BAB II
TINJAUAN PUSTAKA

2.1. PT Ninety Nine Dot Co

PT Ninety Nine Dot Co (Urbanindo sebelum akuisisi) adalah situs jual beli properti online di Indonesia. Didirikan pada November 2011, PT Ninety Nine Dot Co memberikan layanan bagi penggunanya untuk membeli, menyewa, atau menjual properti berdasarkan kata kunci, lokasi di peta dan berdasarkan agen penjual properti. Sejak akhir 2018 urbanindo telah diakuisisi oleh *startup* properti asal singapura yaitu Ninety Nine Dot Co. Kini urbanindo berubah nama dan beroperasi sebagai PT Ninety Nine Dot Co Indonesia. Website portal 99.co dapat diakses melalui tautan https://www.99.co/id.

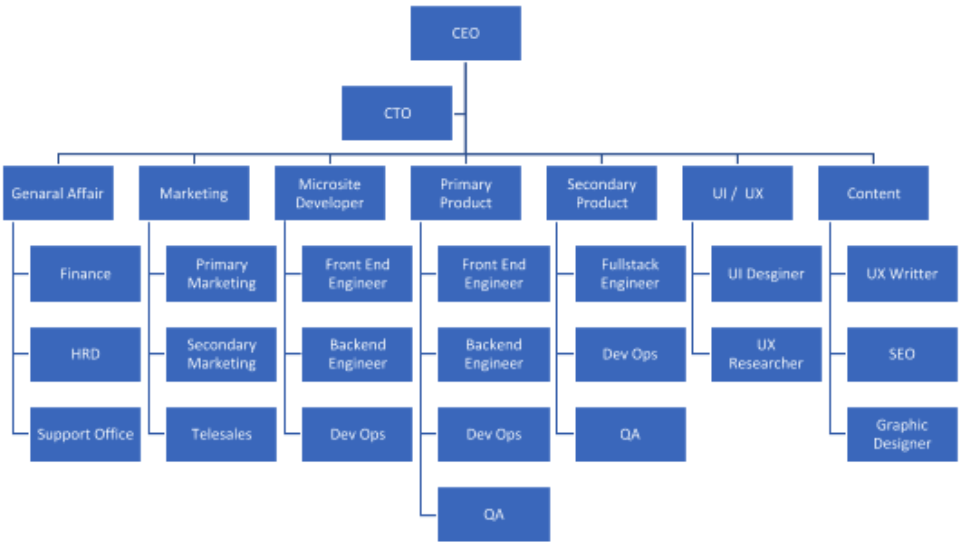
Gambar 2.1 Logo PT Ninety Nine Dot Co

Pada saat ini PT Ninety Nine Dot Co sedang mengembangkan produk – produk seperti *Website Portal 99.co/id*, *Agent Apps*, *Shortlist*, *Cross Post* dan lain sebagainya. Aplikasi yang dkembangkan oleh PT Ninety Nine Dot Co tersebut berguna untuk membantu pengguna maupun agent properti untuk menjual properti dan memberikan opsi kepada pengguna untuk membeli properti.

Adapun visi PT ninety nine dotco adalah *it is a place where you can return to, keep your loved ones close, grow a family and build careers* yang berarti PT Ninety Nine Dot Co mempunyai visi menjadi salah satu perusahaan yang memberikan jaminan bahwa setiap warga dapat mempunyai hunian sesuai dengan kemampuan agar dapat berkumpul dengan sanak keluarga dan menjadi salah satu

perusahaan yang mengutamakan pertumbuhan karir didalamnya. Adapun misi PT Ninety Nine Dotco sebagai berikut *By providing the largest set of listings in the market, designing easy beautiful tools and synthesizing data to help property seekers search for, learn about, sieve through and compare thousands of potential options* yang berarti PT Ninety Nine Dot Co mempunyai misi membantu para pengguna untuk mencari properti idaman sesuai dengan potensial diri pribadi.

2.1.1. Struktur Organisasi PT Ninety Nine Dot Co



Gambar 2.2 Bagan Perusahaan

Struktur organisasi di PT Ninety Nine Dot Co terdiri dari beberapa bagian yang mempunyai masing masing fungsi tersendri. Pada perusahaan ini mempunyai dua pimpinan utama yaitu CEO (*Chief Excecutive Officer*) dan CTO (*Chief Technology Officer*) yang menjadi pilar utama perusahaan ini berjalan. Didalam perusahaan ini terdapat tujuh bagian, berikut adalah penjelasan tujuh bagian tersebut.

a. General Affair

General affair adalah bagian yang mengatur administrasi perusahaan termasuk penggajian, absensi dan peralatan yang dibutuhkan oleh karyawan. Pada bagian ini terdiri dari *Finance, Human Resource Development* dan *Support Office*.

b. Marketing

Marketing adalah bagian yang mengatur akan transaksi pembelian properti baik bersifat *primary* (properti baru) maupun *secondary* (property bekas yang akan dijual). Pada bagian ini terdiri dari *Primary Marketing, Secondary Marketing* dan *Telesales*.

c. Microsite Developer

Microsite Developer adalah bagian yang membuat sebuah homepage / landing page sebuah perusahaan properti baru (*primary*) yang nantinya akan ditampilkan di aplikasi portal. Pada bagian ini terdiri dari *Frontend Engineer, Backend* dan *DevOps*.

d. Primary Product

Primary Product adalah bagian yang membuat sebuah halaman di aplikasi portal untuk menampilkan proyek properti baru (*primary*) yang nantinya akan dijual oleh agent yang tersedia pada proyek properti tersebut. Pada bagian ini terdiri dari *Frontend Engineer, Backend, Quality Assurance* dan *DevOps*.

e. Secondary Product

Secondary Product adalah bagian yang membuat sebuah halaman di aplikasi portal untuk menampilkan proyek properti bekas (*secondary*) yang nantinya akan dijual oleh penjual properti tersebut. Pada bagian ini terdiri dari *Frontend Engineer, Backend, Quality Assurance* dan *DevOps*.

f. UI / UX

User Interface / *User Experience* adalah bagian yang membuat, merancang dan menganalisa tampilan aplikasi yang nanti nya akan dipakai oleh pengguna / penjual properti. Pada bagian ini terdiri dari *UI Designer* dan *Ux Researcher*.

g. Content

Content adalah bagian yang membuat sebuah artikel maupun optimasi URL yang ada pada perusahaan ini, yang nantinya akan membantu pada SEO perusahaan.

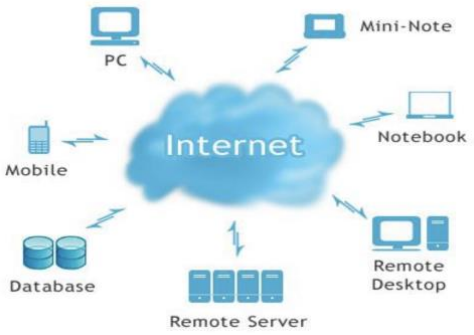
2.2. Cloud Computing

Komputasi Awan (*Cloud Computing*) adalah gabungan antara pemanfaatan teknologi komputer dengan pengembangan berbasis internet [1]. Awan (*cloud*) adalah gambaran lain dari internet, sebagaimana awan yang sering dijadikan simbol pada diagram jaringan komputer. Selain seperti awan dalam diagram jaringan komputer, awan (*cloud*) dalam *cloud computing* juga merupakan abstraksi dari infrastruktur kompleks yang disembunyikan.

Teknologi komputasi komputer dengan memanfaatkan internet sebagai terminal utamanya guna mengelola piranti lunak hingga infrastruktur sebagai suatu bentuk layanan [2]. *Cloud computing* menerapkan suatu metode komputasi, yaitu kapabilitas yang terkait teknologi informasi disajikan sebagai suatu layanan / *service* sehingga pengguna dapat mengaksesnya lewat internet, tanpa mengetahui apa yang ada di dalamnya, ahli dengannya, atau memiliki kendali terhadap infrastruktur teknologi.

2.2.1. Pengertian Cloud Computing

Cloud computing adalah suatu paradigma, yaitu sebuah informasi yang secara permanen tersimpan di *server* pada internet dan tersimpan secara sementara di komputer pengguna (*client*), termasuk didalamnya adalah komputer personal, komputer tablet, notebook, sensor-sensor, monitor dan lain sebagainya. [menurut sebuah makalah pada tahun 2008, yang dipublikasikan IEEE Internet Computing].



Gambar 2.3 Ilustrasi Cloud Computing

Ada beberapa pemahaman tentang Cloud Computing yaitu :

h. Dalam perspektif teknologi komunikasi sendiri, cloud computing atau komputasi cloud dapat diartikan sebagai suatu teknologi yang memanfaatkan internet sebagai resource untuk komputasi yang dapat di–request oleh pengguna dan merupakan sebuah layanan dengan pusat server bersifat virtual atau berada dalam cloud (internet) itu sendiri.
i. *Cloud computing* adalah evolusi selanjutnya dari internet "Awan" pada *cloud computing* merupakan penyedia (hal-hal yang berkaitan) dari tenaga komputasi hingga infrastruktur komputasi, aplikasi-aplikasi, proses bisnis hingga kolaborasi yang muncul sebagai layanan yang dapat diakses pada saaat dibutuhkan kapanpun dan dimanapun
j. Sebuah gaya komputasi yang besar, elastis, dan terukur yang berhubungan dengan kemampuan IT sebagai penyedia pelayanan (*as a service*) kepada pelanggan eksternal menggunakan teknologi internet.

2.2.2. *Software as a Service* (SaaS)

Model layanan *Software as a Service* (SaaS) adalah model layanan yang paling banyak dikembangkan untuk sistem cloud computing. Model layanan ini berupa aplikasi atau software berbasis web, yang diberikan kepada berbagai pengguna oleh vendor atau pemilik sistem tersebut. Pengguna pun tidak perlu

memiliki aplikasi tersebut untuk menggunakannya, melainkan pengguna memerlukan koneksi internet untuk dapat mengakses ataupun menggunakannya. Contoh perusahaan yang mengembangkan layanan ini adalah Google dengan contoh aplikasinya yaitu Google Docs, yang berfungsi sebagai *word processor*, *spreadsheet*, *presentation creator*.

Gambar 2. 4 Arsitektur Model Layanan SaaS

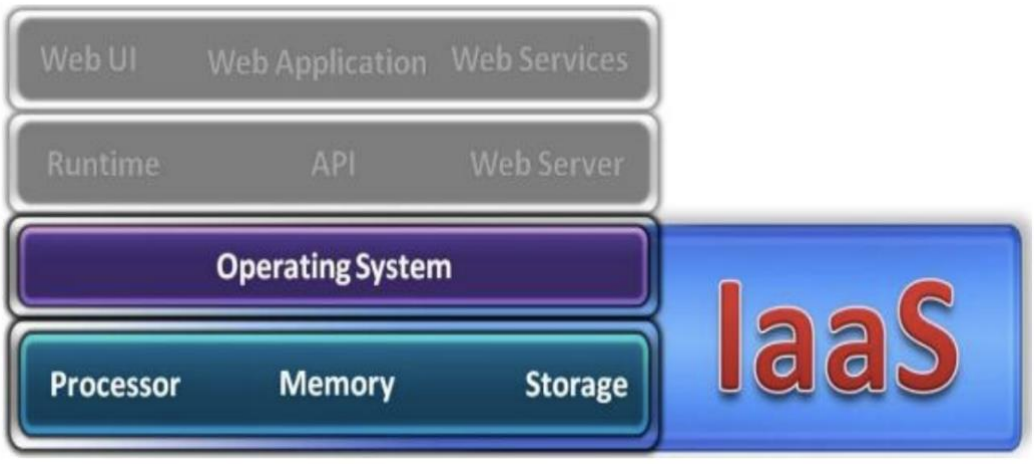
2.2.3. *Platform as a Service* (PaaS)

Model layanan *platform as a service* merupakan layanan yang menyediakan platform seperti bahasa pemrograman, *tools*, ataupun sistem operasi yang nantinya mampu mengembangkan aplikasi berbasis konsumen. Keuntungan dari PaaS adalah bahwa perusahaan yang menggunakan layanan ini tidak perlu khawatir untuk instalasi, perawatan, dan keamanan pada server karena provider PaaS akan menangani itu semua. Contoh implementasi dari PaaS adalah Microsoft Windows Azure dan Google App Engine.

Gambar 2.5 Arsitektur Model Layanan PaaS

2.2.4. *Infrastructure as a Service* (IaaS)

Model layanan disediakan dari IaaS adalah sumber daya pemroses, storage, kapasitas jaringan, dan sumber daya komputasi lainnya. Dimana konsumen dapat mengembangkan dan menjalankan aplikasi khusus. IaaS memungkinkan pengguna untuk menjalankan beberapa tugas komputer pada mesin virtual yang tak terbatas. Artinya pengguna diberikan kebebasan untuk merakit PC virtual sesuai dengan keinginan istilah dari konsep ini biasa disebut virtualization, salah satu implementasi dari model layanan ini adalah Amazon EC2.



Gambar 2.6 Arsitektur Model Layanan IaaS

2.3. AWS

Amazon Web Services (AWS) adalah penyedia layanan cloud yang aman, AWS menawarkan tenaga komputasi, ruang penyimpanan database, "content delivery network" dan fungsionalitas lainnya yang membantu banyak bisnis untuk berkembang dan menjalankan aplikasi dengan baik[3]. AWS memberikan banyak pilihan produk yang sangat memudahkan dalam membangun bisnis. Infrastructure as a service, adalah istilah yang tepat untuk AWS. Istilah lainnya ada juga Software as a service dan Platform as a service.

Beberapa produk AWS telah didesain untuk memudahkan pengguna, contohnya Elastic Compute Cloud atau EC2. Pada dasarnya EC2 adalah sebuah server virtual. Kemudahan yang ditawarkan oleh EC2 yaitu jika kita bandingkan dengan server tradisional. Pada server tradisional, kita harus mengetahui secara detail spesifikasi server yang kita inginkan seperti apa, support, spare part, lalu penempatan di data center, dan proses tersebut bukanlah proses yang mudah. Jika kita menyiapkan terlalu banyak server, sedangkan kebutuhan komputasi tidak banyak, kita akan membuang banyak biaya. Sedangkan jika kita menyiapkan sedikit server sedangkan kebutuhan komputasi sangat banyak, hal ini akan menggangu jalannya bisnis.

Selain menawarkan kemudahan, salah satu keutamaan dari AWS adalah fitur "pay as you go", yaitu bayar ketika dipergunakan. Seperti AWS EC2, pricing model yang digunakan adalah per jam untuk jenis deployment on-demand. Lalu untuk Kinesis Streams adalah shard hour, PUT Payload Unit, dan extended data retention. Dan dengan biaya yang cukup murah, kita memiliki layanan yang powerful, contohnya kinesis streams, tanpa perlu me-manage aplikasi streaming seperti apache kafka, kita bisa menyelesaikan masalah yang sama dengan waktu yang lebih singkat. Efeknya, kita bisa men-deliver fitur lebih cepat, kita bisa fokus pada pengguna, kita bisa fokus pada bisnis, dan kita tidak perlu memikirkan "scaling".

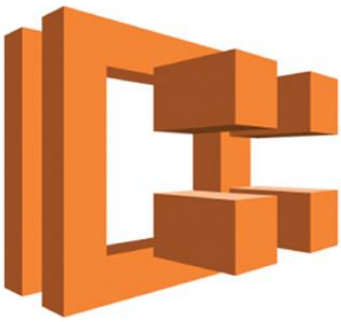
Gambar 2.7 Logo Aws

2.3.1. AWS ECS

Amazon Elastic Container Service (Amazon ECS) adalah layanan orkestrasi kontainer yang terkelola sepenuhnya[4]. Pelanggan seperti Duolingo, Samsung, GE, dan Cookpad menggunakan ECS untuk menjalankan aplikasi paling sensitif dan misi kritis karena keamanan, keandalan, dan skalabilitasnya. ECS adalah pilihan luar biasa untuk menjalankan kontainer untuk beberapa alasan. Pertama, Anda dapat memilih untuk menjalankan klaster ECS dengan menggunakan AWS Fargate, yang merupakan komputasi tanpa server untuk kontainer. Fargate menghilangkan perlunya menyediakan dan mengelola server, memungkinkan Anda menentukan dan membayar sumber daya per aplikasi, dan meningkatkan keamanan melalui isolasi aplikasi sesuai desain. Kedua, ECS digunakan secara luas di dalam Amazon untuk mendukung layanan seperti Amazon SageMaker, AWS Batch, Amazon Lex, and mesin rekomendasi Amazon.com, yang memastikan ECS diuji secara luas untuk keamanan, keandalan, dan ketersediaan.

Selain itu, karena ECS sudah menjadi pilar fondasi untuk layanan Amazon utama, maka ECS dapat terintegrasi secara natif dengan layanan lainnya seperti Amazon Route 53, Secrets Manager, AWS Identity and Access Management (IAM), dan Amazon CloudWatch yang memberi Anda pengalaman umum untuk menerapkan dan menskalakan kontainer Anda. ECS juga dapat dengan cepat terintegrasi dengan layanan AWS lainnya untuk membawa kapabilitas baru ke ECS. Misalnya, ECS memungkinkan fleksibilitas pada aplikasi Anda untuk menggunakan campuran Amazon EC2 dan AWS Fargate dengan opsi harga Spot

dan Sesuai Permintaan. ECS juga terintegrasi dengan AWS App Mesh, yang merupakan mesh layanan, untuk memberikan fitur yang kaya observabilitas, kontrol lalu lintas, dan keamanan ke aplikasi Anda. ECS telah tumbuh dengan cepat sejak diluncurkan dan saat ini meluncurkan 5x kontainer lagi setiap jam dibandingkan instans peluncuran EC2.



Gambar 2.8 Logo AWS ECS

2.3.2. AWS Fargate

AWS Fargate adalah mesin komputasi tanpa server untuk container yang berfungsi dengan Amazon Elastic Container Service (ECS) dan Amazon Elastic Kubernetes Service (EKS) [5]. Fargate mempermudah Anda fokus pada membangun aplikasi. Fargate menghilangkan perlunya menyediakan dan mengelola server, memungkinkan Anda menentukan dan membayar sumber daya per aplikasi, dan meningkatkan keamanan melalui isolasi aplikasi berdasarkan desain.

Fargate mengalokasikan jumlah komputasi yang tepat, menghilangkan keperluan instans dan meningkatkan skala kapasitas klaster. Anda hanya membayar sumber daya yang diperlukan untuk menjalankan container Anda, sehingga tidak perlu penyediaan berlebihan dan membayar server tambahan. Fargate menjalankan setiap tugas atau pod di dalam kernelnya sendiri yang menyediakan lingkungan komputasi terisolasi tersendiri bagi tugas dan pod. Ini memungkinkan aplikasi Anda memiliki isolasi beban kerja dan peningkatan keamanan berdasarkan desain. Inilah mengapa pelanggan seperti Vanguard, Accenture, Foursquare, dan Ancestry memilih menjalankan aplikasi terpenting mereka pada Fargate.

Gambar 2.9 Logo Aws Fargate

2.4. Docker

Docker adalah platform perangkat lunak yang memungkinkan *developer* membuat, menguji, dan menerapkan aplikasi dengan cepat. Docker mengemas perangkat lunak ke dalam unit standar yang disebut kontainer yang memiliki semua yang diperlukan perangkat lunak agar dapat berfungsi termasuk pustaka, alat sistem, kode, dan waktu proses[6]. Dengan menggunakan Docker, *developer* dapat dengan cepat menerapkan dan menskalakan aplikasi ke lingkungan apa pun dan yakin bahwa kode akan berjalan.

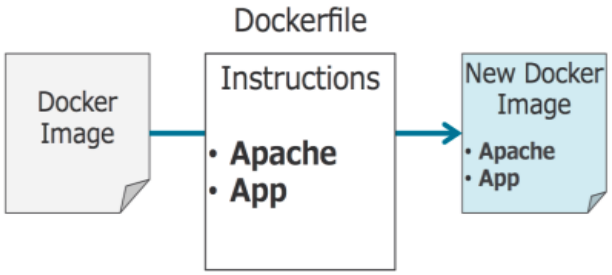
Menjalankan Docker di AWS memberi pengembang dan admin cara yang sangat andal dan murah untuk membuat, mengirim, dan menjalankan aplikasi terdistribusi dalam segala skala. AWS mendukung kedua model lisensi Docker: Docker Community Edition (CE) open source dan Docker Enterprise Edition (EE) berbasis langganan.

2.4.1. Dockerfile

Dockerfile merupakan kumpulan script atau intruksi perintah berbasis *command line* yang digunakan untuk mengunduh dan menjalankan aplikasi secara otomatis pada *docker container* [7]. Pembentukan Dockerfile merupakan tahap awal untuk membangun sebuah komponen docker image yang nantinya berguna untuk menjalankan aplikasi di environtment *development* atau *production*. Setiap dockerfile yang dibuat nantinya akan dikemas kedalam sebuah file teks berekstensi Dockerfile untuk dijadikan *docker image*, dalam hal ini proses *continuos delivery*

berguna untuk mengotomatisasi pembuatan *docker image* tanpa adanya campur tangan *software engineer*.

Hasil dari pengkemasan dockerfile yaitu adalah sebuah *docker image* yang nantinya akan dipakai sesuai dengan kebutuhan. Tahapan pengemasan dockerfile berawal dari pemilihan docker image dasar yang digunakan hal ini sangat penting untuk dilakukan dikarenakan kita akan membuat landasan paling awal dockerfile, landasan tersebut bisa berupa sistem operasi, bahasa pemrograman dan lain sebagainya. Setelah kita melakukan pemilihan docker image dasar kita akan diwajibkan dengan memasukan instruksi – instruksi yang dibutuhkan pada pembangunan *docker image* menggunakan dockerfile, instruksi – instruksi berupa instalasi software maupun konfigurasi yang dibuthkan. Hasil akhir dari pengemasan dockerfile adalah sebuat *docker image*.



Gambar 2.10 Proses Pengemasan Dockerfile

2.4.2. Docker Image

Docker Image adalah *template* yang bersifat *read only* (hanya bisa dibaca saja) dan bertujuan untuk mendefinisikan *docker container*[8]. *Docker image* yang berisi sistem operasi, aplikasi perangkat lunak maupun kode yang akan dimasukkan pada *docker container*. *Docker image* sendiri dapat digunakan menjadi *source code* untuk *docker container* yang mana sangat *portable*.

Kode yang terdapat pada *docker image* merupakan format berlapis menggunakan *union file system* yang dibangun langkah demi langkah menggunakan beberapa instruksi, seperti:

1. Add a file

   Add a file adalah salah satu fitur yang terdapat di dalam docker image yang berguna untuk melakukan *copy file* dari *folder* aplikasi kedalam *docker image*.
2. Run a command

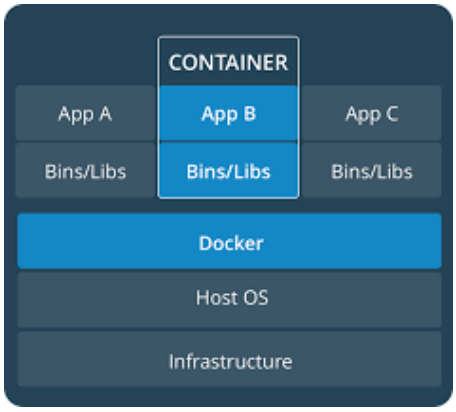
   Dalam *docker image* dapat dilakukan proses menjalankan aplikasi dengan menggunakan attribute CMD atau RUN pada instruksi *docker image*.
3. Open a port

   Dalam *docker image* dapat memunculkan *port* tertentu (*expose port*) sesuai dengan aplikasi yang dijalankan di dalam *docker image.*

2.4.3. Docker Container

Docker Container dapat berjalan diatas *docker images* dan juga dapat melakukan sath atau lebih proses yang berjalan pada *docker images*[8]. *Docker container* sangat berbeda dengan *docker image*, karena *docker container* berbentuk sebuah *docker image* yang dapat dikemas, dibaca dan ditulis (*writeable*). *Layer* baru diatas dasar *docker image* akan terbentuk disaat ada perubahan yang disimpan dalam *docker container*.

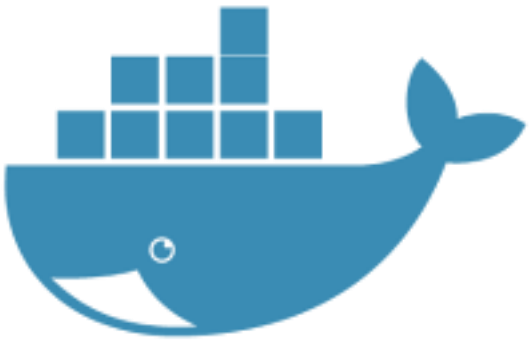
Di dunia nyata proses pengiriman barang melalui kontainer atau *shipping container* merupakan ide awal yang mencetus adanya *docker container* saat ini. Bila dianalogikan dalam istilah *shipping container,* setiap kontainer berisi aplikasi perangkat lunak yang berarti seperti halnya kargo, kemudian kontainer tersebut dapat diberikan intruksi (*command*) untuk melakukan *created, started, stopped, restarted* dan *destroyed*.



Gambar 2.11 Skema Docker Container

2.4.4. Docker Registry

Docker Registry adalah kumpulan *docker image* yang bersifat *private* (akses hanya orang tertentu) atau *public* (semua orang bisa mengakses) dan dapat diakses di Dockerhub maupun *docker registry* yang telah dilakukan instalasi sendiri[9]. Dengan menggunakan *docker registry*, seseorang dapat menggunakan *docker image* yang telah dibuat oleh pengembangan yang lain, sehingga mempermudah dalam pengembangan aplikasi. Docker secara resmi menyediakan *docker registry* yang bisa kita gunakan secara *public* yaitu docker hub. Namun demikian terdapat juga beberapa vendor pihak ketiga yang menyediakan *docker registry* antara lain Google Container Registry atau AWS Elastic Container Registry.



Gambar 2.12 Logo Dockerhub

2.5. Makefile

Makefile adalah *file* yang berisi serangkaian arahan yang digunakan oleh alat otomatisasi *make build* untuk menghasilkan target / sasaran [21]. File deskripsi (makefile) berisi deskripsi hubungan antar file, dan perintah yang perlu dijalankan untuk memperbarui target untuk mencerminkan perubahan dalam prasyaratnya. Setiap spesifikasi, atau aturan, harus terdiri dari target, prasyarat opsional, dan perintah opsional yang harus dijalankan ketika prasyarat lebih baru daripada target.

2.6. Version Control (VCS)

Version Control adalah sebuah sistem yang mencatat perubahan file dari waktu ke waktu [10]. Sebagai contoh, seorang *web designer* akan menyimpan setiap versi dari rancangan yang dibuatnya. Dengan menggunakan VCS maka *developer* memungkinkan untuk kembali ke berkas rancangan sebelumnya, membandingkan perubahan yang terjadi, dan melihat siapa yang telah membuat perubahan.

Jika seorang perancang grafis atau web dan ingin menyimpan setiap versi dari sebuah gambar atau *layout*, sebuah Version Control System (VCS) adalah hal yang bijak untuk digunakan. VCS memperbolehkan untuk mengembalikan berkas-berkas ke keadaan sebelumnya, mengembalikan seluruh proyek kembali ke keadaan sebelumnya, membandingkan perubahan-perubahan di setiap waktu, melihat siapa yang terakhir mengubah sesuatu yang mungkin menimbulkan masalah, siapa dan kapan yang mengenalkan sebuah isu dan banyak lagi. Menggunakan VCS secara umum juga berarti bahwa jika terjadi kesalahan atau kehilangan berkas, maka dapat dengan mudah memulihkannya. Semua hal tersebut dapat digunakan dengan biaya yang sangat sedikit.

2.6.1. Github

Github adalah layanan *open source* untuk proyek pengembangan perangkat lunak yang menggunakan sistem kendali versi Git dan layanan *hosting* internet. Github banyak digunakan untuk kode komputer karena memberikan kontrol akses dan beberapa fitur kolaborasi seperti pelacakan *bug*, permintaan fitur,

manajemen tugas, dan *wiki* untuk setiap proyek[11]. Fungsi utama dari Github adalah mengatur versi dari *source code* program dengan memberikan tanda baris dan baris kode mana yang ditambah, dihapus, dimodifikasi atau dilakukan penggabungan kode antar kode proyek.

Github menawarkan paket repositori pribadi dan gratis pada akun yang sama dan digunakan untuk proyek perangkat lunak sumber terbuka. Pada bulan April 2017, Github melaporkan bahwa mereka mempunyai lebih dari 20 juta pengguna dan lebih dari 57 juta repositori, menjadikannya layanan terbesar dari kode sumber di dunia.

Gambar 2.13 Logo Github

2.7. Extreme Programming

Extreme programming (XP) adalah metodologi pengembangan perangkat lunak yang dimaksudkan untuk meningkatkan kualitas perangkat lunak dan responsif terhadap perubahan kebutuhan pelanggan[12]. Sebagai jenis pengembangan perangkat lunak yang cepat dan responsif sesuai dengan kebutuhan pelanggan, pada metode pengembangan perangkat lunak ini menganjurkan "rilis" dalam siklus pengembangan secara singkat, yang dimaksudkan untuk meningkatkan produktivitas dan memperkenalkan tahap-tahapan pemeriksaan aplikasi perangkat lunak di mana syarat-syarat berikut menjadi prasyarat yang dapat digunakan oleh pelanggan.

Sejarah pengembangan *extreme programming* diawali oleh Kent Beck yang mengembangkan pemrograman ekstrim selama bekerja pada proyek sistem penggajian chrysler. Beck menjadi pemimpin proyek tersebut pada Maret 1996. Dia mulai memperbaiki metodologi pengembangan yang digunakan dalam proyek dan

menulis buku tentang metodologi (*Extreme Programming*, diterbitkan pada Oktober 1999).

Elemen-elemen lain dari *extreme programming* meliputi pemrograman berpasangan (*pair programming*) atau melakukan tinjauan kode secara keseluruhan, *unit testing* terhadap semua kode, kesederhanaan dan kejelasan kode, dan sering berkomunikasi dengan pelanggan dan *software engineer* berkaitan dengan kebutuhan yang dibutuhkan oleh pelanggan. Penamaan metodologi ini diambil dari namanya yaitu gagasan bahwa elemen menguntungkan dari praktik rekayasa perangkat lunak tradisional dibawa ke tingkat yang lebih lanjut (*ekstrim*).

2.8. Node JS

Node JS adalah bahasa pemrograman yang dikembangkan secara *open source*, lintas *platform* yang bisa digunakan oleh aplikasi *desktop*, web maupun *command line* di *console terminal*. Node.js memungkinkan pengembang menggunakan Javascript untuk membuat sebuah *website* yang menghasilkan konten halaman web dinamis sebelum halaman dikirim ke *browser* pengguna. Oleh karena itu, Node JS mewakili paradigma "JavaScript everywhere", yang berakibat pengembangan aplikasi web hanya menggunakan satu bahasa pemrograman yaitu Javascript baik di bagian *server* maupun *client*. Node JS awalnya ditulis oleh Ryan Dahl pada tahun 2009, sekitar tiga belas tahun setelah pengenalan lingkungan Javascript di sisi *server* pertama kali yaitu LiveWire Pro Web Netscape. Pada perilisan awal, Node JS hanya mendukung sistem operasi Linux dan Mac OS. Pengembangan dan pemeliharaannya dipimpin oleh Dahl dan kemudian disponsori oleh Joyent.

Meskipun .js adalah ekstensi *file* standar untuk kode Javascript, nama "Node.js" tidak merujuk ke file tertentu dan dalam konteks ini dan hanya nama produk atau pustaka saja. Node JS memiliki arsitektur yang digerakkan oleh perubahan yang merujuk kepada *input* dan *output* secara *asynchronous*. Pilihan desain ini bertujuan untuk mengoptimalkan performa dan skalabilitas dalam aplikasi web dengan banyak operasi *input*/*output* yang terdapat di dalamnya. Beberapa korporat yang menggunakan Node JS sebagai penunjang pembangunan

perangkat lunak diantaranya adalah GoDaddy, Groupon, IBM, LinkedIn, Microsoft, Netflix, PayPal, Rakuten, SAP, Voxer, Walmart, dan Yahoo !.

Gambar 2.14 Logo Node JS

2.8.1. NPM

NPM adalah manajer distribusi paket untuk pemograman *JavaScript* untuk environment Node.js. Pengembang *open source* menggunakan npm untuk berbagi dan meminjam paket, dan banyak organisasi menggunakan npm untuk mengelola pengembangan private juga[22]. Npm memiliki 3 komponen yaitu Web, CLI (*Command Line Interface*) dan *registry*.

Ketika npm digunakan untuk manajer distribusi paket lokal, npm bisa menginstal semua *dependency package* dalam satu perintah dengan satu perinah berkas package.json. Dalam berkas package.json, setiap paket dapat ditentukan versi yang akan dipasang dengan menggunakan skema versi semantik. Hal ini memungkinkan pengembang untuk melakukan pemutakhiran paket otomatis dan juga menghindari benturan perubahan yang tidak diinginkan

Gambar 2.15 Logo Npm

2.9. Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) adalah bahasa yang digunakan untuk mempercantik tampilan dan menggambarkan penyajian dokumen yang ditulis dalam bahasa *markup* seperti HTML[13]. CSS adalah teknologi dasar yang digunakan di *World Wide Web*, bersama HTML dan Javascript. CSS dirancang untuk memungkinkan pemisahan presentasi dan konten, termasuk tata letak, warna, dan font. Pemisahan ini dapat meningkatkan aksesibilitas konten, memberikan lebih banyak fleksibilitas dan kontrol dalam spesifikasi karakteristik presentasi, memungkinkan beberapa halaman web untuk berbagi format dengan menentukan CSS yang relevan dalam file .css yang terpisah, dan mengurangi kompleksitas dan pengulangan dalam konten struktural. Pemisahan pemformatan dan konten juga memungkinkan untuk menyajikan halaman *markup* yang sama dalam tampilan yang berbeda untuk metode penampilan HTML. CSS juga memiliki aturan untuk pemformatan yang disesuaikan jika konten diakses pada perangkat seluler (*mobile*).

Spesifikasi CSS yang dikelola oleh World Wide Web Consortium (W3C). Jenis media internet atau MIME Type dari file css adalah teks/css yang terdaftar pada aturn RFC 2318. W3C memberikan layanan validasi CSS secara gratis untuk dokumen CSS. Selain HTML, bahasa markup lainnya mendukung penggunaan CSS adalah XHTML, XML, SVG, dan XUL.

Gambar 2.16 Logo CSS 3

2.10. Continuous Integration

Continuous Integration adalah salah satu kegiatan dalam pengembangan aplikasi *software* yang seluruh hasil kerja dari developer digabungkan ke dalam satu wadah[7]. Proses penggabungan bisa dalam kurun waktu yang ditetapkan, tergantung dari proyek pengembangan perangkat lunak tersebut berjalan. Kata kuncinya adalah bahwa seluruh hasil kerja digabungkan di satu tempat. Dengan menggabungkan hasil kerja dalam kurun waktu tertentu, maka dapat mengurangi kemungkinan konflik besar terjadi. Karena pengerjaan pengembangan aplikasi akan secara otomatis dilakukan sedikit demi sedikit oleh masing-masing *software engineer*. Para *software engineer* tidak akan menunggu hingga modul besar selesai atau bagian *library* lain lengkap, namun akan membagi dalam sub-modul agar dapat melakukan *commit* ke repositori minimal dalam kurun waktu yang disepakati.

Selain itu, dengan adanya satu wadah dimana seluruh hasil kerja digabungkan, maka dapat dilakukan testing secara menyeluruh. Testing bisa dilakukan dengan *testing developer* atau dengan konsep *Test Driven Development*, dimana testing dilakukan otomatis.

2.10.1. Unit Testing

Unit testing adalah metode verifikasi perangkat lunak dimana *software engineer* menguji suatu unit program layak untuk tidaknya dipakai[14]. *Unit testing* hanya berfokus pada verifikasi unit terkecil pada desain/kode perangkat lunak (komponennatau modul perangkat lunak).

Dalam sebuah perangkat lunak terdapat unit-unit kecil maka untuk mengujinya biasanya dibuat program kecil atau main program untuk menguji unit-unit perangkat lunak. Unit-unit kecil ini dapat berupa prosedur atau fungsi, sekumpulan prosedur atau fungsi yang ada dalam satu *file* jika dalam pemrograman terstruktur, atau kelas, bisa juga kumpulan kelas dalam satu package dalam PBO. Pengujian unit biasanya dilakukan saat kode program dibuat.

2.10.2. Test Driven Development

Test Driven Development adalah adalah proses pengembangan perangkat lunak yang bergantung pada pengulangan siklus pengembangan yang sangat singkat[15]. Persyaratan diubah menjadi kasus uji yang sangat spesifik, kemudian kode ditingkatkan sehingga tes lulus. Ini bertentangan dengan pengembangan perangkat lunak yang memungkinkan ditambahkannya kode yang tidak terbukti memenuhi persyaratan. *Sofware engineer* Amerika, Kent Beck, yang dianggap telah mengembangkan atau 'menemukan kembali' teknik tersebut, menyatakan pada tahun 2003 bahwa TDD mendorong desain sederhana dan menginspirasi kepercayaan. Pengembangan yang digerakkan oleh tes terkait dengan konsep pemrograman uji-pertama *extreme programming*, dimulai pada tahun 1999, tetapi baru-baru ini telah menciptakan minat yang lebih umum pada haknya sendiri. Pemrogram juga menerapkan konsep ini untuk meningkatkan dan *debugging* kode lama yang dikembangkan dengan teknik lama. Berikut adalah contoh cara implementasi *Test Driven Development* menurut buku adalah sebagai berikut.

1. Menambahkan naskah pengujian

   Dalam *test driven development*, setiap fitur baru dimulai dengan menulis tes. Tulis skenario tes yang mendefinisikan fungsi atau peningkatan fungsi, yang seharusnya sangat ringkas. Untuk menulis skenario tes, *software engineer* harus memahami spesifikasi dan persyaratan fitur dengan jelas. *Software engineer* dapat mencapai ini melalui kasus penggunaan dan kisah pengguna untuk mencakup persyaratan dan kondisi pengecualian, dan dapat menulis skenario tes dalam kerangka pengujian apa pun yang sesuai dengan lingkungan perangkat lunak. Itu bisa menjadi versi modifikasi dari tes yang ada.

2. Menjalankan semua skenario pengujian

   Ini memvalidasi bahwa proses pengujian harus berjalan dengan benar, menunjukkan bahwa skenario tes baru tidak lulus tanpa memerlukan kode baru karena perilaku yang diperlukan sudah ada, dan mengesampingkan kemungkinan bahwa tes baru cacat dan akan selalu lulus. Tes baru harus gagal karena alasan yang diharapkan. Langkah ini meningkatkan kepercayaan *software engineer* dalam pengujian baru.

3. Menulis kode

   Langkah selanjutnya adalah menulis beberapa kode yang menyebabkan pengujian lulus. Kode baru yang ditulis pada tahap ini tidak sempurna dan mungkin akan terjadi misalnya, lulus ujian dengan cara yang tidak sesuai dengan ekspektasi . Itu dapat diterima karena akan ditingkatkan dan diasah pada tahap 5. Pada titik ini, satu-satunya tujuan kode tertulis adalah untuk lulus ujian. *Software engineer* tidak boleh menulis kode yang melampaui fungsi yang diperiksa oleh tes.

4. Menjalankan *Testing*

   Jika semua pengujian lolos tidak terdapat *error* maupun *bug*, *software engineer* dapat yakin bahwa kode baru memenuhi persyaratan pengujian, dan tidak merusak atau menurunkan fitur yang ada. Jika terdapat *error* maupun *bug* maka kode baru tersebut harus disesuaikan hingga sesuai dengan kualifikasi skenario tes.

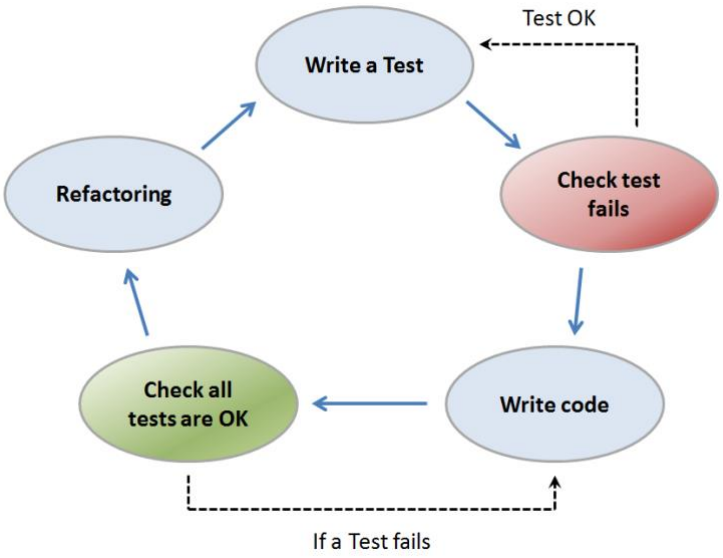
5. *Refactor* kode

   Basis kode yang berkembang harus dibersihkan secara teratur selama *Test Driven Development.* Kode baru dapat dipindahkan dari tempat yang sesuai untuk lulus dalam pengujian kode. Duplikasi kode harus dihapus, nama objek, kelas, modul, variabel dan metode harus jelas mewakili tujuan dan penggunaannya saat ini, karena fungsi tambahan telah ditambahkan. Ketika fitur ditambahkan, badan metode bisa lebih panjang dan objek lain lebih besar.

6. Mengulang semua tahapan

   Dimulai dengan penambahan skenario tes baru lainnya, siklus kemudian diulang untuk mendorong kebutuhan fungsionalitas. Ukuran langkah-langkah harus selalu kecil, dengan sedikit antara 1 hingga 10 suntingan antara setiap uji coba. Jika kode baru tidak dengan cepat memenuhi tes baru, atau tes lain gagal secara tak terduga, *programmer* harus membatalkan atau mengembalikan preferensi untuk *debugging* yang berlebihan. Integrasi berkelanjutan membantu dengan menyediakan pos pemeriksaan yang dapat dikembalikan. Ketika menggunakan perpustakaan eksternal, penting untuk tidak membuat peningkatan yang sangat kecil sehingga hanya efektif menguji kode itu sendiri,

kecuali terdapat alasan untuk membiarkan kode itu bermasalah atau tidak cukup fitur lengkap untuk melayani semua kebutuhan perangkat lunak yang sedang dikembangkan.



Gambar 2.17 Lifecycle Test Driven Development.

2.10.3. Jest

Jest adalah pustaka pengujian JavaScript terbuka dari Facebook[16]. Slogannya adalah "Pengujian JavaScript yang Menyenangkan". Sementara Jest dapat digunakan untuk menguji perpustakaan JavaScript apa pun, itu bersinar ketika datang ke React dan React Native. Ini tidak mengherankan karena React dan Jest berasal dari Facebook, yang merupakan pengguna utama keduanya.

Jest hadir dengan pencocokan bawaan, mata-mata, dan perpustakaan tiruannya yang luas. Dulu didasarkan pada Jasmine, sehingga mewarisi semua kebaikan Jasmine. Tetapi dalam versi yang lebih baru Jest berangkat dari Jasmine, namun tetap memiliki fungsionalitas yang sama dan menambahkan rasa dan peningkatannya sendiri. Ketika membandingkannya dengan solusi pengujian khusus berdasarkan Mocha, jelas bahwa kemudahan penggunaan merupakan perhatian utama dari desain Jest.

Jest memiliki laporan cakupan bawaan. Jika *programmer* menguji hanya 80% dari kode, maka bug di 20% lainnya akan ditemukan hanya dalam produksi. Terkadang, masuk akal dari perspektif bisnis untuk melewati pengujian untuk beberapa bagian sistem. Misalnya, alat internal yang sering digunakan dan diubah oleh teknisi ahli mungkin tidak memerlukan tingkat pengujian ketat yang sama dengan kode produksi. Tetapi, bagaimanapun juga, ini harus menjadi keputusan sadar dan harus dapat melihat dengan tepat cakupan tes dari berbagai bagian sistem.

Gambar 2.18 Logo Jest

2.10.4. Linter

Lint atau linter adalah alat untuk menganalisa *source code* untuk menandai *errors, bug, code convention* dan memverifikasi kualitas kode[17]. Dengan adanya linter kita dipaksa untuk belajar dan disiplin untuk menulis kode yang bagus dan rapi tapi tetap tidak *bug free*. Untuk proyek skala besar linter sangat dianjurkan agar semua yang ikut berkontribusi menggunakan *code convention* yang sama dan tidak berbeda-beda. Berikut adalah contoh linter yang biasa digunakan pada bahasa pemrograman CSS dan Javascript.

1. Javascript Linter

   Berikut adalah linter yang biasa dan terkenal digunakan untuk bahasa pemrograman Javascript:

   a. Eslint

   b. Prettier

c. StandardJS
d. JSLint
e. JSHint

2. CSS Linter

Berikut adalah linter yang biasa dan terkenal digunakan untuk bahasa pemrograman css:

a. Stylelint
b. SCSSLint
c. Stylefmt
d. CSSLint

2.10.5. ESLint

ESLint adalah alat analisis kode statis untuk mengidentifikasi kode bermasalah yang ditemukan dalam kode Javascript[18]. Eslint dibuat oleh Nicholas C. Zakas pada tahun 2013 silam. Aturan dalam ESLint dapat dikustomisasi, dan aturan khusus dapat didefinisikan dan dimuat. ESLint mengatur kualitas kode dan masalah konsistensi kode. ESLint mendukung standar ECMAScript saat ini, dan *exprerimental script* dari draft untuk standar di masa mendatang. Kode menggunakan JSX atau TypeScript juga dapat diproses ketika *plugin* atau *transpiler* digunakan.

Salah satu fungsi ESLint adalah untuk memperingatkan jika ada baris kode yang berpotensi *error* maupun *bug*. Selain membuat lebih waspada terhadap *error* maupun bug salah satu tujuan ESLint adalah membuat *software engineer* dapat mengarahkan struktur kode menjadi lebih konsisten.

Gambar 2.19 Logo Eslint Linter

2.10.6. Stylelint

Stylelint adalah alat yang berfungsi mengecek CSS yang memiliki konfigurasi dari ratusan aturan-aturan yang telah ditetapkan. Proses berjalannya stylelint adalah berawal dari melakukan parsing kepada kamus data di css, lalu melakukan perbaikan satu persatu baris secara sekuensial. Fitur yang tersedia pada pustaka Stylelint adalah dapat melakukan pengubahan string CSS menjadi string CSS baru yang telah diperbaiki secara penulisan sintaksis dan kamus data. Pustaka Stylelint ditulis dalam bahasa Javascript dan didistribusika melalui plattform npm.

Gambar 2.20 Logo StyleLint Linter CSS

2.10.7. Code Coverage

Dalam ilmu komputer, *code coverage* adalah ukuran yang digunakan untuk menggambarkan sejauh mana kode dari suatu program dieksekusi ketika *unit test* berjalan[19]. Sebuah program dengan *code coverage* tinggi, diukur sebagai

persentase, memiliki lebih banyak kode sumbernya dieksekusi selama pengujian, yang menunjukkan kemungkinan lebih kecil untuk mengandung *bug* perangkat lunak yang tidak terdeteksi dibandingkan dengan program dengan cakupan pengujian rendah. Banyak matriks yang berbeda dapat digunakan untuk menghitung code *coverage*, beberapa yang paling mendasar adalah persentase sub rutin program dan persentase pernyataan program yang dipanggil selama pelaksanaan *unit test*.

Cakupan pengujian adalah salah satu metode pertama yang ditemukan untuk pengujian perangkat lunak sistematis. Referensi pertama yang diterbitkan adalah oleh Miller dan Maloney dalam Communications of the ACM pada tahun 1963.

Perangkat lunak yang akan dibangun dengan pilihan atau pustaka khusus dan berjalan di bawah lingkungan yang terkontrol, untuk memetakan setiap fungsi yang dieksekusi ke titik fungsi dalam kode sumber. Ini memungkinkan pengujian bagian-bagian dari perangkat lunak target yang jarang atau tidak pernah diakses dalam kondisi normal, dan membantu memastikan bahwa kondisi yang paling penting (titik fungsi) telah diuji. *Output* yang dihasilkan kemudian dianalisis untuk melihat area kode mana yang belum dilakukan dan tes diperbarui untuk memasukkan area-area ini sebagaimana diperlukan. Dikombinasikan dengan metode cakupan tes lainnya, tujuannya adalah untuk mengembangkan serangkaian uji regresi yang ketat, namun dapat dikelola.

2.11. Continuos Delivery

Continuous Delivery (CD) adalah pendekatan rekayasa perangkat lunak di mana tim memproduksi perangkat lunak dalam siklus yang singkat, memastikan bahwa perangkat lunak dapat diandalkan dan dirilis setiap saat[7]. Hal ini bertujuan untuk membangun, melakukan pengujian, dan merilis *software* lebih cepat dan lebih sering sehingga antar pengembangan perangkat lunak dan operasi (DevOps) dapat saling menilai hasil kinerja satu dengan lainnya.

Pendekatan ini membantu mengurangi biaya, waktu, dan risiko dalam perubahan dengan memungkinkan untuk menambah update lebih banyak untuk

aplikasi yang sedang berjalan. Sebuah proses penyebaran langsung dan berulang sangat penting dalam pengiriman berkelanjutan (*Continuos Delivery*).

2.11.1. Travis CI

Travis CI adalah layanan integrasi berkesinambungan yang di-host yang digunakan untuk membangun dan menguji proyek perangkat lunak yang dihosting di GitHub dan Bitbucket[20]. Travis CI menyediakan berbagai rencana berbayar untuk proyek-proyek swasta, dan rencana gratis untuk sumber terbuka. TravisPro menyediakan penyebaran kustom versi eksklusif pada perangkat keras milik pelanggan.

Sumbernya adalah perangkat lunak bebas teknis dan sedikit demi sedikit tersedia di GitHub di bawah lisensi permisif. Perusahaan mencatat, bagaimanapun, bahwa sejumlah besar tugas yang perlu dipantau dan dilakukan pengguna dapat menyulitkan beberapa pengguna untuk berhasil mengintegrasikan versi Enterprise dengan infrastruktur mereka sendiri.

Ketika Travis CI telah diaktifkan pada repositori yang telah dipilih, GitHub akan memberitahukannya setiap kali software engineer melakukan *commit* baru dan di *push* kedalam repositori itu atau *pull request* yang diajukan. Ini juga dapat dikonfigurasikan untuk hanya berjalan untuk beberapa *branch* tertentu didalam repositori. Travis CI kemudian akan memeriksa branch yang sesuai dan menjalankan perintah yang ditentukan dalam file .travis.yml, yang biasanya membangun perangkat lunak dan menjalankan pengujian secara otomatis. Ketika proses itu telah selesai, Travis memberi tahu *software engineer*, atau dengan memberikan pesan pada *email*. Dalam hal proses *pull request* akan diberi catatan dengan hasil dan tautan menggunakan integrasi GitHub.

Travis CI dapat dikonfigurasikan untuk menjalankan unit test pada berbagai sistem operasi yang berbeda, dengan berbagai perangkat lunak yang diinstal, dan mendukung pembangunan perangkat lunak dalam berbagai bahasa pemrograman termasuk C , C ++, C #, Clojure, D, Erlang, F #, Go, Apache Groovy, Haskell, Java, JavaScript, Julia, Perl, PHP, Python, R, Ruby, Rust, Scala, Swift, dan Visual Basic.