TUGAS AKHIR

Rancang Bangun *Tool* Otomasi Infrastruktur *Three-Tier* Berbasis *Terraform* dan *Docker* pada Layanan *Amazon Web Services*

DISUSUN OLEH:

Muhammad Abdurrouf Firmansyah

NIM. 22106050010

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA

YOGYAKARTA

2026

LEMBAR PENGESAHAN

KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Marsda Adisucipto Telp. (0274) 540971 Fax. (0274) 519739 Yogyakarta 55281

PENGESAHAN TUGAS AKHIR

Nomor : B-1268/Un.02/DST/PP.00.9/06/2026

Tugas Akhir dengan judul : Rancang Bangun Tool Otomasi Infrastruktur Three-Tier Berbasis Terraform dan Docker pada Layanan Amazon Web Services

yang dipersiapkan dan disusun oleh:

Nama : MUHAMMAD ABDURROUF FIRMANSYAH
Nomor Induk Mahasiswa : 22106050010
Telah diujikan pada : Jumat, 05 Juni 2026
Nilai ujian Tugas Akhir : A-

dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

TIM UJIAN TUGAS AKHIR

Ketua Sidang

Muhammad Galih Wonoseto, M.T.
SIGNED

Valid ID: 6a278f229049d

Penguji I

Ir. Muhammad Didik Rohmad Wahyudi, S.T., MT.
SIGNED

Valid ID: 6a278668467e3

Penguji II

Ir. Maria Ulfah Siregar, S.Kom., MIT., Ph.D.
SIGNED

Valid ID: 6a275c262475e

Yogyakarta, 05 Juni 2026
UIN Sunan Kalijaga
Dekan Fakultas Sains dan Teknologi

Prof. Dr. Dra. Hj. Khurul Wardati, M.Si.
SIGNED

Valid ID: 6a27a5464c3c9

LEMBAR PERNYATAAN

LEMBAR PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama : Muhammad Abdurrouf Firmansyah
NIM : 22106050010
Program Studi : Informatika
Fakultas : Sains dan Teknologi

Menyatakan dengan sesungguhnya, bahwa skripsi saya yang berjudul "**RANCANG BANGUN TOOL OTOMASI INFRASTRUKTUR THREE-TIER BERBASIS TERRAFORM DAN DOCKER PADA LAYANAN AMAZON WEB SERVICES**" merupakan penelitian saya sendiri, tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi dan bukan plagiasi karya orang lain kecuali secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka. Sebagai salah satu syarat untuk memperoleh gelar Sarjana Program Studi Informatika pada Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga.

Yogyakarta, 02 Juni 2026
Yang membuat pernyataan,

Muhammad Abdurrouf Firmansyah
22106050010

LEMBAR PERSETUJUAN

LEMBAR PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi / Tugas Akhir
Lamp : -

Kepada
Yth.
Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga
DI Yogyakarta

Assalamu'alaikum wr. wb.

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka saya selaku pembimbing berpendapat bahwa skripsi Saudara:

Nama	:	Muhammad Abdurrouf Firmansyah
NIM	:	22106050010
Judul Skripsi	:	Rancang Bangun Tool Otomasi Infrastruktur Three-Tier Berbasis Terraform dan Docker pada Layanan Amazon Web Services

Sudah dapat diajukan kepada Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Program Studi Informatika.

Dengan ini kami mengharap agar skripsi / tugas akhir Saudara dapat segera dimunaqosyahkan. Atas perhatiannya saya ucapkan terima kasih.

Wassalamu'alaikum wr. wb.

Yogyakarta, 02 Juni 2026
Pembimbing,

Muhammad Galih Wonoseto, M.T.
NIP. 19901113 201903 1 012

INTISARI

Konfigurasi infrastruktur *cloud* secara manual melalui *AWS Management Console* membutuhkan waktu yang relatif lama, rentan terhadap kesalahan manusia, dan sulit direplikasi secara konsisten. Permasalahan tersebut menjadi tantangan dalam pembangunan infrastruktur *cloud*, khususnya pada arsitektur *three-tier* yang membutuhkan pengaturan jaringan, komputasi, basis data, dan keamanan secara terstruktur. Penelitian ini bertujuan untuk merancang dan membangun *tool* otomasi infrastruktur *three-tier* berbasis *Terraform* dan *Docker* pada layanan *Amazon Web Services*, serta menerapkan model *Waterfall* dalam proses pengembangannya.

Metode pengembangan sistem yang digunakan adalah *Waterfall* dengan tahapan analisis kebutuhan, desain sistem, implementasi, dan pengujian. *Tool* dikembangkan menggunakan *Terraform* sebagai *Infrastructure as Code* untuk melakukan *provisioning resource AWS*, sedangkan *Docker* digunakan sebagai media kontainerisasi agar *tool* dapat dijalankan secara *portabel* tanpa instalasi *Terraform* secara manual pada perangkat pengguna. Infrastruktur yang dibangun meliputi *Virtual Private Cloud, public subnet, private subnet, Internet Gateway, Route Table, Security Group, EC2 instance* sebagai *web/application tier*, dan *RDS MySQL* sebagai *database tier*.

Hasil pengujian menunjukkan bahwa seluruh kebutuhan fungsional sistem berhasil dipenuhi. *Tool* mampu menjalankan proses inisialisasi, *plan, apply*, menampilkan *output deployment*, serta *destroy* melalui kontainer *Docker*. Hasil pengukuran waktu menunjukkan bahwa *deployment* manual membutuhkan waktu 24 menit 47 detik, sedangkan *deployment* otomatis menggunakan *tool* membutuhkan waktu 5 menit 58 detik. Dengan demikian, metode otomatis mampu mengurangi waktu *deployment* sebesar 75,92% dan berjalan sekitar 4,15 kali lebih cepat dibandingkan metode manual. Evaluasi penerimaan pengguna dilakukan melalui kuesioner berbasis model *DeLone* dan *McLean* kepada mahasiswa Informatika UIN Sunan Kalijaga angkatan 2022 dan 2023 yang pernah menggunakan *AWS*. Dari 24 butir pertanyaan, 18 butir dinyatakan valid berdasarkan uji korelasi dan r tabel. Selanjutnya, evaluasi penerimaan pengguna dilakukan terhadap 30 responden menggunakan 18 butir pertanyaan valid. Hasil evaluasi menunjukkan rata-rata keseluruhan sebesar 4,4128 dari skala 5, sehingga termasuk dalam kategori Sangat Tinggi. Berdasarkan hasil tersebut, *tool* otomasi yang dikembangkan dinilai mampu membangun infrastruktur *three-tier* secara lebih cepat, konsisten, *portabel*, dan terstruktur.

Kata kunci: *Amazon Web Services, Docker, Infrastructure as Code, Terraform, three-tier, Waterfall*.

ABSTRACT

Manual cloud infrastructure configuration through the AWS Management Console requires a relatively long time, is prone to human error, and is difficult to replicate consistently. These issues become a challenge in cloud infrastructure development, particularly in a three-tier architecture that requires structured configuration of networking, computing, database, and security components. This study aims to design and develop a three-tier infrastructure automation tool based on Terraform and Docker on Amazon Web Services, as well as to apply the Waterfall model in the development process.

The system development method used in this study is the Waterfall model, consisting of requirements analysis, system design, implementation, and testing stages. The tool was developed using Terraform as Infrastructure as Code to provision AWS resources, while Docker was used as a containerization platform to enable portable execution without requiring users to manually install Terraform on their local devices. The infrastructure built by the tool includes a Virtual Private Cloud, public subnet, private subnet, Internet Gateway, Route Table, Security Group, EC2 instance as the web/application tier, and RDS MySQL as the database tier.

The testing results show that all functional requirements were successfully fulfilled. The tool is able to execute initialization, plan, apply, display deployment outputs, and destroy processes through a Docker container. The time measurement results show that manual deployment required 24 minutes and 47 seconds, while automated deployment using the tool required 5 minutes and 58 seconds. Therefore, the automated method reduced deployment time by 75.92% and ran approximately 4.15 times faster than the manual method. User acceptance evaluation was conducted using a questionnaire based on the DeLone and McLean model, involving Informatics students of UIN Sunan Kalijaga from the 2022 and 2023 cohorts who had experience using AWS. Out of 24 questionnaire items, 18 items were declared valid based on correlation testing and r table comparison. The user acceptance evaluation was then conducted on 30 respondents using the 18 valid questionnaire items. The evaluation results showed an overall average score of 4.4128 out of 5, which falls into the Very High category. Based on these results, the developed automation tool is considered capable of building three-tier infrastructure in a faster, more consistent, portable, and structured manner.

Keywords: Amazon Web Services, Docker, Infrastructure as Code, Terraform, three-tier, Waterfall.

MOTTO

خَيْرُ النَّاسِ أَنْفَعُهُمْ لِلنَّاسِ

“But perhaps you hate a thing, And it's good for you
And perhaps you love a thing, And it's bad for you” (2:216)
Jakarta Hari Ini, For Revenge

اعْمَلْ لِدُنْيَاكَ كَأَنَّك تَعِيشُ أبَدًا، وَاعْمَلْ لِآخِرَتِكَ كَأَنَّكَ تَمُوْتُ غَدًا

HALAMAN PERSEMBAHAN

Dengan penuh rasa syukur kepada Allah SWT atas segala limpahan rahmat, nikmat, kekuatan, serta kemudahan yang telah diberikan, Tugas Akhir ini penulis persembahkan kepada:

1. Kedua orang tua tercinta, yang senantiasa memberikan doa, kasih sayang, dukungan, pengorbanan, serta semangat yang tidak pernah berhenti dalam setiap langkah penulis. Segala pencapaian ini tidak terlepas dari doa dan restu yang selalu mengiringi perjalanan penulis.
2. Orang-orang yang telah menjadi sumber motivasi, penguat, dan alasan bagi penulis untuk terus bertahan, melangkah, serta tidak menyerah dalam menghadapi setiap proses kehidupan. Kehadiran, doa, dukungan, dan kebaikan kalian menjadi bagian penting yang menguatkan penulis hingga mampu bertahan sampai sekarang.
3. Teman-teman seperjuangan Program Studi Informatika, khususnya angkatan 2022, yang telah menjadi bagian dari perjalanan akademik penulis serta memberikan semangat, bantuan, dan kebersamaan selama masa perkuliahan.

Semoga karya sederhana ini dapat menjadi bentuk kecil dari rasa terima kasih penulis kepada semua pihak yang telah memberikan doa, dukungan, dan kepercayaan selama ini.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas segala rahmat, taufik, hidayah, serta inayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Rancang Bangun *Tool* Otomasi Infrastruktur *Three-Tier* Berbasis *Terraform* dan *Docker* pada Layanan *Amazon Web Services*" dengan baik. Shalawat serta salam semoga senantiasa tercurahkan kepada Nabi Muhammad SAW, keluarga, sahabat, serta seluruh umatnya hingga akhir zaman.

Tugas Akhir ini disusun sebagai salah satu syarat untuk menyelesaikan studi pada Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga Yogyakarta. Penelitian ini membahas perancangan dan pembangunan *tool* otomasi infrastruktur cloud berbasis *Terraform* dan *Docker* pada layanan *Amazon Web Services* (AWS). *Tool* yang dikembangkan bertujuan untuk membantu proses penyediaan infrastruktur *three-tier* secara lebih otomatis, terstruktur, konsisten, dan efisien dibandingkan dengan konfigurasi manual melalui AWS *Management Console*.

Dalam proses penyusunan Tugas Akhir ini, penulis menyadari bahwa penyelesaian penelitian tidak terlepas dari bantuan, bimbingan, dukungan, dan doa dari berbagai pihak. Oleh karena itu, dengan penuh rasa hormat dan terima kasih, penulis menyampaikan penghargaan kepada:

1. Prof. Noorhaidi Hasan S.Ag., M.A., M.Phil., Ph.D. selaku Rektor UIN Sunan Kalijaga Yogyakarta.
2. Prof. Dr. Dra. Hj. Khurul Wardati, M.Si. selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
3. Bapak Dr. Muhammad Mustakim, S.T. M.T. selaku Ketua Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
4. Dr. Agus Mulyanto, S.Si., M.Kom., ASEAN Eng. selaku dosen penasihat akademik yang telah memberikan arahan dan bimbingan selama masa perkuliahan penulis.
5. Muhammad Galih Wonoseto, M.T. selaku dosen pembimbing yang telah memberikan bimbingan, masukan yang berharga, serta dukungan penuh dengan penuh kesabaran dalam setiap tahapan penyusunan Tugas Akhir ini.

6. Seluruh dosen Program Studi Informatika yang telah memberikan ilmu dan pengalaman selama masa perkuliahan.
7. Kedua orang tua dan keluarga penulis yang selalu memberikan doa, dukungan, kasih sayang, dan motivasi tanpa henti.
8. Teman-teman Program Studi Informatika yang telah memberikan bantuan, semangat, dan kebersamaan selama proses perkuliahan maupun penyusunan Tugas Akhir.
9. Seluruh responden dan pihak lain yang telah membantu dalam proses penelitian ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki kekurangan, baik dari segi isi, penulisan, maupun penyajian. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun agar karya ini dapat menjadi lebih baik. Semoga Tugas Akhir ini dapat memberikan manfaat bagi penulis, pembaca, serta pihak-pihak yang tertarik pada bidang *cloud computing*, *Infrastructure as Code, Terraform*, *Docker*, dan otomasi infrastruktur.

Yogyakarta, 02 Juni 2026

Penulis,

Muhammad Abdurrouf Firmansyah

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LISTING

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi saat ini menuntut organisasi dan pengembang perangkat lunak untuk membangun sistem yang tidak hanya mampu menjalankan fungsi utama, tetapi juga memiliki performa tinggi, skalabilitas, dan keandalan [1]. Tuntutan tersebut muncul karena sistem modern harus mampu merespons peningkatan jumlah pengguna, pertumbuhan volume data, serta perubahan kebutuhan layanan yang berlangsung secara dinamis [1], [2] [3]. Dalam konteks pengembangan perangkat lunak, performa berkaitan dengan kemampuan sistem dalam memberikan respons secara cepat, skalabilitas berkaitan dengan kemampuan sistem untuk menyesuaikan kapasitas terhadap peningkatan beban kerja, sedangkan keandalan berkaitan dengan kemampuan sistem untuk tetap beroperasi secara konsisten dalam berbagai kondisi penggunaan [1], [4]. Oleh karena itu, pembangunan sistem tidak lagi hanya berfokus pada aspek fungsional, tetapi juga harus memperhatikan kualitas arsitektur dan kesiapan infrastruktur agar layanan dapat berjalan secara stabil, terukur, dan berkelanjutan [1], [5], [6].

Dalam menghadapi lonjakan lalu lintas data yang dinamis, adopsi komputasi awan (*cloud computing*) telah menjadi standar industri yang krusial karena mampu menyediakan sumber daya komputasi secara fleksibel sesuai kebutuhan sistem [7], [8]. Melalui layanan *cloud*, organisasi dan pengembang perangkat lunak tidak lagi harus bergantung sepenuhnya pada pengadaan perangkat keras fisik, melainkan dapat menyediakan sumber daya seperti server, jaringan, penyimpanan, dan basis data secara virtual [7], [8]. Layanan *cloud*, khususnya pada model *Infrastructure as a Service* (*IaaS*), menawarkan fleksibilitas sumber daya yang memungkinkan penyediaan infrastruktur secara cepat tanpa kendala pengelolaan perangkat keras fisik [7], [9]. Model IaaS juga memberikan keleluasaan bagi pengguna untuk merancang, mengatur, dan mengelola komponen infrastruktur sesuai kebutuhan arsitektur sistem, seperti konfigurasi jaringan, mesin komputasi, aturan keamanan, serta layanan basis data [7], [8]. *Amazon Web Services*

(AWS) sebagai salah satu penyedia layanan IaaS terkemuka, menyediakan berbagai layanan komputasi dan jaringan yang sangat mendukung implementasi arsitektur sistem yang kompleks [10], [11], [12]. Dengan ketersediaan layanan seperti *Virtual Private Cloud* (VPC), *Elastic Compute Cloud* (EC2), *Security Group*, dan *Relational Database Service* (RDS), AWS dapat digunakan untuk membangun infrastruktur *cloud* yang terstruktur, tersegmentasi, dan sesuai dengan kebutuhan sistem modern [10], [11].

Meskipun layanan cloud memberikan kemudahan, proses konfigurasi dan manajemen infrastruktur di dalamnya sering kali masih dilakukan secara manual melalui antarmuka *web* (*user interface*) [13], [14]. Pendekatan manual ini merujuk pada proses penyediaan dan pengaturan sumber daya *cloud* secara satu per satu melalui *AWS Management Console*, seperti membuat *Virtual Private Cloud* (VPC), *subnet, Internet Gateway, Route Table, Security Group, EC2 instance*, dan *database RDS* dengan memasukkan parameter konfigurasi secara langsung melalui formulir dan menu konfigurasi yang tersedia [14], [10]. Proses semacam ini membutuhkan ketelitian tinggi karena setiap komponen infrastruktur saling berkaitan, sehingga kesalahan dalam menentukan parameter jaringan, aturan keamanan, maupun hubungan antar sumber daya dapat memengaruhi keberhasilan *deployment* secara keseluruhan [9], [13], [14]. Kondisi tersebut menimbulkan sejumlah kelemahan kelemahan yang cukup signifikan, di antaranya memakan waktu yang lama, rentan terhadap kesalahan manusia (*human error*), serta sulit untuk direplikasi secara presisi ketika membangun lingkungan yang berbeda seperti lingkungan pengujian dan produksi [13], [15]. Ketidakkonsistenan ini dapat menghambat siklus pengembangan perangkat lunak yang menuntut kecepatan dan ketepatan tinggi [14], [15].

Untuk menjawab tantangan keamanan dan skalabilitas, pemisahan sistem ke dalam beberapa lapisan arsitektur seperti *three-tier* menjadi sangat ideal karena pola ini memisahkan fungsi utama sistem ke dalam lapisan antarmuka, logika aplikasi, dan basis data [16], [17]. Pemisahan komponen ke dalam lapisan antarmuka, logika aplikasi, dan basis data meminimalkan risiko keamanan karena akses dari publik dapat dibatasi dengan ketat. Pemisahan tersebut memungkinkan

setiap lapisan memiliki tanggung jawab yang lebih jelas, sehingga pengelolaan akses, beban kerja, dan perlindungan data dapat dilakukan secara lebih terstruktur [16], [17]. Dalam konteks *cloud computing*, arsitektur berlapis juga mendukung isolasi jaringan karena komponen yang berhadapan langsung dengan pengguna dapat ditempatkan pada jaringan publik, sedangkan komponen sensitif seperti basis data dapat ditempatkan pada jaringan privat [17], [18]. Dengan demikian, pemisahan komponen ke dalam lapisan antarmuka, logika aplikasi, dan basis data meminimalkan risiko keamanan karena akses dari publik dapat dibatasi dengan ketat [17], [18].

Namun, membangun infrastruktur *cloud* yang terisolasi di AWS—melalui konfigurasi layanan komputasi dasar, jaringan virtual (VPC), keamanan, hingga *load balancer*—membutuhkan ketelitian tinggi yang sangat rentan kegagalan jika dikonfigurasi satu per satu secara manual [19], [7], [10]. Setiap komponen AWS seperti VPC, *subnet, route table, security group*, EC2, dan RDS memiliki parameter konfigurasi yang saling berkaitan, sehingga kesalahan pada satu bagian dapat memengaruhi konektivitas, keamanan, maupun keberhasilan *deployment* secara keseluruhan [10], [8]. Kondisi tersebut menjadi semakin kompleks ketika infrastruktur harus direplikasi pada lingkungan berbeda, karena konfigurasi manual tidak selalu menjamin konsistensi hasil antar lingkungan [13], [15]. Oleh karena itu, pembangunan infrastruktur *three-tier* di lingkungan *cloud* membutuhkan pendekatan yang lebih terstruktur agar proses konfigurasi dapat dilakukan secara konsisten, terdokumentasi, dan mengurangi risiko kesalahan manusia [13], [15].

Di sinilah peran *Infrastructure as Code* (IaC) menjadi sangat krusial dalam rekayasa perangkat lunak modern karena pendekatan ini memungkinkan infrastruktur didefinisikan, dikelola, dan direplikasi melalui kode yang terstruktur [12], [15]. IaC memungkinkan pengelolaan infrastruktur secara otomatis melalui penulisan baris kode yang terstruktur dan deklaratif, sehingga proses penyediaan sumber daya *cloud* tidak lagi bergantung sepenuhnya pada konfigurasi manual melalui antarmuka *web* [13], [14]. Dengan menggunakan pendekatan berbasis kode, konfigurasi infrastruktur dapat terdokumentasi dengan lebih baik, digunakan kembali, serta dijalankan secara konsisten pada lingkungan yang berbeda [9], [15].

Pendekatan ini menjadi penting dalam pembangunan infrastruktur *cloud* karena dapat membantu mengurangi risiko kesalahan manusia, mempercepat proses *deployment*, dan menjaga keseragaman konfigurasi antar lingkungan [13], [15].

Terraform muncul sebagai perangkat lunak IaC yang unggul karena mampu memetakan arsitektur kompleks ke dalam skrip kode yang konsisten, terdokumentasi, dan memiliki kontrol versi [19], [20]. Terraform menggunakan pendekatan deklaratif, yaitu pengguna mendefinisikan kondisi akhir infrastruktur yang diinginkan, kemudian *Terraform* melakukan proses *provisioning* berdasarkan konfigurasi tersebut [9], [19]. Dalam konteks pembangunan infrastruktur *three-tier* di AWS, *Terraform* dapat digunakan untuk mendefinisikan komponen seperti VPC, *subnet, route table, security group*, EC2, dan RDS secara terstruktur dalam satu rangkaian konfigurasi [14], [20]. Penggunaan *Terraform* memastikan bahwa setiap perubahan infrastruktur dapat dilacak melalui *state management* yang akurat, sehingga kondisi aktual sumber daya *cloud* dapat disesuaikan dengan konfigurasi yang telah didefinisikan dalam kode [20], [21]. Dengan demikian, *Terraform* menjadi pendekatan yang relevan untuk membangun infrastruktur *cloud* secara lebih otomatis, konsisten, terdokumentasi, dan mudah direplikasi dibandingkan konfigurasi manual [13], [15].

Berdasarkan permasalahan efisiensi konfigurasi manual serta pentingnya keandalan arsitektur jaringan *cloud*, penelitian ini berfokus pada rancang bangun *tool* otomasi infrastruktur *three-tier* berbasis *Terraform* dan *Docker*. *Terraform* dipilih karena mampu mendefinisikan dan menyediakan sumber daya infrastruktur *cloud* melalui pendekatan *Infrastructure as Code*, sehingga konfigurasi seperti VPC, *subnet, route table, security group*, EC2, dan RDS dapat ditulis secara deklaratif, terdokumentasi, serta dijalankan secara konsisten [19], [20]. Namun, penggunaan *Terraform* secara langsung masih memiliki ketergantungan terhadap lingkungan lokal pengguna, seperti kebutuhan instalasi *Terraform*, kesesuaian versi, struktur direktori, serta konfigurasi dependensi yang dapat memengaruhi keberhasilan eksekusi *tool* [20], [22]. Oleh karena itu, *Docker* diintegrasikan untuk mengemas konfigurasi *Terraform* beserta dependensi eksekusinya ke dalam *kontainer*,

sehingga *tool* dapat dijalankan secara lebih *portabel* dan konsisten pada berbagai perangkat selama tersedia lingkungan *Docker* [22], [23].

Integrasi *Terraform* dan *Docker* dalam penelitian ini menjadi penting karena keduanya memiliki peran yang saling melengkapi dalam proses otomasi infrastruktur. Terraform berperan sebagai mesin utama untuk melakukan *provisioning* sumber daya AWS secara otomatis, sedangkan *Docker* berperan sebagai media distribusi dan eksekusi agar pengguna tidak perlu melakukan instalasi *Terraform* serta pengaturan lingkungan secara manual [14], [22]. Dengan pendekatan ini, proses pembangunan infrastruktur *three-tier* dapat dilakukan melalui eksekusi kontainer yang lebih sederhana, terstandar, dan mengurangi potensi perbedaan konfigurasi antar perangkat pengguna [22], [23]. Pengembangan *tool* dilakukan menggunakan model *Waterfall* agar proses analisis kebutuhan, perancangan, implementasi, dan pengujian sistem dapat berjalan secara terstruktur dan terdokumentasi [24]. Melalui integrasi *Terraform*, *Docker*, dan AWS, diharapkan pembangunan infrastruktur siap pakai dapat dilakukan secara presisi, otomatis, *portabel*, dan konsisten. Oleh karena itu, penelitian ini mengambil judul "Rancang Bangun *Tool* Otomasi Infrastruktur *Three-Tier* Berbasis *Terraform* dan *Docker* pada Layanan *Amazon Web Services*."

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana merancang dan membangun *tool* otomasi infrastruktur *three-tier* pada layanan *Amazon Web Services* berbasis *Terraform* dan *Docker* agar proses *deployment* dapat dilakukan secara otomatis tanpa konfigurasi manual melalui AWS *Management Console*?
2. Bagaimana menerapkan model *Waterfall* dalam pengembangan *tool* otomasi infrastruktur *three-tier* berbasis *Terraform* dan *Docker* pada layanan *Amazon Web Services*?

1.3 Tujuan

Adapun tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Membangun *tool* otomasi infrastruktur *three-tier* berbasis *Terraform* dan *Docker* yang mampu melakukan *provisioning* sumber daya AWS secara otomatis.
2. Menerapkan model *Waterfall* dalam proses pengembangan *tool*, mulai dari analisis kebutuhan, perancangan sistem, implementasi, hingga pengujian.

1.4 Manfaat

Penelitian ini diharapkan dapat memberikan kontribusi yang signifikan, baik dalam pengembangan khazanah ilmu pengetahuan maupun dalam penyelesaian masalah teknis di lapangan. Adapun manfaat penelitian ini adalah sebagai berikut:

1.4.1 Manfaat Akademis

1. Pengembangan Metodologi Rekayasa Perangkat Lunak: Memberikan kontribusi teoritis mengenai penerapan model *Waterfall* dalam pengembangan *tool* otomasi infrastruktur *cloud* berbasis *Terraform* dan *Docker*.
2. Referensi Keilmuan IaC: Menjadi referensi bagi literatur informatika mengenai konsep *Infrastructure as Code* (IaC) yang diintegrasikan dengan teknologi kontainerisasi untuk menciptakan sistem distribusi perangkat lunak yang bersifat *portable* dan *immutable*.
3. Kajian Model Distribusi *Black-box*: Memperkaya studi mengenai model distribusi infrastruktur yang aman, di mana kode sumber (*source code*) dikemas dalam bentuk *black-box* guna menjaga integritas konfigurasi dari perubahan yang tidak terstandarisasi.

1.4.2 Manfaat Praktis

1. Efisiensi Operasional: Membantu praktisi teknologi informasi dalam melakukan *deployment* infrastruktur *three-tier* secara cepat dan konsisten tanpa harus melakukan konfigurasi manual yang repetitif melalui *dashboard* AWS *Management Console*.

2. Reduksi *Human error*: Meminimalisir potensi kesalahan manusia (*human error*) dalam pengaturan jaringan dan basis data melalui penggunaan templat kode yang sudah teruji dan tersegel di dalam image *Docker*.
3. Aksesibilitas Alat Otomasi: Memberikan solusi praktis bagi pengguna atau pengembang yang tidak memiliki pemahaman mendalam mengenai sintaks Terraform agar tetap dapat menyediakan lingkungan server yang andal hanya melalui satu baris perintah eksekusi.
4. Portabilitas Sistem: Menyediakan alat penyedia infrastruktur yang dapat dijalankan di berbagai sistem operasi selama tersedia lingkungan *Docker*, sehingga memudahkan proses pemindahan dan standarisasi lingkungan laboratorium jaringan.

1.5 Batasan Masalah

Agar penelitian ini lebih terarah dan tidak menyimpang dari tujuan yang telah ditetapkan, maka diberikan batasan masalah sebagai berikut:

1. Layanan *Cloud*: Infrastruktur yang dibangun terbatas pada layanan *Amazon Web Services* (AWS) dengan menggunakan *region ap-southeast-1* (Singapore).
2. Komponen Arsitektur: Arsitektur *three-tier* pada penelitian ini dibatasi pada rancangan infrastruktur *cloud* sederhana yang memisahkan lapisan akses publik dan lapisan data privat melalui VPC, *public subnet, private subnet,* satu *instance* EC2 sebagai *web/application tier*, dan satu *instance* RDS MySQL sebagai *database tier*.
3. Lingkup Infrastruktur: Penelitian ini tidak mencakup implementasi *Load Balancer* maupun *Auto Scaling Group*, melainkan berfokus pada isolasi jaringan dan otomasi penyediaan komponen dasar *three-tier*.
4. Alat dan Teknologi: Otomasi infrastruktur dilakukan menggunakan *Terraform* (HCL), dan distribusi alat dilakukan melalui kontainerisasi *Docker*.
5. Metodologi: Pengembangan *tool* menggunakan model *Waterfall* yang meliputi tahapan analisis kebutuhan, desain sistem, implementasi, dan

pengujian. Tahap pemeliharaan tidak dibahas karena penelitian berfokus pada pembangunan dan validasi awal *tool* otomasi infrastruktur dalam lingkup tugas akhir.

6. Sistem Distribusi: Alat bersifat *black-box* (kode sumber *Terraform* berada di dalam *image Docker*) dan didistribusikan melalui *registry publik Docker Hub*.
7. Pengujian: Pengujian dilakukan pada aspek fungsionalitas eksekusi alat (plan, apply, destroy) serta verifikasi konektivitas antar lapisan (*tier*) pada infrastruktur yang telah terbentuk.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa tool otomasi infrastruktur *three-tier* berbasis *Terraform* dan *Docker* pada layanan *Amazon Web Services* berhasil dirancang dan dibangun sesuai dengan kebutuhan sistem yang telah ditetapkan. *Tool* ini mampu menjalankan konfigurasi Terraform melalui kontainer *Docker*, melakukan inisialisasi, menampilkan rencana perubahan infrastruktur, membangun *resource* AWS, menampilkan output deployment, serta menghapus kembali *resource* yang telah dibuat. Hasil pengujian fungsional menunjukkan bahwa seluruh kebutuhan fungsional sistem, mulai dari KF-01 sampai KF-10, telah terpenuhi.

Penerapan model *Waterfall* dalam penelitian ini dapat digunakan secara terstruktur untuk mengembangkan *tool* otomasi infrastruktur. Tahapan *Waterfall* yang meliputi analisis kebutuhan, desain sistem, implementasi, dan pengujian telah diterapkan secara berurutan. Tahap analisis kebutuhan menghasilkan kebutuhan fungsional, kebutuhan nonfungsional, kebutuhan lingkungan lokal, dan kebutuhan infrastruktur cloud. Tahap desain menghasilkan rancangan topologi *three-tier* dan alur kerja otomasi. Tahap implementasi menghasilkan konfigurasi *Terraform* modular, *Dockerfile*, dan *script entrypoint*. Tahap pengujian menghasilkan bukti bahwa tool dapat berjalan sesuai rancangan dan memenuhi kebutuhan sistem.

Hasil pengukuran efisiensi waktu deployment menunjukkan bahwa metode otomatis menggunakan Docker dan Terraform lebih efisien dibandingkan metode manual melalui AWS *Management Console*. *Deployment* manual membutuhkan waktu 24 menit 47 detik atau 1.487 detik, sedangkan deployment otomatis membutuhkan waktu 5 menit 58 detik atau 358 detik. Selisih waktu antara kedua metode adalah 1.129 detik atau sekitar 18 menit 49 detik. Berdasarkan hasil tersebut, metode otomatis mampu mengurangi waktu deployment sebesar 75,92% dan berjalan sekitar 4,15 kali lebih cepat dibandingkan metode manual.

Evaluasi penerimaan pengguna menunjukkan bahwa tool otomasi infrastruktur yang dikembangkan memperoleh penerimaan yang baik dari responden. Dari 24 butir pertanyaan kuesioner yang diuji menggunakan korelasi dan r tabel, sebanyak 18 butir dinyatakan valid dan 6 butir dinyatakan tidak valid. Hasil ini menunjukkan bahwa sebagian besar instrumen mampu merepresentasikan penilaian responden terhadap kualitas sistem, kualitas informasi, kualitas layanan, penggunaan, kepuasan pengguna, dan manfaat tool. Evaluasi penerimaan pengguna terhadap 30 responden menggunakan 18 butir pertanyaan valid menghasilkan rata-rata keseluruhan sebesar 4,4128 dari skala 5, sehingga termasuk dalam kategori Sangat Tinggi. Hasil ini menunjukkan bahwa tool otomasi infrastruktur berbasis Terraform dan Docker memperoleh penerimaan yang sangat baik dari pengguna. Selain itu, verifikasi kebutuhan nonfungsional menunjukkan bahwa tool telah memenuhi aspek portabilitas, konsistensi konfigurasi, keamanan kredensial, pengurangan interaksi manual, isolasi jaringan, ketersediaan informasi eksekusi, dan kemampuan penghapusan resource setelah deployment selesai.

Dengan demikian, penelitian ini berhasil menghasilkan *tool* otomasi infrastruktur yang dapat membantu proses deployment arsitektur *three-tier* pada layanan *Amazon Web Services* secara lebih cepat, konsisten, dan terstruktur. *Tool* yang dikembangkan juga mampu mengurangi ketergantungan terhadap konfigurasi manual melalui AWS *Management Console*, sehingga dapat meminimalkan potensi kesalahan manusia dalam proses penyediaan infrastruktur *cloud*.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa saran yang dapat digunakan untuk pengembangan penelitian selanjutnya. Pertama, tool otomasi infrastruktur dapat dikembangkan dengan menambahkan komponen cloud yang lebih kompleks, seperti Load Balancer, Auto Scaling Group, NAT Gateway, dan monitoring service. Penambahan komponen tersebut dapat membuat arsitektur yang dibangun menjadi lebih mendekati kebutuhan lingkungan produksi.

Kedua, pengelolaan state Terraform pada penelitian selanjutnya dapat dikembangkan menggunakan remote backend, seperti Amazon S3 yang

dikombinasikan dengan DynamoDB untuk mekanisme state locking. Penggunaan remote backend akan membantu menjaga konsistensi state ketika tool digunakan oleh lebih dari satu pengguna atau dijalankan dalam lingkungan kolaboratif.

Ketiga, penelitian selanjutnya dapat menambahkan dukungan multi-environment, seperti development, *staging*, dan *production*. Dengan adanya pemisahan environment, tool dapat digunakan untuk membangun infrastruktur dengan konfigurasi yang berbeda sesuai kebutuhan masing-masing lingkungan tanpa mengubah struktur utama modul Terraform.

Keempat, tool dapat dikembangkan dengan menambahkan antarmuka penggunaan yang lebih mudah, misalnya Command Line Interface (CLI) khusus atau antarmuka berbasis web sederhana. Pengembangan ini dapat membantu pengguna yang belum terbiasa dengan Docker dan Terraform agar dapat menjalankan proses deployment dengan lebih mudah.

Kelima, evaluasi pengguna pada penelitian selanjutnya dapat dilakukan dengan jumlah responden yang lebih besar dan latar belakang yang lebih beragam, misalnya praktisi cloud, DevOps engineer, atau pengembang perangkat lunak yang telah menggunakan layanan cloud dalam proyek nyata. Dengan cakupan responden yang lebih luas, hasil evaluasi penerimaan pengguna dapat memberikan gambaran yang lebih kuat mengenai kelayakan penggunaan tool di luar lingkungan akademik.

Keenam, pengujian sistem dapat diperluas tidak hanya pada aspek fungsionalitas dan efisiensi waktu, tetapi juga pada aspek keamanan, reliability, dan cost estimation. Pengujian tersebut penting untuk mengetahui sejauh mana tool dapat digunakan pada skenario deployment yang lebih kompleks dan mendekati kebutuhan operasional sebenarnya.

.

DAFTAR PUSTAKA

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice, 4th Edition*, 4th ed. Addison-Wesley Professional, 2021. [Online]. Available: https://www.sei.cmu.edu/library/software-architecture-in-practice-fourth-edition/

[2] G. Vale, F. F. Correia, E. Martins Guerra, T. De Oliveira Rosa, J. Fritzsch, and J. Bogner, "Summary of the artifact accompanying the article : 'designing Microservice Systems Using Patterns: An Empirical Study on Quality Trade-Offs,'" *2022 IEEE 19th Int. Conf. Softw. Archit. Companion, ICSA-C 2022*, p. 57, 2022, doi: 10.1109/ICSA-C54293.2022.00020.

[3] R. Srinivas Ramesh, "Scalable Systems and Software Architectures for High-Performance Computing on cloud platforms," 2024, [Online]. Available: https://arxiv.org/pdf/2408.10281

[4] D. Di Pompeo and M. Tucci, "Quality Attributes Optimization of Software Architecture: Research Challenges and Directions," *Proc. - IEEE 20th Int. Conf. Softw. Archit. Companion, ICSA-C 2023*, pp. 252–255, 2023, doi: 10.1109/ICSA-C57050.2023.00061.

[5] I. Sommerville, *Software Engineering 6TH Edition Synopses and Reviews Table of Contents*, vol., no. I. Sommerville, "Software Engineering 6TH Edition Synopses and Reviews Table of Contents," vol., no., pp. 1–7, 2016., 2016. [Online]. Available: https://www.mpgcamb.com/wp-content/uploads/2024/12/International-Computer-Science-Series-Sommerville-Ian-Software-engineering-Pearson-2015_2016.pdf

[6] Z. Wan, Y. Zhang, X. Xia, Y. Jiang, and D. Lo, "Software Architecture in Practice: Challenges and Opportunities," *ESEC/FSE 2023 - Proc. 31st ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, pp. 1457–1469, 2023, doi: 10.1145/3611643.3616367.

[7] E. Wahyuningsih, Dian; Affandi, Luqman; Sophia, "Evaluasi Performa Model Layanan Cloud Computing : Studi Komparatif Iaas , Paas , Dan Saas Dengan Pengujian Beban," vol. 16, no. 2, 2025, doi: 10.36382/jti-tki.v16i2.625.

[8] R. L. Rahardian, "Analisa Perbandingan Cloud Management Pada Google Cloud Platform dan Amazon Web Services," *Digit. Transform. Technol.*, vol. 5, no. 1, pp. 97–106, 2025, doi: 10.47709/digitech.v5i1.5862.

[9] A. Pessa, "Comparative Study of Infrastructure aas Code Tools," no. June, 2023, [Online]. Available: https://urn.fi/URN:NBN:fi:tuni-202306056530

[10] J. Varia and S. Mathew, "Overview of Amazon Web Services," no. January, pp. 1–30, 2014, [Online]. Available: http://media.amazonwebservices.com/AWS_Overview.pdf

[11] A. Amrullah, A. Nugroho, and Z. Ramadhan, "Perbandingan Kinerja Webserver pada Penyedia Layanan Cloud Microsoft Azure dan Amazon Web Services Menggunakan Metode Benchmarking," vol. 6, no. 0, pp. 167–186, 2025, doi: 10.51401/jinteks.v5i1.2487.

[12] U. Ijaz, "A Comparative Lens on Infrastructure-as-Code Provisioning Solutions for Microsoft Azure," KTH Royal Institute of Technology, 2024. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1941451/FULLTEXT01.pdf

[13] D. Gustian, Y. Fitrisia, W. Novayani, and S. Purwantoro E.S.G.S, "Implementasi Automation Deployment pada Google Cloud Compute VM menggunakan Terraform," *INOVTEK Polbeng - Seri Inform.*, vol. 8, no. 1, p. 50, 2023, doi: 10.35314/isi.v8i1.3095.

[14] Y. Hidayat and B. Arifwidodo, "Implementasi Web Server Menggunakan Infrastructure As Code Terraform Berbasis Cloud Computing," *Format J. Ilm. Tek. Inform.*, vol. 10, no. 2, p. 192, 2021, doi: 10.22441/format.2021.v10.i2.010.

[15] H. G. Gowda, "Infrastructure as Code in Action : Secure , Scalable Cloud Provisioning with Terraform and Hashicorp," 2021, [Online]. Available: https://www.ijset.in/wp-content/uploads/IJSET_V9_issue6_527..pdf?

[16] G. Alzboon and A. Al-Said Ahmad, "A Performance Evaluation Approach for n-tier Cloud-Based Software Services," *ACM Int. Conf. Proceeding Ser.*, pp. 31–36, 2022, doi: 10.1145/3555962.3555968.

[17] M. G. Labrador, A. Bordios, and W. Hou, "A cloud based 3-tier data security

framework for industrial internet of things," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 24, no. 2, p. 780, 2021, doi: 10.11591/ijeecs.v24.i2.pp780-788.

[18] A. Salmijärvi, "Cloud Architecture Evaluation," *Stud. Syst. Decis. Control*, vol. 29, pp. 109–126, 2015, doi: 10.1007/978-3-319-18778-5_6.

[19] D. Karlsson, "Comparison of Infrastructure as Code Frameworks from a Developer Perspective," 2023, [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-196142

[20] J. Nawagamuwa, "Infrastructure as Code Frameworks Evaluation for Serverless Applications Testing in AWS," no. May, 2023, [Online]. Available: https://trepo.tuni.fi/bitstream/handle/10024/149140/NawagamuwaJanaka.pdf?sequence=2&isAllowed=y

[21] Y. Omofoyewa, A. Grebe, and P. Leusmann, "IaC reusability for Hybrid Cloud Environment," 2009, [Online]. Available: https://www.researchgate.net/profile/Yaqub-Omofoyewa/publication/357281177_IaC_reusability_for_Hybrid_Cloud_Environment/links/61c4a1390ae6751c882ea59e/IaC-reusability-for-Hybrid-Cloud-Environment.pdf

[22] P. Muzumdar, A. Bhosale, G. P. Basyal, and G. Kurian, "Navigating the Docker Ecosystem: A Comprehensive Taxonomy and Survey," *Asian J. Res. Comput. Sci.*, vol. 17, no. 1, pp. 42–61, 2024, doi: 10.9734/ajrcos/2024/v17i1411.

[23] J. Z. Tam *et al.*, *A Containerization Framework for Bioinformatics Software to Advance Scalability, Portability, and Maintainability*, vol. 1, no. 1. Association for Computing Machinery, 2023. doi: 10.1145/3584371.3612948.

[24] P. G. Q. B. Liandaru, "Otomatisasi Manajemen File Konfigurasi Piranti Remote Laboratorium," *J. Sains, Nalar, dan Apl. Teknol. Inf.*, vol. 4, no. 1, pp. 9–17, 2025, doi: 10.20885/snati.v4.i1.2.

[25] Y. C. Eka Paksi and I. R. Widiasari, "Website Network Automation Design and Implementation in Rt Rw Net Senden Dusun Magelang With Django Framework," *J. Tek. Inform.*, vol. 3, no. 5, pp. 1313–1322, 2022, doi:

10.20884/1.jutif.2022.3.5.350.

[26] A. Hamidi Younessi, “Evaluating the advantages of Cisco ACI over traditional Three-Tier Architecture Subject Evaluating the advantages of Cisco ACI over traditional Three-Tier Architecture,” 2025, [Online]. Available: https://urn.fi/URN:NBN:fi:amk-2025060319662

[27] E. William, “Immutable Infrastructure : Principles and Implementations,” 2025, [Online]. Available: https://www.researchgate.net/profile/Elijah-William-2/publication/391023516_Immutable_Infrastructure_Principles_and_Implementations/links/68088599bd3f1930dd631396/Immutable-Infrastructure-Principles-and-Implementations.pdf?origin=publicationDetail&_sg%5B0

[28] R. S. Pressman, *Software Engineering: A Practitioner’s Approach*. 2015. doi: 10.1145/336512.336521.

[29] Estevania and H. Septanto, “Penerapan Metode Waterfall dalam Pengembangan Sistem Informasi Penggajian Karyawan Berbasis Web (Studi Kasus: PT Bumi Biru Prakarsa),” vol. 10, no. 2, pp. 2855–2860, 2026, [Online]. Available: https://www.researchgate.net/publication/403586509_PENERAPAN_METODE_WATERFALL_DALAM_PENGEMBANGAN_SISTEM_INFORMASI_PENGGAJIAN_KARYAWAN_BERBASIS_WEB_STUDI_KASUS_PT_BUMI_BIRU_PRAKARSA/fulltext/6a24e5592a38bf6a964dd15f/PENERAPAN-METODE-WATERFALL-DALAM-PENGEMB

[30] A. A. Azis and A. Voutama, “Rancangan Sistem Monitoring Siswa Berbasis Web dengan Metode Waterfall,” *J. Inform. dan Tek. Elektro Terap.*, vol. 13, no. 3, 2025, doi: 10.23960/jitet.v13i3.6676.

[31] D. S. Purnia, A. Rifai, and S. Rahmatullah, “Penerapan Metode Waterfall dalam Perancangan Sistem Informasi Aplikasi Bantuan Sosial Berbasis AndroidPenerapan Metode Waterfall dalam Perancangan Sistem Informasi Aplikasi Bantuan Sosial Berbasis Android,” vol. 5, no. 2, pp. 64–73, 2023, [Online]. Available: https://jurnal.umj.ac.id/index.php/semnastek/article/view/5238/3516

[32] Uminingsih, M. N. Ichsanudin, M. Yusuf, and Suraya, "Perpustakaan Dengan Metode Black Box Testing Bagi Pemula," *STORAGE-Jurnal Ilm. Tek. dan Ilmu Komput.*, vol. 1, no. 2, pp. 1–8, 2022, doi: 10.55123/storage.v1i2.270.

[33] D. Febiharsa, I. M. Sudana, and N. Hudallah, "Uji Fungsionalitas (Blackbox Testing) Sistem Informasi Lembaga Sertifikasi Profesi (SILSP) Batik dengan AppPerfect Web Test dan Uji Pengguna," *J. Informatics Educ.*, vol. 1, no. 2, pp. 117–126, 2018, doi: 10.31331/joined.v1i2.752.

[34] D. S. Putra and M. A. Darmawan, "Analisis Kepuasan Pengguna Sistem Informasi Administrasi Rumah Sakit (SIARS) dengan Model Delone and Mclean," *J. Sist. Inf. Bisnis*, vol. 11, no. 1, pp. 78–85, 2021, doi: 10.21456/vol11iss1pp78-85.

[35] T. B. Narendra and T. W. Rineksi, "Geoprocessing Model For Automation Of Land Use Balance Analysis Model Geoprocessing Untuk Otomatisasi Analisis Neraca Penatagunaan Tanah," vol. 5, no. 2, pp. 829–850, 2026, doi: 10.37676/jmcs.v5i2.10733.

[36] H. A. Damanik and M. Anggraeni, "Manajemen Jaringan Terpusat untuk Konfigurasi dan Otomatisasi Pemulihan Menggunakan Paramiko dan Django Framework," vol. 11, pp. 369–382, 2025, doi: 10.28932/jutisi.v11i3.11431.

[37] R. Purwasih and Firdaus, "Evaluasi Kesuksesan Sistem Informasi Berdasarkan Kualitas, Kepuasan Pengguna, dan Manfaat Bersih Menggunakan Model DeLone dan McLean," pp. 187–201, 2026, [Online]. Available: https://rcf-indonesia.org/jurnal/index.php/jsit/article/download/951/563

[38] D. E. Sanusi, M. Haikal, B. D. Gunawan, M. H. Aiman, and S. Umaroh, "Pengaruh Kualitas Sistem Perwalian pada SIKAD ITENAS terhadap Kepuasan Mahasiswa Menggunakan Model DeLone & McLean," vol. 1, no. 1, pp. 1–10, 2025, [Online]. Available: https://publikasi.kocenin.com/index.php/pakar/article/download/642/527

[39] Sugiyono, *Metode Penelitian Kuantitatif, Kualitatif dan R&D*, Edisi 2, C.

Bandung: Alfabeta, 2019. [Online]. Available: https://dn721804.ca.archive.org/0/items/buku-metode-penelitian-sugiyono/buku-metode-penelitian-sugiyono.pdf

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice, 4th Edition*, 4th ed. Addison-Wesley Professional, 2021. [Online]. Available: https://www.sei.cmu.edu/library/software-architecture-in-practice-fourth-edition/

[2] G. Vale, F. F. Correia, E. Martins Guerra, T. De Oliveira Rosa, J. Fritzsch, and J. Bogner, "Summary of the artifact accompanying the article : 'designing Microservice Systems Using Patterns: An Empirical Study on Quality Trade-Offs,'" *2022 IEEE 19th Int. Conf. Softw. Archit. Companion, ICSA-C 2022*, p. 57, 2022, doi: 10.1109/ICSA-C54293.2022.00020.

[3] R. Srinivas Ramesh, "Scalable Systems and Software Architectures for High-Performance Computing on cloud platforms," 2024, [Online]. Available: https://arxiv.org/pdf/2408.10281

[4] D. Di Pompeo and M. Tucci, "Quality Attributes Optimization of Software Architecture: Research Challenges and Directions," *Proc. - IEEE 20th Int. Conf. Softw. Archit. Companion, ICSA-C 2023*, pp. 252–255, 2023, doi: 10.1109/ICSA-C57050.2023.00061.

[5] I. Sommerville, *Software Engineering 6TH Edition Synopses and Reviews Table of Contents*, vol., no. I. Sommerville, "Software Engineering 6TH Edition Synopses and Reviews Table of Contents," vol., no., pp. 1–7, 2016., 2016. [Online]. Available: https://www.mpgcamb.com/wp-content/uploads/2024/12/International-Computer-Science-Series-Sommerville-Ian-Software-engineering-Pearson-2015_2016.pdf

[6] Z. Wan, Y. Zhang, X. Xia, Y. Jiang, and D. Lo, "Software Architecture in Practice: Challenges and Opportunities," *ESEC/FSE 2023 - Proc. 31st ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, pp. 1457–1469, 2023, doi: 10.1145/3611643.3616367.

[7] E. Wahyuningsih, Dian; Affandi, Luqman; Sophia, "Evaluasi Performa Model Layanan Cloud Computing : Studi Komparatif Iaas , Paas , Dan Saas

Dengan Pengujian Beban," vol. 16, no. 2, 2025, doi: 10.36382/jti-tki.v16i2.625.

[8] R. L. Rahardian, "Analisa Perbandingan Cloud Management Pada Google Cloud Platform dan Amazon Web Services," *Digit. Transform. Technol.*, vol. 5, no. 1, pp. 97–106, 2025, doi: 10.47709/digitech.v5i1.5862.

[9] A. Pessa, "Comparative Study of Infrastructure aas Code Tools," no. June, 2023, [Online]. Available: https://urn.fi/URN:NBN:fi:tuni-202306056530

[10] J. Varia and S. Mathew, "Overview of Amazon Web Services," no. January, pp. 1–30, 2014, [Online]. Available: http://media.amazonwebservices.com/AWS_Overview.pdf

[11] A. Amrullah, A. Nugroho, and Z. Ramadhan, "Perbandingan Kinerja Webserver pada Penyedia Layanan Cloud Microsoft Azure dan Amazon Web Services Menggunakan Metode Benchmarking," vol. 6, no. 0, pp. 167–186, 2025, doi: 10.51401/jinteks.v5i1.2487.

[12] U. Ijaz, "A Comparative Lens on Infrastructure-as-Code Provisioning Solutions for Microsoft Azure," KTH Royal Institute of Technology, 2024. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1941451/FULLTEXT01.pdf

[13] D. Gustian, Y. Fitrisia, W. Novayani, and S. Purwantoro E.S.G.S, "Implementasi Automation Deployment pada Google Cloud Compute VM menggunakan Terraform," *INOVTEK Polbeng - Seri Inform.*, vol. 8, no. 1, p. 50, 2023, doi: 10.35314/isi.v8i1.3095.

[14] Y. Hidayat and B. Arifwidodo, "Implementasi Web Server Menggunakan Infrastructure As Code Terraform Berbasis Cloud Computing," *Format J. Ilm. Tek. Inform.*, vol. 10, no. 2, p. 192, 2021, doi: 10.22441/format.2021.v10.i2.010.

[15] H. G. Gowda, "Infrastructure as Code in Action : Secure , Scalable Cloud Provisioning with Terraform and Hashicorp," 2021, [Online]. Available: https://www.ijset.in/wp-content/uploads/IJSET_V9_issue6_527..pdf?

[16] G. Alzboon and A. Al-Said Ahmad, "A Performance Evaluation Approach for n-tier Cloud-Based Software Services," *ACM Int. Conf. Proceeding Ser.*,

pp. 31–36, 2022, doi: 10.1145/3555962.3555968.

[17] M. G. Labrador, A. Bordios, and W. Hou, "A cloud based 3-tier data security framework for industrial internet of things," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 24, no. 2, p. 780, 2021, doi: 10.11591/ijeecs.v24.i2.pp780-788.

[18] A. Salmijärvi, "Cloud Architecture Evaluation," *Stud. Syst. Decis. Control*, vol. 29, pp. 109–126, 2015, doi: 10.1007/978-3-319-18778-5_6.

[19] D. Karlsson, "Comparison of Infrastructure as Code Frameworks from a Developer Perspective," 2023, [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-196142

[20] J. Nawagamuwa, "Infrastructure as Code Frameworks Evaluation for Serverless Applications Testing in AWS," no. May, 2023, [Online]. Available: https://trepo.tuni.fi/bitstream/handle/10024/149140/NawagamuwaJanaka.pdf?sequence=2&isAllowed=y

[21] Y. Omofoyewa, A. Grebe, and P. Leusmann, "IaC reusability for Hybrid Cloud Environment," 2009, [Online]. Available: https://www.researchgate.net/profile/Yaqub-Omofoyewa/publication/357281177_IaC_reusability_for_Hybrid_Cloud_Environment/links/61c4a1390ae6751c882ea59e/IaC-reusability-for-Hybrid-Cloud-Environment.pdf

[22] P. Muzumdar, A. Bhosale, G. P. Basyal, and G. Kurian, "Navigating the Docker Ecosystem: A Comprehensive Taxonomy and Survey," *Asian J. Res. Comput. Sci.*, vol. 17, no. 1, pp. 42–61, 2024, doi: 10.9734/ajrcos/2024/v17i1411.

[23] J. Z. Tam *et al.*, *A Containerization Framework for Bioinformatics Software to Advance Scalability, Portability, and Maintainability*, vol. 1, no. 1. Association for Computing Machinery, 2023. doi: 10.1145/3584371.3612948.

[24] P. G. Q. B. Liandaru, "Otomatisasi Manajemen File Konfigurasi Piranti Remote Laboratorium," *J. Sains, Nalar, dan Apl. Teknol. Inf.*, vol. 4, no. 1, pp. 9–17, 2025, doi: 10.20885/snati.v4.i1.2.

[25] Y. C. Eka Paksi and I. R. Widiasari, "Website Network Automation Design

and Implementation in Rt Rw Net Senden Dusun Magelang With Django Framework," *J. Tek. Inform.*, vol. 3, no. 5, pp. 1313–1322, 2022, doi: 10.20884/1.jutif.2022.3.5.350.

[26] A. Hamidi Younessi, "Evaluating the advantages of Cisco ACI over traditional Three-Tier Architecture Subject Evaluating the advantages of Cisco ACI over traditional Three-Tier Architecture," 2025, [Online]. Available: https://urn.fi/URN:NBN:fi:amk-2025060319662

[27] E. William, "Immutable Infrastructure : Principles and Implementations," 2025, [Online]. Available: https://www.researchgate.net/profile/Elijah-William-2/publication/391023516_Immutable_Infrastructure_Principles_and_Implementations/links/68088599bd3f1930dd631396/Immutable-Infrastructure-Principles-and-Implementations.pdf?origin=publicationDetail&_sg%5B0

[28] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. 2015. doi: 10.1145/336512.336521.

[29] Estevania and H. Septanto, "Penerapan Metode Waterfall dalam Pengembangan Sistem Informasi Penggajian Karyawan Berbasis Web (Studi Kasus: PT Bumi Biru Prakarsa)," vol. 10, no. 2, pp. 2855–2860, 2026, [Online]. Available: https://www.researchgate.net/publication/403586509_PENERAPAN_METODE_WATERFALL_DALAM_PENGEMBANGAN_SISTEM_INFORMASI_PENGGAJIAN_KARYAWAN_BERBASIS_WEB_STUDI_KASUS_PT_BUMI_BIRU_PRAKARSA/fulltext/6a24e5592a38bf6a964dd15f/PENERAPAN-METODE-WATERFALL-DALAM-PENGEMB

[30] A. A. Azis and A. Voutama, "Rancangan Sistem Monitoring Siswa Berbasis Web dengan Metode Waterfall," *J. Inform. dan Tek. Elektro Terap.*, vol. 13, no. 3, 2025, doi: 10.23960/jitet.v13i3.6676.

[31] D. S. Purnia, A. Rifai, and S. Rahmatullah, "Penerapan Metode Waterfall dalam Perancangan Sistem Informasi Aplikasi Bantuan Sosial Berbasis AndroidPenerapan Metode Waterfall dalam Perancangan Sistem Informasi Aplikasi Bantuan Sosial Berbasis Android," vol. 5, no. 2, pp. 64–73, 2023,

[Online]. Available: https://jurnal.umj.ac.id/index.php/semnastek/article/view/5238/3516

[32] Uminingsih, M. N. Ichsanudin, M. Yusuf, and Suraya, "Perpustakaan Dengan Metode Black Box Testing Bagi Pemula," *STORAGE-Jurnal Ilm. Tek. dan Ilmu Komput.*, vol. 1, no. 2, pp. 1–8, 2022, doi: 10.55123/storage.v1i2.270.

[33] D. Febiharsa, I. M. Sudana, and N. Hudallah, "Uji Fungsionalitas (Blackbox Testing) Sistem Informasi Lembaga Sertifikasi Profesi (SILSP) Batik dengan AppPerfect Web Test dan Uji Pengguna," *J. Informatics Educ.*, vol. 1, no. 2, pp. 117–126, 2018, doi: 10.31331/joined.v1i2.752.

[34] D. S. Putra and M. A. Darmawan, "Analisis Kepuasan Pengguna Sistem Informasi Administrasi Rumah Sakit (SIARS) dengan Model Delone and Mclean," *J. Sist. Inf. Bisnis*, vol. 11, no. 1, pp. 78–85, 2021, doi: 10.21456/vol11iss1pp78-85.

[35] T. B. Narendra and T. W. Rineksi, "Geoprocessing Model For Automation Of Land Use Balance Analysis Model Geoprocessing Untuk Otomatisasi Analisis Neraca Penatagunaan Tanah," vol. 5, no. 2, pp. 829–850, 2026, doi: 10.37676/jmcs.v5i2.10733.

[36] H. A. Damanik and M. Anggraeni, "Manajemen Jaringan Terpusat untuk Konfigurasi dan Otomatisasi Pemulihan Menggunakan Paramiko dan Django Framework," vol. 11, pp. 369–382, 2025, doi: 10.28932/jutisi.v11i3.11431.

[37] R. Purwasih and Firdaus, "Evaluasi Kesuksesan Sistem Informasi Berdasarkan Kualitas, Kepuasan Pengguna, dan Manfaat Bersih Menggunakan Model DeLone dan McLean," pp. 187–201, 2026, [Online]. Available: https://rcf-indonesia.org/jurnal/index.php/jsit/article/download/951/563

[38] D. E. Sanusi, M. Haikal, B. D. Gunawan, M. H. Aiman, and S. Umaroh, "Pengaruh Kualitas Sistem Perwalian pada SIKAD ITENAS terhadap Kepuasan Mahasiswa Menggunakan Model DeLone & McLean," vol. 1, no. 1, pp. 1–10, 2025, [Online]. Available:

https://publikasi.kocenin.com/index.php/pakar/article/download/642/527

[39] Sugiyono, *Metode Penelitian Kuantitatif, Kualitatif dan R&D*, Edisi 2, C. Bandung: Alfabeta, 2019. [Online]. Available: https://dn721804.ca.archive.org/0/items/buku-metode-penelitian-sugiyono/buku-metode-penelitian-sugiyono.pdf

LAMPIRAN

Lampiran 1 Panduan Penggunaan Tool Otomasi Infrastruktur AWS

Prasyarat Sebelum Memulai

Sebelum menggunakan alat ini, pastikan komputer Anda telah memenuhi hal berikut:

1. Docker Desktop sudah terinstal dan sedang berjalan.
2. Akun AWS yang aktif dan memiliki izin untuk membuat layanan VPC, EC2, dan RDS.
3. Akses Internet untuk mengunduh image dari Docker Hub.

Langkah Langkah

1. Persiapan Kredensial (File `.env`)
   Karena alat ini bersifat tertutup (*black-box*), Anda perlu menyediakan kredensial AWS melalui file rahasia bernama `.env`.
   - Buat folder baru di komputer Anda.
   - Di dalam folder tersebut, buat file baru bernama `.env` (pastikan tidak ada ekstensi `.txt`).
   - buat file baru bernama 'terraform.tfstate'
   - Buka file '.env' dengan Notepad atau teks editor lainnya, lalu masukkan kode berikut:

     ```
     # Kredensial AWS
     AWS_ACCESS_KEY_ID=MASUKKAN_ACCESS_KEY_ANDA
     AWS_SECRET_ACCESS_KEY=MASUKKAN_SECRET_KEY_ANDA
     AWS_DEFAULT_REGION=ap-southeast-1

     # Pengaturan Database
     TF_VAR_db_username=MasukkanUsername
     TF_VAR_db_password=MasukkanPasswordDatabase123!
     ```

     Catatan: Jangan bagikan file ini kepada siapa pun karena berisi akses penting ke akun AWS Anda.

2. Mengambil Alat dari Docker Hub
   - Buka Terminal atau PowerShell di folder tempat file `.env` berada, lalu jalankan perintah berikut untuk mengunduh versi terbaru dari alat ini:

     ```
     docker pull rouffirmansyah/modpro-infra:latest
     ```

3. Simulasi Arsitektur (Fase Prototyping)
Sebelum benar-benar membangun infrastruktur, Anda harus melakukan simulasi terlebih dahulu untuk melihat apa saja yang akan dibuat.
Jalankan perintah ini:

```
docker run --rm -v
"${PWD}/terraform.tfstate:/app/environments/dev/terraf
orm.tfstate" --env-file .env rouffirmansyah/modpro-
infra:latest plan
```

Sistem akan melakukan pengecekan ke AWS dan menampilkan daftar komponen (VPC, Subnet, EC2, RDS) yang akan dibangun. Jika tidak ada pesan kesalahan, Anda siap lanjut ke tahap berikutnya.

4. Membangun Infrastruktur (Fase Implementation)
Sekarang, saatnya membangun infrastruktur 3-Tier Anda secara otomatis. Proses ini biasanya memakan waktu 5-10 menit karena AWS perlu menyiapkan server dan basis data untuk Anda.
Jalankan perintah ini:

```bash
docker run --rm -v
"${PWD}/terraform.tfstate:/app/environments/dev/terraf
orm.tfstate" --env-file kunci.env
rouffirmansyah/modpro-infra:latest apply
```

Hasil akhir: Setelah selesai, sistem akan menampilkan alamat IP atau DNS yang bisa digunakan untuk mengakses server Anda. Infrastruktur 3-Tier Anda kini sudah aktif di AWS!

5. Menghapus Infrastruktur (Pembersihan)
Jika infrastruktur sudah tidak digunakan lagi, sangat disarankan untuk menghapusnya agar Anda tidak mendapatkan tagihan berkelanjutan dari AWS.
Jalankan perintah ini:

```bash
docker run --rm -v
"${PWD}/terraform.tfstate:/app/environments/dev/terraf
orm.tfstate" --env-file kunci.env
rouffirmansyah/modpro-infra:latest destroy

```

Hasil akhir: Semua layanan yang dibuat di Langkah 4 akan dihapus secara bersih dan otomatis dari akun AWS Anda.

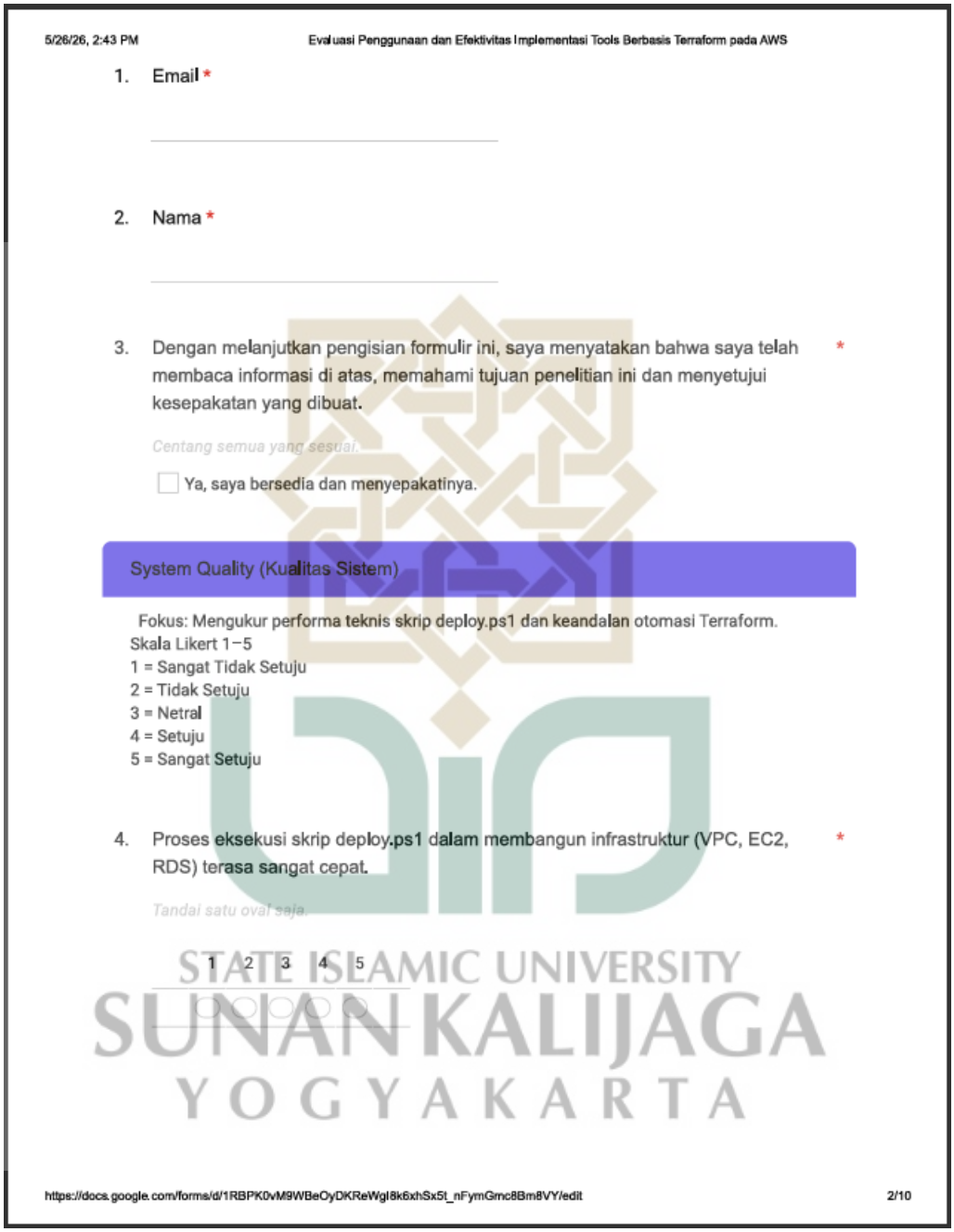
Lampiran 2 Link Video Deployment Manual

https://drive.google.com/drive/folders/1XcXjZgZ0rBe-PQ6h2NN5iYjIRc0GU68C?usp=sharing

Lampiran 3 Link Video Deployment Otomatis

https://drive.google.com/drive/folders/1P9hJJQuSVNqDuguBV6gtY5yE2uWlH-4p?usp=sharing

Lampiran 4 Instrumen Penelitian



5/26/26, 2:43 PM Evaluasi Penggunaan dan Efektivitas Implementasi Tools Berbasis Terraform pada AWS

1. Email *

2. Nama *

3. Dengan melanjutkan pengisian formulir ini, saya menyatakan bahwa saya telah membaca informasi di atas, memahami tujuan penelitian ini dan menyetujui kesepakatan yang dibuat. *

Centang semua yang sesuai.

- [ ] Ya, saya bersedia dan menyepakatinya.

System Quality (Kualitas Sistem)

Fokus: Mengukur performa teknis skrip deploy.ps1 dan keandalan otomasi Terraform.
Skala Likert 1–5
1 = Sangat Tidak Setuju
2 = Tidak Setuju
3 = Netral
4 = Setuju
5 = Sangat Setuju

4. Proses eksekusi skrip deploy.ps1 dalam membangun infrastruktur (VPC, EC2, RDS) terasa sangat cepat. *

Tandai satu oval saja.

1 2 3 4 5

https://docs.google.com/forms/d/1RBPK0vM9WBeOyDKReWgI8k6xhSx5t_nFymGmc8Bm8VY/edit 2/10

5. Otomasi Terraform ini mampu membangun sumber daya AWS secara stabil tanpa kegagalan sistem yang berarti. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

6. Sistem ini mampu secara konsisten menghasilkan konfigurasi infrastruktur yang sama setiap kali dijalankan (Idempotensi). *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

7. Fitur otomatisasi variabel (seperti input password DB) sangat memudahkan proses konfigurasi. *

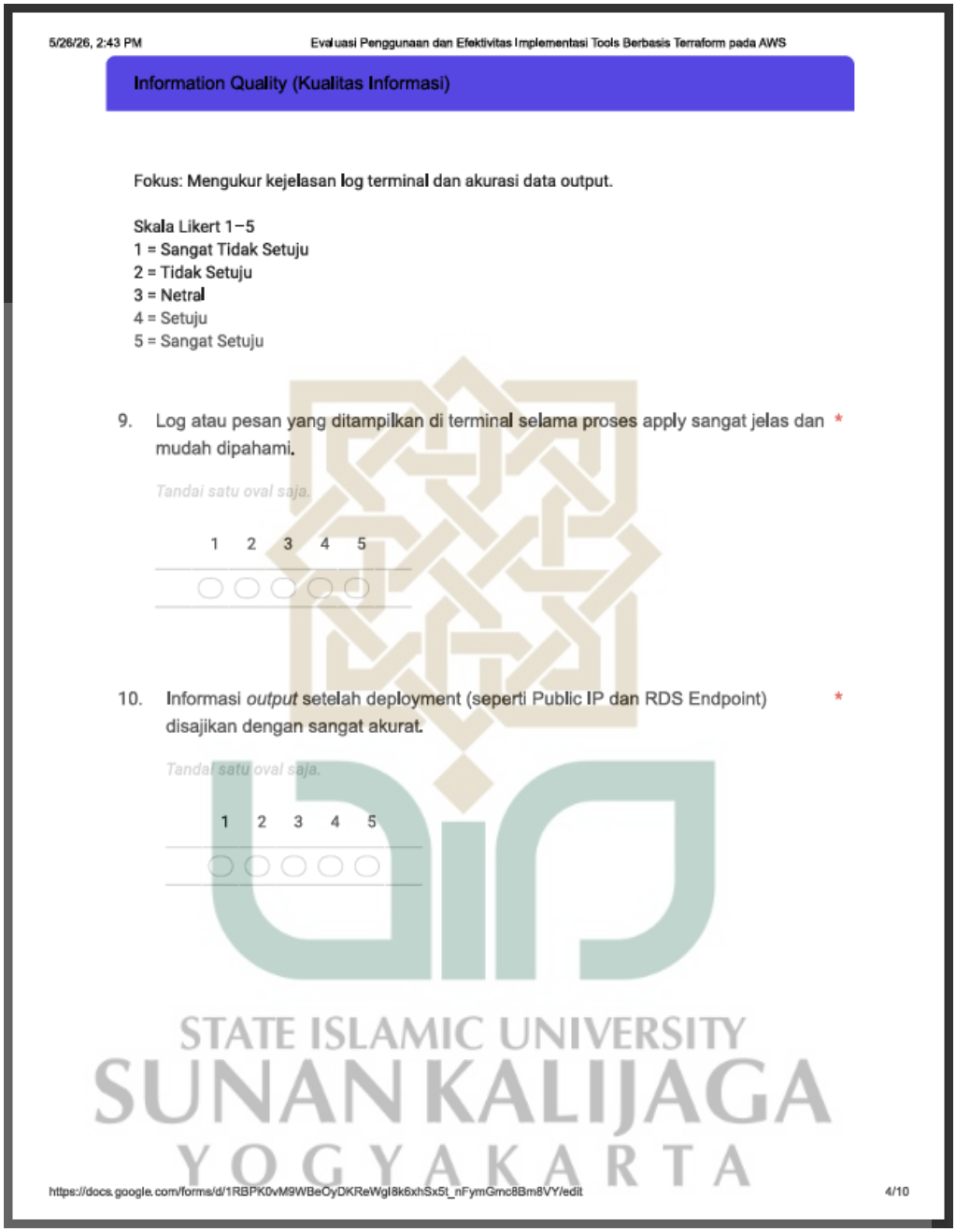
Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

8. Struktur modul Terraform yang digunakan memudahkan proses modifikasi di masa depan. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯



Information Quality (Kualitas Informasi)

Fokus: Mengukur kejelasan log terminal dan akurasi data output.

Skala Likert 1–5
1 = Sangat Tidak Setuju
2 = Tidak Setuju
3 = Netral
4 = Setuju
5 = Sangat Setuju

9. Log atau pesan yang ditampilkan di terminal selama proses apply sangat jelas dan mudah dipahami. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

10. Informasi *output* setelah deployment (seperti Public IP dan RDS Endpoint) disajikan dengan sangat akurat. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

11. Pesan kesalahan (*error message*) yang muncul memberikan instruksi yang cukup untuk melakukan *troubleshooting*. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

12. Informasi status pembangunan sumber daya (VPC, Subnet, Gateway) ditampilkan secara *real-time* dan transparan. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

13. Dokumentasi README yang disediakan memberikan instruksi yang lengkap untuk menjalankan alat ini dari awal. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

Service Quality (Kualitas Layanan)

Fokus: Mengukur kemudahan dalam mengikuti panduan dan konfigurasi standar.

Skala Likert 1–5
1 = Sangat Tidak Setuju
2 = Tidak Setuju
3 = Netral
4 = Setuju
5 = Sangat Setuju

14. Panduan langkah demi langkah untuk melakukan koneksi AWS hingga terraform apply sangat mudah diikuti. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

15. Konfigurasi standar (*standardized environment*) yang disediakan memudahkan pengguna tanpa harus pusing memikirkan pengaturan manual. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

16. Pilihan parameter (seperti engine RDS dan tipe EC2) sudah sesuai dengan standar kebutuhan lingkungan pengembangan. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

17. Dukungan dokumentasi teknis yang menyertai alat ini membantu menyelesaikan kendala saat instalasi. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

Use (Penggunaan)

Fokus: Mengukur niat responden untuk terus menggunakan otomasi ini.
Skala Likert 1–5
1 = Sangat Tidak Setuju
2 = Tidak Setuju
3 = Netral
4 = Setuju
5 = Sangat Setuju

18. Saya lebih memilih menggunakan modul Terraform ini dibandingkan melakukan klik manual di AWS Management Console. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

19. Saya berniat menggunakan alat otomasi ini untuk proyek pengembangan web saya di masa mendatang. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

20. Saya akan merekomendasikan penggunaan modul IaC ini kepada rekan sejawat untuk mempercepat pembangunan infrastruktur. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

User Satisfaction (Kepuasan Pengguna)

Fokus: Mengukur kenyamanan dan kepuasan subjektif responden.
Skala Likert 1–5
1 = Sangat Tidak Setuju
2 = Tidak Setuju
3 = Netral
4 = Setuju
5 = Sangat Setuju

21. Saya sangat puas dengan hasil akhir infrastruktur yang dibangun oleh alat ini. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

22. Menggunakan otomatisasi Terraform ini memberikan pengalaman yang jauh lebih nyaman dibandingkan cara manual. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

23. Secara keseluruhan, kinerja alat otomasi ini memenuhi ekspektasi saya untuk sebuah infrastruktur *Three-Tier*. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

Net Benefits (Manfaat Bersih)

Fokus: Mengukur dampak nyata alat ini terhadap produktivitas dan pengurangan kesalahan.
Skala Likert 1–5
1 = Sangat Tidak Setuju
2 = Tidak Setuju
3 = Netral
4 = Setuju
5 = Sangat Setuju

24. Penggunaan alat ini secara signifikan memangkas waktu pembangunan infrastruktur (dari menit menjadi detik). *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

25. Otomatisasi ini sangat efektif dalam mengurangi risiko kesalahan konfigurasi manusia (*human error*). *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

26. Keamanan infrastruktur lebih terjamin karena mengikuti standar arsitektur yang sudah ditentukan secara otomatis. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

27. Secara keseluruhan, alat ini memberikan manfaat nyata dalam proses pengerjaan proyek berbasis *cloud*. *

Tandai satu oval saja.

1	2	3	4	5
◯	◯	◯	◯	◯

Lampiran 5 Dokumentasi Hasil Kuesioner Responden

https://docs.google.com/spreadsheets/d/1bmHT194anhUKSEVhDRdL50lsVohORO-37Key3glLck4/edit?usp=sharing

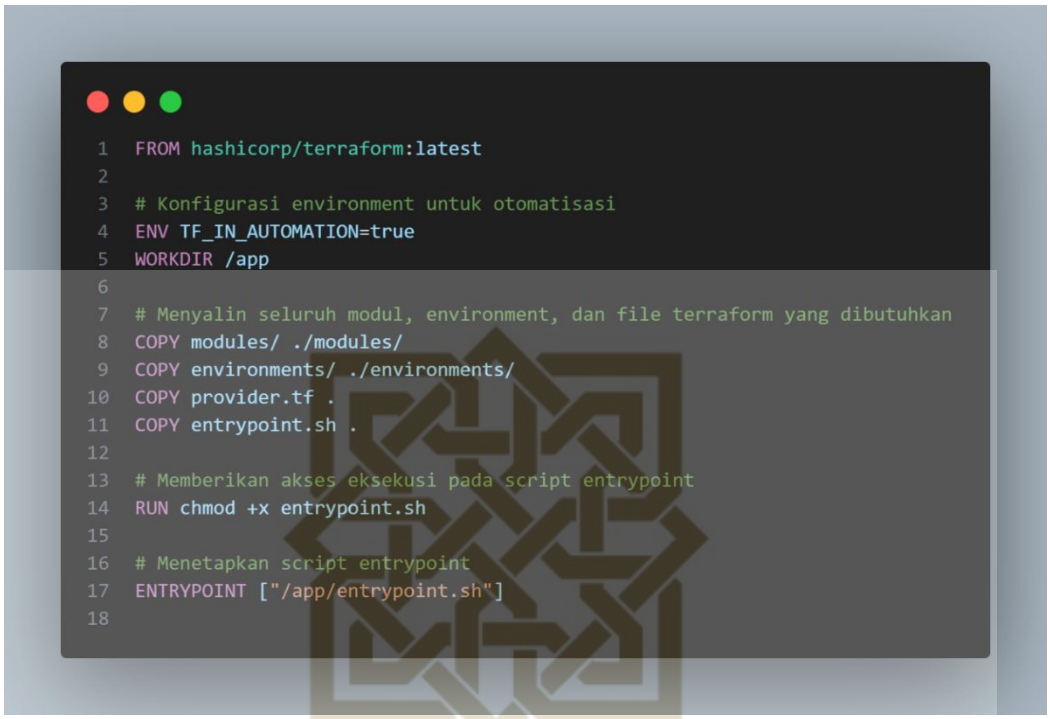
Lampiran 6 Dokumentasi Source Code Terraform dan Docker

Struktur directory proyek

iac-terraform-threetier/

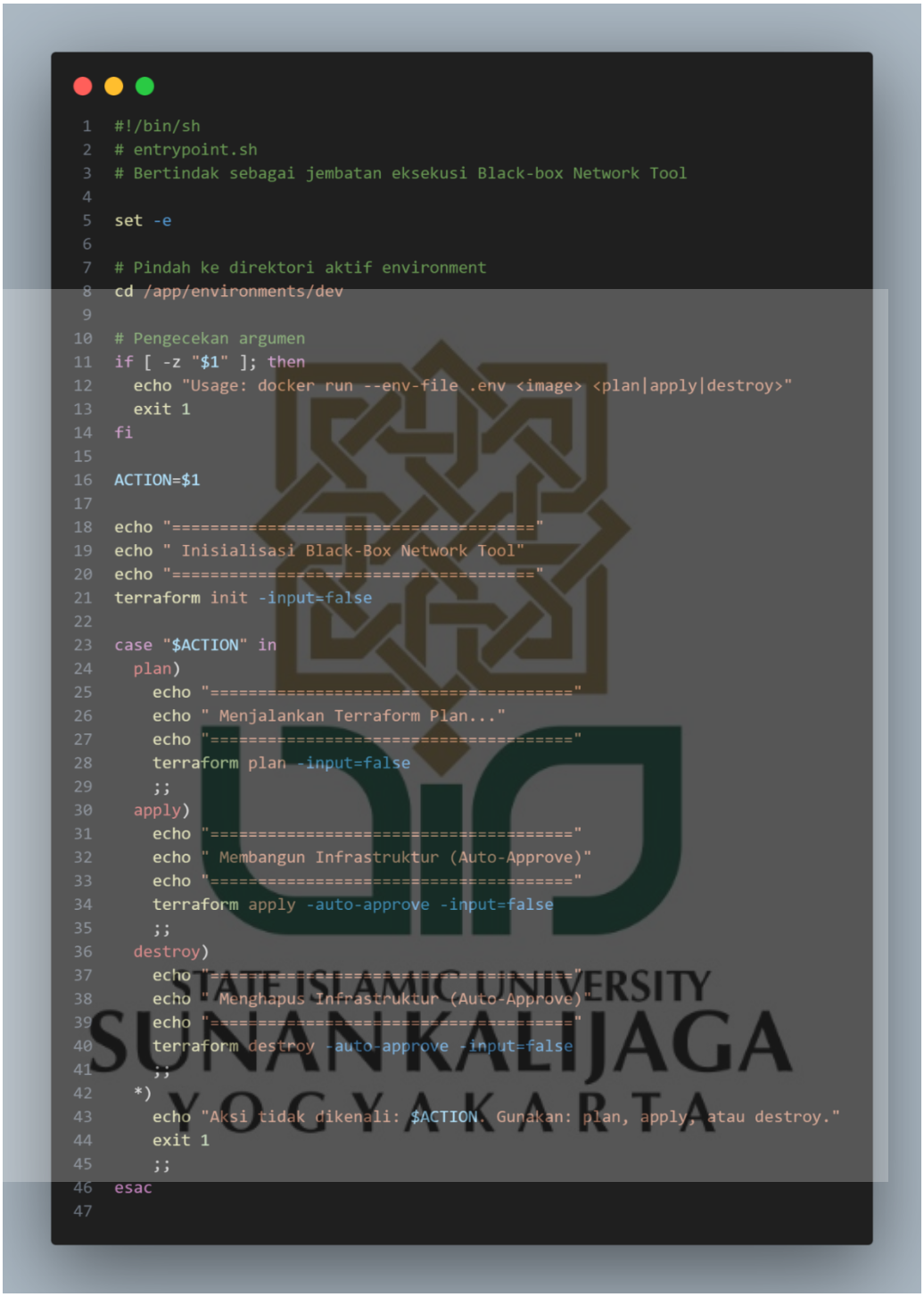
```
├─── Dockerfile
├─── entrypoint.sh
├─── provider.tf
├─── modules/
│    ├─── vpc/
│    ├─── ec2/
│    └─── rds/
└─── environments/
     └─── dev/
          ├─── main.tf
          ├─── variables.tf
          └─── outputs.tf
```

File/Direktori	Fungsi
Dockerfile	Membentuk image Docker berisi lingkungan eksekusi Terraform
entrypoint.sh	Mengatur alur eksekusi perintah plan, apply, dan destroy
provider.tf	Mendefinisikan *provider* AWS
modules/vpc	Berisi konfigurasi VPC, subnet, Internet Gateway, dan Route Table
modules/ec2	Berisi konfigurasi EC2 instance dan Security Group *web server*
modules/rds	Berisi konfigurasi RDS MySQL, DB Subnet Group, dan Security Group database
environments/dev	Memanggil seluruh modul untuk lingkungan development



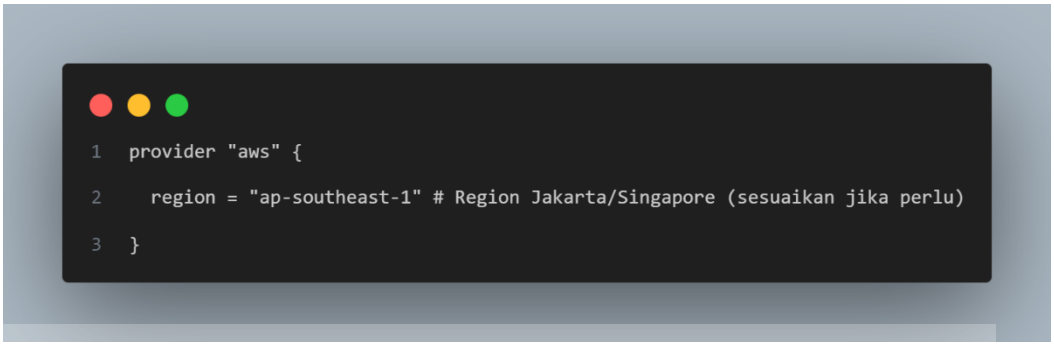
```dockerfile
FROM hashicorp/terraform:latest

# Konfigurasi environment untuk otomatisasi
ENV TF_IN_AUTOMATION=true
WORKDIR /app

# Menyalin seluruh modul, environment, dan file terraform yang dibutuhkan
COPY modules/ ./modules/
COPY environments/ ./environments/
COPY provider.tf .
COPY entrypoint.sh .

# Memberikan akses eksekusi pada script entrypoint
RUN chmod +x entrypoint.sh

# Menetapkan script entrypoint
ENTRYPOINT ["/app/entrypoint.sh"]
```

gambar lampiran 1 dockerfile



```sh
#!/bin/sh
# entrypoint.sh
# Bertindak sebagai jembatan eksekusi Black-box Network Tool

set -e

# Pindah ke direktori aktif environment
cd /app/environments/dev

# Pengecekan argumen
if [ -z "$1" ]; then
  echo "Usage: docker run --env-file .env <image> <plan|apply|destroy>"
  exit 1
fi

ACTION=$1

echo "====================================="
echo " Inisialisasi Black-Box Network Tool"
echo "====================================="
terraform init -input=false

case "$ACTION" in
  plan)
    echo "====================================="
    echo " Menjalankan Terraform Plan..."
    echo "====================================="
    terraform plan -input=false
    ;;
  apply)
    echo "====================================="
    echo " Membangun Infrastruktur (Auto-Approve)"
    echo "====================================="
    terraform apply -auto-approve -input=false
    ;;
  destroy)
    echo "====================================="
    echo " Menghapus Infrastruktur (Auto-Approve)"
    echo "====================================="
    terraform destroy -auto-approve -input=false
    ;;
  *)
    echo "Aksi tidak dikenali: $ACTION. Gunakan: plan, apply, atau destroy."
    exit 1
    ;;
esac
```


gambar lampiran 2 entrypoint.sh



```
1  provider "aws" {
2    region = "ap-southeast-1" # Region Jakarta/Singapore (sesuaikan jika perlu)
3  }
```

gambar lampiran 3 *provider*.tf

```
resource "aws_vpc" "main" {
  cidr_block           = var.vpc_cidr
  enable_dns_hostnames = true
  enable_dns_support   = true

  tags = {
    Name = "${var.project_name}-vpc"
  }
}

# Public Subnets
resource "aws_subnet" "public" {
  count                   = length(var.public_subnet_cidrs)
  vpc_id                  = aws_vpc.main.id
  cidr_block              = var.public_subnet_cidrs[count.index]
  availability_zone       = var.availability_zones[count.index]
  map_public_ip_on_launch = true

  tags = {
    Name = "${var.project_name}-public-subnet-${count.index + 1}"
  }
}

# Private Subnets
resource "aws_subnet" "private" {
  count             = length(var.private_subnet_cidrs)
  vpc_id            = aws_vpc.main.id
  cidr_block        = var.private_subnet_cidrs[count.index]
  availability_zone = var.availability_zones[count.index]

  tags = {
    Name = "${var.project_name}-private-subnet-${count.index + 1}"
  }
}

# Internet Gateway
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "${var.project_name}-igw"
  }
}

# Route Table for Public Subnets
resource "aws_route_table" "public" {
  vpc_id = aws_vpc.main.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }

  tags = {
    Name = "${var.project_name}-public-rt"
  }
}

# Route Table Association for Public Subnets
resource "aws_route_table_association" "public" {
  count          = length(var.public_subnet_cidrs)
  subnet_id      = aws_subnet.public[count.index].id
  route_table_id = aws_route_table.public.id
}
```

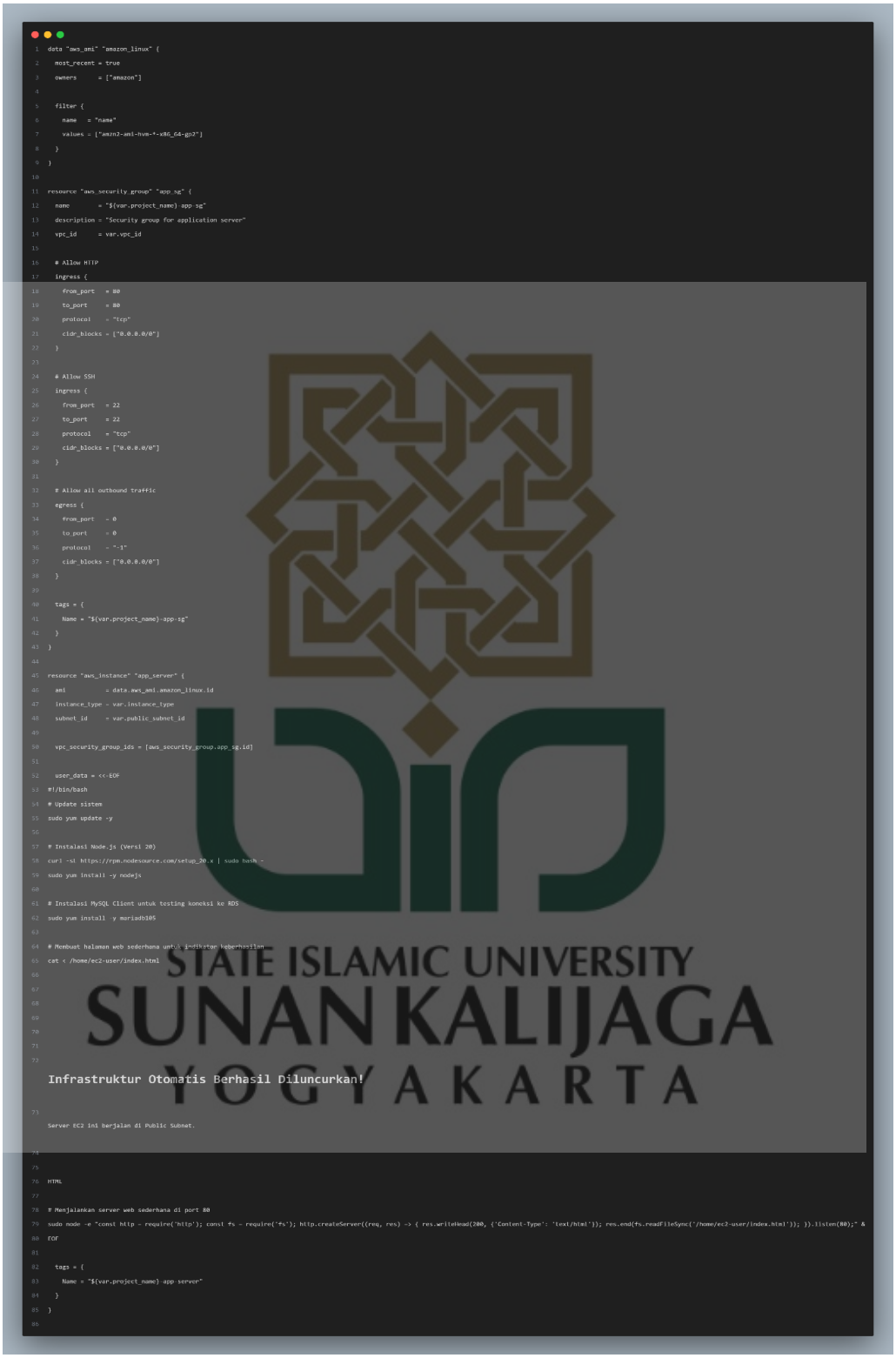
gambar lampiran 4 vpc/main.tf

```
resource "aws_vpc" "main" {
  cidr_block           = var.vpc_cidr
  enable_dns_hostnames = true
  enable_dns_support   = true

  tags = {
    Name = "${var.project_name}-vpc"
  }
}

# Public Subnets
resource "aws_subnet" "public" {
  count                   = length(var.public_subnet_cidrs)
  vpc_id                  = aws_vpc.main.id
  cidr_block              = var.public_subnet_cidrs[count.index]
  availability_zone       = var.availability_zones[count.index]
  map_public_ip_on_launch = true

  tags = {
    Name = "${var.project_name}-public-subnet-${count.index + 1}"
  }
}

# Private Subnets
resource "aws_subnet" "private" {
  count             = length(var.private_subnet_cidrs)
  vpc_id            = aws_vpc.main.id
  cidr_block        = var.private_subnet_cidrs[count.index]
  availability_zone = var.availability_zones[count.index]

  tags = {
    Name = "${var.project_name}-private-subnet-${count.index + 1}"
  }
}

# Internet Gateway
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "${var.project_name}-igw"
  }
}

# Route Table for Public Subnets
resource "aws_route_table" "public" {
  vpc_id = aws_vpc.main.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }

  tags = {
    Name = "${var.project_name}-public-rt"
  }
}

# Route Table Association for Public Subnets
resource "aws_route_table_association" "public" {
  count          = length(var.public_subnet_cidrs)
  subnet_id      = aws_subnet.public[count.index].id
  route_table_id = aws_route_table.public.id
}

```

gambar lampiran 5 vpc/ouputs.tf

```
variable "vpc_cidr" {
  description = "CIDR block untuk VPC"
  type        = string
}

variable "project_name" {
  description = "Nama proyek untuk tagging resource"
  type        = string
}

variable "public_subnet_cidrs" {
  description = "List CIDR block untuk subnet public"
  type        = list(string)
}

variable "private_subnet_cidrs" {
  description = "List CIDR block untuk subnet private"
  type        = list(string)
}

variable "availability_zones" {
  description = "List availability zones yang akan digunakan"
  type        = list(string)
}
```


gambar lampiran 6 vpc/variables.tf



```
data "aws_ami" "amazon_linux" {
  most_recent = true
  owners      = ["amazon"]

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*-x86_64-gp2"]
  }
}

resource "aws_security_group" "app_sg" {
  name        = "${var.project_name}-app-sg"
  description = "Security group for application server"
  vpc_id      = var.vpc_id

  # Allow HTTP
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Allow SSH
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Allow all outbound traffic
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "${var.project_name}-app-sg"
  }
}

resource "aws_instance" "app_server" {
  ami           = data.aws_ami.amazon_linux.id
  instance_type = var.instance_type
  subnet_id     = var.public_subnet_id

  vpc_security_group_ids = [aws_security_group.app_sg.id]

  user_data = <<-EOF
#!/bin/bash
# Update sistem
sudo yum update -y

# Instalasi Node.js (Versi 20)
curl -sL https://rpm.nodesource.com/setup_20.x | sudo bash -
sudo yum install -y nodejs

# Instalasi MySQL Client untuk testing koneksi ke RDS
sudo yum install  y mariadb105

# Membuat halaman web sederhana untuk indikator keberhasilan
cat < /home/ec2-user/index.html



  Infrastruktur Otomatis Berhasil Diluncurkan!


  Server EC2 ini berjalan di Public Subnet.



  HTML

# Menjalankan server web sederhana di port 80
sudo node -e "const http = require('http'); const fs = require('fs'); http.createServer((req, res) => { res.writeHead(200, {'Content-Type': 'text/html'}); res.end(fs.readFileSync('/home/ec2-user/index.html')); }).listen(80);" &
EOF

  tags = {
    Name = "${var.project_name}-app-server"
  }
}
```

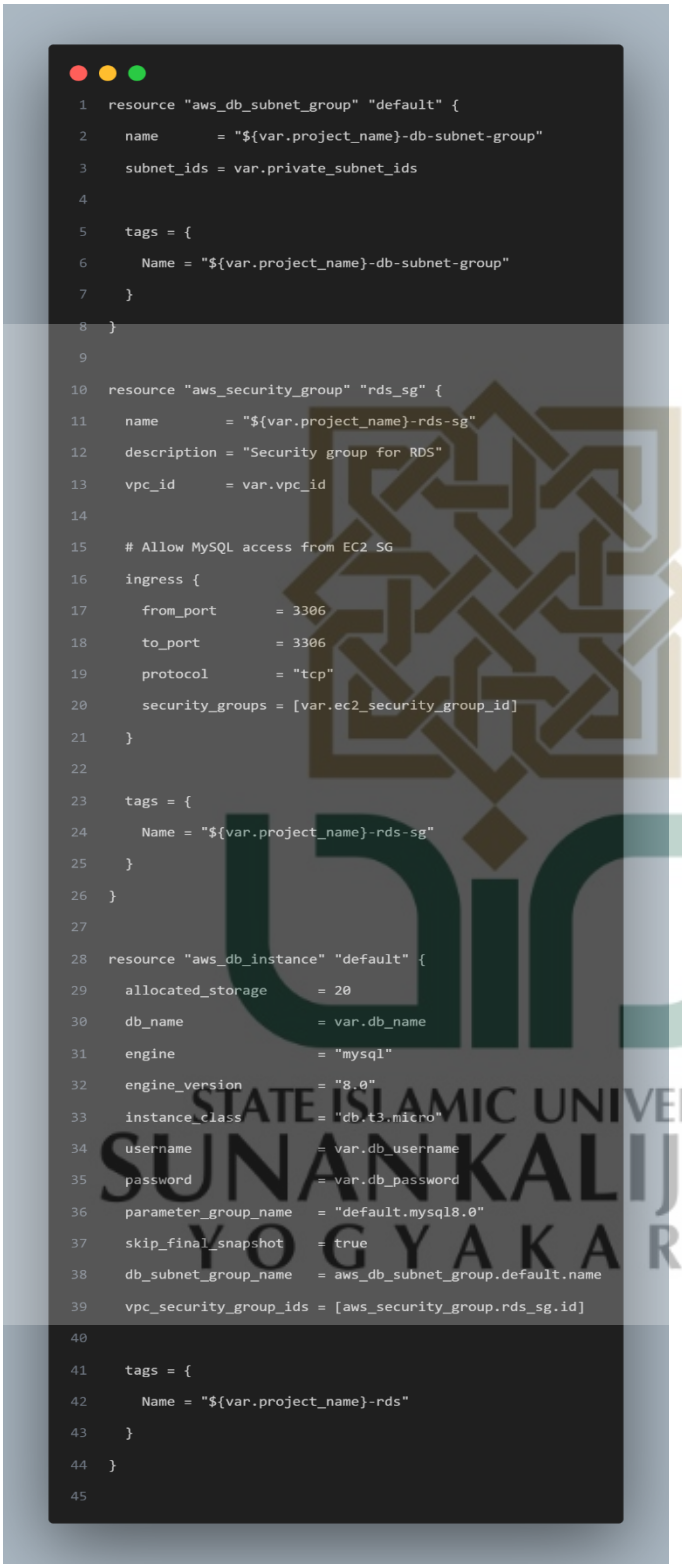

gambar lampiran 7 ec2/main.tf

```
output "instance_id" {
  description = "ID dari EC2 instance"
  value       = aws_instance.app_server.id
}

output "public_ip" {
  description = "Public IP dari EC2 instance"
  value       = aws_instance.app_server.public_ip
}

output "security_group_id" {
  description = "ID dari Security Group EC2"
  value       = aws_security_group.app_sg.id
}
```


gambar lampiran 8 ec2/outputs.tf

```
variable "vpc_id" {
  description = "ID VPC tempat EC2 akan dibuat"
  type        = string
}

variable "public_subnet_id" {
  description = "ID Subnet Publik tempat EC2 akan ditempatkan"
  type        = string
}

variable "project_name" {
  description = "Nama proyek untuk tagging"
  type        = string
}


variable "instance_type" {
  description = "Tipe instance EC2"
  type        = string
  default     = "t2.micro"
}

```


gambar lampiran 9 ec2/variables.tf

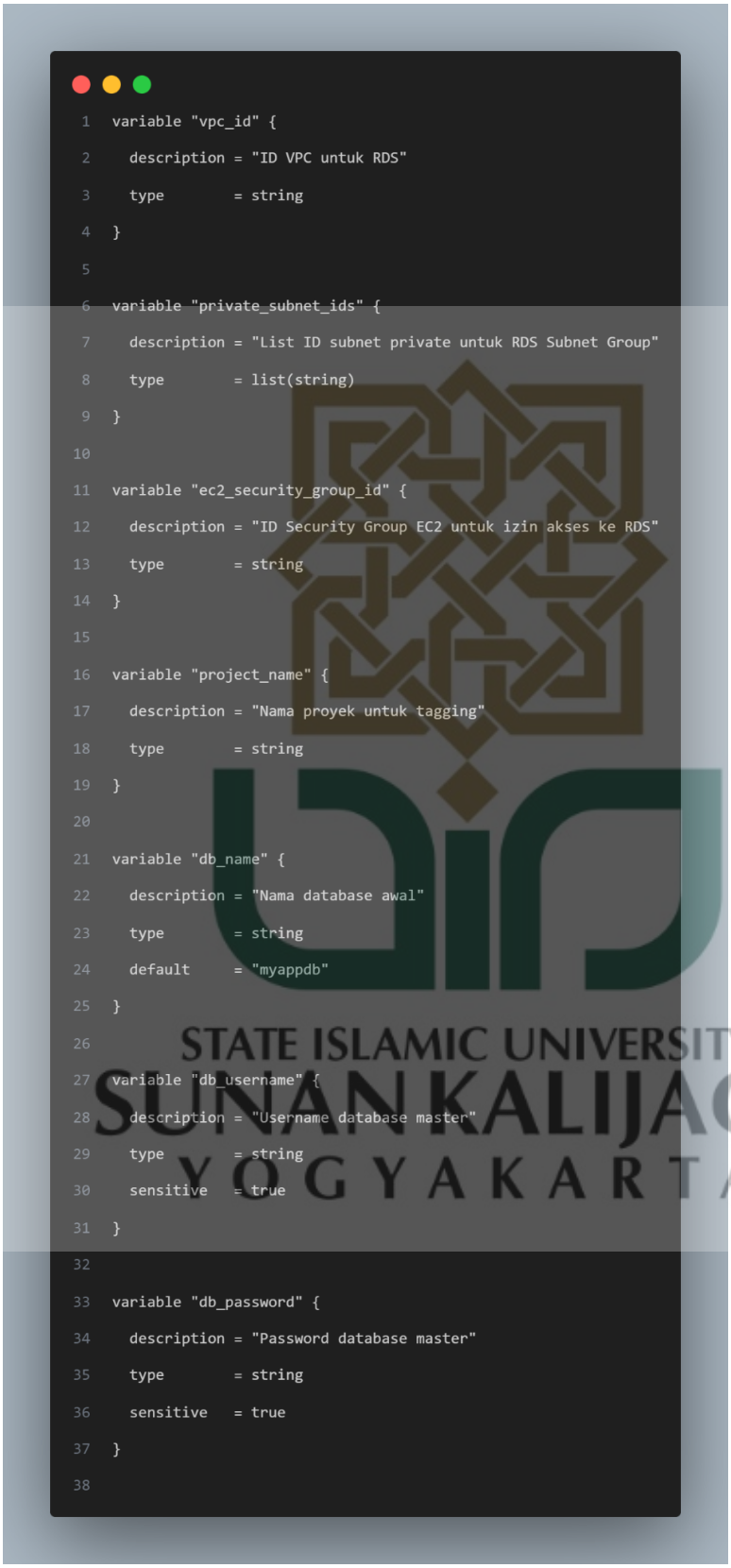


```
resource "aws_db_subnet_group" "default" {
  name       = "${var.project_name}-db-subnet-group"
  subnet_ids = var.private_subnet_ids

  tags = {
    Name = "${var.project_name}-db-subnet-group"
  }
}

resource "aws_security_group" "rds_sg" {
  name        = "${var.project_name}-rds-sg"
  description = "Security group for RDS"
  vpc_id      = var.vpc_id

  # Allow MySQL access from EC2 SG
  ingress {
    from_port       = 3306
    to_port         = 3306
    protocol        = "tcp"
    security_groups = [var.ec2_security_group_id]
  }

  tags = {
    Name = "${var.project_name}-rds-sg"
  }
}

resource "aws_db_instance" "default" {
  allocated_storage      = 20
  db_name                = var.db_name
  engine                 = "mysql"
  engine_version         = "8.0"
  instance_class         = "db.t3.micro"
  username               = var.db_username
  password               = var.db_password
  parameter_group_name   = "default.mysql8.0"
  skip_final_snapshot    = true
  db_subnet_group_name   = aws_db_subnet_group.default.name
  vpc_security_group_ids = [aws_security_group.rds_sg.id]

  tags = {
    Name = "${var.project_name}-rds"
  }
}

```


gambar lampiran 10 rds/main.tf

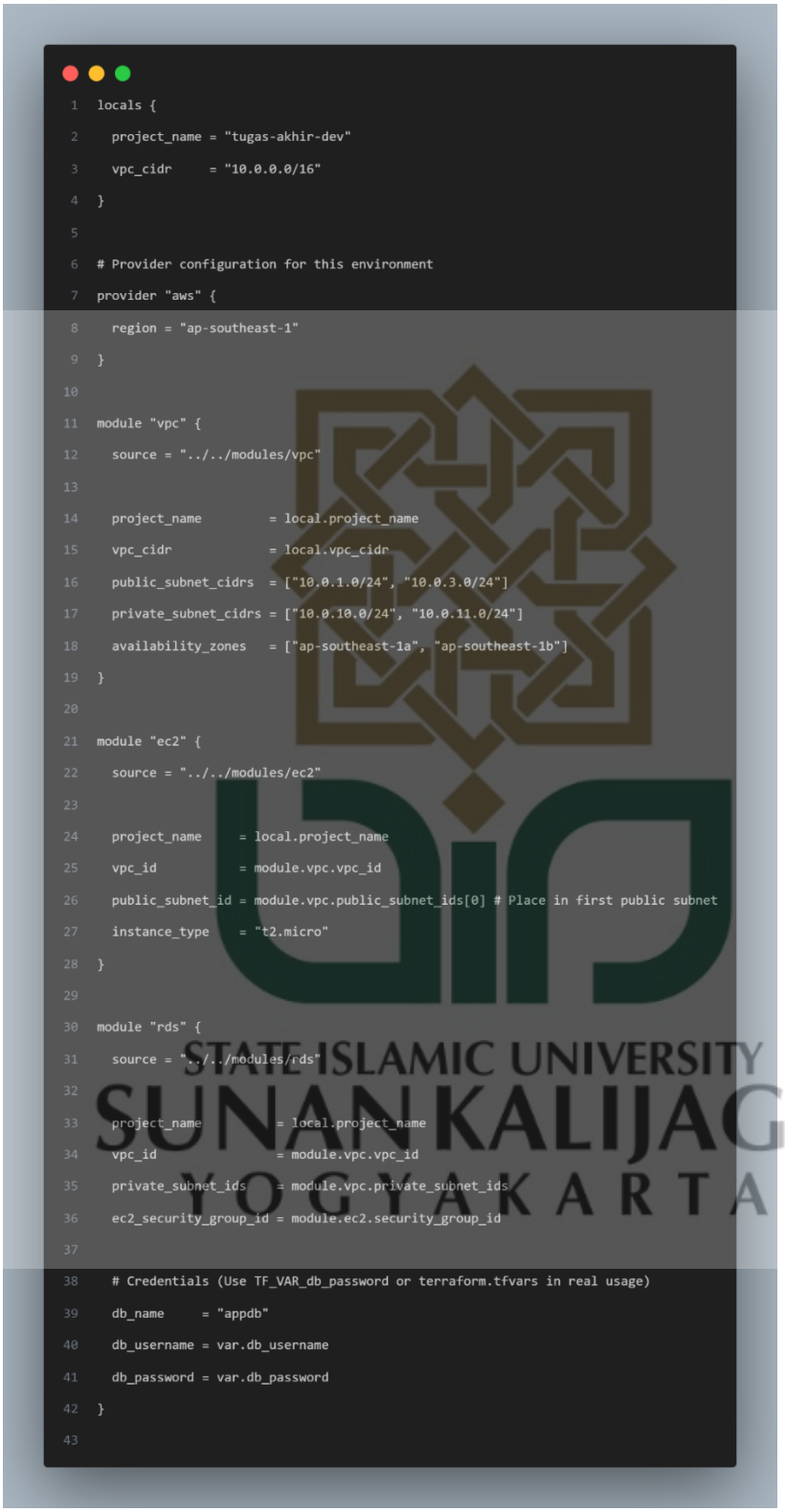
```
output "db_endpoint" {
  description = "Endpoint koneksi database"
  value       = aws_db_instance.default.endpoint
}

output "db_port" {
  description = "Port database"
  value       = aws_db_instance.default.port
}
```

gambar lampiran 11rds/outputs.tf



```
variable "vpc_id" {
  description = "ID VPC untuk RDS"
  type        = string
}

variable "private_subnet_ids" {
  description = "List ID subnet private untuk RDS Subnet Group"
  type        = list(string)
}

variable "ec2_security_group_id" {
  description = "ID Security Group EC2 untuk izin akses ke RDS"
  type        = string
}

variable "project_name" {
  description = "Nama proyek untuk tagging"
  type        = string
}

variable "db_name" {
  description = "Nama database awal"
  type        = string
  default     = "myappdb"
}

variable "db_username" {
  description = "Username database master"
  type        = string
  sensitive   = true
}

variable "db_password" {
  description = "Password database master"
  type        = string
  sensitive   = true
}
```


gambar lampiran 12 rds/variables



```hcl
locals {
  project_name = "tugas-akhir-dev"
  vpc_cidr     = "10.0.0.0/16"
}

# Provider configuration for this environment
provider "aws" {
  region = "ap-southeast-1"
}

module "vpc" {
  source = "../../modules/vpc"

  project_name         = local.project_name
  vpc_cidr             = local.vpc_cidr
  public_subnet_cidrs  = ["10.0.1.0/24", "10.0.3.0/24"]
  private_subnet_cidrs = ["10.0.10.0/24", "10.0.11.0/24"]
  availability_zones   = ["ap-southeast-1a", "ap-southeast-1b"]
}

module "ec2" {
  source = "../../modules/ec2"

  project_name     = local.project_name
  vpc_id           = module.vpc.vpc_id
  public_subnet_id = module.vpc.public_subnet_ids[0] # Place in first public subnet
  instance_type    = "t2.micro"
}

module "rds" {
  source = "../../modules/rds"

  project_name          = local.project_name
  vpc_id                = module.vpc.vpc_id
  private_subnet_ids    = module.vpc.private_subnet_ids
  ec2_security_group_id = module.ec2.security_group_id

  # Credentials (Use TF_VAR_db_password or terraform.tfvars in real usage)
  db_name     = "appdb"
  db_username = var.db_username
  db_password = var.db_password
}
```


gambar lampiran 13 environtment/dev/main.tf

```
output "Public_IP_EC2" {
  description = "Alamat IP Publik untuk mengakses Web Server"
  value       = module.ec2.public_ip
}

output "RDS_Endpoint" {
  description = "Endpoint DNS untuk koneksi database"
  value       = module.rds.db_endpoint
}

```

gambar lampiran 14 environtment/dev/outputs.tf

```
output "Public_IP_EC2" {
  description = "Alamat IP Publik untuk mengakses Web Server"
  value       = module.ec2.public_ip
}

output "RDS_Endpoint" {
  description = "Endpoint DNS untuk koneksi database"
  value       = module.rds.db_endpoint
}

```


gambar lampiran 15 environtment/dev/variables.tf

Lampiran 7 Link Docker Hub Tool Otomasi Infrastruktur

https://hub.docker.com/r/rouffirmansyah/modpro-infra

CURRICULUM VITAE (CV)

Data Diri

Nama Lengkap	: Muhammad Abdurrouf Firmansyah
NIM	: 22106050010
TTL	: Blitar, 13 Maret 2003
Jenis Kelamin	: Laki-laki
Agama	: Islam
Program Studi	: Informatika
Fakultas	: Sains dan Teknologi
Perguruan Tinggi	: Universitas Islam Negeri Sunan Kalijaga Yogyakarta

Kontak

Alamat	: Jl. Bali No. 236 Karangtengah, Kec. Sananwetan, Kota Blitar Jawa Timur
No. WA	: 0896-9704-8176
Email	: rouf.2003@gmail.com
GitHub	: https://github.com/AbdurroufFirmansyah
DockerHub	: https://hub.docker.com/repositories/rouffirmansyah
LinkedIn	: www.linkedin.com/in/muhammad-abdurrouf-firmansyah

Riwayat Pendidikan

2022-2026	: S1 Informatika, UIN Sunan Kalijaga Yogyakarta
2019-2022	: MA Ma'arif NU Kota Blitar
2016-2019	: MTs Ma'arif NU Kota Blitar
2010-2016	: SDI Kardina Massa Kota Blitar