BAB 2
LANDASAN TEORI

2.1 Kementerian PUPR

Kementerian PUPR (Pekerjaan Umum dan Perumahan Rakyat) merupakan sebuah Lembaga Pemerintah yang berlokasi di Jl. Pattimura No. 20 Kebayoran Baru Jakarta Selatan 12110 / (021) 158. Website mengenai Kementerian PUPR dapat diakses melalui tautan https://pu.go.id.

Berdasarkan Peraturan Presiden Nomor 27 Tahun 2020 tentang Kementerian Pekerjaan Umum dan Perumahan Rakyat, Kementerian PUPR mempunyai tugas menyelenggarakan urusan pemerintahan di bidang pekerjaan umum dan perumahan rakyat untuk membantu Presiden dalam menyelenggarakan pemerintahan negara.

Dalam melaksanakan tugas sebagaimana dimaksud di atas, Kementerian Pekerjaan Umum dan Perumahan Rakyat menyelenggarakan fungsi:

1. Perumusan, penetapan, dan pelaksanaan kebijakan di bidang pengelolaan sumber daya air, penyelenggaraan jalan, penyelenggaraan sistem penyediaan air minum, pengelolaan air limbah domestik, pengelolaan drainase lingkungan, dan pengelolaan persampahan, penataan bangunan gedung, pengembangan kawasan permukiman, pengembangan sarana prasarana strategis, penyelenggaraan perumahan, pelaksanaan pembiayaan infrastruktur pekerjaan umum dan perumahan, serta pembinaan jasa konstruksi;
2. Koordinasi pelaksanaan tugas, pembinaan, dan pemberian dukungan administrasi kepada seluruh unsur organisasi di lingkungan Kementerian Pekerjaan Umum dan Perumahan Rakyat;
3. Pengelolaan barang milik/kekayaan negara yang menjadi tanggung jawab Kementerian Pekerjaan Umum dan Perumahan Rakyat;
4. Pengawasan atas pelaksanaan tugas di lingkungan Kementerian Pekerjaan Umum dan Perumahan Rakyat;
5. Pelaksanaan bimbingan teknis dan supervisi atas pelaksanaan urusan Kementerian Pekerjaan Umum dan Perumahan Rakyat di daerah;

6. Pelaksanaan penyusunan kebijakan teknis dan rencana terpadu program pembangunan infrastruktur pekerjaan umum dan perumahan rakyat berdasarkan pendekatan pengembangan wilayah;
7. Pelaksanaan pengembangan sumber daya manusia di bidang pekerjaan umum dan perumahan rakyat;
8. Pelaksanaan dukungan yang bersifat substantif kepada seluruh unsur organisasi di lingkungan Kementerian Pekerjaan Umum dan Perumahan Rakyat; dan
9. Pelaksanaan fungsi lain yang diberikan oleh Presiden.

Kementerian PUPR mempunyai banyak organisasi yang memiliki tugasnya masing-masing, diantaranya:

1. Sekretariat Jenderal
2. Inspektorat Jenderal
3. Direktorat Jenderal Sumber Daya Air
4. Direktorat Jenderal Bina Marga
5. Direktorat Jenderal Perumahan
6. Direktorat Jenderal Bina Konstruksi
7. Direktorat Jenderal Pembiayaan Infrastruktur Pekerjaan Umum dan Perumahan
8. Badan Pengembangan Infrastruktur Wilayah
9. Badan Pengembangan Sumber Daya Manusia
10. Badan Pengatur Jalan Tol

2.1.1 Direktorat Jenderal Bina Marga

Direktorat Jenderal Bina Marga / Ditjen Bina Marga merupakan sebuah Lembaga pemerintah yang berada dalam naungan Kementerian Pekerjaan Umum dan Perumahan Rakyat (Kementerian PUPR). Direktorat Jenderal Bina Marga sendiri berlokasi di Gedung. Binamarga Lt. 3, Jl. Pattimura No. 20 Kebayoran Baru, Jakarta Selatan / (021) 7203165 dan mempunyai banyak balai yang tersebar di Indonesia.

Website mengenai Direktorat Jenderal Bina Marga dapat diakses melalui tautan https://binamarga.pu.go.id/. Berdasarkan Permen PUPR Nomor 3 Tahun 2019, Direktorat Jenderal Bina Marga mempunyai tugas menyelenggarakan perumusan dan pelaksanaan kebijakan di bidang penyelenggaraan jalan sesuai dengan ketentuan peraturan perundang-undangan. Direktorat Jenderal Bina Marga menyelenggarakan fungsi:

1. Perumusan kebijakan di bidang penyelenggaraan jalan sesuai dengan ketentuan peraturan perundang-undangan;
2. Pelaksanaan kebijakan di bidang penyelenggaraan jalan nasional;
3. Pelaksanaan kebijakan di bidang penguatan konektivitas yang menjadi prioritas nasional;
4. Penyusunan norma, standar, prosedur, dan kriteria di bidang penyelenggaraan jalan;
5. Pemberian bimbingan teknis dan supervisi di bidang penyelenggaraan jalan;
6. Pelaksanaan evaluasi dan pelaporan di bidang penyelenggaraan jalan;
7. Pelaksanaan administrasi Direktorat Jenderal Bina Marga; dan
8. Pelaksanaan fungsi lain yang diberikan oleh Menteri Pekerjaan Umum dan Perumahan Rakyat.

Berdasarkan RPJMN 2020-2024, Direktorat Jenderal Bina Marga memiliki arah kebijakan terkait penyelenggaraan jalan seperti:

a) Preservasi Jalan

Peningkatan kualitas kemantapan jalan dilaksanakan melalui pemenuhan kebutuhan pemeliharaan jalan, termasuk pemeliharaan rutin jalan serta pemenuhan kelengkapan jalan. Pemeliharaan jalan ini bertujuan untuk meningkatkan kualitas jalan nasional yang diukur dari rata-rata nilai kekasaran jalan (IRI), indeks perkerasan jalan (PCI), umur struktur jalan, dan drainase jalan. Pemenuhan kebutuhan pemeliharaan jalan didorong melalui perbaikan tata kelola penyelenggaraan jalan yang memprioritaskan kegiatan pemeliharaan rutin, berkala, rehabilitasi, dan rekonstruksi.

b) Pembangunan Jalan dan Jembatan

Peningkatan ketersediaan jaringan jalan yang mendukung pengembangan wilayah dilaksanakan melalui pembangunan jalan pada jalan lintas utama pulau, jalan yang mendukung kawasan industri dan pariwisata prioritas, jalan akses ke simpul transportasi prioritas, jalan lingkar/trans pulau terluar dan jalan akses mendukung wilayah 3T dan kawasan perbatasan. Sebagai contoh, penyelesaian jalan tol Trans Sumatera yang menghubungkan Aceh – Lampung menjadi program prioritas untuk menurunkan waktu tempuh dan menyediakan akses ke pelabuhan utama Kuala Tanjung sehingga mendorong berkembangnya kawasan industri di sepanjang koridor tersebut. Pembangunan jalan akses ke simpul prioritas diarahkan untuk menjamin kemanfaatan infrastruktur secara optimal seperti pelabuhan dan bandara. Diperlukan penguatan koordinasi perencanaan pembangunan simpul transportasi dan akses jalan dengan Kementerian/Lembaga terkait.

c) Penataan Jalan Nasional di Kawasan Perkotaan

Rencana peningkatan pelayanan jaringan jalan perkotaan melalui pemeliharaan jalan perkotaan yang efektif dan efisien serta peningkatan kapasitas jalan perkotaan berkeselamatan yang mendukung keterpaduan transportasi masal perkotaan, diantaranya: (i) Revitalisasi jaringan jalan perkotaan terutama penataan jalan nasional dengan konsep jalan lingkar agar tidak mengganggu lalu lintas dalam kota; (ii) Penataan simpang sebidang untuk mengurangi kemacetan jalan di persimpangan; (iii) Dukungan jalan pada pengembangan moda transportasi angkutan masal; (iv) Pengembangan jaringan jalan baru di wilayah perkotaan dengan memanfaatkan ROW jalan eksisting berupa elevated/submerged road untuk meminimalisir pengadaan lahan yang biayanya sangat tinggi di perkotaan; (v) Perbaikan kondisi jalan perkotaan dengan menyediakan sistem drainase yang lebih untuk mencegah banjir dan trotoar yang ramah pejalan kaki; (vi) Optimalisasi metode pembangunan simpang tak sebidang (fly over) terutama lintasan kereta api dengan pemanfaatan lahan seminimal mungkin.

d) Pemenuhan Gap Funding melalui Skema Pembiayaan Alternatif

Skema Kerjasama Pemerintah dan Badan Usaha (KPBU) Availability Payment (AP) yang berbasis kinerja serta Program Hibah Jalan Daerah yang mendorong kinerja pemeliharaan jalan perlu diperluas. Skema KPBU-AP juga mendukung kualitas jalan melalui keterpaduan penyelenggaraan jalan dan pengoperasian jembatan timbang untuk mengendalikan perilaku pembebanan berlebih di jalan (Over Dimension Over Loading)

2.1.1.1 Balai Jembatan

Balai Jembatan merupakan unit kerja / unit pelaksana teknis di bawah Direktorat Jenderal Bina Marga dan bertanggung jawab kepada Direktur Jenderal Bina Marga melalui Direktur Pembangunan Jembatan, dibentuk pada tahun 2020 melalui Peraturan Menteri PUPR Nomor 16 Tahun 2020 tentang Struktur Organisasi dan Tata Kerja Unit Pelaksana Teknis Kementerian PUPR.

Balai Jembatan berlokasi di Jl. AH. Nasution No.264 Kota Bandung Jawa Barat dengan Kode Pos 40294 / (022) 7802251 / balai.jembatan@pu.go.id. Website mengenai Balai Jembatan Direktorat Jenderal Bina Marga dapat diakses melalui https://binamarga.pu.go.id/balai-sj/

2.1.1.2 BBPJN DKI Jakarta – Jawa Barat

BBPJN DKI Jakarta – Jawa Barat atau Balai Besar Pelaksanaan Jalan Nasional DKI Jakarta – Jawa Barat merupakan salah satu dari 8 (delapan) Balai Besar Pelakasanaan Jalan Tipe A di Indonesia yang bertanggung jawab dalam pengadaan maupun pemeliharaan inftrastruktur jalan dan jembatan. Wilayah kerja BBPJN IV terdiri dari 3 (tiga) provinsi yaitu: Provinsi Banten, DKI Jakarta dan Jawa Barat dan berada di bawah naungan Direktorat Jenderal Bina Marga Kementerian Pekerjaan Umum dan Perumahan Rakyat.

Balai Besar Pelaksanaan Jalan Nasional dibentuk untuk meningkatkan dan mendorong pertumbuhan ekonomi nasional melalui percepatan pembangunan dan prasarana jalan dan jembatan yang handal dan akuntabel guna mewujudkan program pemerintah. Memiliki lokasi yang sama dengan Balai Jembatan, yaitu di Jl. A.H Nasution No. 264, Sindanglaya - Kota Bandung dengan Kode Pos 40294. Website mengenai Balai Besar Pelaksanaan Jalan Nasional DKI Jakarta – Jawa Barat dapat diakses melalui https://binamarga.pu.go.id/balai-dki-jabar/.

2.2 Visi dan Misi Kementerian PUPR

Visi Kementerian Pekerjaan Umum dan Perumahan Rakyat tahun 2015-2019 adalah "Terwujudnya Infrastruktur Pekerjaan Umum dan Perumahan Rakyat yang handal dalam mendukung Indonesia yang berdaulat, mandiri dan berkepribadian berlandaskan gotong royong".

Misi Kementerian Pekerjaan Umum dan Perumahan Rakyat yang merupakan rumusan upaya yang akan dilaksanakan selama periode Renstra 2015 – 2019 dalam rangka mencapai visi serta mendukung upaya pencapaian target pembangunan nasional, berdasarkan mandat yang diemban oleh Kementerian Pekerjaan Umum dan Perumahan Rakyat sebagaimana yang tercantum di dalam Peraturan Pemerintah Nomor 165 Tahun 2014 tentang Penataan Tugas dan Fungsi Kabinet Kerja, amanat RPJMN tahap ketiga serta perubahan kondisi lingkungan strategis yang dinamis adalah sebagai berikut:

1. Mempercepat pembangunan infrastruktur sumberdaya air termasuk sumber daya maritim untuk mendukung ketahanan air, kedaulatan pangan, dan kedaulatan energi, guna menggerakkan sektor-sektor strategis ekonomi domestik dalam rangka kemandirian ekonomi;
2. Mempercepat pembangunan infrastruktur jalan untuk mendukung konektivitas guna meningkatkan produktivitas, efisiensi, dan pelayanan sistem logistik nasional bagi

penguatan daya saing bangsa di lingkup global yang berfokus pada keterpaduan konektivitas daratan dan maritim;

3. Mempercepat pembangunan infrastruktur permukiman dan perumahan rakyat untuk mendukung layanan infrastruktur dasar yang layak dalam rangka mewujudkan kualitas hidup manusia Indonesia sejalan dengan prinsip ‘infrastruktur untuk semua’;
4. Mempercepat pembangunan infrastruktur pekerjaan umum dan perumahan rakyat secara terpadu dari pinggiran didukung industri konstruksi yang berkualitas untuk keseimbangan pembangunan antardaerah, terutama di kawasan tertinggal, kawasan perbatasan, dan kawasan perdesaan, dalam kerangka NKRI;
5. Meningkatkan tata kelola sumber daya organisasi bidang pekerjaan umum dan perumahan rakyat yang meliputi sumber daya manusia, pengendalian dan pengawasan, kesekertariatan serta penelitian dan pengembangan untuk mendukung fungsi manajemen meliputi perencanaan yang terpadu, pengorganisasian yang efisien, pelaksanaan yang tepat, dan pengawasan yang ketat.

2.2.1 Visi dan Misi Direktorat Jenderal Bina Marga

Direktorat Jenderal Bina Marga memiliki visi yaitu “Direktorat Jenderal Bina Marga yang Andal, Profesional, Inovatif, dan Berintegritas dalam menyelenggarakan perumusan dan pelaksanaan kebijakan di bidang penyelenggaraan jalan untuk mendukung terwujudnya visi dan misi presiden yakni Indonesia Maju yang berdaulat, mandiri dan berkepribadian berlandaskan gotong royong”

Adapun Misi dari Direktorat Jenderal Bina Marga adalah:

1. Meningatkan konektivitas dan pelayanan jalan nasional
2. Meningatkan efektifitas penyelenggaraan jalan nasional

2.2.1.1 Visi dan Misi Balai Jembatan

Balai Jembatan memiliki visi yaitu “Balai Jembatan yang andal, profesional, inovatif dan berintegritas dalam menyelenggarakan pengaturan keamanan jembatan khusus dan terowongan jalan untuk mendukung terwujudnya tujuan Direktorat Jenderal Bina Marga; Mewujudkan konektivitas jalan nasional yang andal dan prima dalam mendukung perwujudan Indonesia maju yang berdaulat, mandiri dan berkepribadian berlandaskan gotong royong".

Balai Jembatan juga memiliki misi yaitu “Mewujudkan tertib penyelenggaraan jembatan khusus dan terowongan jalan yang meliputi tahap perencanaan teknis, pelaksanaan/konstruksi, pengelolaan dan pemeliharaan untuk memenuhi daya layan dalam

pengelolaannya; meningkatkan keandalan jembatan khusus dan terowongan jalan, sehingga dapat mencegah atau sekurang-kurangnya mengurangi resiko kegagalan bangunan/pekerjaan konstruksi jembatan dan terowongan jalan."

2.2.1.2 Visi dan Misi BBPJN DKI Jakarta – Jawa Barat

BBPJN DKI Jakarta – Jawa Barat (Balai Besar Pelaksanaan DKI Jakarta – Jawa Barat) memiliki visi yaitu "Terwujudnya Sistem Jaringan Jalan Yang Andal, Terpadu & Berkelanjutan Di Seluruh Wilayah Nasional Untuk Mendukung Pertumbuhan Ekonomi Dan Kesejahteraan Sosial "

BBPJN DKI Jakarta – Jawa Barat memiliki misi yaitu:

1. Mewujudkan jaringan jalan nasional yang berkelanjutan dengan mobilitas, aksesibilitas dan keselamatan yang memadai, untuk melayani pusat-pusat kegiatan nasional, wilayah dan kawasan strategis nasional.
2. Mewujudkan jaringan jalan nasional bebas hambatan antar perkotaan dan di kawasan perkotaan yang memiliki intensitas pergerakan logistik tinggi yang menghubungkan dan melayani pusat-pusat kegiatan ekonomi utama nasional.
3. Memfasilitasi agar kapasitas pemerintah daerah meningkat dalam menyelenggarakan jalan daerah yang berkelanjutan dengan mobilitas, aksesibilitas dan keselamatan yang memadai.

2.3 Logo Kementerian PUPR

Kementerian PUPR memiliki sebuah logo yang dijadikan sebagai lambang identitas pengenalan Lembaga secara umum serta menjadi identitas utama Kementerian PUPR. Berikut adalah logo dari Kementerian PUPR :

Gambar 2.1 Logo Kementerian PUPR

2.3.1 Logo Direktorat Jenderal Bina Marga

Direktorat Jenderal Bina Marga memiliki sebuah logo yang dijadikan sebagai lambang identitas pengenalan lembaga secara umum serta menjadi identitas utama Direktorat Jenderal Bina Marga. Berikut adalah logo dari Direktorat Jenderal Bina Marga

Gambar 2.2 Logo Direktorat Jenderal Bina Marga

2.3.1.1 Logo Balai Jembatan

Balai Jembatan memiliki sebuah logo yang dijadikan sebagai lambang identitas pengenalan lembaga secara umum serta menjadi identitas utama Balai Jembatan. Berikut adalah logo dari Balai Jembatan.

Gambar 2.3 Logo Balai Jembatan

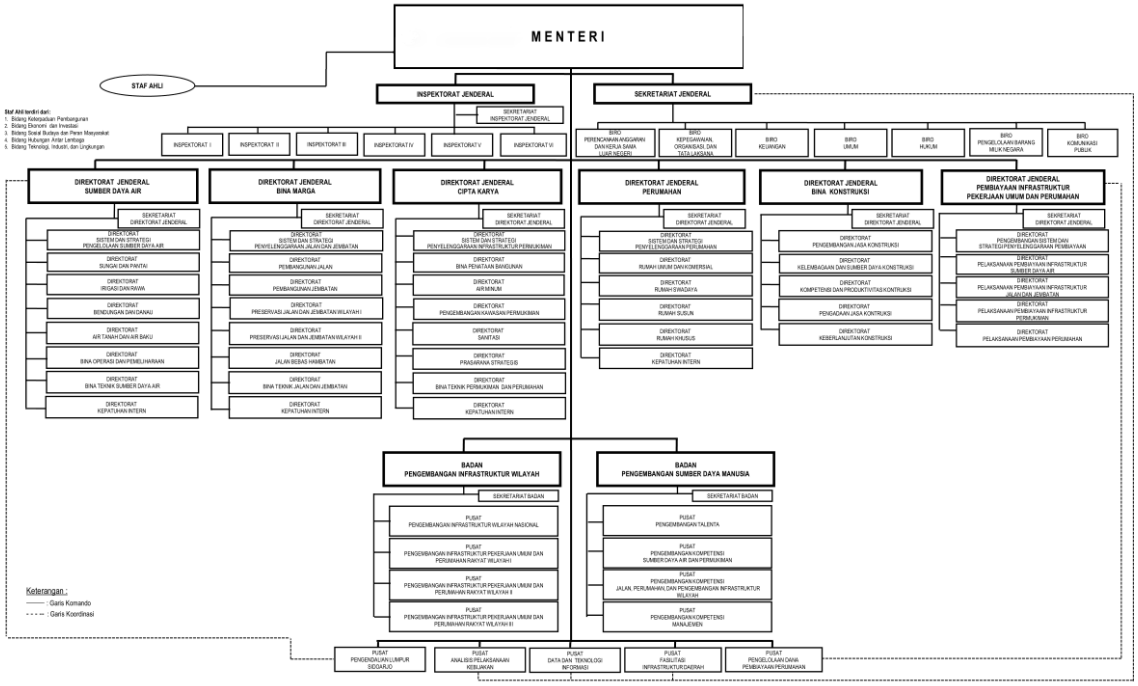
2.3.1.2 Logo BBPJN DKI Jakarta – Jawa Barat

BBPJN DKI Jakarta – Jawa Barat (Balai Besar Penyelenggaraan Jalan Nasional) DKI Jakarta – Jawa Barat memiliki sebuah logo yang dijadikan sebagai lambing identitas pengenalan lembaga secara umum serta menjadi identitas utama BBPJN DKI Jakarta – Jawa Barat. Berikut adalah logo dari BBPJN DKI Jakarta – Jawa Barat

Gambar 2.4 Logo BBPJN DKI Jakarta - Jawa Barat

2.4 Struktur Organisasi Kementerian PUPR

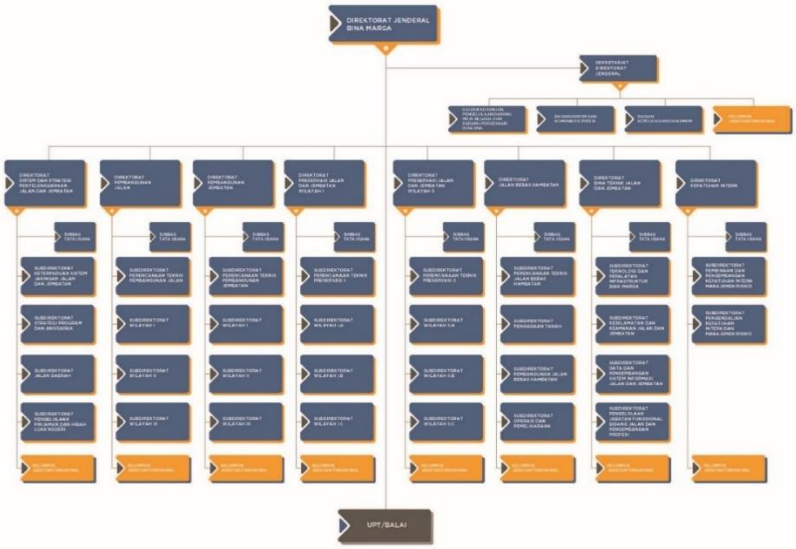
Dalam menjalankan kegiatannya, Kementerian PUPR memiliki struktur organisasi yang memiliki tugas dan kewajiban tersendiri berdasarkan bidangnya untuk membantu Menteri PUPR dalam melakukan pembangunan Indonesia. Berikut merupakan gambar dari struktur organisasi Kementerian PUPR



Gambar 2.5 Struktur Organisasi Kementerian PUPR

2.4.1 Struktur Organisasi Direktorat Jenderal Bina Marga

Dalam menjalankan kegiatannya untuk melakukan pembangunan sampai pengujian infrastruktur jalan maupun jembatan, Direktorat Jenderal Bina Marga memiliki sruktur organisasi yang memiliki tugas dan kewajiban tersendiri pada setiap divisi yang dijalankan. Berikut merupakan gambar dari struktur organisasi Direktorat Jenderal Bina Marga:



Gambar 2.6 Struktur Organisasi Direktorat Jenderal Bina Marga

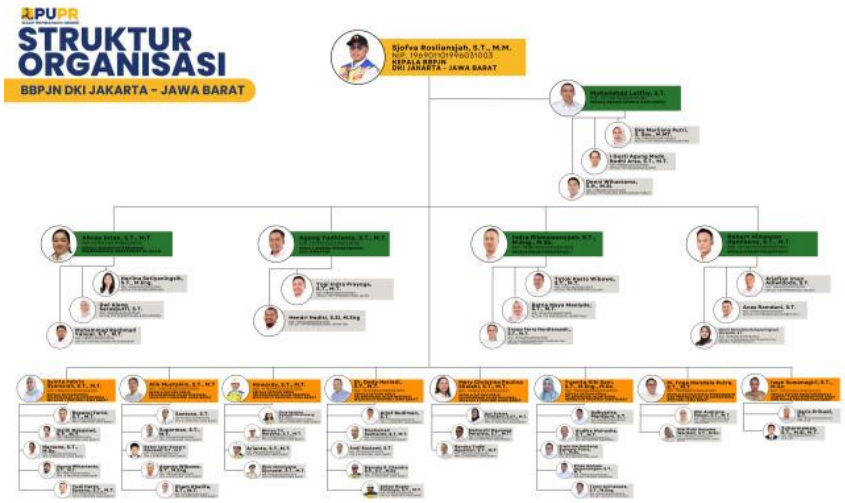
2.4.1.1 Struktur Organisasi Balai Jembatan

Dalam menjalankan kegiatannya untuk melakukan pembangunan sampai pengujian jembatan maupun terowongan, Balai Jembatan memiliki struktur organisasi yang memiliki tugas dan kewajiban tersendiri pada setiap divisi yang dijalankan. Berikut merupakan gambar dari struktur organisasi Balai Jembatan

Gambar 2.7 Struktur Organisasi Balai Jembatan

2.4.1.2 Struktur Organisasi BBPJN DKI Jakarta – Jawa Barat

Dalam menjalankan kegiatannya untuk melakukan pembangunan sampai pengujian infrastruktur nasional DKI Jakarta – Jawa Barat baik jalan maupun jembatan, Balai Besar Pelaksanaan Jalan Nasional DKI Jakarta – Jawa Barat memiliki sruktur organisasi yang memiliki tugas dan kewajiban tersendiri pada setiap divisi yang dijalankan. Berikut merupakan gambar dari struktur organisasi Balai Besar Pelaksanaan Jalan Nasional DKI Jakarta – Jawa Barat:



Gambar 2.8 Struktur Organisasi BBPJN DKI Jakarta - Jawa Barat

2.5 Pembangunan

Pembangunan adalah suatu proses perubahan yang terjadi melalui pencarian perbaikan secara sadar dan terencana. [4] Pembangunan juga dapat diartikan sebagai langkah-langkah seperti kalkulasi dan perencanaan dengan tujuan untuk membangun suatu hal baru dengan tujuan tertentu

2.6 Infrastruktur

Infrastruktur dapat diartikan sebagai sistem yang dapat mendukung sistem sosial dan kehidupan ekonomi yang sekaligus menjadi suatu penghubung sistem lingkungan sebagai dasar dalam mengambil kebijakan atau aturan. [5]

Oleh karenanya, infrastruktur dapat diartikan sebagai semua ragam fasilitas yang dibutuhkan khalayak umum guna mendukung kegiatan dan kehidupan kesehariannya. Infrastruktur juga dapat disimpulkan sebagai segala fasilitas, termasuk dalam bentuk fisik dan non-fisik yang dibangun pemerintah atau pun perorangan dalam rangka memenuhi kebutuhan dasar masyarakat dalam ruang lingkupnya yang meliputi sektor dan aspek sosial-ekonomi. [5]

2.7 Git

Git merupakan sebuah VCS (Version Control System) yang digunakan untuk melacak riwayat perubahan yang dilakukan oleh perorangan dalam tim dalam proyek. Pengguna git dapat melacak riwayat tentang: [6]

1. Perubahan apa saja yang dibuat?
2. Siapa yang membuat perubahan?
3. Kapan perubahan dibuat?
4. Mengapa perubahan diperlukan?

VCS memberikan setiap kontributor pandangan yang terpadu dan konsisten terhadap sebuah proyek, memunculkan pekerjaan yang sudah berjalan. Melihat riwayat perubahan yang transparan, siapa yang membuatnya, dan bagaimana mereka berkontribusi pada pengembangan proyek membantu anggota tim tetap selaras saat bekerja secara mandiri. [6]

Dalam DVCS (Distributed Version Control System), setiap pengembang memiliki salinan lengkap proyek dan riwayat proyek. Tidak seperti *Centralized Version Control System* (sistem kontrol versi terpusat) yang dulu populer, DVCS tidak memerlukan koneksi konstan ke repositori pusat [6]. Selain itu git juga merupakan DVCS dan bersifat *open source* yang dapat digunakan dalam proyek pembangunan aplikasi berskala kecil maupun berskala besar.

Repositori, atau proyek Git, mencakup seluruh koleksi file dan folder yang terkait dengan sebuah proyek, bersama dengan riwayat revisi setiap file. Riwayat file muncul sebagai snapshot dalam waktu yang disebut *commit*. Commit dapat diatur ke dalam beberapa jalur pengembangan yang disebut *branch* (cabang). Karena Git adalah DVCS, repositori adalah unit mandiri dan siapa pun yang memiliki salinan repositori dapat mengakses seluruh basis kode dan riwayatnya. Dengan menggunakan baris perintah atau antarmuka yang mudah digunakan lainnya, repositori Git juga memungkinkan untuk: interaksi dengan riwayat, mengkloning repositori, membuat cabang, melakukan *commit*, menggabungkan, membandingkan perubahan di seluruh versi kode, dan banyak lagi. Berikut ini adalah beberapa contoh command git secara umum [6].

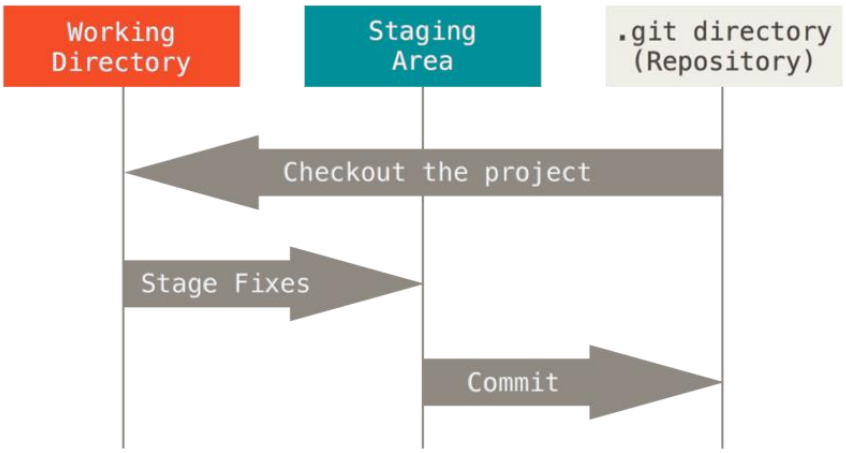
1. *git init* berfungsi untuk menginisialisasi repositori Git yang baru dan mulai melacak direktori yang sudah ada. Serta menambahkan subfolder tersembunyi di dalam direktori yang sudah ada dan menampung struktur data internal yang diperlukan untuk kontrol versi. [6].
2. *git clone* berfungsi untuk membuat salinan lokal dari sebuah proyek yang sudah ada dari jarak jauh. Klon tersebut mencakup semua berkas, riwayat, dan cabang (*branch*) proyek. [6].
3. *git add* berfungsi untuk menambahkan tahapan perubahan. Git melacak perubahan pada basis kode pengembang (*developer*), tetapi perlu dilakukan tahap *staging* (dan

pengambilan cuplikan perubahan untuk memasukkannya ke dalam riwayat proyek. Perintah ini melakukan penahapan, bagian pertama dari proses dua langkah tersebut. Setiap perubahan yang dipentaskan akan menjadi bagian dari snapshot berikutnya dan menjadi bagian dari sejarah proyek. Melakukan *staging* dan *commit* secara terpisah memberikan kontrol penuh kepada pengembang atas sejarah proyek mereka tanpa mengubah cara mereka membuat kode dan bekerja. [6].

4. *git commit* berfungsi untuk menyimpan snapshot ke riwayat proyek dan menyelesaikan proses pelacakan perubahan. Singkatnya, komit berfungsi seperti mengambil foto. Apa pun yang telah dipentaskan dengan git add akan menjadi bagian dari snapshot dengan git commit. [6].
5. *git status* berfungsi untuk menunjukkan status perubahan sebagai *untracked* (tidak terlacak), *modified* (dimodifikasi), atau *staged* (distaging). [6].
6. *git branch* berfungsi menunjukkan cabang yang sedang dikerjakan secara lokal. [6].
7. *git merge* berfungsi untuk menggabungkan beberapa baris pengembangan menjadi satu. Perintah ini biasanya digunakan untuk menggabungkan perubahan yang dibuat pada dua cabang yang berbeda. Sebagai contoh, seorang pengembang akan melakukan penggabungan ketika mereka ingin menggabungkan perubahan dari cabang fitur ke dalam cabang utama untuk penerapan. [6].
8. *git pull* berfungsi untuk memperbarui garis pengembangan lokal dengan pembaruan dari rekan jarak jauhnya. Pengembang menggunakan perintah ini jika rekan satu tim telah melakukan komit ke sebuah cabang di remote, dan mereka ingin merefleksikan perubahan tersebut di lingkungan lokal mereka. [6].
9. *git push* berfungsi untuk memperbarui repositori jarak jauh dengan komit apa pun yang dibuat secara lokal ke sebuah cabang. [6].

Berikutnya, Git memiliki tiga status utama di mana file yang dikerjakan sedang berada dalam keadaan *modified, staged,* dan *committed* [7]

1. *Modified* yang berarti sudah melakukan modifikasi pada file namun belum melakukan *commit* kedalam database [7]
2. *Staged* yang berarti file sudah ditandai pada versi saat ini untuk nanti dimasukkan *commit snapshot* [7]
3. *Commited* yang berarti file sudah disimpan secara aman didalam database [7]



Gambar 2.9 Working tree, staging area, dan Git directory *[7]*

Berdasarkan gambar 2.9 diatas, berikut adalah penjelasan secara deskriptif dari *working tree*, *staging area* dan *Git directory*

1. *Working tree* adalah satu pemeriksaan dari satu versi proyek. File-file ini diambil dari basis data terkompresi di direktori Git dan ditempatkan pada disk untuk digunakan atau dimodifikasi. [7]
2. *Staging area* adalah sebuah file, yang umumnya terdapat di direktori Git yang menyimpan informasi tentang apa yang akan dimasukkan ke dalam proses commit berikutnya. Nama teknisnya dalam bahasa Git adalah “index”, tetapi frasa “staging area” juga bisa digunakan. [7]
3. *Git directory* adalah tempat bagi Git untuk menyimpan metadata dan database objek untuk proyek pada proyek. Ini adalah bagian terpenting dari Git, dan inilah yang disalin ketika mengkloning repositori dari komputer lain. [7]

2.8 DevOps

DevOps adalah kombinasi dari filosofi budaya, praktik, serta alat yang dapat digunakan untuk meningkatkan kemampuan organisasi dalam menghadirkan aplikasi, mengembangkan serta meningkatkan kualitas produk aplikasi maupun layanan secara cepat daripada menggunakan metode pembangunan perangkat lunak tradisional dan proses manajemen infrastruktur secara manual. [8]. Dengan kata lain, DevOps adalah sebuah metode pembangunan perangkat lunak yang menggabungkan *Development* dan *Operations* guna meringkas dan mempercepat perilisan aplikasi.

Terdapat beberapa praktik utama bagi organisasi dalam menerapkan DevOps, yang diantaranya adalah *Infrastructure as Code (IaC), Continuous Integration, Continuous Delivery/Deployment (CI/CD), Microservices, Monitoring & Logging,* serta *Communication & Collaboration* [8] . Sumber lain mengatakan bahwa, DevOps adalah sebuah istilah dalam sebuah integrasi yang mengacu pada *development* dan *operations* serta secara budaya yang mewakili interkoneksi antara *developer* dan *operator* [9]. Pada penelitian ini, praktik DevOps yang akan dibahas adalah *Infrastructure as Code (IaC)* dan *CI/CD*.

2.8.1 Infrastructure as Code

Infrastructure as Code (IaC) adalah kemampuan untuk menyediakan dan mendukung infrastruktur komputasi menggunakan kode, bukan proses dan pengaturan manual. Ini mengotomasikan manajemen infrastruktur sehingga developer dapat fokus pada pembangunan dan peningkatan aplikasi daripada mengelola lingkungan. Organisasi dapat menggunakan Infrastructure as Code dengan tujuan untuk mengendalikan biaya, mengurangi risiko, dan merespons peluang bisnis baru dengan cepat. [9]

Sumber lain menjelaskan bahwa *Infrastructure as Code* (IaC) adalah sebuah metodologi yang digunakan dalam pembangunan perangkat lunak yang memungkinkan untuk melakukan penyediaan dan pengelolaan infrastruktur menggunakan file definisi yang dapat dibaca oleh mesin dan bukan melalui konfigurasi perangkat keras fisik atau alat konfigurasi interaktif. [11]

2.8.1.1 Keuntungan Infrastructure as Code

Terdapat beberapa keuntungan dalam mengimplementasikan Infrastructure as Code (IaC), diantaranya adalah :

1. Meningkatkan kecepatan, karena membangun dan melakukan manajemen infrastruktur komputasi dilakukan dengan menggunakan script. [10]
2. Sifatnya yang konsisten, karena biasanya dalam manajemen infrastruktur komputasi secara manual kadang bisa saja terjadi ketidaksesuaian. [10]
3. Bisa di pertanggungjawabkan, karena *Infrastructure as Code* berbentuk seperti source code yang jika terdapat perubahan dalam infrastruktur komputasi mudah untuk di lacak. [10]

2.8.2 CI/CD

CI/CD merupakan singkatan dari *Continuous Integration* dan *Continuous Delivery/Deployment*, bertujuan untuk merampingkan dan mempercepat siklus pengembangan perangkat lunak. [11]

Continuous Integration (CI) mengacu pada praktik mengintegrasikan perubahan kode secara otomatis dan sering ke dalam repositori kode bersama. *Continuous Delivery* dan/atau *Deployment (CD*) adalah proses 2 bagian yang mengacu pada integrasi, pengujian, dan pengiriman perubahan kode. *Continuous Delivery* berhenti pada penerapan produksi otomatis, sedangkan *Continuous Deployment* secara otomatis merilis pembaruan ke dalam lingkungan produksi. [11]

Secara keseluruhan, praktik-praktik yang saling terhubung ini sering disebut sebagai " CI/CD pipeline" dan didukung oleh tim *development* dan *operation* yang bekerja sama menggunakan pendekatan DevOps atau *Site Reliability Engineering (SRE)*. CI/CD dapat membantu organisasi untuk menghindari *bug* maupun kegagalan kode dengan tetap mempertahankan siklus pengembangan dan pembaruan perangkat lunak yang berkesinambungan. [11]

Gambar 2.10 CI/CD Pipeline

Dapat disimpulkan bahwa CI/CD Pipeline terdiri dari Continuous Integration (CI) yang merupakan praktik di mana developer secara berkala menggabungkan perubahan kode mereka ke dalam repositori pusat, diikuti dengan pembuatan dan pengujian otomatis. Praktik ini membantu dalam mengidentifikasi masalah integrasi di awal, untuk memastikan bahwa basis kode tetap dalam kondisi yang dapat diterapkan, sedangkan Continuous Deployment (CD) mengotomatiskan pelepasan kode terintegrasi ini ke produksi. Setiap perubahan yang melewati semua tahapan pipeline produksi akan secara otomatis dideploy ke lingkungan produksi dan memastikan bahwa fitur-fitur baru, perbaikan, dan peningkatan dapat dirilis dengan cepat dan berkelanjutan. [12]

2.9 Terraform

Terraform adalah salah satu produk yang dimiliki oleh HashiCorp berupa tool *Infrastructure as Code (IaC)* yang memungkinkan untuk mendefinisikan sumber daya cloud dan on-prem dalam file konfigurasi yang dapat dibaca oleh manusia yang dapat di ubah versinya, di gunakan kembali, dan di bagikan. Terraform juga memungkinkan penggunanya untuk dapat berkolaborasi serta dapat menghubungkannya kedalam Version Control System (VCS) seperti Github, Gitlab dan yang lainnya. Penggunanya dapat menggunakan alur kerja yang konsisten untuk menyediakan dan mengelola semua infrastruktur di sepanjang siklus hidupnya. Terraform dapat mengelola komponen tingkat rendah seperti komputasi, penyimpanan, dan sumber daya jaringan, serta komponen tingkat tinggi seperti entri DNS dan fitur SaaS. [12]

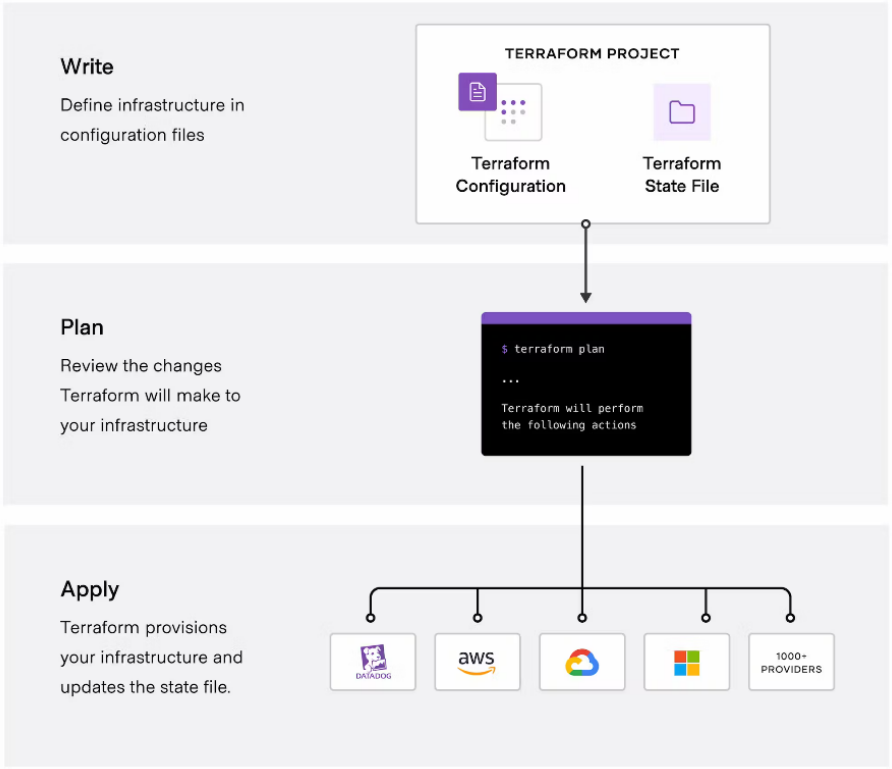
Terraform menciptakan dan mengelola sumber daya pada platform cloud dan layanan lainnya melalui Application Programming Interface (API) mereka. Penyedia memungkinkan Terraform untuk bekerja dengan hampir semua platform atau layanan dengan API yang dapat diakses. [12]

File-file dalam Terraform dibuat dengan menggunakan bahasa konfigurasi HashiCorp Configuration Language (HCL). Sintaksnya bersifat deklaratif dimana setiap blok kode mendefinisikan sumber daya yang akan disediakan. Definisi deklaratif (versus imperatif) ialah memungkinkan pengguna untuk mendefinisikan keadaan yang diinginkan, daripada membuat daftar lengkap dari semua langkah-langkah sementara yang diperlukan untuk mencapai kondisi tersebut. [16]

Inti alur kerja Terraform terdiri dari tiga tahap, yaitu :

1. Write: Menentukan sumber daya, yang mungkin ada di beberapa penyedia dan layanan cloud. Misalnya, ketika membuat konfigurasi untuk menerapkan aplikasi pada mesin virtual di jaringan Virtual Private Cloud (VPC) dengan grup keamanan dan penyeimbang beban (load balancing). [12]
2. Plan : Terraform membuat rencana eksekusi yang menjelaskan infrastruktur yang akan dibuat, diperbarui, atau dihancurkan berdasarkan infrastruktur yang ada dan konfigurasi yang dibuat. [12]
3. Apply : Setelah disetujui, Terraform melakukan operasi yang diusulkan dalam urutan yang benar, dengan memperhatikan ketergantungan sumber daya apa pun. Misalnya, jika

memperbarui properti VPC dan mengubah jumlah mesin virtual di VPC tersebut, Terraform akan membuat ulang VPC sebelum menskalakan mesin virtual. [12]



Gambar 2.11 Alur Kerja Terraform

2.10 Cloud Computing

Cloud computing, atau komputasi awan, merujuk pada model pengelolaan sumber daya komputasi melalui jaringan internet. Dalam model ini, sumber daya seperti server, penyimpanan data, jaringan, dan perangkat lunak diakses dan digunakan secara virtual melalui internet, tanpa memerlukan kepemilikan atau pengelolaan langsung atas perangkat keras fisik. Dengan kata lain, cloud computing mengizinkan pengguna untuk “menyewa” atau “mengakses” sumber daya komputasi dari penyedia layanan melalui jaringan. [14]

Konsep cloud computing dapat diibaratkan sebagai penyediaan layanan seperti listrik atau air. Sebagaimana Anda tidak perlu memiliki pembangkit listrik sendiri untuk mendapatkan listrik, begitu pula dengan cloud computing, Anda tidak perlu memiliki server atau infrastruktur komputer fisik untuk memanfaatkan layanan ini. Semua itu tersedia dan dapat diakses melalui “awan” atau jaringan internet. [14]

Selain itu, Cloud Computing adalah metode komputasi komputer yang menggunakan Internet sebagai terminal utama dan mengelola perangkat lunak berikut infrastruktur sebagai sebuah layanan (*as a services*). Dengan cloud computing, program perangkat lunak yang digunakan tidak disimpan di komputer pengguna, tetapi di server yang dapat diakses melalui Internet, sehingga pengguna dapat mengakses semua layanan (*services*) dan penyimpanan (*storage*) cloud Anda kapan saja dan di mana saja selama memiliki koneksi Internet. [15]

2.10.1 Karakteristik Cloud Computing

National Institute of Standards and Technology (NIST) mengidentifikasi lima karakteristik penting dari komputasi awan / *cloud computing* (Mell & Grance, 2009) sebagai berikut: [16]

1. *On-demand self-service.*

   Pengguna dapat memesan dan mengelola layanan tanpa interaksi manusia dengan penyedia layanan, misalnya dengan menggunakan, sebuah portal web dan manajemen antarmuka. Pengadaan dan perlengkapan layanan serta sumberdaya yang terkait terjadi secara otomatis pada penyedia. [16]
2. *Broad network access*.

   Kemampuan yang tersedia melalui jaringan dan diakses melalui mekanisme standar, yang mengenalkan penggunaan berbagai platform (misalnya, telepon selular, laptop, dan PDA). [16]
3. *Resource pooling*.

   Penyatuan sumber daya komputasi yang dimiliki penyedia untuk melayani beberapa konsumen menggunakan model multi-penyewa, dengan sumberdaya fisik dan virtual yang berbeda, ditetapkan secara dinamis dan ditugaskan sesuai dengan permintaan konsumen. Ada rasa kemandirian lokasi bahwa pelanggan umumnya tidak memiliki kontrol atau pengetahuan atas keberadaan lokasi sumberdaya yang disediakan, tetapi ada kemungkinan dapat menentukan lokasi di tingkat yang lebih tinggi (misalnya, negara, negara bagian, atau datacenter). Contoh sumber daya termasuk penyimpanan, pemrosesan, memori, bandwidth jaringan, dan mesin virtual. [16]
4. *Rapid elasticity*.

   Kemampuan dapat dengan cepat dan elastis ditetapkan. [16]
5. *Measured Service*.

   Sistem komputasi awan secara otomatis mengawasi dan mengoptimalkan penggunaan sumber daya dengan memanfaatkan kemampuan pengukuran (metering) pada beberapa

tingkat yang sesuai dengan jenis layanan (misalnya, penyimpanan, pemrosesan, bandwidth, dan account pengguna aktif). Penggunaan sumber daya dapat dipantau, dikendalikan, dan dilaporkan sebagai upaya memberikan transparansi bagi penyedia dan konsumen dari layanan yang digunakan. [16]

2.10.2 Model Deployment Cloud Computing

Model Deployment (penerapan) Cloud Computing mendefinisikan bagaimana layanan Cloud Computing tersedia untuk penggunanya. Berikut adalah model deployment dalam Cloud Computing :

1. *Public Cloud*, model deployment cloud computing ini mendukung semua pengguna yang ingin menggunakan sumber daya komputasi, seperti perangkat keras (OS, CPU, memori, penyimpanan) atau perangkat lunak (server aplikasi, basis data) dengan basis langganan. [17]
2. *Private Cloud*, biasanya merupakan infrastruktur yang digunakan oleh satu organisasi. Infrastruktur tersebut dapat dikelola oleh organisasi itu sendiri untuk mendukung berbagai kelompok pengguna, atau dapat dikelola oleh penyedia layanan yang menanganinya baik di lokasi maupun di luar lokasi. [17]
3. *Hybrid Cloud,* dalam cloud hybrid, sebuah organisasi memanfaatkan infrastruktur cloud pribadi dan publik yang saling terhubung. Banyak organisasi memanfaatkan model ini ketika mereka perlu meningkatkan infrastruktur IT mereka dengan cepat, seperti ketika memanfaatkan cloud publik untuk menambah kapasitas yang tersedia dalam private cloud. [17]
4. *Community Cloud,* Model penerapan ini mendukung beberapa organisasi yang berbagi sumber daya komputasi yang merupakan bagian dari sebuah komunitas; contohnya adalah universitas yang bekerja sama dalam bidang penelitian tertentu, atau departemen kepolisian di suatu wilayah atau negara bagian yang berbagi sumber daya komputasi. Akses ke lingkungan cloud komunitas biasanya dibatasi untuk anggota komunitas. [17]

2.10.3 Layanan Cloud Computing

Cloud Computing menyediakan berbagai layanan dimana layanan tersebut dapat digunakan oleh pengguna berdasarkan kebutuhan dan sistem yang akan di bangun nantinya. Layanan Cloud Computing yang ada meliputi:

1. *Infrastructure as a Service (IaaS)*

   IaaS menyediakan sumber daya infrastruktur sesuai permintaan, seperti komputasi, penyimpanan, jaringan, dan virtualisasi. Dengan IaaS, penyedia layanan memiliki dan

mengoperasikan infrastruktur, tetapi pelanggan perlu membeli dan mengelola perangkat lunak, seperti sistem operasi, middleware, data, dan aplikasi. [18]

2. *Platform as a Service (PaaS)*

   PaaS menyediakan dan mengelola sumber daya perangkat keras dan perangkat lunak untuk mengembangkan, menguji, mengirim, dan mengelola aplikasi cloud. Penyedia biasanya menawarkan middleware, alat pengembangan, dan basis data cloud dalam penawaran PaaS mereka. [18]

3. *Software as a Service (SaaS)*

   SaaS menyediakan tumpukan aplikasi lengkap sebagai layanan yang dapat diakses dan digunakan oleh pelanggan. Solusi SaaS sering kali hadir sebagai aplikasi siap pakai, yang dikelola dan dipelihara oleh penyedia layanan cloud. [18]

4. *Serverless Computing*

   Komputasi tanpa server (Serverless Computing) dalam model layanan cloud juga disebut *Function as a Service* (FaaS). Ini adalah model layanan cloud yang relatif baru yang menyediakan solusi untuk membangun aplikasi sebagai fungsi sederhana yang dipicu oleh peristiwa tanpa mengelola atau menskalakan infrastruktur apa pun. [18]

2.11 Google Cloud Platform

Google Cloud Platform merupakan layanan Cloud yang berjalan pada infrastruktur yang sama dengan yang digunakan Google secara internal untuk produk konsumen mereka sendiri, seperti Google Search, Gmail, dan YouTube. [19]. Google Cloud Platform termasuk salah satu layanan public cloud seperti halnya Amazon Web Services, Alibaba Cloud, Microsoft Azure, Oracle dan lainnya.

Google Cloud Platform memiliki kurang lebih 150 produk layanan cloud computing yang sesuai dengan kebutuhan industri. Google Cloud Platform mempunyai layanan trial yang berlaku 90 hari beserta dana untuk mengakses layanannya sebesar $300. Selain itu juga, ada beberapa resource (sumber daya) yang selalu gratis/always free yang dengan catatan harus mengikuti kebijakan yang telah ditetapkan.

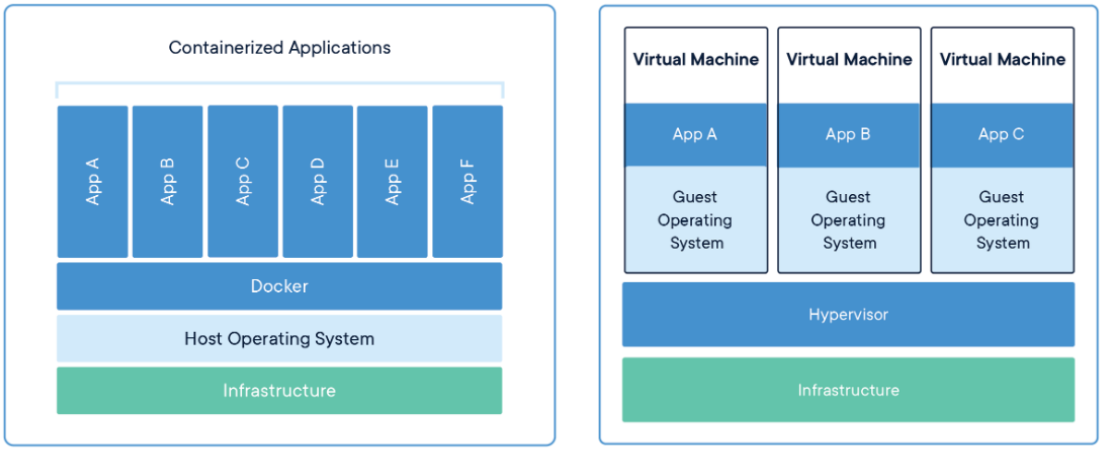
2.11.1 Google Cloud SDK

Google Cloud SDK (Software Development Kit) merupakan sebuah koleksi *tools* yang digunakan untuk membantu developer maupun praktisi dalam berinteraksi dengan Google Cloud API dan manajemen sumber daya di dalam project Google Cloud. Berikut adalah koleksi tools terdapat di Google Cloud SDK :

1. Google Cloud CLI (Command Line Interface), digunakan untuk menginstall dan mengakses resources google cloud melalui command line.
2. Google Cloud Client Libraries, digunakan untuk mengintegrasikan Google Cloud API kedalam bahasa pemrograman yang didukung. Seperti Java, Node JS, Python, Go, Ruby, C#, .NET, ABAP, dan PHP.
3. Documentation, digunakan sebagai dokumentasi dalam menggunakan resource google cloud maupun client library terkait.
4. Cloud Code, merupakan sebuah Integrated Development Environment atau IDE *extension* yang berfungsi sebagai alat bantu dalam proses write, debug, dan deploy aplikasi cloud native.

2.12 Container

Container atau kontainer adalah sebuah unit standar perangkat lunak yang mengemas kode dan semua dependensinya sehingga aplikasi dapat berjalan dengan cepat dan andal dari satu lingkungan komputasi ke lingkungan komputasi lainnya atau dapat disebut juga sebagai sebuah abstraksi pada lapisan aplikasi yang mengemas kode dan dependensinya secara bersamaan. Beberapa container dapat berjalan di mesin yang sama dan berbagi kernel OS dengan kontainer lain, masing-masing berjalan secara terisolasi di ruang pengguna. [20]



Gambar 2.12 Container vs Virtual Machine

Container menggunakan lebih sedikit ruang daripada VM (*container images* biasanya berukuran puluhan MB), yang dapat menangani lebih banyak aplikasi dan membutuhkan lebih sedikit VM dan sistem Operasi. Tidak seperti Virtual machine (VM) yang merupakan abstraksi perangkat keras fisik dengan mengubah satu server menjadi banyak server. Hypervisor memungkinkan beberapa VM berjalan pada satu mesin. Setiap VM menyertakan salinan lengkap sistem operasi, aplikasi, binari, dan pustaka yang diperlukan - yang membutuhkan puluhan GB yang sangat lambat dalam proses boot. [20]

Container bersifat ringan dan berisi semua dependensi yang diperlukan untuk menjalankan aplikasi, sehingga tidak perlu bergantung pada apa yang terinstal pada host. [21]

2.12.1 Docker

Docker merupakan sebuah platform container terbuka untuk mengembangkan, mengirim, dan menjalankan aplikasi dalam lingkungan terisolasi. Docker memungkinkan penggunanya untuk memisahkan aplikasi dari infrastruktur sehingga dapat merilis perangkat lunak dengan cepat. Pengguna Docker dapat mengelola infrastruktur dengan cara yang sama seperti mengelola aplikasi. Yang mana dengan memanfaatkan metodologi Docker untuk mengirim, menguji dan men-deploy kode dapat mengurangi penundaan secara signifikan antara menulis kode dan menjalankannya dalam produksi. [21]

2.13 YAML

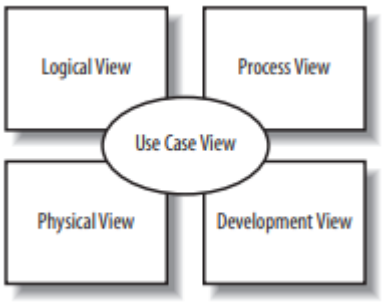
YAML (*.yaml/Yet Another Markup Language) merupakan format serialisasi data untuk bertukar data secara terstruktur antara aplikasi dan layanan perangkat lunak. Yaml mendukung semua tipe data melalui pengumpulan data bersarang dan terdiri dari urutan, skalar, dan pemetaan. [22]

YAML adalah format serialisasi data primer untuk file konfigurasi di banyak alat dan layanan otomatisasi, DevOps, serta Infrastructure as Code (IaC). [22]

2.14 UML

Unified Modeling Language (UML) adalah sebuah bahasa pemodelan standar untuk pengembangan perangkat lunak dan sistem. Dimana bahasa pemodelan yang dimaksud dapat

terdiri dari pseudecode, kode aktual, gambar, diagram, ataupun deskripsi panjang yang membantu menggambarkan perangkat lunak maupun sistem yang akan dibangun [23]. Selain itu terdapat beberapa cara untuk memecah model UML menjadi beberapa perspektif, dimana model tampilan 4+1 dari Philippe Kruchten dapat membantu menunjukkan bagaimana setiap jenis diagram berperan bagian dalam keseluruhan model [23]. Berikut adalah gambar 2.12 yang merupakan model tampilan Philippe Kruchten 4+1



Gambar 2.13 Model Tampilan Philippe Kruchten 4+1

Berikut adalah penjelasan deskriptif dari model tampilan Philippe Kruchten 4+1 dari gambar 2.12

1. Logical View

   Menggambarkan deskripsi abstrak dari bagian-bagian sistem. Digunakan untuk memodelkan terdiri dari apa sebuah sistem dan bagaimana bagian-bagiannya berinteraksi satu sama lain. Logical View ini biasanya berisi *class*, *object*, *state machine*, dan *interaction diagram* [23].
2. Process View

   Menggambarkan proses-proses di dalam sistem dimana Process View ini biasanya berisi *activity diagram* [23].
3. Development View

   Menggambarkan bagaimana bagian-bagian sistem diorganisasikan ke dalam modul dan komponen. Ini sangat berguna untuk mengelola lapisan dalam arsitektur sistem dimana Development View ini biasanya berisi *package* dan *component diagram* [23].
4. Physical View

   Menggambarkan bagaimana desain sistem, seperti yang dijelaskan dalam tiga tampilan sebelumnya, kemudian dihidupkan sebagai sekumpulan entitas dunia nyata. Physical View ini menunjukkan bagaimana bagian-bagian abstrak dipetakan ke dalam sistem akhir yang diterapkan dan biasanya berisi *deployment diagram* [23].

5. Use Case View

   Menggambarkan fungsionalitas sistem yang sedang dimodelkan dari perspektif dunia luar. Ini diperlukan untuk menggambarkan apa yang seharusnya dilakukan oleh sistem. Semua model view lainnya bergantung pada Use Case View untuk memandu model view lainnya dan mengapa model ini disebut model 4+1. Use Case View biasanya berisi *use case diagram*, *use case description*, dan *overview diagram* [23].

Pada UML sendiri terdapat banyak tipe diagram beserta apa yang dapat dimodelkan sesuai dengan kebutuhan perangkat lunak ataupun sistem, seperti pada tabel 2.1

Tabel 2.1 Tipe diagram dan apa yang bisa dimodelkan dalam UML *[23]*

Diagram Type	What can be modeled?
Use Case	*Interactions between your system and users or other external systems. Also helpful in mapping requirements to your systems*
Activity	*Sequential and parallel activities within your system*
Class	*Classes, types, interfaces, and the relationships between them*
Object	*Object instances of the classes defined in class diagrams in configurations that are important to your system*
Sequence	*Interactions between objects where the order of the interactions is important.*
Communication	*The ways in which objects interact and the connections that are needed to support that interaction*
Timing	*Interactions between objects where timing is an important concern*
Interaction Overview	*Used to collect sequence, communication, and timing diagrams together to capture an important interaction that occurs within your system.*
Composite Structure	*The internals of a class or component, and can describe class relationships within a given context.*
Component	*Important components within your system and the interfaces they use to interact with each other*
Package	*The hierarchical organization of groups of classes and components.*