ANALISIS KINERJA SISTEM *AUTO SCALING* PADA SISTEM WEB SERVER BERBASIS *CLUSTERING* MENGGUNAKAN SISTEM VIRTUAL

TUGAS AKHIR

Disusun dalam rangka memenuhi salah satu persyaratan

Untuk menyelesaikan program Strata-1 Departemen Teknik Informatika

Fakultas Teknik Universitas Hasanuddin

Makassar

Disusun Oleh :

MUHAMMAD ZULFACHRIL ASIARI

D421 15 316

DEPARTEMEN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

MAKASSAR

2022

LEMBAR PENGESAHAN SKRIPSI

ABSTRAK

Arus teknologi yang semakin cepat menyebabkan beban *traffic* yang semakin meningkat pada *web server*. Hal ini dapat menyebabkan beban kerja pada suatu *server* menjadi kelebihan beban. Oleh karena itu, dibutuhkan *web server* yang dapat mengelola ratusan bahkan ribuan data per detik agar tidak terjadi *server down* dan untuk menekan waktu *down time*. Penelitian ini bertujuan membuat suatu sistem web server menggunakan metode *auto scaling* yang responsif dan memiliki *high availability* pada sistem virtual AWS, sehingga dapat menangani jumlah *request* yang bertambah banyak melebihi kemampuan sistem virtual sebelumnya.

Sistem dibuat dengan mengambil contoh kasus website *e-commerce* yang dapat di akses oleh *client*, dimana website tersebut di-*setup* di atas sistem virtual menggunakan metode *auto scaling* pada *cloud service* AWS. Sedangkan untuk website *e-commerce* dibuat dengan menggunakan *framework* laravel dengan implementasi *progressive web apps* (PWA). Selain itu sistem virtual di hubungkan ke service *monitoring tools* seperti *Newrelic* dan *Uptime robot* sehingga utilitas yang digunakan pada sistem virtual dapat di-*monitoring* secara *real time.*

Pengujian sistem terbagi atas dua, yaitu pengujian menggunakan sistem virtual dan pengujian *user testing.* Pengujian dengan sistem virtual dilakukan menggunakan 5 *instance* server dengan jumlah *request* 1, 5, 10, 15, 30, dan 60 per detik yang masing-masing diuji selama 900 detik (15 menit). Pengujian pada *auto scaling* menunjukkan hanya jumlah *request* 1/detik dan 5/detik yang dapat diproses dengan *high availibility*, sedangkan request dengan jumlah lainnya tidak berhasil mempertahankan *high availability* dikarenakan proses *warm up* dari *instance* servernya yang lambat. Berbeda dengan menggunakan *clustering load balancing*, pada pengujian *load balancer* sistem virtual memperlihatkan kelebihannya untuk membagi rata semua *request* ke seluruh *instance* yang sudah aktif sejak awal. Sehingga pada semua pengujian, sistem virtual menggunakan metode *load balancing* tidak pernah mengalami gagal akses.

Hasil pengujian *user testing* sistem web server yang dilakukan selama 6 jam dapat dan dilakukan oleh 30 *user* menunjukkan bahwa semua halaman berhasil diakses oleh pengguna dengan *success rate* mencapai 100% dengan menggunakan metode sistem virtual *clustering auto scaling*. Selama proses pengujian *user testing*, jumlah *instance* server tertinggi yang dinyalakan secara otomatis oleh *auto scaling* adalah 3 *instance* dan jumlah instance server terendah adalah 1 *instance*.

Kata Kunci : web server, *clustering, auto scaling, load balancing,* AWS.

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh.

Segala puji dan syukur kami panjatkan ke hadirat Allah S.W.T Tuhan Yang Maha Esa yang dengan limpahan rahmat dan hidayah-Nya sehingga tugas akhir dengan judul "Analisis Kinerja Sistem *Auto Scaling* Pada Sistem Web Server Berbasis *Clustering* Menggunakan Sistem Virtual" ini dapat diselesaikan sebagai salah satu syarat dalam menyelesaikan jenjang Strata-1 pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Dalam penyusunan penelitian ini disajikan hasil penelitian terkait judul yang telah diangkat dan telah melalui proses pencarian dari berbagai sumber baik jurnal penelitian, prosiding pada seminar-seminar nasional/internasional, buku maupun dari situs-situs di internet.

Penulis menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, mulai dari masa perkuliahan sampai dengan masa penyusunan tugas akhir, sangatlah sulit untuk menyelesaikan tugas akhir ini. Oleh karena itu, pada kesempatan ini penulis menyampaikan ucapan terima kasih sedalam-dalamnya kepada:

1) Tuhan Yang Maha Esa atas semua berkat, karunia serta pertolongan-Nya yang tiada batas, yang telah diberikan kepada penulis disetiap langkah dalam pembuatan program hingga penulisan laporan skripsi ini.
2) Kedua orang tua penulis serta saudari-saudara penulis, serta keluarga yang senantiasa memberikan kekuatan, inspirasi, motivasi, bimbingan moral, materi, kepercayaan dan kasih sayang yang tidak terbatas kepada penulis.

3) Bapak Dr. Adnan, S.T., M.T., Ph.D., selaku pembimbing 1 yang telah banyak memberi bimbingan, inspirasi, motivasi, dan masukan yang bermanfaat kepada penulis.
4) Bapak Dr. Eng Zulkifli Tahir, S.T., M.Sc., selaku pembimbing II yang telah banyak memberi keyakinan, perhatian, bimbingan, motivasi, dan masukan yang bermanfaat kepada penulis.
5) Bapak Ir. Christoforus Yohannes, M.T., dan Dr.Eng. Ady Wahyudi Paundu, S.T., M.T., selaku dosen penguji yang telah memberikan saran sehingga laporan skripsi ini menjadi lebih baik.
6) Bapak Dr. Amil Ahmad Ilham, S.T., M.IT., Ph.D., selaku Ketua Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin atas bantuan dan bimbingannya selama masa perkuliahan penulis.
7) Bapak Robert dan Bapak Zainuddin serta segenap staf Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah membantu kelancaran penyelesaian tugas akhir penulis.
8) Kepada teman-teman penulis Kak Ulya, Kak Nunu, dan Evita yang telah memberikan bantuan dalam proses penyelesaian Tugas Akhir ini;
9) Kepada teman-teman komunitas Aeon yang selalu memberikan dukungan dan menghibur penulis dalam proses pengerjaan tugas akhir.
10) Kepada keluarga Sapu Lidi (Fuad Khairi Hamid, Jusmiati, Muhammad Arief Wicaksono, Laura Natalia Nainggolan, Fadel Rezky Ramadhan, Charina, Khusnul Khatima, Reka Regina, Ryan Rafli, Sabtian Juliana, Said Syamil Amas, dan Umniyah Nur Aprilyah), yang telah memberikan semangat dan menjadi

teman penulis selama menjalani perkuliahan sampai dengan penyelesaian Tugas Akhir.

11) Teman-teman Lab UBICON, yang telah memberikan dukungan dan semangat;
12) Teman-teman HYPERV15OR FT-UH atas dukungan dan semangat yang telah diberikan selama ini.
13) Serta seluruh pihak yang tidak sempat penulis sebutkan satu persatu, yang telah meluangkan waktu, tenaga dan pikiran selama penyusunan laporan Tugas Akhir ini.

Akhir kata, penulis berharap semoga Tuhan Yang Maha Kuasa berkenan membalas segala kebaikan dari semua pihak yang telah banyak membantu. Semoga Tugas Akhir ini dapat memberikan manfaat dan menambah wawasan pembaca khusus.

Gowa, Oktober 2021

Penulis,

Muhammad Zulfachril Asiari

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Seiring dengan berkembangnya teknologi informasi, maka berkembang pula kebutuhan pengguna, salah satunya adalah meningkatnya kebutuhan permintaan informasi di internet atau *website*. Dengan meningkatnya permintaan informasi di internet tersebut menyebabkan beban *traffic* pada web server meningkat yang dapat menyebabkan beban kerja pada suatu layanan web server menjadi kelebihan beban, sehingga ketersediaan web server yang mempunyai kinerja sangat handal yang dapat mengelola ratusan bahkan ribuan data per detik agar tidak terjadi server *down* merupakan permasalahan yang paling sering dihadapi untuk memenuhi kebutuhan akan layanan tersebut.

Selain itu, dengan memiliki server dengan kinerja yang sangat handal dapat menekan waktu *down time* (gangguan) atau jika bisa *zero down time* (tidak ada gangguan). Penekanan *waktu down time* seminimal mungkin atau bahkan *zero down time*, hanya dapat dilakukan apabila kita mempunyai lebih dari satu buah server . Sebuah teknologi untuk meningkatkan kinerja server dikenal dengan nama *clustering* (Sumarna, 2015).

Clustering web server merupakan pengelompokan beberapa *node* ke dalam suatu sistem *web server*. Salah satu teknik *clustering* web server adalah *auto scaling*. Dengan menggunakan *auto scaling*, dapat menambah jumlah *server* pada saat *demand* bertambah melebihi kapasitas selain itu *auto scaling* akan menyesuaikan kinerja sistem *cluster web server* menjadi lebih responsif. Metode

*auto scali*ng yang akan digunakan pada penelitian ini adalah *horizontal scaling* yaitu metode yang akan menambahkan jumlah *instance* server dengan spesifikasi yang sama ketika beban kapasitasnya cukup tinggi. Teknik ini dapat dimanfaatkan untuk menunjang kebutuhan layanan pengguna salah satunya yaitu pada layanan website *e-commerce*.

Hal ini dipilih karena biasanya ketika sebuah perusahan *e-commerce* sedang menawarkan promo yang menarik, website tersebut akan mengalami peningkatan pengunjung. Menurut Katadata.co.id, peningkatan tersebut bisa mencapai sekitar enam kali lipat dibandingkan dengan hari biasanya. Karena peningkatan pengunjung yang mengakses website secara drastis, maka diperlukan suatu sistem pelayanan yang pas dan sesuai, sehingga bisa melayani semua pengunjung tersebut dalam satu waktu sekaligus. Pemanfaat keadaan seperti ini sebagai salah satu model pengujian *load testing* yang berguna untuk mengukur seberapa tanggap website tersebut untuk melayani pengunjung yang jumlahnya banyak dalam satu waktu. Pengujian ini juga berguna untuk mengukur seberapa baik kualitas pelayanan website tersebut terhadap pengunjung yang mengakses website tersebut.

Karena itu akan dirancang *cluster web server* sehingga dapat meningkatkan *availability* sistem pada saat jumlah *request* bertambah banyak dan besar melebihi kemampuan sistem virtual sebelumnya. Dengan demikian judul penelitian yang diusulkan adalah “Analisis Kinerja Sistem *Auto Scaling* Pada Sistem Web Server Berbasis *Clustering* Menggunakan Sistem Virtual”.

1.2. Rumusan Masalah

Berdasarkan latar belakang, maka rumusan masalah dalam tugas akhir ini adalah :

1. Bagaimana membuat *clustering web server* menggunakan *auto scaling*?
2. Bagaimana kinerja sistem *clustering web server* dengan *auto scaling*?

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah membuat suatu sistem *web server* menggunakan metode *auto scaling* yang respontan dan *high availability* pada sistem virtual AWS.

1.4. Manfaat Penelitian

Manfaat yang diharapkan dalam penelitian ini adalah :

1. Bagi Masyarakat : Membantu mengetahui keterbatasan kapasitas sistem virtual serta kinerja sistem *auto scaling* pada *web server*.
2. Bagi Peneliti : diharapkan dapat menambah wawasan mengenai *clustering web server* menggunakan sistem virtual dan bagaimana *auto scaling* meningkatkan sistemnya.
3. Bagi Pendidikan : Mampu menjadi bahan referensi mengenai *clustering web server* menggunakan sistem virtual dan bagaimana *auto scaling* meningkatkan sistemnya.

1.5. Batasan Masalah

Agar penelitian tepat sasaran, batasan-batasan permasalahan terhadap topik penelitian ditentukan sebagai berikut:

1. Membuat *clustering web server* menggunakan AWS *auto scaling*.

2. Pola *auto scaling* yang digunakan adalah *dinamically.*
3. Metode *auto scaling* yang digunakan adalah *horizontal scaling.*
4. Menganalisa keterbatasan kemampuan AWS *auto scaling* pada *clustering web server*.

1.6. Sistematika Penulisan

Untuk memberikan gambaran singkat mengenai isi tulisan secara keseluruhan, maka akan diuraikan beberapa tahapan dari penulisan secara sistematis, yaitu :

BAB I PENDAHULUAN

Bab ini memberikan gambaran tentang latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan penelitian, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada bab ini memberikan sejumlah kajian pustaka yang berhubungan dengan topik dan teori-teori yang digunakan dalam penelitian.

BAB III METODE PENELITIAN

Pada bab ini memberikan gambaran tentang perancangan sistem yang akan dibuat dan perancangan pengujian sistem yang dibuat.

BAB IV HASIL DAN PEMBAHASAN

Dalam bab ini menjelaskan hasil dari penelitian dan pembahasan yang disertai tabel terkait hasil penelitian yang dilakukan.

BAB V PENUTUP

Bab ini berisi tentang kesimpulan yang didapatkan berdasarkan hasil penelitian yang telah dilakukan serta saran-saran untuk pengembangan lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1. *Clustering*

Cluster merupakan sekelompok mesin yang bertindak sebagai sebuah entitas tunggal untuk menyediakan sumber daya dan layanan ke jaringan dengan tujuan untuk menjaga ketersediaan resource bagi klient ketika terjadi kegagalan *software* maupun *hardware* pada server (Sumarna, 2015).

2.1.1. *Auto Scaling*

Auto scaling adalah suatu teknik yang digunakan untuk mengelola kapasitas komputasi secara otomatis sehingga sehingga administrator tidak perlu melakukan penambahan ataupun pengurangan *resource* secara *manual* ketika menjalankan suatu layanan *cloud.* Dengan menerapkan *auto-scaling*, dapat meningkatkan efektivitas dan efisiensi *resource cloud computing* maupun mengurangi beban administrator *cloud.* Ada dua aturan yang digunakan dalam *auto-scaling* ini yaitu *predictably* (prediksi) dan *dynamically* (Wardhana & Suwastika, 2014).

1) *Predictably* (prediksi)

   Menggunakan aturan dengan melakukan prediksi terhadap layanan yang dijalankan, dengan melakukan prediksi akan diketahui kapan kira-kira terjadi lonjakan trafik ataupun penurunan trafik yang kemudian akan dilakukan *scaling* yang disesuaikan dengan trafik yang dibutuhkan, misalnya suatu ketika trafik naik secara drastis maka CPU *utilization* dapat ditingkatkan menjadi 50% dari normal, *Memory usage* ditingkatkan, atau bahkan penambahan *resource* dari server lainnya dan seterusnya. Sistem prediksi sangat efektif jika

predictive system yang digunakan tepat, namun sebaliknya, sistem prediksi ini dapat memperburuk penggunaan *resource* jika *predictive system* yang digunakan tidak tepat.

2) *Dynamically*

Melakukan proses *scaling* secara dinamis tidak tergantung dari trafik yang berjalan, metode ini akan melakukan *scaling* untuk layanan-layanan tertentu yang telah diatur oleh administrator, misalnya ketika sistem menjalankan layanan dari Amazon EC2 maka CPU *utilization* ditingkatkan beberapa persen, memori ditingkatkan menjadi beberapa *gigabytes*, *port-port* tertentu dibuka, dan sebagainya. Metode ini akan baik jika memang layanan tersebut ketika dijalankan membutuhkan *resource* yang sesuai, namun jika layanan tersebut dijalankan sedangkan pemakaian layanan sangat sedikit maka hal tersebut dapat mengurangi efisiensi penggunaan *resource* dan menambah biaya dari *resource* yang berjalan.

2.1.2. *Load Balancing*

Load-balancing merupakan *cluster* server dimana anggota *cluster* server dikonfigurasikan untuk saling berbagi beban yang berfungsi mendistribusikan *request* dari klien ke anggota server *load balanced cluster*. Tipe konfigurasi *load balancing cluster* sering disebut *load balanced cluster*, sedangkan teknologi *platform load balancing* sering disebut sebagai *load balancers*. Secara umum cara kerja *load balancer* adalah menerima *incoming request* dari klien dan meneruskan *request* tersebut pada server tertentu jika dibutuhkan. *Load balancer* menggunakan

beberapa algoritma yang berbeda untuk melakukan control *traffic network*, yaitu (Sumarna, 2015):

a) *Round-Robin*. Algoritma *round-robin* mendistribusikan beban ke semua server anggota *cluster* sehingga masing-masing server mendapat beban yang sama dalam waktu yang sama. *Round-robin* cocok saat server anggota *cluster* memiliki kemampuan *proccessing* yang sama, jika tidak, beberapa server bisa jadi menerima *request* lebih dari kemampuan *processing* server itu sendiri sedang yang lainnya hanya mendapat beban lebih sedikit dari *resource* yang dimiliki.

b) *Least-connection*. Algoritma *least-connection* melakukan pengiriman *request* pada server anggota *cluster*, berdasarkan pada server mana yang memiliki *fewest connections* (koneksi paling sedikit).

c) *Source*. Algoritma *source* merupakan sebuah algoritma dalam *software haproxy (load balanc*e), dimana cara kerja algoritma tersebut dengan melakukan pencatatan alamat IP (IP *Address*) dari *user* yang ingin mengakses sebuah website. Pencatatan IP *address* tersebut dimaksudkan agar *user* yang sama diarahkan pada server yang sama apabila server tersebut tidak *down* (mati). Algoritma *source* merupakan algoritma statis, artinya algoritma ini tidak berpengaruh terhadap perbedaan beban server, pada algoritma ini yang didahulukan adalah *traffic* server, sehingga memungkinkan terjadinya perbedaan yang cukup signifikan pada web server.

2.2. Amazon Web Service (AWS)

Amazon Web Service (AWS) merupakan *platform cloud* paling komprehensif dan digunakan secara luas di dunia yang menawarkan lebih dari 200 layanan unggulan yang lengkap dari pusat data secara global. Dibandingkan dengan penyedia *cloud* lainnya, AWS memiliki lebih banyak fitur dalam layanannya mulai dari teknologi infrastruktur seperti perhitungan, penyimpanan, dan *database* hingga teknologi yang berkembang seperti *machine learning*, *artificial intelligence*, *data lake* dan analitik, dan *Internet of Things*. AWS juga menawarkan berbagai ragam *database* paling luas yang dibangun untuk berbagai jenis aplikasi sehingga kita dapat memilih layanan yang tepat untuk tugas tersebut untuk mendapatkan biaya dan peforma tebaik.

AWS dirancang menjadi salah satu komputasi *cloud* paling fleksibel dan aman yang tersedia saat ini. Infrastruktur AWS dibangun untuk memenuhi persyaratan kemananan militer, bank global, dan organisasi dengan tingkat sensitivitas tinggi lainnya. AWS memiliki 230 layanan dan fitur keamanan, kepatuhan, dan tata kelola. Selain itu, AWS juga mendukung 90 standar keamanan sertifikat kepatuhan. (What is AWS: Amazon Web Service, 2022)

2.2.1. Amazon Elastic Compute Cloude (Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) adalah layanan web yang memberikan kapasitas komputasi yang aman dan berukuran fleksibel di *cloud*. Amazon EC2 menawarkan banyak opsi untuk membangun dan menjalankan hampir semua aplikasi. Dengan kemungkinan ini, Amazon EC2 bisa dimulai

dengan cepat dan mudah. (Amazon EC2 Auto Scaling: Amazon Web Service, 2022).

2.2.2. Amazon Relational Database Service (RDS)

Relational Database Service (RDS) merupakan layanan pengelolaan RDBMS (*relational database*) dari AWS yang berisi kumpulan item data dengan hubungan yang telah ditentukan sebelumnya. Berbagai item ini disusun menjadi satu set tabel dengan kolom dan baris. Tabel digunakan untuk menyimpan informasi tentang objek yang akan direpresentasikan dalam *database*. Tiap kolom pada tabel berisi jenis data tertentu dan bidang yang menyimpan nilai aktual atribut. Baris pada tabel merepresentasikan kumpulan nilai terkait dari satu objek atau entitas. Tiap baris pada tabel dapat ditandai dengan pengidentifikasi unik yang disebut kunci utama (*primary key*), dan baris di antara beberapa tabel dapat dibuat saling terkait menggunakan kunci asing (*foreign key*). Data ini dapat diakses dengan berbagai cara tanpa menyusun ulang tabel *database* itu sendiri (Relational Database: Amazon Web Service, 2022).

Amazon RDS mendukung beberapa tipe *instance database* yaitu Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle, dan SQL Server. Untuk memigrasikan *database* ke Amazon RDS, kita dapat menggunakan AWS Database Migration. Amazon RDS menangani tugas *database* rutin seperti penyediaan, *patching*, pencadangan, pemulihan, pendeteksian kegagalan, dan perbaikan. Amazon RDS memiliki manfaat sebagai berikut. (id-rds:Amazon Web Service, 2022)

1. Mudah dikelola, Amazon RDS memudahkan proses dari pembuatan konsep proyek hingga *launching*. Kita tidak perlu penyediaan infrastruktur, dan tidak perlu memasang serta mengelola *software database*. Kita dapat menggunakan Konsol Manajemen Amazon RDS, AWS RDS Command-Line Interface, atau panggilan API sederhana untuk mengakses kemampuan relasional *database* yang siap produksi dalam hitungan menit.
2. Mudah diskalakan, kita dapat memperbesar sumber daya komputasi dan penyimpanan *database* hanya dengan melakukan pemanggilan API.
3. Tersedia dan tahan lama, Amazon RDS berjalan dengan infrastruktur tepercaya yang juga digunakan oleh Amazon Web Services lainnya. Ketika kita menyediakan *Instance* DB Multi-AZ, Amazon RDS secara serentak mereplikasi data ke *instance standby* dalam *Availability Zone* (AZ) berbeda. Amazon RDS memiliki banyak fitur lain yang meningkatkan keandalan untuk *database* produksi penting, termasuk pencadangan otomatis, *snapshot database*, dan penggantian *host* otomatis.
4. Cepat, Amazon RDS mendukung aplikasi *database* yang paling mendesak. Amazon RDS memiliki dua opsi penyimpanan yang didukung SSD yaitu, dioptimakan untuk aplikasi OLTP performa tinggi, dan yang lainnya untuk penggunaan umum hemat biaya. Selain itu, Amazon Aurora memberikan performa yang setara dengan *database* komersial dengan perbandingan biaya 1/10.
5. Aman, Amazon RDS mempermudah mengendalikan akses jaringan ke *database*. Amazon RDS juga memungkinkan untuk menjalankan *instance*

database dalam Amazon *Virtual Private Cloud* (Amazon VPC), sehingga kita dapat memisahkan *instance database* dan menghubungkan infrastruktur IT yang sudah ada melalui IPsec VPN dienkripsi standar industri. Banyak jenis mesin Amazon RDS menawarkan *encryption at rest* dan *encryption in transit*.

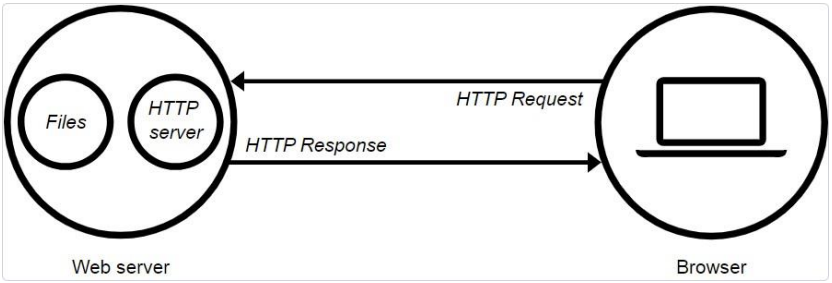
2.3. Web Server

Web server merupakan suatu *software* yang berada di dalam server yang berfungsi untuk menerima halaman web sebagai permintaan melalui protokol HTTPS/HTTP atau yang biasa dikenal dengan "browser". Setelah itu harus mengirimkan kembali hasil dari permintaan atau respon tersebut menjadi dokumen HTML (Web Server: idCloudHost, 2020).

Web server dapat juga berfungsi sebagai *hardware* dan *software*, yaitu apabilasebagai *hardware* maka web server berfungsi untuk menyimpan *web server software* dan komponen file situs web seperti dokumen HTML, gambar, CSS, dan fileJavaScript. Web server terhubung ke internet dan mendukung pertukaran data fisikdengan perangkat lain yang terhubung ke web. Apabila sebagai *software* maka webserver mencakup beberapa bagian yang mengontrol bagaimana pengguna web dapatmengakses file yang dihosting atau dengan kata lain web server berfungsi untuk mengatur dan memproses segala aktivitas permintaan yang diterima dan dibaca pada halaman browser. (What is web server: Developer Mozilla, 2022)

Secara sederhana, tugas web server adalah untuk menerima permintaan dari klien kemudian mengirimkan kembali data yang diminta oleh klien tersebut. Sama

halnya dengan komputer klien, komputer server juga harus terhubung dengan jaringan internet untuk dapat diakses oleh klien. Ilustrasi cara kerja layanan web server dapat dilihat pada gambar 2.1.



Gambar 2.1. Ilustrasi cara kerja layanan web server

Berdasarkan gambar 2.1. pada saat browser (klien) melakukan permintaan data ke web server, maka instruksi permintaan data tersebut akan dikemas di dalam TCP yang merupakan protokol *transport* dan kemudian akan dikirimkan ke alamat yang dibutuhkan yaitu HTTP atau HTTPS. Data yang diminta oleh browser ke web server disebut dengan HTTP *request* yang kemudian akan dicarikan oleh web server di dalam komputer server. Jika ditemukan, data tersebut akan dikemas oleh web server dalam TCP dan dikirim kembali ke browser untuk ditampilkan. Data yang dikirim dari server ke browser dikenal dengan HTTP *response*. Jika data yang diminta oleh browser tersebut ternyata tidak ditemukan oleh web server, maka web server akan menolak permintaan tersebut dan browser akan menampilkan notifikasi *Page Not Found* atau *Error* 404.

2.3.1. Apache Web Server

Apache merupakan salah satu web server yang bertanggung jawab pada *request response* HTTP dan *logging* informasi secara detail. Secara umum

arsitektur *apache* bekerja dengan model *thread-based* atau berbasis proses. Web server berbasis proses menggunakan proses (*thread*) untuk menerima dan merespon permintaan. Setiap permintaan akan diciptakan sebuah *thread* yang disimpan dalam sebuah pool pada alokasi memori tertentu dan dilakukan proses untuk menjawab, setelah proses dieksekusi kemudian hasil proses dikirimkan kembali ke klien serta dicatat dalam sebuah *log*

Beberapa dukungan yang terdapat pada *apache* web server yang menjadi alasan *client* untuk menggunakannya dibanding web server yang lain diantaranya (Arunawati, 2020).

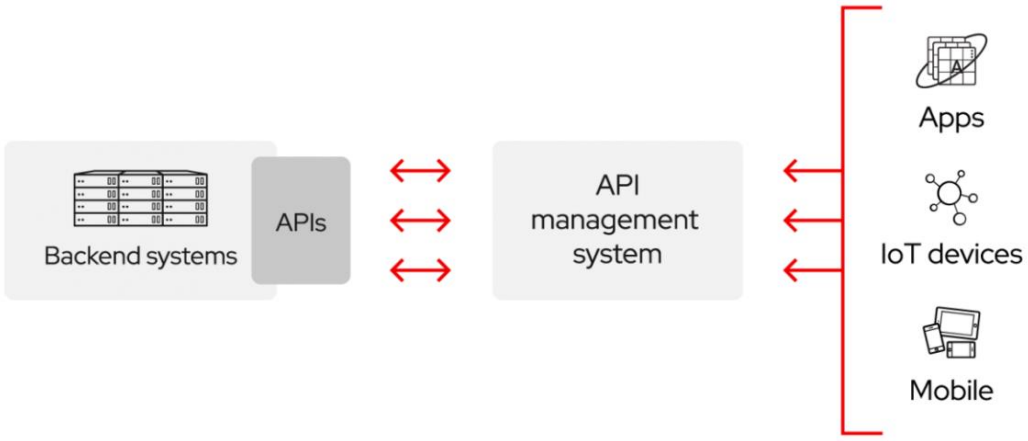
1. Kontrol akses, yang dapat dijalankan berdasarkan nama *host*.
2. CGI (*Common Gateway Interface*). Yang paling terkenal untuk digunakan adalah PERL (*Practical Extraction and Report Language*), didukung oleh *apache* dengan menempatkannya sebagai modul (mod_perl).
3. PHP (*Personal Home Page/PHP Hypertext Processor*) Program dengan metode semacam CGI, yang memproses teks dan bekerja di server. *Apache* mendukung PHP dengan menempatkannya sebagai salah satu modulnya (mod_php).
4. SSI (*Server Side Includes*)

Apache merupakan salah satu server web yang paling banyak digunakan di dunia, beberapa keunggulan Apache dibandingkan dengan web server yang lain seperti IIS (*Internet Information Service*) dari Microsoft adalah kemampuannya untuk mendukung berbagai bahasa script paling populer seperti PHP (*Personal Home Page*) dan JSP (*Java Server Pages*). Hal lain yang membuat *Apache* lebih

diminati adalah sistem lisensinya yang gratis sehingga mengurangi biaya yang perlu dikeluarkan dalam membangun situs web dinamis (Emanuel, 2006).

2.4. REST API

API (*Application Program Interface*) merupakan antarmuka yang menjadi perantara atau jembatan antara sistem aplikasi yang berbeda tanpa harus mengetahui cara penerapannya. Hal ini dapat menyederhanakan pengembangan aplikasi, menghemat waktu dan biaya. Saat mendesain aplikasi atau mengelola aplikasi yang sudah ada, API memberikkan fleksibilitas berupa menyederhanakan desain, administrasi, dan penggunaan; dan memberikan peluang untuk inovasi. API adalah cara yang disederhanakan untuk menghubungkan infrastruktur Anda sendiri melalui pengembangan aplikasi *cloud-native*, tetapi API juga memungkinkan untuk berbagi data dengan pelanggan dan pengguna eksternal lainnya. Ilustrasi cara kerja API dapat dilihat pada gambar 2.2 berikut (What is an API: Red Hat, 2017).



Gambar 2.2. Cara Kerja API

Berdasarkan gambar 2.2., cara kerja API yaitu ketika pengguna mengakses sebuah aplikasi misalnya sebuah aplikasi penerbangan X maka aplikasi tersebut akan mengakses API maskapai penerbangan yang sudah dihubungkan. Setelah aplikasi berhasil mengakses alamat API, permintaan tersebut akan diteruskan ke server maskapai penerbangan. API akan memberitahukan bahwa aplikasi X membutuhkan data penerbangan untuk tanggal dan tujuan yang telah disebutkan. Ketika menemukan data yang sesuai permintaan, server akan memberikan respon ke API untuk memberikan data yang dibutuhkan. Selanjutnya, API meneruskan informasi dari server ke aplikasi X. Proses ini berlangsung bersama dengan permintaan ke maskapai penerbangan lain. Oleh karena itu, dalam satu pencarian bisa menampilkan jadwal penerbangan dari berbagai maskapai sekaligus. (What is an API: Red Hat, 2017)

Dalam penggunaannya, terdapat empat jenis API berdasarkan dengan hak aksesnya, yaitu

1. *Public API* atau juga disebut open API merupakan sebuah *programming interface* yang dapat diakses secara publik. Developer dapat mengakses sistem pemrograman dibalik sebuah aplikasi atau layanan web melalui *public* API untuk mengembangkan aplikasi mereka sendiri dengan lebih cepat.
2. *Private API* merupakan sebuah *interface* pemrograman yang tidak terbuka untuk umum. Pada umumnya, jenis ini diciptakan untuk memenuhi kebutuhan pengembangan aplikasi secara internal. Jenis ini berperan sebagai *interface* bagian *front end* yang digunakan untuk mengakses data dan fungsi aplikasi di *back end*.

3. Partner API merupakan jenis *interface* yang dapat diakses oleh pihak-pihak tertentu yang telah ditunjuk sebagai rekanan bisnis dari pihak pemilik aplikasi atau *web service*. Jenis ini tidak tersedia untuk umum dan memerlukan kredensial tertentu untuk mengaksesnya.
4. Composite API merupakan jenis *interface* yang terdiri dari gabungan berbagai jenis data dari berbagai server dan *hosting* dalam satu tempat. Tipe API ini sangat berguna bagi developer karena mereka dapat mengakses banyak informasi dalam satu tempat saja.

API tidak hanya dalam bentuk Web *Service*, bisa saja berupa SDK (*Software Development Kit*) ataupun lainnya.

2.5. *Framework* Laravel

Laravel adalah sebuah *framework* web berbasis PHP yang *open-source* dan tidak berbayar yang diciptakan oleh Taylor Otwell yang dapat digunakan untuk pengembangan aplikasi web dengan menggunakan pola arsitektur MVC (*Model-View-Controller*). Struktrur pola MVC pada laravel sedikit berbeda pada struktur pola MVC pada umumnya. Di laravel terdapat *routing* yang menjembatani antara *request* dari *user* dan *controller*. Jadi *controller* tidak langsung menerima *request* tersebut (Sari & Wijanarko, 2019).

Laravel dibangun menggunakan bahasa pemrograman PHP yang cukup populer di Indonesia. Selain itu jika dibandingkan dengan *framework* PHP lainnya, laravel memiliki banyak keunggulan seperti *authentication* yang memudahkan pembuatan fitur *login* dan register secara otomatis tanpa harus menulis kode program, *Eloquent Orm* untuk mengelola *database* tanpa

menggunakan bahasa SQL, dan cocok untuk membuat layanan *restfull* API (L. Adi dkk, 2018). Selain itu, keunggulan *framework* laravel lainnya adalah *routing*, *session manager*, *caching*, dan beberapa fungsi lain dalam laravel. Laravel juga menyediakan fitur seperti *database migration* dan integrasi *unit testing support* yang memudahkan *developer* untuk membangun aplikasi yang kompleks (Somya & Nathanael, 2019).

2.6. *Progressive Web Application (PWA)*

Progressive Web Apps (PWA) adalah sebuah istilah untuk aplikasi berbasis web yang menggunakan teknologi web paling mutakhir. PWA sebenarnya hanyalah aplikasi berbasis web biasa, tapi memanfaatkan fitur perambanan yang modern agar tampil seolah-olah merupakan aplikasi asli. PWA digambarkan sebagai kumpulan dari teknologi, konsep desain dan WEB API (*Application Programming Interface*) yang bekerja secara bersama untuk memberikan sentuhan aplikasi pada sebuah mobile web (Kurniawan, Areni, & Andani, 2017).

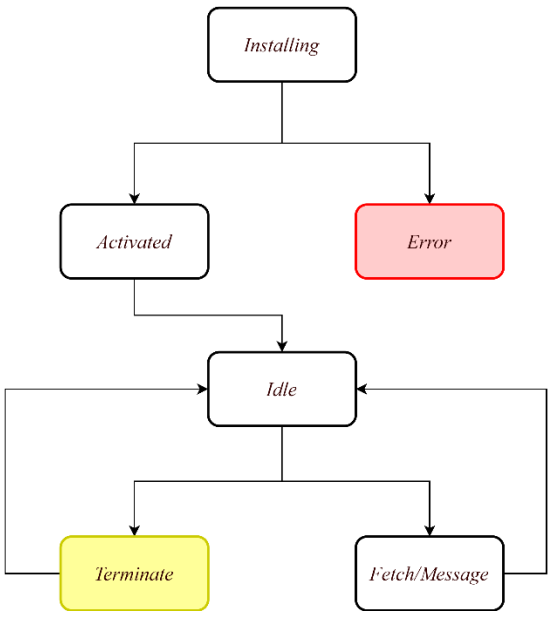
Dengan membangun aplikasi menggunakan konsep PWA akan membuat aplikasi web berjalan diberbagai *platform* seperti website, desktop, dan *platform* mobile atau Android, PWA sendiri memiliki berbagai keunggulan seperti (Aripin & Somantri, 2021) :

1. Memiliki sifat aplikasi *native* layaknya Android, tetapi tidak memerlukan spesifikasi *hardware* yang tinggi dalam membuatnya.
2. Dapat mempercepat proses pembuatan website dan mengurangi beban server.
3. Dapat berfungsi secara *offline*, tidak seperti website yang tidak dapat diakses jika tidak ada koneksi internet.

4. Banyak digunakan di perusahaan besar seperti Twitter, Facebook, Bukalapak, Tokopedia, dan sebagainya.

Hasil yang didapat dari penerapan metode PWA adalah kemampuan sebuah aplikasi untuk memiliki berbagai fitur dari aplikasi web pada umumnya, dengan optimasi mobile experience yang ada pada aplikasi native. PWA juga dapat mengirim *push notification* yang relevan kepada pengguna, dapat memiliki *icon* aplikasi di layar beranda ponsel, dan hanya membutuhkan sedikit ruang memori (Kurniawan A. A., 2020).

Keuntungan dari PWA ini adalah dapat memuat konten seketika bahkan dalam kondisi jaringan yang tidak baik. Ketika digunakan dari layar awal pengguna, maka *service worker* diaktifkan pada PWA untuk dimuat seketika (Kurniawan, Areni, & Andani, 2017).



Gambar 2.3. Siklus *service worker*

2.7. *New Relic*

Salah satu hal penting dalam merawat server adalah memantau kesehatannya dan melakukan diagnosa jika terjadi masalah pada server. *New Relic* merupakan sebuah aplikasi berbasis *cloud* yang berberfungsi untuk memantau aktivitas server beserta proses-prosesnya. Dengan menggunakan *new relic*, kita dapat memahami dan lacak dependensi di seluruh sistem terdistribusi sehingga kita dapat mendeteksi anomali, mengurangi latensi, menghentikan kesalahan, dan mengoptimalkan *user experience* (Application Monitoring: Ner Relic, 2008).

Selain itu aplikasi seperti PHP, Ruby, Phyton dan lainnya bisa melaporkan detail kerjanya ke *new relic*. Dengan begitu kita dapat mengetahui apa penyebab jika suatu sistem mengalami VPS penting *down* dalam waktu yang lama, *lifecycle*-nya tidak stabil dan *system load*-nya terlalu lama (Chandra, 2015).

2.8. *Uptime Robot*

Uptime Robot merupakan sebuah aplikasi web layanan *monitoring* yang berfungsi untuk memantau server atau website apakah dapat diakses dengan baik atau tidak. Tidak bisa teraksesnya website biasanya disebabkan oleh *hacker*, CPanel stuck, gangguan *hardware*, *maintenance* rutin, *over* CPU, dan sebagainya. Dengan menggunakan aplikasi *Uptime Robot* kita dapat mengetahui penyebab dari website tersebut sehingga tidak bisa diakses. Setiap kali aplikasi mengalami *downtime*, *Uptime Robot* akan mengirimkan pemberitahuan melalui email yang terdaftar sehingga kita dapat mengetahuinya sedini mungkin (Gutama, 2021). Sebaliknya

jika aplikasi mengalami *up time*, *Uptime robot* juga akan memberitahukannya. Oleh karena itu, aplikasi web tidak perlu dipantau secara terus menerus.

Uptime robot memiliki fitur sebagai berikut (Features: Uptime Robot, 2022):

1. *Website monitoring* : Jadilah yang pertama mengetahui bahwa situs web Anda sedang down! Pemantauan yang andal memperingatkan Anda sebelum ada masalah signifikan dan menghemat uang Anda.
2. *SSL monitoring*
3. *Ping monitoring* : memeriksa ketersediaan jaringan.
4. *Port monitoring* : memantau layanan spesifik apa pun yang berjalan di *port* mana pun seperti memeriksa apakah layanan email masih *up* dan memeriksa server *database* penting.
5. *Cron job monitoring* : juga dikenal sebagai pemantauan detak jantung. memantau *recurring background jobs* atau perangkat intranet yang terhubung ke internet.
6. *Keyword monitoring* : memeriksa ada atau tidaknya teks tertentu pada *request response* (biasanya HTML atau JSON).

2.9. *Artillery*

Artillery merupakan perangkat pengujian *performance* yang modern, kuat dan mudah digunakan. *Artillery* digunakan untuk mengirimkan aplikasi skalabel yang tetap *performance* dan tangguh di bawah beban yang tinggi. Pengujian *performance* yang dimaksud adalah (Documentation: Artillery, 2022):

1. Pengujian yang memberikan beban pada suatu sistem, yaitu pengujian beban, tegangan, dan pengujian rendam.

2. Pengujian yang memverifikasi bahwa suatu sistem bekerja seperti yang diharapkan, yaitu pengujian fungsional berkelanjutan, juga dikenal dengan sejumlah nama lain seperti : pemantauan sintetis, pemantauan semantik, pengujian skrip produksi, dan verifikasi berkelanjutan.

Artillery dirancang untuk menguji sistem *backend*, seperti layanan API, *backend* e-niaga, sistem obrolan, *backend* game, *database*, perantara pesan dan antrean, serta apa pun yang dapat dikomunikasikan melalui jaringan. Juga dapat digunakan untuk menguji *backend* apa pun terlepas dari protokol apa yang digunakan atau bahasa apa yang digunakannya. *Artillery* mendukung HTTP, *WebSocket*, dan Socket.io, serta banyak protokol tambahan seperti HLS, Kinesis, dan Kafka melalui *plugins*. Untuk dukungan protokol tambahan, dapat ditambahkan melalui antarmuka *plugin Artillery*. Berbagai macam skenario yang biasanya digunakan dalam *Artillery* meliputi (Documentation: Artillery, 2022) :

1. Menjalankan pengujian beban ad-hoc terhadap API atau layanan mikro individual sebagai bagian dari proses pengembangan untuk mengeksplorasi karakteristik kinerjanya, dan membuat pengoptimalan kinerja jika diperlukan (misalnya, mencegah kebocoran memori atau mengoptimalkan CPU-*hevy code*).
2. Menjalankan pengujian terhadap lingkungan *staging*/fitur sebagai bagian dari *pipeline* CI/CD untuk menangkap regresi kinerja lebih awal, dan untuk memverifikasi SLO.

3. Menguji beban skala besar sebelum merilis layanan baru, atau untuk menyiapkan aplikasi dan infrastruktur untuk *traffic* yang padat, seperti *Black Friday/Cyber Monday traffic*.
4. Menambahkan *traffic* sintetis dalam produksi untuk mempertahankan margin keamanan terhadap lonjakan *traffic*.
5. Menjalankan pemantauan sintetis terhadap API utama dari beberapa lokasi geografis untuk memverifikasi bahwa transaksi dan alur utama berfungsi seperti yang diharapkan, dan memperingatkan jika ada yang rusak.

Artileri biasanya digunakan di seluruh tim yang bertanggung jawab untuk pengiriman, pengujian, dan pengoperasian sistem *backend* produksi: dari pengembang aplikasi, hingga insinyur pengujian & QA, dan ops/SRE.