IMPLEMENTASI CLUSTER WEB SERVER DINAMIS BERBASIS OPERATING SYSTEM-LEVEL VIRTUALIZATION MENGGUNAKAN DOCKER DAN KUBERNETES PADA API SIAKAD STIMATA

Faudji [1)], Weda Adistianaya Dewa [2)], Nasrul Firdaus [3)]
[(1,2,3)]Sistem Informasi, STMIK PPKIA Pradnya Paramita, Malang
leofaudji@gmail.com, weda@stimata.ac.id, nasrul@stimata.ac.id

Abstrak

Kebutuhan civitas kampus stimata akan informasi digital sangatlah tinggi. Kebutuhan akses penggunaan yang berkembang begitu pesat menyebabkan banyaknya tuntutan pada server untuk menjalankan tugasnya. Kemampuan server yang melebihi batas akan mengalami overload sehingga banyak dampak buruk yang akan ditimbulkan antara lain penurunan kecepatan waktu mengakses bahkan menimbulkan server mati. Arsitektur webserver dinamis berbasis container mulai banyak digunakan dikarenakan mampu mengatasi permasalahan yang terjadi pada server fisik. Hal tersebut mempengaruhi tingkat availability dalam menangani jumlah request data yang tinggi dan banyak. Penelitian ini melakukan implementasi cluster webserver dinamis menggunakan docker dan kubernetes. Cluster webserver ini diinstall dengan aplikasi(API) siakad yang terdiri dari 9 API. Hasil implementasi menunjukkan tingkat high availability webserver adalah 99,94%. Adapun penanganan failover membutuhkan rata-rata kurang dari 1 menit untuk mencapai uptime.

***Kata Kunci** : Availability, Virtualization, Container, Web Server, Kubernetes.*

Abstract

The need for the Stimata campus community for digital information protection is high. Need access to usage that is growing so rapidly causes the use of server to run. Server capabilities that exceed the limit will experiencing an overload so that many bad impacts will be caused, including: a decrease in access speed even causes the server to die. Architecture Container-based dynamic webservers are starting to be widely used because they are able to solve problems that occur on physical servers. It affects level of availability in handling high and large number of data requests. This study implements a dynamic webserver cluster using docker and kubernetes. This webserver cluster is installed with a Siakad application (API) consisting of 9 APIs. The implementation results show a high level of web server availability is 99.94%. The failover handling requires an average of less than 1 minute to reach uptime.

***Keyword** : Availability, Virtualization, Container, Web Server, Kubernetes.*

PENDAHULUAN

Latar Belakang

Kebutuhan civitas kampus stimata akan informasi digital sangatlah tinggi. Kemampuan server akan semakin menurun dengan banyaknya akses yang datang dari client. Kemampuan server yang melebihi batas akan mengalami overload sehingga banyak dampak buruk yang akan ditimbulkan antara lain penurunan kecepatan waktu mengakses bahkan menimbulkan server mati. Ketika server sudah mengalami overload banyak akibat yang terjadi antara lain penurunan

kecepatan saat user mengakses, user dapat ditolak untuk mengakses server, atau bahkan server menjadi mati. Sebagai mahasiswa STIMATA ingin mengakses suatu data seperti data absensi, nilai-nilai, administrasi pembayaran, jadwal mata kuliah maupun tugas-tugas, dan lain sebagainya. Akan tetapi data itu semua tidak bisa diakses secara real time sesuai keinginan mahasiswa atau hanya bisa diakses di momen-momen tertentu. Server belum ada sistem backup apabila ada gangguan yang terjadi di dalam server. Jika ada gangguan dari sisi server, maka aplikasi tidak bisa diakses. Arsitektur aplikasi masih menerapkan *single* server basis data. Hal tersebut menyebabkan ketidakmampuan dalam menangani permintaan data yang besar. Ketidakmampuan tersebut mengakibatkan server mati dan aplikasi tidak bisa diakses.

Rumusan Masalah

Permasalahan yang akan dibahas adalah : Bagaimana desain dan konfigurasi model cluster web server dinamis berbasis operating system-level virtualization untuk meningkatkan availability server dan sumber daya aplikasi di dalamnya ? Dan Bagaimana menurunkan jumlah gangguan pada server ?

Tujuan Penelitian

Berdasarkan permasalahan yang telah dirumuskan, maka penelitian ini memiliki tujuan, yaitu : Meningkatkan availability server dan sumber daya aplikasi di dalamnya. Dan menurunkan jumlah gangguan pada server.

TINJAUAN PUSTAKA

Availability

Availability atau ketersediaan artinya dalam konteks keamanan informasi upaya untuk menjaga agar sebuah sistem tetap bisa digunakan adalah hal penting yang perlu dilakukan. Kelangsungan sebuah data akan sangat bergantung pada pemeliharaan performa perangkat keras, perangkat lunak, dan saluran komunikasi yang digunakan untuk menyimpan dan memproses informasi. Ketika sebuah aplikasi terganggu dan tidak dapat diakses, maka dapat kehilangan banyak kepercayaan.

Menurut (Wendy Torell, 2004) mengatakan *"Assures that systems work promptly and service is not denied to authorized users"* yang artinya memastikan bahwa sistem dapat bekerja dengan segera dan layanan tidak ditolak untuk pengguna yang berwenang. Dikutip dari referensi lainnya (Deepali Mittal, 2015) *"Availability is the guarantee that information will be available to the consumer in a timely and uninterrupted manner when it is needed regardless of location of the user. This means that the cloud infrastructure, the security controls, and the networks connecting the clients and the cloud infrastructure should always be functioning correctly. Availability is ensured by: fault tolerance, authentication and network security"* artinya adalah jaminan bahwa informasi akan tersedia bagi konsumen secara tepat waktu dan tanpa gangguan saat dibutuhkan terlepas dari lokasi pengguna. Ini berarti bahwa infrastruktur, kontrol keamanan, dan jaringan yang menghubungkan klien dan infrastruktur server harus selalu berfungsi dengan benar. Ketersediaan dipastikan dengan :

toleransi kesalahan, otentikasi dan keamanan jaringan.

Load Balancing

Load Balancing adalah salah satu metode untuk mendistribusikan traffic ke beberapa server agar tidak membebani satu server saja. Alhasil, kekuatan server menjadi seimbang sehingga tidak terjadi overload.

Penelitian terdahulu yang dilakukan (Moch. Wahyu Imam Santosa, 2018) horizontal scalling dilakukan dengan menambahan unit pemrosesan seperti node, instance ataupun kontainer merupakan sebuah solusi dalam meningkatkan kinerja sebuah sistem dalam menangani banyaknya permintaan. Load balancing berjalan menggunakan fitur load balancer dan nodeport yang disediakan oleh kubernetes. Dengan terhubungnya setiap replika pada sebuah sistem penyimpanan terpusat menjadikan data bersifat konsisten.

Virtualisasi

Virtualisasi adalah konsep dimana akses ke sebuah hardware seperti server diatur sehingga beberapa operating system (guest operation system) dapat berbagi sebuah hardware. Tujuan dari virtualisasi adalah kinerja tingkat tinggi, ketersediaan, kehandalan, atau untuk membuat dasar keamanan dan manajemen yang terpadu. Menurut (Khasanah, 2019) pengembangan teknologi virtualisasi server dengan memanfaatkan sumber daya komputer yang dimiliki oleh instansi menjadi salah satu solusi untuk mengatasi permasalahan tersebut dalam menekan biaya yang dikeluarkan perusahaan untuk pembelian server fisik dan juga meningkatkan performance server.

Container

Container adalah paket atau aplikasi yang mengandalkan isolasi virtual untuk menjalankan aplikasi yang dapat menjalankan sistem operasi kernel secara simultan tanpa memerlukan mesin virtual (VMs). Container dapat langsung berjalan diatas Sistem Operasi (OS) tanpa menggunakan hypervisor. Seperti pada gambar berikut, container berjalan secara langsung diatas Sistem Operasi (OS). Container dapat membagi resource pada OS beserta fungsinya.

Docker

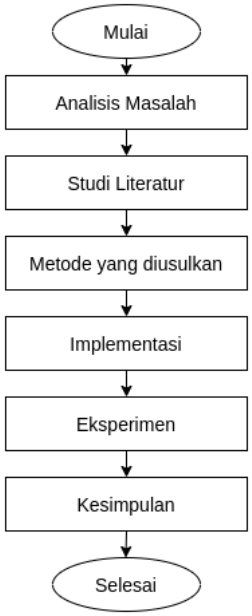
Menurut penelitian selanjutnya (Dwiyatno, 2020) docker juga sangat ringan dan cepat jika dibandingkan dengan mesin virtual yang berbasis hypervisor. Oleh karena itu, dilakukan implementasi virtualisasi berbasis docker container supaya dapat meningkatkan efektifitas dalam penggunaan sumber daya CPU dan memori pada server.

Kubernetes

Menurut penelitian lainnya selanjutnya (Kezia Yedutun, 2020) teknologi containerazitation ini diimplementasikan dengan memanfaatkan scalability. Scalability pada container akan menyesuaikan dengan kebutuhan client maupun server. Dengan teknologi container, akses terhadap server dapat dilakukan dengan penambahan container untuk jalur akses jika dirasa penuh dan akan mengurangi container jika jumlah client yang mengakses berkurang. Hal ini akan meminimalisir server yang overload sehingga client dapat melakukan akses terhadap server tanpa kesulitan akibat sedikitnya jalur akses yang ada.

METODE PENELITIAN

Tahapan Penelitian



Gambar 1. Metode Penelitian

Penelitian ini menggunakan load balancing dan failover sebagai metodenya. Dan gambar 1 diatas menggambarkan alur dari penelitian ini.

Analisis Permasalahan

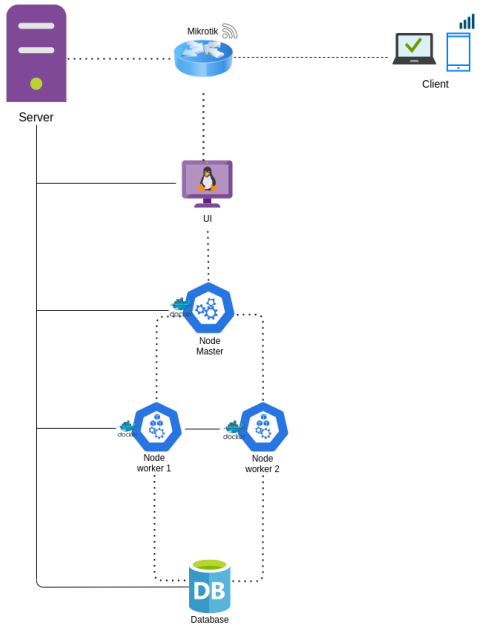
Kampus STIMATA memiliki beberapa aplikasi untuk menunjang kegiatan akademiknya. Aplikasi-aplikasi tersebut disimpan di beberapa lokasi seperti VPS maupun server lokal kampus. Server VPS dan server lokal sendiri sudah dikelola mandiri oleh petugas administrator server kampus. Sejauh ini kedua server tersebut sudah berjalan baik, namun ada beberapa kendala seperti sering terjadinya aplikasi tidak bisa diakses, server down, dan sebagainya. Sistem yang saat ini digunakan masih menerapkan arsitektur single server basis data. Hal tersebut menyebabkan ketidakmampuan server basis data dalam menangani permintaan data yang besar dan tugas administrator server menjadi lebih ekstra untuk selalu monitoring server supaya aplikasi tetap berjalan normal dan semestinya. Hal yang penting dalam kasus ini adalah sebuah arsitektur aplikasi yang bisa meningkatkan tingkat *availability* (ketersediaan) server untuk memitigasi bila sewaktu-waktu terjadi gangguan.

Studi Literatur

Tahapan ini membahas tentang dasar teori yang berguna untuk mendukung penelitian ini. Teori yang diperoleh dari berbagai sumber, diantaranya adalah jurnal, buku, serta yang paling terbanyak adalah internet.

Metode yang diusulkan

Pada tahap ini akan dirancang dan dibangun rancangan sistem yang didapat dari referensi penelitian yang sudah pernah dilakukan sebelumnya yang didapat dari studi literatur dan analisa kebutuhan sistem. Perancangan sistem dapat dilihat pada gambar berikut ini :



Gambar 2. Metode yang diusulkan

Implementasi

Bagian ini membahas tentang langkah dalam penyelesaian implementasi sistem. Adapun langkah-langkah tersebut antara lain :

1. Instalasi Docker
2. Instalasi Kubernetes
3. Instalasi Dashboard Kubernetes
4. Membuat Images Paket Aplikasi
5. Mengupload Images Paket Aplikasi
6. Instalasi aplikasi ke dalam cluster kubernetes
7. Instalasi Database Mysql

Eksperimen

Alat

Dalam penelitian ini diperlukan alat yang mampu membuat penelitian ini berhasil, berikut spesifikasinya :

Tabel 1. Spesifikasi perangkat keras

No	Keterangan
1	Laptop Intel Core i5 8250U @ 1.60 GHz RAM 8GB Hardisk 1TB, SSD 240GB
2	Mikrotik RB 941 2nd 1 Core 650Mhz Memory 32 Mb Hardisk 16 Mb Versi 6.47.9 long term

Bahan

Perancangan infrastruktur ini memerlukan bahan-bahan penunjang yang berupa perangkat lunak yang mendukung dalam penelitian ini. Bahan yang digunakan adalah :

1. VMWare untuk Virtual Machine (VM).
2. Ubuntu Server 18.04 LTS sebagai node-node.

Tabel 2. Spesifikasi Node

Keterangan	Spesifikasi
UI	Laptop Intel Core i5 8250U @ 1.60 GHz RAM 8GB Hardisk 1TB, SSD 240GB
Node Master	CPU 2 Core Memory 2 GB Hardisk 15 MB
Node Worker	CPU 2 Core Memory 1 GB Hardisk 15 MB
Storage	CPU 1 Core Memory 512 MB Hardisk 10GB

Tabel 3. Pembagian Port Aplikasi

Keterangan	IP
Mikrotik RB 941 2nd	DHCP Client
Network	100.1.1.0/24
Gateway	100.1.1.100
UI	100.1.1.50
Node Master	100.1.1.10
Node Worker 1	100.1.1.11
Node Worker 2	100.1.1.12
Database	100.1.1.20
Dashboard	100.1.1.10:30003
Database	100.1.1.20:3306
UI	100.1.1.50:8080
API User	100.1.1.100:30031

API Mahasiswa	100.1.1.100:30032
API Karyawan	100.1.1.100:30033
API Keuangan	100.1.1.100:30034
API KHS	100.1.1.100:30035
API KRS	100.1.1.100:30036
API Presensi	100.1.1.100:30037
API Sidang	100.1.1.100:30038
API Kurikulum	100.1.1.100:30039
API Arsip	100.1.1.100:30040

3. Docker Engine versi 20.10.5 sebagai container.
4. Kubernetes 1.21.0 sebagai administrator container.
5. Aplikasi Siakad.
6. Database MySQL.
7. Apache JMeter untuk alat pengujian.

Rancangan Pengujian

Ada tiga jenis pengujian yang dilakukan pada sistem, yakni pengujian fungsional, pengujian failover, dan pengujian high availability.

1. Skenario pertama pengujian fungsional
2. Skenario kedua pengujian failover
3. Skenario ketiga pengujian high availability

Kesimpulan

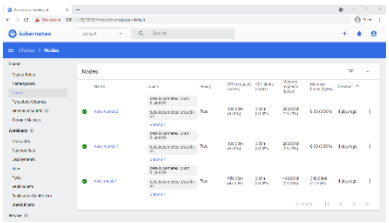
Kesimpulan didapatkan dari hasil pengujian dan hasil analisis. Bagian terakhir pada penulisan adalah pemberian saran yang bertujuan untuk bisa memperbaiki kesalahan dan memberi pertimbangan untuk penelitian selanjutnya.

PENGUJIAN DAN HASIL

Pengujian Fungsional

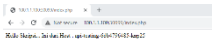
Pengujian fungsional dilakukan untuk mengetahui system cluster web server yang telah diimplementasikan pada kubernetes sudah berjalan dengan semestinya sesuai kebutuhan fungsional yang sudah dirancang. Berikut ini adalah proses pengujian fungsional yang dilakukan :

1. Akses web server melalui IP address



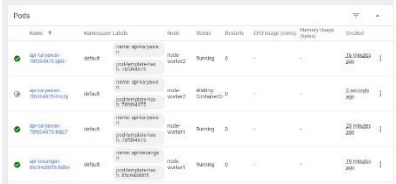
Gambar 3. Dashboard

2. Dynamic Web Server



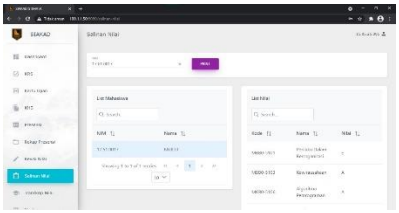
Gambar 4. Webserver

3. Backup Container



Gambar 5. Backup container

4. User Melakukan Select Data

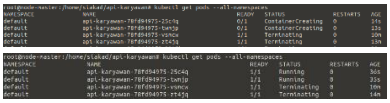


Gambar 6. Select Data

Pengujian Failover

Pengujian failover ini dilakukan untuk mengetahui berapa lama waktu yang dibutuhkan oleh layanan web server untuk melakukan failover dari suatu node yang dinonaktifkan ke node yang tersedia pada kubernetes. Berikut skenario pengujiannya :

1. Skenario pertama menonaktifkan node



Gambar 7. Menonaktifkan node

Berikut ini adalah hasil pengujian menonaktifkan node yang tersedia ditampilkan berupa table :

Tabel 4. Hasil pengujian

Keterangan	Runtime
Percobaan 1	6 menit
Percobaan 2	7 menit
Percobaan 3	6 menit
Percobaan 4	7 menit
Percobaan 5	7 menit
Rata-rata	**6.6 menit**

2. Skenario kedua menghapus aplikasi.

Skenario pengujiannya adalah dengan cara menghapus container atau aplikasi yang sedang berjalan untuk melihat respon waktu yang dibutuhkan service yang sedang berjalan untuk pindah ke node yang tersedia.

Tabel 5. Hasil pengujian

Keterangan	Runtime
Percobaan 1	55 detik
Percobaan 2	54 detik
Percobaan 3	41 detik
Percobaan 4	50 detik
Percobaan 5	46 detik
Rata-rata	**51.2 detik**

Pengujian High Availability

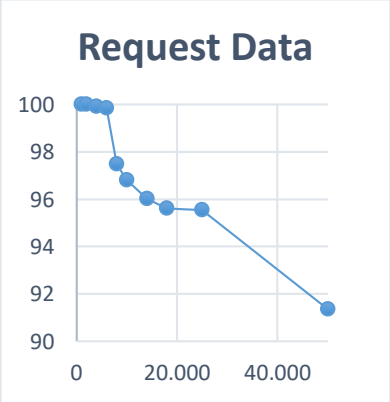
Pengujian high availability ini dilakukan untuk mengetahui ketersediaan layanan web server dalam menangani peningkatan load request yang berjalan pada kubernetes. Dalam pengujian ini, ada 2 skenario yang dilakukan untuk menguji load request pada web server yang sedang berjalan. Berikut proses pengujian yang dilakukan :

1. Pengujian HA 1 Webserver

Dalam pengujian ini, penguji memberikan HTTP request data terhadap layanan 1 webserver yang dijalankan. Prosedurnya adalah melakukan http request dengan jumlah paket mulai dari 1000 s/d 50.000 paket secara bergantian.

Tabel 6. Hasil pengujian

Keterangan	% Sukses
1000 request	100 %
2000 request	100 %
4000 request	99.97 %
6000 request	99.85 %
8000 request	97.47 %
10.000 request	96.80 %
14.000 request	96.01 %
18.000 request	95.62 %
25.000 request	95.54 %
50.000 request	91.35 %



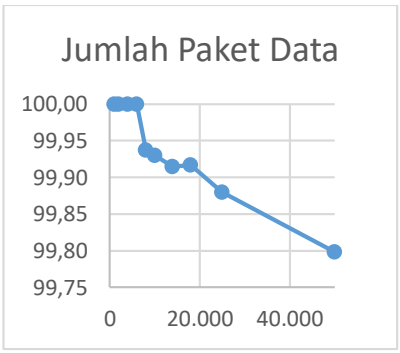
Gambar 8. Chart HA 1 Webserver

2. Pengujian HA 2 Webserver

Sama dengan skenario pertama, penguji memberikan HTTP request data terhadap layanan webserver yang dijalankan. Akan tetapi berjalan diatas 2 webserver yang load balancing. Prosedurnya adalah melakukan http request dengan jumlah paket mulai dari 1000 s/d 50.000 paket secara bergantian.

Tabel 7. Hasil Pengujian

Keterangan	% Sukses
1000 request	100 %
2000 request	100 %
4000 request	100 %
6000 request	100 %
8000 request	99.94 %
10.000 request	99.93 %
14.000 request	99.92 %
18.000 request	99.92 %
25.000 request	99.88 %
50.000 request	99.80 %



Gambar 9. Char HA 2 Webserver

Berdasarkan data tersebut performa availability mencapai 100% mengalami peningkatan yang semula dari 2000 paket (menggunakan 1 webserver) menjadi 6000 request paket data (menggunakan 2 webserver). Namun diatas itu terjadi penurunan availability ketika paket data yang dikirim mulai dari angka 8000. Penurunan ini bisa terjadi disebabkan oleh beberapa kemungkinan, seperti dari resource/spesifikasi hardware yang terbatas, banyaknya servis atau aplikasi yang sedang berjalan di dalam server. Solusi dari permasalahan tersebut adalah dengan menambah atau mengupgrade resource hardware dan menambah jumlah server.

KESIMPULAN DAN SARAN

Kesimpulan

Diperoleh data yaitu semakin banyak jumlah node/server yang dijalankan, akan semakin meningkat pula tingkat prosentase availability.

Performa availability mencapai puncak 100% pada jumlah request 2000 paket (1 webserver) dan 6000 paket (2 webserver). Terjadi penurunan availability ketika paket data yang dikirim mulai dari angka 4000 paket (menggunakan 1 webserver) dan 8000 paket (menggunakan 2 webserver). Penurunan ini bisa terjadi disebabkan oleh beberapa kemungkinan, seperti dari resource/spesifikasi hardware yang terbatas, banyaknya servis atau aplikasi yang sedang berjalan di dalam server. Solusi dari permasalahan tersebut adalah dengan menambah atau mengupgrade resource hardware dan menambah jumlah server.

Saran

Cluster web server dinamis bisa dikembangkan lagi dengan berbagai macam skenario yang lebih kompleks. Serta bisa memberikan hasil yang lebih baik pada percobaan lainnya.

Penelitian berikutnya dapat dilakukan dengan menggunakan lebih dari 1 node master untuk menghasilkan hasil performa availability yang lebih baik. Untuk hasil yang jauh lebih maksimal disarankan memakai server fisik dan server lebih dari satu.

DAFTAR PUSTAKA

Deepali Mittal, N. A. (2015). *Rev Paper On Fault Tolerance In Cloud Computing. IEEE.*

Dwiyatno, S. (2020). *Implementasi virtualisasi server berbasis docker container. Jurnal PROSISKO, 7, 2.*

Hakim A, R. M. (2018). *Implementasi Failover Clustering Server Untuk Mengurangi Resiko Downtime Pada Web Server. Jurnal AKRAB JUARA, 3(3), 76-82.*

Irwan, D. (2017). *Service Availability Dan Performa Sumber Daya Prosessor Pada Infrastruktur Server Virtual. Jurnal Penelitian Ilmu Komputer, 5(1), 42-50.*

Kezia Yedutun, A. N. (2020). *Implementasi Container Kubernetes untuk Mendukung Scalability. 1, 1.*

Khasanah, S. N. (2019). *Rancangan Virtualisasi Server Menggunakan VMWare Vsphere. Jurnal Evolusi, 7, 1.*

Moch Wahyu Imam Santosa, R. P. (2018). *Implementasi Load Balancing Server Basis Data Pada Virtualisasi Berbasis Kontainer. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2, 12*.

Muhamad Ikmal Wiawan, A. B. (2018). *Peningkatan Availability Infrastruktur Storage Pada Cluster Computing Menggunakan Metode Fault Tolerance*. *Jurnal Produktif, 2*, 41.

Saleh Dwiyatno, E. R. (2020). *Implementasi virtualisasi server berbasis docker container*. *Jurnal PROSISKO, 7*, 2.

Serverfault. (2020, May 20). *Failover cluster of 2+ servers for disasters*. Retrieved from Serverfault: https://serverfault.com/

Wendy Torell, V. A. (2004). *Mean Time Between Failure: Explanation and Standards.* APC.