JURNAL ILMIAH BETRIK
Besemah Teknologi Informasi dan Komputer
Editor Office : LPPM Sekolah Tinggi Teknologi Pagar Alam, Jln. Masik Siagim No. 75 Simpang Mbacang, Pagar Alam, SUM-SEL, Indonesia
Phone : +62 852-7901-1390.
Email : betrik@sttpagaralam.ac.id | admin.jurnal@sttpagaralam.ac.id
Website : https://ejournal.sttpagaralam.ac.id/index.php/betrik/index
ISSN : 2339 - 1871

PERFORMANSI RESPON TIME WEB SERVER DAN FAILOVER MENGGUNAKAN KUBERNETES

Sugiyatno[1], Ishak M[2]
Program Studi Informatika STMIK EL RAHMA YOGYAKARTA[12]
Jl. Sisingamangaraja No. 76 Yogyakarta
Sur-el : Sugiyatno@stmikelrahma.ac.id1, ishak@stmikelrahma.ac.id[2]

Abstrak*:* Sebuah aplikasi web dan mobile tentu membutuhkan sebuah server untuk pengembangan aplikasi. Ketersediaan Informasi ditentukan oleh ketersediaan dari Web Server yang mampu melayani permintaan dari klien. Kegagalan server dalam merespon permintaan atau single point of failure dapat terjadi kapan saja. Akibat dari kegagalan server tersebut dapat membuat aplikasi tidak dapat diakses oleh klien. Beberapa penyebab server failure atau kegagalan pada server adalah putusnya sumber daya listrik, malfungsi perangkat keras, sistem operasi yang crash dan kegagalan jaringan. Salah satu langkah yang dapat dilakukan dalam mengatasi kegagalan pada server adalah dengan menggunakan teknik server cluster dengan mekanisme failover. Failover merupakan sebuah mekanisme pergantian sebuah layanan seperti software atau hardware dari sebuah server yang sedang mengalami failure ke server lain yang tersedia didalam cluster. Failover pada umumnya diimplementasikan untuk tujuan meningkatkan ketersediaan layanan yang disediakan. Ukuran yang paling umum dari kategori ini adalah dua node, yang merupakan syarat minimum untuk melakukan redundansi. Implementasi klaster jenis ini akan mencoba untuk menggunakan redundansi komponen klaster untuk menghilangkan kegagalan di satu titik (Single Point of Failure). Kubernetes merupakan sebuah platform open source berpusat pada pengelolaan kontainer yang salah satu fungsinya adalah membuat sebuah sistem server cluster yang menjadi wadah kerja sama bagi beberapa server dalam penyediaan layanan. Dengan menggunakan metode failover ini kegagalan pada permintaan dapat dialihkan secara otomatis ke node-node yang telah ada. Respon waktu yang dibutuhkan untuk memberikan layanan kepada layanan memiliki waktu rata-rata relatif cepat kurang lebih 00:00:11 detik dengan menggunakan testing upload file. Hal ini dapat menunjukkan bahwa kinerja dari web server dengan menggunakan failover dapat diandalkan.

Kunci Utama: Failover, Kubernetes, Web Server, Respon Time.

Abstract*: A web and mobile application certainly requires a server for application development. Information availability is determined by the availability of a Web Server that is capable of serving requests from clients. Server failure in responding to requests or single point of failure can occur at any time. The result of server failure can make the application inaccessible to clients. Some of the causes of server failure or server failure are disconnection of electrical power sources, hardware malfunctions, operating system crashes and network failures. One step that can be taken to overcome server failures is to use a server cluster technique with a failover mechanism. Failover is a mechanism for replacing a service such as software or hardware from a server that is experiencing failure to another server available in the cluster. Failover is generally implemented for the purpose of increasing the availability of the services provided. The most common size of this category is two nodes, which is the minimum requirement for*

redundancy. This type of cluster implementation will try to use cluster component redundancy to eliminate failure at one point (Single Point of Failure). Kubernetes is an open source platform centered on container management, one of its functions is to create a cluster server system which becomes a collaborative forum for several servers in providing services. By using this failover method, failures in requests can be transferred automatically to existing nodes. The response time required to provide services to the service has a relatively fast average time of approximately 00:00:11 seconds using file upload testing. This can show that the performance of the web server using failover is reliable.

***Keywords** : Failover, Kubernetes, Web Server, Respon Time*

1. PENDAHULUAN

Jaringan dapat dikatakan sebagai sebuah hubungan maupun komunikasi yang dilakukan oleh beberapa perangkat yang terhubung pada sebuah titik tertentu. Beberapa jaringan juga dapat dihubungkan maupun disatukan dengan menggunakan sebuah teknik yang disebut virtualisasi. Virtualisasi merupakan sebuah teknik yang dimana dapat menciptakan sebuah wadah virtual dari sebuah objek yang memiliki wujud fisik seperti wadah penyimpanan data, sistem operasi dan juga server. Salah satu teknik virtualisasi yaitu virtualisasi berbasis kontainer.

Prasetyo, (2017) menyebutkan bahwa teknologi ini menjadi solusi permasalahan penurunan performansi server database yang bersifat sentralisasi dan dengan adanya teknologi virtualisasi ini administrator dapat membuat/mempunyai server database cadangan sebagai backup apabila server database utama nya down.

Virtualisasi berbasis kontainer merupakan sebuah teknik dimana virtualiasi tersebut tidak memerlukan hypervisor untuk berfungsi, kontainer ini dapat dijalankan langsung pada sistem operasi Pada VMWare mengharuskan pengguna untuk mengatur resource ketika instalasi, sedangkan pada kontainer tidak diharuskan karena kontainer memiliki resource tersendiri yang disediakan oleh host. Jika kekurangan resource maka kontainer akan mengambil resource yang terdapat pada hardware sesuai yang dibutuhkan. Salah satu virtualisasi berbasis kontainer yaitu Docker.

Docker merupakan sebuah tempat atau wadah yang menggunakan sistem berbasis kontainer, yang digunakan dalam mengembangkan aplikasi web server maupun mobile apps guna mempermudah fase deploy pada software (Rexa, Data and Yahya, 2019). Namun, di era teknologi yang semakin maju dimana aplikasi web ataupun mobile menjadi kebutuhan dasar untuk mendapatkan informasi maupun berbagi data. Sebuah aplikasi web dan mobile tentu membutuhkan sebuah server untuk pengembangan aplikasi. Namun, hal tersebut tidak akan cukup jika hanya menggunakan satu server hosting, karena kegagalan server dalam merespon permintaan atau single point of failure dapat terjadi kapan saja. Akibat dari kegagalan server tersebut dapat membuat aplikasi tidak dapat diakses oleh klien.

Beberapa penyebab server failure atau kegagalan pada server adalah putusnya sumber daya listrik, malfungsi perangkat keras, sistem operasi yang crash dan kegagalan jaringan (Oracle, 2018). Salah satu langkah yang dapat dilakukan dalam mengatasi kegagalan pada server adalah dengan menggunakan teknik server cluster dengan mekanisme failover. Server cluster merupakan sebuah gabungan dari beberapa komputer server yang digunakan oleh sebuah lembaga maupun organisasi untuk mencapai kebutuhan yang diperlukan oleh server untuk melampaui kemampuan sebuah mesin (Bella, Data, and Yahya 2018). Sedangkan Failover merupakan sebuah mekanisme pergantian sebuah layanan seperti software atau hardware dari sebuah server yang sedang mengalami failure ke server lain yang tersedia didalam cluster (jayaswal, 2005).

Failover merupakan salah satu fitur yang terdapat pada banyak platform termasuk Kubernetes. Kubernetes merupakan sebuah platform open source berpusat pada

pengelolaan kontainer yang salah satu fungsinya adalah membuat sebuah sistem server cluster yang menjadi wadah kerja sama bagi beberapa server dalam penyediaan layanan. Berdasarkan permasalahan beserta konsep penyelesaian yang dijabarkan di atas, Kubernetes memberikan suatu solusi dalam mengatasi server failure dengan mekanisme failover. Penelitian ini dilaksanakan dengan mengimplementasikan Kubernetes guna untuk menganalisis mekanisme dan respon failover dalam mengatasi server failure serta menguji performansi respon waktu web server ketika file upload yang berjalan didalam cluster.

2. METODE PENELITIAN (Font 12)

Pada pembuatan rancangan penelitian, metode yang digunakan adalah pengumpulan data dan perancangan sistem. Adapun alat dan bahan adalah sebagai berikut:

2.1 Alat dan Bahan

a. Alat
1. Macbook Pro (Spesifikasi Ram 8GB)
2. Lenovo Ideapad 3 (Spesifikasi Ram 8GB)

b. Bahan
1. Terminal
2. Docker
3. Kubernetes
4. VSCode
5. Vagrant
6. GitBash
7. Docker Image OS Ubuntu Server

2.2 Metode Pengumpulan Data

Metode pengumpulan data yang dapat digunakan adalah:

a. Metode Observasi

Metode observasi adalah metode pengumupulan data dengan melakukan pengamatan langsung terhadap obyek yang diteliti untuk mengumpulkan data dan informasi yang berkaitan dengan permasalahan yang ada.

b. Metode Studi Literatur

Studi literatur merupakan teknik pengumpulan data yang dilakukan dengan cara mengumpulkan dan membaca berbagai sumber tertulis seperti buku atau literatur yang menjelaskan tentang landasan teori. Pengumpulan data dan informasi dilakukan melalui penggalian informasi dan pengetahuan dari sumber-sumber seperti buku, karya tulis, serta beberapa sumber lainnya yang ada hubungannya dengan objek yang relevan terhadap penelitian.

2.3. Failover

Failover pada umumnya diimplementasikan untuk tujuan meningkatkan ketersediaan layanan yang disediakan. Elemen klaster akan bekerja dengan nodenode redundan yang kemudian digunakan untuk menyediakan layanan saat salah satu elemen klaster mengalami kegagalan. Ukuran yang paling umum dari kategori ini adalah dua node, yang merupakan syarat minimum untuk melakukan redundansi. Implementasi klaster jenis ini akan mencoba untuk menggunakan redundansi komponen klaster untuk menghilangkan kegagalan di satu titik (Single Point of Failure).

Failover dapat memudahkan administrator jaringan dalam manajemen jaringan, karena tidak perlu melakukan konfigurasi jaringan jika jaringan utama down. Dengan adanya failover juga dapat menjamin ketersediaan tinggi dan jaringan yang andal Riko Rudiyanto (2023).

Sumbogo, Data and Siregar (2018), mengimplementasikan Failover Dan Autoscaling Kontainer Web Server Nginx Pada Docker Menggunakan Kubernetes. Persamaan pada penelitian yang dilakukan yaitu sama-sama menggunakan Kubernetes, Docker kontainer, Web Server Nginx. Perbedaan dengan penelitian yang dilakukan terletak pada implementasi pada failover yang secara spesifik dilakukan pada web server Nginx dan proses autoscaling.

2.4. Docker

Docker menurut Sugianto (2016), adalah suatu platform terbuka bagi pengembang perangkat lunak dan pengelola sistem jaringan untuk membangun, mengirimkan dan menjalanan aplikasi-aplikasi terdistribusi. Definisi tersebut membawa pengertian praktis bahwa Docker merupakan suatu cara memasukkan layanan ke dalam lingkungan terisolasi bernama container,

sehingga layanan tersebut dapat dipaketkan menjadi satu bersama dengan semua pustaka dan software lain yang dibutuhkan (Bik and Asmunin, 2017)

2.5.Objek Kubernetes

a. Pods

Pod merupakan sebuah abstraksi dari Kubernetes yang bertugas untuk merepresentasikan sebuah grup yang terdiri dari satu atau lebih kontainer (seperti Docker) serta beberapa sumber daya bersama untuk kontainer-kontainer itu. Sumber daya tersebut termasuk penyimpanan atau yang disebut volume, jaringan, sebagai satu alamat IP klaster unik serta informasi tentang cara menjalankan tiap kontainer, seperti versi *image* atau porta spesifik yang digunakan oleh kontainer.

Pod melakukan pemodelan sebuah "*logical host*" yang spesifik pada aplikasi dan dapat berisi beberapa kontainer aplikasi yang berbeda dan relatif terkait erat. Contoh pengaplikasiannya adalah sebuah Pod yang mungkin terdiri atas kontainer aplikasi Node.js dan juga kontainer berbeda yang bertugas menyediakan data untuk dipublikasikan ke server web Node.js.

Container-kontainer yang terdapat di dalam sebuah Pod berbagi satu alamat IP dan ruang *porta*, selalu berada pada letak, dan jadwal, serta operasi yang berjalan dalam satu konteks bersama (*shared context*) pada Node yang sama.

Pod merupakan unit terkecil dalam *platform* Kubernetes. Ketika pengguna membuat sebuah deployment, deployment tersebut membuat pod dengan kontainer-kontainer di dalamnya (bukan dengan membuat kontainer secara langsung). Setiap pod terikat langsung dengan node yang telah dijadwalkan dan tetap berada di sana hingga proses diterminasi (berdasarkan *restart policy*) atau penghapusan. Jika terjadi kegagalan pada sebuah node, pod identik akan dijadwalkan di node lain dalam klaster.

b. Service

Service berfungsi untuk mengarahkan *traffic* ke aplikasi yang berada pada kluster. Pada cluster, Pod secara dinamis dikelola oleh Kubernetes *control plane*, Pod dapat hilang dan muncul karena proses penjadwalan dan *autoscaling* yang mengakibatkan perubahan pada alamat IP (*ephemeral*). Pemberian akses yang konsisten terhadap aplikasi di sekumpulan Pod tersebut memerlukan bentuk *service* ini. Kubernetes menyediakan tiga jenis *service* yang dapat digunakan oleh pengguna dalam memenuhi kebutuhannya yaitu ClusterIP, NodePort, dan LoadBalancer.

ClusterIP merupakan layanan yang disediakan oleh Kubernetes default. Layanan pengguna akan terdaftar di ClusterIP kecuali pengguna menentukan jenis yang berbeda secara manual. ClusterIP berfungsi untuk memberikan satu alamat IP pada aplikasi yang dapat diakses pada lingkungan internal kluster.

NodePort menyediakan akses pada *port statis* yang sama di semua node yang pada tahapan selanjutnya akan mengarahkan *traffic* menuju ke aplikasi yang menjadi sasaran. Saat pengguna membuat NodePort maka *service* ClusterIP akan secara otomatis dibuat. NodePort akan mengarahkan *traffic* menuju ke ClusterIP, sedangkan *Loadbalancer akan* menyediakan dan memberikan akses dari luar menuju ke dalam aplikasi yang berjalan pada kluster melalui mekanisme *load balancing*.

Saat pengguna membuat service LoadBalancer maka service NodePort dan ClusterIP akan secara otomatis dibuat. Mekanisme internal yang terjadi adalah LoadBalancer akan mengarahkan *traffic* ke NodePort lalu selanjutnya menuju ClusterIP. Agar dapat menggunakan service ini, pengguna harus memiliki kluster Kubernetes yang berjalan *cloud provider* di AWS, GKE atau AKS. LoadBalancer harus mengerjakan *setup* dengan cara mandiri jika ingin menjalankannya secara lokal.

c. c. Replicaset

ReplicaSet berperan dalam melakukan pemeliharaan himpunan yang stabil dari

replika Pod yang sedang bekerja pada satu waktu tertentu. Sehingga, untuk menjamin ketersediaan dari beberapa Pod yang identik dalam jumlah tertentu pilihan untuk menggunakan ReplicaSet menjadi pilihan yang lebih umum. Sebuah ReplicaSet diidentifikasikan dengan beberapa *field* termasuk selektor yang menentukan cara untuk mengenali Pod yang dapat diakuisisi, jumlah replika yang mengindikasikan banyaknya jumlah Pod yang harus dikelola, dan sebuah *template* pod yang menentukan data dari berbagai Pod baru yang harus dibuat untuk memenuhi kriteria jumlah replika.

Sebuah ReplicaSet pada tahapan selanjutnya akan memenuhi tujuannya dengan cara membuat dan menghapus Pod sesuai dengan kebutuhan untuk mencapai jumlah yang diinginkan. Ketika ReplicaSet butuh untuk membuat Pod baru, maka template Pod dapat digunakan.

d. Deployment

Deployment mendeskripsikan sebuah *state* yang diinginkan, kemudian Deployment Pengontrol melakukan perubahan state yang saat ini digunakan menjadi seperti pada deskripsi secara bertahap. Pengguna dapat mendefinisikan *Deployment* untuk membuat *ReplicaSets* baru atau untuk menghapus *Deployment* yang telah ada dan mengadopsi semua *resource* untuk *Deployment* baru. Adapun contoh penggunaan *deployment* adalah sebagai berikut:

1. *Deployment* dibuat untuk merilis *ReplicaSet*. *ReplicaSet* membuat Pod di belakang layar. Lakukan pemeriksaan pada status rilis untuk mengetahui kesuksesan proses perilisan.
2. State baru dari Pods dirilis dengan cara melakukan pembaruan *PodTemplateSpec* milik *Deployment*. *ReplicaSet* baru akan dibuat dan Deployment akan mengatur perpindahan Pod secara teratur dari ReplicaSet lama ke ReplicaSet baru. Setiap ReplicaSet yang baru akan mengganti revisi dari Deployment. (Kubernates, 2024)
3. Jika *state* Deployment sekarang mengalami ketidakstabilan, maka akan dilakukan pengembalian ke revisi Deployment sebelumnya. (kubernates, 2024) Setiap prosedur pengembalian akan mengganti revisi Deployment.
4. Deployment akan melakukan pembesaran untuk memfasilitasi beban yang lebih.
5. Proses depolyment akan dijeda untuk menerapkan perbaikan pada PodTemplateSpec-nya, lalu dilanjutkan untuk memulai pengrilisan baru.

2.6.Objek Kubernetes

Pengertian server adalah sebuah sistem computer yang menyediakan jenis layanan tertentu dalam sebuah jaringan komputer. Terkadang istilah server disebut sebagai web server. Server juga bertugas untuk menjalankan software administratif yakni software yang mengontrol akses terhadap jaringan dan sumber daya yang terdapat di dalamnya. Hal ini termasuk file atau alat pencetak (printer), dan memberikan akses kepada workstation anggota jaringan. Di dalam sistem operasi server, umumnya terdapat berbagai macam service yang menggunakan arsitektur klien/server. Contoh dari service yang diberikan oleh server ini antara lain Mail Server, DHCP Server, HTTP Server, DNS Server, FTP Server dan lain lain. Setiap sistem operasi server umumnya merangkai berbagai layanan tersebut. Atau bisa juga layanan tersebut diperoleh dari pihak ketiga. Setiap layanan tersebut akan merespons terhadap request dari klien (Prakoso and Asmunin, 2018).

2.6 Langkah-Langkah Penelitian

Gambar 1. Tahapan Penelitian

a. Perencanaan

Tahap awal yang dilakukan dalam proses perencanaan yaitu kegiatan ini dilakukan setelah pengumpulan data yang diperlukan untuk membuat sistem dan hasil dari pengumpulan data adalah sebagai bahan dalam perancangan sistem.

b. Perancangan dan Desain

Setelah melakukan pelaksanaan penelitian dan kajian literatur sehingga didapatkan data digital, maka selanjutnya dilakukan perancangan sistem yaitu persiapan mesin, desain topologi jaringan dari server cluster dan perancangan mekanisme yang digunakan.

c. Implementasi

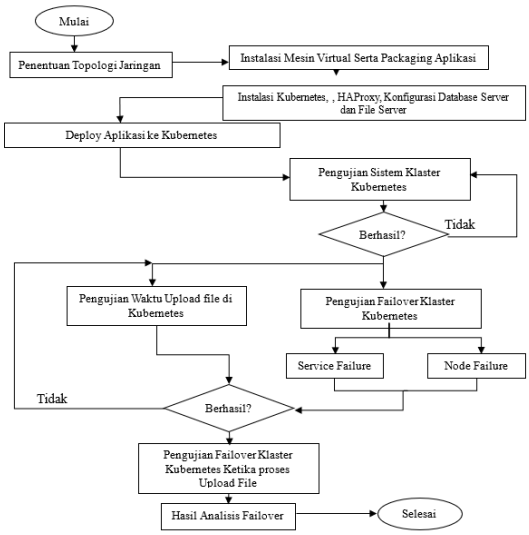
Implementasi dari perancangan dan desain sistem yang digunakan yaitu dengan instalasi server cluster dan pemasangan software dan tools dari platform yang sudah ditentukan.

3. HASIL DAN PEMBAHASAN

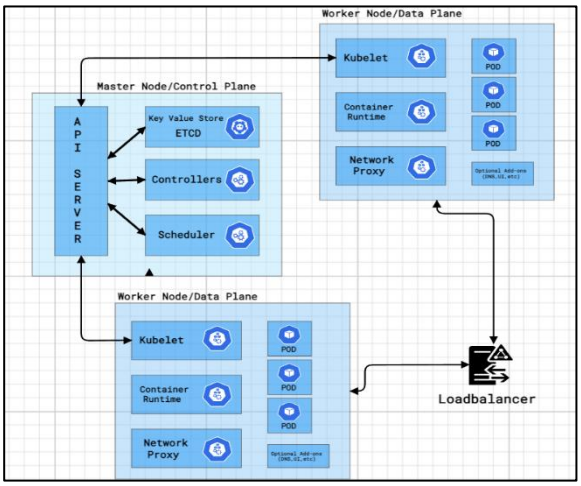
3.1. Perancangan Infrastruktur

Analisis penelitian dilakukan mulai dari memasukkan proses pembuatan topologi jaringan sampai dengan pengambilan data hasil analisis dari waktu respon Ketika *upload file*, *failover klaster* dan *failover* Ketika proses *upload file.*

Rancangan Infra struktur dapat dilihat dalam gambar 2.



Gambar 2. Rancangan Infrastuktur

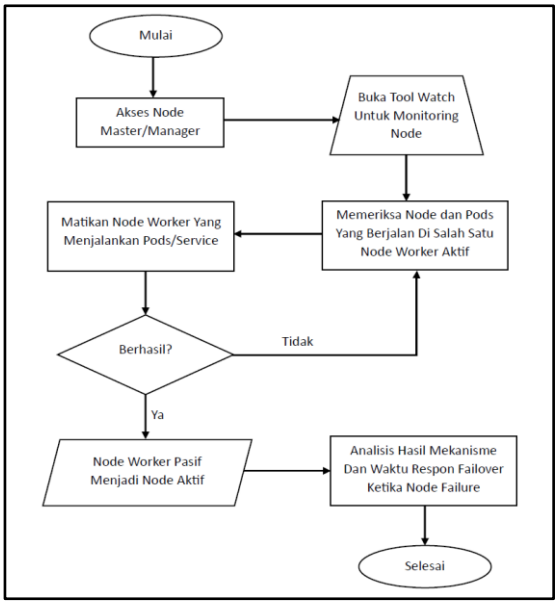


Gambar 3. Diagram *Infrastruktur Kubernetes Cluster*

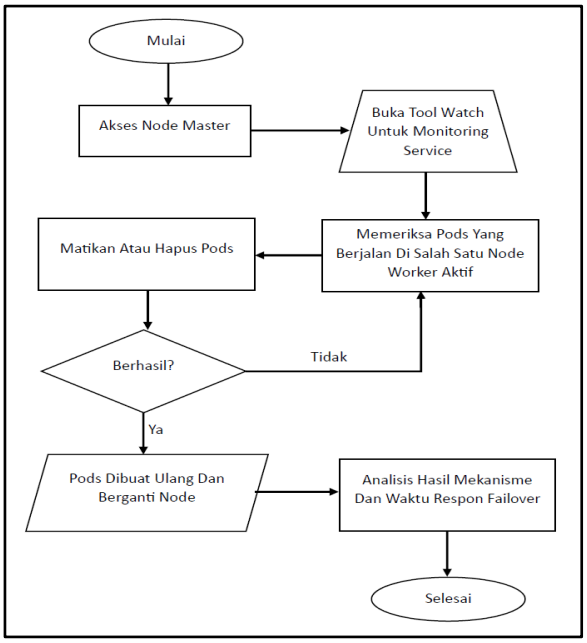
Deskripsi pada Gambar 3 di atas menunjukkan bahwa dalam pengelolaan *cluster*, administrator dari Kubernetes Cluster membutuhkan akses langsung ke dalam *node master (control plane)*, melalui CLI ataupun UI (*dashboard*) agar dapat berkomunikasi langsung dengan API Server yang membutuhkan *tool* yaitu kubectl.

3.2. Skenario Pengujian Failover Kubernetes

Pada skenario ini diperlukan 2 pengujian yaitu failover ketika node failure dan failover ketika service failure.



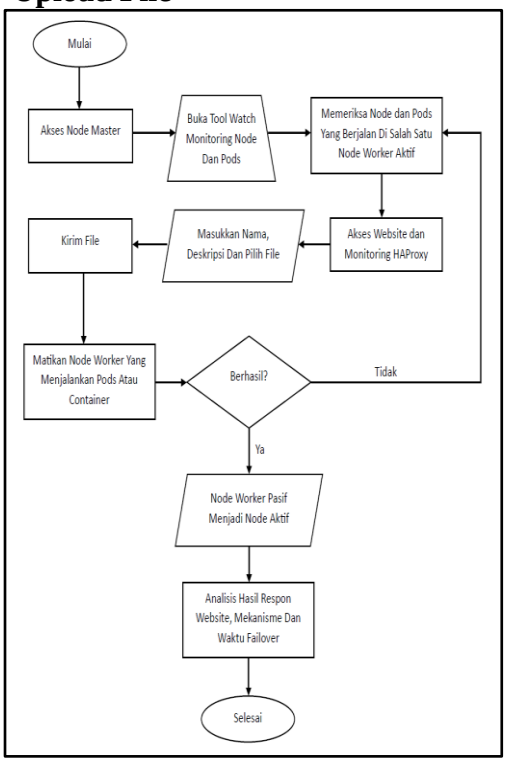
Gambar 4. Bagan Pengujian Failover Ketika Node Failure



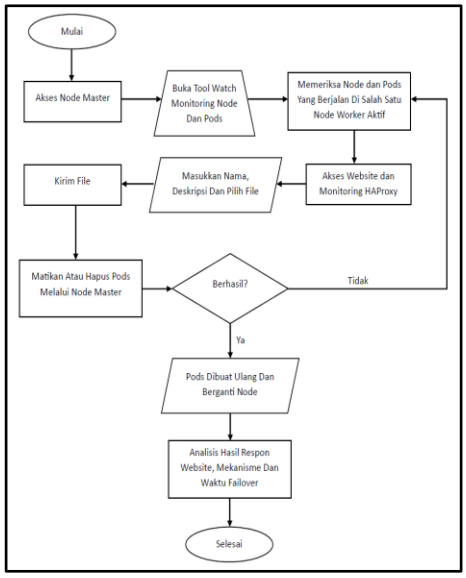
Gambar 5. Bagan Pengujian Failover Ketika Service Failure

Pada Gambar 4 dan Gambar 5 pertama-tama mengakses node master pada kubernetes. Setelah berada di dalam *node CLI*, buka *tool watch* untuk memantau node. Selanjutnya adalah memeriksa node dan pods/service yang berjalan di salah satu node aktif kemudian langkah pengujian node failure adalah mematikan node worker yang menjalankan pods/service, jika proses node failure tidak berhasil maka perlu melakukan pengecekan ulang node dan pods/service. Namun, jika proses failover berhasil maka node yang sebelumnya pasif akan mengambil alih *traffic* dan menjadi node aktif lalu node yang sebelumnya aktif akan *down* atau berstatus *not ready*. Kemudian proses berikutnya adalah melakukan analisis terhadap hasil proses failover pada node failure dan hasil analisis failover pada service failure.

3.3.Skenario Pengujian Failover Ketika Upload File



Gambar 6. Bagan Pengujian Failover Node Ketika Upload File



Gambar 7. Bagan Pengujian Failover Service Ketika Upload File

Gambar 6 dan Gambar 7 merupakan alur proses pengujian dari failover ketika upload file. Tahap pertama yaitu mengakses node master atau node manager kemudian buka *tool watch* untuk pemantauan node dan pods/service selanjutnya memeriksa node dan pods/service tersebut telah berjalan disalah satu node worker aktif lalu akses website dan akses dashboard monitoring HAProxy yang bertujuan untuk memantau aktifitas dari node worker yang terdaftar di loadbalancer. Tahap selanjutnya mengisi form nama, deskripsi dan memilih file lalu proses kirim file. Ketika proses kirim file sedang berlangsung, matikan node worker yang aktif untuk failover node dan hapus pods/service untuk failover service. Pada tahap selanjtunya merupakan suatu kondisi jika tidak berhasil, maka diperlukan untuk memeriksa ulang status dari node dan pods/service namun jika berhasil maka node worker pasif akan menjadi aktif dan node worker yang aktif sebelumnya akan down untuk failover node dan pods/service akan dibuat ulang untuk failover service. Selanjutnya adalah tahap analisis dari pengujian yang telah dilakukan yaitu analisis respon website, mekanisme dan waktu dari failover tersebut.

3.4. Analisis Hasil Failover

Pada percobaan yang telah dilakukan menunjukkan bahwa proses *failover* pada *web server* berhasil dilakukan saat sistem mengalami *node failure* ataupun *service failure*. Dalam hal *clustering*, ketika pertama kali membuat servis kubernetes hanya mengekspos node worker.

Selanjutnya dalam hal *node failure* pada Kubernetes beroperasi dengan baik. Ketika terjadi *node failure, node master* pada Kubernetes, dan *node manager*, maka pada Kubernetes akan memberikan status *terminating*. sehingga scheduler pada kubernetes melakukan failover dengan memindahkan service yang berjalan ke node worker yang sedang aktif.

Tabel 1. Skenario Analisis Hasil *Failover Kubernetes*

No	Skenario	Hasil	Keterangan
1	Akses *web server* dalam kondisi seluruh *node worker* aktif	Sistem mampu melayani *request*	Sistem *server* berjalan dengan baik
2	Akses *web server* dalam kondisi salah satu *node worker* mengalami kegagalan	Berhasil diakses walaupun salah satu *node worker* mengalami kegagalan/*down*	*pod* pada *node worker* yang *down* berpindah pada *node worker* yang tersedia/aktif.
3	Akses *web server* dalam keadaan seluruh *pod down*	berhasil diakses walaupun seluruh *pod down*	Kubernetes melakukan *service failover* dengan pembuatan *pod* baru pada semua *node worker* aktif

Tabel 1 menunjukan bahwa sistem mampu melayani permintaan dari klien dalam kondisi seluruh *node worker* aktif maupun dalam kondisi salah satu *node worker* tidak aktif dan ketika seluruh *pod down*. *Scheduler* Kubernetes akan menjadwalkan untuk membuat ulang *Pods* yang down atau mengalami crash selama node didalam cluster tersedia. Namun hal tersebut akan berbeda

kasus Ketika semua node worker mengalami down.

Hal tersebut dapat tercapai sebagai dampak dari fungsi dari ketersediaan yang diberikan oleh Kubernetes. Jika salah satu *node worker down* maka *service* akan digantikan oleh *node worker* yang sedang aktif.

Saat salah satu *node worker* berada dalam keadaan *down*, maka *pod* yang berada pada *node* tersebut akan melakukan *service failover* dengan cara membuat *pod* baru yang secara otomatis akan muncul pada salah satu *node worker* yang aktif.

Selain itu, ketika *pods* pada seluruh *server node* mengalami *down*, maka sistem secara otomatis akan melakukan *failover* melalui pembuatan *pod* baru.

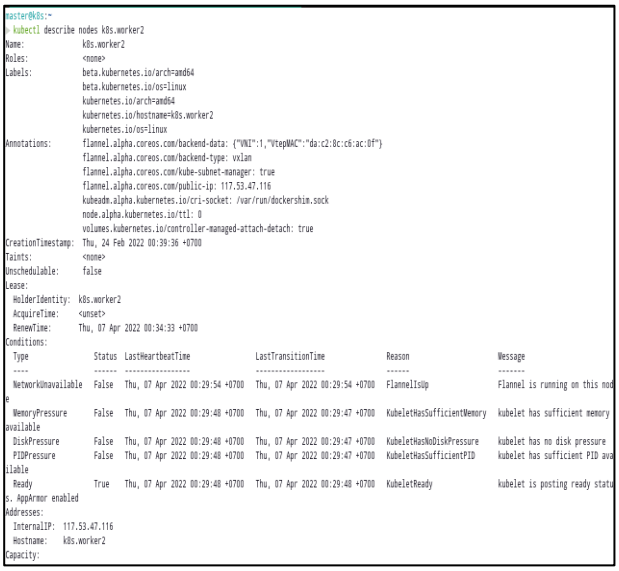
Hal ini cukup berbeda dengan Docker Swarm yang diharuskan melakukan *scaling* terlebih dahulu disebabkan oleh *node manager* yang terekspos sehingga ikut berperan dalam menangani proses *request* ke *website*. Berikut tabel hasil dari pengujian failover pada docker swarm.

3.4.Pengujian Waktu Failover Pada Kubernetes Cluster

Skenario pada kubernetes dilakukan dengan penghentian *pod* jika berhasil berpindah *node*, maka seterusnya akan dilakukan penghentian sebanyak lima kali. Begitu pula dengan *node worker* akan dihentikan secara bergantian sesuai dengan *node worker* yang sedang menjalankan *service*. Waktu yang diambil sejak *pod* atau *node worker* dihentikan. Detail waktu yang digunakan *log* dari Kubernetes dapat dilihat dengan menggunakan perintah “kubectl describe <nama pod/proses>” dan “kubectl describe <nama node>”.

Pada Gambar 8 menunjukan status *log deployment* seperti waktu mulai, *images* yang digunakan, *node* tempat *deployment* dijalankan dan keterangan lainnya, sementara Gambar 9 menunjukan *log* dari *node* atau dalam Kubernetes disebut sebagai *event.*

Gambar 8. Status *Describe Pod*



Gambar 9. Status *Describe Node Worker2*

Pengecekan dan penghentian pods dan node worker akan dilakukan bergantian sehingga kalkulasi dari rata-rata waktu dapat ditentukan.

3.5. Analisis Hasil Pengujian Waktu *Failover*

Berikut adalah data yang didapatkan dari hasil pengujian *failover*. Tabel 3.2 menunjukan hasil rata-rata waktu pengujian ketika *node* diberhentikan secara paksa dan Tabel 3.3 menunjukan hasil rata-rata waktu ketika layanan *web server* diberhentikan secara paksa.

Tabel 2. Hasil Pengujian *Failover Node Failure*

Pengujian	Waktu failover(detik)
1	345
2	344
3	346

4	346
5	345
Rata-rata	**345,2**

Tabel 3. Hasil Pengujian *Failover Service Failure*

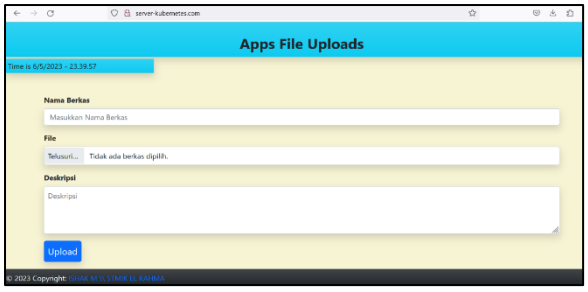
Pengujian	Waktu failover(detik)
1	5
2	5
3	7
4	6
5	5
Rata-rata	**5,6**

Dari tabel Tabel 2 dan Tabel 3 dapat ditunjukkan waktu yang dibutuhkan untuk melakukan *failover* pada sebuah layanan *web server*. Rata-rata waktu yang dibutuhkan adalah 345.02 *second*, atau 5 menit 45 detik untuk *node failure* sedangkan proses *failover* pada pengujian *service failure* memiliki rata-rata waktu yang sangat singkat yaitu 5.6 detik.

Hal ini terjadi karena ketika *node* diberhentikan secara paksa atau mengalami masalah yang menyebabkan *down*, modul Kubelet pada *node worker* tidak dapat berkomunikasi dengan *node master*, sehingga *node master* membutuhkan waktu lebih lama untuk melakukan pemeriksaan.

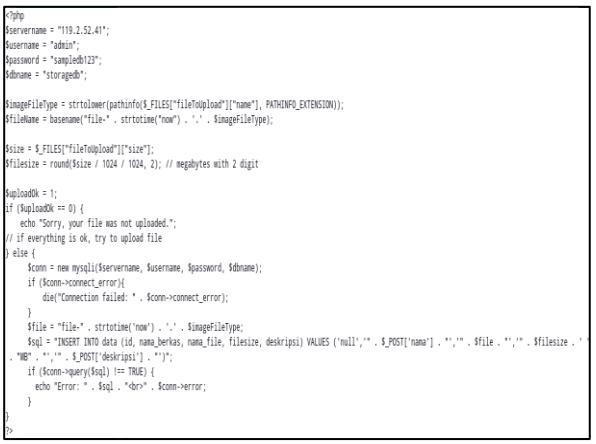
3.6 Pengujian Respon Waktu Upload File

Pada tahap pengujian ini, dilakukan beberapa kali percobaan untuk menemukan durasi yang diperlukan saat pengiriman data. Percobaan yang dilaksanakan melalui prosedur *upload file* sebanyak 30 kali dengan data yang telah ditentukan yaitu *file* Gambar berukuran 10MB.



Gambar 10. Halaman *Website* pada Kubernetes Cluster

Pengujian untuk melakukan proses upload menggunakan script yang ditulis dalam bahasa pemrograman PHP. Script upload dapat dilihat dalam gambar 11.



```php
<?php
$servername = "119.2.52.41";
$username = "admin";
$password = "sampledb123";
$dbname = "storagedb";

$imageFileType = strtolower(pathinfo($_FILES["fileToUpload"]["name"], PATHINFO_EXTENSION));
$fileName = basename("file-" . strtotime("now") . "." . $imageFileType);

$size = $_FILES["fileToUpload"]["size"];
$filesize = round($size / 1024 / 1024, 2); // megabytes with 2 digit

$uploadOk = 1;
if ($uploadOk == 0) {
   echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
    $conn = new mysqli($servername, $username, $password, $dbname);
    if ($conn->connect_error){
       die("Connection failed: " . $conn->connect_error);
    }
    $file = "file-" . strtotime('now') . "." . $imageFileType;
    $sql = "INSERT INTO data (id, nama_berkas, nama_file, filesize, deskripsi) VALUES ('null','" . $_POST['nama'] . "','" . $file . "','" . $filesize . " MB" . "','" . $_POST['deskripsi'] . "')";
     if ($conn->query($sql) !== TRUE) {
      echo "Error: " . $sql . "<br>" . $conn->error;
     }
}
?>
```

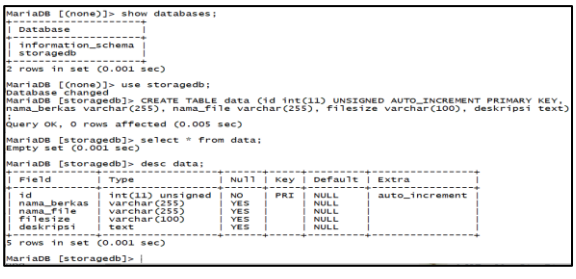
Gambar 11. Sourcecode upload Ke Database

```php
<?php
if (isset($_POST['submit']))
{
    $ftp_hostname = '119.2.52.41';
    $ftp_username = 'ishak';
    $ftp_password = 'ishak123';
    $src_file = $_FILES['fileToUpload']['name'];
    $size = $_FILES["fileToUpload"]["size"];
    $filesize = round($size / 1024 / 1024, 2); // megabytes with 2 digit
    $temp_file_path = $_FILES['fileToUpload']['tmp_name'];
    $upload_time = time() - $_SERVER['REQUEST_TIME'];
    $timestamp = $_SERVER["REQUEST_TIME"];
    $file_name = $_POST["nama"];
    $dateupload = date('H:i:s', $timestamp);
    if ($src_file!='')
    {
        $ftpcon = ftp_connect($ftp_hostname) or die('Error connecting to ftp server...');
        $ftplogin = ftp_login($ftpcon, $ftp_username, $ftp_password);
        ftp_pasv($ftpcon, TRUE);
        ftp_put($ftpcon, $src_file, $temp_file_path, FTP_BINARY);
        ftp_close($ftpcon);

        echo "<h1> $file_name Success </h1>"."<br>";
        echo "<p> File $src_file has $filesize MB of size </p>"."<br>";
        echo "<p> $dateupload </p>";
    }
    else
        echo "$src_file Failed Upload";
}
?>
```

Gambar 12. Sourcecode upload Ke Server

Gambar 12 menunjukkan source code untuk mengupload file ke dalam server dalam bahasa PHP.



```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| storagedb          |
+--------------------+
2 rows in set (0.001 sec)

MariaDB [(none)]> use storagedb;
Database changed
MariaDB [storagedb]> CREATE TABLE data (id int(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
nama_berkas varchar(255), nama_file varchar(255), filesize varchar(100), deskripsi text)
;
Query OK, 0 rows affected (0.005 sec)

MariaDB [storagedb]> select * from data;
Empty set (0.001 sec)

MariaDB [storagedb]> desc data;
+-----------+---------------------+------+-----+---------+----------------+
| Field     | Type                | Null | Key | Default | Extra          |
+-----------+---------------------+------+-----+---------+----------------+
| id        | int(11) unsigned    | NO   | PRI | NULL    | auto_increment |
| nama_berkas | varchar(255)      | YES  |     | NULL    |                |
| nama_file | varchar(255)        | YES  |     | NULL    |                |
| filesize  | varchar(100)        | YES  |     | NULL    |                |
| deskripsi | text                | YES  |     | NULL    |                |
+-----------+---------------------+------+-----+---------+----------------+
5 rows in set (0.001 sec)
MariaDB [storagedb]>
```

Gambar 13. Database Server dengan Mysql

Gambar 3.12 menunjukkan desain table untuk menyimpan record file upload.

Tabel 4. *Upload File* pada Kubernetes Cluster

Data Upload	Mulai Upload	Selesai Upload	Lama Upload(detik)
1	00:17:51	00:18:03	00:00:12
2	00:19:31	00:19:41	00:00:10
3	00:20:16	00:20:26	00:00:10
4	00:21:27	00:21:38	00:00:11
5	00:22:38	00:22:51	00:00:13
6	00:23:41	00:23:52	00:00:11
7	00:24:22	00:24:33	00:00:11
8	00:25:40	00:25:51	00:00:11
9	00:26:37	00:26:50	00:00:13
10	00:27:33	00:27:44	00:00:11
11	00:28:19	00:28:32	00:00:13
12	00:29:02	00:29:15	00:00:13
13	00:29:52	00:30:02	00:00:10
14	00:30:34	00:30:44	00:00:10
15	00:31:26	00:31:36	00:00:10
16	00:32:08	00:32:19	00:00:11
17	00:32:58	00:33:10	00:00:12
18	00:33:49	00:34:02	00:00:13
19	00:34:43	00:34:58	00:00:15
20	00:35:40	00:35:50	00:00:10
21	00:36:15	00:36:26	00:00:11
22	00:37:31	00:37:42	00:00:11
23	00:38:21	00:38:31	00:00:10
24	00:39:34	00:39:46	00:00:12
25	00:40:26	00:40:36	00:00:10
26	00:41:06	00:41:18	00:00:12
27	00:41:45	00:41:56	00:00:11
28	00:42:38	00:42:49	00:00:11
29	00:43:17	00:43:28	00:00:11
30	00:43:56	00:44:06	00:00:10
Rata-rata			**00:00:11**

4. SIMPULAN

Dari proses percobaan, pengujian dan analisis yang telah dilakukan, ditemukan beberapa data bahwa performansi respon waktu web server dan failover pada kubernetes dan docker swarm memiliki perbedaan yang cukup signifikan.

1. Proses pengiriman file pada kubernetes rata-rata memiliki waktu 11 detik pada kubernetes
2. Mekanisme failover dan respon waktu yang dimiliki kubernetes membutuhkan waktu beberapa saat untuk berpindah dari percobaan rata-rata membutuhkan waktu . Hal tersebut sangat dipengaruhi oleh infrastruktur pada docker swarm yang sudah terintegrasi dengan docker engine sedangkan kubernetes menjadikan docker engine sebagai komponen pihak ketiga dan ketika failover terjadi, scheduler kubernetes perlu melewati rules dan kebijakan yang dapat menyebabkan keterlambatan dalam mengalihkan workload ketika failover.
3. Kubernetes memiliki fitur untuk mengelola *pods/container* dan *service network* untuk mengelola jaringan yang dapat diatur sedemikian rupa berdasarkan kebutuhan sedangkan docker swarm tidak memiliki fitur untuk mengelola *container* dan *service network*, namun hal tersebut dapat dilakukan dengan perintah dasar docker. Container adalah unit *software* yang mengemas semua kode dengan dependensinya, sehingga aplikasi dapat berjalan dengan cepat dari satu lingkungan komputasi ke lingkungan komputasi yang lain. Sehingga, dengan kata lain container dapat diasumsikan sebagai suatu wadah yang dapat menampung berbagai data dan aplikasi yang bertujuan untik membuatnya menjadi lebih ringkas dan efisien untuk berjalan pada sistem (Vaucher, 2015).
4. Pada pengujian failover ketika upload file sedang berlangsung pada masing-masing cluster, tidak ditemukan perbedaan. Pengujian tersebut sama-sama memberikan hasil yaitu web server gagal merespon atau koneksi terputus ketika merespon permintaan atau memproses pengunggahan tersebut. Kegagalan web server tersebut disebabkan oleh traffic yang berjalan ke satu node yang aktif lalu node tersebut mengalami crash atau down, maka diperlukan jeda agar koneksi dapat terhubung lagi ke web server dengan node yang masih tersedia.

DAFTAR RUJUKAN

[1] Bik, F.R. and Asmunin (2017) 'Implementasi Docker Untuk Pengelolaan Banyak Aplikasi Web (Studi Kasus : Jurusan Teknik Informatika UNESA)',

Jurnal Manajemen Informatika, 7(2), pp. 46–50.

[2] Kubernates, https://kubernetes.io/id/docs/concepts/workloads/controllers/_print/# membalikkan-deployment, Online, 05, Maret 2024

[3] Rexa, M., Data, M. and Yahya, W. (2019) 'Implementasi Load Balancing Server Web Berbasis Docker Swarm Berdasarkan Penggunaan Sumber Daya Memory Host', *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya*, 3(4), pp. 3478–3487.

[4] Riko Rudiyanto (2023) 'Implementasi Load Balancing Dan Failover Dengan Border Gateway Protocol Menggunakan Router Juniper (Studi Kasus Stasiun Jombang)', Program Studi Informatika Program Sarjana Fakulas Teknologi Informasi Universitas Teknologi Digital Indonesia Yogyakarta, pp. 5–24. Available at: https://eprints.utdi.ac.id/9832/3/3_155410081_BAB_II - riko rudiyanto.pdf.

[5] Prakoso, R.D. and Asmunin (2018) 'Implementasi dan Perbandingan Performa Proxmox Dalam Virtualisasi dengan Tiga Virtual Server', Jurnal Manajemen Informatika, 8(1), pp. 79–86. Available at: https://ejournal.unesa.ac.id/index.php/jurnal-manajemen-informatika/article/view/22864.

[6] Prasetyo, A. (2017) 'Perancangan Virtualisasi Replikasi Database Pada Arsitektur Cloud Computing', Research Report, 0(0), pp. 207–210. Available at: http://research-report.umm.ac.id/index.php/research-report/article/view/1213.

[7] Sumbogo, Y.T., Data, M. and Siregar, R.A. (2018) 'Implementasi Failover Dan Autoscaling Kontainer Web Server Nginx Pada Docker Menggunakan Kubernetes', Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2(12), pp. 6849–6854. Available at: http://j-ptiik.ub.ac.id.

[8] Vaucher, S. (2015) 'Comparing Virtual Machines and Linux Containers', Université de Neuchâtel [Preprint]. Available at: https://github.com/docker/docker/blob/master/.