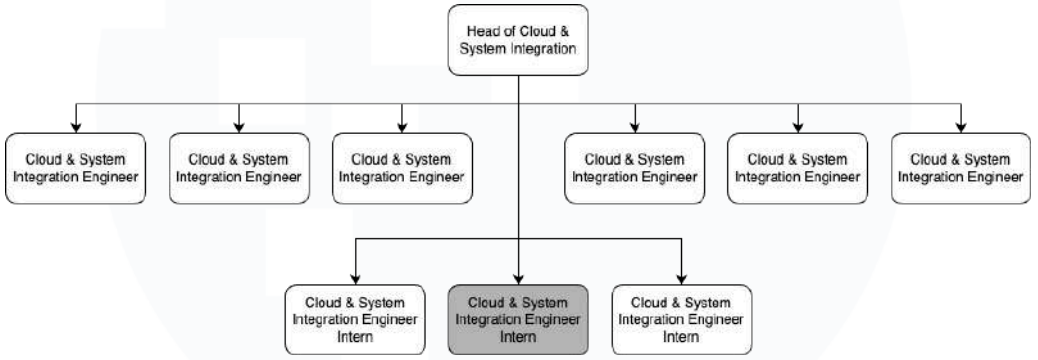
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama program kerja magang di PT InfraCom Technology, posisi yang diberikan adalah sebagai anggota tim *Cloud & System Integration Engineer Intern*.



Gambar 3.1 Struktur Kedudukan Cloud & System Integration Engineer Intern

Gambar 3.1 ini, tanggung jawab mencakup dukungan terhadap integrasi infrastruktur cloud perusahaan, peningkatan efisiensi sistem, serta memastikan kelancaran layanan teknologi informasi yang mendukung berbagai aktivitas bisnis perusahaan.

3.1.1 Media yang Digunakan dalam Proses Magang

Dalam proses magang, berbagai media digunakan untuk mendukung kegiatan dan mempermudah komunikasi, yaitu:

- Microsoft Teams → digunakan sebagai sarana untuk share screen saat pertemuan mingguan yang dilakukan secara *offline*, memudahkan penyampaian materi dan kolaborasi visual dengan tim.

- WhatsApp → digunakan sebagai platform komunikasi utama untuk koordinasi harian dan pertukaran informasi antar anggota tim, memastikan respons yang cepat dan efisien terhadap perubahan situasi.
- Google Spreadsheet → digunakan untuk pencatatan dan pelaporan daily task yang dikerjakan selama magang, memungkinkan pengawasan *progress* secara terstruktur dan transparan oleh *Supervisor*. Selain itu, digunakan untuk menampilkan daftar tugas yang perlu dikerjakan serta status tugas yang sudah selesai, sehingga memudahkan manajemen waktu dan prioritas kerja.

3.2 Tugas dan Uraian Kerja Magang

Tugas yang diberikan adalah ikut serta membantu mengerjakan hal-hal teknis yang dibutuhkan klien mengenai *cloud computing*. Namun sebelum dapat membantu mengerjakan hal-hal tersebut, akan diberikan waktu untuk meningkatkan pengetahuan, pemahaman dan keterampilan teknis terkait layanan *cloud computing* selama kurang lebih 1,5 bulan kemudian mengambil sertifikasi untuk memvalidasi apakah pengetahuan dan pemahaman sudah cukup untuk membantu mengerjakan hal-hal tersebut. Tugas yang diberikan saat magang ditampilkan pada Tabel 3.1.

Tabel 3.1 Tugas Kerja Magang

No	Tugas	Tanggal Mulai	Tanggal Selesai
1	Pembelajaran Cloud	17 Februari 2025	13 Juni 2025
1.1	AWS	17 Februari 2025	13 Juni 2025
1.2	OCI	18 Maret 2025	9 April 2025
2	Project Monitoring Service Klien	9 April 2025	9 Juni 2025
2.1	Laporan Mingguan	30 April 2025	9 Juni 2025
2.2	Laporan Bulanan	9 April 2025	5 Juni 2025
2.3	Laporan Preventive Maintenance	9 April 2025	5 Juni 2025
3	Project Migrasi Klien	2 Mei 2025	13 Juni 2025
3.1	Migrasi Dari GCP ke AWS	2 Mei 2025	13 Juni 2025

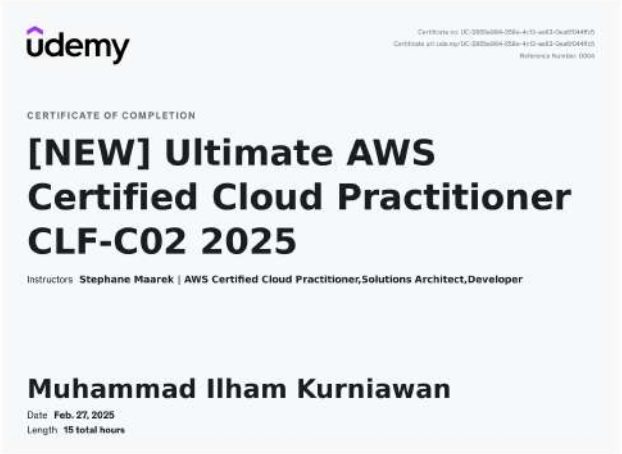
3.2.1 Pembelajaran Cloud

Kegiatan pembelajaran mandiri untuk meningkatkan pengetahuan, pemahaman dan keterampilan teknis terkait layanan *cloud computing* yang sesuai dengan kebutuhan proyek perusahaan. Pembelajaran dilakukan melalui berbagai sumber seperti dokumentasi resmi, kursus *online*, dan *hands-on lab* untuk mempersiapkan diri dalam menangani tugas-tugas proyek.

3.2.1.1 AWS

Mempelajari layanan-layanan fundamental dari AWS dilakukan secara sistematis dan bertahap, dengan tujuan membangun pemahaman mulai dari konsep dasar hingga kemampuan untuk merancang solusi yang cukup kompleks. Setiap tahapan dirancang untuk saling melengkapi dan menjawab tantangan yang dihadapi selama periode magang.

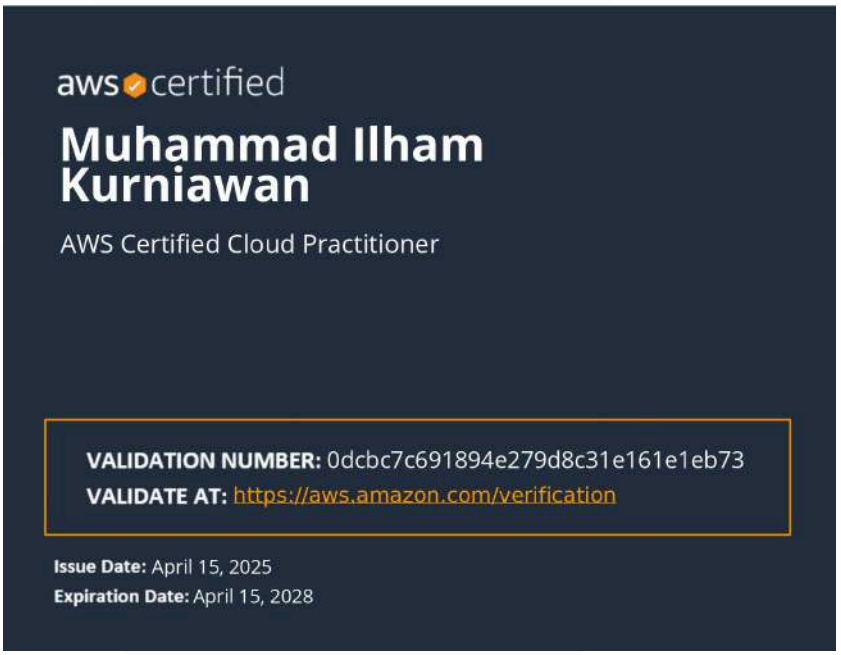
Pintu utama untuk memasuki ekosistem AWS adalah melalui persiapan sertifikasi **AWS *Certified Cloud Practitioner***. Pembelajaran pada tahap ini tidak hanya berfokus pada "apa" itu layanan AWS, tetapi juga "mengapa" dan "bagaimana" AWS memberikan nilai bagi bisnis. Pada proses pembelajarannya dengan mengikuti kursus online di Udemy yang secara sistematis membahas empat domain utama: konsep *cloud*, keamanan dan kepatuhan, teknologi, serta penagihan dan harga. Pada prosesnya akan belajar mengenai konsep fundamental seperti *Shared Responsibility Model*, di mana AWS bertanggung jawab atas keamanan dari *Cloud*, dan pelanggan bertanggung jawab atas keamanan di dalam *Cloud*. Memahami struktur harga dan alat seperti AWS Cost Explorer juga menjadi kunci, karena efisiensi biaya adalah pilar penting dalam adopsi cloud.



Gambar 3.2 Sertifikat Udemy *AWS Certified Cloud Practitioner*

Setelah berhasil, menyelesaikan kursus online di Udemy kemudian akan mendapatkan sertifikat seperti pada Gambar 3.2, dimana sertifikat ini menjadi pertanda bahwa telah berhasil menyelesaikan kursus AWS *Certified Cloud Practitioner*, kemudian dilanjutkan dengan mengikuti ujian AWS *Certified Cloud Practitioner* yang diselenggarakan langsung oleh AWS dengan sertifikat seperti pada Gambar 3.3.

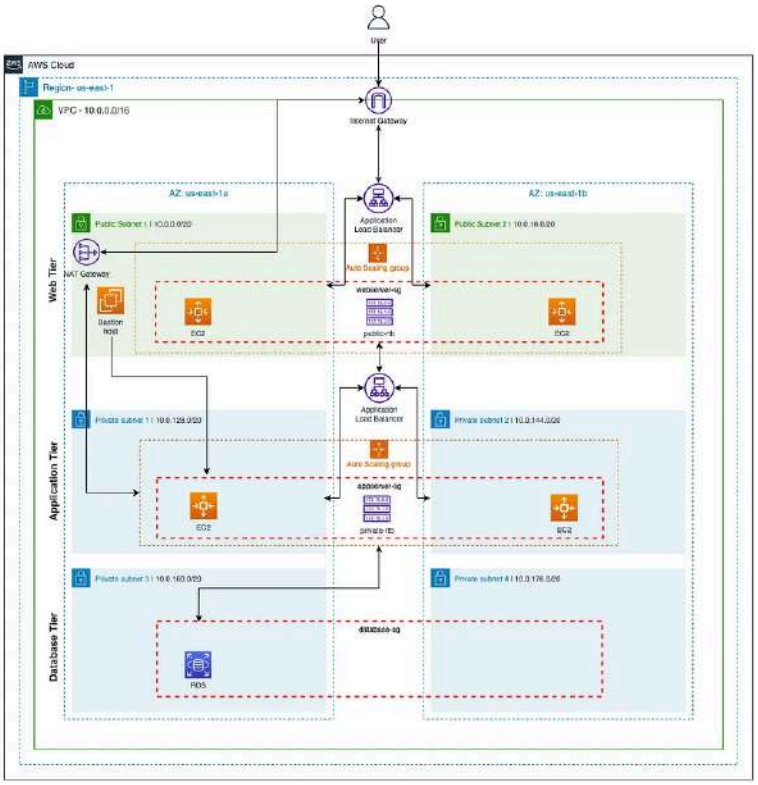
Gambar 3.3 *AWS Certified Cloud Practitioner Badge*



Gambar 3.4 *AWS Certified Cloud Practitioner*

Tahapan ini diakhiri dengan keberhasilan lulus ujian sertifikasi resmi **AWS *Certified Cloud Practitioner***. Sertifikasi ini menjadi validasi formal atas pengetahuan dasar yang solid dan membangun kepercayaan diri untuk melangkah ke topik yang lebih kompleks, dengan bukti sertifikat bisa dilihat pada Gambar 3.4.

Setelah memiliki pondasi yang kuat, *supervisor* memberikan tugas untuk mencoba melakukan hands on lab dengan membuat *3-tier Architecture* dengan dimulai membuat dari sisi *network, security, virtual machine,* dan *database*. dengan *design architecture* sebagai berikut.



Gambar 3.5 *3-tier Architecture design*

Pembelajaran yang dilakukan seperti pada Gambar 3.5 bertujuan untuk meningkatkan kemampuan yang tidak hanya mengeksekusi tugas, tetapi juga berkontribusi pada diskusi desain arsitektur, memberikan rekomendasi, dan pada akhirnya memberikan nilai lebih bagi tim dan perusahaan.

3.2.1.2 OCI

Sebagai bagian dari strategi pengembangan kompetensi *multi-cloud*, memperluas wawasan di luar AWS dengan mendalami ***Oracle Cloud Infrastructure*** **(OCI)**. Keputusan untuk mempelajari OCI didasarkan pada reputasinya yang kuat di kalangan enterprise, terutama untuk beban kerja yang menuntut performa tinggi seperti database, serta arsitektur jaringannya yang dirancang untuk kinerja yang dapat diprediksi. Tujuannya adalah untuk memahami secara mendalam keunggulan kompetitif OCI dan mampu melakukan analisis perbandingan yang objektif terhadap pendekatan arsitektur cloud yang berbeda.

Seluruh proses pembelajaran menggunakan kursus *online* dari Oracle yaitu **MyLearn Oracle**, portal edukasi resmi yang menyediakan alur belajar (*learning path*) yang komprehensif dan mandiri (*self-paced*). Mengikuti kurikulum untuk ***Oracle Cloud Infrastructure Foundations***, yang dirancang untuk membangun pemahaman konseptual dan praktis dari nol dengan tujuan untuk menyamakan persepsi antara AWS dan *Oracle Cloud Infrastructure*. Setelah berhasil menyelesaikan kursus *online* dari Oracle, kemudian mempersiapkan diri dan mengikuti ujian sertifikasi yang diselenggarakan langsung oleh Oracle, kemudian berhasil lulus dan secara resmi memperoleh sertifikasi ***Oracle Cloud Infrastructure* 2025 *Foundations Associate*** seperti yang bisa dilihat pada Gambar 3.6 dan Gambar 3.7.

Gambar 3.6 *Oracle Certified Foundations Associate Badge*

Gambar 3.7 *Oracle Certified Foundations Associate*

Pencapaian ini lebih dari sekadar sertifikat, ini merupakan komitmen dan merupakan bukti atas kemampuan untuk beradaptasi dan

menguasai ekosistem cloud yang berbeda. Dengan memiliki pemahaman fundamental di dua platform cloud utama (AWS dan OCI), akan memberikan perspektif yang lebih luas, tidak terjebak dalam pola pikir satu vendor, dan mampu memberikan analisis serta rekomendasi teknis yang lebih objektif dan berbobot.

3.2.2 Project Monitoring Service Client

Setelah berhasil memahami dasar cloud computing kini diberikan tugas dan terlibat secara aktif dalam proyek pemantauan layanan (*service monitoring*) untuk klien perusahaan. Proyek ini bertujuan untuk memastikan ketersediaan (*availability*), kinerja (*performance*), dan keandalan (*reliability*) dari aplikasi atau infrastruktur milik klien. Tugas yang diberikan adalah membantu dalam pembuatan laporan mingguan, membuat laporan bulanan dan membuat laporan *Preventive Maintenance* (PM).

3.2.2.1 Laporan Mingguan

Laporan Mingguan adalah dokumen operasional yang dibuat secara rutin setiap hari rabu untuk mengetahui kondisi dan performa layanan klien dalam skala waktu yang singkat. Laporan ini bersifat taktis dan berfungsi sebagai alat komunikasi utama bagi tim teknis *internal* untuk memantau kesehatan sistem secara berkala, mendeteksi anomali lebih awal, dan merencanakan tindakan perbaikan untuk minggu berikutnya.

Pada pembuatan laporan mingguan akan diberikan tugas dan peran untuk mengambil tangkapan layar dari data secara rutin dari sistem *monitoring* **Amazon CloudWatch**, dan grafik-grafik yang relevan, kemudian membuat rangkuman dan dikirimkan ke penanggung jawab pembuatan laporan mingguan.

3.2.2.2 Laporan Bulanan

Laporan Bulanan merupakan dokumen analisis komprehensif yang bertujuan untuk memberikan gambaran tingkat tinggi mengenai performa, kesehatan, dan tren layanan klien selama satu bulan penuh. Berbeda dengan laporan mingguan yang lebih bersifat taktis, laporan bulanan

berfokus pada analisis strategis dan ditujukan bagi manajer, pimpinan tim, atau klien untuk pengambilan keputusan.

Pada pembuatan laporan bulanan akan diberikan tugas dan peran untuk mengambil tangkapan layar dari data secara rutin dari sistem monitoring **Amazon CloudWatch**, dan grafik-grafik yang relevan, kemudian mengolah data tersebut menjadi tabel dan penjelasan yang mudah dibaca dan dipahami, dan menyusun draft laporan berdasarkan template yang ada untuk kemudian ditinjau oleh supervisor.

3.2.2.3 Laporan Preventive Maintenance

Laporan *Preventive Maintenance* (PM) adalah dokumen teknis yang berorientasi pada tindakan proaktif untuk mencegah terjadinya masalah di masa depan. Laporan ini mencatat semua aktivitas pemeliharaan rutin yang telah dilakukan pada infrastruktur klien dan temuan-temuan dari aktivitas tersebut.

Dalam kegiatan ini, diberikan tugas untuk membantu dalam menjalankan beberapa tugas, melakukan verifikasi hasil (misalnya, memastikan versi *software* sudah ter-*update*), mendokumentasikan setiap langkah dan hasilnya ke dalam laporan PM, serta melacak status temuan untuk memastikan semuanya ditindaklanjuti.

3.2.3 Project Migrasi Klien

Kemudian ikut berpartisipasi dalam proyek migrasi infrastruktur *cloud* milik klien dari *Google Cloud* ke *Amazon Web Service*. Proyek ini melibatkan perencanaan, eksekusi, dan validasi pemindahan sistem untuk mencapai efisiensi biaya yang lebih baik, performa yang lebih tinggi, atau memanfaatkan fitur spesifik dari platform tujuan.

Salah satu aspek terpenting dalam setiap proyek migrasi adalah memastikan keamanan data klien selama proses pemindahan. Untuk klien yang memiliki data sensitif atau aplikasi *internal*, memindahkan data melalui internet publik seringkali bukan pilihan utama. Oleh karena itu, pendekatan migrasi melalui jaringan *private* menjadi strategi krusial yang harus dikuasai.

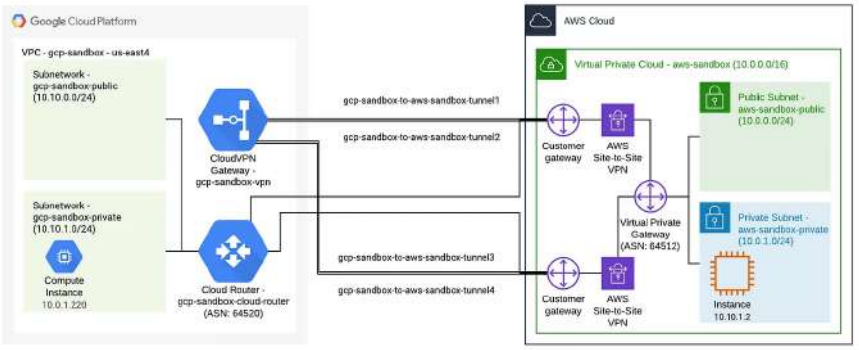
3.2.3.1 Simulasi Proses Migrasi dari GCP ke AWS Melalui Jaringan Privat

Simulasi ini akan menjabarkan langkah-langkah teknis yang digunakan dalam memigrasikan sebuah *Virtual Machine* (VM) dari lingkungan *Google Cloud Platform* (GCP) ke *Amazon Web Services* (AWS). Sesuai dengan judul, fokus utama dari skenario ini adalah pelaksanaan seluruh proses migrasi melalui koneksi jaringan *private* yang aman, tanpa ada data yang terekspos ke internet publik.

Untuk mewujudkan hal tersebut, pendekatan yang digunakan adalah dengan membangun koneksi *Virtual Private Network* (VPN) *site-to-site* antara VPC di GCP dan VPC di AWS. Koneksi ini akan membentuk jalur komunikasi terenkripsi dan tertutup yang memungkinkan *resource* dari kedua *cloud* dapat saling berinteraksi secara langsung, seolah berada dalam jaringan lokal yang sama. Dengan cara ini, seluruh proses transfer data dapat dilakukan tanpa melibatkan internet publik, sehingga keamanan dan integritas data tetap terjaga selama proses migrasi.

a. Tahap Pertama: Membangun Koneksi Privat antar Cloud

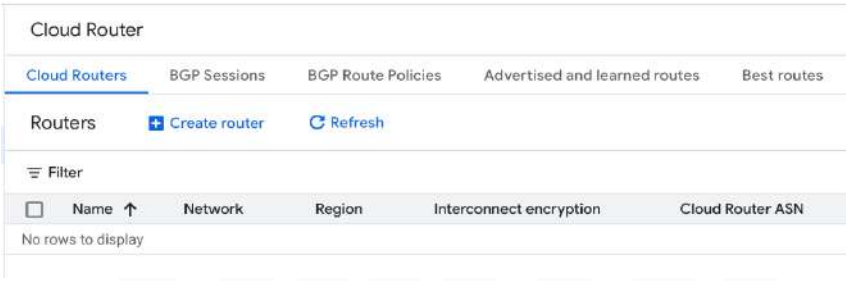
Tahap pertama dalam simulasi ini adalah membangun infrastruktur koneksi *private* tersebut, dimulai dari pembuatan dan konfigurasi VPN di kedua lingkungan *cloud*. Tahapan ini mencakup penyusunan arsitektur jaringan, pembuatan VPN *Gateway*, pengaturan routing, serta pengujian konektivitas antar-*cloud* untuk memastikan kedua lingkungan dapat berkomunikasi secara aman dan stabil.



Gambar 3.8 Arsitektur Jaringan *Site-to-Site* VPN

Gambar 3.8 merupakan gambar arsitektur jaringan VPN dengan menggunakan *High-availability (HA)* VPN yang terdapat 4 *tunnel* yang menghubungkan antara *Google Cloud Platform* (GCP) *dengan Amazon Web Service* (AWS). dibawah ini merupakan langkah langkah untuk mengkonfigurasi sebuah jaringan *site-to-site* VPN.

Masuk ke *console* GCP, cari ***Cloud Router***, klik ***Create Router*** seperti pada Gambar 3.9 di bawah ini**.**

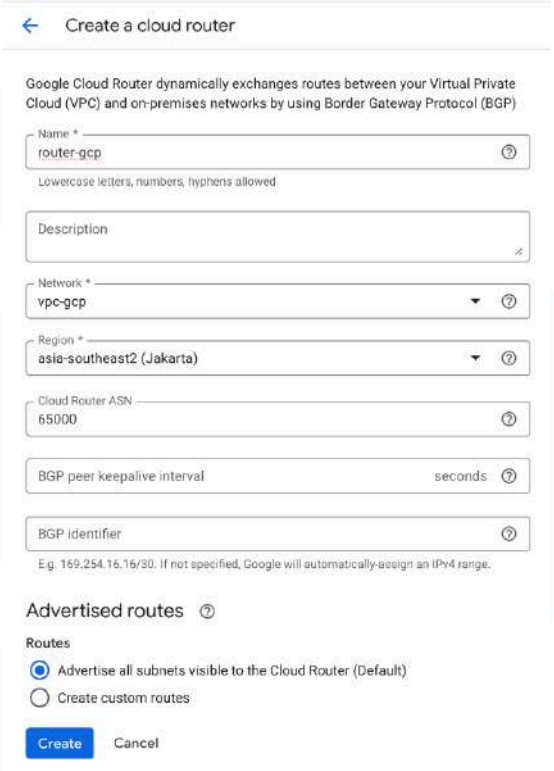


Gambar 3.9 *Console GCP Dashboard Cloud Router*

Konfigurasinya adalah sebagai berikut.

- *Name* : router-gcp
- *Network* : Pilih VPC yang digunakan
- *Cloud Router ASN* : 65000
- Klik ***Create***.

Untuk lebih lengkapnya, bisa dilihat pada Gambar 3.10 di bawah ini.

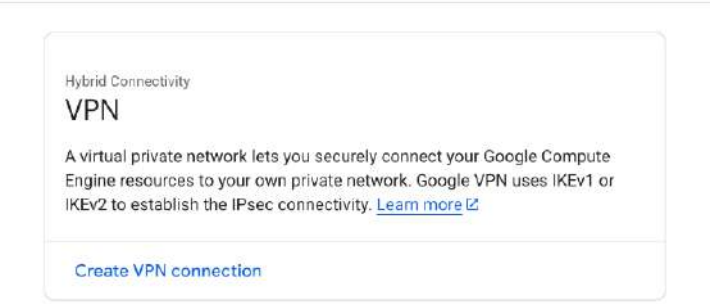


Gambar 3.10 *Console GCP Create Cloud Router*

Cloud Router yang telah berhasil dibuat akan seperti Gambar 3.11.

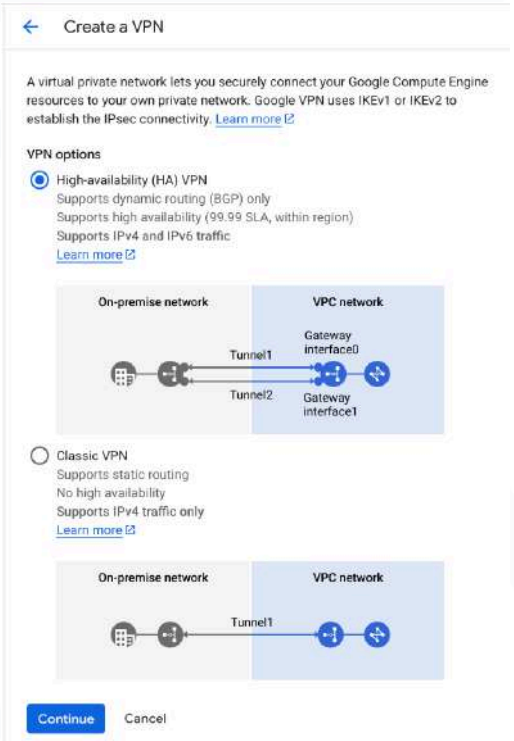
Gambar 3.11 *Console GCP Cloud Router*

Masuk ke halaman VPN, klik ***Create VPN Connection*** seperti pada Gambar 3.12 di bawah ini.



Gambar 3.12 *Console GCP Dashboard VPN*

Pilih ***High-availability (HA) VPN***, klik ***Continue*** seperti pada Gambar 3.13 di bawah ini.



Gambar 3.13 *Console GCP Create VPN*

Isi konfigurasinya sebagai berikut.

- *VPN Gateway Name* : vpg-gcp
- *Network* : Pilih VPC yang digunakan

- *Region* : Pilih asia-southeast2 (Jakarta)
- klik ***Create & continue***

Jika sudah melakukan semua tahapan, maka akan seperti gambar 3.14 di bawah ini.

Gambar 3.14 *Console GCP Create VPN Gateway*

Akan diberikan 2 IP **34.153.44.155** dan **34.153.236.200**, yang akan digunakan *Customer Gateway AWS* seperti pada Gambar 3.15 di bawah ini.

Gambar 3.15 *Console GCP Add VPN Tunnels*

Buka *console* AWS, cari ***Customer Gateways***, klik ***Create Customer Gateway*** seperti pada Gambar 3.16 di bawah ini.

Gambar 3.16 *Console AWS Dashboard Customer Gateway*

Konfigurasinya adalah sebagai berikut.

- *Name* : cgw-aws-1
- *BGP ASN* : 65000 (Isi dengan ASN GCP)
- *IP address* : 34.153.44.155 (IP dari GCP)
- Klik ***Create***.

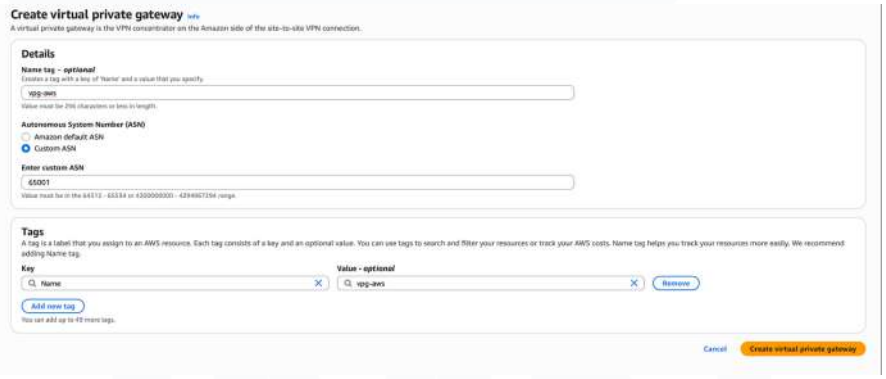
Untuk lengkapnya bisa dilihat pada Gambar 3.17 di bawah, dengan tambahan catatan buat 2 *customer gateway* dengan IP yang diberikan gcp.

Gambar 3.17 *Console AWS Create Customer Gateway*

Selanjutnya cari ***Virtual Private Gateways***, klik ***Create Virtual Private Gateway*** seperti pada Gambar 3.18 di bawah ini.

Gambar 3.18 *Console AWS Dashboard Virtual Private Gateway*

Isi konfigurasinya seperti ***Name*** dan ***Custom ASN*** **(65001 (ASN untuk AWS)),** klik ***Create Virtual Private Gateway*** seperti pada Gambar 3.19 di bawah ini.

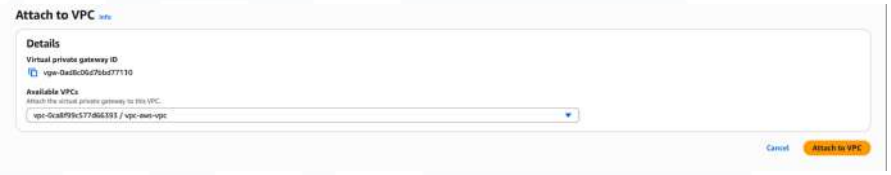


Gambar 3.19 *Console AWS Create Virtual Private Gateway*

Tunggu hingga statusnya berubah menjadi *detached.* Pilih ***virtual private gateway*** yang sudah dibuat, klik ***action*** dan ***attach to VPC*** seperti pada Gambar 3.20 di bawah ini.

Gambar 3.20 *Console AWS Dashboard Virtual Private Gateway*

Pilih VPC yang digunakan, klik ***attach to VPC*** seperti pada Gambar 3.21 di bawah ini.



Gambar 3.21 *Console AWS Attach to VPC*

Apabila berhasil mengaitkan dengan VPC, maka tampilannya akan seperti Gambar 3.22 di bawah ini.

Gambar 3.22 *Console AWS Dashboard Virtual Private Gateway*

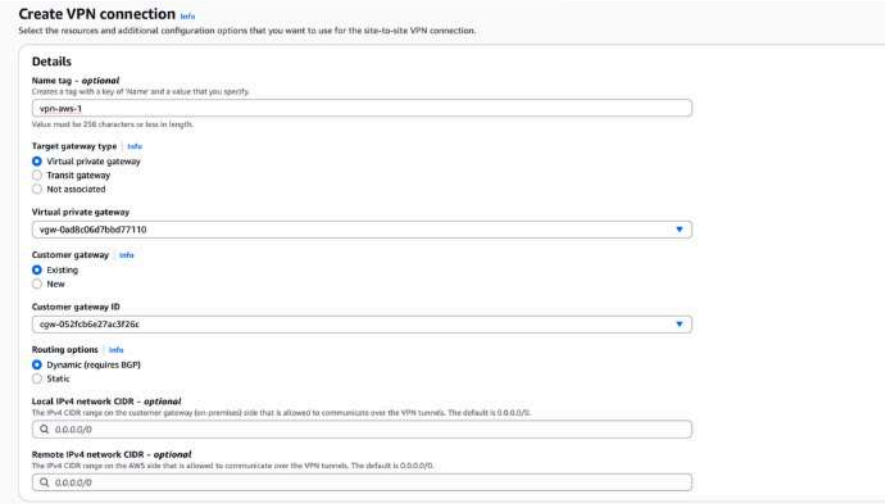
Selanjutnya cari ***Site-to-site VPN***, klik ***Create VPN connection*** seperti pada Gambar 3.23 di bawah ini.

Gambar 3.23 *Console AWS Dashboard VPN Connections*

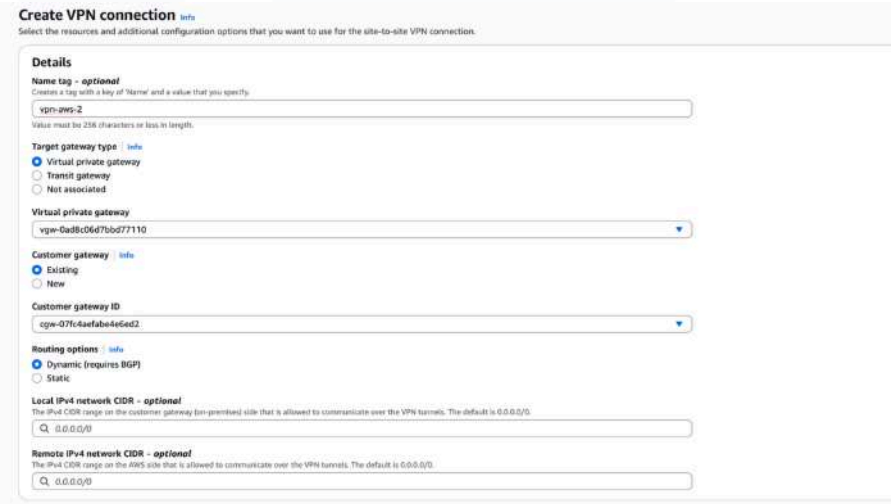
Konfigurasinya adalah sebagai berikut.

- Name : vpn-aws-1
- *Virtual private gateway* : vpg-aws (pilih *virtual private gateway* yang telah dibuat)
- *Customer Gateway* : cgw-aws-1 (pilih *customer gateway* yang telah dibuat)
- *Routing Options* : pilih *Dynamic*
- Klik ***Create***.

Untuk lengkapnya bisa dilihat pada Gambar 3.24 dan Gambar 3.25, dengan catatan tambahan buat 2 *VPN Connection* untuk *customer gateway 1* dan *customer gateway 2.*



Gambar 3.24 *Console AWS Create VPN Connection*



Gambar 3.25 *Console AWS Create VPN Connection*

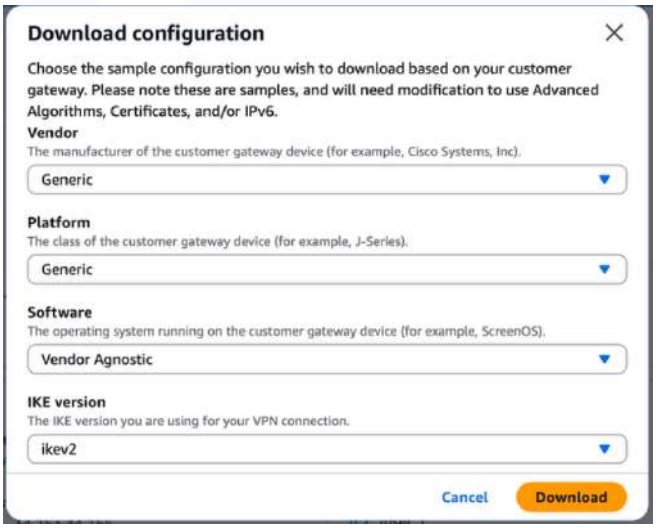
VPN berhasil dibuat, pilih **vpn-aws-1**, dan klik ***download configuration*** seperti pada Gambar 3.26 di bawah ini.

Gambar 3.26 *Console AWS Dashboard VPN Connection*

Konfigurasinya sebagai berikut

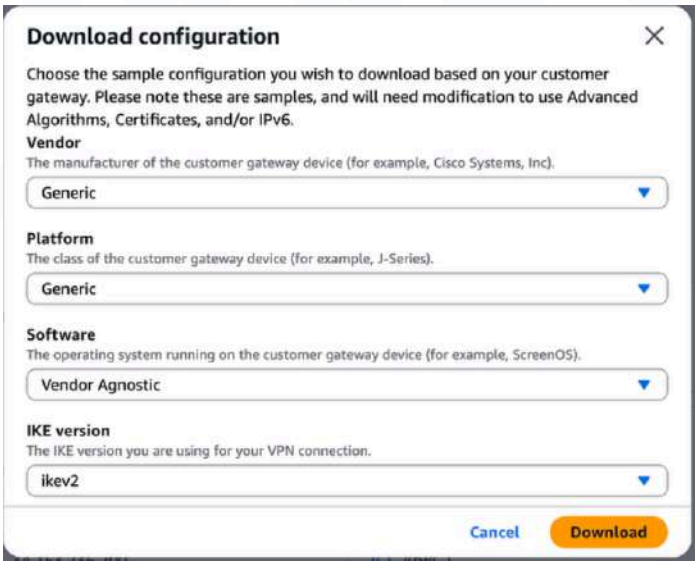
- *Vendor* = ***Generic***
- *IKE version* = ***ikev2***
- klik ***Download***

Ketika berhasil diisi maka tampilannya akan seperti pada Gambar 3.27 di bawah ini.



Gambar 3.27 *Console AWS Download Configuration 1*

Lakukan hal yang sama untuk vpn-aws-2 seperti pada Gambar 3.28 di bawah ini.



Gambar 3.28 *Console AWS Download Configuration 1*

Setelah 2 konfigurasi VPN AWS diunduh, kembali ke halaman GCP, klik ***Peer VPN gateway name***, klik ***Create new peer VPN gateway*** seperti pada Gambar 3.29 di bawah ini.

Gambar 3.29 *Console GCP Add VPN Tunnels*

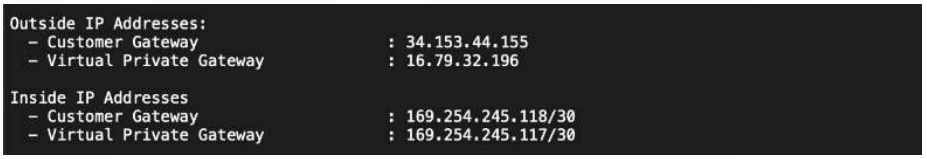
Konfigurasi peer VPN gateway adalah sebagai berikut

- Name : vpg-interface-gcp
- *Interface* : *four interface*, kemudian isi dengan IP yang sudah di *download*

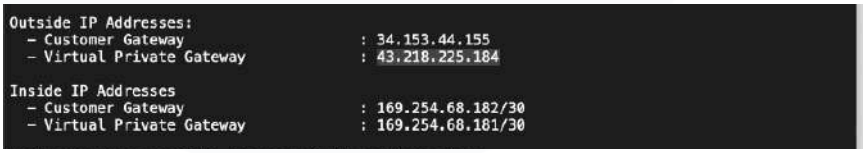
Jika sudah terisi, maka tampilannya akan seperti Gambar 3.30 di bawah ini.

Gambar 3.30 *Console GCP Add Peer VPN Gateway*

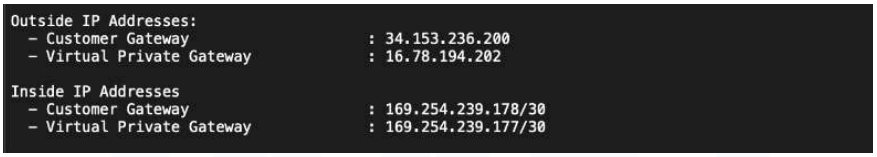
Buka konfigurasi aws yang sudah di unduh, kemudian cari *Outside IP Addresses* dan *Inside IP Addresses* seperti gambar di bawah ini, kemudian copy *Virtual Private Gateway* yang *Outside IP Addresses*. Pastikan urut dari *Customer gateway* 1 dan *Customer gateway* 2 agar lebih mudah dan pastikan urut juga berdasarkan tunnel 1 dan tunnel 2 masing masing *customer gateway*. Kemudian klik *Create*, maka tampilannya akan seperti Gambar 3.31 sampai dengan Gambar 3.34.



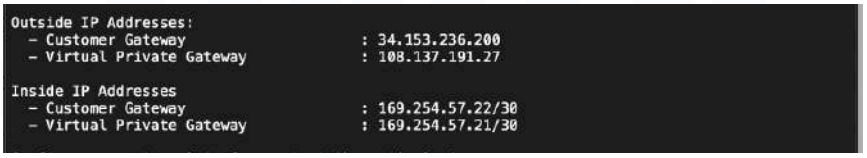
Gambar 3.31 *Configuration Tunnel 1*



Gambar 3.32 *Configuration Tunnel 2*

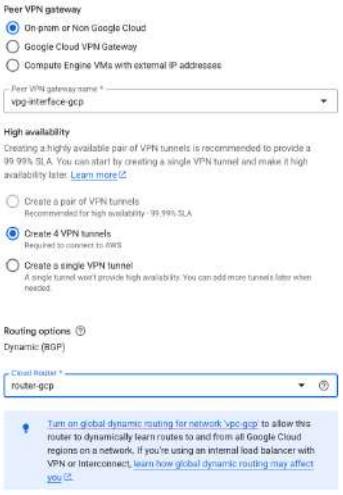


Gambar 3.33 *Configuration Tunnel 3*



Gambar 3.34 *Configuration Tunnel 4*

Setelah berhasil membuat *peer VPN gateway*, pilih ***Cloud Router*** yang sudah dibuat dengan tampilan seperti pada Gambar 3.35 di bawah ini.



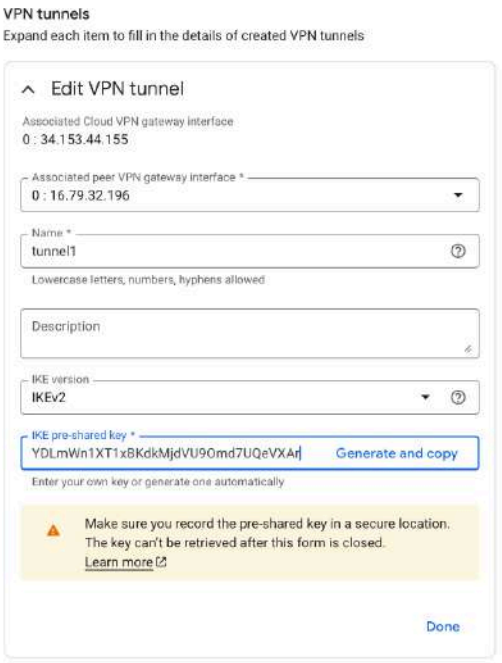
Gambar 3.35 *Console GCP Create Peer VPN Gateway*

Kemudian konfigurasi untuk tunnel seperti dibawah ini seperti berikut:

- Name : tunnel1
- IKE pre-shared key : dapat dari konfigurasi AWS yang di unduh. Pastikan *Pre-shared key* nya sudah sesuai dengan ipnya.

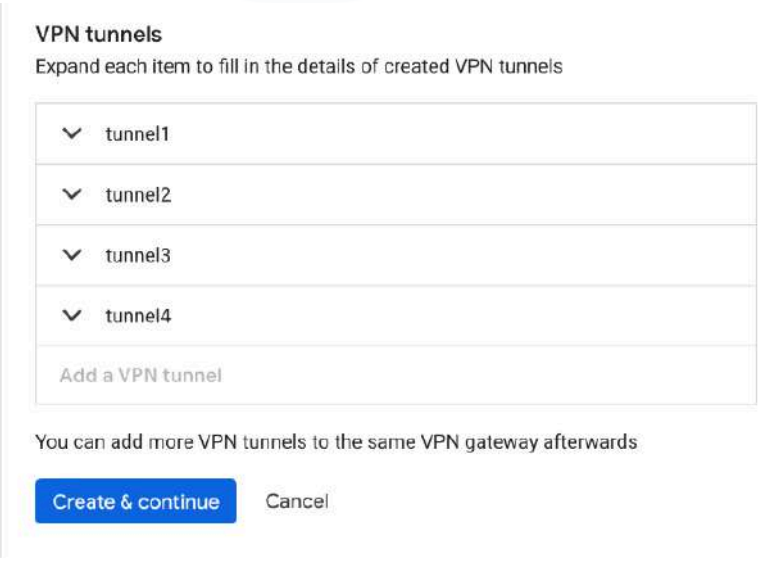
Jika sudah sesuai, maka tampilannya akan seperti Gambar 3.36 dan Gambar 3.37 di bawah ini.

Gambar 3.36 *Configuration Tunnel 1*



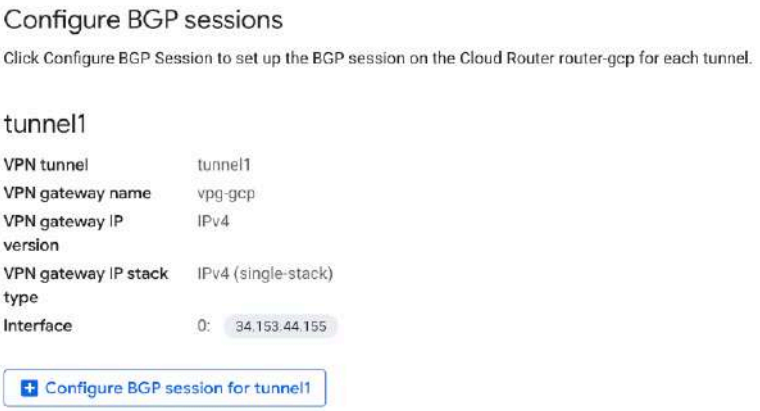
Gambar 3.37 *Console GCP Configuration Tunnel*

Lakukan langkah tersebut untuk semua tunnel, klik ***Create & continue*** seperti yang ada pada Gambar 3.38 di bawah ini.



Gambar 3.38 *Console GCP Create Peer VPN Gateway*

Setelah berhasil membuat VPN tunnel, selanjutnya adalah mengkonfigurasi BGP dengan klik ***Configure BGP session for tunnel1*** seperti pada Gambar 3.39 di bawah ini.

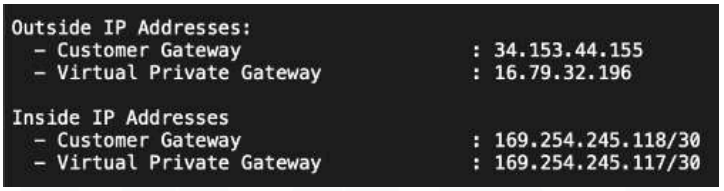


Gambar 3.39 *Console GCP Configuration BGP Sessions*

Konfigurasi adalah sebagai berikut.

- *Name* : bgp1
- *Peer ASN* : 65001 (ASN yang dibuat di AWS)
- *Allocate BPG IPv4 address* : *Manually*
- Kemudian isi IP dengan *Inside IP addresses*

Setelah selesai, maka tampilannya akan seperti pada Gambar 3.40 di bawah ini.

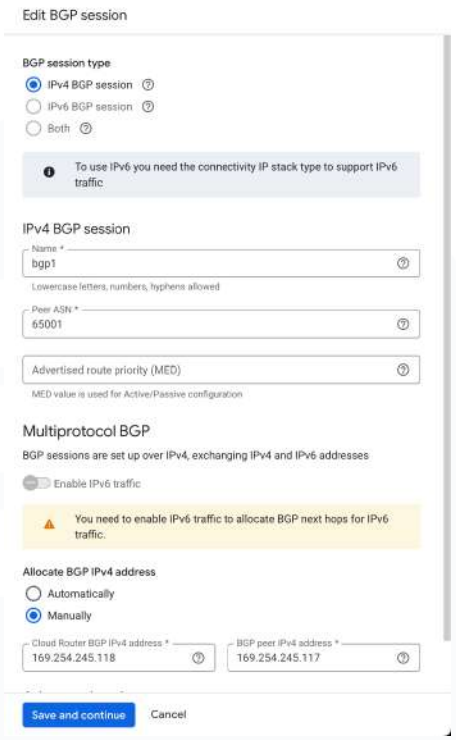


Gambar 3.40 *Configuration Inside IP addresses*

Cloud Router BGP IPv4 address : ***Customer Gateway***

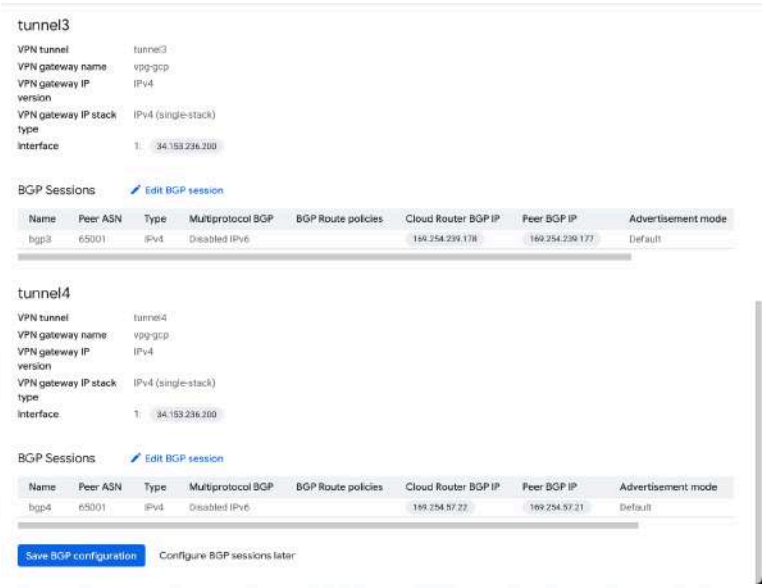
BGP peer IPv4 address : ***Virtual Private Gateway***

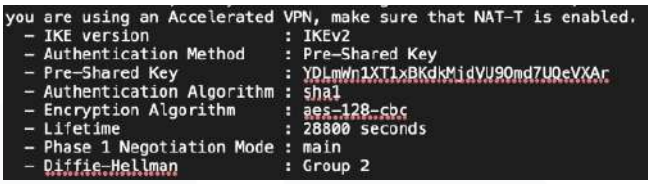
Jika sudah sesuai, maka tampilannya akan seperti Gambar 3.41 di bawah ini.



Gambar 3.41 *Console GCP Configuration BGP Session*

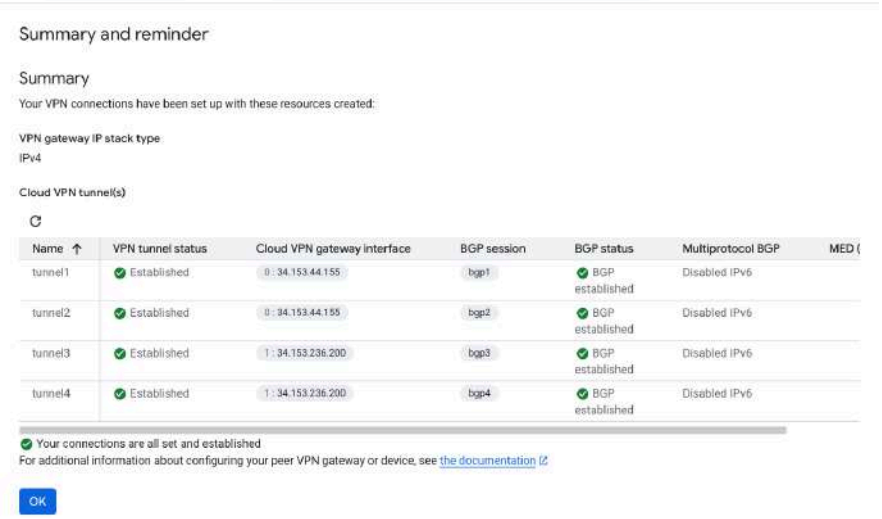
Ulangi langkah diatas untuk *tunnel2, tunnel3, dan tunnel4*. Kemudian klik ***Save BGP configuration*** seperti pada Gambar 3.42 di bawah ini.





Gambar 3.42 *Console GCP Configuration BGP Session*

Setelah berhasil mengkonfigurasi BGP, akan muncul *page* di bawah ini, yang akan menunjukan apakah VPN yang dibuat sudah berjalan dengan baik atau belum. Berdasarkan gambar dibawah ini ***VPN tunnel status Established*** dan ***BGP status BGP established*** yang berarti vpn sudah berjalan dengan baik, namun jika masih bertuliskan *first shake hand* atau yang lain bisa klik tombol *refresh*, jika masih sama kemungkinan besar ada yang salah di konfigurasinya. Jika sudah klik **Ok** seperti yang ada pada Gambar 3.43 di bawah ini.

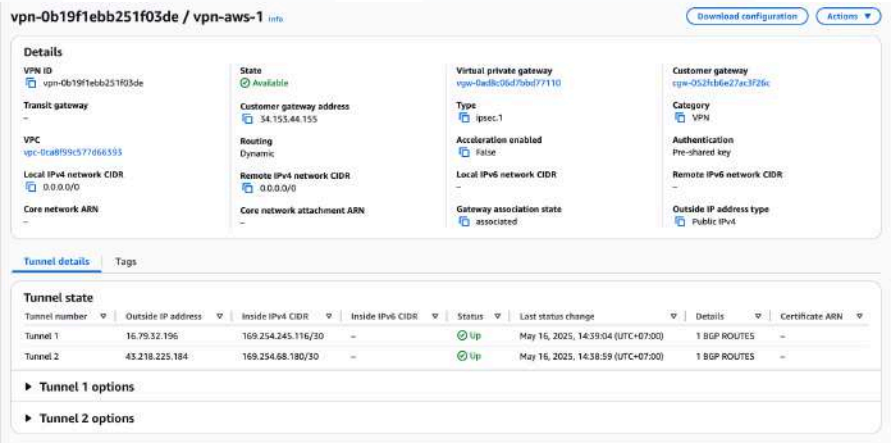


Gambar 3.43 *Console GCP Summary VPN*

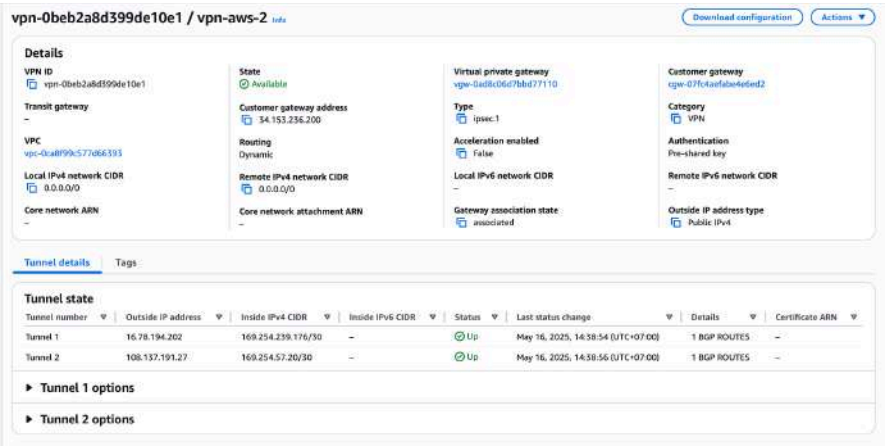
Untuk *crosscheck* apakah VPN nya sudah berjalan dengan baik atau belum kita akan periksa di sisi AWS, dengan klik VPN id dari vpn-aws-1 ataupun vpn-aws-2 seperti pada Gambar 3.44 di bawah ini.

Gambar 3.44 *Console AWS Dashboard VPN Connections*

Kemudian akan muncul page di bawah ini, kemudian periksa pada bagian *Tunnel details* dan status seperti pada Gambar 4.45 dan Gambar 3.46. Jika status *up* berarti VPN berjalan dengan baik jika status masih *down* coba *refresh* dan tunggu beberapa saat, Jika masih *down* berarti kemungkinan besar konfigurasinya ada yang salah.



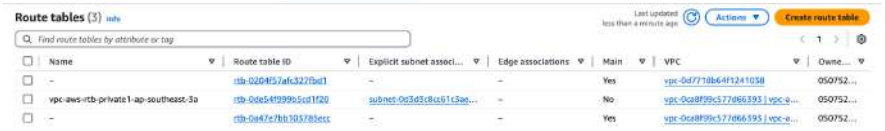
Gambar 3.45 *Console AWS vpn-aws-1 details*



Gambar 3.46 *Console AWS vpn-aws-2 details*

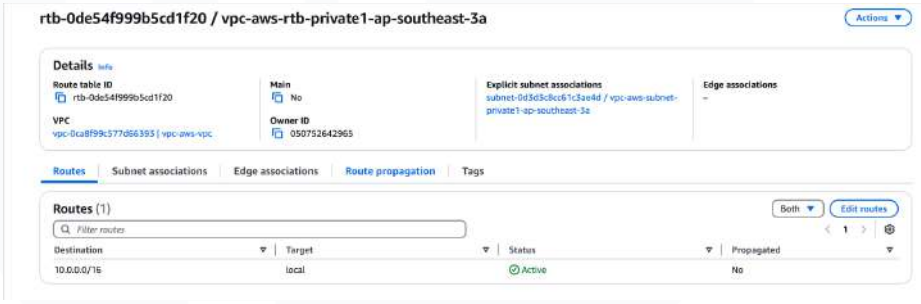
Setelah VPN sudah ***Established*** dan ***Up***, selanjutnya kita akan mengkoneksikan antara *subnet GCP* dengan *subnet AWS* dengan cara

buka ***Route tables***, kemudian klik ***route table id*** yang digunakan seperti pada Gambar 3.47.



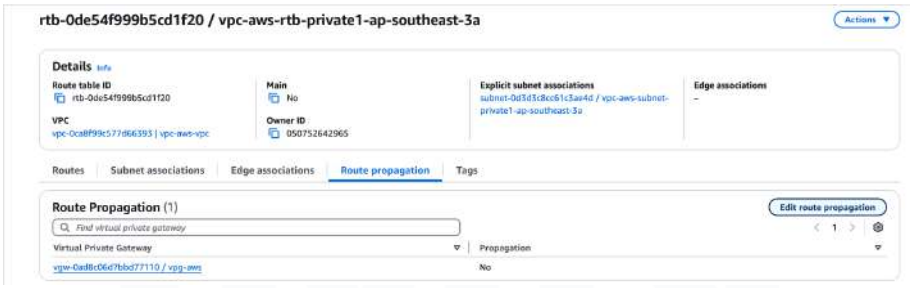
Gambar 3.47 *Console AWS Dashboard Route Table*

Kemudian akan muncul halaman seperti Gambar 3.48 di bawah ini, klik ***Route propagation***.



Gambar 3.48 *Console AWS Route Table Details*

Kemudian klik ***Edit route propagation*** seperti pada Gambar 3.49 di bawah ini.



Gambar 3.49 *Console AWS Route Table Details*

Kemudian ***enable propagation*** dan klik ***save*** seperti pada Gambar 3.50 di bawah ini.

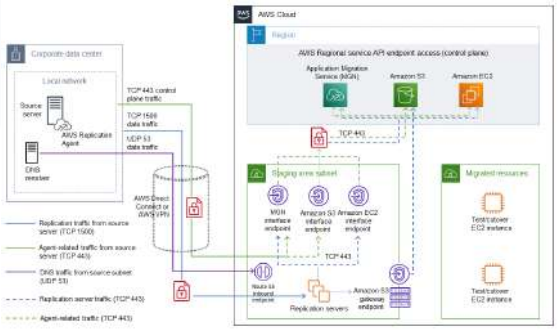
Gambar 3.50 *Console AWS Edit Route Propagation*

Setelah berhasil mengaktifkan ***Route propagation***, maka pada *routes* secara otomatis akan muncul CIDR subnet dari GCP seperti pada Gambar 3.51 di bawah ini.

Gambar 3.51 *Console AWS Dashboard Route Table*

b. Tahap Kedua: Migrasi VM

Setelah berhasil membangun jaringan *Site-to-Site VPN*, proses dilanjutkan ke tahap migrasi VM. Migrasi dilakukan dengan cara replikasi menggunakan layanan AWS *Application Migration Service* (MGN) melalui jalur VPN. Setelah VM berhasil dipindahkan, aplikasi disusun ulang dan dijalankan kembali di lingkungan AWS, misalnya menggunakan Amazon EC2. Tahap ini diakhiri dengan pengujian fungsionalitas, serta *cut over* layanan termasuk pengalihan DNS apabila seluruh sistem telah dipastikan berjalan normal di AWS.



Gambar 3.52 *Arsitektur Migrasi*

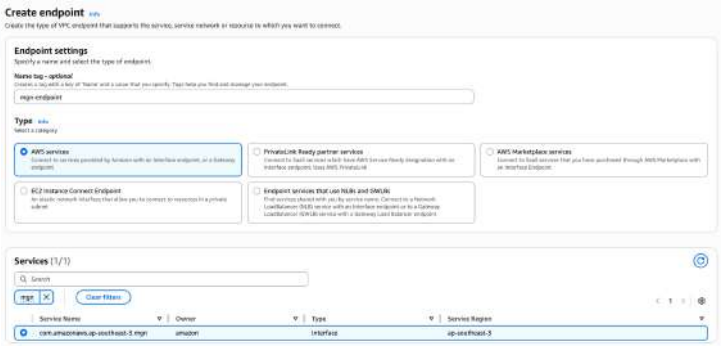
Gambar 3.52 merupakan gambar arsitektur migrasi dengan menggunakan servis AWS *Application Migration Service* (MGN)

melalui jaringan privat untuk menghindari kebocoran data. dibawah ini merupakan langkah langkah untuk mengkonfigurasi AWS *Application Migration Service* (MGN).

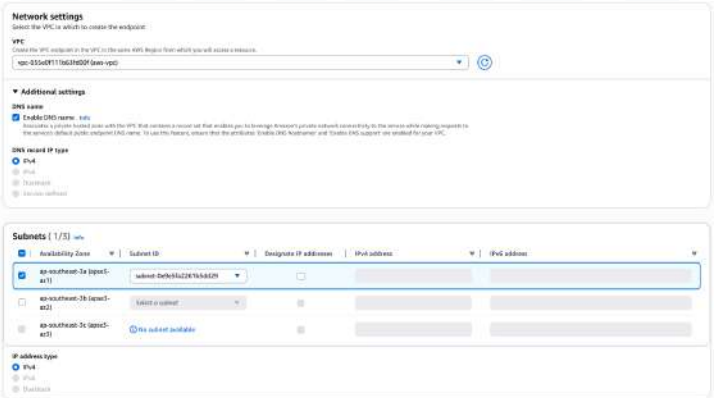
Membuat MGN Endpoint

- *Name tag* : **mgn-endpoint**
- *Type* : **AWS *services***
- *Service* : **com.amazonaws.ap-southeast-3.com.mgn**
- *Enable* DNS *name* : ***Checklist***
- *Subnets* : **Sesuaikan dengan *subnet* yang akan digunakan**
- *Security Group* : **Gunakan yang sudah ada atau buat dengan port 443 dan 1500**
- Klik ***Create Endpoint***

Untuk lengkapnya bisa dilihat pada Gambar 3.53 dan Gambar 3.54 seperti yang ada di bawah ini.



Gambar 3.53 *Console AWS Create Endpoint*

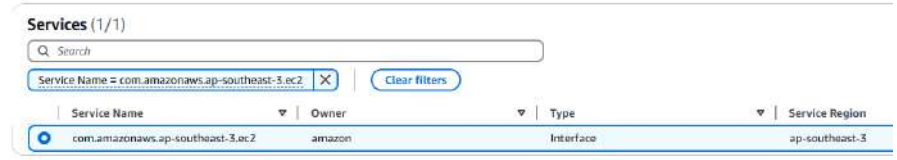


Gambar 3.54 *Console AWS Create MGN Endpoint*

Gambar 3.55 *Console AWS Create Endpoint*

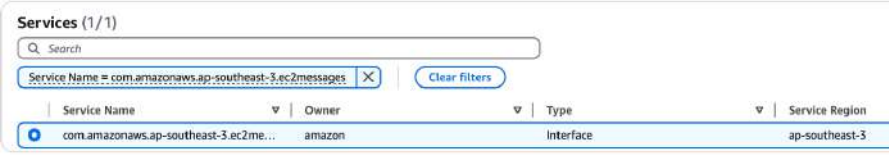
Ulangi langkah diatas untuk membuat *endpoint* dari EC2, EC2 *Message*, SSM, SSM *Message*, S3 *Gateway*, dan S3 *Interface* seperti Gambar 3.55 di atas.

Untuk EC2 *Endpoint,* bisa dilihat pada Gambar 3.56 di bawah ini.



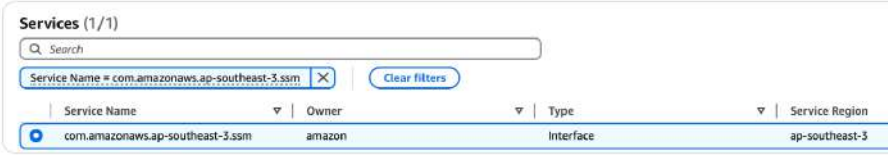
Gambar 3.56 *Console AWS Create EC2 Endpoint*

Sedangkan untuk EC2 *Message Endpoint,* bisa dilihat pada Gambar 3.57 di bawah ini.



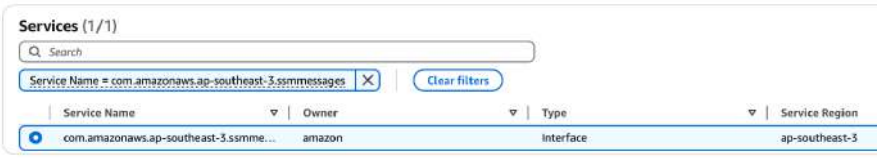
Gambar 3.57 *Console AWS Create EC2 Message Endpoint*

Untuk SSM *Endpoint*, bisa dilihat pada Gambar 3.58 di bawah ini.



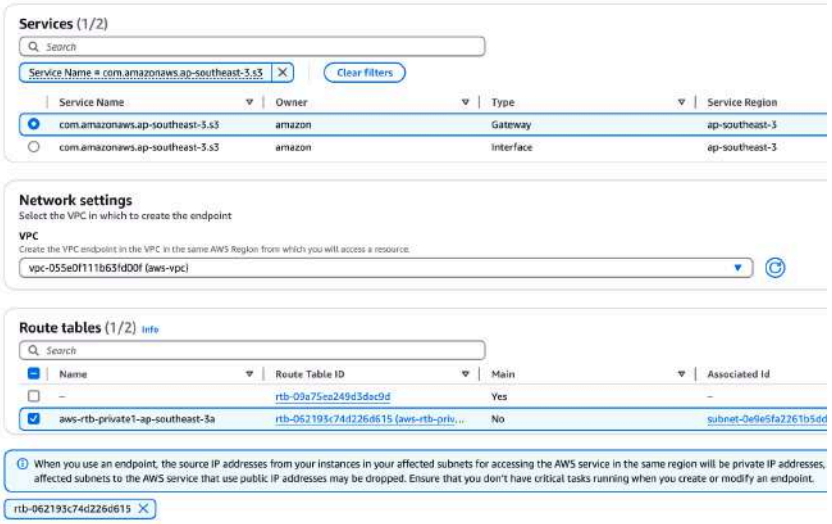
Gambar 3.58 *Console AWS Create SSM Endpoint*

Sedangkan untuk SSM *Message Endpoint*, bisa dilihat pada Gambar 3.59 di bawah ini.



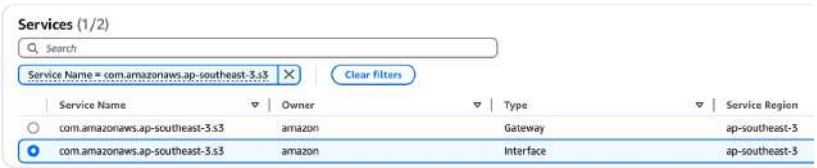
Gambar 3.59 *Console AWS Create SSM Message Endpoint*

Selanjutnya pada S3 *Gateway*, terdapat sedikit perbedaan karena *endpoint* ini akan dikaitkan dengan *routing table* seperti pada Gambar 3.60 di bawah ini.



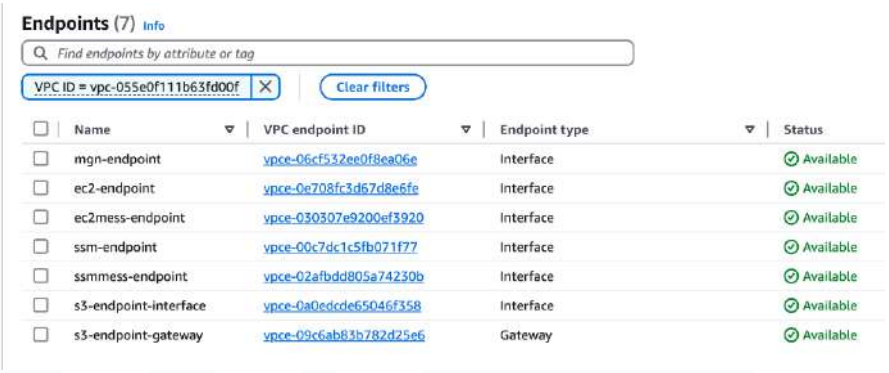
Gambar 3.60 *Console AWS Create S3 Gateway Endpoint*

Untuk S3 *Endpoint* (*Interface*), bisa dilihat pada Gambar 3.61 di bawah ini.



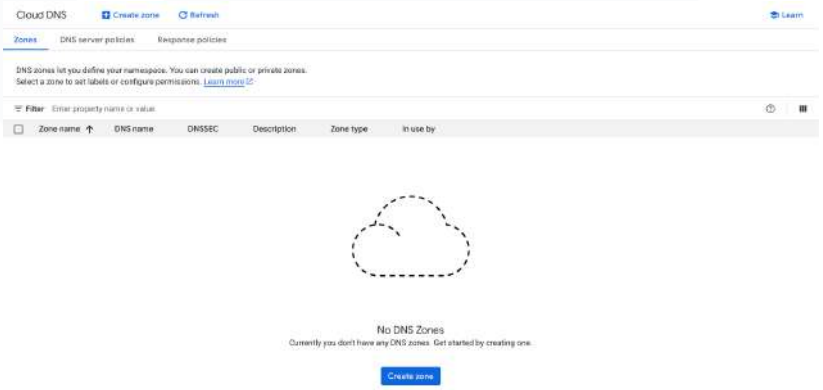
Gambar 3.61 *Console AWS Create S3 Interface Endpoint*

Tampilan *Endpoint* yang berhasil dibuat akan seperti Gambar 3.62 di bawah ini.



Gambar 3.62 *Console AWS Dashboard Endpoint*

Selanjutnya buka **GCP**, Cari ***Cloud DNS***, klik ***Create zone*** seperti pada Gambar 3.63 di bawah ini.

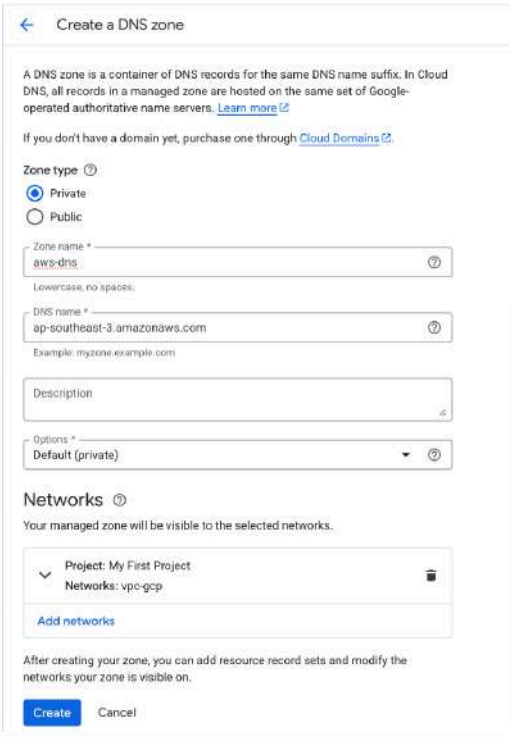


Gambar 3.63 *Console GCP Dashboard Cloud DNS*

Setelah itu akan muncul form untuk membuat *DNS zone*, isi form tersebut seperti berikut.

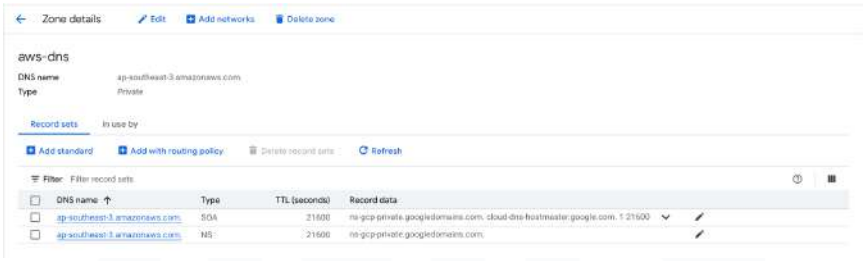
- *Zone type : Private*
- *DNS name* : ap-southeast-3.amazonaws.com
- *Networks* : Pilih VPC yang akan digunakan
- klik ***Create***.

Jika sudah, maka tampilannya akan sama seperti Gambar 3.64 di bawah ini.



Gambar 3.64 *Console GCP Create DNS Zone*

Setelah *DNS zone* berhasil dibuat akan muncul tampilan seperti dibawah ini, selanjutnya buat *record* baru dengan klik ***Add standard*** seperti pada Gambar 3.65 di bawah ini**.**



Gambar 3.65 *Console GCP DNS Zone Details*

Selanjutnya muncul tampilan form *Create record set*, isi form tersebut sebagai berikut.

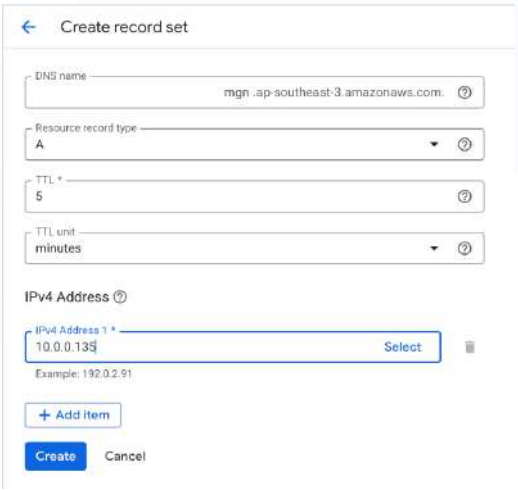
- DNS *name* : mgn
- *Resource record type* : A
- TTL : 5

- *IPv4 Address* : 10.0.0.135 (Sesuaikan dengan IP *endpoint* yang sudah dibuat)

Jika sudah, maka tampilannya akan seperti Gambar 3.66 di bawah ini.

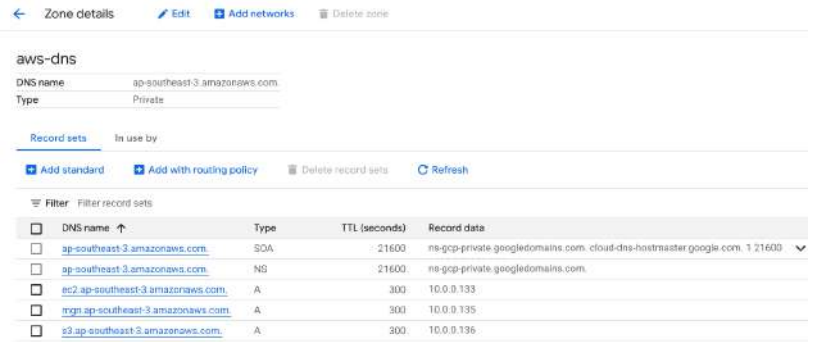
Gambar 3.66 *Console AWS MGN Endpoint Details*

Kemudian Klik ***Create*** seperti pada Gambar 3.67 di bawah ini.



Gambar 3.67 *Console GCP Create Record*

Kemudian ulangi langkah tersebut untuk *endpoint* ec2 dan s3 seperti Gambar 3.68 di bawah ini.

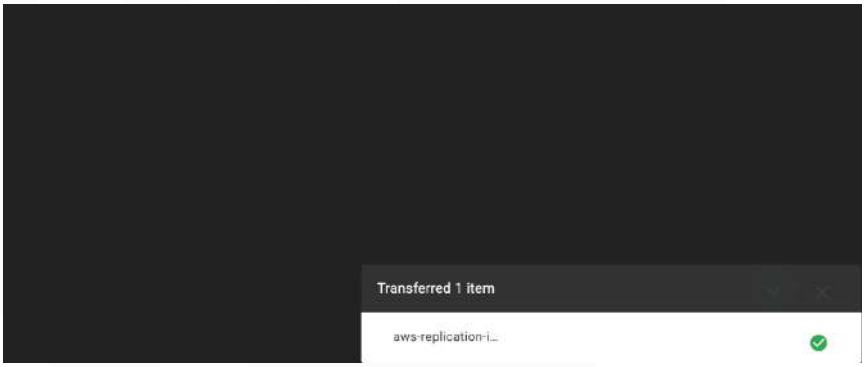


Gambar 3.68 *Console GCP Dashboard DNS Zone*

Install Agent di VM *Source*

Masuk ke VM dengan menggunakan ssh, Kemudian *Upload File* atau *Download File* dengan *command* dibawah ini dan pasang nat *gateway* untuk melakukan *download agent* dan beberapa *package* yang diperlukan sampai muncul *Transferred item* seperti pada Gambar 3.69 di bawah.

```
sudo wget -O ./aws-replication-installer-init
https://aws-application-migration-service-ap-southeast-3.s3.ap-southeast
-3.amazonaws.com/latest/linux/aws-replication-installer-init
```



Gambar 3.69 *Terminal Upload File*

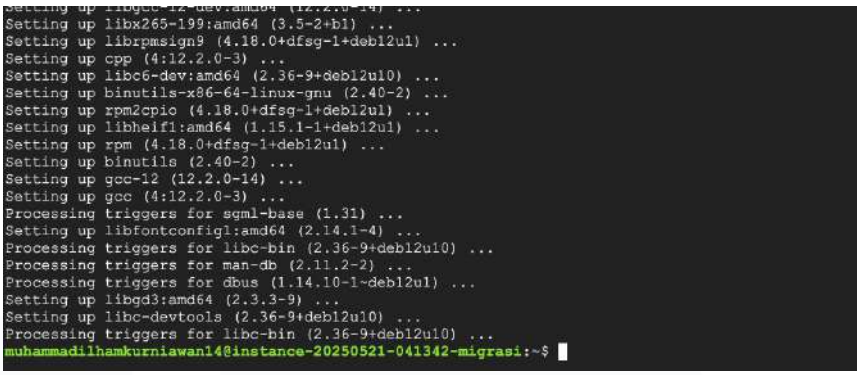
Setelah selesai *upload file*, *install* beberapa package dibawah ini sampai berhasil seperti pada Gambar 3.70.

amazon linux/centos/redhat:

```
sudo yum install make gcc perl tar gawk rpm -y
```

ubuntu/debian:

```
sudo apt install make gcc perl tar gawk rpm -y
```



Gambar 3.70 *Terminal Installation Package*

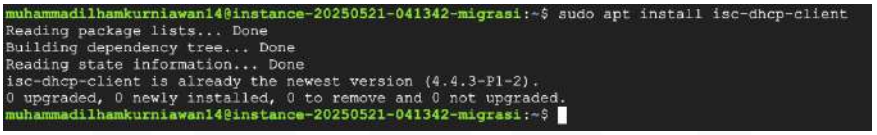
Setelah berhasil melakukan *install* make, gcc, perl, tar, gawk, dan rpm selanjutnya *install package* dhcp *client* di bawah ini sampai pada terminal muncul indikator seperti pada Gambar 3.71.

On Amazon linux/Centos/Redhat:

```
sudo yum install dhclient atau sudo yum install dhcp-client
```

On Ubuntu/Debian:

```
sudo apt install isc-dhcp-client
```



Gambar 3.71 *Terminal Install Package*

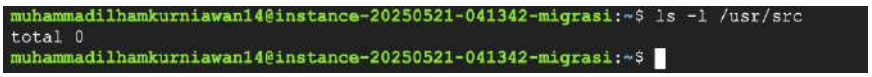
Setelah berhasil melakukan *install* beberapa *package* di atas, sekarang cek *kernel-devel/linux-headers* dengan menjalankan *command* dibawah ini.

On RHEL/CENTOS/Oracle:

```
ls -l /usr/src/kernels
```

On Debian/Ubuntu/SUSE:

```
ls -l /usr/src
```



Gambar 3.72 *Terminal Check Kernel*

Jika hasilnya menunjukan total 0 seperti pada Gambar 3.72, maka *install kernel-devel/linux-headers*, dengan menjalankan *command* dibawah ini.

On RHEL/CENTOS/Oracle:

```
sudo yum install kernel-devel-`uname -r`
```

On Oracle with Unbreakable Enterprise Kernel:

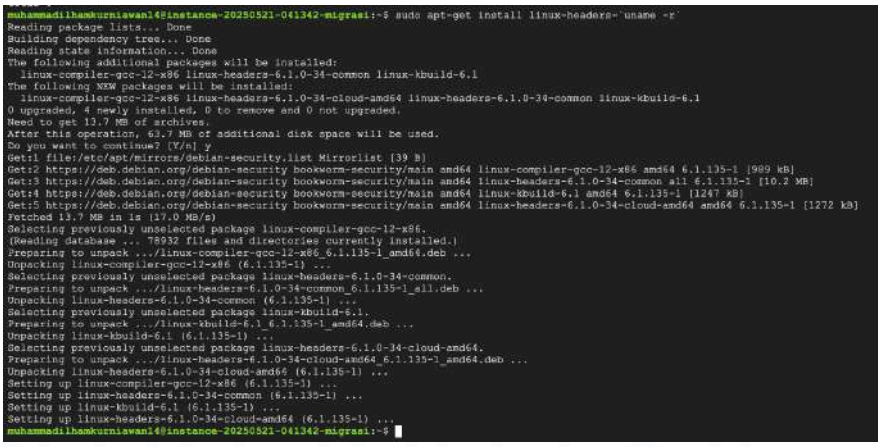
```
sudo yum install kernel-uek-devel-`uname -r`
```

On Debian/Ubuntu:

```
sudo apt-get install linux-headers-`uname -r`
```

On SUSE:

```
sudo zypper install kernel-default-devel-`uname -r`
```



Gambar 3.73 *Terminal Installation Kernel*

Setelah berhasil melakukan *install kernel-devel/linux-headers* seperti pada Gambar 3.73, coba cek lagi dengan menjalankan *command* dibawah ini.

On RHEL/CENTOS/Oracle:

```
ls -l /usr/src/kernels
```

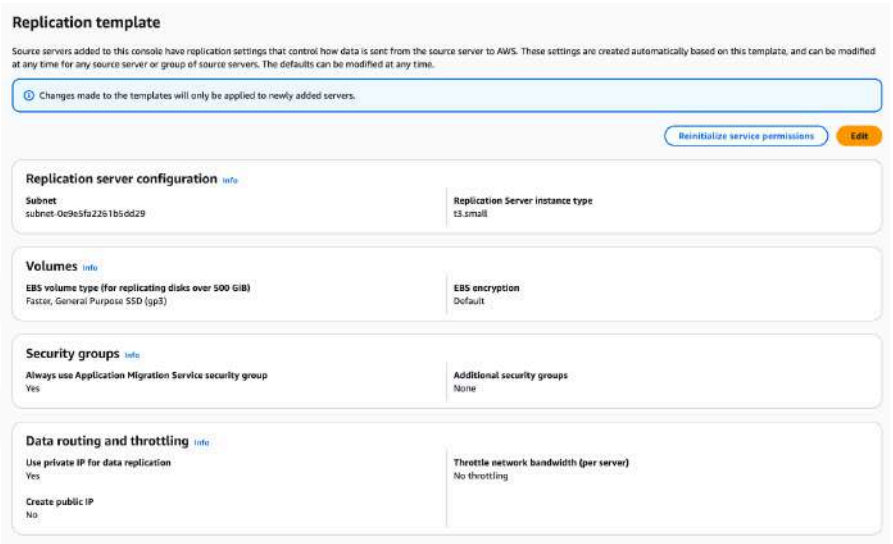
On Debian/Ubuntu/SUSE:

```
ls -l /usr/src
```

```
muhammadilhamkurniawan14@instance-20250521-041342-migrasi:~$ ls -l /usr/src
total 8
drwxr-xr-x 4 root root 4096 May 21 04:25 linux-headers-6.1.0-34-cloud-amd64
drwxr-xr-x 4 root root 4096 May 21 04:25 linux-headers-6.1.0-34-common
lrwxrwxrwx 1 root root   23 Apr 25 19:51 linux-kbuild-6.1 -> ../lib/linux-kbuild-6.1
muhammadilhamkurniawan14@instance-20250521-041342-migrasi:~$
```

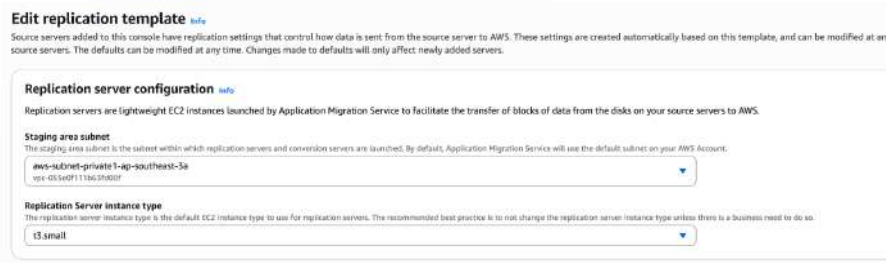
Gambar 3.74 *Terminal Check Kernel*

Berdasarkan hasilnya sudah muncul *kernel-devel/linux-headers* dengan total menunjukan angka 8 seperti pada Gambar 3.74. Sebelum melanjutkan *install* mgn agent, buka MGN (*Application Migration Service*), kemudian buka *replication template*, kemudian klik *edit* seperti pada Gambar 3.75 di bawah ini.



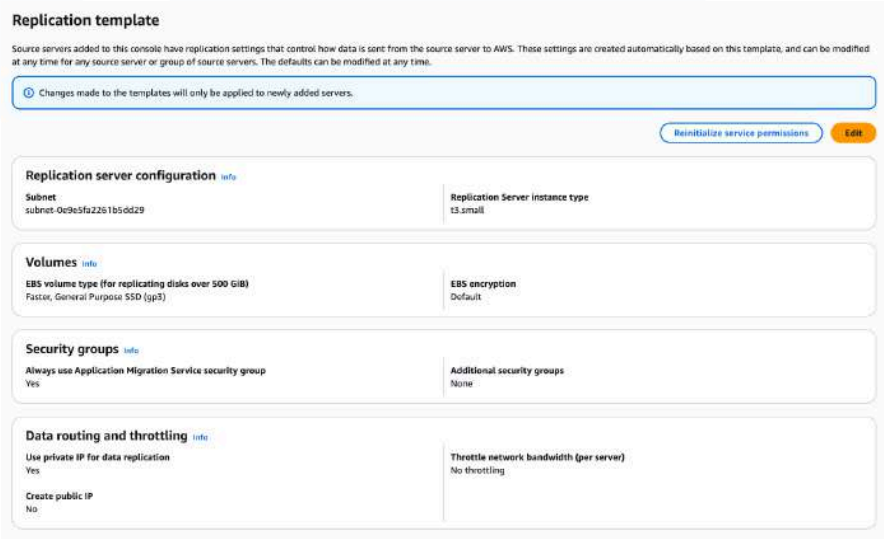
Gambar 3.75 *Console AWS Replication Template*

Selanjutnya pastikan subnet yang digunakan untuk replikasi (*staging area subnet*) sudah sesuai, kemudian klik ***Save Template*** seperti pada Gambar 3.76 di bawah ini**.**



Gambar 3.76 *Console AWS Edit Replication Template*

Untuk memastikan sudah terpasang klik ***Reinitialize service permission*** seperti pada Gambar 3.77 di bawah ini.



Gambar 3.77 *Console AWS Replication Template*

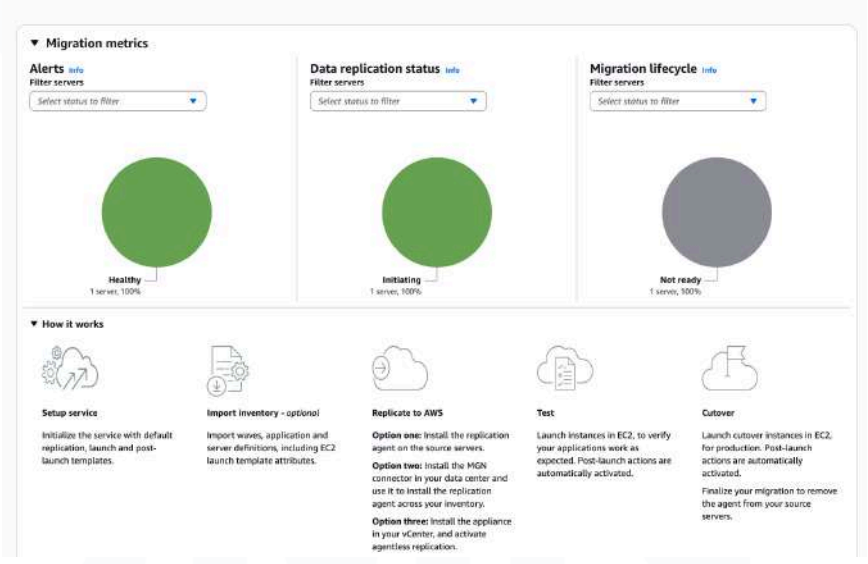
Selanjutnya *install* mgn *agent* dengan menjalankan *command* di bawah ini dengan menonaktifkan nat *gateway* hingga indikatornya sama dengan Gambar 3.78 di bawah.

```
sudo chmod +x aws-replication-installer-init; sudo ./aws-replication-installer-init
--region <region> --aws-access-key-id <MGN_IAM_ACCESS_KEY>
--aws-secret-access-key <MGN_IAM_SECRET> --no-prompt --endpoint
vpce-<VPC-id>-<suffix>.mgn.<region>.vpce.amazonaws.com --s3-endpoint
vpce-<VPC-ID>-<suffix>.s3.<region>.vpce.amazonaws.com
```

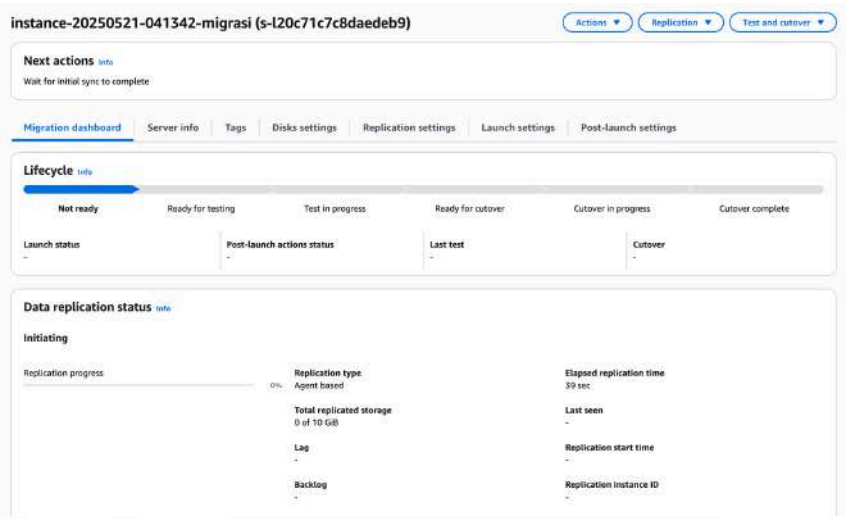
Gambar 3.78 *Terminal Installation Agent MGN*

Replikasi Server

Selanjutnya buka kembali MGN (*Application Migration Service*) di AWS, Kemudian tunggu hingga muncul *source server*. Kemudian klik *source* servernya seperti pada langkah Gambar 3.79 sampai dengan 3.81 di bawah ini.



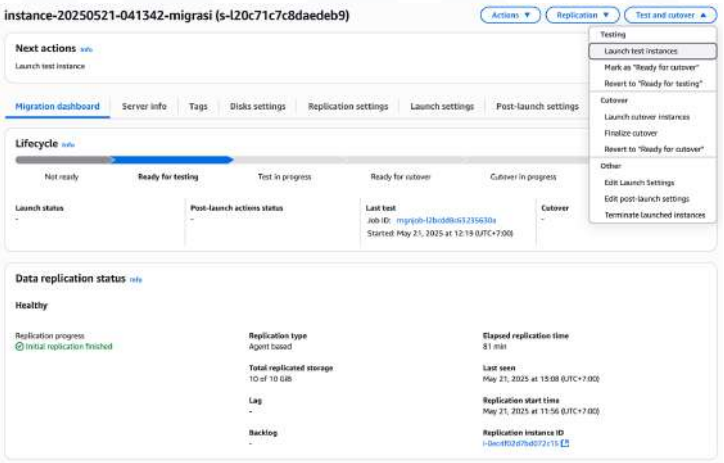
Gambar 3.79 *Console AWS Dashboard MGN*



Gambar 3.80 *Console AWS Dashboard MGN Details*

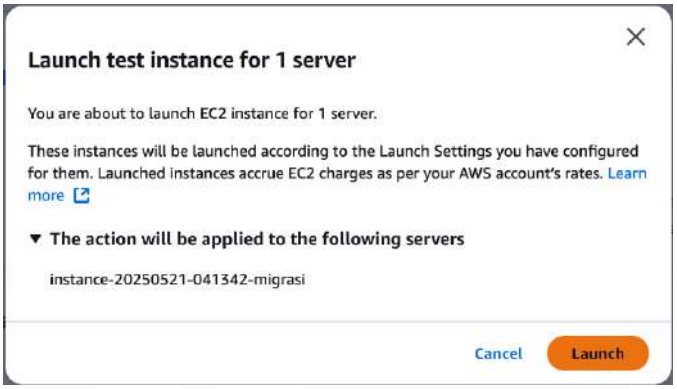
Gambar 3.81 *Console AWS Dashboard MGN Details*

Replikasi server sudah berhasil, selanjutnya lanjut ke tahap testing dengan klik ***Test and cutover***, kemudian klik ***Launch test instances*** seperti pada Gambar 3.82 **.**



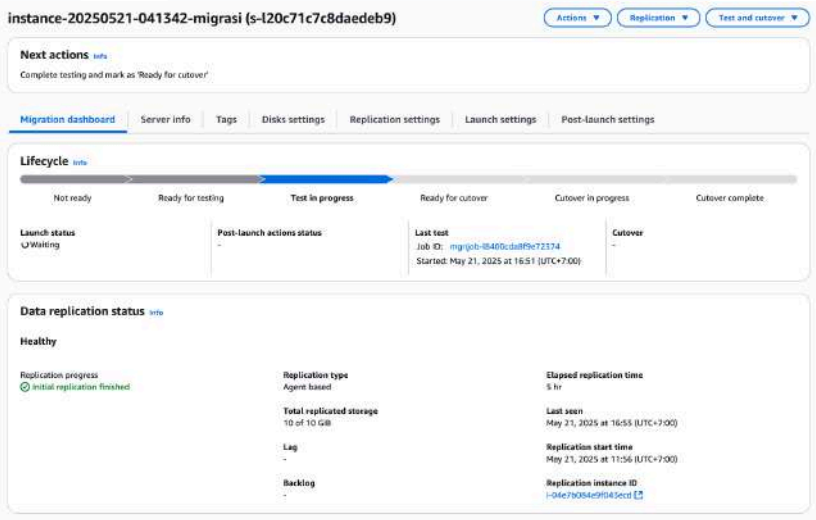
Gambar 3.82 *Console AWS Dashboard MGN Details*

Kemudian klik ***Launch*** seperti pada Gambar 3.83 di bawah ini.



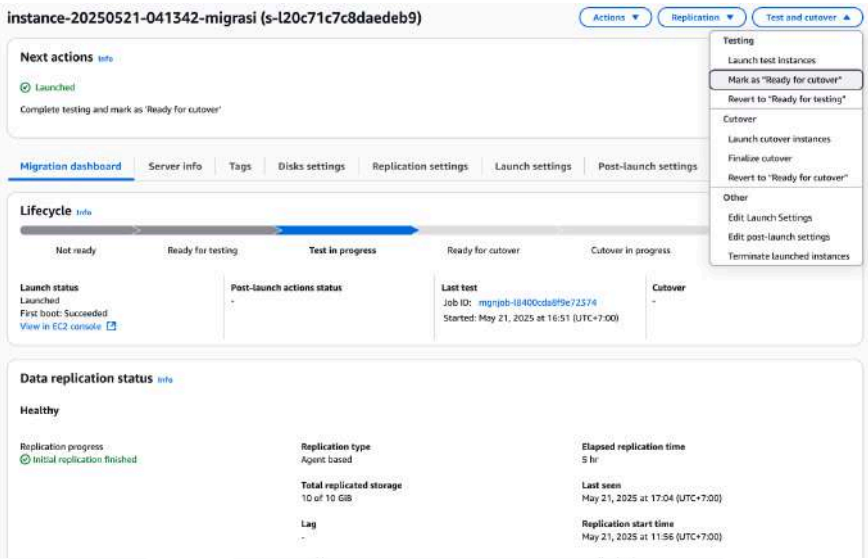
Gambar 3.83 *Console AWS Modal Approval Launch Instance*

Mgn akan menjalankan *testing* untuk *instance* yang akan di migrasi sesuai pada Gambar 3.84 di bawah.



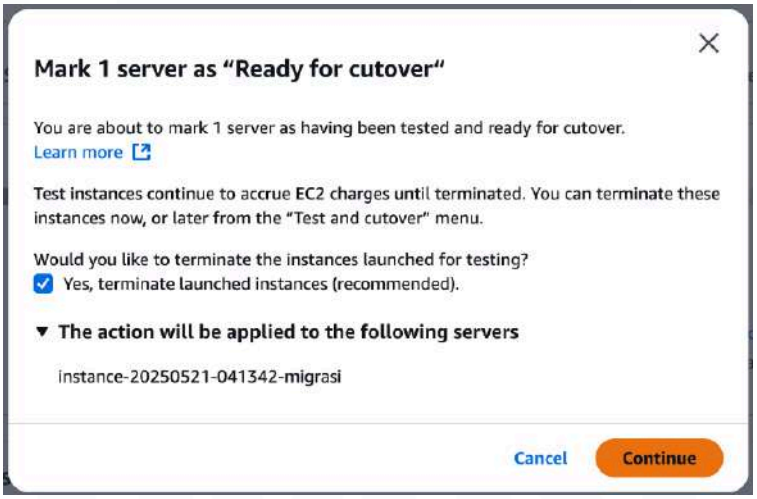
Gambar 3.84 *Console AWS Dashboard MGN Details*

Setelah *testing* selesai dan berjalan dengan baik bisa dilanjutkan dengan ***Mark as "Ready for cutover"*** seperti pada Gambar 3.85 di bawah ini



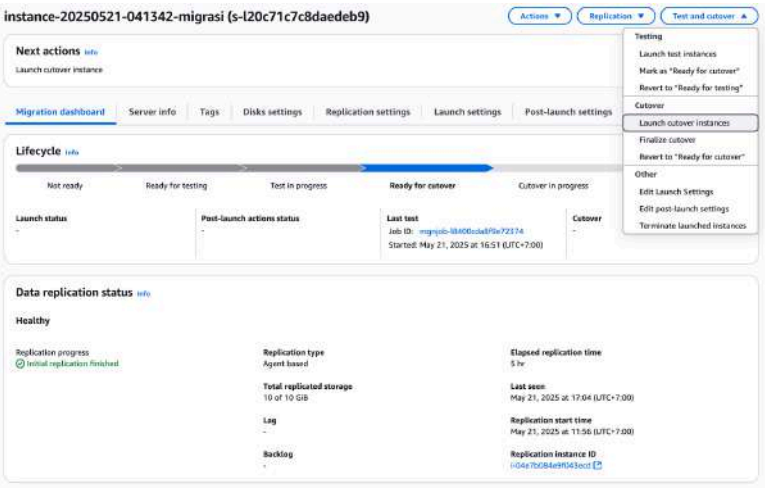
Gambar 3.85 *Console AWS Dashboard MGN Details*

Kemudian klik ***Continue*** seperti pada Gambar 3.86 di bawah ini.



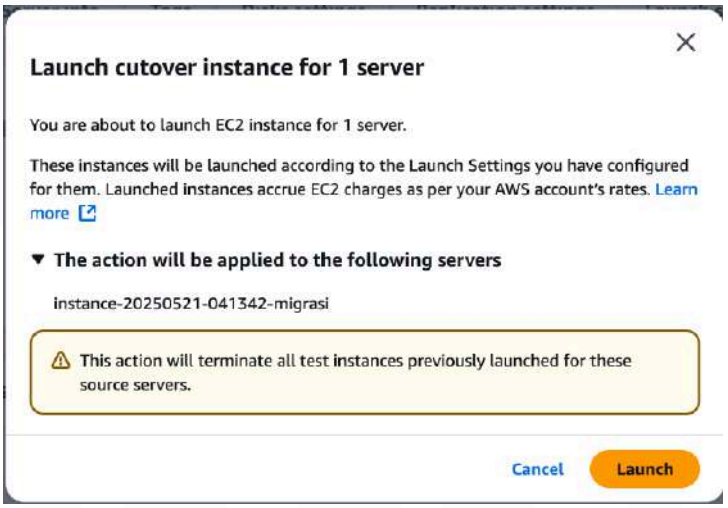
Gambar 3.86 *Console AWS Modal Approval Mark as Ready for Cutover*

Setelah *Mark as "Ready for cutover"* selesai, tahap selanjutnya adalah klik ***Launch cutover instance*** seperti pada Gambar 3.87 di bawah ini.



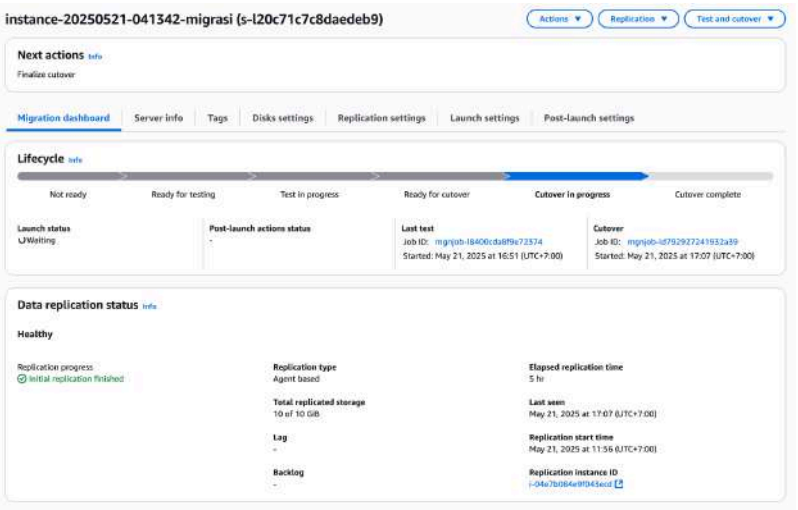
Gambar 3.87 *Console AWS Dashboard MGN Details*

Kemudian klik ***Launch*** seperti pada Gambar 3.88 di bawah ini.



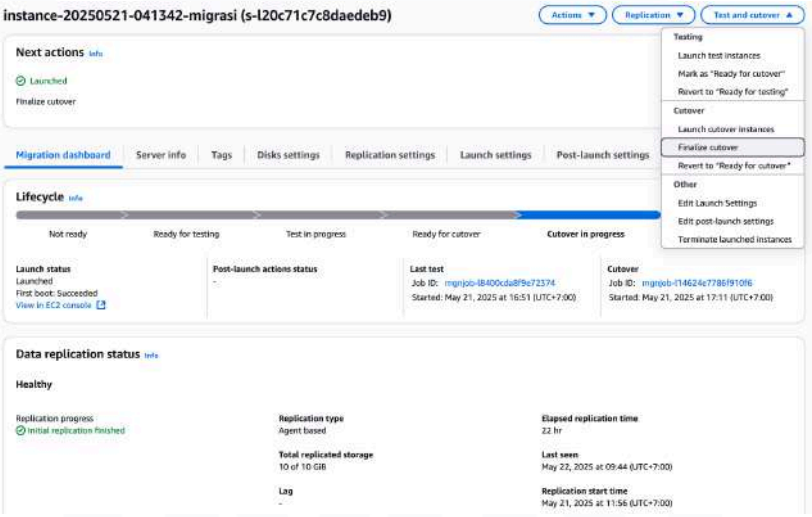
Gambar 3.88 *Console AWS Modal Approval Launch Instance*

Mgn akan membuat *cut over instance* yang akan menjadi *instance* utama atau *instance* yang sudah berhasil di migrasi seperti pada Gambar 3.89 di bawah ini.



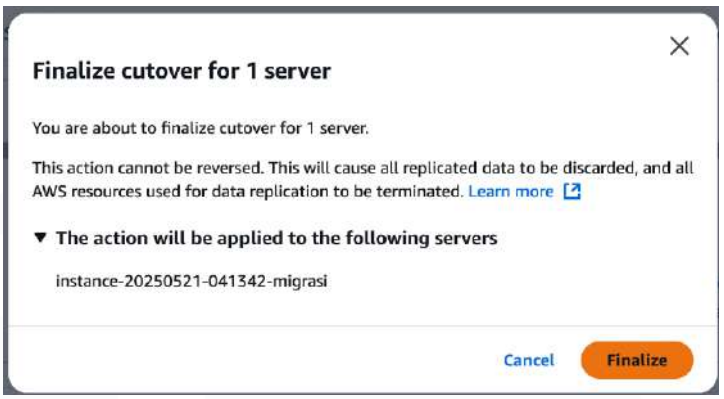
Gambar 3.89 *Console AWS Dashboard MGN Details*

Setelah berhasil melakukan ***Launch Cutover Instance,*** selanjutnya klik ***Finalize cutover*** seperti pada Gambar 3.90 di bawah ini**.**



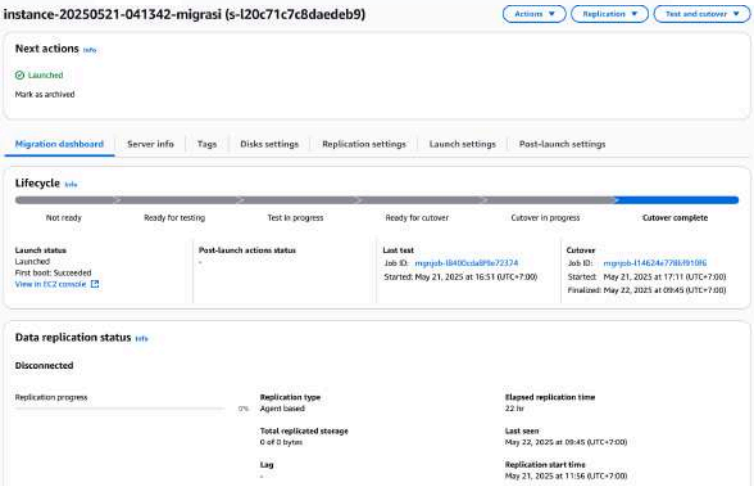
Gambar 3.90 *Console AWS Dashboard MGN Details*

Kemudian klik ***Finalize*** seperti pada Gambar 3.91 di bawah ini.



Gambar 3.91 *Console AWS Modal Approval Finalize Cutover*

Tampilan jika sudah berhasil melakukan semua tahap yang ada di MGNakan seperti Gambar 3.92 di bawah.

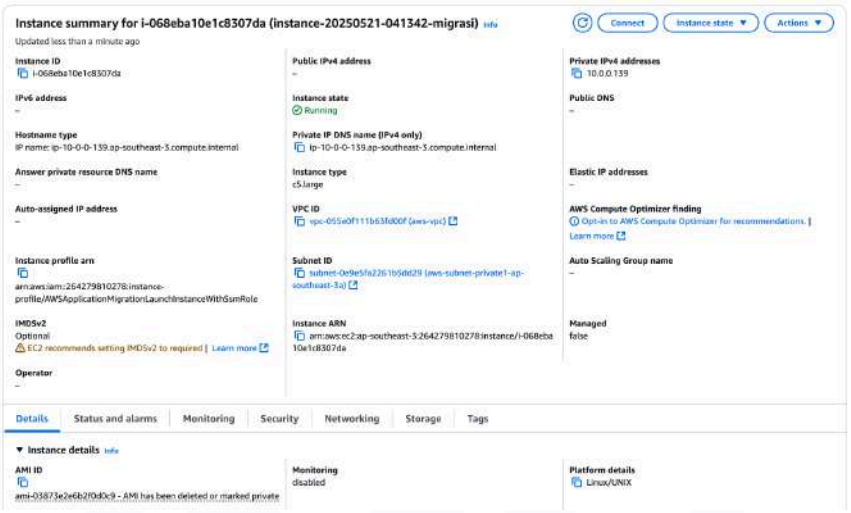


Gambar 3.92 *Console AWS Dashboard MGN Details*

Selanjutnya kita akan *check instance* yang sudah berhasil di migrasi, dengan membuka ec2, kemudian cari *instance* nya kemudian klik *instance* ID seperti Gambar 3.93 di bawah ini.

Gambar 3.93 *Console AWS Dashboard Instances*

Selanjutnya akan masuk kedalam *instance summary*, kemudian klik ***Connect*** seperti pada Gambar 3.94 di bawah.



Gambar 3.94 *Console AWS Instance Details*

Kemudian pilih *session manager*, kemudian klik ***connect*** seperti pada Gambar 3.95 di bawah.

Gambar 3.95 *Console AWS Connect to Instance*

Setelah berhasil masuk ke terminal melalui *session manager*, dapat di cek semua data yang sudah di migrasi apakah sudah sesuai atau belum.

c. Kesimpulan

Migrasi infrastruktur dari *Google Cloud Platform* (GCP) ke *Amazon Web Services* (AWS) dengan memanfaatkan koneksi VPN melalui jaringan privat telah berhasil dilakukan secara aman dan terkendali. Pendekatan ini memungkinkan konektivitas langsung antara VPC di GCP dan VPC di AWS tanpa harus mengekspos layanan ke internet publik, sehingga meningkatkan keamanan komunikasi antar lingkungan *cloud*. Implementasi VPN *site-to-site* menggunakan *IPsec tunnel* mampu menyediakan jalur terenkripsi untuk pertukaran data, yang sangat penting dalam menjaga integritas dan kerahasiaan informasi selama proses migrasi berlangsung. Meskipun VPN *site-to-site* memiliki latensi yang sedikit lebih tinggi dibandingkan Direct Connect, pendekatan ini dinilai lebih fleksibel dan cepat dalam hal implementasi, serta lebih ekonomis untuk kebutuhan migrasi skala menengah. Selama proses migrasi, routing dinamis melalui BGP membantu dalam menjaga ketersediaan koneksi dan meminimalkan gangguan layanan. Selain itu, pengujian berlapis dan pengawasan terhadap throughput, latensi, serta kestabilan koneksi VPN menjadi faktor penting dalam memastikan kelancaran perpindahan data dan layanan. Dengan selesainya migrasi ini, klien akan mendapatkan infrastruktur AWS yang lebih modern, skalabel, dan terintegrasi, sekaligus mempertahankan standar keamanan tinggi melalui koneksi privat antar cloud. Langkah ini merupakan bagian penting dalam strategi transformasi digital yang berorientasi pada efisiensi, keamanan, dan pertumbuhan jangka panjang.

3.3 Kendala yang Ditemukan

Dalam proses pelaksanaan kerja magang, tidak dapat dipungkiri bahwa berbagai kendala dan tantangan muncul, baik dari aspek teknis maupun non-teknis. Kendala ini menjadi bagian dari proses pembelajaran dan adaptasi terhadap dunia kerja yang sesungguhnya, terutama ketika menghadapi lingkungan teknologi yang dinamis dan menuntut pemahaman mendalam dalam waktu yang relatif singkat. Adapun beberapa kendala utama yang dihadapi selama masa magang adalah sebagai berikut.

1. **Kurangnya Latar Belakang di Bidang Infrastruktur dan *Cloud Computing***

   Sebelum memasuki program magang, pemahaman terhadap bidang infrastruktur teknologi informasi dan layanan *cloud computing* masih sangat terbatas. Sebagai mahasiswa yang sebagian besar aktivitas pembelajarannya berfokus pada teori dan pengembangan perangkat lunak berskala kecil, mahasiswa belum memiliki cukup pengalaman praktis dalam hal implementasi dan manajemen infrastruktur berbasis *cloud* yang digunakan secara nyata di dunia industri. Hal ini menjadi tantangan ketika harus mempelajari dan memahami berbagai komponen teknis seperti konfigurasi jaringan, layanan *compute*, penyimpanan *cloud*, *load balancer*, serta praktik terbaik dalam hal keamanan dan efisiensi sistem *cloud*. Ketika pertama kali diperkenalkan dengan *tools* dan *platform* seperti *Amazon Web Services* (AWS) dan *Oracle Cloud Infrastructure* (OCI), mahasiswa mengalami kesulitan dalam memahami fungsi masing-masing layanan, cara kerjanya, serta integrasi antar komponen dalam sebuah arsitektur infrastruktur TI. Selain itu, penggunaan istilah-istilah teknis yang belum familiar juga menambah tingkat kesulitan dalam beradaptasi dengan lingkungan kerja yang berbasis *cloud*. Akibatnya, proses onboarding dan pelaksanaan tugas-tugas awal membutuhkan waktu lebih lama karena harus dibarengi dengan pembelajaran mandiri untuk mengejar pemahaman dasar. Hal ini menjadi salah satu hambatan utama dalam mengikuti alur kerja dan menyelesaikan tugas dengan optimal, terutama pada fase-fase awal magang.

2. **Biaya Penggunaan *Cloud***

   Tantangan berikutnya yang cukup signifikan adalah kendala biaya dalam penggunaan layanan *cloud*. Beberapa eksperimen dan tugas yang membutuhkan pengujian langsung di platform *cloud* seringkali terhambat oleh keterbatasan anggaran. Layanan-layanan premium seperti *database* berperforma tinggi, *instance* dengan spesifikasi besar, serta fitur *monitoring* tingkat lanjut pada *platform* seperti AWS atau OCI

memerlukan biaya tambahan di luar batasan layanan gratis atau *free tier*. Kondisi ini mempengaruhi ruang lingkup eksplorasi terhadap teknologi *cloud* yang sebenarnya sangat bermanfaat untuk meningkatkan pemahaman dan kemampuan teknis. Ketidakmampuan untuk mengakses layanan tertentu membuat simulasi arsitektur atau *deployment* sistem harus dilakukan secara terbatas atau bahkan hanya melalui dokumentasi dan diagram, tanpa praktik langsung. Di sisi lain, tidak adanya akun perusahaan yang menyediakan akses *cloud* berskala *enterprise* juga menambah kompleksitas. Oleh karena itu, selama pelaksanaan magang, sangat penting untuk menyusun strategi pengelolaan sumber daya *cloud* yang bijak dan efisien. Hal ini mencakup pengaturan *lifecycle resource* (seperti menghentikan instance saat tidak digunakan), pemanfaatan maksimal dari *free tier*, serta penghematan penggunaan *storage* dan *compute* untuk mencegah lonjakan biaya.

3. **Manajemen Waktu**

   Selain kendala teknis, aspek manajemen waktu juga menjadi tantangan yang cukup krusial selama pelaksanaan magang. Sebagai mahasiswa aktif yang masih menjalani kegiatan perkuliahan, mahasiswa perlu membagi waktu secara proporsional antara tanggung jawab akademik dan tanggung jawab kerja magang. Tantangan ini semakin terasa ketika terjadi tumpang tindih antara jadwal magang dengan kegiatan kampus seperti ujian, tugas kelompok, atau presentasi mata kuliah. Kesulitan dalam mengatur waktu seringkali berdampak pada keterlambatan penyelesaian tugas magang, terutama ketika tugas tersebut bersifat teknis dan membutuhkan fokus serta waktu yang cukup panjang untuk dikerjakan. Di sisi lain, fleksibilitas waktu yang diberikan oleh pihak perusahaan tetap menuntut inisiatif pribadi dalam mengatur ritme kerja agar tidak mengganggu aktivitas perkuliahan maupun performa magang. Maka dari itu, tantangan ini menuntut kemampuan manajemen waktu yang lebih baik, termasuk menyusun jadwal harian yang terstruktur, membuat *to-do-list* dengan skala prioritas, serta berkomunikasi secara terbuka dengan *supervisor* apabila

ada kendala jadwal yang tidak bisa dihindari. Dengan penyesuaian yang tepat, tantangan ini bisa dikendalikan meskipun tetap membutuhkan usaha konsisten.

3.4 Solusi atas Kendala yang Ditemukan

Selama pelaksanaan kegiatan magang, berbagai kendala dan tantangan muncul baik dari aspek teknis maupun non-teknis. Untuk menjaga kelancaran proses pembelajaran dan pelaksanaan tugas, diperlukan upaya aktif dalam mencari solusi secara mandiri maupun melalui bimbingan dari *supervisor* serta rekan satu tim di tempat magang. Solusi yang diambil bersifat adaptif terhadap konteks permasalahan yang dihadapi, serta berorientasi pada peningkatan kompetensi dan efektivitas kerja. Adapun penjelasan lebih lanjut mengenai solusi terhadap kendala-kendala tersebut adalah sebagai berikut.

1. **Kurangnya Latar Belakang di Bidang Infrastruktur dan *Cloud Computing***

   Salah satu tantangan utama yang dihadapi selama masa magang adalah keterbatasan pemahaman awal terhadap konsep infrastruktur cloud, khususnya dalam konteks penggunaan platform seperti *Amazon Web Services* (AWS) dan *Google Cloud Platform* (GCP). Hal ini menjadi hambatan terutama dalam memahami skenario migrasi *cloud*, konfigurasi layanan, serta pemanfaatan berbagai *tools* dan *service* yang digunakan dalam lingkungan *cloud*. Untuk mengatasi hal tersebut, mahasiswa harus mengambil inisiatif untuk mempelajari materi pendukung secara mandiri. Sumber-sumber pembelajaran yang digunakan antara lain sebagai berikut.

   a. Dokumentasi resmi dari AWS dan GCP yang menyajikan penjelasan teknis secara terstruktur.
   b. Mengikuti kursus daring seperti *AWS Cloud Practitioner Essentials*, *Oracle Cloud Infrastructure (OCI) Foundations*, dan materi video dari *platform* seperti Coursera, YouTube, serta AWS Skill Builder.

Selain pembelajaran mandiri, mahasiswa juga aktif berdiskusi dengan rekan kerja serta *supervisor* untuk memperdalam pemahaman terhadap konteks pekerjaan yang sedang dijalankan. Diskusi ini membantu menjelaskan berbagai istilah teknis, praktik terbaik atau *best practices*, serta tantangan umum yang sering terjadi dalam implementasi dan migrasi infrastruktur *cloud*.

2. Kendala Biaya Penggunaan *Cloud*

Pemanfaatan layanan *cloud* publik seperti AWS dan GCP secara langsung memiliki potensi biaya yang cukup tinggi apabila tidak diatur dengan baik. Dalam konteks kegiatan magang yang memiliki keterbatasan anggaran dan tidak memiliki akun dengan skema enterprise, pengelolaan biaya menjadi sangat penting agar eksperimen atau pengujian teknis tetap bisa dilakukan tanpa menimbulkan pemborosan. Solusi yang diterapkan untuk mengatasi kendala ini antara lain sebagai berikut.

a. Menggunakan layanan *Free Tier* dari AWS atau GCP yang menyediakan akses gratis ke sejumlah layanan tertentu dengan batasan sumber daya.
b. Mengatur penggunaan *resource* secara efisien, seperti mematikan instance yang tidak digunakan, menggunakan instance dengan kapasitas minimal (misalnya t2.micro atau t3.micro di AWS), serta menghindari penggunaan layanan premium yang berbiaya besar.
c. Melakukan simulasi infrastruktur menggunakan diagram arsitektur, seperti melalui AWS *Architecture Diagram Tool* atau Lucidchart, untuk merancang skenario migrasi atau *deployment* tanpa perlu langsung mengaktifkan resource yang bersifat berbayar.
d. Melakukan *monitoring* dan evaluasi biaya secara berkala dengan menggunakan fitur AWS Billing & Cost Explorer untuk memastikan tidak ada pengeluaran di luar perkiraan.

Langkah-langkah ini terbukti efektif dalam menjaga pengeluaran *cloud* tetap dalam batas wajar, tanpa mengurangi kualitas pembelajaran teknis yang diperoleh selama magang.

3. Manajemen Waktu

Selama masa magang, mahasiswa menghadapi tantangan dalam mengelola waktu antara tanggung jawab magang dan kewajiban akademik di perkuliahan. Kurangnya manajemen waktu yang baik pada awalnya sempat menimbulkan tekanan dalam menyelesaikan tugas-tugas dengan efisien. Untuk mengatasi hal ini, mahasiswa mulai menerapkan beberapa strategi perencanaan waktu yang lebih terstruktur, antara lain sebagai berikut.

a. Membuat jadwal kerja mingguan dan harian, yang memuat alokasi waktu untuk kegiatan magang, kuliah, serta waktu belajar mandiri.
b. Menetapkan prioritas tugas berdasarkan tingkat urgensi dan tenggat waktu atau *deadline*, sehingga fokus dapat diarahkan pada hal yang paling penting terlebih dahulu.
c. Menggunakan alat bantu manajemen tugas, seperti *Google Calendar*, guna mencatat progres pekerjaan serta mengingatkan jadwal rapat atau tugas penting.

Dengan pendekatan ini, mahasiswa mampu menyeimbangkan antara kegiatan magang dan aktivitas akademik, sehingga keduanya tetap dapat berjalan secara bersamaan tanpa saling mengganggu.