Journal of Applied Informatics and Computing (JAIC)
Vol.x, No.x, Juli 201x, pp. xx~xx
e-ISSN: 2548-6861

BLUE GREEN DEPLOYMENT PADA OBSERVIUM DENGAN INFRASTRUKTUR REDUNDAN

Muhammad Faaiz Muazi [1]*, Antoni Haikal [2]**
* Teknik Informatika, Politeknik Negeri Batam
** Rekayasa Keamanan Siber, Politeknik Negeri Batam
faaizmuazi20@gmail.com [1], antoni@polibatam.ac.id [2]

Article Info

Article history:

Received …
Revised …
Accepted …

Keyword:

Blue-Green Deployment, Observium, High Availability, Redundant Infrastructure, Zero Downtime, Network Monitoring.

ABSTRACT

Ketersediaan layanan monitoring jaringan yang berkelanjutan merupakan aspek penting dalam menjaga stabilitas infrastruktur teknologi informasi. Observium sebagai sistem monitoring jaringan sering mengalami downtime ketika proses pembaruan aplikasi maupun sistem operasi dilakukan, sehingga berpotensi mengganggu visibilitas dan pengawasan jaringan secara real-time. Penelitian ini bertujuan untuk merancang dan mengimplementasikan metode *blue-green deployment* pada Observium dengan dukungan infrastruktur redundan guna meminimalkan downtime serta meningkatkan ketersediaan layanan. Arsitektur sistem dibangun menggunakan dua lingkungan aplikasi yang identik, yaitu lingkungan Blue dan Green, yang beroperasi secara bergantian. Proses pengalihan trafik dilakukan melalui *HaProxy* sehingga pembaruan sistem dapat dilakukan pada lingkungan yang tidak aktif tanpa mengganggu layanan yang sedang berjalan. Selain itu, arsitektur ini didukung oleh sistem database cluster dan mekanisme failover untuk menjaga konsistensi serta ketersediaan data apabila terjadi kegagalan pada salah satu node. Metode pengujian dilakukan dengan mengukur waktu switching layanan, ketersediaan sistem, serta mekanisme failover dan rollback selama proses deployment. Hasil penelitian diharapkan dapat menunjukkan bahwa penerapan *blue-green deployment* berbasis arsitektur redundan mampu meningkatkan keandalan sistem monitoring Observium serta mengurangi downtime selama proses pembaruan sistem.

I. PENDAHULUAN

Sistem monitoring jaringan merupakan komponen kritis dalam menjaga keberlangsungan operasional infrastruktur TI. Observium, sebagai *tools* monitoring, banyak digunakan untuk memantau kondisi perangkat jaringan secara *real-time*. Namun, arsitektur *single-server* yang umum diterapkan menimbulkan risiko *single point of failure* dan mengharuskan layanan dihentikan saat dilakukan pembaruan [9], sehingga visibilitas terhadap infrastruktur hilang pada momen yang justru paling rentan terhadap gangguan.

Blue-green deployment merupakan strategi yang memungkinkan dua lingkungan produksi identik beroperasi bergantian sehingga pembaruan dapat dilakukan tanpa menghentikan layanan aktif. Beberapa penelitian telah membuktikan efektivitas metode ini dalam mencapai *zero downtime* [5], sementara penelitian lain menunjukkan bahwa *database clustering* dengan mekanisme *failover* otomatis mampu meningkatkan ketersediaan layanan secara signifikan [6]. Namun demikian, integrasi kedua pendekatan tersebut secara khusus pada sistem monitoring jaringan belum banyak dikaji.

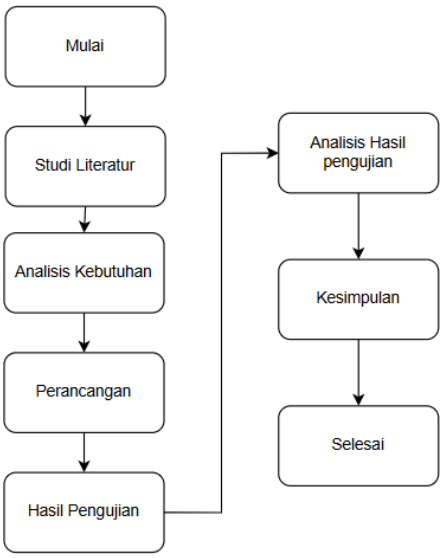
Penelitian ini menerapkan metode *blue-green deployment* yang dipadukan dengan arsitektur redundan pada Observium menggunakan HAProxy sebagai *traffic switcher*, MariaDB Galera Cluster untuk replikasi *database*, dan MaxScale sebagai *failover controller*. Pengujian dilakukan terhadap

e-ISSN: 2548-6861

empat parameter utama yaitu *availability*, durasi *downtime* saat *switching*, respon *rollback*, dan mekanisme *failover database* untuk mengukur efektivitas arsitektur yang dirancang.

II. METODE

A. Tahapan Penelitian



Gambar 1. Diagram alur penelitian

Penelitian ini menggunakan metode eksperimen dengan pendekatan kuantitatif [1]. Sistem dibangun pada lingkungan virtualisasi dan diuji melalui serangkaian skenario terkendali untuk mengukur performa arsitektur yang dirancang. Tahapan penelitian ini digambarkan dalam diagram alur pada Gambar 1, yang dimulai dari studi literatur untuk memahami konsep blue-green deployment, HAProxy sebagai traffic switcher, MariaDB Galera Cluster, serta MaxScale sebagai database proxy. Tahap selanjutnya adalah analisis kebutuhan untuk menentukan arsitektur sistem dan komponen yang dibutuhkan, kemudian dilakukan perancangan dan implementasi sistem, pengujian, analisis hasil pengujian, hingga penarikan kesimpulan.

Arsitektur yang diimplementasikan terdiri dari tiga lapisan utama. Pada lapisan proxy, HAProxy berfungsi sebagai traffic switcher yang mengatur distribusi trafik antara dua lingkungan aplikasi. Pada lapisan aplikasi, terdapat dua server identik yaitu Blue Server (Observium versi lama) dan Green Server (Observium versi baru) yang beroperasi secara bergantian. Pada lapisan database, digunakan MariaDB Galera Cluster yang terdiri dari primary node dan replica node dengan replikasi sinkron, serta MariaDB MaxScale sebagai database proxy yang menangani query routing dan failover otomatis.

Proses deployment dilakukan dengan menerapkan pembaruan pada lingkungan Green sementara Blue tetap melayani trafik. Setelah verifikasi berhasil, HAProxy mengalihkan trafik ke Green dengan memodifikasi bobot (weight) backend. Apabila terjadi kegagalan, trafik dikembalikan ke Blue melalui mekanisme rollback. Selama pengujian, sistem dimonitor secara kontinu menggunakan script cURL yang mengirimkan permintaan HTTP ke layanan Observium dengan interval satu detik. Setiap respons dicatat dalam bentuk log untuk mengidentifikasi waktu terjadinya kegagalan maupun pemulihan layanan. Nilai availability dihitung berdasarkan persamaan berikut:

Availability (%) = (Total *Uptime* / Total Waktu Pengujian) × 100%

di mana Total *Uptime* merupakan jumlah waktu ketika layanan Observium merespons permintaan HTTP dengan status berhasil. Durasi *downtime* dihitung dari selisih waktu antara respons terakhir yang berhasil hingga respons pertama yang kembali berhasil setelah terjadi gangguan [2].

B. Analisis Kebutuhan

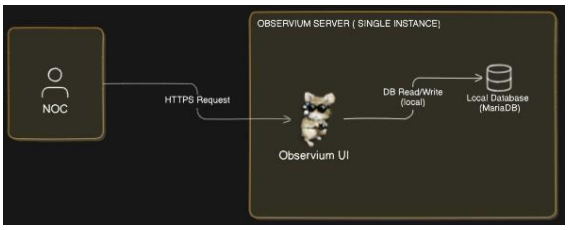
Implementasi arsitektur *blue-green deployment* pada penelitian ini memerlukan beberapa komponen. Sistem terdiri dari enam *instance* dengan spesifikasi yang disajikan pada tabel 1.

Tabel 1 Kebutuhan Hardware

Kebutuhan Hardware	Detail Spesifikasi
Prosesor	2 core (1 core, 1 socket)
Ram	2Gb
Storage	30Gb

C. Perancangan Sistem

Perancangan sistem dilakukan untuk mengatasi keterbatasan arsitektur *single-server* yang digunakan sebelumnya. Pada kondisi eksisting, aplikasi Observium dan *database* berjalan pada satu server tunggal tanpa mekanisme redundansi maupun lapisan *proxy*, sebagaimana ditunjukkan pada Gambar 2. Arsitektur ini menyebabkan seluruh layanan monitoring bergantung pada satu mesin, sehingga kegagalan pada server tersebut maupun proses pembaruan akan langsung menghentikan layanan.



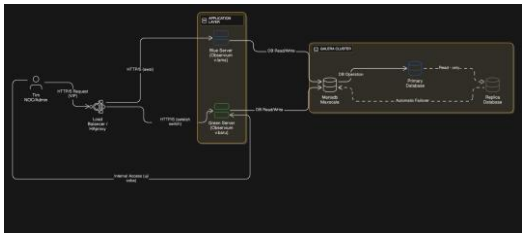
Gambar 2. Gambaran Umum Sistem

Sebelum arsitektur blue-green diterapkan, dilakukan pengukuran baseline pada arsitektur *single-server eksisting*. Pengujian baseline dilakukan dengan menjalankan cURL scripting yang mengirimkan permintaan HTTP setiap satu detik ke server Observium tunggal (172.16.100.66), kemudian service Apache2 dihentikan dan dijalankan kembali untuk mensimulasikan proses maintenance. Pengujian diulang sebanyak tiga iterasi untuk memperoleh data yang representatif. Hasil pengujian menunjukkan downtime konsisten selama 29–31 detik (rata-rata 30 detik) pada setiap iterasi, dengan seluruh permintaan HTTP selama periode tersebut mendapatkan respons HTTP 000 (connection refused), membuktikan bahwa arsitektur single-server tidak mampu mempertahankan ketersediaan layanan selama proses maintenance berlangsung. Hasil baseline ini menjadi acuan kuantitatif untuk mengukur efektivitas arsitektur *blue-green deployment* yang diusulkan.

```
ubuntu@AB:~$ cat /tmp/baseline_*.log | grep "HTTP 000" | head -5
09:01:56 - HTTP 000
09:01:57 - HTTP 000
09:01:58 - HTTP 000
09:01:59 - HTTP 000
09:02:00 - HTTP 000
ubuntu@AB:~$ cat /tmp/baseline_*.log | grep "HTTP 000" | tail -5
09:02:21 - HTTP 000
09:02:22 - HTTP 000
09:02:23 - HTTP 000
09:02:24 - HTTP 000
09:02:25 - HTTP 000
ubuntu@AB:~$
```

Gambar 3. *Log cURL saat downtime pada arsitektur single-server (HTTP 000 = connection refused)*

Arsitektur baru dirancang dengan menerapkan metode *blue-green deployment* yang dipadukan dengan infrastruktur redundan pada lapisan *database*, sebagaimana ditunjukkan pada Gambar 3. Trafik dari pengguna, dalam hal ini tim *Infrastructure Operation Center* (IOC), diarahkan melalui HAProxy yang berfungsi sebagai *traffic switcher*. HAProxy mengelola dua *backend* yaitu Observium yang ada pada server blue yang menjalankan Observium versi lama dan Observium yang ada pada server green untuk Observium versi baru. Dalam kondisi normal, HAProxy mengarahkan seluruh trafik ke server Blue dengan konfigurasi *weight* 100, sementara server Green tetap aktif dalam mode *standby* dengan *weight* 1 [8].



Gambar 4. Rancangan Topologi Sistem

Pada lapisan database, kedua server aplikasi terhubung ke MariaDB MaxScale yang berperan sebagai *database proxy*. MaxScale bertugas mengatur distribusi query dari aplikasi ke node database dalam cluster, seperti mengarahkan query berdasarkan kebijakan routing yang telah dikonfigurasi. Seluruh node database dikonfigurasikan menggunakan MariaDB Galera Cluster dengan mekanisme replikasi sinkron berbasis *multi-master*, sehingga setiap perubahan data yang terjadi pada satu node akan direplikasi secara langsung dan konsisten ke seluruh node lainnya sebelum transaksi di-*commit*. Dengan mekanisme ini, setiap node memiliki salinan data yang sama tanpa adanya perbedaan antara primary dan replica secara fisik. Ketika salah satu node mengalami kegagalan, node lain dalam cluster tetap dapat melayani permintaan tanpa memerlukan proses replikasi ulang, sementara MaxScale secara otomatis mengarahkan koneksi ke node yang masih tersedia.

Mekanisme *blue-green deployment* pada arsitektur ini bekerja dalam tiga tahap. Pertama, pembaruan aplikasi diterapkan pada lingkungan Green sementara Blue tetap melayani trafik produksi. Kedua, setelah pembaruan diverifikasi dan dinyatakan stabil, administrator mengubah konfigurasi *weight* pada HAProxy untuk mengalihkan seluruh trafik dari Blue ke Green. Ketiga, apabila terjadi kegagalan pada Green pasca-*deployment*, trafik segera dikembalikan ke Blue melalui mekanisme *rollback* dengan mengembalikan konfigurasi *weight* ke kondisi semula [7].

D. Implementasi Sistem

Implementasi dilakukan secara bertahap pada lingkungan virtualisasi dengan mengintegrasikan empat komponen utama. Dua instance Observium disiapkan sebagai lingkungan Blue dan Green, masing-masing dengan sistem operasi yang berbeda untuk mensimulasikan skenario upgrade. Lapisan database dikonfigurasi menggunakan MariaDB Galera Cluster dengan replikasi sinkron multi-master, dikelola oleh MaxScale sebagai database proxy dengan kemampuan automatic failover. HAProxy dikonfigurasi sebagai traffic switcher yang mengatur perpindahan trafik antar lingkungan tanpa menghentikan layanan aktif.

E. Pengujian Sistem

Pengujian sistem dirancang untuk mengevaluasi efektivitas arsitektur blue-green deployment yang telah diimplementasikan melalui beberapa skenario yang mencerminkan kondisi operasional nyata

e-ISSN: 2548-6861

Tabel 1. Definisi Skenario Pengujian Sistem

ID	Skenario	Kondisi Awal	Tindakan Uji	Kriteria Sukses	Hasil Aktual
TC-01	Switching Blue ke Green	Traffic 100% di Blue; Green standby (weight=1)	Eksekusi ./blue-green.sh switch; pantau cURL dan HAProxy stats	0 s downtime; HTTP 200 langsung diterima pada detik pertama pasca-switching	Downtime = 0 s; 326/326 request berhasil; backend berpindah ke Green seketika
TC-02	Failover DB primary node	server1=Master, server2=Slave; cluster wsrep_cluster_size=3	systemctl stop mariadb pada server1; pantau log MaxScale real-time	Replica dipromosikan otomatis ≤5 s; write tidak terganggu	Promosi dalam <1 s; notifikasi Telegram terkirim; aplikasi tetap melayani query
TC-03	Pemulihan DB node gagal	server1 down; server2=Master aktif; cluster size=2	systemctl start mariadb pada server1; pantau wsrep_local_state	Node bergabung kembali otomatis ≤60 s; status Synced; cluster size kembali 3	Bergabung dalam ~35 s; wsrep_local_state_comment=Synced; cluster normal
TC-04	Availability 24 jam (normal load)	Sistem operasional; cURL monitoring aktif setiap 1 detik ke 172.16.100.45	Monitor 24 jam; sertakan siklus TC-01, TC-02, TC-03 dalam sesi pengujian	Availability ≥99,9%; 0 failed request; avg response <200 ms	Availability 100% (326/326); avg response 66,31 ms; 0 downtime terukur
TC-05	Availability under high load (Apache Bench)	Sistem operasional; AB siap dengan 10.000 req, 50 concurrent users	ab -n 10000 -c 50 http://172.16.100.45/; eksekusi TC-01, TC-02, TC-03 selama AB berjalan	Failed requests = 0; throughput stabil; tidak ada HTTP error selama switching/failover	0 failed req / 10.000; throughput 34,88 req/s; P95=2.888 ms; P99=10.467 ms

1. Availability

   Pengujian availability dilakukan menggunakan dua instrumen secara paralel yaitu Uptime Kuma dan cURL scripting. Uptime Kuma terlebih dahulu dikonfigurasi untuk memantau seluruh VM yang terlibat dalam arsitektur mencakup instance Blue, Green, node MariaDB Galera Cluster, dan VM MaxScale, dengan interval pemantauan satu detik dan notifikasi alert melalui Telegram apabila salah satu server terdeteksi down. cURL scripting dikonfigurasi untuk mengirimkan permintaan HTTP ke endpoint layanan Observium dengan interval satu detik secara otomatis dan mencatat setiap respons yang diterima beserta timestamp dan status HTTP-nya. Kedua instrumen dijalankan secara bersamaan selama sesi pengujian switching dan rollback berlangsung sehingga data yang dihasilkan dapat saling memvalidasi.

2. Load Testing dengan Apache Bench

   Untuk melengkapi pengujian availability berbasis cURL scripting dan membuktikan stabilitas sistem di bawah beban trafik nyata, dilakukan pengujian beban menggunakan Apache Bench (ab). Pengujian dikonfigurasi dengan 10.000 total permintaan HTTP dan 50 concurrent users yang dikirimkan secara bersamaan ke endpoint HAProxy frontend (172.16.100.45). Selama pengujian beban berlangsung, skenario switching blue-green, rollback, dan failover database dieksekusi secara berurutan untuk mengamati dampaknya terhadap throughput dan response time. Parameter konfigurasi pengujian beban disajikan pada Tabel 2.

Tabel 2. Konfigurasi Apache Bench

Parameter Pengujian	Nilai	Keterangan
Total Permintaan	10.000	Flag -n 10000
Concurrent Users	50	Flag -c 50
Target endpoint	172.16.100.45	HAProxy frontend
Protocol	HTTP/1.1	GET request
Server Software	Apache/2.4.52	Backend Observium

3. Switching Blue Green
   Pengujian switching dilakukan dengan mengeksekusi perpindahan trafik dari server Blue ke server Green melalui HAProxy. Sebelum pengujian dimulai, kondisi awal sistem dipastikan bahwa seluruh trafik dilayani oleh server Blue dengan server Green dalam kondisi standby. cURL scripting dijalankan terlebih dahulu untuk memastikan pemantauan aktif sebelum perintah switching dieksekusi. Proses switching dilakukan menggunakan bash script dengan mengirimkan perintah set server ke HAProxy stats socket melalui *socat*, yang secara otomatis mengubah weight server Green menjadi 100 sebagai server aktif dan weight server Blue menjadi 1 sebagai standby tanpa melakukan restart layanan.

4. Rollback Green Blue
   Pengujian rollback dilakukan segera setelah skenario switching selesai dieksekusi, dengan kondisi awal trafik aktif di server Green. Rollback disimulasikan sebagai kondisi kegagalan pasca-deployment yang mengharuskan sistem dikembalikan ke lingkungan sebelumnya. Proses rollback dieksekusi menggunakan bash script yang sama dengan switching namun dengan parameter berbeda, yaitu mengembalikan weight server Blue menjadi 100 sebagai server aktif dan weight server Green menjadi 1 sebagai standby melalui HAProxy stats socket tanpa restart layanan, sementara cURL scripting tetap berjalan secara paralel untuk memantau kontinuitas layanan selama proses berlangsung.

5. Failover Database

   Pengujian *failover* dilakukan dengan mensimulasikan kegagalan pada *primary node*. Pemantauan dilakukan melalui log MariaDB MaxScale. Hasil log menunjukkan bahwa ketika *primary node* mengalami kegagalan, *replica node* secara otomatis dipromosikan dari peran *Slave* menjadi *Master* tanpa intervensi manual. Selanjutnya, *node* yang sempat gagal kembali *online* dan bergabung ke *cluster* dengan status tersinkronisasi, sementara *node* yang sebelumnya dipromosikan otomatis kembali ke peran *Slave*. Hasil ini menunjukkan kemampuan *self-healing* dari MariaDB Galera Cluster yang didukung MaxScale dalam menjaga ketersediaan lapisan *database*.

III. HASIL DAN PEMBAHASAN

A. Implementasi Sistem

Pada tahap implementasi, arsitektur blue-green deployment diterapkan pada lingkungan virtualisasi dengan mengintegrasikan empat komponen utama yaitu HAProxy sebagai traffic switcher, MariaDB Galera Cluster sebagai lapisan database redundan, MaxScale sebagai database proxy dengan automatic failover, serta dua instance Observium sebagai lingkungan Blue dan Green. Implementasi dilakukan secara bertahap dimulai dari konfigurasi masing-masing komponen hingga pengujian integrasi keseluruhan sistem. Dengan arsitektur tersebut, proses pembaruan sistem monitoring dapat dilakukan tanpa menghentikan layanan aktif sehingga ketersediaan layanan tetap terjaga selama siklus deployment berlangsung.

1. Implementasi HAProxy
   HAProxy dikonfigurasi sebagai traffic switcher yang menerima trafik HTTP dan meneruskannya ke backend aplikasi Observium. Frontend *http_front* diarahkan ke backend *observium_cluster* yang mendaftarkan dua server yaitu server Blue (172.16.100.41) dan server Green (172.16.100.40) dengan mekanisme health check aktif pada endpoint `/`. Proses switching dan rollback dilakukan secara dinamis tanpa restart layanan melalui stats socket di */var/run/haproxy.sock* dengan permission mode *660* dan level *admin*, sehingga perintah *enable/disable server* dapat dikirimkan langsung ke HAProxy pada saat runtime. Selain itu, halaman statistik tersedia pada port *8404* untuk pemantauan status aktif atau standby masing-masing server Blue dan Green secara real-time.

```
stats socket /var/run/haproxy.sock mode 660 level admin
```

Gambar 4. Stats socket

2. Implementasi Mariadb dan Galera Cluster
   MariaDB Galera Cluster dikonfigurasi dengan topologi dua node database aktif, ditambah satu Galera Arbitrator (garbd) yang dijalankan pada VM yang sama dengan MaxScale. Garbd tidak menyimpan data maupun mereplikasi transaksi, melainkan hanya berpartisipasi dalam mekanisme voting quorum untuk mencegah split-brain saat salah satu node database mengalami kegagalan [4]. Konfigurasi mengaktifkan wsrep provider Galera dengan format binlog row-based dan InnoDB sebagai storage engine, di mana klaster diinisialisasi dari node primary kemudian node replica

e-ISSN: 2548-6861

bergabung dan melakukan sinkronisasi data secara otomatis. Keberhasilan implementasi dikonfirmasi melalui status ***wsrep_cluster_size =3, wsrep_cluster_status = Primary***, dan ***wsrep_local_state_comment = Synced*** yang menunjukkan bahwa kedua node database dan garbd telah membentuk klaster yang beroperasi penuh.

```
wsrep_on                = ON
wsrep_provider          = /usr/lib/galera/libgalera_smm.so
wsrep_cluster_name      = "MariaDB Galera Cluster"
wsrep_cluster_address   = "gcomm://172.16.100.42,172.16.100.43,172.16.100.44"
wsrep_slave_threads     = 4
wsrep_node_name         = node1
wsrep_node_address      = "172.16.100.42"
```

Gambar 5. Konfigurasi wsrep provider

3. Implementasi MariaDB MaxScale

MaxScale dikonfigurasi sebagai database proxy pada VM yang sama dengan Galera Arbitrator (garbd). Modul ***readwritesplit*** digunakan untuk mengarahkan query write ke node master dan query read ke node slave secara otomatis. Pemantauan kondisi klaster dilakukan oleh ***Galera-Monitor*** dengan interval 2000ms menggunakan parameter ***use_priority=true*** sehingga promosi master mengikuti nilai priority yang telah ditetapkan, yaitu server1 (172.16.100.42) dengan priority 1 dan server2 (172.16.100.43) dengan priority 2. Layanan read-write splitting dikonfigurasi ***master_failure_mode=fail_on_write*** yang memastikan operasi write segera dialihkan saat master mengalami kegagalan.

```
[Galera-Monitor]
type=monitor
module=galeramon
servers=server1,server2
monitor_interval=2000ms
use_priority=true

[Read-Write-Service]
type=service
router=readwritesplit
servers=server1,server2
master_failure_mode=fail_on_write
```

Gambar 6. Nilai priority

B. Pengujian Sistem dan Hasil

Pengujian dilakukan terhadap empat skenario utama untuk mengevaluasi efektivitas arsitektur yang telah diimplementasikan, yaitu pengujian availability, switching blue-green, rollback, dan failover database. Setiap skenario dijalankan secara terpisah dengan instrumen pemantauan yang sesuai guna memperoleh data yang akurat dan dapat dipertanggungjawabkan.

1. Skenario Pengujian Availability

Pengujian availability diawali dengan menjalankan cURL scripting yang menargetkan endpoint layanan Observium di 172.16.100.45. Script secara otomatis mengirimkan permintaan HTTP setiap satu detik, mencatat timestamp, kode respons HTTP, dan backend yang melayani permintaan berdasarkan header x-backend pada setiap respons yang diterima. Script juga mendeteksi perpindahan backend secara otomatis dengan membandingkan nilai backend pada respons sebelumnya, serta mencatat dan menghitung total downtime apabila terjadi kegagalan koneksi. Seluruh hasil dicatat ke dalam file log secara real-time dan dihitung nilai availability-nya pada saat monitoring dihentikan.

Secara paralel, Uptime Kuma yang telah dikonfigurasi memantau seluruh VM yang terlibat dalam arsitektur dan mengirimkan notifikasi alert melalui Telegram apabila salah satu server terdeteksi dalam kondisi down.

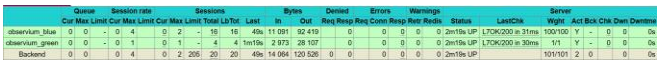
Pengujian beban dilakukan menggunakan Apache Bench dengan 10.000 permintaan HTTP dan 50 concurrent users yang dikirimkan ke endpoint 172.16.100.45. Pengujian berlangsung selama 286,720 detik dan menghasilkan 0 failed requests dari 10.000 total permintaan, membuktikan stabilitas sistem di bawah beban trafik tinggi. Throughput yang dihasilkan sebesar 34,88 request per detik dengan rata-rata response time 1.433,6 ms. Hasil selengkapnya disajikan pada Tabel 3.

Tabel 3. Hasil Pengujian Beban Apache Bench (10.000 Request, 50 Concurrent Users)

Parameter	Nilai
Total Permintaan	10.000
Failed Request	0(0%)
Throughput	34,88 req/detik
Rata-rata Response Time	1.433,6 ms
Median Response Time (P50)	744 ms
90th Percentile (P90)	2.517 ms
95th Percentile (P95)	2.888 ms
99th Percentile (P99)	10.467 ms
Durasi Pengujian	286,720 detik
Transfer Rate	153,33 KB/detik

1. Skenario Pengujian Switching Blue-Green

Pengujian switching diawali dengan mengkonfirmasi kondisi awal sistem melalui dashboard HAProxy, di mana kolom *Weight* menunjukkan **observium_blue** dengan nilai 100 sebagai server aktif dan **observium_green** dengan nilai 1 sebagai standby. cURL scripting kemudian dijalankan terlebih dahulu untuk memastikan pemantauan aktif sebelum switching dieksekusi. Proses switching dilakukan dengan menjalankan perintah **./blue-green.sh switch** yang mengirimkan perintah **set server** ke HAProxy stats socket melalui **socat**, mengubah weight server Green menjadi 100 sebagai server aktif dan weight server Blue menjadi 1 sebagai standby secara instan tanpa restart layanan.

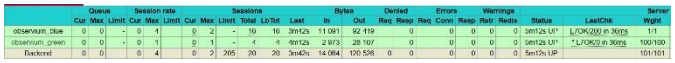


	Queue			Session rate			Sessions					Bytes		Denied		Errors		Warnings		Server						
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme
observium_blue	0	0	-	0	4		0	2	-	16	16	49s	11 091	92 419	0		0	0	0	0	0	2m19s UP	L7OK/200 in 31ms	100/100	Y	-	0	0	0s
observium_green	0	0	-	0	1		0	1	-	4	4	1m19s	2 973	28 107	0		0	0	0	0	0	2m19s UP	L7OK/200 in 30ms	1/1	Y	-	0	0	0s
Backend	0	0		0	4		0	2	205	20	20	49s	14 064	120 526	0	0		0	0	0	0	2m19s UP		101/101	2	0		0	0s

Gambar 7. Dashboard HAProxy sebelum switching

2. Skenario Pengujian Rollback Green-Blue

Pengujian rollback diawali dengan mengkonfirmasi kondisi awal sistem melalui dashboard HAProxy, di mana kolom *Weight* menunjukkan **observium_green** dengan nilai 100 sebagai server aktif dan **observium_blue** dengan nilai 1 sebagai standby setelah proses switching sebelumnya berhasil dieksekusi. Rollback kemudian dilakukan dengan menjalankan perintah **./blue-green.sh rollback** yang mengirimkan perintah **set server** ke HAProxy stats socket untuk mengembalikan weight server Blue menjadi 100 sebagai server aktif dan weight server Green menjadi 1 sebagai standby. cURL scripting tetap berjalan secara paralel selama proses berlangsung untuk memantau apakah terdapat kegagalan koneksi pada saat pengembalian trafik dieksekusi.



	Queue			Session rate			Sessions					Bytes		Denied		Errors		Warnings		Server						
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght
observium_blue	0	0	-	0	1		0	2	-	16	16	3m12s	11 091	92 419	0		0	0	0	0	0	3m12s UP	L7OK/200 in 30ms	1/1
observium_green	0	0	-	0	1		0	1	-	4	4	4m12s	2 973	28 107	0		0	0	0	0	0	3m12s UP	* L7OK/0 in 36ms	100/100
Backend	0	0		0	4		0	2	205	20	20	3m42s	14 064	120 526	0	0		0	0	0	0	5m12s UP		101/101

Gambar 8. Dashboard HAProxy saat rollback

3. Skenario pengujian Failover Database

Pengujian failover database diawali dengan mengkonfirmasi kondisi awal cluster melalui dashboard MaxScale, di mana Galera-Monitor menampilkan status Running dengan server1 (172.16.100.42) berstatus Master Synced Running dan server2 (172.16.100.43) berstatus Slave Synced Running.

MONITOR (1)	STATE	SERVERS (2)	ADDRESS	CONNECTIONS	STATE	GTID	SERVICES (1)
Galera-Monitor	Running	server1	172.16.100.42:3306	0	Down		Read-Write-Service
		server2	172.16.100.43:3306	0	Master, Synced, Running		Read-Write-Service

Gambar 9. Dashboard MaxScale

Kegagalan primary node kemudian disimulasikan dengan menjalankan perintah **systemctl stop mariadb** pada server1. Setelah perintah dieksekusi, log MaxScale dipantau secara real-time melalui perintah **tail -f /var/log/maxscale/maxscale.log** untuk mengamati urutan kejadian mulai dari deteksi kegagalan, promosi replica menjadi master, hingga pemulihan node yang gagal kembali ke cluster.

```
(log_connect_error): Monitor was unable to connect to server server1[172.16.100.42:330
6] : 'Can't connect to server on '172.16.100.42' (115)'
(log_state_change): Server changed state: server1[172.16.100.42:3306]: master_down. [Mas
ter, Synced, Running] -> [Down]
(log_state_change): Server changed state: server2[172.16.100.43:3306]: new_master. [Slav
e, Synced, Running] -> [Master, Synced, Running]: cluster_size: 3 -> 2
```

Gambar 10. Log Maxscale

C. Analisis Hasil Pengujian

Hasil pengujian sistem mencakup empat aspek pengujian yaitu availability, switching blue-green, rollback, dan failover database. Hasil pengukuran availability menggunakan Uptime Kuma menunjukkan nilai 100% selama periode 24 jam dengan rata-rata waktu respons 66,31 ms. Hasil tersebut dikonfirmasi oleh cURL scripting yang mengirimkan 326 permintaan HTTP selama sesi pengujian, di mana seluruh permintaan mendapatkan respons HTTP 200 tanpa satu pun kegagalan sehingga availability terhitung sebesar (326/326) × 100% = 100%. Pada pengujian switching, perpindahan trafik dari server Blue ke server Green melalui HAProxy berlangsung instan dengan downtime 0 detik, dibuktikan dengan respons HTTP 200 yang langsung diterima pada detik pertama setelah perintah switching dieksekusi. Pengujian rollback menunjukkan adanya flicker singkat sekitar dua detik akibat propagasi konfigurasi HAProxy, namun seluruh permintaan pada periode tersebut tetap mendapatkan respons 200 sehingga ketersediaan layanan tidak terganggu dari sisi pengguna. Pada pengujian failover database, MaxScale berhasil mempromosikan replica menjadi master dalam waktu kurang dari satu detik setelah primary node terdeteksi gagal, dan node yang sempat gagal berhasil kembali bergabung ke cluster dalam kondisi tersinkronisasi dalam waktu sekitar 35 detik tanpa intervensi manual, disertai notifikasi alert yang terkirim melalui Telegram secara real-time. Secara keseluruhan, hasil pengujian membuktikan bahwa arsitektur blue-green deployment yang diintegrasikan dengan MariaDB Galera Cluster dan MaxScale efektif mengeliminasi downtime pada proses pembaruan sistem sekaligus menjaga ketersediaan lapisan database secara otomatis, sehingga permasalahan single point of failure yang menjadi latar belakang penelitian ini berhasil diatasi.

e-ISSN: 2548-6861

III. KESIMPULAN

Penelitian ini berhasil mengimplementasikan arsitektur blue-green deployment pada sistem monitoring jaringan Observium dengan mengintegrasikan HAProxy sebagai traffic switcher, MariaDB Galera Cluster sebagai database redundan dengan replikasi sinkron multi-master, Galera Arbitrator sebagai penjaga kuorum, dan MaxScale sebagai database proxy dengan automatic failover. Hasil pengujian membuktikan sistem mampu mempertahankan ketersediaan 100% selama 24 jam dengan rata-rata waktu respons 66,31 ms, di mana seluruh 326 permintaan HTTP mendapatkan respons 200 tanpa satu pun kegagalan. Proses switching dan rollback berlangsung instan tanpa downtime, failover database terjadi dalam kurang dari satu detik, dan notifikasi alerting melalui Telegram berjalan secara real-time. Secara keseluruhan, integrasi keempat komponen tersebut terbukti efektif dalam mewujudkan sistem monitoring jaringan yang dapat diperbarui tanpa downtime sekaligus memiliki ketahanan terhadap kegagalan pada lapisan aplikasi maupun database.

UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan kehadirat Allah Subhanahu wa Ta'ala, yang telah melimpahkan rahmat, hidayah, serta karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik. Shalawat dan salam senantiasa tercurahkan kepada Nabi Muhammad SAW, yang menjadi teladan bagi seluruh umat manusia.

Ucapan terima kasih yang sebesar-besarnya penulis sampaikan kepada kedua orang tua tercinta, Ayahanda dan Ibunda, atas segala doa, kasih sayang, pengorbanan, dan dukungan yang tak pernah berhenti mengalir. Setiap langkah yang penulis tempuh selalu teriring oleh doa-doa tulus mereka. Semoga Allah SWT senantiasa menjaga, melindungi, dan membalas seluruh kebaikan mereka dengan balasan terbaik di dunia dan akhirat.

Daftar Pustaka

[1] A. Hasiholan and M. Fani, “Analisis Komparatif Deteksi Malware pada SIEM Wazuh Melalui Integrasi VirusTotal dan Yara : Studi Kasus File Upload di Rootguards,” no. x, pp. 1–8.

[2] Nurdiansyah, D., & D.P, A. H. (2024, August 16). Analysis of Maintenance Management at PT.XYZ Power Plant (With MTTR, MTTF, and Availability Factors) and Development of Performance Improvement Program. *Formosa Journal of Multidisciplinary Research (FJMR)*, *Vol 3*(No 9), 3363–3376. 10.55927/fjmr.v3i9.11137

[3] Taft, R., Sharif, I., Matei, A., VanBenschoten, N., Lewis, J., Grieger, T., Niemi, K., Woods, A., Birzin, A., Poss, R., Bardea, P., Ranade, A., Darnell, B., Gruneir, B., Jaffray, J., Zhang, L., & Mattis, P. (2020). CockroachDB: The resilient geo-distributed SQL database. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20) (pp. 1493–1509). ACM. https://doi.org/10.1145/3318464.3386134

[4] Cossu, G., Tynan, J., Lopez, F., Morant, S., Enrico, M., Bochow, B., Biasi, I., Nisbet, B., Creti, S., Hughes-Jones, R., Cristellotti, C., Gonzalez, S., Ozdemir, O., Sosa, R., Guenther, T., & Fuentes, A. (2014). D5.2: XIFI Core Backbone (v1.0). eXperimental Infrastructures for the Future Internet (XIFI), Grant Agreement No. 604590, FP7-2012-ICT-FI. European Commission.

[5] Jayawardana, H. M. D. T. An enhanced blue-green deployment for reducing cost and application downtime [MSc research report, University of Moratuwa, Sri Lanka].

[6] Aspriyono, H. Pengembangan server Siakad Universitas Dehasen Bengkulu menggunakan high availability clustering dan MySQL database replication. Jurnal Ilmu Komputer, 1(2), 104–113.

[7] Idowu, M. A deep dive into blue-green and canary deployments: Benefits, challenges, and best practices.

[8] Rawls, C., & Amini Salehi, M. Load balancer tuning: Comparative analysis of HAProxy load balancing methods. *arXiv preprint arXiv:2212.14198*.

[9] Bhagawati, P., & Priya, N. (2021). A study on replication and failover cluster to maximize system uptime. *International Journal of Trend in Scientific Research and Development, 5*(4), 377–379. https://www.ijtsrd.com/papers/ijtsrd41249.pdf