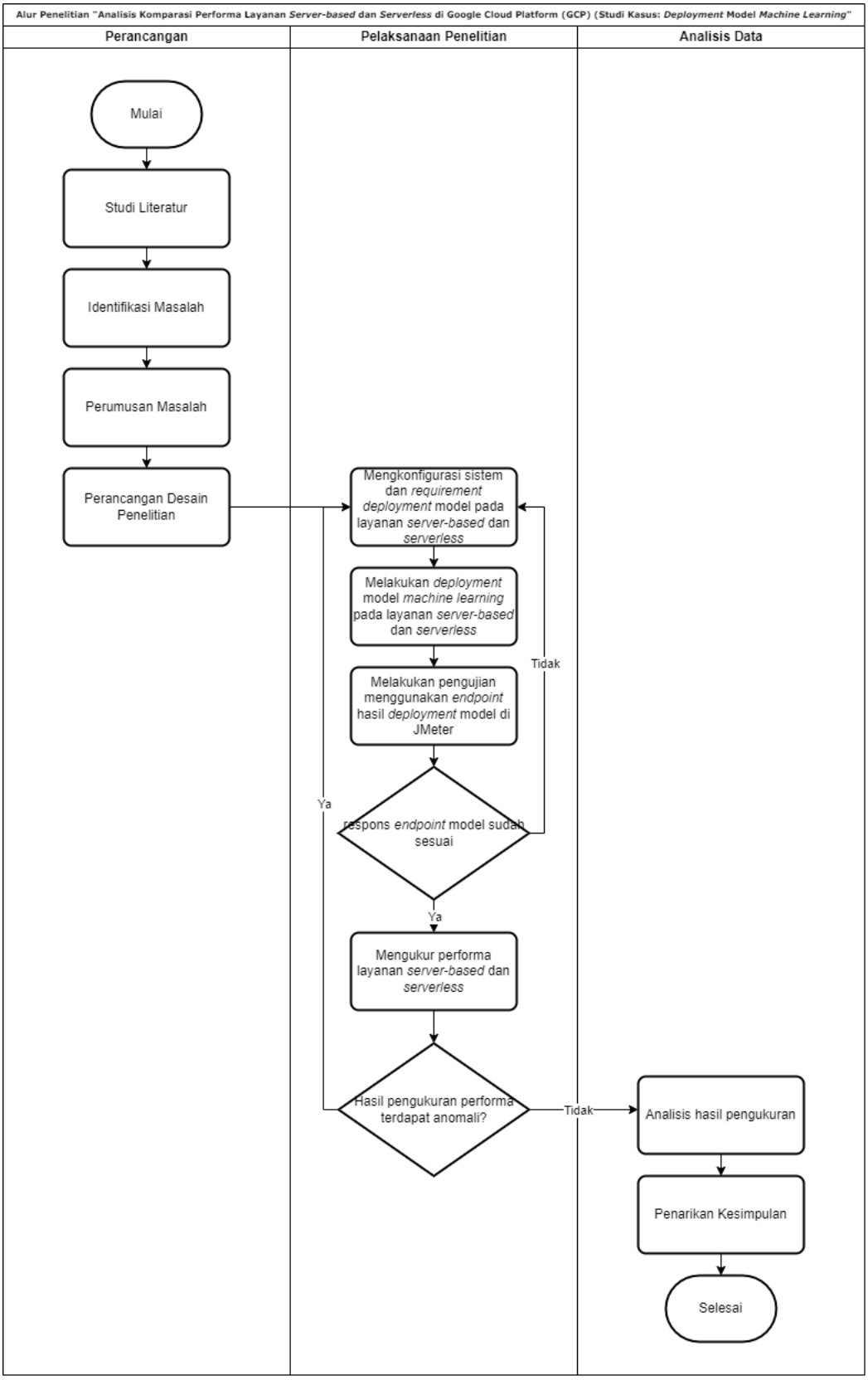
BAB III
METODOLOGI PENELITIAN

3.1 Jenis Penelitian

Jenis penelitian yang digunakan pada penelitian ini adalah komparatif dengan pendekatan kuantitatif. Menurut Sugiyono (2013), penelitian komparatif merupakan penelitian yang membandingkan suatu variabel pada dua atau lebih sampel atau waktu yang berbeda. Sedangkan, penelitian kuantitatif merupakan penelitian yang memenuhi kaidah ilmiah, terukur, objektif, dan sistematik berupa angka-angka dan analisis statistik. Sehingga, penelitian komparatif dengan pendekatan kuantitatif merupakan metode penelitian untuk melakukan suatu pengukuran dari data-data berupa angka yang akan dibandingkan. Metode penelitian yang digunakan adalah eksperimen untuk melakukan *deployment* model *machine learning* pada layanan *server-based* dan *serverless* di GCP dan pengukuran performa menggunakan beberapa parameter, di antaranya CPU dan *memory utilization*, *latency*, dan *pricing*. Untuk parameter komparasi tambahan adalah *developer experiences* yang meliputi kompatibilitas *framework machine learning*, tingkat kemudahan penerapan, dan tingkat ketersediaan dokumentasi. Setelah dilakukan pengukuran, akan dilakukan perbandingan performa layanan *server-based* dan *serverless* untuk mengetahui layanan terbaik dalam *deployment* model *machine learning*.

3.2 Alur Penelitian

Alur penelitian yang digunakan untuk penelitan ini mengacu pada penelitian kuantitatif yang serupa dengan tahapan penelitian secara umum (Ibrahim dkk., 2018) untuk melakukan pengukuran performa layanan *server-based* dan *serverless*. Alur penelitian disajikan pada Gambar 3.1.



Gambar 3.1 Diagram Alur Penelitian

3.2.1 Studi Literatur

Tahap studi literatur merupakan tahap awal yang bertujuan untuk mengeksplor sumber-sumber literatur yang berkaitan dengan penggunaan teknologi *cloud* seperti layanan-layanan *cloud* dan mengumpulkan informasi-informasi yang relevan dengan penggunaan layanan *cloud* tersebut serta menganalisis kebutuhan atau *gap* penelitian pada sumber-sumber literatur tersebut.

3.2.2 Identifikasi Masalah

Setelah melakukan studi literatur, identifikasi masalah berdasarkan kebutuhan atau *gap* penelitian untuk menentukan masalah dari objek yang akan diteliti. Penelitian ini mengidentifikasi permasalahan-permasalahan yang ada pada penggunaan layanan *cloud* setelah mengeksplor sumber literatur yang ada. Lalu mengerucutkan permasalahan pada literatur-literatur mengenai penggunaan layanan komputasi *cloud* untuk *deployment* aplikasi, khususnya aplikasi berupa model *machine learning*.

3.2.3 Perumusan Masalah

Pada tahap perumusan masalah, permasalahan-permasalahan yang sudah diidentifikasi, dirumuskan secara spesifik untuk mencapai tujuan penelitian. Permasalahan dirumuskan dalam bentuk pertanyaan-pertanyaan yang akan menjawab atau menyelesaikan permasalahan yang sudah diidentifikasi, yaitu untuk mengetahui performa layanan *server-based* dan *serverless* dengan melakukan *deployment* model *machine learning* di GCP.

3.2.4 Perancangan Desain Penelitian

3.2.4.1 Kebutuhan Penelitian

Beberapa kebutuhan atau instrumen yang menunjang penelitian ini, di antaranya perangkat keras dengan spesifikasi sebagai berikut:

- *Operating system* : Windows 11 Home Single Language 64-bit
- *Processor* : AMD Ryzen 5 5600H
- RAM : 16 GB
- Memori : 512 GB

Lalu, *browser* untuk mengakses Google Cloud Platform, dan *software* Apache JMeter (v5.6.3) untuk melakukan pengujian melalui *request endpoint* dari hasil *deployment* model *machine learning* pada masing-masing layanan *server-based* dan *serverless* di GCP.

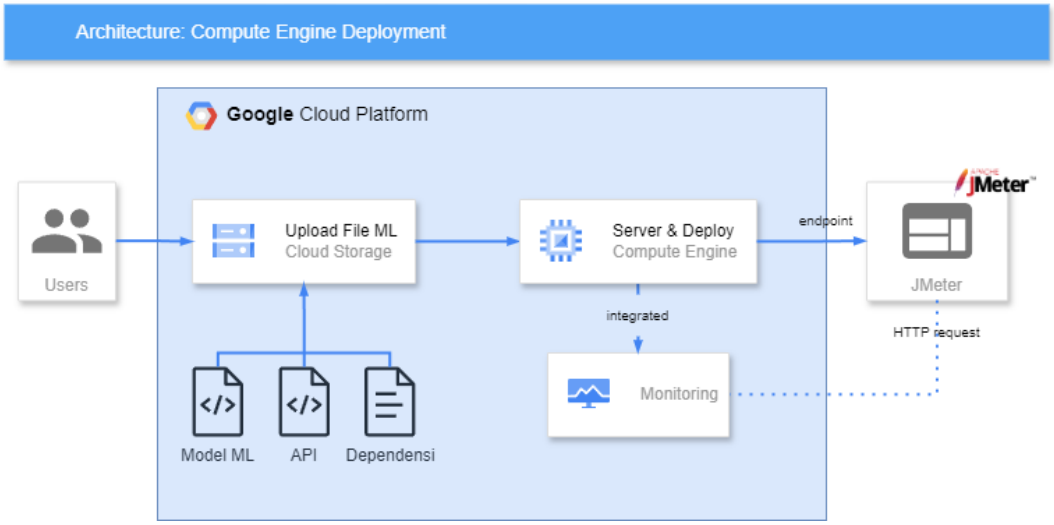
Untuk kebutuhan *deployment*, tentunya model *machine learning*, *file* API untuk menangani HTTP *request* dan *response*, serta dependensi yang terdiri dari Flask (v2.3.2), Gunicorn (v.21.0.0), scikit-learn (v1.3.0), functions-framework (v3.*).

3.2.4.2 Perancangan Arsitektur Sistem

Perancangan arsitektur sistem untuk masing-masing layanan *server*-based dan *serverless* disajikan pada Gambar 3.2 sampai Gambar 3.7. Pembuatan rancangan arsitektur mengacu pada dokumentasi masing-masing layanan pada situs resmi Google Cloud dan dibuat menggunakan *website* draw.io.

a. Desain Arsitektur *Deployment* Compute Engine

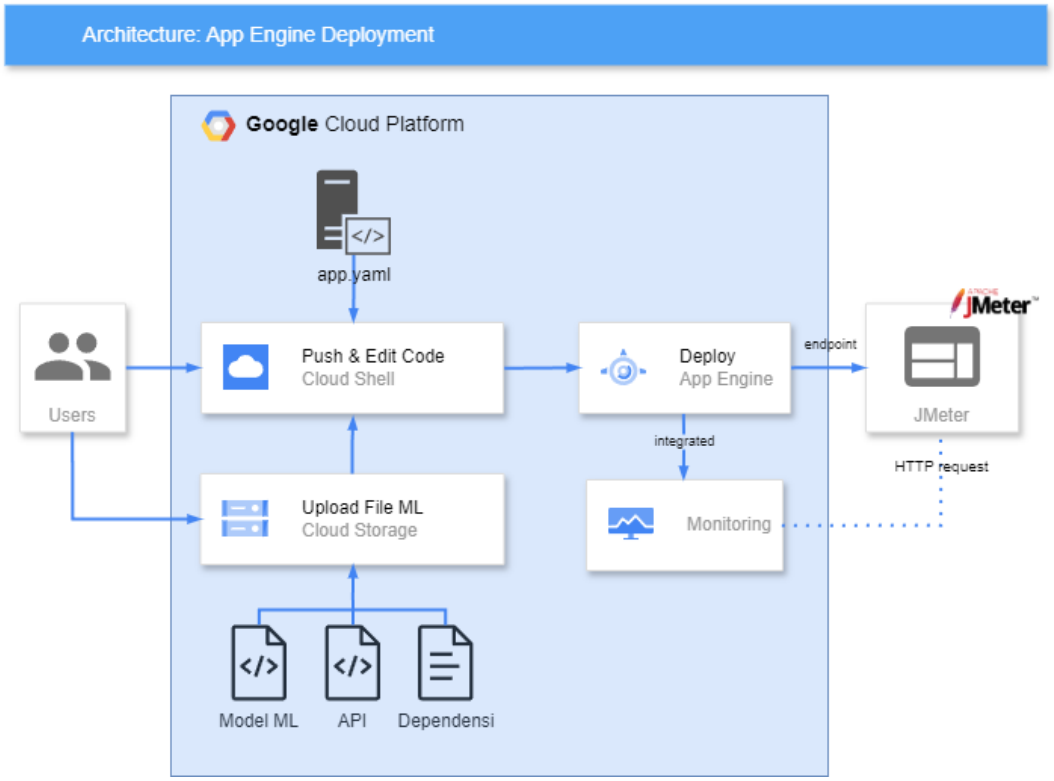
Desain arsitektur *deployment* model *machine learning* pada layanan Compute Engine disajikan pada Gambar 3.2.



Gambar 3.2 Desain Arsitektur *Deployment* Compute Engine

b. Desain Arsitektur *Deployment* App Engine

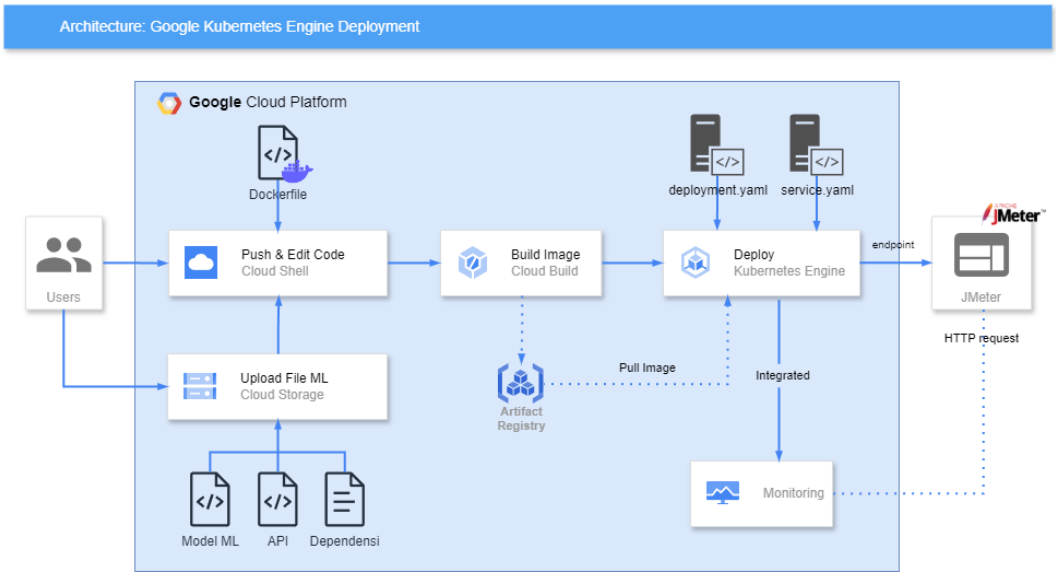
Desain arsitektur *deployment* model *machine learning* pada layanan App Engine disajikan pada Gambar 3.3.



Gambar 3.3 Desain Arsitektur *Deployment* App Engine

c. Desain Arsitektur *Deployment* Kubernetes Engine

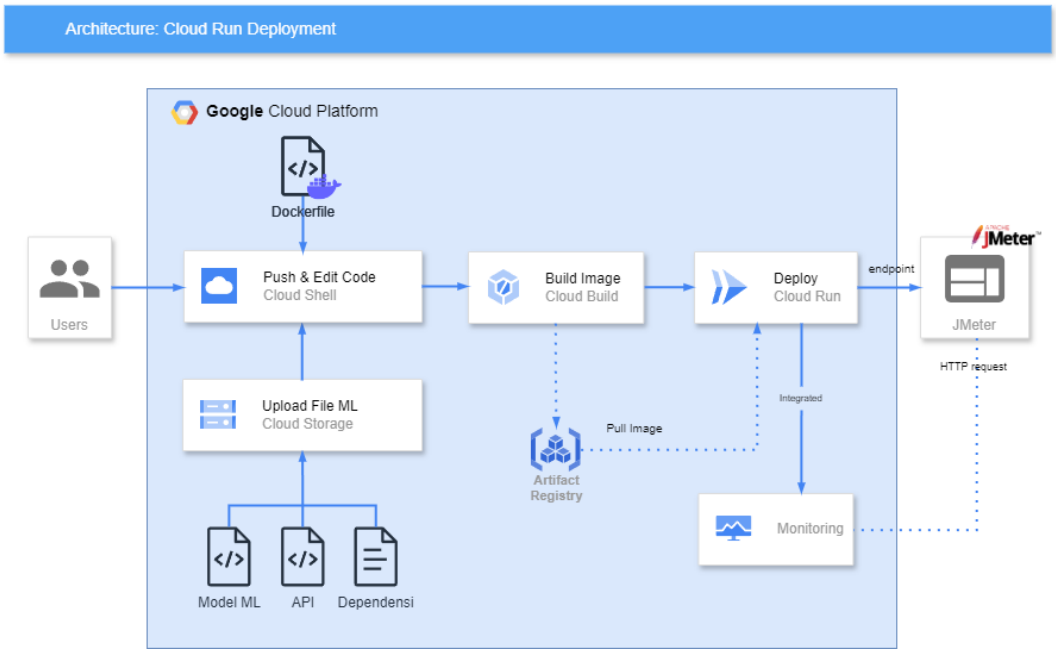
Desain arsitektur *deployment* model *machine learning* pada Kubernetes Engine disajikan pada Gambar 3.4.



Gambar 3.4 Desain Arsitektur *Deployment* Kubernetes Engine

d. Desain Arsitektur *Deployment* Cloud Run

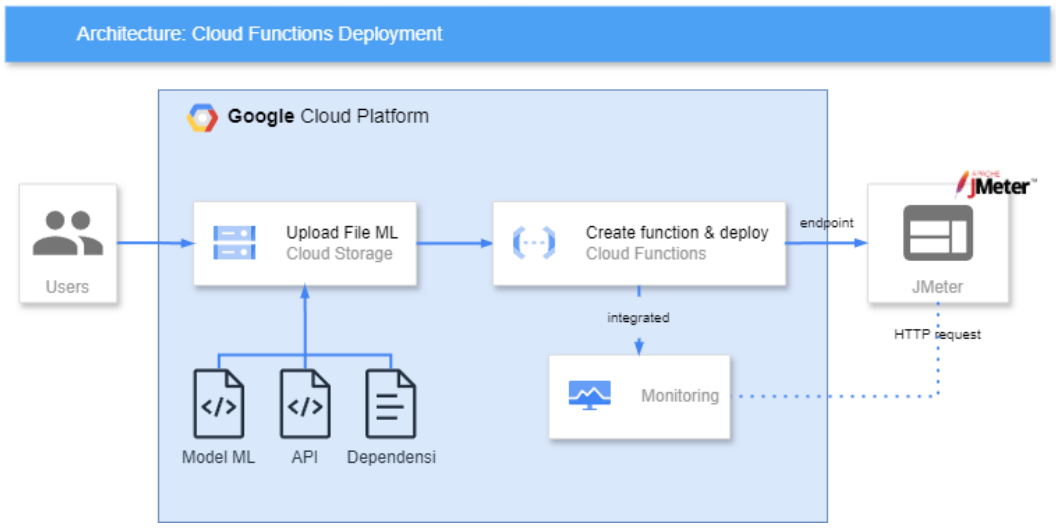
Desain arsitektur *deployment* model *machine learning* pada layanan Cloud Run disajikan pada Gambar 3.5.



Gambar 3.5 Desain Arsitektur *Deployment* Cloud Run

e. Desain Arsitektur *Deployment* Cloud Functions

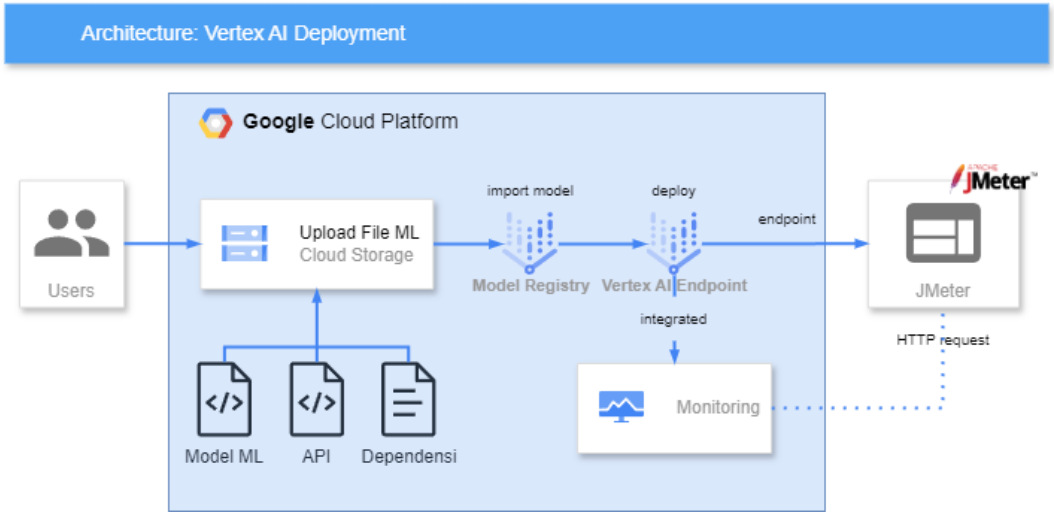
Desain arsitektur *deployment* model *machine learning* pada Cloud Functions disajikan pada Gambar 3.6.



Gambar 3.6 Desain Arsitektur *Deployment* Cloud Functions

f. Desain Arsitektur *Deployment* Vertex AI

Desain arsitektur deployment model *machine learning* pada layanan Vertex AI disajikan pada Gambar 3.7.



Gambar 3.7 Desain Arsitektur *Deployment* Vertex AI

3.2.4.3 Konfigurasi Sistem

Konfigurasi sistem dari masing-masing layanan *server-based* dan *serverless* disajikan pada Tabel 3.1. Model *machine learning* yang akan di-*deploy sudah* dilatih secara lokal menggunakan algoritma Decision Tree pada *framework* scikit-learn versi 1.3.0 dengan keluaran *file* berformat .joblib dan tingkat akurasi prediksi terbaik sebesar 98,41%. Model tersebut berisi *training* code. Lalu, API untuk menangani HTTP *request* dan *response* menggunakan python 3.11.

Tabel 3.1 Konfigurasi Sistem Setiap Layanan

Konfigurasi Sistem	***Server-based***	***Serverless***				
	Compute Engine	**App Engine**	**Kubernetes Engine**	**Cloud Run**	**Cloud Functions**	**Vertex AI**
Instance Type	e2-standard-4	-	e2-standard-4	-	-	e2-standard-4
Number of vCPU	4	4	4	4	4	4
Memory	16 GB	16 GB	16 GB	16 GiB	16 GiB	16 GB

Konfigurasi sistem pada Tabel 3.1 merupakan spesifikasi *resource* setiap layanan untuk *deployment* model *machine learning* pada *penelitian* ini. Mengacu pada dokumentasi GCP, *instance type* E2 memberikan layanan yang seimbang antara harga dan kinerja (*price-to-performance*) serta cocok digunakan untuk pengujian aplikasi. *Instance type* e2-standard-4 secara *default* memiliki jumlah vCPU 4 dan memori sebesar 16 GB. Untuk menghasilkan pengukuran performa yang konsisten pada semua layanan dan mendapatkan gambaran performa layanan yang akurat, jumlah vCPU dan besar memori disamakan pada semua layanan. Selain itu, dalam beberapa penelitian yang juga mengkomparasi layanan *cloud*, 4 vCPU dan 16 GB memori digunakan sebagai spesifikasi layanan, salah satunya adalah penelitian yang dilakukan oleh Dianti (2023) sehingga menjadi acuan dalam penetapan kebutuhan spesifikasi sistem pada penelitian ini.

3.2.5 Pelaksanaan Penelitian

Pelaksanaan penelitian meliputi *deployment* model dan pengujian untuk mengukur *performa* masing-masing layanan *server-based* dan *serverless* di GCP. Implementasi yang dilakukan berbeda-beda tergantung dari *requirement* pada setiap layanan dan proses yang dilakukan. Setelah dilakukan *deployment* model pada masing-masing layanan, terdapat *endpoint* hasil *deployment* yang akan diuji melalui JMeter sebanyak 10 set pengujian dengan jumlah setiap set sebanyak 10 kali *request* selama 30 menit terhadap aplikasi melalui *endpoint* tersebut. Sehingga, total pengujian akan terdiri dari 100 kali *request*. Setelah itu, proses *request* akan dimonitoring menggunakan Cloud Monitoring dan masing-masing data parameter performa akan dikumpulkan.

3.2.5.1 *Deployment* Layanan *Server-Based*

Deployment model *machine learning* pada layanan *server-based*, *yaitu* Compute Engine meliputi pembuatan *Virtual Machine Instance* (VM *Instance*) sebagai *server* sehingga terdapat penggunaan sistem operasi, yaitu Ubuntu 20.04 LTS (*free license*) yang sudah diinstalasikan python versi 3.11 serta instalasi beberapa dependensi, di antaranya:

- Flask versi 2.3.2

- scikit-learn versi 1.3.2
- Gunicorn versi 21.0.0 untuk menangani HTTP *request*

Dependensi tersebut dapat bertambah jika diharuskan *memenuhi requirement* tambahan dalam *deploy* model *machine learning*. Jika aplikasi ingin dijalankan pada *background* atau berjalan selama *instance* aktif, pengguna dapat memanfaatkan sebuah *package* seperti supervisor.

3.2.5.2 *Deployment* Layanan *Serverless*

Deployment model *machine learning* pada layanan *serverless* hanya perlu memenuhi *requirement* seperti *source code*, dependensi, dan lainnya.

a. App Engine

Penelitian ini menggunakan App Engine Flexible Environment. *Deployment* model pada App Engine, baik mode *standard* atau *flexible* harus memenuhi *requirement* yang akan diinput pada *file deployment* app.yaml. *File* ini seminimal-minimalnya berisi *runtime* dari model yang akan di-*deploy*, karena penelitian ini menggunakan python versi 3.11 (di atas versi 3.8) maka *runtime* pada *file* app.yaml adalah Ubuntu 22.

b. Kubernetes Engine

Deployment model pada Kubernetes Engine memanfaatkan *container* sehingga dilakukan pembuatan *file* docker untuk menambahkan *requirement deployment* aplikasi berbasis *machine learning*. *File* docker (*container*) ini akan diunggah menggunakan Cloud Build untuk disimpan di Artifact Registry. Setelah itu, baru dilakukan *deployment* pada Kubernetes Engine disertai konfigurasi Kubernetes melalui *file* deployment.yaml dan service.yaml. Supaya aplikasi dapat diakses, ekspos Kubernetes dengan konfigurasi *load balancing* pada *service* Kubernetes Engine.

c. Cloud Run

Deployment model pada Cloud Run sama seperti Kubernetes Engine, memanfaatkan *container* untuk *deploy* aplikasi sehingga dilakukan pembuatan Dockerfile. Perbedaan terletak pada *file deployment* yang tidak menggunakan

deployment.yaml dan service.yaml seperti Kubernetes Engine. Selain itu, setelah di-*deploy*, *endpoint* aplikasi akan muncul tanpa harus diekspos terlebih dahulu.

d. Cloud Functions

Deployment model pada Cloud Functions memanfaatkan *fuction* yang ditambahkan pada *file* API untuk menangani *request* dan *response* melalui fungsi. Pada Cloud Functions, terdapat dependensi tambahan yaitu functions-framework (v.3) dan penambahan kode fungsi pada file API, yaitu "@functions_framework.http" yang diletakkan pada kode yang digunakan untuk *request* data. *File* API yang digunakan juga harus bernama "main.py" supaya dapat dijalankan. Setelah di-*deploy*, akan muncul *endpoint* yang menjadi *trigger* terhadap *function*. Pada penelitian ini, *trigger* yang digunakan adalah *trigger* HTTP untuk mendapatkan respons dari HTTP *request*.

e. Vertex AI

Deployment model pada Vertex AI memanfaatkan Model Registry untuk mendaftarkan model yang akan di-*deploy*. Jika model dilatih secara eksternal, maka model harus diimport terlebih dahulu ke Model Registry dengan *file* harus diberi nama "model", misalnya model.joblib (sesuai dengan format *framework* pelatihan). Setelah didaftarkan, model dapat di-*deploy* menggunakan Vertex AI *endpoint*.

3.2.5.3 Pengumpulan Data Performa

Pengumpulan data performa dilakukan menggunakan Cloud Monitoring dan akan diambil sebanyak 30 data untuk dikomparasi. Jumlah data ini mengacu pada teori Roscoe (dalam Sugiyono (2013), yang memberikan salah satu saran mengenai ukuran sampel yaitu bahwa sampel yang layak dalam penelitian berjumlah antara 30 sampai dengan 500 sampel. Berdasarkan teori tersebut, jumlah minimal sampel yang dapat diteliti adalah 30 sampel. Oleh karena itu, penelitian ini akan mengambil sampel data sebanyak 30 data untuk masing-masing parameter yang diukur.

Pengambilan data dilakukan menggunakan *trimming data* (menghapus data) melalui deteksi *outlier*. *Outlier* merupakan penyimpangan suatu pengamatan dengan pengamatan lain (Hawkins, 1980). *Outlier* dapat diidentifikasi menggunakan metode *boxplot* yang memanfaatkan nilai kuartil dan jangkauan

interkuartil data (IQR) (Azzahro & Sofro, 2023). *Outlier* pada metode kuartil ditemukan jika posisi data kurang dari batas kuartil pertama (Q_1) atau lebih besar dari batas kuartil ketiga (Q_3) (Sihombing dkk., 2023). Cara menghitung kuartil dan interkuartil disajikan pada Persamaan (3.1), (3.2), dan (3.3).

$$Q_1 = \frac{n+1}{4} \tag{3.1}$$

$$Q_3 = \frac{3(n+1)}{4} \tag{3.2}$$

$$IQR = Q_3 - Q_1 \tag{3.3}$$

dimana,

n = jumlah atau banyaknya data

Setelah mengetahui nilai IQR, hitung nilai pagar atas (batas pada Q_3) dan pagar bawah (batas pada Q_1) menggunakan rumus yang disajikan pada Persamaan (3.4) dan (3.5).

$$Pagar\ bawah = Q_1 - (1{,}5 \times IQR) \tag{3.4}$$

$$Pagar\ atas = Q_3 + (1{,}5\ \times IQR) \tag{3.5}$$

Nilai *outlier* akan berada pada posisi kurang dari nilai pagar bawah dan lebih dari nilai pagar atas. Jika tidak ada *outlier* dalam data, data dapat diambil dengan mengeliminasi nilai yang memiliki jarak maksimum dengan rata-rata untuk menemukan data yang ekstrem karena rata-rata merupakan representasi dari sekumpulan data. Namun, data dengan jarak yang dekat dengan rata-rata juga dapat dieliminasi jika data ekstrem memiliki karakteristik khusus yang dapat memengaruhi analisis sehingga tidak dapat dieliminasi. Eliminasi tersebut disesuaikan dengan karakteristik dan tujuan dari analisis yang dibuat. Eliminasi penentuan jarak data ini mengacu pada teknik *sampling purposive* yang sampelnya ditentukan secara subjektif melalui pertimbangan peneliti (Sahabuddin dkk., 2021).

3.2.6 Analisis Data

Analisis data yang digunakan pada penelitian ini adalah analisis statistik deskriptif dan analisis komparatif. Penelitian ini akan menyajikan hasil pengukuran performa setiap layanan menggunakan grafik dan tabel hasil perhitungan rata-rata (mean) masing-masing performa yang akan dikomparasi. Rumus mean disajikan dalam persamaan (3.6).

$$Rata - rata\ (\bar{x}) = \frac{total\ hasil\ repetisi}{jumlah\ repetisi} \tag{3.6}$$

Sebelum melakukan perhitungan rata-rata, hasil performa masing-masing layanan disajikan berdasarkan parameter-parameternya lalu dilakukan komparasi dan hasil pengukuran performa akan divalidasi kualitasnya menggunakan standardisasi QoS TIPHON. Selain pengukuran performa, informasi mengenai biaya dan *developer experiences* akan menjadi pembanding dan dianalisis bersamaan dengan hasil pengukuran performa layanan.

3.2.7 Penarikan Kesimpulan

Pada tahap ini, hasil analisis data akan disimpulkan untuk menemukan jawaban dari rumusan masalah penelitian. Setelah analisis data yang diperoleh melalui pengukuran performa dan parameter lain masing-masing layanan *server-based* dan *serverless*, akan ditarik kesimpulan layanan mana yang terbaik dalam melakukan *deployment* model *machine learning*.