# Implementasi *Ansible* pada Otomasi *Honeypot Deployment* Berbasis Web

Dharma Kurniawan[1], Yuris Mulya Saputra[1,*]
[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada;
dharma.k@mail.ugm.ac.id
*Korespondensi: ym.saputra@ugm.ac.id;

***Abstract – In this increasingly complex digital era, information system security has become a crucial issue for organizations across various industries. The rising number of cyberattacks and their evolving patterns pose threats to the confidentiality and integrity of an organization's data. In Indonesia, the National Cyber and Crypto Agency (BSSN) reported that as of April 2022, cyberattacks in Indonesia had reached 100 million cases, dominated by ransomware and malware attacks. One solution to address this problem is the implementation of a honeypot system. Honeypots have evolved into a valuable tool for identifying attacks and studying attacker strategies, enabling organizations to strengthen their defenses. However, manually implementing and managing honeypots can be time-consuming and resource-intensive. Therefore, this research aims to automate the honeypot deployment process using a popular configuration management tool known as Ansible. Additionally, to streamline Ansible's operation in honeypot deployment, a user-friendly web-based application has been created. This application not only deploys honeypots but also monitors the process. Based on the test results, the application successfully deployed honeypots to all sensors running on Google Cloud Platform across three different regions.***

***Keywords – Automation, Ansible, Web Application, Honeypot, Flask, Ansible AWX, Google Cloud Platform***

**Intisari – Dalam era digital yang semakin kompleks ini, keamanan sistem informasi menjadi masalah penting bagi organisasi di berbagai industri. Peningkatan serangan siber serta polanya yang semakin bervariatif bisa menimbulkan ancaman bagi kerahasiaan data atau integritas dari suatu organisasi. Di Indonesia, Badan Siber dan Sandi Negara (BSSN) mencatat bahwa hingga sepanjang bulan April 2022, serangan siber di Indonesia telah mencapai angka 100 juta kasus dengan jenis serangan yang didominasi oleh serangan *ransomware* dan *malware*. Salah satu solusi untuk mengatasi permasalahan ini adalah dengan mengimplementasikan sistem *honeypot*. *Honeypot* telah berkembang menjadi salah satu alat yang berguna untuk mengidentifikasi serangan dan mempelajari strategi penyerang, yang memungkinkan organisasi untuk memperkuat pertahanan mereka. Namun, penerapan dan pengelolaan *honeypot* secara manual dapat memakan banyak waktu dan sumber daya. Oleh karena itu, penelitian ini bertujuan untuk melakukan otomasi proses *honeypot deployment* dengan menggunakan alat manajemen konfigurasi yang populer yang dikenal sebagai *Ansible*. Selain itu, untuk mempermudah pengoperasian *Ansible* dalam melakukan *honeypot deployment*, maka dibuat sebuah aplikasi *user friendly* berbasis web. Aplikasi tidak hanya sekedar melakukan *honeypot deployment*, namun juga memantau prosesnya. Berdasarkan hasil pengujian, aplikasi berhasil melakukan *honeypot deployment* ke semua sensor yang berjalan di *Google Cloud Platform* dengan tiga *region* berbeda.**

**Kata kunci – Otomasi, *Ansible*, Aplikasi Web, *Honeypot, Flask, Ansible AWX, Google Cloud Platform***

## I. PENDAHULUAN

Dalam era digital saat ini yang semakin berkembang pesat, keamanan sistem informasi merupakan salah satu isu yang menjadi perhatian penuh di berbagai sektor. Keamanan sistem informasi merupakan bidang yang mengacu pada perlindungan sistem informasi dari ancaman yang tidak disengaja atau disengaja selama proses normal operasinya, serta dari upaya untuk mencuri, mengubah, atau menghancurkan bagian-bagiannya [1]. Peningkatan serangan siber serta polanya yang semakin kompleks bisa menimbulkan ancaman bagi kerahasiaan data atau integritas dari suatu organisasi. Di Indonesia, Badan Siber dan Sandi Negara (BSSN) mencatat bahwa hingga sepanjang bulan April 2022, serangan siber di Indonesia telah mencapai angka 100 juta kasus dengan jenis serangan yang didominasi oleh serangan *ransomware* dan *malware* [2]. Menurut Naik, bentuk serangan siber beragam jenisnya, di antaranya *malware, phising, password attack, man-in-the-middle attack, SQL injection, DoS, crypto-jacking,* dan ancaman dari dalam organisasi itu sendiri [3]. Faktor dari terjadinya serangan siber seperti mengambil keuntungan dari data yang dieksploitasi dan yang paling utama adalah lemahnya sistem keamanan pada *server* [4]. Efek yang ditimbulkan bukanlah sesuatu yang bisa diremehkan, seperti kerugian finansial, kerugian reputasi, bahkan kehilangan data berharga tentu sangatlah mengerikan. Maka dari itu, untuk menciptakan sistem keamanan terbaik, organisasi harus memiliki mekanisme pengamanan yang efektif dalam mendeteksi, mengidentifikasi, serta mampu merespons serangan tersebut. *Honeypot* merupakan salah satu inovasi dalam dunia keamanan siber yang diciptakan untuk memancing penyerang serta merekam semua aktivitas mereka, kemudian data tersebut bisa diteliti dan dipelajari, sehingga strategi baru bisa diterapkan untuk memperkuat pertahanan sistem. Menurut Verma & Dubey, *honeypot* adalah sistem berwujud *virtual machine* yang menyimulasikan *service* atau *port* yang terbuka secara disengaja agar memiliki potensi untuk diserang dan diretas [5]. *Honeypot* memiliki cara kerja dengan menyerupai sistem atau layanan, sehingga mengalihkan perhatian penyerang agar lebih tertarik berinteraksi dengan *honeypot* tersebut. Di saat yang sama, *honeypot* akan mengambil data penyerang seperti aktivitas, teknik, serta alat yang digunakan. Namun, instalasi *honeypot* serta manajemen yang masih dilakukan secara manual adalah tindakan yang kurang efisien dan memakan cukup banyak waktu. Terlebih, sebagian besar organisasi menerapkan infrastruktur yang kompleks.

Banyaknya mesin dan sistem untuk menjalankan *honeypot* ditambah konfigurasi yang beragam serta pengelolaan *honeypot* secara individu di setiap mesinnya menjadi tugas yang cukup merepotkan.

Berbagai permasalahan yang muncul ini kemudian disimpulkan menjadi beberapa poin, antara lain konfigurasi yang mulai kompleks, konsumsi waktu dan daya yang besar, konsistensi yang sulit dipertahankan, skalabilitas terbatas, dan kurang fleksibel dalam merespons perubahan. Dari banyaknya poin yang disampaikan, proses *honeypot deployment* yang masih dilakukan secara manual secara tidak langsung dapat memberikan efek yang cukup merugikan bagi pihak organisasi dari segi waktu hingga finansial. Maka dari itu, diperlukan setidaknya sistem atau alat yang mampu untuk mengeliminasi beberapa permasalahan akibat dari proses *honeypot deployment* yang masih manual.

*Ansible* merupakan salah satu alat pengelolaan konfigurasi yang populer dan efisien dalam menjalankan tugas-tugas manajemen sistem secara otomatis. Menurut Lovdianchel, *Ansible* adalah *tools open source* yang menerapkan konsep *infrastructure as a code* untuk melakukan berbagai tugas seperti *provisioning server*, *configuration management*, dan *deployment* aplikasi secara terotomasi [6]. *Ansible* menggunakan bahasa yang mudah untuk dipahami manusia, sehingga admin sistem dapat membuat konfigurasi sesuai dengan kebutuhan. Setelah itu, *Ansible* akan menjalankan konfigurasi tersebut di seluruh infrastruktur secara konsisten. Dengan memanfaatkan *Ansible*, admin sistem akan lebih mudah dalam memasang *honeypot* di berbagai mesin dan konfigurasi yang telah didefinisikan seragam dan konsisten.

Tujuan dari penelitian ini adalah untuk mengisi ketimpangan pengetahuan mengenai *honeypot deployment* secara efektif dan efisien menggunakan *Ansible*. Selain itu, dengan penambahan fitur *dashboard* dapat memudahkan admin dalam mengelola dan juga memantau proses *honeypot deployment*. Dengan penelitian ini, diharapkan hasil yang didapat mampu berperan penting dalam pengembangan praktik terbaik dan menjadi pedoman bagi para praktisi keamanan untuk mengotomasi *honeypot deployment* dan meningkatkan pertahanan sistem dari serangan siber.

## II. DASAR TEORI

### A. Penelitian Sebelumnya

Penelitian dengan judul *"Securing a Small Network by using Raspberry Pi Honeypot"* membahas tentang penggunaan *honeypot* dengan memanfaatkan perangkat *Raspberry Pi* untuk memantau serta mengamankan jaringan. Dari penelitian tersebut menghasilkan kesimpulan bahwa penggunaan *honeypot* yang dikombinasikan dengan *Raspberry Pi* merupakan cara efisien serta mampu menekan biaya untuk membangun sistem pemantauan dan pengamanan jaringan rumah atau bisnis level bawah [7]. Selain itu, penelitian sejenis dengan judul *"Dynamic Honeypot Deployment in the Cloud"* membahas tentang konsep baru *honeypot deployment* di lingkungan *cloud* untuk mengatasi keterbatasan dari sistem yang ada saat ini. Dari penelitian tersebut menghasilkan kesimpulan bahwa sistem yang diusulkan mencapai otomasi dalam menyediakan serta menjalankan *honeypot* dengan menggunakan salah satu *service* pada *cloud AWS* yaitu *AWS Lambda* [8].

Penelitian dengan judul *"Infrastructure as Code for Security Automation and Network Infrastructure Monitoring"* membahas tentang pemanfaatan konsep *infrastructure as code* untuk mempermudah dan membuat pekerjaan pada pengelolaan infrastruktur jaringan menjadi lebih cepat dan efisien. Selain itu, penelitian ini mengusulkan penggunaan *Ansible* sebagai *tools* otomasi untuk melakukan berbagai pekerjaan seperti membuat *virtual machine*, *intrusion detection system*, *honeypot*, dan pengelolaan *security information* dan *event* agar dapat terhubung pada infrastruktur jaringan secara efektif [9]. Selain itu, terdapat penelitian sejenis dengan judul *"Implementation of Devops Method for Automation of Server Management Using Ansible"* membahas tentang pemanfaatan metode *DevOps* yaitu suatu konsep pengembangan dan penyampaian perangkat lunak ke infrastruktur dengan mengambil pendekatan secara kolaboratif dan integratif antara *developer* dan *software operation*. Selain itu, penelitian ini juga mengusulkan penggunaan *Ansible* dengan metode *DevOps* untuk melakukan otomasi pengelolaan banyak *server* [10].

Penelitian dengan judul *"Creation of a High-Interaction Honeypot System based-on Docker Containers"* membahas tentang pembuatan sistem *high-interaction honeypot* dengan memanfaatkan *Docker* sebagai wadah untuk menampung *honeypot* serta mendeteksi serangan pada *network-level* dan *host-level*. Hasil dari penelitian ini adalah sistem *honeypot* berbasis *Docker* yang lebih sulit untuk dideteksi dan diterapkan pada sistem *Linux* [11]. Selain itu, terdapat penelitian sejenis dengan judul *"Containerized cloud-based honeypot deception for tracking attackers"* membahas tentang pengenalan konsep *honeypot* serta *honeypot deployment* secara *container-based* di *cloud* untuk mencapai sistem yang portabel, kuat, dan mudah dalam instalasi dan pengelolaannya. Penelitian ini memberikan kesimpulan bahwa sistem yang dibuat secara efektif mampu mengumpulkan lebih banyak informasi serangan, namun masih terbatas pada tipe ancaman yang bisa dideteksi [12].

Penelitian dengan judul "Implementasi *Flask Framework* pada Pembangunan Aplikasi Sistem Informasi *Helpdesk (SIH)"* membahas tentang pengembangan aplikasi berbasis web dengan memanfaatkan *framework Flask* untuk pembuatan *API*, *Bootstrap* untuk *framework* pembuatan *UI*, dan *PostgreSQL* untuk penyimpanan *database*. Penelitian ini memberikan kesimpulan bahwa implementasi *framework Flask* pada pengembangan aplikasi berbasis web dapat meningkatkan efisiensi dan efektivitas pada proses bisnis [13]. Selain itu, terdapat penelitian sejenis dengan judul "Implementasi *Flask Framework* pada Pembangunan Aplikasi *Purchasing Approval Request"* membahas tentang pengembangan aplikasi dengan memanfaatkan *framework Flask*. Dari penelitian tersebut menghasilkan kesimpulan

bahwa penggunaan *framework Flask* pada aplikasi mampu meminimalisasi *error* pada aplikasi, menyederhanakan proses pembuatan aplikasi, dan mempercepat komunikasi antara *backend* dan *frontend* aplikasi [14].

Penelitian dengan judul *"*Penerapan *React JS* pada Pengembangan *FrontEnd* Aplikasi *Startup Ubaform"* membahas tentang pengembangan aplikasi berbasis dengan memanfaatkan *React JS* untuk pengembangan tampilan *UI* atau *frontend* pada web. Dari penelitian tersebut menghasilkan kesimpulan bahwa pengembangan *frontend* dengan memanfaatkan *React JS* mampu menghemat waktu serta mempermudah pengelolaan karena setiap bagian pada tampilan web dikemas menjadi *component,* sehingga mempermudah dalam proses *debugging* [15].

Berdasarkan penelitian sebelumnya, otomasi *honeypot deployment* sudah pernah dilakukan dengan alat dan metode yang beragam. Namun, penelitian tentang pemanfaatan *Ansible* yang difokuskan untuk *honeypot deployment* masih terbatas. Selain itu, implementasi *Ansible* masih berbasis *Command Line Interface (CLI)*, belum menerapkan penggunaan berbasis *API* atau web seperti salah satunya menggunakan *AWX Ansible*. Penelitian ini dimaksudkan untuk mengimplementasikan *Ansible* untuk melakukan otomasi *honeypot deployment* yang diintegrasikan dengan aplikasi web. Selain itu, beberapa pengujian juga dilakukan untuk memastikan aplikasi web berfungsi dengan baik serta mengetahui *honeypot* berhasil menerima *traffic* dari *port* yang diekspos.

B. *Honeypot*

*Honeypot* merupakan sistem yang terhubung ke jaringan berfungsi sebagai umpan untuk memikat penyerang dunia maya dan membantu mereka mendeteksi, mengalihkan, dan mempelajari upaya peretasan untuk mendapatkan akses tidak sah ke sistem informasi. Menurut Titarmare dkk., menyatakan bahwa *honeypot* adalah sistem tiruan yang dirancang untuk mengalihkan penyerang dari sistem yang dilindungi dan mendapatkan informasi tentang aktivitas mencurigakan dari mereka, serta dilengkapi dengan kemampuan untuk memantau dan menyimpan semua *logs* aktivitas yang terjadi pada *honeypot* [16]. *Honeypot* menampilkan dirinya sebagai target potensial penyerang di internet, biasanya *server* atau data berharga lainnya, dan mengumpulkan informasi dan memberi tahu korban tentang segala upaya untuk mendapatkan akses tidak sah ke sistem informasi. Sistem *honeypot* sering bekerja dengan sistem operasi yang diperkuat, yang memiliki langkah-langkah keamanan tambahan untuk meminimalkan potensi bahaya. Biasanya, mereka dikonfigurasi dengan kerentanan yang disengaja sehingga dapat dimanfaatkan oleh penyerang. Misalnya, seperti membuka dan mengekspos *port* pada protokol yang sering diincar oleh penyerang contohnya *SMB*, *FTP*, *SSH*, *RDP*, dsb.

C. *DevOps*

*DevOps* adalah praktik manajemen infrastruktur dan pengembangan perangkat lunak yang bertujuan untuk mengintegrasikan tim pengembangan dan tim operasi dalam siklus pengembangan perangkat lunak. Tujuan utama *DevOps* adalah untuk meningkatkan kualitas perangkat lunak, mengurangi risiko dan biaya dalam pengembangan dan pengoperasian sistem, dan mempercepat pengiriman perangkat lunak ke produksi.

*DevOps* mengintegrasikan budaya, praktik, dan sumber daya untuk mencapai tujuan ini. Beberapa konsep dan praktik utama *DevOps* termasuk:

1. Otomasi: Mengurangi kesalahan manusia dan mempercepat rilis perangkat lunak dengan mengotomasikan proses pengujian, integrasi, pengiriman, dan pengoperasian.
2. Kolaborasi: Mendorong tim operasi, pengembangan, dan tim lain yang terlibat dalam siklus pengembangan, seperti *QA (Quality Assurance)*, untuk bekerja sama lebih erat.
3. Pengelolaan Konfigurasi: Menggunakan alat otomatis untuk mengelola konfigurasi sistem dan aplikasi untuk memastikan konsistensi dan skalabilitas.
4. Orkestrasi: Menggunakan alat orkestrasi untuk mengelola dan mengotomasikan proses yang kompleks seperti penyebaran perangkat lunak.
5. Kontainerisasi: Menggunakan teknologi seperti *Docker* untuk menggabungkan aplikasi dan komponennya ke dalam wadah yang dapat digunakan di berbagai tempat.
6. *Infrastructure as Code* (IaC): Mendefinisikan infrastruktur dengan kode yang memungkinkan untuk membuat, mengelola, dan merespons perubahan infrastruktur dengan cara yang seragam dan otomatis.

Beberapa alat dan teknologi yang digunakan oleh *DevOps* adalah sebagai berikut:

1. *Docker* adalah *platform* kontainer yang memungkinkan aplikasi berjalan secara konsisten.
2. *Kubernetes* adalah *orchestrator* kontainer yang digunakan untuk mengelola dan merencanakan aplikasi berbasis kontainer dalam proses produksi.
3. *Ansible* adalah alat untuk otomatisasi konfigurasi dan manajemen yang membantu menjalankan infrastruktur dengan kode.
4. *Git* adalah sistem kontrol versi yang bertanggung jawab atas kode sumber aplikasi.

D. *Ansible*

*Ansible* adalah alat *deployment* aplikasi, manajemen konfigurasi, dan penyediaan perangkat lunak berbasis *open source*. *Ansible* berjalan pada banyak sistem *Unix* dan memiliki kemampuan untuk mengkonfigurasi sistem *Unix* dan *Microsoft Windows*. Untuk mengonfigurasi *server* jarak jauh, *Ansible* menggunakan protokol *SSH* dan mengirimkan *command* melalui koneksi *SSH* [17]. Berdasarkan penelitian yang dilakukan oleh Elradi, menyatakan bahwa pekerjaan yang diotomasi menggunakan *Ansible* mampu mempercepat penyelesaian tugas hingga 70% lebih cepat jika dibandingkan tanpa menggunakan otomasi [18]. Selain itu, *Ansible* menggunakan bahasa yang mudah untuk dibaca manusia, yaitu *YAML* untuk mendefinisikan tugas yang akan dieksekusi.

Menurut salah satu artikel oleh Maha, *Ansible* memiliki beberapa komponen di antaranya [19]:

1. *Inventory*, berisi alamat *IP* dari setiap perangkat yang dikelola.
2. *Modules*, sebuah *package* yang berisi unit kode untuk dieksekusi serta memiliki fungsi yang spesifik.
3. *Ad-hoc*, baris perintah untuk menjalankan *task* melalui *Command Line Interface* atau *CLI*.
4. *Playbooks*, sebuah *file* dengan format *YAML* yang berisi serangkaian *tasks* yang dieksekusi secara urut.

Selain itu, terdapat komponen lain seperti *roles* yaitu suatu metode untuk memecah kompleksitas *playbook* menjadi lebih kecil yang direpresentasikan ke dalam bentuk komponen modular yang di dalamnya memuat gabungan dari beberapa konten dengan fungsi spesifik.

E. *Ansible AWX*

Menurut Freeman dan Keating, *Ansible AWX* merupakan proyek *upstream open source* dari versi komersial *Ansible Tower* yang dikembangkan oleh *Red Hat* dan memiliki fitur yang sama dengan *Ansible Tower*, namun tanpa *support* atau versi *stable* dari *Red Hat* [20]. *Ansible AWX* mengadopsi teknologi *Ansible* dengan fitur tambahan berbasis web dan *API* untuk mengelola *playbook, inventory, credential*, dan *resources* lainnya. *Ansible AWX* menyediakan *API* yang bisa dikombinasikan dengan aplikasi lain, sehingga berbagai macam aplikasi bisa dikembangkan dengan memanfaatkan alat ini.

F. *Flask*

*Flask* merupakan *framework Python* yang dimanfaatkan dalam pembuatan web. *Flask* seringkali disebut sebagai *micro framework* karena tidak membutuhkan alat atau *library* tertentu. Menurut Steffi, *micro framework* merupakan versi minimal dari sebuah *framework* yang di dalamnya berisi kode dengan fungsi tertentu dan dibangun pada *platform* tertentu juga [21]. Sehingga jika ingin menambahkan fungsi-fungsi lainnya seperti integrasi ke *database*, validasi *form*, dan sebagainya membutuhkan modul pihak ketiga. Namun, sekarang sudah banyak modul yang memang dikembangkan untuk menunjang fungsionalitas *Flask* seolah-olah modul tersebut diimplementasikan oleh *Flask* sendiri.

G. *React*

Menurut Boduch dan Derks, *React* atau *React JS* merupakan *library* yang dimanfaatkan dalam pembuatan tampilan pengguna atau *user interface* secara interaktif pada sebuah aplikasi [22]. Dengan menggunakan *React*, pengembang dapat mengelompokkan *UI* menjadi beberapa potongan kode yang disebut sebagai *component*. *Component* ini bersifat independen dan dapat digunakan kembali sesuai kebutuhan. *React* ini bukanlah *framework* karena fungsi utamanya hanya untuk memuat *component* pada tampilan aplikasi.

H. *Docker*

*Docker* merupakan *platform* layanan yang menggunakan teknologi virtualisasi pada level *OS* untuk menghasilkan perangkat lunak yang dikemas dalam paket dengan *environment* terisolasi yang disebut sebagai *container* [23]. *Container* ini bersifat isolatif antara satu dengan yang lain serta menggabungkan perangkat lunak mereka sendiri, *library*, beserta berkas konfigurasi. *Container* juga mampu berkomunikasi satu sama lain dan bahkan bisa juga ke luar mesin *host* dengan konfigurasi tertentu.

I. *Kubernetes*

*Kubernetes* atau biasa disebut *K8s* merupakan *platform* orkestrasi *open source* yang ditujukan untuk otomasi *deployment*, pengelolaan beban kerja, penyediaan konfigurasi, dan pengelolaan pada aplikasi *container* [24]. Pada level produksi, *Kubernetes* sangat dibutuhkan untuk memastikan *container* tetap berjalan dan meminimalisasi waktu *downtime*. Selain itu, *Kubernetes* juga menyediakan fitur-fitur lain seperti *load balancing*, pembaruan dan *scaling* aplikasi, dan otomasi *rollback* dan *rollout container* [25].

J. *Google Cloud Platform*

*Google Cloud Platform* atau biasa disingkat *GCP* merupakan salah satu penyedia terkemuka layanan *cloud computing* yang ditawarkan oleh *Google*. *Cloud computing*, penyimpanan, *cloud database*, analitik, pembelajaran mesin, kecerdasan buatan, *Internet of Things (IoT)*, dan lainnya adalah beberapa layanan infrastruktur *cloud* yang disediakan oleh *Google Cloud Platform*.

Beberapa layanan yang tersedia di *GCP* di antaranya sebagai berikut:

1. *Google Compute Engine* adalah layanan *cloud computing* yang memungkinkan *user* membuat dan mengelola mesin virtual di infrastruktur *Google*, yang memungkinkan *user* menjalankan berbagai jenis tugas.
2. *Google Cloud Storage* adalah layanan penyimpanan *cloud* yang memungkinkan *user* menyimpan dan mengelola data secara skalabilitas dengan berbagai jenis penyimpanan,

seperti penyimpanan objek dan penyimpanan berbasis blok, dan lainnya.

3. *Google BigQuery* adalah layanan analitik data yang memungkinkan analisis data secara cepat dan skalabel.
4. *Google Cloud SQL* adalah layanan sistem *database* yang mendukung *database SQL Server*, *MySQL*, dan *PostgreSQL*.

## III. METODOLOGI

### A. Bahan

Beberapa perangkat lunak dan *framework* yang dibutuhkan dalam penelitian ini adalah sebagai berikut:

1. Sistem Operasi *Windows 10*, yang akan digunakan pada perangkat admin.
2. Sistem Operasi *Ubuntu Server 20.04 LTS*, yang akan digunakan pada *VPS*.
3. *Ansible*, sebagai alat untuk otomasi *honeypot deployment*.
4. *Ansible AWX*, sebagai alat untuk menyediakan *API* dari *Ansible*.
5. *Minikube*, sebagai alat untuk menjalankan *Kubernetes single-node*.
6. *Flask*, sebagai *framework* untuk pembuatan *API*.
7. *React JS*, sebagai *library* untuk pembuatan *UI dashboard*.
8. Beberapa *honeypot* yang sudah di-*build* menggunakan *Docker* (*Dionaea*, *Cowrie*, *Gridpot*, *Honeytrap*, *Elasticpot*, *RDPY*).
9. *Python*, sebagai bahasa pemrograman untuk pembuatan *API*.
10. *PostgreSQL*, sebagai sistem penyimpanan *database*.
11. *Google Cloud Platform (GCP)*, sebagai *cloud environment* untuk pengujian *honeypot deployment*.
12. *Google Compute Engine*, sebagai sensor berwujud *VM* yang berjalan pada *cloud GCP*.

### B. Peralatan

Beberapa peralatan yang digunakan pada penelitian ini di antaranya sebagai berikut:

1. Laptop

   Laptop yang digunakan dalam penelitian ini memiliki spesifikasi disajikan pada Tabel 1.

Tabel 1. Spesifikasi laptop

| Sistem Operasi | Memori | CPU | *Storage* |
|---|---|---|---|
| Windows 10 | 8 GB | AMD Ryzen 5 4500U 2,38 GHz | 500 GB |

2. *Virtual Machine GCE*

   *Virtual Machine GCE* yang digunakan dalam penelitian ini memiliki spesifikasi disajikan pada Tabel 2.

Tabel 2. Spesifikasi *GCE*

| Sistem Operasi | Memori | CPU | Storage |
|---|---|---|---|
| Ubuntu 20.04 LTS | 8 GB | single core 2 vCPU 2,20 GHz | 20 GB |

3. *VPS*

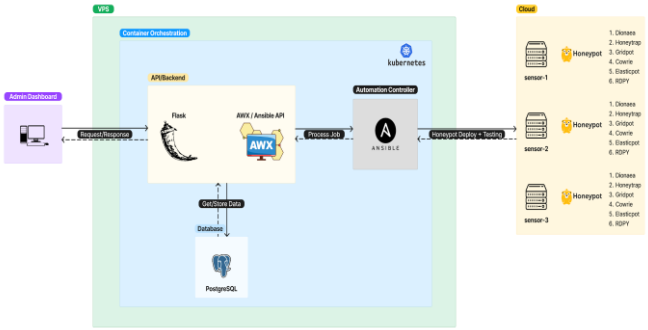
   *VPS* yang digunakan dalam penelitian ini memiliki spesifikasi disajikan pada Tabel 3.

Tabel 3. Spesifikasi *VPS*

| Sistem Operasi | Memori | CPU | *Storage* |
|---|---|---|---|
| Windows 10 | 16 GB | Dual core 3,50 GHz | 1,5 TB |

### C. Arsitektur Sistem *Honeypot Deployment*

Sistem terbagi menjadi beberapa komponen di antaranya adalah web *dashboard*, *API backend* dan *Ansible*, dan sensor di *cloud environment*. Proses *honeypot deployment* diawali dengan admin menambahkan sensor yang ingin diinstal melalui *form* di web *dashboard*. Pada *form* terdapat beberapa parameter yang wajib diisi seperti *IP address* sensor, nama sensor, dan *honeypot* yang ingin diinstal. Setelah *submit form*, data akan dikirimkan ke *backend* atau *Flask API* untuk diproses. Beberapa data akan disimpan ke *database* untuk kebutuhan web *dashboard* dan sisanya akan diteruskan ke *API Ansible* yaitu *AWX*. *AWX* memproses data tersebut untuk mempersiapkan *job template honeypot deployment*. Setelah *job template* selesai disiapkan, *Ansible* mulai mengeksekusi proses *honeypot deployment* ke sensor yang berjalan di *cloud GCP* sekaligus menjalankan *testing* secara terotomasi untuk memastikan *honeypot* telah berhasil terpasang. Arsitektur sistem *honeypot deployment* secara lengkap disajikan pada Gambar 1.




Gambar 1. Arsitektur sistem *honeypot deployment*

D. Topologi Jaringan Sistem

Topologi jaringan pada penelitian ini terbagi menjadi 2 *network*, yaitu *home network* dan *VPC network*. *Home network* merupakan jaringan *private* yang di dalamnya terdapat perangkat *PC* admin, *VPS*, dan *router*. *PC* admin dan *VPS* terhubung ke *router* untuk mendapatkan *private IP* agar mampu berkomunikasi di jaringan lokal. Selain itu, *router* juga mendapatkan *NAT IP* dari *ISP* agar perangkat yang terhubung ke *router* memiliki akses ke internet. *VPC network* merupakan jaringan *cloud* yang di dalamnya terdapat perangkat *VM GCE*, *VPC Router*, dan *VPC firewall*. *GCE* tersebar di beberapa *region* sehingga masing-masing terhubung pada *network address* yang berbeda. Selain itu, fungsinya juga dibagi menjadi sensor dan *main server*. Sensor merupakan *VM* yang memiliki fungsi khusus sebagai *honeypot server*. *Server* ini menjadi target utama untuk proses *honeypot deployment*. Semua *port* yang terdapat pada sensor diekspos untuk menerima semua *traffic*. Namun, terdapat *port* khusus yaitu *port* 22888 untuk koneksi *SSH* dari *PC* admin dan tidak bisa diakses dari *IP* lain. Sementara itu, *main server* berfungsi sebagai *server* yang dilindungi dari sebagian besar *traffic*. *Server* tersebut hanya bisa diakses oleh *traffic* yang berasal dari *PC* admin. Untuk mendapatkan hasil tersebut diperlukan *VPC firewall* yang mengatur setiap *traffic* yang masuk ke *VPC network*. Spesifikasi pembagian *IP address* dan *rules* pada *firewall* disajikan pada Tabel 4 dan Tabel 5.

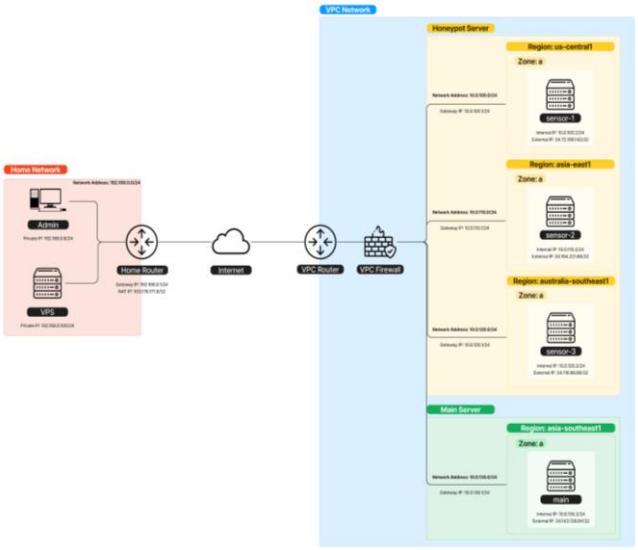
Tabel 4. Pembagian *IP address GCE*

| No | Perangkat | *IP Address* | *Subnet* | Jenis *IP* |
|---|---|---|---|---|
| 1 | Admin *PC* | 192.168.0.9 | /24 | *Private* |
| 2 | *VPS* | 192.168.0.100 | /24 | *Private* |
| 3 | *Home Router* | 192.168.0.1 | /24 | *Gateway* |
| | | 103.178.171.9 | /32 | *NAT* |
| 4 | *VPC Router* | 10.0.100.1 | /24 | *Gateway* |
| | | 10.0.110.1 | /24 | *Gateway* |
| | | 10.0.120.1 | /24 | *Gateway* |
| | | 10.0.130.1 | /24 | *Gateway* |
| 5 | sensor-1 | 10.0.100.2 | /24 | *Internal* |
| | | 34.72.199.142 | /32 | *External* |
| 6 | sensor-2 | 10.0.110.2 | /24 | *Internal* |
| | | 35.194.221.89 | /32 | *External* |
| 7 | sensor-3 | 10.0.120.2 | /24 | *Internal* |
| | | 34.116.96.86 | /32 | *External* |
| 8 | main | 10.0.130.2 | /24 | *Internal* |
| | | 34.143.139.84 | /32 | *External* |

Tabel 5. Daftar *rules firewall*

| No | Nama | *Port* | *Source IP* | Target | *Priority* | *Action* |
|---|---|---|---|---|---|---|
| 1 | *hp-allow-custom-ssh-admin* | tcp:22888 | 103.178.171.9/32 | *hp-server* | 1 | *allow* |
| 2 | *hp-deny-custom-ssh* | tcp:22888 | 0.0.0.0/0 | *hp-server* | 2 | *deny* |
| 3 | *hp-allow-all-inbound* | all | 0.0.0.0/0 | *hp-server* | 1000 | *allow* |
| 4 | *main-allow-all-inbound-admin* | all | 103.178.171.9/32 | *main-server* | 1 | *allow* |
| 5 | *main-deny-all-inbound* | all | 0.0.0.0/0 | *main-server* | 1000 | *deny* |

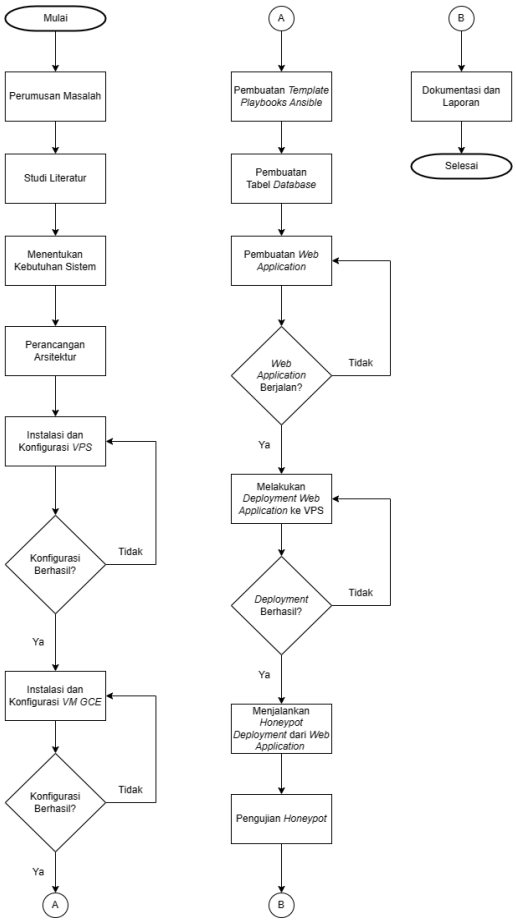
Topologi jaringan sistem pada *home network* dan *VPC network* secara keseluruhan disajikan pada Gambar 2.




Gambar 2. Topologi jaringan sistem

E. Diagram Alir Penelitian

Diagram alir dari penelitian yang dilakukan disajikan pada Gambar 3.




Gambar 3. Diagram alir penelitian

F. Instalasi dan Konfigurasi *VPS*

Secara umum langkah-langkah instalasi dan konfigurasi *VPS* adalah sebagai berikut.

1. Konfigurasi *User*

Konfigurasi *user* dilakukan dengan cara membuat *user* baru bernama *ansiadmin*. *User* ini akan menjadi *user* utama untuk mengeksekusi berbagai macam perintah di *server*. Selanjutnya menambahkan *user* tersebut ke grup *root* dan menambahkan akses *sudo*.

2. Konfigurasi *SSH*

Konfigurasi *SSH* dilakukan dengan menjalankan *generate SSH keygen* pada *user "ansiadmin"*. Penggunaan sistem *key* ini bertujuan agar koneksi *SSH* dari *server* ke sensor dilakukan secara *password-less* atau tanpa *password*, sehingga proses *honeypot deployment* oleh *Ansible* dapat berfungsi dengan baik.

3. Instalasi *Package*

Instalasi *package* dilakukan dengan menginstal beberapa *package* seperti *net-tools, lynx, acl, python3-pip,* dan *ansible*.

4. Instalasi dan Konfigurasi *Docker*

Melakukan instalasi *Docker* dengan nama *package docker.io* dan menambahkan *user "ansiadmin"* ke dalam grup *Docker* agar bisa memiliki akses untuk mengeksekusi perintah *Docker*.

5. Instalasi *Minikube*

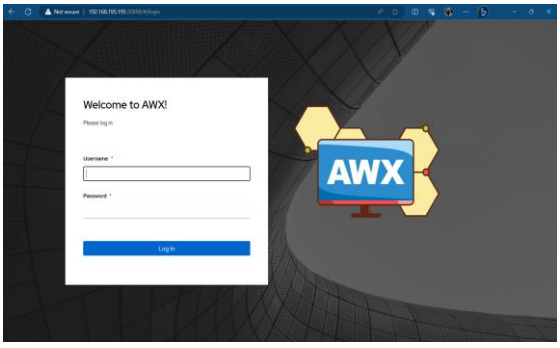
Melakukan instalasi *Minikube* sebagai *single cluster Kubernetes* dan instalasi *kubectl* sebagai *command-line tool* untuk pengelolaan *cluster Kubernetes*.

6. Instalasi *PostgreSQL*

Melakukan instalasi *PostgreSQL* di dalam *Minikube* atau *cluster Kubernetes*. *PostgreSQL* berperan sebagai sistem *database* utama pada penelitian ini.

7. Instalasi *AWX Ansible*.

Melakukan instalasi *AWX* di dalam *Minikube* dan menjalankan *port forwarding* agar memiliki akses ke *dashboard* web *AWX*. Tampilan awal instalasi *AWX Ansible* disajikan pada Gambar 4.




Gambar 4. Halaman *login AWX*

G. Konfigurasi *AWX Ansible*

Secara umum konfigurasi yang dilakukan pada *AWX Ansible* adalah sebagai berikut.

1. Menambahkan *Organizations*

Pada *AWX Ansible*, *Organizations* merupakan reprensentasi dari pengelompokan *resources*, seperti *host, inventory, project*, dan *credential* di dalam *environment AWX Ansible* dalam rangka pengelolaan *resources* dan akses kontrol.

2. Menambahkan *Execution Environment*

Pada *AWX Ansible*, *Execution Environment* merupakan *runtime environment* yang telah ditentukan sebelumnya untuk mengeksekusi *playbooks* dan *jobs* pada *Ansible*. *Execution Environment* dibangun pada kontainer *Docker* yang di dalamnya sudah terinstal *Ansible*, *Python*, dan *dependency* lainnya sehingga mampu mengontrol serta mengelola jalannya *playbooks Ansible*.

3. Menambahkan *Projects*

Pada *AWX Ansible*, *Projects* merupakan komponen mendasar yang digunakan untuk mengatur dan mengelola *playbooks*, *templates*, dan *resources* lainnya serta mendapatkan berkas tersebut melalui *version control system* seperti *Git*.

4. Menambahkan *Inventories*

Pada *AWX Ansible*, *Inventories* merupakan kumpulan *hosts, groups*, dan *variables* yang menjadi target bagi *Ansible* untuk mengeksekusi *playbooks* dan *jobs*.

5. Menambahkan *Credential Types*

Pada *AWX Ansible*, *Credential Types* merupakan konfigurasi tambahan berupa tipe kredensial yang telah didefinisikan terlebih dahulu sebelumnya untuk mengakses berbagai sistem, layanan, dan *resources* saat *playbooks* dan *jobs Ansible* dieksekusi. *Credential Types* berisi informasi dan protokol yang digunakan saat membangun koneksi ke sistem target. *"Sensor Credential"* merupakan jenis kredensial yang digunakan untuk autentikasi ke sensor. *"Postgres Credential"* merupakan jenis kredensial yang digunakan untuk autentikasi ke *database*.

6. Menambahkan *Credentials*

Pada *AWX Ansible*, *Credentials* merupakan informasi yang dibutuhkan untuk mengakses dan autentikasi pada berbagai sistem, layanan, dan *resources* saat mengeksekusi *playbooks* dan *jobs Ansible*. *Credentials* memanfaatkan *Credential Types* sebagai *template* untuk mendefinisikan informasi seperti *username, password, SSH Keys*, dan sebagainya yang dibutuhkan ketika membangun koneksi ke sistem.

Kredensial *"ansible"* merupakan kredensial yang digunakan oleh *server* untuk mendapatkan akses *SSH* ke sensor. Kredensial *"sensor_init"* merupakan kredensial *user* yang digunakan oleh *server* untuk melakukan konfigurasi awal pada sensor sebelum *user "ansigent"* dibuat. Kredensial *"ansigent"* merupakan kredensial *user* yang digunakan oleh *server* untuk melakukan *deployment package* dan *honeypot* pada sensor. Kredensial *"postgres"* merupakan kredensial yang digunakan oleh *server* untuk mengakses *database* penelitian.

7. Menambahkan *Applications*

Pada *AWX Ansible*, *Applications* merupakan fitur yang berfungsi untuk memudahkan integrasi *API Ansible* dengan aplikasi lain. *Applications* menyediakan *token* yang bisa digunakan oleh aplikasi lain agar bisa memiliki akses ke *resources Ansible*. Setelah *Applications* ditambahkan *token* tidak langsung didapatkan karena *token* tersebut harus diasosiasikan dengan *user* terlebih dahulu.

8. Menambahkan *Templates*

Pada *AWX Ansible*, *Templates* merupakan serangkaian konfigurasi dan parameter yang digunakan untuk mendefinisikan serta mengeksekusi *playbooks* atau *jobs Ansible*. *Templates* menyajikan konfigurasi yang dapat digunakan berulang kali dengan aturan dan masukan yang sudah ditentukan, yang disebut sebagai *Job Templates*. *Workflow Job Templates* merupakan serangkaian *Job Templates* yang disusun dan dieksekusi secara berurutan sesuai dengan aturan yang telah didefinisikan.

a. *Job template: 1. Initial Configuration Honeypot*

*Template* ini mendefinisikan konfigurasi awal pada sensor sebelum melakukan *honeypot deployment*. Di dalamnya terdapat *playbook* dengan berbagai macam *tasks* seperti melakukan membuat *user "ansigent"*, menambahkan akses *sudo* ke *user "ansigent"*, menambahkan *SSH public key* ke *user "ansigent"*, dan memperbarui *port SSH* menjadi 22888.

b. *Job Template: 2. Package Installation Honeypot*

*Template* ini mendefinisikan instalasi *package* yang akan diinstal pada sensor. Di dalamnya terdapat *playbook* dengan berbagai macam *tasks* seperti instalasi *base-tools* seperti *curl, git, nano,* dsb., instalasi *QEMU* yaitu *emulator* untuk menjalankan *image Docker* di berbagai arsitektur *CPU*, instalasi *Docker*, dan instalasi modul *Python PostgreSQL*.

c. *Job Template*: *3. Deploy Selected Honeypot*

*Template* ini mendefinisikan instalasi jenis *honeypot* yang akan diinstal pada sensor. Di dalamnya terdapat *playbook* dengan berbagai macam *tasks* seperti menjalankan *task honeypot deployment*, mengecek status *deployment* secara berkala, menghentikan *service honeypot* yang tidak dipilih.

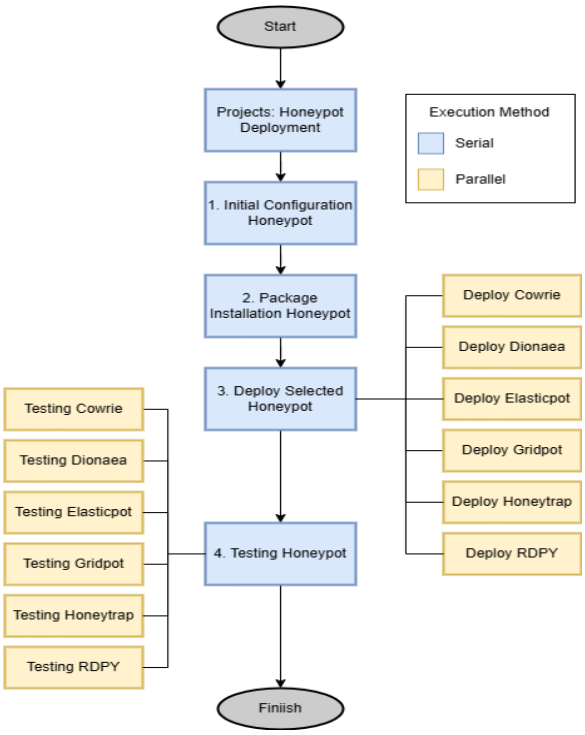
d. *Job Template*: *4. Testing Honeypot*

*Template* ini mendefinisikan pengecekan status *honeypot* pasca proses *deployment*. Di dalamnya

terdapat *playbook* dengan berbagai macam *tasks* seperti mengecek status *honeypot* berdasarkan *port* yang dibuka, menunggu proses pengecekan selesai, menampilkan *output* dari hasil pengecekan.

e. *Workflow Job Template*: *Initial Deployment Workflow*

*Template* ini mendefinisikan alur eksekusi *Job Templates* yang telah dibuat sebelumnya saat proses *honeypot deployment* pertama kali dijalankan. *Initial Deployment Workflow* disajikan pada Gambar 5.




Gambar 5. Visualisasi *Initial Deployment Workflow*

f. *Workflow Job Template*: *Initial Deployment Workflow*

*Template* ini mendefinisikan alur eksekusi *Job Templates* apabila terdapat perubahan pada *honeypot* yang sudah diinstal pada sensor.

H. Konfigurasi Sensor

Pada sensor hanya perlu melakukan konfigurasi *SSH*. Konfigurasi dilakukan dengan cara menyalin *public key SSH* yang telah dibuat di *VPS* ke sensor. *VPS* membutuhkan akses *SSH* ke sensor untuk bisa mengeksekusi *jobs honeypot deployment* pada sensor.

I. Pembuatan Tabel *Database*

Pembuatan tabel *database* pada penelitian ini tidak dilakukan secara manual, melainkan menggunakan fitur migrasi. Migrasi ini merupakan sebuah *version control system* untuk *database*. Beberapa *framework* web sudah mendukung fitur ini, salah satunya *Flask*. Fitur ini sangat membantu dalam pengelolaan tabel sehingga tabel selalu tersinkronisasi dengan struktur model data yang didefinisikan pada *source code* aplikasi web. Caranya cukup sederhana, yaitu diawali dengan membuat komponen model yang isinya terdapat nama tabel, nama kolom, tipe data, dan *constraint*. Lalu, komponen tersebut diinisialisasi ke dalam proyek *Flask*. Terakhir, menjalankan *command* migrasi dari *Flask*.

J. Pembuatan *API*

Hal pertama yang perlu dilakukan sebelum membuat *API* adalah menentukan spesifikasi dari *API* tersebut. Berikut spesifikasi *API* yang dibuat.

1. *Users*: *API* ini berfungsi untuk pengelolaan *users* yang terdaftar pada aplikasi web.
2. *Authentications*: *API* ini berfungsi untuk pengelolaan *token* milik *user* yang terdaftar pada aplikasi web. *Token* ini berfungsi agar *user* memiliki akses ke *resources* pada web.
3. *Sensors*: *API* ini berfungsi untuk pengelolaan sensor yang terdaftar pada aplikasi web.
4. *Honeypots*: *API* ini berfungsi untuk pengelolaan *honeypot* yang terdaftar pada aplikasi web.
5. *Jobs*: *API* ini terhubung langsung dengan *API AWX Ansible*. *API* ini berfungsi untuk mengeksekusi *Workflow Job Templates* yang terdaftar pada *AWX Ansible*.
6. *HoneypotSensor*: Setiap sensor memiliki *honeypot* sendiri. *API* ini berfungsi sebagai relasi antara sensor dengan *honeypot* yang terinstal di dalamnya. *Honeypot* yang terinstal di satu sensor tidak bergantung dengan *honeypot* di sensor lainnya. Sehingga *API* ini mengatasi ketergantungan tersebut.

K. Pembuatan *Dashboard*

Pada tahapan ini adalah melakukan pembuatan *dashboard* mengacu desain *mockup* yeng telah dibuat. Berikut uraian tampilan *mockup* halaman pada *dashboard*.

1. *Register*: Pada halaman *register* digunakan untuk mendaftarkan *user* baru ke dalam aplikasi.
2. *Login*: Pada halaman *login* digunakan untuk melakukan proses autentikasi *user*.
3. *Dashboard*: Halaman *dashboard* merupakan halaman utama yang digunakan untuk menampilkan beberapa informasi umum seperti jumlah sensor, jumlah *honeypot*, jumlah *deployment* yang sudah dijalankan, daftar riwayat *honeypot deployment* di semua sensor, dan navigasi ke beberapa halaman lain.
4. *Sensors*: Pada halaman *sensors* menampilkan daftar sensor yang terdaftar pada aplikasi.
5. *Add New Sensor*: Halaman *add new sensor* merupakan halaman untuk menambahkan sensor baru ke dalam aplikasi.
6. *Sensor Details*: Pada halaman *sensors details* menampilkan informasi sensor yang terdaftar secara detil.
7. *Sensor Logs*: Pada halaman *sensors logs* menampilkan informasi *logs* saat proses *honeypot deployment* dijalankan.
8. *Honeypots*: Pada halaman *honeypots* menampilkan informasi daftar *honeypot* yang tersedia pada aplikasi.

9. *Add New Honeypot*: Halaman *add new honeypot* merupakan halaman untuk menambahkan *honeypot* baru ke dalam aplikasi.
10. *Honeypot Details*: Pada halaman *honeypots details* menampilkan informasi *honeypot* yang terdaftar secara detil.
11. *Users*: Pada halaman *users* menampilkan informasi daftar *user* yang terdaftar pada aplikasi.
12. *Users Details*: Pada halaman *users details* menampilkan informasi *user* yang terdaftar secara detil.
13. *Profile*: Pada halaman *profile* menampilkan informasi *user* yang digunakan saat mengakses aplikasi web.

L. *Deployment* Aplikasi Web

Setelah *API* dan *dashboard* selesai dibuat, maka langkah berikutnya adalah melakukan *deployment* aplikasi ke *VPS*. Aplikasi tersebut dijalankan di dalam *Minikube*. Namun, sebelum dijalankan aplikasi tersebut harus melewati proses *building docker image* terlebih dahulu.

M. Metode Pengujian

Pada penelitian ini terdapat dua pengujian yaitu pengujian *dashboard* dan pengujian *honeypot*. Pengujian *dashboard* dilakukan dengan menguji berbagai menu dan fungsi yang terdapat pada *dashboard*. Pengujian ini bertujuan untuk mengetahui apakah aplikasi sudah berfungsi sesuai dengan yang dibutuhkan. Pengujian *honeypot* dilakukan dengan melakukan *scanning port* pada sensor menggunakan *tools netcat*. Setiap *honeypot* memiliki beragam *service* yang diekspos. Namun, untuk pengujian ini hanya diambil satu *service* dari setiap *honeypot*. *Service* atau *port honeypot* yang diuji disajikan pada Tabel 6 berikut.

Tabel 6. Daftar *service/port honeypot* yang diuji

| No | *Honeypot* | *Port* |
|---|---|---|
| 1 | *Cowrie* | tcp:22 |
| 2 | *Dionaea* | tcp:21 |
| 3 | *Elasticpot* | tcp:9200 |
| 4 | *Gridpot* | tcp:8000 |
| 5 | *Honeytrap* | tcp:631 |
| 6 | *RDPY* | tcp:3389 |

## IV. HASIL DAN PEMBAHASAN

Hasil yang didapatkan dari penelitian ini adalah sebuah sistem atau aplikasi yang mampu melakukan *honeypot deployment* pada sensor secara otomatis dengan memanfaatkan *tools* seperti *Ansible* dan *dashboard* yang membantu *user* untuk mengeksekusi dan *monitoring* proses *deployment*. Untuk memastikan apakah aplikasi dapat berjalan dengan baik maka diperlukan pengujian. Ada dua jenis pengujian yang dilakukan pada aplikasi, yaitu pengujian *dashboard* dan pengujian *honeypot*.

A. Pengujian *Dashboard*

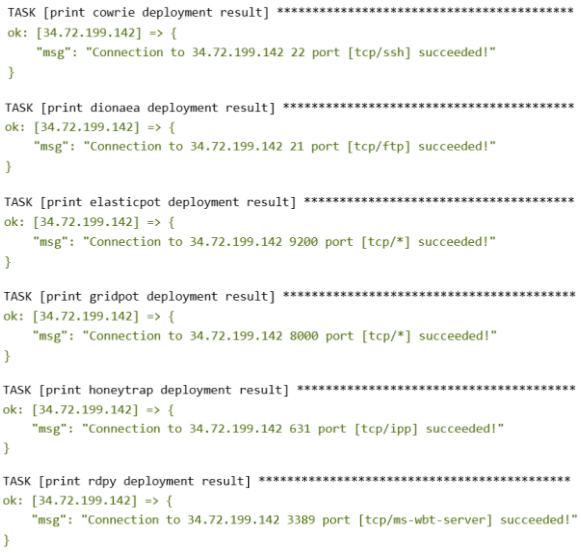
Pengujian *dashboard* berfungsi untuk mengetahui apakah aplikasi sudah berfungsi sesuai yang diharapkan. Pengujian ini dimulai dari penambahan *user* baru atau *register* hingga proses *honeypot deployment* dijalankan oleh *user*. Rincian hasil pengujian *dashboard* disajikan pada Tabel 7.

Tabel 7. Rincian hasil pengujian *dashboard*

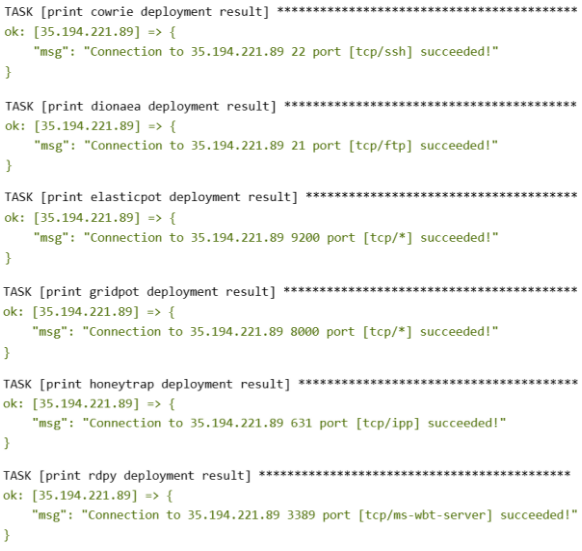
| Komponen Pengujian | Skenario yang diuji | Hasil Pengujian | Kesimpulan |
|---|---|---|---|
| Fitur *Register* | Melakukan pendaftaran *user* dengan memasukkan *username*, *password*, dan nama lengkap. | *User* berhasil terdaftar dan memiliki akses ke dalam *dashboard*. | ☒ Berhasil ☐ Tidak |
| Fitur *Login* | Melakukan *login* dengan menggunakan *username* dan *password* | *User* berhasil *login* serta mendapatkan *token*. | ☒ Berhasil ☐ Tidak |
| Menu *Dashboard* | Menampilkan jumlah total sensor dan *honeypot* yang terdaftar pada aplikasi dan *history honeypot deployment* dari semua sensor. | Pada menu *Dashboard* sudah bisa menampilkan data sesuai dengan skenario. | ☒ Berhasil ☐ Tidak |
| Menu *Sensors* | Menampilkan daftar sensor yang terdaftar pada aplikasi. | Pada menu *Sensors* sudah bisa menampilkan daftar sensor yang terdaftar pada aplikasi. | ☒ Berhasil ☐ Tidak |
| Fitur *Add Sensor* | Mendaftarkan sensor baru dengan memasukkan nama sensor, *IP address*, *description*, dan *honeypot* yang ingin diinstal. | Sensor berhasil ditambahkan dan ditampilkan di menu *Sensors*. | ☒ Berhasil ☐ Tidak |
| Menu *Sensor Details* | Menampilkan data informasi sensor secara detil. | Pada menu *Sensor Details* sudah bisa menampilkan informasi sensor secara detil. | ☒ Berhasil ☐ Tidak |
| Menu *Sensor Logs* | Menampilkan informasi *logs* proses *honeypot deployment*. | Pada menu *Sensor Logs* sudah bisa menampilkan *logs* dari proses | ☒ Berhasil ☐ Tidak |

| Komponen Pengujian | Skenario yang diuji | Hasil Pengujian | Kesimpulan |
|---|---|---|---|
| | | *honeypot deployment*. | |
| Fitur *Relaunch Job* | Menjalankan ulang proses *honeypot deployment.* | Proses *honeypot deployment* berhasil dijalankan ulang menggunakan fitur *Relauch Job*. | ☒ Berhasil ☐ Tidak |
| Fitur *Update Sensor* | Memperbarui data sensor. | Data sensor berhasil diperbarui dengan fitur *Update Sensor*. | ☒ Berhasil ☐ Tidak |
| Fitur *Delete Sensor* | Menghapus data sensor dari aplikasi. | Data sensor berhasil dihapus dari aplikasi. | ☒ Berhasil ☐ Tidak |
| Menu *Honeypots* | Menampilkan daftar *honeypot* yang terdaftar pada aplikasi. | Pada menu *Honeypots* sudah bisa menampilkan daftar *honeypot* yang terdaftar pada aplikasi. Menu ini juga hanya bisa diakses oleh *roles* admin. | ☒ Berhasil ☐ Tidak |
| Fitur *Add Honeypot* | Mendaftarkan *honeypot* baru dengan memasukkan nama *honeypot* dan *description*. | *Honeypot* berhasil ditambahkan dan ditampilkan di menu *Honeypots*. Fitur ini juga hanya bisa diakses oleh *roles* admin. | ☒ Berhasil ☐ Tidak |
| Menu *Honeypot Details* | Menampilkan data informasi *honeypot* secara detil. | Pada menu *Honeypot Details* sudah bisa menampilkan informasi *honeypot* secara detil. Menu ini juga hanya bisa diakses oleh *roles* admin. | ☒ Berhasil ☐ Tidak |
| Fitur *Update Honeypot* | Memperbarui data *honeypot.* | Data *honeypot* berhasil diperbarui dengan fitur *Update Honeypot*. Fitur ini juga hanya bisa diakses oleh *roles* admin. | ☒ Berhasil ☐ Tidak |
| Fitur *Delete Honeypot* | Menghapus data *honeypot* dari aplikasi. | Data *honeypot* berhasil dihapus dari aplikasi. Fitur ini juga hanya bisa diakses oleh *roles* admin. | ☒ Berhasil ☐ Tidak |
| Menu *Users* | Menampilkan daftar *user* yang terdaftar pada aplikasi. | Pada menu *Users* sudah bisa menampilkan daftar *user* yang terdaftar pada aplikasi. Menu ini juga hanya bisa diakses oleh *roles* admin. | ☒ Berhasil ☐ Tidak |
| Menu *User Details* | Menampilkan data informasi *user* secara detil. | Pada menu *User Details* sudah bisa menampilkan informasi *user* secara detil. Menu ini juga hanya bisa diakses oleh *roles* admin. | ☒ Berhasil ☐ Tidak |
| Fitur *Update User* | Memperbarui *roles* pada *user*. | *Roles user* berhasil diperbarui dengan fitur *Update User*. Menu ini juga hanya bisa diakses oleh *roles* admin. | ☒ Berhasil ☐ Tidak |
| Menu *Profile* | Menampilkan data informasi pemilik akun. | Pada menu *Profile* sudah bisa menampilkan data informasi pemilik akun. | ☒ Berhasil ☐ Tidak |
| Fitur *Update Profile* | Memperbarui data informasi pemilik akun. | Data informasi pemilik akun berhasil diperbarui menggunakan fitur *Update Profile*. | ☒ Berhasil ☐ Tidak |

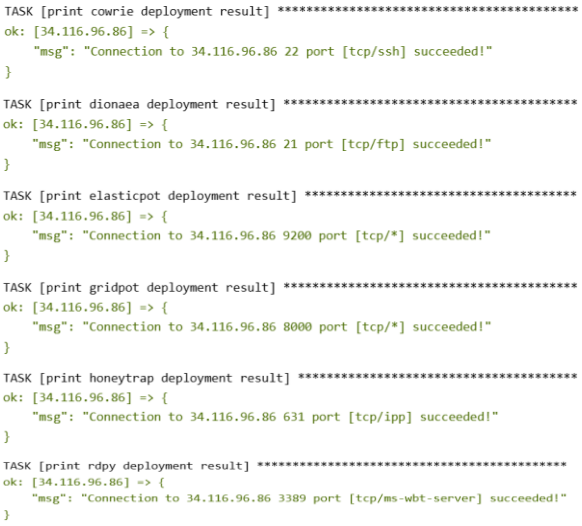
Pengujian *honeypot* berfungsi untuk mengetahui apakah *honeypot* berhasil terpasang dan *service*-nya berjalan di sensor. Pengujian ini dilakukan pada 3 sensor yang tersebar di beberapa *region*. Dari hasil pengujian didapatkan bahwa semua *honeypot* yang telah selesai diinstal pada setiap sensor dapat menerima *traffic* dari *port* atau *service* yang dijalankan. Pengujian ini dilakukan pada 3 sensor yang tersebar di beberapa *region*. Dari hasil pengujian didapatkan bahwa semua *honeypot* yang telah selesai diinstal pada setiap sensor dapat menerima *traffic* dari *port* atau *service* yang dijalankan. Hasil pengujian *honeypot* sensor-1 disajikan pada Gambar 6. Hasil pengujian *honeypot* sensor-2 disajikan pada Gambar 7. Hasil pengujian *honeypot* sensor-3 disajikan pada Gambar 8. Berikut ini hasil dari *honeypot deployment* di semua sensor.




Gambar 6. Hasil pengujian *honeypot* sensor-1

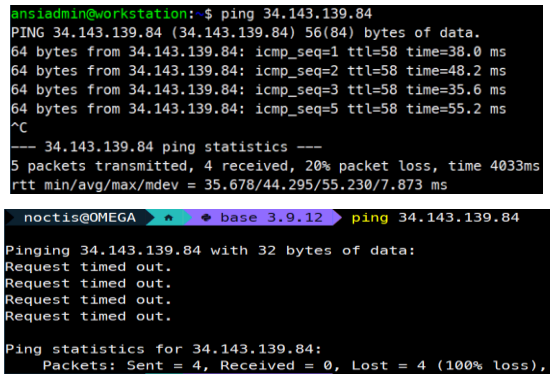



Gambar 7. Hasil pengujian *honeypot* sensor-2




Gambar 8. Hasil pengujian *honeypot* sensor-3

Pengujian untuk *main server* juga dilakukan untuk memastikan bahwa *server* hanya bisa diakses oleh *PC* admin. Pengujian dijalankan dengan melakukan *ping* ke *main server*. Pengujian pertama dilakukan dari *PC* admin, berikutnya dari *PC* lain. Pengujian pertama sudah dipastikan berhasil melakukan *ping*, sedangkan berikutnya gagal melakukan *ping* karena *traffic* diblok oleh *firewall* pada *VPC*. Hasil pengujian disajikan pada Gambar 9 berikut.




Gambar 9. Hasil pengujian *ping* ke *main server*

## V. SIMPULAN

Berdasarkan penelitian yang sudah dilaksanakan dapat dianalisis dan menghasilkan beberapa kesimpulan bahwa, penelitian ini menghasilkan aplikasi web yang difungsikan khusus untuk *honeypot deployment* dengan memanfaatkan *tools Ansible*. *API Flask* dan *Ansible AWX* terintegrasi dengan baik sehingga proses *honeypot deployment* dapat dieksekusi melalui *dashboard* aplikasi web. Aplikasi sudah bisa menampilkan *logs* sehingga proses *honeypot deployment* bisa terpantau dengan baik.

Berdasarkan kesimpulan yang sudah dijelaskan, peneliti menambahkan beberapa saran yang dapat dilakukan untuk mengembangkan penelitian selanjutnya yaitu proses instalasi dan konfigurasi pada *VPS* dan sensor masih dilakukan secara manual, sehingga diperlukan otomasi. Otomasi juga diperlukan pada tahap pengkodingan sampai *deployment* aplikasi ke *VPS* dengan menerapkan konsep *DevOps*. Jumlah *honeypot* yang tersedia pada aplikasi masih terbatas dan sangat memungkinkan jumlahnya bertambah, sehingga dibutuhkan fitur untuk menambahkan *honeypot* baru pada aplikasi dan diotomasi prosesnya.

## REFERENSI

[1] W. N. M. Sukma, A. Reynata, D. A. Yasmine and D. M. P. Pratama, "SYSTEMATIC LITERATURE REVIEW (SLR) : KEAMANAN DALAM SISTEM INFORMASI," *Journal of Comprehensive Science*, vol. II, no. 6, 2023.

[2] Kominfo, "Antisipasi Bersama Tingkatkan Sistem dan Cegah Serangan Siber," 22 September 2022. [Online]. Available: https://aptika.kominfo.go.id/2022/09/antisipasi-bersama-tingkatkan-sistem-dan-cegah-serangan-siber/.

[3] N. Naik, "IMPLEMENTATION OF TECHNIQUES TO AVOID CYBER ATTACKS," *The Online Journal of Distance Education and e-Learning*, vol. XI, no. 2, 2023.

[4] S. Parulian, D. A. Pratiwi and M. C. Yustina, "Ancaman dan Solusi Serangan Siber di Indonesia," *Telecommunications, Networks,*

*Electronics, and Computer Technologies*, vol. I, no. 2, pp. 85-92, 2021.

[5] A. S. Verma and A. Dubey, "A Review on Honeypot Deployment," *London Journal of Research in Computer Science and Technology*, vol. XX, no. 1, 2020.

[6] J. Lovdianchel, "Mengenal Ansible Automation," 2020. [Online]. Available: https://medium.com/dot-intern/configuration-management-dengan-ansible-case-study-dcd0fe925064. [Accessed 8 Agustus 2023].

[7] M. Catherine and A. Kumawat, "Securing a Small Network by using Raspberry Pi Honeypot," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. IX, no. 7, pp. 773-778, 2020.

[8] I. Beres and D. Hurley-Smith, "Dynamic honeypot deployment in the cloud," 2022.

[9] M. Hasbi, A. R. A. Nurwa, D. F. Priambodo and W. R. A. Putra, "Infrastructure as Code for Security Automation and Network Infrastructure Monitoring," *Matrik: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer*, 2022.

[10] A. Khumaidi, "IMPLEMENTATION OF DEVOPS METHOD FOR AUTOMATION OF SERVER MANAGEMENT USING ANSIBLE," *TRANSFORMTIKA*, 2021.

[11] J. Buzzio-Garcia, "Creation of a High-Interaction Honeypot System based-on Docker containers," in *World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, 2021.

[12] V. S. D. Priya and S. S. Chakkaravarthy, "Containerized cloud-based honeypot deception for tracking attackers," Scientific Reports, 2023.

[13] C. Wijayanto and Y. A. Susetyo, "IMPLEMENTASI FLASK FRAMEWORK PADA PEMBANGUNAN APLIKASI SISTEM INFORMASI HELPDESK (SIH)," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, 2022.

[14] D. F. Ningtyas and N. Setiyawati, "Implementasi Flask Framework pada Pembangunan Aplikasi Purchasing Approval Request," *Jurnal Janitra Informatika dan Sistem Informasi*, 2021.

[15] Nasution and L. Iswari, "Penerapan React JS Pada Pengembangan FrontEnd Aplikasi Startup Ubaform," *AUTOMATA*, 2021.

[16] N. Titarmare, N. Hargule and A. Gupta, "An Overview of Honeypot Systems," *International Journal of Computer Sciences and Engineering*, 2019.

[17] M. Zadka, DevOps in Python, 2nd ed., Apress, Berkeley, CA, 2022.

[18] M. D. Elradi, "Ansible: A Reliable Tool for Automation," *Electrical and Computer Engineering Studies*, vol. II, no. 1, 2023.

[19] P. Maha, "Konsep Dasar Ansible," 2020. [Online]. Available: https://medium.com/prastamaha/konsep-dasar-ansible-6e7451cb0502. [Accessed 7 Agustus 2023].

[20] J. Freeman and J. Keating, Mastering Ansible. Effectively automate configuration management and deployment challenges with Ansible 2.7, 3rd ed., Packt Publishing Ltd, 2019.

[21] Steffi, "What is Flask?," 2022. [Online]. Available: https://medium.com/data-science-ai-learning-journey/what-is-flask-cab5eb6e74f0. [Accessed 8 Agustus 2023].

[22] A. Boduch and R. Derks, React and React Native: A complete hands-on guide to modern web and mobile development with React.js, 3rd ed., Packt Publishing Ltd., 2020.

[23] R. Setiawan, "Apa Itu Docker? Apa Kegunaan dan Kelebihannya?," 2021. [Online]. Available: https://www.dicoding.com/blog/apa-itu-docker/. [Accessed 8 Agustus 2023].

[24] Kubernetes, "Apa itu Kubernetes?," 2020. [Online]. Available: https://kubernetes.io/id/docs/concepts/overview/what-is-kubernetes/. [Accessed 8 Agustus 2023].

[25] S. K. Mondal, R. Pan, H. M. D. Kabir, T. Tian and H.-N. Dai, "Kubernetes in IT administration and serverless computing: An empirical study and research challenges," *The Journal of Supercomputing*, 2021.