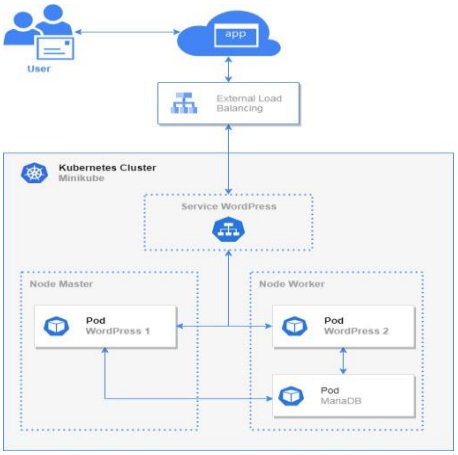
1. *Roundrobin* Merupakan algoritma yang sederhana, Cara kerjanya adalah dengan merotasi server. Trafik dikirim ke server pertama yang tersedia, lalu dikirim ke server kedua, dan seterusnya sesuai dengan jumlah server yang tersedia..
2. *Leastconn* Metode ini bekerja dengan membagi permintaan ke server dengan koneksi paling sedikit untuk mencegah overload.
3. *Least Response Time* Metode ini menginstruksikan load balancer untuk mengarahkan trafik ke server dengan koneksi aktif terkecil namun waktu respons tercepat saat terjadi permintaan data.
4. *Least Bandwidth* Pada metode ini, load balancer akan mengirimkan permintaan akses data kepada server yang memiliki jumlah trafik megabit per detik/second (Mbps) yang paling rendah.
5. *IP Hash* Metode ini menggunakan data IP seperti alamat IP destinasi, port nomor, URL, dan nama domain untuk menentukan permintaan akses data ke server.

HASIL DAN PEMBAHASAN

Dari Hasil analisis dan juga permasalahan yang ada, maka perancangan sistemnya adalah sebagai berikut :

A. Topologi Sistem

dalam perancangan sistem berdasarkan analisis maka penulis membuat topologi berdasarkan sistem yang ada, maka berikut topologinya



Sumber : Penelitian (2023)
Gambar 1 : Topologi

Topologi pada Gambar 1 merupakan sebuah rancangan untuk menhubungkan sistem dengan client dan server. Ketika melakukan pemanggilan request akan diarahkan ke external load balancing dan kemudian setelah itu masuk ke server. Didalam server tersebut terdapat service wordpress yang ada di dalam cluster kubernetes. Service wordpress itu sendiri merupakan gerbang untuk mengakses ke pod – pod yang ada. Pod wordpress merupakan unit terkecil yang bisa di deploy di Kubernetes cluster. pod wordpress adalah aplikasi yang kita running. Untuk menjalankan sistem tersebut penulis menggunakan 2 server yang berfungsi sebagai node master dan node worker, sedangkan untuk implementasinya menggunakan sebuah laptop sebagai server dengan sistem operasi ubuntu dan untuk aplikasinya menggunakan wordpress yang akan diuji dengan melakukan stress test pada sistem tersbut.

B. Spesifikasi Hardware

Adapun dalam setiap pembuatan jaringan ataupun sistem dibutuhkan perangkat keras atau hardware untuk menajalankannnya , begitu juga dalam pembuatan sistem ini dengan pemanfaatan sumberdaya laptop, penulis menjalankan tes dan uji coba dengan spesifikasi sebagai berikut :

Tabel 1. Spesifikasi Perangkat Keras

No.	Hardware	Spesifikasi
1	RAM	16 GB

C. Implementasi Sistem

Di tahap ini penulis akan memebuat dan menguji untuk mengetahui sistem sudah dapat diakses atau belum.

1. Membuat Node Master dan Worker

```
PS C:\Windows\system32> minikube status -p svr-lab3
svr-lab3
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

svr-lab3-m02
type: Worker
host: Running
kubelet: Running
```

Sumber : Penelitian (2023)

Gambar 2 : Node Master dan Worker

Pada Gambar 2 merupakan hasil pembuatan node master yang dinamai svr-lab3 dan node worker yang dinamai svr-lab3-m02 yang sudah dibuat untuk 2 server dan hasilnya berjalan dengan normal.

2. Menjalankan Minikube

```
PS C:\Windows\system32> minikube addons enable ingress-dns -p svr-lab3
* ingress-dns is an addon maintained by Google. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Using image gcr.io/k8s-minikube/minikube-ingress-dns:0.0.2
* The 'ingress-dns' addon is enabled
PS C:\Windows\system32>
```

```
PS C:\Windows\system32> minikube addons enable ingress-dns -p svr-lab3
* ingress-dns is an addon maintained by Google. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Using image gcr.io/k8s-minikube/minikube-ingress-dns:0.0.2
* The 'ingress-dns' addon is enabled
PS C:\Windows\system32> minikube addons enable metrics-server -p svr-lab3
* metrics-server is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Using image k8s.gcr.io/metrics-server/metrics-server:v0.6.1
* The 'metrics-server' addon is enabled
PS C:\Windows\system32>
```

Sumber : Penelitian (2023)

Gambar 3 : List perintah menjalankan minikube

Seperti gambar 3 diatas adalah perintah untuk menjalankan minikube yang berfungsi dimana untuk mengekspos *service* yang berjalan atau beroperasi di level HTTP dan memantau penggunaan sumber daya CPU di seluruh kluster dan ditampilkan secara statistic baik penggunaan *resource* atau sumber daya ke node ataupun ke pod.

D. Deploy Aplikasi

Langkah awal untuk melakukan deploy aplikasi yaitu membuat namespace untuk wordpress yang berfungsi sebagai lembar kerja nama di sistem yang memebedakan dengan nama sistem lainnya. Selanjutnya melakukan pengecekan untuk port dan nodeport yang nantinya berfungsi untuk menggunakan ip untuk instalasi wordpress. Dan berikut adalah printah lengkapnya.

```
PS D:\minikube\app\wordpress> kubectl create ns web
namespace/web created
PS D:\minikube\app\wordpress> kubectl apply -k ./ -n web
secret/mysql-pass-mgdt4d9ch7 created
service/wordpress created
service/wordpress-mysql created
persistentvolumeclaim/mysql-pv-claim created
persistentvolumeclaim/wp-pv-claim created
deployment.apps/wordpress created
deployment.apps/wordpress-mysql created
PS D:\minikube\app\wordpress> kubectl get pod -n web
NAME                               READY   STATUS              RESTARTS   AGE
wordpress-57449f9975-1kq82         0/1     ContainerCreating   0          15s
wordpress-mysql-977bc7485-gp46m    0/1     ContainerCreating   0          15s
PS D:\minikube\app\wordpress>
PS D:\minikube\app\wordpress>
PS D:\minikube\app\wordpress> kubectl get svc -n web
NAME              TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
wordpress         NodePort   10.99.233.243   <none>        80:32015/TCP     47s
wordpress-mysql   NodePort   10.107.249.86   <none>        3306:32052/TCP   47s
PS D:\minikube\app\wordpress> kubectl get pod -n web
NAME                               READY   STATUS    RESTARTS   AGE
wordpress-57449f9975-1kq82         1/1     Running   0          8m48s
wordpress-mysql-977bc7485-gp46m    1/1     Running   0          8m48s
PS D:\minikube\app\wordpress>
PS D:\minikube\app\wordpress>
PS D:\minikube\app\wordpress> kubectl scale deployment/wordpress --replicas=3 -n web
deployment.apps/wordpress scaled
PS D:\minikube\app\wordpress> kubectl get pod -n web
NAME                               READY   STATUS              RESTARTS   AGE
wordpress-57449f9975-2nj44         1/1     Running             0          5s
wordpress-57449f9975-9pc44         0/1     ContainerCreating   0          6s
wordpress-57449f9975-1kq82         1/1     Running             0          9m59s
wordpress-mysql-977bc7485-gp46m    1/1     Running             0          9m59s
PS D:\minikube\app\wordpress> kubectl get pod -n web
NAME                               READY   STATUS              RESTARTS   AGE
wordpress-57449f9975-2nj44         1/1     Running             0          12s
wordpress-57449f9975-9pc44         0/1     ContainerCreating   0          13s
wordpress-57449f9975-1kq82         1/1     Running             0          10m
wordpress-mysql-977bc7485-gp46m    1/1     Running             0          10m
PS D:\minikube\app\wordpress>
```

Sumber : Penelitian (2023)

Gambar 4 : List Perintah untuk Deploy Aplikasi

Setelah pembuatan namespace selanjutnya hal yang dilakukan yaitu instalasi wordpress dengan melihat terlebih dahulu servicenya untuk mengetahui

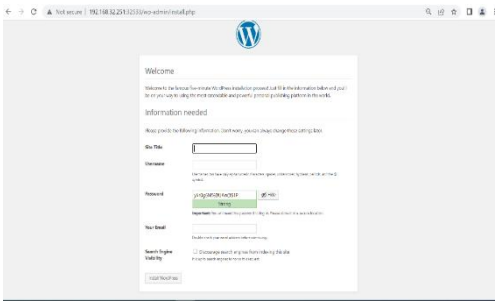
```
PS D:\minikube\app\wordpress> kubectl get svc -n web
NAME              TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
wordpress         NodePort   10.109.233.217  <none>        80:32533/TCP     3m52s
wordpress-mysql   NodePort   10.98.105.148   <none>        3306:32271/TCP   3m52s
PS D:\minikube\app\wordpress>
```

IP address ataupun port yang digunakan seperti perintah gambar 5 dibawah berikut.

Sumber : Penelitian (2023)

Gambar 5 : Perintah pengecekan service

Kemudian tinggal mencoba akses atau instalasi wordpress via browser dengas menyesuaikan cluster ip wordpress



Sumber : Penelitian (2023)

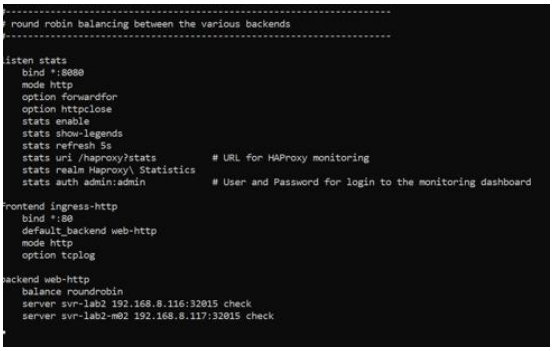
Gambar 6 : Aplikasi Wordpress

Tampilan pada gambar 6 diatas menujukkan bahwa pembuatan web server menggunakan minikube cluster sudah berhasil diakses melalui media wordpress yang ada didalam *container*.

E. Konfigurasi HAProxy

Untuk metode yang di gunakan pada HAProxy jenisnya round robin dimana request dibagi secara bergilir dan berurutan ke masing-masing server. Pada gambar 7 dibawah ini, *frontend ingress-http* bertugas untuk memanggil atau mengekspos service dimana *bind* yang digunakan yaitu port 80 berbasis

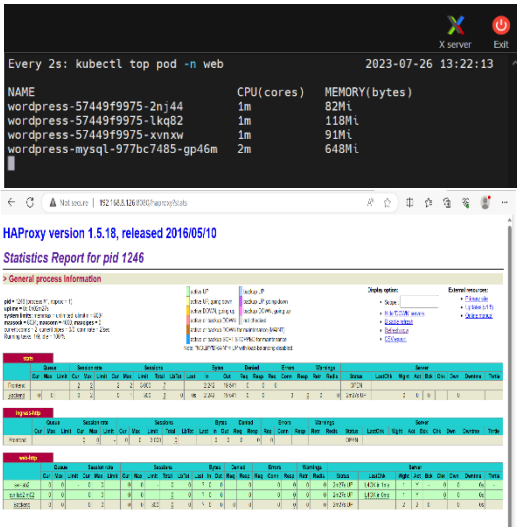
HTTP dan *default_backend web-http* yang berfungsi memanggil ip server svr-lab2 dan svr-lab2-m02 untuk meneruskan request secara merata untuk kedua server tersebut.



Sumber : Penelitian (2023)
Gambar 7 : Konfigurasi HAProxy

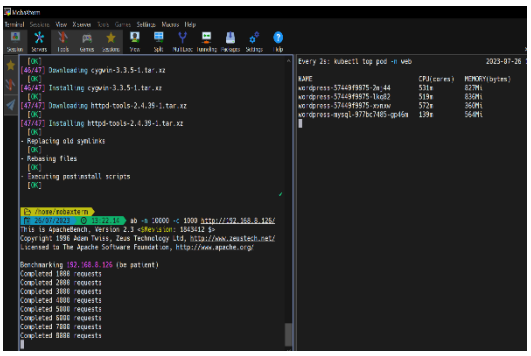
F. Pengujian Sistem

Tahapan ini akan dilakukan pengujian untuk melihat penggunaan sumber daya atau *resource* dengan menggunakan aplikasi Mobaxterm dan dari dashboard HAProxy. Data dari hasil penelitian akan memperlihatkan penggunaan perangkat keras seperti penggunaan cpu dan memory dalam keadaan idle. Idle sendiri adalah suatu kondisi dimana sistem dalam keadaan diam untuk mengurangi sumber daya perangkat yang diguankan, tetapi tetap menjalankan proses yang sedang dikerjakan seperti pada gambar 8.



Sumber : Penelitian (2023)
Gambar 8 : Pengujian sistem dalam keadaan Idle

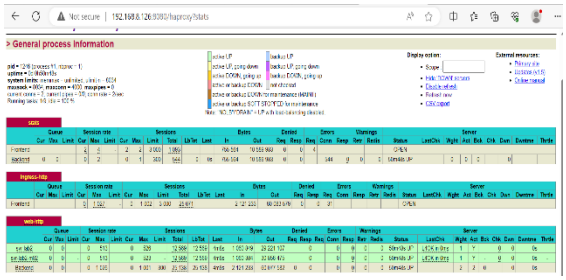
Keadaan diatas merupakan keadaan dimana kodisi *system* dalam keadaan idle sehingga bisa dillihat kondisi cpu yang masih rendah dan memory dalam kondisi normal. Begitu juga jika dilihat dari dashboard HAProxy belum ada request yang masuk ke node-node atau server.



Sumber : Penelitian (2023)
Gambar 7 : Pengujian Usage Memory dan CPU

Pada gambar diatas dapat dijelaskan bahwa proses stress test dengan melakukan *hit* ke server dengan permintaan 10000 request dengan jumlah client 1000 hasilnya bisa dilihat disebelah kanan dimana penggunaan cpu yang tadinya hanya satu digit naik ke ratusan digit diangka rata -rata 500m dan penggunaan memori juga ikut meningkat diangka 800Mi.

Sumber : Penelitian (2023)



Gambar 7 : Test Usage Request HAProxy

Untuk hasil diatas dapat lihat pada *session rate* yang diartikan sebagai user yang terhubung ke HAProxy dimana pada *fronted* maxnya sebanyak seribu dan pada *backend* maxnya terbagi secara merata ke server dengan jumlah yang hampir sama dengan jumlah total yang sama dengan fronted. Sedangkan pada kolom *session* yang berfungsi menghitung jumlah sesi secara penuh koneksi user ke server dengan menggunakan HAProxy. Dari gambar diatas juga kita bisa melihat bahwa penggunaan load balancer menghasilkan hasil yang hampir sama baik di svr-lab2 maupun svr-lab2-m02

KESIMPULAN

Berdasarkan kegiatan penelitian yang telah dilakukan Adapun beberapa kesimpulan yang diperoleh yaitu :

Dengan adanya external load balancing ketika dilakukan stress test permintaan request sebesar 10.000 request dari 1000 client, aplikasi tidak terlalu berat dan respond time yang juga tidak terlalu tinggi apabila dibandingkan dengan yang sebelumnya tidak menggunakan load balancer.
Dengan penggunaan kubernetes cluster bisa menjadi salah satu alternatif untuk penerapan virtualisasi server berbasis open source, serta hasil pembuatan rancangan bisa mencakup kebutuhan instansi perusahaan ataupun Lembaga terhadap sistem virtualisasi server. Adapun sistem aplikasi dapat berjalan secara terpisah di dalam satu server dengan memanfaatkan fitur pod sehingga aplikasi baik frontend dan backend tetap bisa berjalan ketika salah satu mati dan jika ada penambahan ataupun deploy aplikasi baru bisa langsung scale up sehingga resource yang digunakan sedikit dan lebih efisien dalam waktu dan *hardware*.

KESIMPULAN

Berdasarkan kegiatan penelitian yang telah dilakukan Adapun beberapa kesimpulan yang diperoleh yaitu :

1. external load balancing ketika dilakukan stress test permintaan request sebesar 10.000 request dari 1000 client, aplikasi tidak terlalu berat dan respond time yang juga tidak terlalu tinggi apabila dibandingkan dengan yang sebelumnya tidak menggunakan load balancer.
2. Dengan penggunaan kubernetes cluster bisa menjadi salah satu alternatif untuk penerapan virtualisasi server berbasis open source, serta hasil pembuatan rancangan bisa mencakup kebutuhan instansi perusahaan ataupun Lembaga terhadap sistem virtualisasi server. Adapun sistem aplikasi dapat berjalan secara terpisah di dalam satu server dengan memanfaatkan fitur pod sehingga aplikasi baik frontend dan backend tetap bisa berjalan ketika salah satu mati dan jika ada penambahan ataupun deploy aplikasi baru bisa langsung scale up sehingga resource yang digunakan sedikit dan lebih efisien dalam waktu dan *hardware*sumber.

REFERENSI

Azi, M. N. A., Arifwidodo, B., & Wahyudi, E. (2023). Analisis Performansi Web Server Saat Menangani Permintaan Client Menggunakan Metode Reserve Proxy Caching dan Varnish. *Journal of Telecommunication, Electronics, and Control Engineering (JTECE)*, *5*(1), 14–21. doi:10.20895/jtece.v5i1.843

Cesar de la Torre, Bill Wagner, & Mike Rousos. (2023). *NET-Microservices-Architecture-for-Containerized-NET-Applications*. (Mike Pope & Steve Hoag, Eds.) (Vol. 351). Redmond, Washington 98052-6399: Microsoft Developer Division, .NET and Visual Studio product teams A division of Microsoft Corporation One Microsoft Way.

Dewaweb Ninja. (2023, February 14). Cara Konfigurasi HAProxy Sebagai Load Balancer di CentOS 7.

Dr Joseph Teguh Santoso S.Kom, M. K. (2023). *Komputasi Awan (Cloud Computing)*. (M. K. M. Sholikan, Ed.). Semarang: Yayasan Prima Agus Teknik.

Dwiyatno, S., Rakhmat, E., & Gustiawan, O. (2020). IMPLEMENTASIVIRTUALISASI SERVER BERBASIS DOCKER CONTAINER, *7*(2).

Hideto Saito, Hui-Chuan Chloe Lee, & Ke-Jou Carol Hsu. (2016). *KubernetesCookbook* (First Edition, Vol. 379). Birmingham B3 2PB, UK: Packt Publishing Ltd.

Khalida, R., Muhajirin, A., & Setiawati, S. (2019). *Teknis Kerja Docker Container untuk Optimalisasi Penyebaran Aplikasi*.

Kusuma Hakim, D., Kun Riyanto, J., & Fauzan, A. (2019). Pengujian Algoritma Load Balancing pada Virtualisasi Server (Testing the Load Balancing Algorithm on Server Virtualization), *16*(1).

Miles, & Sheldon. (2020). *Kubernetes: A Step-By-Step Guide For Beginners To Build, Manage, Develop, and Intelligently Deploy Applications By Using Kubernetes (2020 Edition)* (2020 edition, Vol. 118). Independently Published.

Sayfan, G. (2017). *Mastering Kubernetes : automating container deployment and management*. (Sais Editing, Ed.) (First Edition, Vol. 426). Birmingham: Packt Publishing Ltd.

Sumbogo, Y. T., Data, M., & Siregar, R. A. (2018). *Implementasi Failover Dan Autoscaling Kontainer Web Server Nginx Pada Docker Menggunakan Kubernetes* (Vol. 2). Retrieved from http://j-ptiik.ub.ac.id

Zaida Efrizal. (2014). *Kupas Tuntas Teknologi Virtualisasi*. Andi Publisher.