BAB 2
TINJAUAN PUSTAKA

2.1 E-learning

E-Learning adalah sebuah sistem pendidikan yang berbasis teknologi informasi yang memungkinkan siswa dan guru untuk mengakses materi pembelajaran secara *online*. *E-Learning* telah menjadi sangat populer dan digunakan secara luas dalam berbagai bidang pendidikan, termasuk sekolah, universitas, dan organisasi lainnya[1]. *E-Learning* memungkinkan siswa untuk mengakses materi pembelajaran dari mana saja dan kapan saja, sehingga meningkatkan aksesibilitas dan efisiensi dalam proses belajar. Dalam sistem *E-Learning*, materi pembelajaran biasanya disajikan dalam bentuk digital seperti video, gambar, teks, dan *audio*, serta dapat diakses melalui perangkat elektronik seperti komputer, laptop, tablet, atau smartphone. *E-Learning* juga memungkinkan guru untuk memantau kemajuan siswa secara *online* dan memberikan umpan balik yang lebih efektif. Guru dapat menggunakan *E-Learning* untuk membuat materi pembelajaran yang lebih interaktif dan dinamis, serta untuk meningkatkan partisipasi siswa dalam proses belajar.

Dalam beberapa tahun terakhir, *E-Learning* telah berkembang dengan sangat cepat dan telah menjadi bagian penting dari sistem pendidikan di seluruh dunia. Perkembangan *E-Learning* dipengaruhi oleh berbagai faktor, termasuk kemajuan teknologi, perubahan perilaku siswa, dan kebutuhan akan aksesibilitas pendidikan yang lebih luas. Dalam beberapa contoh, *E-Learning* telah digunakan dalam berbagai cara[1].

Dalam beberapa penelitian, *E-Learning* telah ditemukan memiliki dampak positif pada proses belajar-mengajar, seperti meningkatkan efisiensi dan efektivitas, serta meningkatkan partisipasi siswa. Namun, *E-Learning* juga memiliki beberapa tantangan, seperti masalah teknis dan kebutuhan akan infrastruktur yang lebih baik.

Di beberapa tahun mendatang, *E-Learning* diharapkan akan terus berkembang dan menjadi lebih efektif dalam meningkatkan kualitas pendidikan. Dengan demikian, *E-Learning* akan terus berperan penting dalam meningkatkan aksesibilitas pendidikan dan meningkatkan kualitas pendidikan secara global[2].

2.2 Learning Management System (LMS)

Learning Management System (LMS) adalah sebuah *platform* teknologi yang dirancang untuk memanajemen, mengorganisir, dan mengelola proses pembelajaran secara *online*. LMS biasanya digunakan oleh institusi pendidikan, organisasi, dan perusahaan untuk mengelola kursus, materi, dan aktivitas pembelajaran yang dilakukan oleh siswa, guru, dan peserta pelatihan[3]. Fungsi utama LMS meliputi pengelolaan materi, pengelolaan siswa, pengelolaan guru, dan pengelolaan evaluasi. LMS dapat membantu dalam meningkatkan efisiensi dan efektivitas proses pembelajaran dengan fitur-fitur yang lengkap seperti administrasi guru, kalender akademik, memberikan materi dan tugas yang lebih variatif, forum diskusi, pelacakan, dan penilaian. Kelebihan penggunaan LMS antara lain guru mudah dalam pemberian tugas kepada siswa, fitur forum *chat* untuk memudahkan proses diskusi dalam pengerjaan tugas, fitur participation untuk absensi siswa, penilaian kuis langsung tampil di LMS siswa, pembelajaran menjadi lebih menarik dan hemat biaya dan waktu [4].

2.3 Moodle

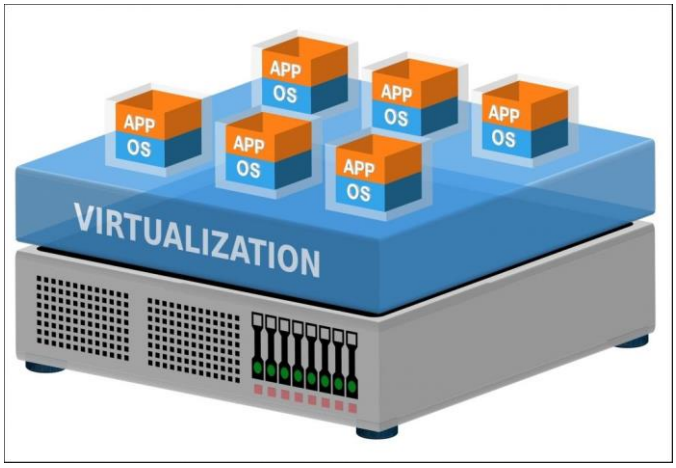
Moodle yang merupakan akronim dari *Modular Object Oriented Dynamic Learning Environment* adalah sebuah program LMS (*learning management system*) gratis dan bersifat *open source* yang dapat mengubah sebuah media pembelajaran ke dalam bentuk *web* yang mana memungkinkan setiap orang dapat memberikan materi-materi pembelajarannya secara *online*[4]. Pengembanganya didesain untuk mendukung kerangka konstruksi sosial (*social construct*) dalam pendidikan. *Moodle* termasuk dalam model CAL+CAT (*Computer Assisted Learning+Computer Assisted Teaching*) yang kemudian disebut dengan LMS (*Learning Management System*). LMS menjadi fasilitas utama dan penting dalam

proses pengajaran dan pembelajaran daring. Kumpulan perangkat lunak yang ada diatur sedemikian rupa pada tingkat individu, ruang belajar, maupun institusi. Ciri khas LMS adalah antara pengguna yang merupakan pengajar dan peserta didik, keduanya harus terkoneksi dengan internet saat menggunakan aplikasi ini.

Moodle juga memungkinkan pengajar untuk mengawasi kemajuan siswa secara lebih efektif dengan menggunakan fitur-fitur seperti analisis log data, yang memungkinkan mereka untuk melihat bagaimana siswa menggunakan *platform* tersebut dan menentukan strategi yang lebih efektif untuk meningkatkan hasil belajar[5]. Selain itu, *Moodle* juga dapat digunakan untuk mengorganisir dan mengelola kegiatan pembelajaran, termasuk membuat dan mengelola kelas, mengupload materi, dan mengawasi kemajuan siswa. *Moodle* juga memungkinkan pengajar untuk mengawasi kemajuan siswa secara lebih efektif dengan menggunakan fitur-fitur seperti analisis *log data,* yang memungkinkan mereka untuk melihat bagaimana siswa menggunakan *platform* tersebut dan menentukan strategi yang lebih efektif untuk meningkatkan hasil belajar

2.4 Virtualisasi

Virtualisasi adalah teknologi yang memungkinkan beberapa sistem operasi atau lingkungan komputasi untuk berjalan pada satu mesin fisik, tanpa memerlukan perangkat keras yang terpisah untuk setiap sistem operasi[6]. Dalam virtualisasi, mesin virtual atau "*virtual machine*" dibuat untuk setiap sistem operasi yang ingin dijalankan, dan setiap mesin virtual ini berjalan seperti mesin fisik, tetapi tidak memiliki akses langsung ke perangkat keras. Virtualisasi memungkinkan penggunaan sumber daya komputasi yang lebih efisien dan meningkatkan keamanan serta skalabilitas sistem[7].



Gambar 2. 1 Arsitektur Virtualisasi

Virtualisasi memungkinkan penggunaan sistem operasi yang berbeda-beda pada mesin fisik yang sama, memungkinkan penggunaan sumber daya yang lebih efektif, dan meningkatkan keamanan dengan cara memisahkan sistem operasi yang berbeda-beda[8]. Dengan demikian, virtualisasi memungkinkan penggunaan sumber daya komputasi yang lebih efisien, meningkatkan keamanan serta skalabilitas sistem, serta memungkinkan penggunaan sistem operasi yang berbeda-beda pada mesin fisik yang sama. Virtualisasi dapat dibagi menjadi dua jenis utama: virtualisasi berbasis *hypervisor* dan virtualisasi berbasis *container*

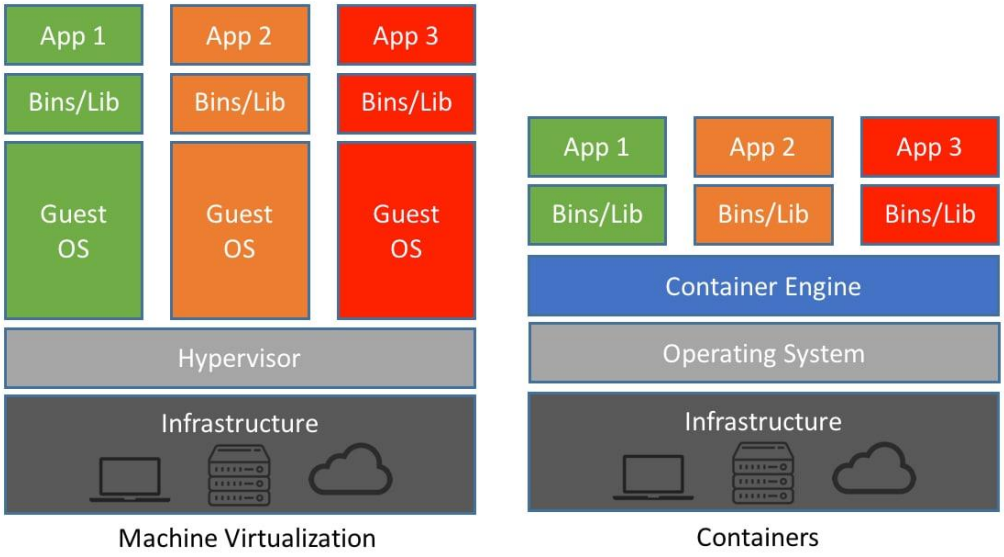
- Virtualisasi Berbasis *Hypervisor*: *Hypervisor* adalah virtualisasi yang memungkinkan suatu mesin untuk menjalankan beberapa mesin virtual dengan sistem operasi yang berbeda. *Hypervisor* berfungsi sebagai lapisan antara sistem operasi host dan sistem operasi guest, memungkinkan penggunaan sumber daya komputasi secara efisien dan meningkatkan keamanan. Contoh software *hypervisor* adalah *VMware ESXi*
- Virtualisasi Berbasis *Container*: Virtualisasi berbasis *container* mengisolasi pada level sistem operasi. *Container* adalah sebuah konsep yang memungkinkan beberapa aplikasi untuk berjalan dalam satu lingkungan operasi yang sama, tetapi dengan menggunakan sistem operasi yang berbeda. *Container* tidak memerlukan *hypervisor* dan dapat berjalan

pada sistem operasi yang sama dengan host. Contoh software *container* adalah *Docker*

Kedua jenis virtualisasi memiliki kelebihan dan kekurangannya sendiri. Virtualisasi berbasis *hypervisor* lebih efektif dalam menghemat sumber daya komputasi dan meningkatkan keamanan, tetapi memiliki *overhead* yang lebih tinggi. Virtualisasi berbasis *container* lebih ringkas dan lebih mudah digunakan, namun memiliki batasan dalam penggunaan sumber daya komputasi.

2.5 Container

Container adalah sebuah teknologi yang memungkinkan pengembangan aplikasi dalam sebuah lingkungan yang terisolasi dan dapat dijalankan secara independen tanpa memerlukan perubahan pada infrastruktur yang lebih besar[9]. Dalam konteks teknologi *Docker*, *container* adalah sebuah aplikasi yang berbasis teknologi *open source* yang memungkinkan *developer* atau siapapun untuk membuat, menjalankan, melakukan percobaan dan meluncurkan aplikasi didalam sebuah *container* yang terisolasi. *Docker* membuat proses pemaketan aplikasi bersama komponennya secara cepat dalam sebuah *container* yang terisolasi, sehingga dapat dijalankan dalam infrastruktur lokal tanpa melakukan perubahan konfigurasi pada *container*. *Container* juga sangat ringan dan cepat jika dibandingkan dengan mesin virtual yang berbasis *hypervisor*.



Gambar 2. 2 Perbedaan Machine Virtulization dan Containers

Virtualisasi *container* dapat dilakukan menggunakan *Docker*, sebuah aplikasi yang berbasis teknologi *open source* yang memungkinkan *developer* atau siapapun untuk membuat, menjalankan, melakukan percobaan dan meluncurkan aplikasi dalam sebuah *container* yang terisolasi[7]. *Docker* membuat proses pemaketan aplikasi bersama komponennya secara cepat dalam sebuah *container* yang terisolasi, sehingga dapat dijalankan dalam infrastruktur lokal tanpa melakukan perubahan konfigurasi pada *container*. *Docker* juga sangat ringan dan cepat jika dibandingkan dengan mesin virtual yang berbasis *hypervisor*.

Penerapan teknologi virtualisasi dan *container* merupakan bagian yang sangat penting dalam penerapan cloud computing, karena sangat berpengaruh pada efisiensi pengelolaan sumber daya infrastruktur cloud computing[10]. Pada konsep virtualisasi, dibutuhkan sebuah sistem yang dapat mempunyai teknik virtualisasi yang mampu menyediakan sebuah sistem dengan performa mendekati atau sama dengan native. Salah satu faktor yang mempengaruhi performansi sebuah teknik virtualisasi yaitu dari sisi keamanannya. Terdapat banyak jenis serangan terhadap sistem computer, yaitu salah satunya *Denial of Service*, jenis serangan ini senantiasa berkembang dalam berbagai bentuk[11].

2.6 Cloud Computing

Cloud Computing atau komputasi awan adalah model penggunaan sumber daya komputasi yang memungkinkan pengguna untuk mengakses dan menggunakan sumber daya komputasi, seperti prosesor, penyimpanan, dan jaringan, secara *online* dan berbasis jaringan[12], [13]. Dalam model ini, sumber daya komputasi disimpan dan diproses di server-server yang terletak di lokasi yang berbeda, biasanya di pusat data yang terhubung ke internet. Pengguna dapat mengakses sumber daya ini melalui internet menggunakan perangkat lunak khusus yang disediakan oleh penyedia layanan *cloud computing*.

Komputasi awan memungkinkan pengguna untuk menggunakan sumber daya komputasi secara dinamis dan skalabel, sehingga pengguna dapat meningkatkan atau mengurangi sumber daya yang digunakan sesuai dengan kebutuhan[13], [14]. Hal ini memungkinkan pengguna untuk lebih efektif dalam penggunaan sumber daya komputasi dan menghemat biaya operasional. Komputasi awan juga memungkinkan pengguna untuk mengakses aplikasi dan data dari mana saja, kapan saja, dan menggunakan perangkat apa pun yang terhubung ke internet. Hal ini memungkinkan pengguna untuk bekerja secara lebih fleksibel dan meningkatkan produktivitas.

Beberapa kelebihan dari komputasi awan antara lain:

- Skalabilitas: Pengguna dapat meningkatkan atau mengurangi sumber daya yang digunakan sesuai dengan kebutuhan.
- Flexibilitas: Pengguna dapat mengakses aplikasi dan data dari mana saja, kapan saja, dan menggunakan perangkat apa pun yang terhubung ke internet.
- Biaya efektif: Pengguna dapat menghemat biaya operasional dengan menggunakan sumber daya komputasi yang dapat disesuaikan dengan kebutuhan.

- Keamanan: Penyedia layanan *cloud computing* biasanya memiliki sistem keamanan yang lebih baik daripada sistem keamanan yang digunakan oleh pengguna secara lokal.

2.7 PPDIOO

Gambar 2. 3 Alur PPDIOO

Metode PPDIOO *(Prepare, Plan, Design, Implement, Operate, dan Optimize)* adalah sebuah pendekatan yang digunakan dalam perancangan dan *implement*asi infrastruktur jaringan yang dikembangkan oleh *CISCO*, khususnya dalam pengembangan sistem server[15]. Metode ini terdiri dari enam tahapan yang berurutan dan saling terkait untuk memastikan bahwa sistem yang dibangun dapat berfungsi dengan baik dan dapat disesuaikan dengan kebutuhan organisasi.

- *Prepare*: Tahapan ini melibatkan persiapan sumber daya dan perencanaan skop proyek. Dalam tahapan ini, didefinisikan kebutuhan sistem, tujuan, dan anggaran yang diperlukan.
- *Plan*: Dalam tahapan ini, perencanaan rinci dilakukan. Dalam tahapan ini, didefinisikan arsitektur sistem, komponen dan peran masing-masing, serta jadwal proyek.
- *Design*: Tahapan ini melibatkan perancangan rinci sistem, termasuk arsitektur jaringan, konfigurasi perangkat keras dan lunak, serta pengamanan. Tahapan ini sangat penting untuk memastikan bahwa sistem

dapat berfungsi dengan baik dan dapat disesuaikan dengan kebutuhan organisasi.

- *Implement*: Dalam tahapan ini, sistem diterapkan sesuai dengan spesifikasi desain. Dalam tahapan ini, konfigurasi perangkat keras dan lunak dilakukan, serta pengintegrasian komponen.
- *Operate*: Dalam tahapan ini, sistem diterapkan dan tim yang bertanggung jawab atas pengawasannya mengambil alih. Dalam tahapan ini, sistem diperiksa untuk memastikan bahwa sistem berfungsi dengan baik dan diperbaiki jika terjadi masalah.
- *Optimize*: Tahapan terakhir melibatkan pengawasan dan perbaikan sistem secara terus-menerus untuk memastikan bahwa sistem dapat disesuaikan dengan kebutuhan organisasi yang terus berkembang. Dalam tahapan ini, ditemukan area untuk perbaikan, perubahan dilakukan, dan sistem tetap skalabel dan aman.

Metode PPDIOO sangat populer dalam perancangan dan implementasi infrastruktur jaringan karena memberikan kerangka kerja yang terstruktur untuk mengelola proyek-proyek kompleks, memastikan bahwa semua aspek dipertimbangkan, dan sistem dapat dirancang dan diimplementasikan untuk memenuhi kebutuhan organisasi[16].

2.8 Kubernetes

Kubernetes adalah sebuah *platform* open-source yang dirancang untuk mengelola dan mengatur aplikasi yang berbasis *container*, seperti *Docker*. Dengan menggunakan *Kubernetes*, pengembang dapat mengatur aplikasi mereka secara efektif dan efisien, serta memastikan aplikasi tersebut dapat berjalan secara stabil dan skalabel dalam lingkungan produksi[17]. *Kubernetes* memungkinkan pengembang untuk mengelola aplikasi secara otomatis, memantau performa, serta melakukan perawatan dan pemulihan aplikasi secara cepat dan efektif.

Kubernetes juga memungkinkan pengembang untuk melakukan *deployment* aplikasi secara otomatis, memantau performa aplikasi, serta melakukan perawatan

dan pemulihan aplikasi secara cepat dan efektif. Dengan demikian, *Kubernetes* memungkinkan pengembang untuk mengelola aplikasi mereka secara efektif, serta memastikan aplikasi tersebut dapat berjalan secara stabil dan skalabel dalam lingkungan produksi[18].

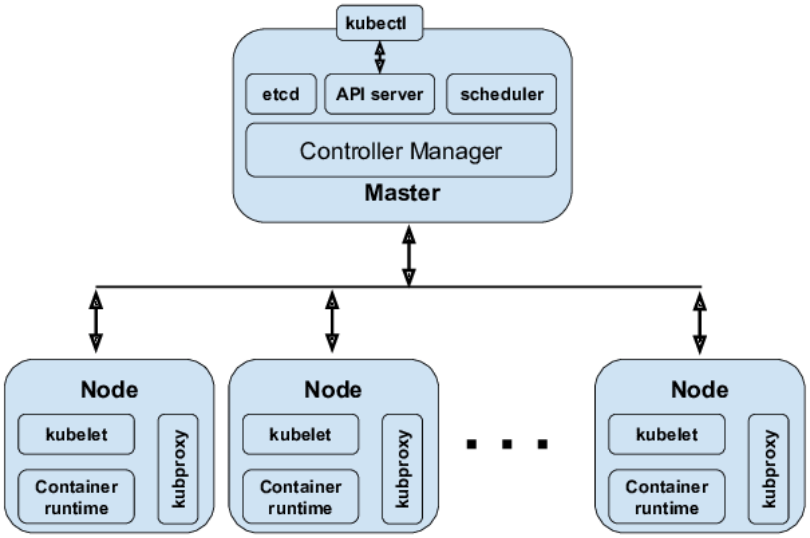
Komponen-komponen yang terdapat dalam *Kubernetes* antara lain:

- *Pod*s: *Pod*s adalah unit terkecil dalam *Kubernetes* yang berisi satu atau lebih *container* yang berjalan bersama-sama. *Pod*s dapat berisi *container* yang berbeda-beda, seperti *container* untuk aplikasi web dan *container* untuk database.
- *ReplicaSets*: ReplicaSets adalah komponen yang digunakan untuk mengatur jumlah *pod*s yang berjalan. Dengan menggunakan *ReplicaSets*, pengembang dapat mengatur jumlah *pod*s yang harus berjalan secara simultan, serta memastikan bahwa jumlah *pod*s tersebut tetap konstan.
- *Deployment*s: *Deployment*s adalah komponen yang digunakan untuk mengatur bagaimana aplikasi dideploy dan diupdate. *Deployment*s memungkinkan pengembang untuk mengatur bagaimana aplikasi dideploy, serta memastikan bahwa aplikasi tersebut tetap berjalan secara efektif setelah *dideploy*.
- *Services*: *Services* adalah komponen yang digunakan untuk mengatur bagaimana aplikasi berinteraksi dengan komponen lainnya dalam sistem. *Services* memungkinkan pengembang untuk mengatur bagaimana aplikasi dapat diakses, serta memastikan bahwa aplikasi tersebut dapat berinteraksi dengan komponen lainnya secara efektif.
- *Persistent Volumes*: *Persistent Volumes* adalah komponen yang digunakan untuk mengatur bagaimana data aplikasi disimpan dan diakses. *Persistent Volumes* memungkinkan pengembang untuk mengatur bagaimana data aplikasi disimpan, serta memastikan bahwa data tersebut tetap tersedia meskipun aplikasi *dideploy* ulang.
- *Persistent Volume Claims*: *Persistent Volume Claims* adalah komponen yang digunakan untuk mengatur bagaimana aplikasi mengakses data yang

disimpan. Persistent Volume Claims memungkinkan pengembang untuk mengatur bagaimana aplikasi dapat mengakses data yang disimpan, serta memastikan bahwa aplikasi tersebut dapat mengakses data secara efektif.

- *Namespaces*: *Namespaces* adalah komponen yang digunakan untuk mengatur bagaimana aplikasi berinteraksi dengan komponen lainnya dalam sistem. Namespaces memungkinkan pengembang untuk mengatur bagaimana aplikasi dapat diakses, serta memastikan bahwa aplikasi tersebut dapat berinteraksi dengan komponen lainnya secara efektif.
- *Labels* and *Selectors*: *Labels* and *Selectors* adalah komponen yang digunakan untuk mengatur bagaimana aplikasi dapat diidentifikasi dan diakses. Labels and Selectors memungkinkan pengembang untuk mengatur bagaimana aplikasi dapat diidentifikasi, serta memastikan bahwa aplikasi tersebut dapat diakses secara efektif.

Kubernetes terbagi menjadi *master node* dan *worker node* yang memiliki fungsi yang berbeda. *Master node* pada *Kubernetes* berfungsi sebgai *master* yang mengontrol keseluruhan unit dan *cluster*, megatur workload serta komunikasi antar sistem. Sedangkan *worker node* atau yang dikenal sebagai pekerja atau minion, merupakan mesin tempat *container* digunakan[19].



Gambar 2. 4 Arsitektur Kubernetes

Master node pada *Kubernetes* berfungsi sebagai *master* yang mengontrol keseluruhan unit dari *cluster*, mengatur *workload* serta komunikasi antar sistem[20]. Berikut adalah komponen pada *master node* :

- *Etcd* : Digunakan untuk menyimpan data *cluster*, serta memberikan notifikasi kepada semua *cluster* ketika mengalami perubahan konfigurasi. Etcd hanya dapat diakses lewat *API* server demi keamanan.
- *API* server : Komponen utama dan melayani *kubernetes API* menggunakan JSON melalui HTTP yang menyediakan *interface* internal maupun eksternal pada *kubernetes*.
- *Scheduler* : Komponen yang mengatur penjadwalan dari *node-node* dan *pod* yang tidak terjadwal. Scheduler melacak berdasar sumber daya pada setiap *node* untuk memastikan bahwa beban kerja tidak melebihi sumber daya yang tersedia. Pada dasarnya *scheduler* melakukan pengecekkan antara sumber daya yang tersedia dengan permintaan.
- *Controller manager* : *Loop* yang mendorong status *cluster* actual menuju status *cluster* yang diinginkan.

Worker node atau yang dikenal sebagai pekerja atau minion, merupakan mesin tempat *container* digunakan. Beberapa komponen pada *worker node* antara lain :

- *Kubelet* : Berfungsi untuk melaporkan kondisi *node* dan *pod* kepada *master node* jika mengalami perubahan. *Kubelet* mengambil konfigurasi dari API server untuk memastikan *pod* berjalan yang tersimpan pada *master node*.
- *Kube–proxy* : Merupakan *implement*asi dari *network proxy* dan *load balancer*, dan mendukung abstraksi layanan bersama dengan operasi jaringan lainnya. Bertugas untuk mengarahkan jalan *container* yang sesuai berdasar IP *address* dan port number dari request yang masuk.
- *Container runtime* : Tingkat terendah dari layanan mikro yang menampung aplikasi yang sedang berjalan, libraries, dan dependensinya. Salah satu contoh yang didukung oleh *Kubernetes* adalah *Docker*.

Addons ialah komponen pendukung selain komponen utama yang disediakan untuk mendukung *Kubernetes cluster*. Beberapa *Addons* yang umum digunakan ialah :

- *Kubectl* : Untuk mengirimkan perintah ke *master node Kubernetes*.
- *Kubeadm* : Untuk mempermudah proses pembuatan *Kubernetes cluster*.
- *Container Resource Monitoring* : Menyediakan runtime aplikasi yang andal dan dapat meningkatkan atau menurunkan beban kerja, dengan arti mampu mengefektifkan dan memantau performance workload.
- *Metric*-server : Berfungsi untuk mengumpulkan data pemakaian sumber daya dari tiap *node*.
- *Cluster* – level logging : Kemampuan untuk memiliki penyimpanan dan life-cycle yang terpisah dari *node*, *pod*, atau *container* dengan tujuan untuk menghindari hilangnya data akibat kegagalan *node* atau *pod*.

2.9 Horizontal Pod Autoscaler (HPA)

*Horizontal Pod Autoscaler (*HPA*)* pada *Kubernetes* adalah sebuah fitur yang memungkinkan pengguna untuk mengatur otomatis skalabilitas aplikasi berbasis kontainer di dalam *cluster Kubernetes*[20]. Dengan menggunakan HPA, pengguna dapat mengatur agar aplikasi dapat menyesuaikan diri dengan perubahan beban kerja secara dinamis, sehingga memastikan aplikasi dapat berjalan dengan efisiensi sumber daya [21]. HPA bekerja dengan cara mengawasi penggunaan sumber daya seperti CPU dan memori pada setiap *node* dalam *cluster*, serta membandingkan penggunaan tersebut dengan target penggunaan yang telah ditentukan. Jika penggunaan sumber daya melebihi target, HPA akan secara otomatis menambahkan lebih banyak *pod* (*instance* aplikasi) untuk memenuhi permintaan. Sebaliknya, jika penggunaan sumber daya kurang dari target, HPA akan mengurangi jumlah *pod* untuk menghemat sumber daya. Fitur-fitur lain yang ditawarkan oleh HPA termasuk:

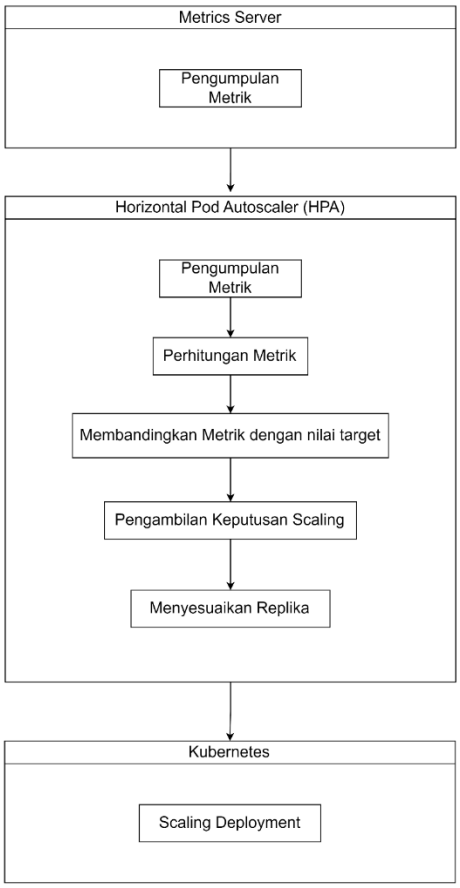
- *Real-time monitoring*: HPA secara terus-menerus memantau penggunaan sumber daya pada setiap *node* dan membandingkannya dengan target penggunaan yang telah ditentukan.

- *Dynamic scaling*: HPA dapat secara otomatis menambahkan atau mengurangi jumlah *pod* untuk memenuhi perubahan beban kerja.

Dengan menggunakan HPA, pengguna dapat meningkatkan efisiensi sumber daya, dan meningkatkan kualitas layanan aplikasi. Namun, pengguna juga perlu memastikan bahwa target penggunaan yang telah ditentukan sesuai dengan kebutuhan aplikasi dan memastikan bahwa HPA tidak mengalami masalah dalam mengatur skalabilitas aplikasi[22].

2.9.1 Cara Kerja HPA

Berikut ini penjelasan lebih fokus tentang cara kerja HPA (Horizontal Pod Autoscaler) di Kubernetes:



Gambar 2. 5 Cara Kerja HPA

1. Pengumpulan Metrik: HPA bergantung pada Metrics Server untuk mengumpulkan metrik sumber daya seperti CPU dan memori dari setiap Pod yang dikelolanya. Metrics Server secara berkala mengumpulkan dan menyimpan metrik ini.
2. Pemantauan Metrik: HPA secara berkala (misalnya setiap 30 detik) memeriksa metrik sumber daya dari objek workload yang dikelolanya dengan mengakses data metrik dari Metrics Server.
3. Perhitungan Metrik: HPA menghitung rata-rata penggunaan sumber daya (misalnya CPU) dari semua Pod yang dikelolanya selama periode pemantauan tertentu (biasanya 5 menit terakhir).
4. Membandingkan dengan Nilai Target: HPA membandingkan rata-rata penggunaan sumber daya dengan nilai target yang telah ditentukan dalam spesifikasi HPA.
5. Pengambilan Keputusan Scaling: Jika rata-rata penggunaan sumber daya melebihi nilai target, HPA akan memutuskan untuk menambah jumlah replika Pod. Sebaliknya, jika rata-rata penggunaan sumber daya di bawah nilai target, HPA akan mengurangi jumlah replika Pod. Keputusan ini diambil berdasarkan algoritma yang dapat dikonfigurasi, seperti faktor penskalaan dan toleransi.
6. Menyesuaikan Replika: HPA memodifikasi jumlah replika yang dikonfigurasi pada objek workload (misalnya Deployment) untuk mencerminkan jumlah Pod baru yang diinginkan.
7. Scaling Deployment: Kubernetes kemudian secara otomatis membuat atau menghapus Pod sesuai dengan jumlah replika baru yang ditetapkan oleh HPA.
8. Stabilisasi: HPA terus memantau metrik sumber daya dan melakukan penyekalaan sesuai kebutuhan untuk mempertahankan penggunaan sumber daya di sekitar nilai target yang ditentukan.

Proses ini berulang secara terus-menerus, sehingga HPA dapat secara dinamis menyesuaikan jumlah replika Pod berdasarkan permintaan sumber daya

yang berubah-ubah. Hal ini memastikan bahwa aplikasi selalu memiliki sumber daya yang cukup untuk memenuhi permintaan, sekaligus menghindari pemborosan sumber daya ketika permintaan rendah.