Volume 5, Nomor 1, Juni 2024, hlm 88-98
INFOTECH: Jurnal Informatika & Teknologi
p ISSN 2722-9378 | e ISSN 2722-9386
http://jurnal.sttmcileungsi.ac.id/index.php/infotech

Optimalisasi monitoring tag dengan AWS *Cloud*: Studi kasus aplikasi tag tracker pada PT. XYZ

Optimizing tag monitoring with AWS cloud: Case study of the tag tracker application at PT. XYZ

Norma Ningsih[1]*, Karimatun Nisa[2], Endryco Farel Rianrachmatullah[1], Bintang Desta Ramadhani[1], Afifah Dwi Ramadhani[1]

[1*] Prodi Teknologi Rekayasa Internet, Politeknik Elektronika Negeri Surabaya, Jl. Raya ITS, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur, Indonesia 60111
[2*] Prodi Teknik Telekomunikasi, Politeknik Elektronika Negeri Surabaya, Jl. Raya ITS, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur, Indonesia 60111

INFORMASI ARTIKEL

Article History:
Submission: 18-05-2024
Revised: 27-05-2024
Accepted: 04-06-2024

Kata Kunci:
Tag tracker; cloud computing; AWS; aplikasi

Keywords:
Tag tracker; cloud computing; AWS; application

*** Korespondensi:**
Norma Ningsih
norma@norma.ac.id

ABSTRAK

Cloud computing telah menjadi fondasi bagi banyak inovasi teknologi modern, dengan AWS (*Amazon Web Services*) sebagai salah satu platform *cloud* yang terkenal dan paling handal. Namun, banyak perusahaan menghadapi tantangan dalam memantau dan menganalisis jumlah *views* (tampilan) dari berbagai "tag" yang digunakan dalam kampanye bisnis mereka. Penelitian ini bertujuan untuk mengatasi permasalahan tersebut dengan mengembangkan aplikasi *Tag Tracker* menggunakan *platform* AWS. Metode yang digunakan dalam penelitian ini melibatkan pengembangan sistem berbasis *serverless* menggunakan AWS Lambda, serta pemanfaatan layanan AWS lainnya seperti *Simple Queue Service (SQS), EventBridge* dan *database* DynamoDB. Sistem *Tag Tracker* dirancang untuk memantau dan menganalisis jumlah *views* dari berbagai tag, memberikan informasi yang dapat digunakan oleh tim internal perusahaan untuk membuat strategi bisnis yang lebih efektif. Metode pengembangan perangkat lunak yang digunakan adalah metode *waterfall*. Hasil dari penelitian ini menghasilkan dua layanan utama, yaitu *dashboard* dan *aggregator*. Berdasarkan hasil pengujian, *service dashboard* membutuhkan waktu komputasi untuk pemrosesan data dari saat data diterima hingga disimpan di DynamoDB dan ditampilkan pada grafik *dashboard* berkisar antara 1-2 menit. Pengujian fungsionalitas aplikasi dilakukan dengan metode *blackbox* dan diperoleh tingkat keberhasilan fungsi sebesar 100%. Optimalisasi performa menunjukkan bahwa sistem mampu menangani beban kerja dengan efisien, memberikan waktu respon yang cepat dan andal. Implementasi sistem *Tag Tracker* menunjukkan performa yang sangat baik dan dapat digunakan oleh internal perusahaan untuk mendapatkan informasi yang akurat terkait popularitas setiap tag di pelanggan, sehingga dapat menjadi pertimbangan dalam menentukan model bisnis perusahaan.

ABSTRACT

Cloud computing has become the foundation for many modern technological innovations, with AWS (Amazon Web Services) being one of the best-known and most reliable cloud platforms. However, many companies face challenges in monitoring and analyzing the number of views of the various "tags" used in their business campaigns. This research aims to overcome this problem by developing the Tag Tracker application

using the AWS platform. The method used in this research involves developing a serverless-based system using AWS Lambda, as well as utilizing other AWS services such as Simple Queue Service (SQS), EventBridge, and the DynamoDB database. The Tag Tracker system is designed to monitor and analyze the number of views from various tags, providing information that internal company teams can use to create more effective business strategies. The results of this research produce two main services, namely dashboard and aggregator. Based on the test results, the service dashboard requires computing time for data processing from the time the data is received until it is stored in DynamoDB and displayed on the dashboard graph, ranging from 1-2 minutes. Application functionality testing was carried out using the black box method and a functional success rate of 100% was obtained. Performance optimization shows that the system is able to handle workloads efficiently, providing fast and reliable response times. The implementation of the Tag Tracker system shows very good performance and can be used by internal companies to obtain accurate information regarding the popularity of each tag among customers, so that it can be taken into consideration in determining the company's business model

1. PENDAHULUAN

Salah satu perkembangan teknologi komputer terbaru saat ini merupakan pemanfaatan teknologi *cloud computing* [1][2]. Teknologi ini merupakan gabungan dari pemanfaatan teknologi komputer dan internet yang memungkinkan pengembangan dan penyebaran aplikasi menjadi lebih efisien dan melakukan *auto scaling* [3][4]. Dengan adanya teknologi ini, para pengembang aplikasi web maupun *mobile* tidak perlu khawatir tentang pengelolaan server fisik [5]. *Cloud computing* memungkinkan penggunaan server virtual dengan kapasitas yang fleksibel dan performa tinggi [6][7]. *Cloud computing* telah menjadi fondasi bagi banyak inovasi teknologi modern, memungkinkan perusahaan dari berbagai skala untuk berinovasi lebih cepat dan lebih efisien [8][9].

AWS (*Amazon Web Services*) merupakan salah satu *platform cloud* yang terkenal dan penyedia layanan *cloud* terbesar secara global [10]. AWS terdiri dari berbagai jenis penyimpanan *database*, komputasi, *content delivery* dan layanan lainnya [11]. *Deploy* sistem pada AWS dapat menghemat biaya, waktu dan tenaga jika dibandingkan dengan cara konvensional. Berbagai manfaat dari penggunaan AWS diantaranya adalah *cost effectiveness* yaitu membayar sesuai dengan apa yang digunakan dalam platform AWS. Fleksibilitas yaitu pengguna AWS dapat membuat berbagai aplikasi menggunakan berbagai pemrograman dan *platform* yang diinginkan atau sesuai kebutuhan. Serta tangguh dan aman, dimana AWS mampu menghadapi gangguan operasional dan melindungi data serta aplikasi dari berbagai ancaman keamanan. AWS adalah kombinasi dari tiga layanan yaitu *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), dan S*oftware as a Service* (SaaS) [12].

Arsitektur tanpa server atau *serverless architecture*, adalah pendekatan untuk membangun dan menjalankan aplikasi serta layanan tanpa harus mengelola infrastruktur server secara langsung. Arsitektur tanpa server atau *Serverless Architecture*, adalah pendekatan untuk membangun dan menjalankan aplikasi serta layanan tanpa harus mengelola infrastruktur server secara langsung. Salah satu contoh *serverless* adalah AWS lambda yang merupakan *platform* komputasi cloud yang didorong oleh pemicu (*trigger-driven*) yang memungkinkan para programmer untuk menulis fungsi dan membayar hanya sesuai dengan penggunaan tanpa harus mendefinisikan sumber daya penyimpanan atau komputasi. Salah satu keunggulan utama dari AWS Lambda adalah bahwa platform ini menggunakan abstraksi dan memisahkan pengelolaan server dari pengguna akhir. Dengan AWS Lambda, Amazon sendiri yang menangani pengelolaan server, sehingga memungkinkan programmer untuk lebih fokus pada penulisan kode untuk

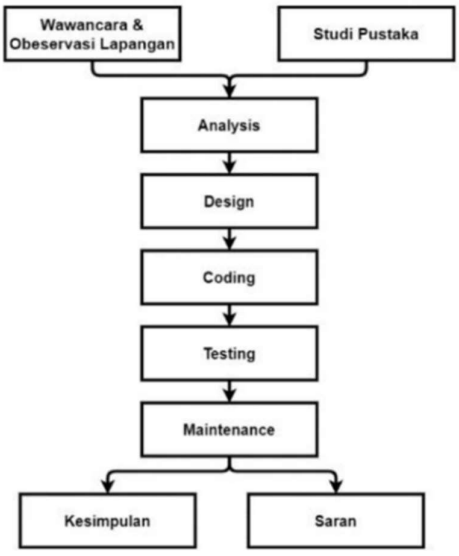
mendevelop sebuah aplikasi. *Platform serverless* paling tidak terdiri dari lima elemen yaitu *functions, API gateways, cloud services, security tools* dan *control plane.*

Pada membuat desain platform komputasi tanpa server (*serverless*) baru yang diimplementasikan menggunakan NET, diterapkan di *Microsoft Azure*, dan menggunakan kontainer Windows sebagai lingkungan eksekusi fungsi. Pengujian pada *prototype* kami serta AWS Lambda, *Azure Functions*, *Google Cloud Functions*, dan implementasi *Apache Openwhisk* oleh IBM [13]. Pengukuran kami menunjukkan bahwa *prototype* kami mencapai *throughput* yang lebih tinggi daripada platform lainnya. Pada penelitian ini menghasilkan perancangan *E-Menu* dengan memanfaatkan teknologi *cloud computing* berbasis *architecture serverless* lambda untuk memberikan manfaat yang berguna bagi pihak restoran maupun pelanggan dalam bidang pemesanan dan transaksi. Fitur pada penelitian ini terdiri dari metode pelayanan yang meliputi proses pemesanan menu oleh pelanggan, pengelolaan kategori oleh admin, pembayaran, dan semua sistem yang terintegrasi.

Dalam penelitian ini, dibangun sebuah aplikasi *Tag Tracker* menggunakan platform AWS seperti SQS, *Eventbridge*, Lambda, dan dua dynamoDB. Sistem *Tag Tracker* merupakan sebuah aplikasi yang terdiri dari dua buah *services* yaitu *dashboard* dan *aggregator.* Pertama adalah layanan *Dashboard* merupakan sebuah *service* yang berbentuk aplikasi web menampilkan statistik dari "tag" pada perusahaan. Kedua adalah layanan *aggregator* adalah Sebuah Lambda *function* yang bertugas untuk menghitung total views sebuah tag dalam satu pekan. Sistem *Tag Tracker* digunakan untuk memantau dan menganalisis jumlah *views* (tampilan) dari berbagai "tag" yang digunakan dalam perusahaan. Tujuan utama dari sistem ini adalah untuk memberikan wawasan yang jelas dan terperinci bagi tim bisnis internal perusahaan mengenai popularitas dan penggunaan tag berdasarkan jumlah views yang diterima setiap minggu.

2. METODE

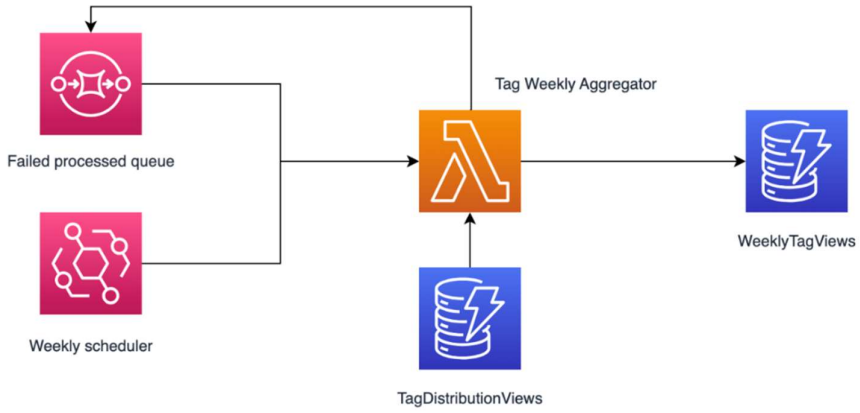
Metode yang digunakan dalam pengembangan perangkat lunak aplikasi *Tag Tracker* yaitu metode *waterfall.* Model *waterfall* adalah model klasik yang bersifat sistematis dan berurutan dalam membangun *software*.



Gambar 1. Tahapan metode *waterfall*.

Gambar 1 menunjukkan tahapan penelitian dengan menggunakan metode *waterfall* pada pengembangan aplikasi *Tag Tracker.* Tahapan ini diawali dengan melakukan wawancara dan observasi terhadap karyawan PT. XYZ dalam melakukan proses analisa terhadap katalog yang

paling populer. Selanjutnya melakukan tahapan analisa dan perancangan, implementasi hingga *maintenance* terhadap perangkat lunak yang akan dikembangkan.



Gambar 2. Arsitektur AWS aplikasi *Tag Tracker*.

Gambar 2 menunjukkan arsitektur penggunaan layanan AWS untuk membangun aplikasi *Tag Tracker* untuk proses agregasi dan monitoring data tag secara berkala. Layanan AWS yang digunakan dalam arsitektur ini adalah SQS, *Eventbridge*, Lambda, dan dua dynamoDB. Amazon SQS (*Simple Queue Service*) merupakan layanan antrian pesan untuk mengirim, menyimpan dan menerima pesan. Amazon *Eventbrigde* merupakan layanan *serveless* yang menggunakan *event* untuk menghubungkan komponen aplikasi secara bersama-sama. *EventBridge* memungkinkan untuk membangun aplikasi *event-driven* dengan menangkap perubahan status atau kejadian yang terjadi di sistem. AWS Lambda adalah layanan komputasi *serverless* yang memungkinkan pengguna untuk menjalankan kode tanpa perlu menyediakan atau mengelola *server*. Amazon DynamoDB adalah layanan basis data NoSQL yang cepat dan fleksibel yang dirancang untuk aplikasi yang memerlukan latensi baca dan tulis milidetik dengan skala yang hampir tidak terbatas. Layanan AWS yang digunakan adalah *serverless* yaitu AWS lambda, AWS SQS, AWS *Eventbridge*, dan AWS DynamoDB. Seluruh *service* yang dibutuhkan untuk aplikasi *Tag Tracker* di-*deploy* secara otomatis menggunakan AWS *Cloud Formation*.

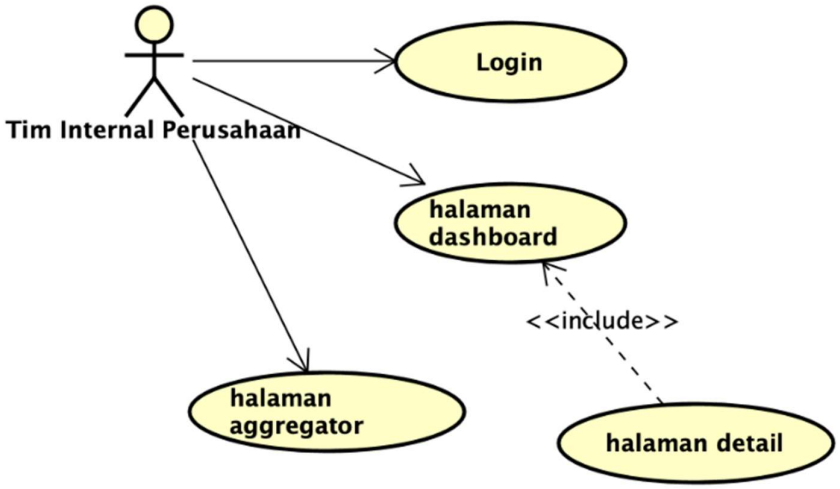
Sistem *Tag Tracker* merupakan sebuah aplikasi yang terdiri dari dua buah *services* yaitu *dashboard* dan *aggregator* [14]. *Dashboard* adalah sebuah s*ervice* yang berbentuk aplikasi web menampilkan statistik dari "*tag*" pada perusahaan. Aplikasi ini digunakan oleh tim bisnis pada internal perusahaan untuk mengetahui jumlah views pada setiap tag. Untuk saat ini, aplikasi hanya mendukung menampilkan statistik mingguan. Aplikasi web ini berisi dua halaman yaitu halaman beranda dan halaman detail. Halaman beranda akan menampilkan daftar tag yang paling banyak memiliki views selama sepekan terakhir. Halaman detail akan menampilkan total views pekanan yang dimiliki oleh sebuah tag tertentu. Terdapat juga diagram untuk memberikan visualisasi yang lebih baik. Service ini dijalankan menggunakan AWS Lambda. Teknologi yang digunakan untuk membangun service ini adalah HTML, CSS dan JavaScript untuk *frontend* dan Golang untuk *backend* sekaligus sebagai web server.

Aggregator adalah Sebuah Lambda *function* yang bertugas untuk menghitung total *views* sebuah *tag* dalam satu pekan. AWS lambda merupakan layanan FaaS (*Function as a Service*) yang memungkinkan pengguna menjalankan kode secara otomatis dilingkungan AWS [16]. Sumber data berasal dari sebuah tabel DynamoDB. Service ini akan jalan pada waktu tertentu selama satu kali dalam sepekan yang dijadwalkan dan dipanggil menggunakan AWS *EventBridge*. Karena pemanggilan dari *EventBridge* bersifat asinkron, Lambda *function* pada *service* ini dikonfigurasi akan menangani percobaan ulang maksimal selama dua kali jika prosesnya gagal. Ketika semua prosesnya gagal atau pemanggilan dari *EventBridge* masih berada dalam antrian dalam waktu lama, maka upaya proses tersebut akan dibuang kedalam *Dead Letter Queue* pada AWS SQS untuk proses debug.

Ketika perbaikan telah rilis, kita dapat memindahkan kembali pesan pemanggilan yang gagal dari DLQ untuk diproses kembali. Adapun format pesan pada setiap pemanggilan dari AWS EventBridge sebagai berikut:

```
{
"version": "0",
"account": "1234567",
"region": "eu-west-1",
"detail": {},
"detail-type": "Scheduled Event",
"source": "aws.events",
"time": "2022-05-16T16:29:37Z",
"id": "b21c416f-e979-647d-3988-a2d03e34fb72",
"resources": [
"arn:aws:events:eu-west-1:1234567:rule/lambda-event"
  ]
}
```

Pesan di atas merupakan format *payload* yang dikirimkan oleh AWS *EventBridge* saat memicu sebuah fungsi AWS Lambda berdasarkan aturan yang sudah dijadwalkan. Pesan tersebut berbasis *payload* JSON yang mendeskripsikan *event* (acara) yang terjadi yang dihasilkan ketika AWS *EventBridge* memicu sebuah *event* yang sudah dijadwalkan sebelumnya. Pada sistem *Tag Tracker,* pesan ini digunakan untuk memicu fungsi AWS Lambda yang melakukan agregasi data view secara mingguan dari tabel DynamoDB, dan menyimpan hasil agregasi ke tabel lain dalam aplikasi *Tag Tracker*. Dengan menggunakan event terjadwal dari AWS *EventBridge*, diharapkan dapat mengotomatiskan tugas-tugas rutin seperti agregasi data tanpa memerlukan intervensi manual, memastikan bahwa data selalu diperbarui tepat waktu.

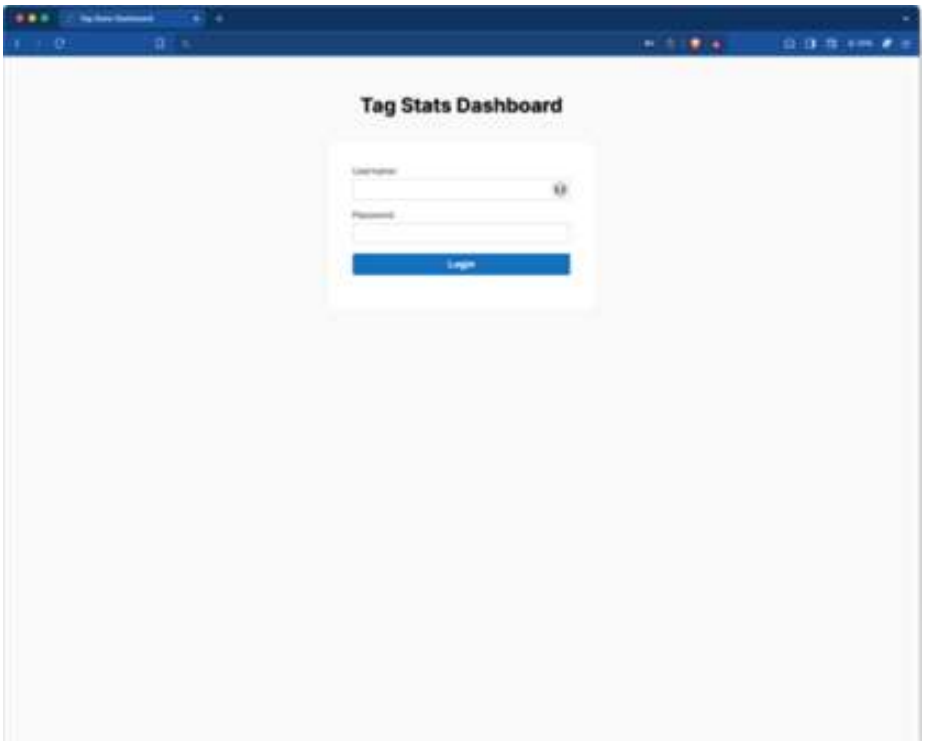


Gambar 3. *Use case diagram* aplikasi *Tag Tracker*.

Pada Gambar 3 menunjukkan *use case diagram* pada aplikasi *Tag Tracker*. Diagram diatas terdiri dari satu aktor yaitu tim internal perusahaan yang dapat mengakses fitur dari aplikasi *TagTrackker*. Diagram ini terdiri dari 3 *use case* yaitu *login* bagi pengguna yang akan mengakses sistem, kemudian ada *use case* halaman *dasboard* yang menampilkan grafik *view* dari masing-masing *tag*. Serta *use case* halaman aggregator untuk menampilkan data menghitung total *views* sebuah *tag* dalam satu pekan.

3. HASIL DAN PEMBAHASAN

Pada bagian ini akan disajikan dan dianalisis hasil pengembangan, *deployment* dan pengujian aplikasi *Tag tracker.* Teknologi yang digunakan untuk membangun service ini adalah HTML, CSS dan JavaScript untuk *frontend* dan Golang untuk *backend* sekaligus sebagai web server. Layanan AWS yang digunakan adalah *serverless* yaitu AWS lambda, AWS SQS, AWS *Eventbridge*, dan AWS DynamoDB. Seluruh *service* yang dibutuhkan untuk aplikasi *Tag Tracker* di-*deploy* secara otomatis menggunakan AWS *Cloud Formation*. Aplikasi *Tag Tracker* memiliki dua layanan yaitu layanan aggregator dan *dashboard* yang bisa diakses oleh pengguna dari tim bisnis internal perusahaan. Pembahasan ini bertujuan untuk memberikan wawasan menyeluruh tentang kinerja dan keandalan aplikasi *Tag Tracker* dalam memenuhi kebutuhan pengguna yang diuji dari sisi fungsionalitas aplikasi menggunakan metode *blackbox testing* dan waktu komputasi dari layanan *dashboard* untuk dapat menampilkan data terkait jumlah *views* dalam satu minggu terakhir.



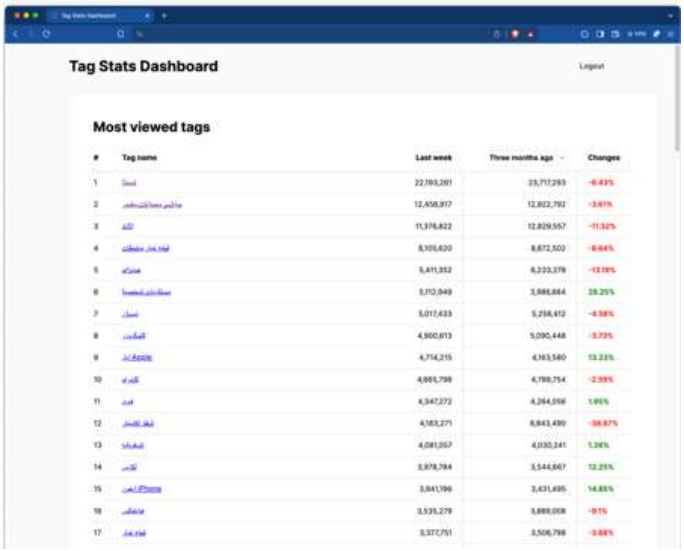
Gambar 4. Halaman login.

Pada Gambar 4 menunjukkan halaman login pada aplikasi *Tag Tracker* yang dirancang untuk mengautentikasi pengguna sebelum mereka dapat mengakses fitur-fitur utama aplikasi. Proses autentikasi ini penting untuk memastikan bahwa hanya pengguna yang sah yang dapat mengakses data dan fungsionalitas yang dilindungi.

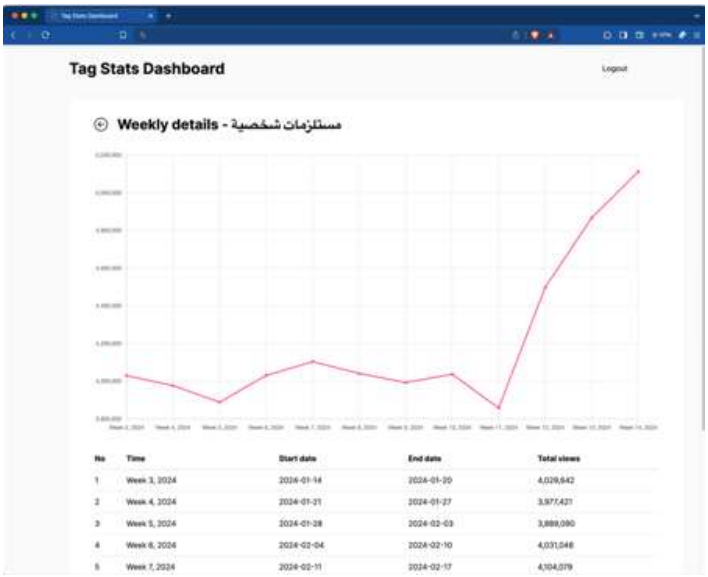
Pada Gambar 5 menunjukkan halaman beranda *dashboard.* Halaman *Tag Start Dashboard* pada aplikasi *Tag Tracker* dirancang untuk memberikan informasi statistik mengenai tag yang digunakan dalam sistem. *Dashboard* ini menampilkan *name tag*, jumlah *view* dalam minggu terakhir, jumlah *view* tiga bulan yang lalu, serta persentase perubahan jumlah *view*. Fitur ini bertujuan untuk memberikan pengguna wawasan yang mendalam tentang popularitas dan tren penggunaan tag dari waktu ke waktu.

Pada Gambar 6 menunjukkan Halaman Detail Statistika pada aplikasi *Tag Tracker* dirancang untuk memberikan informasi statistik yang lebih mendalam tentang *tag* tertentu yang dipilih oleh pengguna. Halaman ini menampilkan detail statistik seperti jumlah *view* mingguan selama periode tertentu, tren pertumbuhan jumlah *view*, dan perbandingan dengan periode sebelumnya. Fitur ini dirancang untuk memberikan pengguna wawasan yang lebih mendetail tentang

bagaimana penggunaan tag telah berubah dari waktu ke waktu dan ditampilkan dalam bentuk visual yaitu dengan grafik baris.



Gambar 5. Halaman beranda.

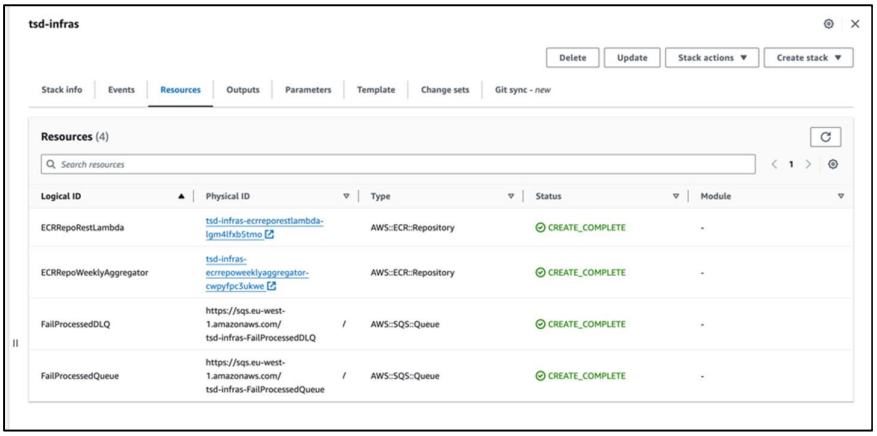


Gambar 6. Halaman detail statistika.

Pada penelitian ini, seluruh *service* yang dibutuhkan untuk aplikasi *Tag Tracker* di-*deploy* secara otomatis menggunakan AWS *Cloud Formation*. AWS *Cloud Formation* adalah layanan manajemen infrastruktur berbasis kode IaC (*Infrastructure as Code*) dari Amazon Web *Services* (AWS) yang memungkinkan pengguna untuk mendefinisikan dan mengelola infrastruktur AWS sebagai kode [15]. *Resource* yang digunakan terdiri dari infrastruktur, *service dashboard* dan *service aggregator.*

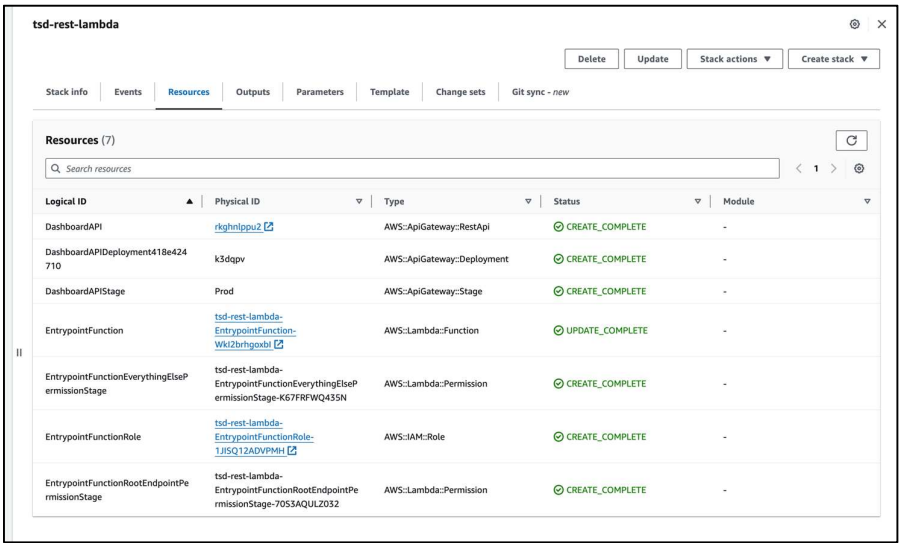
Pada Gambar 7 menunjukkan file AWS CloudFormation yang dideskripsikan sebagai *template* untuk mendeploy infrastruktur yang diperlukan untuk dashboard statistik *Tag*. Di dalam *template* ini, terdapat beberapa sumber daya (*resources*) yang didefinisikan yaitu *ECRRepoRestLambda* merupakan sebuah *repository* Amazon ECR (*Elastic Container Registry*) yang digunakan untuk menyimpan image Docker yang digunakan oleh layanan REST melalui Lambda.

ECR Repo Weekly Aggregator merupakan *repository* Amazon ECR tambahan yang digunakan untuk menyimpan *image Docker* untuk sebuah aggregator mingguan. *Fail Process Queue* merupakan antrian SQS (*Simple Queue Service*) yang digunakan sebagai invoker kedua untuk sebuah layanan dan *Fail Processed DLQ* merupakan antrian SQS yang bertindak sebagai *Dead Letter Queue* (DLQ) untuk antrian *Fail Processed Queue*. Antrian ini juga memiliki konfigurasi dengan masa retensi pesan selama 1 minggu.



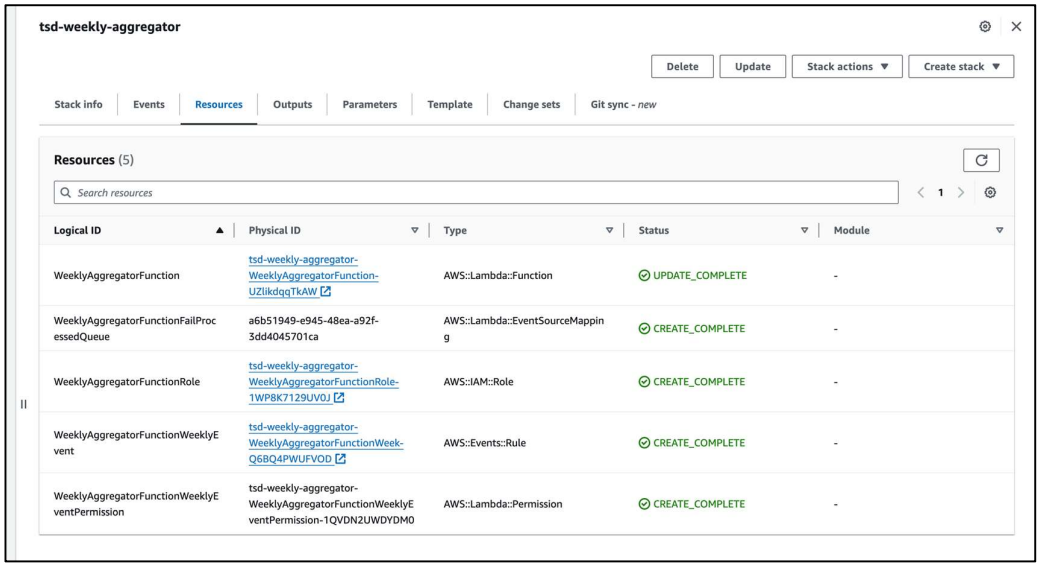
Gambar 7. *Resource* untuk infrastruktur yang digunakan.

Pada Gambar 8 menunjukkan sebuah SAM (*Serverless Application Model*) *template* yang digunakan untuk mendeploy aplikasi web *Dashboard* melalui Lambda untuk statistik *Tag*. Dengan menggunakan *template* SAM ini, infrastruktur untuk aplikasi *dashboard* web dapat dideskripsikan dan di-*deploy* dengan cara yang terstandarisasi dan mudah dikelola di AWS *CloudFormation*.



Gambar 8. Resource untuk service aplikasi web dashboard.

Pada Gambar 9 menunjukkan AM (*Serverless Application Model*) *template* yang digunakan untuk mendeploy layanan *Weekly* Aggregator untuk *dashboard* statistik *Tag*. *Resource Weekly Aggregator Function* merupakan Fungsi Lambda yang melakukan agregasi *views* distribusi tag untuk seminggu dari tabel *TagDistributionViews.* Dengan menggunakan template SAM ini, layanan *Weekly Aggregator* untuk *dashboard* statistik *Tag* dapat dideskripsikan dan deploy dengan cara yang terstandarisasi dan mudah dikelola di AWS *Cloud Formation*. Template ini mengintegrasikan berbagai sumber daya AWS seperti Lambda, DynamoDB, dan SQS untuk membangun solusi yang efisien dan *scalabel.*



Gambar 9. *Resource* untuk *service* aggregator.

Tabel 1. Waktu Komputasi layanan *dashboard*.

Iterasi	Waktu komputasi (detik)
1	1,36
2	1,42
3	1,77
4	1,21
5	1,96

Pada Tabel 1 menunjukkan waktu komputasi yang dibutuhkan pada layanan *dashboard* untuk melakukan pemrosesan saat data diterima hingga disimpan pada dynamoDB dan ditampilkan pada halaman web. Percobaan ini dilakukan sebanyak 5 kali iterasi dan didapat nilai waktu komputasi berada pada range waktu 1 hingga 2 *minutes*. Untuk menampilkan jumlah *view* mingguan pada sistem *Tag Tracker* dengan lambda dan dynamodb. Waktu komputasi ini diukur pada kondisi *hot start*. *Hot start* terjadi ketika fungsi Lambda dipanggil dan ada *instance* yang sudah berjalan dari fungsi tersebut. AWS hanya perlu meneruskan permintaan ke *instance* yang sudah aktif. Sedangkan *cold start* terjadi saat pengguna pertama kali mengakses halaman setelah periode tidak aktif, Lambda mungkin memerlukan waktu lebih lama untuk memproses permintaan karena AWS harus memuat *runtime*, menginisialisasi konteks, dan kemudian menjalankan kode fungsi.

Tabel 2. Pengujian *black box testing*.

Fitur	**Skenario**	**Hasil**
Login	User menginputkan *password* dan *username* dengan benar, maka sistem akan menampilkan halaman utama	Sukses
	User menginputkan *password* dan *username* dengan salah, maka sistem akan menampilkan notifikasi kesalahan input	Sukses

Halaman Dashboard	*User* memilih menu *dashboard*, maka sistem akan menampilkan grafik statistik tag dalam satu minggu	Sukses
Halaman Aggregator	*User* memilih menu aggregator, maka sistem akan menampilkan data statistik jumlah view dari berbagai tag yang telah diambil dari database dynamoDB	Sukses

Pada Tabel 2 menunjukkan pengujian fungsional aplikasi ini dilakukan menggunakan metode *black box testing* untuk memastikan bahwa setiap fitur berfungsi sesuai dengan yang diharapkan. Pertama, pada skenario login, pengguna yang memasukkan *username* dan *password* yang benar akan diarahkan ke halaman utama, menunjukkan bahwa sistem berhasil memverifikasi kredensial dengan benar. Sebaliknya, jika pengguna memasukkan *username* atau *password* yang salah, sistem akan menampilkan notifikasi kesalahan input, menunjukkan bahwa sistem dapat menangani kredensial yang tidak valid dengan benar.

Selanjutnya, pada halaman dashboard, ketika pengguna memilih menu *dashboard*, sistem diharapkan menampilkan grafik statistik tag selama satu minggu terakhir. Pengujian ini bertujuan untuk memastikan bahwa data yang ditampilkan sesuai dengan data yang tersimpan dan dihitung dengan benar. Terakhir, pada halaman aggregator, pengguna yang memilih menu aggregator akan melihat data statistik jumlah view dari berbagai tag yang diambil dari *database* DynamoDB. Pengujian ini memastikan bahwa integrasi antara aplikasi dan *database* berfungsi dengan baik, serta data yang ditampilkan akurat dan *up-to-date*.

Dengan menggunakan metode black box testing, pengujian dilakukan dengan fokus pada validasi output berdasarkan input yang diberikan tanpa memeriksa kode internal. Setiap fungsi diuji dengan berbagai kombinasi input yang mungkin, dan hasil yang diharapkan dibandingkan dengan hasil yang sebenarnya. Hasil tes dicatat untuk menentukan apakah setiap fitur berfungsi dengan benar (sukses) atau memerlukan perbaikan. Pendekatan ini memastikan bahwa aplikasi berfungsi dengan baik dari perspektif pengguna, menangani semua skenario yang mungkin terjadi dengan tepat.

4. SIMPULAN

Penelitian ini menghasilkan sistem *Tag Tracker* yang memiliki dua layanan utama yaitu *dashboard* untuk menampilkan statistik dari "*tag*" mingguan pada perusahaan dan *aggregator* untuk menghitung total *views* sebuah tag dalam satu pekan. Aplikasi ini memanfaatkan *platform* AWS dalam proses *develop* dan deploy *aplikasi*. Berdasarkan hasil pengujian waktu komputasi pada layanan *dashboard* yang dibutuhkan untuk pemrosesan data saat data diterima hingga disimpan di dynamoDB dan ditampilkan pada grafik *dashboard* dibutuhkan waktu antara 1-2 minutes. Pengujian dengan metode *black box* dilakukan untuk menguji fungsionalitas aplikasi dan diperoleh 100% tingkat keberhasilan fungsi. Aplikasi *Tag Tracker* dapat digunakan oleh internal perusahaan untuk mendapatkan informasi yang akurat terkait popularitas setiap tag di pelanggan, sehingga dapat menjadi pertimbangan dalam menentukan model bisnis perusahaan. Pada proses penelitian selanjutnya dapat dikembangkan sebuah sistem yang mengintegrasikan layanan AWS seperti Amazon *SageMaker* untuk menerapkan *machine learning* pada data *tag*. Sehingga analisis prediktif bisa dilakukan untuk memberikan wawasan lebih lanjut tentang tren dan pola dari perilaku pengguna sistem.

5. Ucapan Terima Kasih

Kami ucapkan terima kasih pada pihak Politeknik Elektronika Negeri Surabaya sebagai tempat penelitian terkait perancangan untuk membangun aplikasi "Optimalisasi *Monitoring Tag* Dengan AWS *Cloud*: Studi Kasus Aplikasi *Tag Tracker* pada PT. XYZ" sehingga sistem ini dapat diimplementasikan dengan baik.

REFERENSI

[1] H. Dhika, T. Akhirina, D. Mustari, and F. Destiawati, "Pemanfaatan Teknologi Cloud Computing sebagai Media Penyimpanan Data," *J. PkM Pengabdi. Kpd. Masy.*, vol. 2, no. 03, p. 221, 2019, doi: 10.30998/jurnalpkm.v2i03.3144.

[2] I. I. Sholihin, A. T. Zy, and U. D. Soer, "Rancang bangun sistem aplikasi e-cashier berbasis web dengan metode rapid application development," *INFOTECH J. Inform. Teknol.*, vol. 5, no. 1, pp. 14–26, 2024, doi: 10.37373/infotech.v5i1.970.

[3] Julia, M. I. Mutahari, Renaldi, and Saepullah, "Analisis Kinerja Basis Data Terdistribusi dalam Linkungan Cloud Computing," *Karimah Tauhid*, vol. 3, no. 2, pp. 1771–1782, 2024, doi: 10.30997/karimahtauhid.v3i2.11907.

[4] Haeruddin, "Ketersediaan Tinggi Infrastruktur Elearning Berbasis Komputasi Awan," *J. Tek. Inform. dan Sist. Inf.*, vol. 10, no. 1, pp. 670–682, 2023, doi: https://doi.org/10.35957/jatisi.v10i1.5050.

[5] Z. Z. Maulidia and L. Venica, "Serverless Computing: Cloud Run and App Engine Analysis in Profile Website Deployment," *Sistemasi*, vol. 13, no. 2, p. 492, 2024, doi: 10.32520/stmsi.v13i2.3005.

[6] D. N. Alfarizi and I. H. Ikasari, "Tinjauan Literatur Terhadap Pemanfaatan Cloud Computing," *JURIHUM J. Inov. dan Hum.*, vol. 01, no. 01, pp. 148–154, 2023, [Online]. Available: https://jurnalmahasiswa.com/index.php/jurihum

[7] M. S. Rumetna, "Title Case," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 3, pp. 305–314, 2018, doi: 10.25126/jtiik.201853595.

[8] R. Mardikaningsih and D. Darmawan, "Strategi Inovasi Bisnis Sebagai Upaya Peningkatan Keunggulan Kompetitif Dan Pertumbuhan Bisnis UMKM Industri Kreatif Di Era Digital," *GLORY (Glob. Leadersh. Organ. Res. Manag.*, vol. 1, no. 4, pp. 371–386, 2023, doi: https://doi.org/10.59841/glory.v1i4.925.

[9] E. Barus, K. M. Pardede, and J. A. Putri Br. Manjorang, "Transformasi Digital: Teknologi Cloud Computing dalam Efisiensi Akuntansi," *J. Sains dan Teknol.*, vol. 5, no. 3, pp. 904–911, 2024, doi: 10.55338/saintek.v5i3.2862.

[10] Fandy, Rosmasari, and G. M. Putra, "Pengujian Kinerja Web Server Atas Penyedia Layanan Elastic Cloud Compute (EC2) Pada Amazon Web Services (AWS)," *Adopsi Teknol. dan Sist. Inf.*, vol. 1, no. 1, pp. 21–35, 2022, doi: 10.30872/atasi.v1i1.45.

[11] M. Syahrul Mubarok and M. Izman Herdiansyah, "Implementasi Cloud Computing Amazon Web Services (AWS) Pada Web Reservasi Kamar Hotel," *Kaji. Ilm. Inform. dan Komput.*, vol. 4, no. 2, pp. 698–708, 2023, doi: 10.30865/klik.v4i2.1212.

[12] E. Kristianto, "The Needs of Virtual Machines Implementation in Private Cloud Computing Environment," *ComTech Comput. Math. Eng. Appl.*, vol. 6, no. 4, p. 525, 2015, doi: 10.21512/comtech.v6i4.2181.

[13] E. Riana, "Implementasi Cloud Computing Technology dan Dampaknya Terhadap Kelangsungan Bisnis Perusahaan Dengan Menggunakan Metode Agile dan Studi Literatur," *JURIKOM (Jurnal Ris. Komputer)*, vol. 7, no. 3, p. 439, 2020, doi: 10.30865/jurikom.v7i3.2192.

[14] F. M. Putra and R. Sari, "Aplikasi Business Intelligence Dashboard sebagai Alat Monitoring dan Bahan Pengambilan Keputusan Sales and Account Receivable," *Multinetics*, vol. 2, no. 1, p. 35, 2016, doi: 10.32722/multinetics.vol2.no.1.2016.pp.35-42.

[15] N. Ramsari and A. Ginanjar, "Implementasi Infrastruktur Server Berbasis Cloud Computing Untuk Web Service Berbasis Teknologi Google Cloud Platform," *Conf. Senat. STT Adisutjipto Yogyakarta*, vol. 7, no. August, 2022, doi: 10.28989/senatik.v7i0.472.