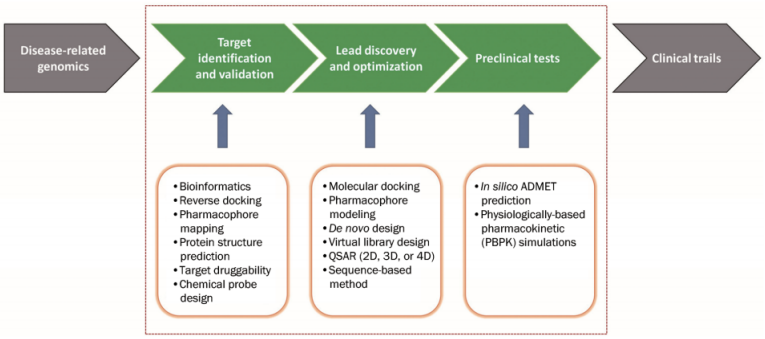
BAB 2
LANDASAN TEORI

Bab ini akan menjelaskan sekilas tentang *computational drug discovery*, apa itu *cloud computing* , dan Docker sebagai platform virtualisasi komputer.

2.1 *Computational Drug discovery*

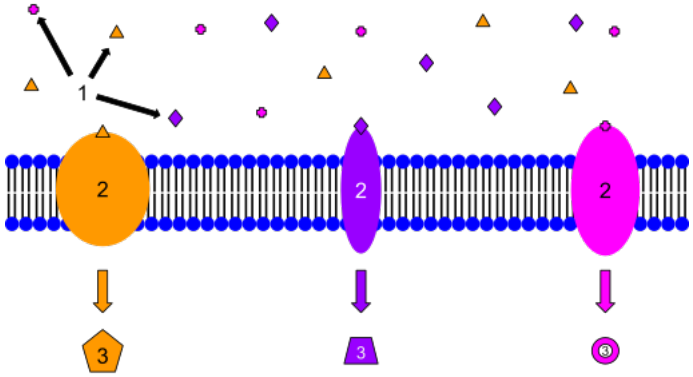
Drug discovery adalah suatu proses menemukan kandidat obat yang berpotensial. Proses ini melibatkan berbagai displin ilmu seperti biologi, kimia, maupun farmakologi. Dalam *drug discovery* dahulu kala, terkadang obat tersebut ditemukan secara tidak sengaja(*serendipity*) atau berdasarkan penyembuhan tradisional. Seiring dengan berkembangnya pengetahuan dan teknologi, khususnya dalam bidang kimia, ekstraksi senyawa kimia dari makhluk hidup merupakan hal yang wajar. Ekstrak tersebut akan disimpan dan akan disimpan sebagai dalam suatu *library* yang nantinya dapat digunakan kembali. Ekstrak tersebut dapat di-*screening*, apakah terdapat senyawa yang berpotensi sebagai penyembuh dari suatu *sympton* penyakit. Hal ini merupakan dasar farmakologi (*classical pharmacology*).

Waktu yang dibutuhkan dalam proses *drug discovery* cukup lama. Tidak hanya itu, proses ini juga sangat beresiko dan membutuhkan dana yang besar [1].Walaupun demikian, investasi dana dalam *drug discovery* juga berkembang pesat. Namun, investasi tersebut tidak dibarengi dengan hasil yang proporsional dengan investasi tersebut. Hal ini disebabkan oleh rendahnya efektivitas dan tingginya kemungkinan kegagalan dalam *drug discovery*. Beberapa pendekatan telah dilakukan dalam meningkatkan efektivitas, salah satu cara tersebut adalah CADD (*Computer Aided Drug Design*).Dengan begitu, penekanan biaya dan siklus waktu penemuan obat juga semakin cepat. Dalam CADD, komputer digunakan sebagai sarana komputasi dan penyimpanan data dalam *drug discovery*. Tidak ketinggalan, CADD menggunakan aplikasi dalam mendesain ikatan kimiawi, memodelkan struktur kimia yang mengandung kandidat calon obat yang berpotensial dan pembuatan *library* yang dapat digunakan untuk pembelajaran selanjutnya.



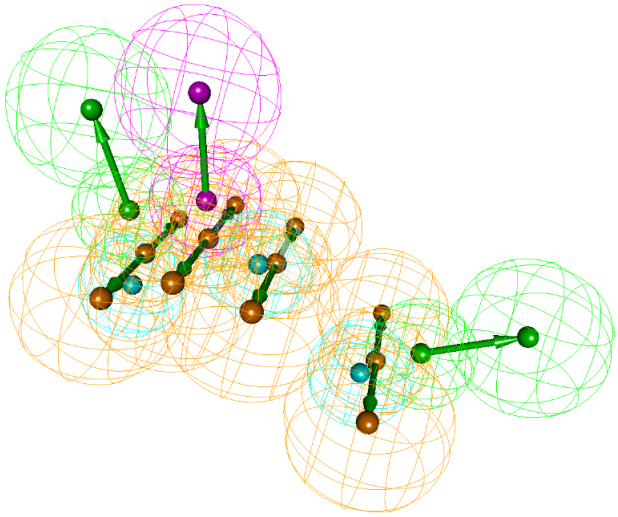
Gambar 2.1: Pemanfaatan komputer sebagai sarana komputasi dalam *drug discovery*

Dalam *computational drug discovery* (penemuan obat dengan memanfaatkan komputer), pendekatan yang dapat dilakukan dibagi menjadi *structure-based drug design*(SBDD), *ligand-based drug design*(LBDD) dan pendekatan berdasarkan sekuens. terdiri dari proses *docking* kandidat *ligand* ke target protein kemudian dilakukan penilaian dengan menggunakan *scoring function* untuk dapat menghitung kemungkinan *ligand*tersebut terikat pada target protein dengan daya tarik yang tinggi.



Gambar 2.2: Ilustrasi *ligand*(nomor 1)yang akan terikat dengan *receptor*(nomor 2)

Dalam LBDD, diberikan sekumpulan *ligand* dan *receptor*, dimana model *receptor*dapat dibuat dengan mengumpulkan informasi dari *ligand*. Model ini dikenal dengan *pharmacophore*. Kemudian, kandidat *ligand* akan dibandingkan dengan *pharmacophore*, apakah kompatibel dengan *pharmacophore* dan dapat terikat.

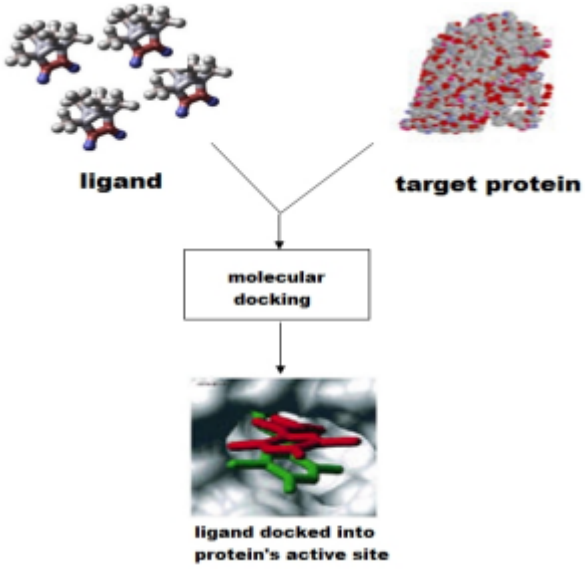


Gambar 2.3: Contoh dari *pharmacophore*

2.2 *Molecular Docking*

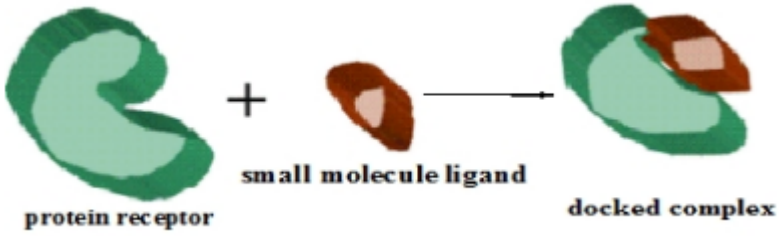
Virtual screening berdasarkan *molecular docking* merupakan metode yang sering digunakan dalam SBDD. Penulis juga akan menggunakan metode tersebut yang terdapat dalam Program Autodock dan Autodock Vina. Pemodelan struktur kimia menggunakan *molecular modeling* untuk mempelajari fenomena dari struktur kimia tersebut. Kemudahan dalam *molecular modeling* sekarang ini dibantu dengan aplikasi komputer yang tersedia, namun masih terdapat kesulitan yaitu bagaiman cara mendapatkan model struktur ikatan kimia yang benar dan interpretasi model yang tepat.Secara umum, *molecular modeling* dapat dikatakan sebagai pemanfaatan komputer dalam mengkonstruksi molekul dan melakukan berbagai macam perhitungan untuk mempelajari karakteristik dan sifat struktur ikatan kimia. Istilah *molecular modeling* terkadang disamakan dengan istilah *computational chemistry*.

Proses *drug discovery* yang semula berupa *trial and error* berubah menjadi proses yang dibantu dengan perhitungan komputer. Perhitungan tersebut digunakan dalam menentukan struktur ikatan kimia baru berdasarkan struktur protein yang sudah diketahui. Pendekatan ini terbagi menjadi dua : *de novo design* dan *docking*. *Docking* merupakan suatu proses untuk menebak struktur *complex* dari struktur *ligand* dan protein. Dalam *molecular modeling*, *docking* dapat dipandang sebagai suatu metode untuk memprediksi orientasi suatu molekul ketika diikat dengan molekul lainnya untuk membentuk *complex* yang stabil.



Gambar 2.4: *Proses molecular docking*

Molecular docking adalah suatu proses komputasi dalam pencarian *ligand* yang paling cocok baik secara geometri maupun energi ketika diikat dengan suatu *receptor*(protein) yang telah diketahui.Aspek utama dalam *molecular docking* adalah perhitungan energi interaksi dan konformasi dengan menggunakan metode dari kuantum mekanik hingga fungsi empiris energi. Permasalahan *molecular docking* sekilas dapat dilihat sebagai suatu permasalahan *lock and key*, dimana *receptor* protein dapat dipandang sebagai *lock* dan *ligand* sebagai *key*. Namun dalam praktiknya, *molecular docking* akan mencari *ligand* yang dapat menyesuaikan dengan *receptor*.karena adaptasi tersebut, *molecular docking* dapat dipandang sebagai permasalahan "*best-fit*" *ligand*. Diperlukan *scoring function* untuk dapat membatasi *best-fit* dari proses *molecular docking*. Fungsi tersebut biasanya berdasarkan *force field* yang digunakan dalam mensimulasikan protein. Beberapa *scoring function* juga menambahkan aspek perhitungan lainnya seperti entropi.



Gambar 2.5: *best-fit docking*

Dalam percobaan ini penulis memanfaatkan aplikasi *molecular docking* Autodock dan Autodock Vina.

2.3 *Autodock dan Autodock Vina*

Aplikasi ini merupakan aplikasi yang dikembangkan oleh *The Scripps Research Institute*, lembaga riset nonprofit yang terletak di California, Amerika Serikat. Terdapat 2 jenis aplikasi, Autodock dan Autodock Vina. Masing masing merupakan aplikasi yang saling berbeda dalam segi *scoring function* yang digunakan dan optimisasi komputasi secara paralel oleh Autodock Vina.

Paket aplikasi Autodock terdiri dari 2 program, Autodock4 dan Autogrid4. AutoGrid akan menghasilkan *pre-calculated map* dari suatu *ligand*.Selain itu juga akan menghasilkan *map* tambahan, ”d” untuk *desolvation* dan ”e” untuk *electrostatic*.Pengoperasian Autodock membutuhkan beberapa file, yaitu : *.dpf untuk pengaturan *docking* parameter , hasil *map* berupa *.gpf dari AutoGrid , dan *.pdbqt untuk *receptor* dan *ligand* yang akan diujicoba. Keluaran yang dihasilkan oleh Autodock berupa *.dlg yang berisi hasil perhitungan konformasi posisi dan energi pada setiap *ligand* yang diujicoba. Paket aplikasi Autodock Vina hanya datang dengan sebuah program saja dikarenakan proses *pre-processing docking* dibuat transparan sehingga pengguna tidak perlu tahu dan memahami bagaimana aplikasi tersebut bekerja. Autodock Vina membutuhkan 2 file, *.pdbqt dari *receptor* dan *.pdbqt dari tiap *ligand* yang akan diujicobakan. Keluaran yang dihasilkan juga beruapa *.pdbqt yang beisikan konformitas posisi dan energi.

Gambar 2.6: Logo Autodock

2.4 *Cloud Computing*

Seperti yang telah dijelaskan sebelumnya, pemanfaatan komputer dalam riset mengalami kendala di negara berkembang. Biaya yang harus dikeluarkan dan sumber daya listrik yang dibutuhkan untuk pengadaan *supercomputer* terbilang besar. Selain itu, dibutuhkan pengetahuan lebih terhadap perangkat keras yang akan digunakan tersebut. Hal ini bertujuan supaya pemeliharaan dan penggunaan *supercomputer* lebih baik. Dengan perkembangan teknologi jaringan dan internet yang pesat, muncul teknologi baru yang dikenal dengan *cloud computing*. Teknologi menjawab permasalahan pengadaan perangkat komputasi dan tidak peneliti tidak perlu memahami secara dalam dibalik layanan yang disediakan oleh teknologi tersebut.

Teknologi *cloud computing* sudah berkembang sejak tahun 1990-an. Pada awalnya teknologi ini dikembangkan sebagai solusi dari permasalahan yang dihadapi oleh perusahaan dengan bertambahnya kapasitas data yang perlu disimpan dan pengeluaran yang cukup besar untuk pengadaan perangkat keras beserta konsumsi daya listrik yang dibutuhkan. Definisi dari *cloud computing* dapat dikatakan sebagai virtualisasi dari perangkat komputasi, dimana pengguna mengakses perangkat tersebut dengan menggunakan jaringan internet. Selain itu, teknologi ini memberikan kemudahan dalam penggunaan secara *multiuser*, dimana dapat diakses secara bersamaan.

Terdapat 3 karakteristik dari teknologi *cloud computing*, yaitu virtualisasi, terdistribusi, dan kemudahan dalam pengembangan selanjutnya (*dynamically extendibility*). Virtualisasi merupakan faktor yang utama. Virtualisasi dapat membagi suatu perangkat komputasi secara fisik menjadi beberapa ”virtual” perangkat komputasi. Dengan begitu, kinerja dari perangkat akan dioptimisasi. Alokasi sumber daya perangkat keras tidak ada yang diam secara percuma. Disamping itu, dengan adanya virtualisasi dapat menekan biaya pengeluaran pengadaan perangkat komputasi secara fisik. Yang dimaksud dengan terdistribusi adalah perangkat keras (*physical node*) yang digunakan dalam *cloud computing* digunakan secara terdistribusi. Dalam tahap pengembangan teknologi ini memberikan kemudahan. Kemudahan tersebut merupakan efek dari virtualisasi. Ketika pengguna ingin mengikuti perkembangan yang ada, tidak perlu merubah / membuat kembali dari awal. Pengguna dapat merubah dari level virtual (virtualisasi).

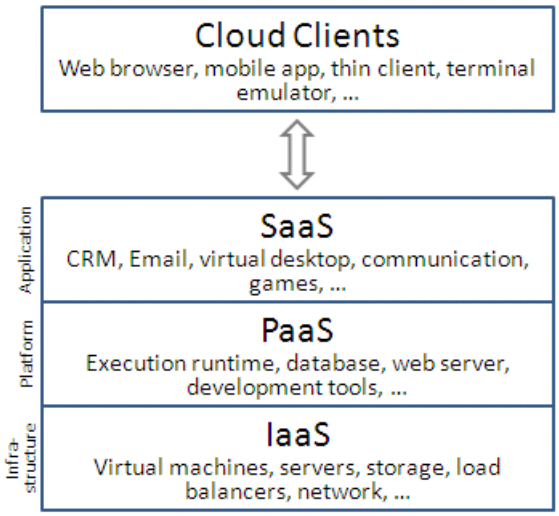
Terdapat 3 jasa yang disediakan dalam *cloud computing* :

- **Software as a Service (SaaS)**
  Layanan jasa ini menyediakan aplikasi dan penyimpanan data (*database*) yang langsung dapat digunakan. Pengguna dapat mengakses aplikasi secara bersamaan dengan pengguna lainnya. SaaS dikenal juga sebagai *on-demand software* dan biaya yang dikenakan oleh penyedia jasa ini merupakan *pay-per-use*. Umumnya, pengguna akan dikenakan biaya bulanan atau tahunan. Pengguna tidak perlu susah - susah untk menginstall aplikasi pada *cloud* dikarenakan penyedia jasa tersebut akan melakukan hal tersebut. Saas memberikan keuntungan bagi perusahaan dengan cara mengurangi biaya operasional IT (Information Technology). Hal ini dikarenakan biaya untuk pemeliharaan dan pengadaan perangkat komputasi ditanggungkan kepada pihak penyedia jasa (*outsourcing*). Contoh dari Saas adalah layanan *e-mail*, media sosial.

- **Platform as a Service (PaaS)**

Berbeda dengan Saas, layanan ini menyediakan perangkat komputasi yang dapat digunakan untuk menjalankan aplikasi dari pengguna. Layanan ini dapat menyediakan sistem operasi atau *framework* yang dibutuhkan dalam menjalankan apliksi tersebut. Contoh dari layanan jasa ini adalah : Windows Azure, Amazon Web Service, dan Google App Engine.

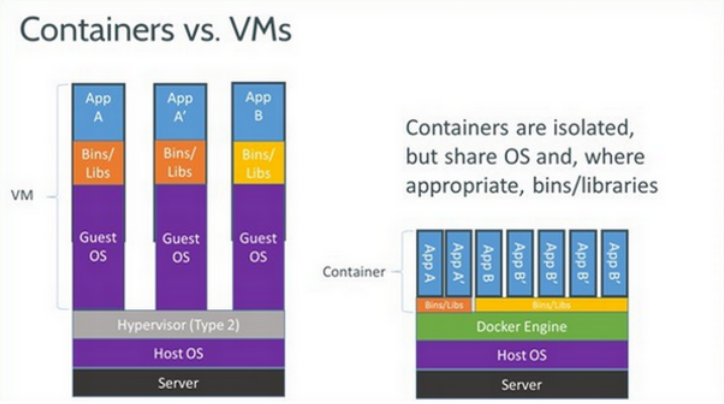
- **Infrastructure as a Service (IaaS)**
  layanan ini menyediakan infrastruktur IT kepada pengguna sesuai dengan permintaan yang dapat diakses dengan jaringan internet. Infrstruktur tersebut meliputi perangkat keras seperti *harddisk, memory, firewall*, tipe server, alamat IP, *virtual local area networks*(VLANs), maupun aplikasi yang harus terinstall dalam server



Gambar 2.7: Layanan dalam *cloud computing*

2.5 Docker

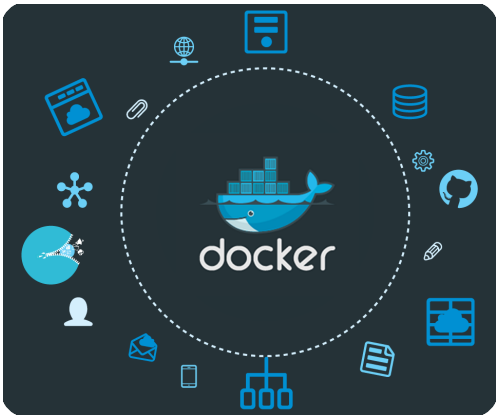
Virtualisasi merupakan faktor utama pada teknologi *cloud computing* dalam mengoptimisasikan kinerja server sehingga tidak ada sumber daya yang tidak terpakai. Virtualisasi yang dimaksud adalah virtualisasi perangkat keras (*hardware*). Virtualisasi tersebut memungkinkan ”virtual” server yang dapat dibuat sesuai dengan kebutuhan yang diperlukan dan berjalan pada server fisik. Spesifikasi untuk masing masing virtual server dapat disesuaikan dengan kebutuhan aplikasi yang akan dijalankan diatasnnya dan tidak melebihi spesifikasi server fisik. Masing - masing virtual server akan saling terisolasi satu dengan yang lainnya. Virtual



Gambar 2.8: Perbandingan virtualisasi pada VM dan container pada Docker

Permasalahan virtualisasi tersebut muncul ketika hendak mengoptimisasikan sumber daya yang ada. Ketika suatu ”virtual” server dibuat untuk menjalankan suatu aplikasi yang spesifik, maka keseluruhan sistem perlu dibuat (Sistem operasi, alokasi sumber daya memori, *processor*) termasuk *library* yang dibutuhkan dalam menjalankan sistem / aplikasi tersebut. ”virtual” server tersebut akan diatur oleh suatu program *hypervisor* yang menjembatani komunikasi dengan server fisik. Dengan begitu, kapasitas memori dan alokasi sumber daya pada server tidak optimal dalam menjalankan ”virtual” server secara bersamaan.

Docker, *platform*virtualisasi yang dikembangkan oleh Perusahaan Docker menggunakan teknik berbeda dengan virtualisasi *hardware*. Docker menggunakan *Docker engine* sebagai ganti dari *hypervisor*.*Docker engine* dikembangkan berdasarkan *Linux containers* (LXC), dimana level virtualisasi terletak pada sistem operasi. Dengan begitu suatu server linux dapat menjalankan beberapa sistem linux yang saling terisolasi satu sama lain. Kernel pada ”virtual” server, dalam hal ini disebut *container* yang dibentuk dengan LXC akan menggunakan kernel yang sama dengan server fisik. Oleh karena itu, *container* yang dibentuk akan lebih kecil dan padat tanpa perlu menginstall sistem secara keseluruhan dari awal. *container* tidak perlu mengalokasikan sumber daya secara tetap dikarenakan *Docker engine* akan mengalokasikan sumber daya berdasarkan aplikasi yang berjalan pada *container*.



Gambar 2.9: Logo Docker

Walaupun begitu, untuk masing masing virtualisasi, baik itu yang menggunakan *hypervisor* maupun *Docker engine* pada Docker memiliki kelebihan dan kekurangan masing - masing. *Container* yang dibentuk pada Docker harus menggunakan sistem operasi yang sama dengan server fisik. Untuk saat ini, Docker baru mendukung sistem operasi Linux sebagai *container* dan berjalan pada sistem operasi Linux. Walaupun dapat berjalan pada sistem operasi lainnya, namun penggunaan Docker masih dijalankan dalam ”virtual” komputer yang menggunakan sistem operasi Linux pada *Virtual Machine*. Dengan alokasi sumber daya yang tidak tetap (sesuai dengan proses yang berjalan), maka *container* yang terbentuk akan lebih banyak dari ”virtual” server yag dihasilkan dengan menggunakan ”hypervisor” pada umumnya terikat dengan batasan sumber daya. Namun ”virtual” server tersebut lebih beragam dan tidak terikat dengan sistem operasi yang berjalan pada server fisik.