

Panduan Praktik Teknologi Cloud Computing



Jurusan : Teknik Informatika

Disusun oleh:
M. Agung Nugroho, M.Kom
LUTHFAN HADI PRAMONO, S.ST., M.T.

STMIK AKAKOM
Yogyakarta
2017

Daftar Isi

Daftar Isi	1
Pertemuan 1 - Pengenalan Cloud Computing	2
Pertemuan 2 - Pengenalan Software as A services	21
Pertemuan 3 - Implementasi Layanan Business Process as a Services.	24
Pertemuan 4 : Pengenalan Docker	28
Pertemuan 5 - Data as a Service dengan Posgresql	39
Pertemuan 6 - Data as a Service dengan NoSQL MongoDB	44
Pertemuan 7 - Pengantar Platform as a Services	53
Pertemuan 8 - Platform as a service Docker di Openshift	64
Pertemuan 9 - Source Control Management dengan Git	71
Pertemuan 10 - Docker Images	85
Pertemuan 11 - Kubernetes	92
Pertemuan 12 - Konsep IaaS dengan Aplikasi Virtualisasi	97
Pertemuan 13 - Pengantar MbaaS pada platform firebase	105
Pertemuan 14 - Proyek Kelompok	118
Referensi	119

Pertemuan 1 - Pengenalan Cloud Computing

Tujuan

1. Memahami pemanfaatan software as a service
2. Memanfaatkan software as a service untuk online collaborative
3. Mampu membedakan layanan cloud computing

Teori

Komputasi awan (cloud computing) adalah gabungan pemanfaatan teknologi komputer ('komputasi') dan pengembangan berbasis Internet ('awan'). Awan (cloud) adalah metafora dari internet, sebagaimana awan yang sering digambarkan di diagram jaringan komputer. Sebagaimana awan dalam diagram jaringan komputer tersebut, awan (cloud) dalam Cloud Computing juga merupakan abstraksi dari infrastruktur kompleks yang disembunyikannya.] Ia adalah suatu metoda komputasi di mana kapabilitas terkait teknologi informasi disajikan sebagai suatu layanan (as a service), sehingga pengguna dapat mengaksesnya lewat Internet ("di dalam awan") tanpa mengetahui apa yang ada didalamnya, ahli dengannya, atau memiliki kendali terhadap infrastruktur teknologi yang membantunya. Menurut sebuah makalah tahun 2008 yang dipublikasi IEEE Internet Computing "Cloud Computing adalah suatu paradigma di mana informasi secara permanen tersimpan di server di internet dan tersimpan secara sementara di komputer pengguna (client) termasuk di dalamnya adalah desktop, komputer tablet, notebook, komputer tembok, handheld, sensor-sensor, monitor dan lain-lain."

Komputasi awan adalah suatu konsep umum yang mencakup SaaS, Web 2.0, dan tren teknologi terbaru lain yang dikenal luas, dengan tema umum berupa ketergantungan terhadap Internet untuk memberikan kebutuhan komputasi pengguna. Sebagai contoh, Google Apps menyediakan aplikasi bisnis umum secara daring yang diakses melalui suatu penjelajah web dengan perangkat lunak dan data yang tersimpan di server. Komputasi awan saat ini merupakan trend teknologi terbaru, dan contoh bentuk pengembangan dari teknologi Cloud Computing ini adalah iCloud.

Praktik

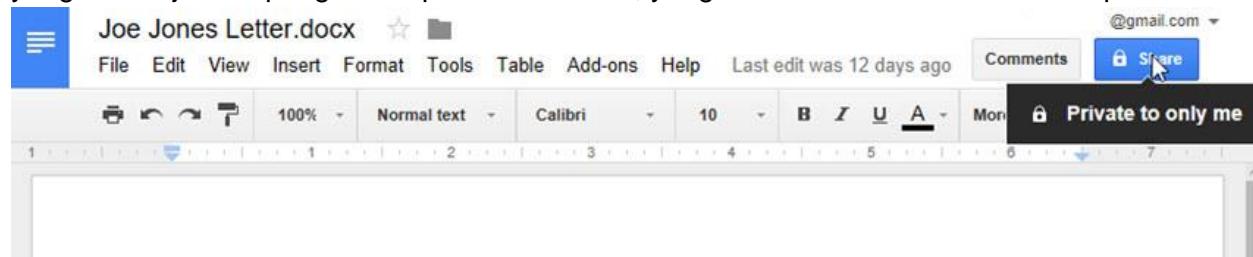
Sekarang, mari kita gali cara membagikan dokumen Anda di Google Docs, mengundang orang untuk berkolaborasi, mengendalikan perizinan dokumen, mendiskusikan dokumen Anda melalui komentar bersama, menangani notifikasi dengan tepat, dan lebih banyak lagi. Temukan apa yang terjadi ketika Anda membagikan suatu dokumen di Google Docs, serta bagaimana mengontrol sejumlah opsi yang bermanfaat.

1. Opsi Visibilitas

Hal pertama yang perlu Anda ketahui tentang berbagi di Google Docs adalah **Opsi Visibilitas**, yang mengontrol siapa saja yang bisa melihat dokumen tersebut. Dokumen, spreadsheet, dan presentasi baru Anda di Google Drive secara default bersifat pribadi, tapi dengan tombol **Share** Anda bisa memilih untuk membuat dokumen Anda bisa dilihat beberapa orang tertentu, siapa saja yang punya tautan dokumen, atau untuk umum di web.

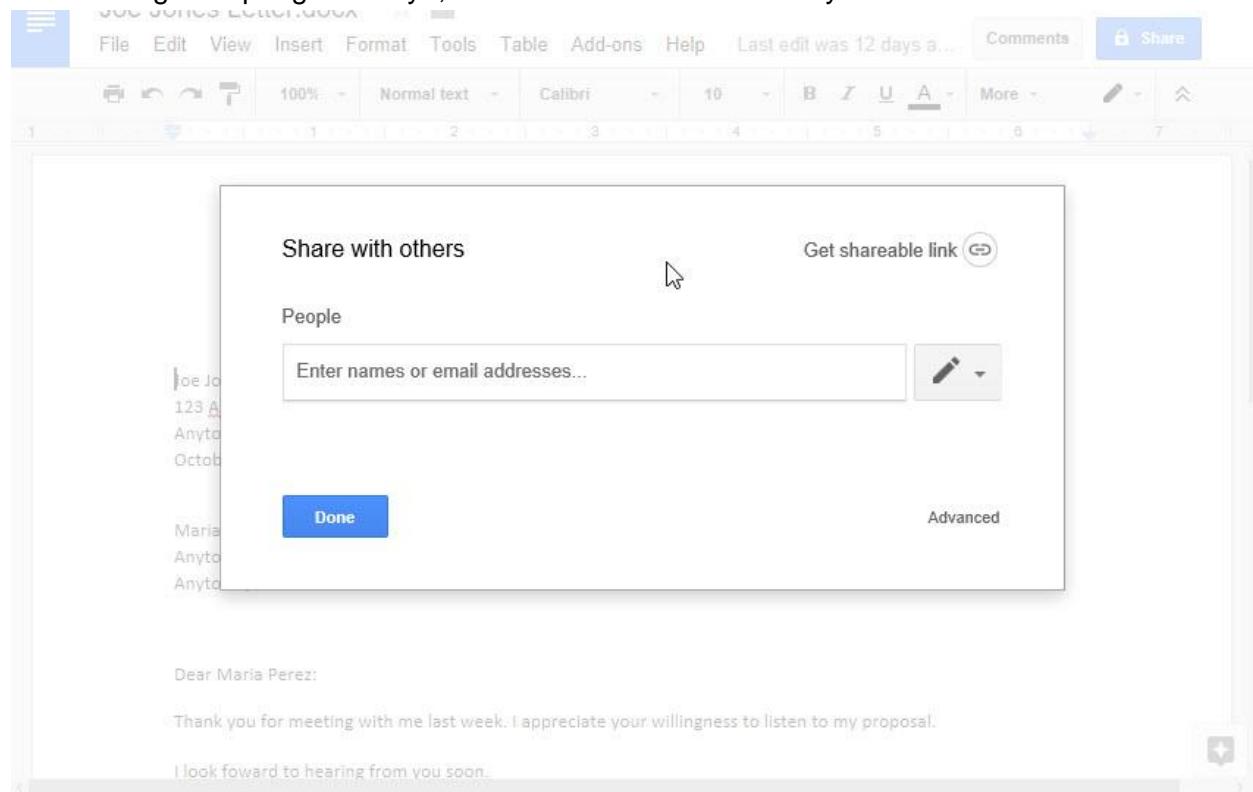
Langkah 1. Bagaimana Membagikan Dokumen di Google Docs

Untuk melihat pengaturan asal Anda ketika membuka dokumen di Google Docs, lewatkan mouse pada tombol **Share** di kanan atas dokumen Anda. Di sana, akan ada menu popover yang menunjukkan pengaturan privasi dokumen, yang secara default selalu bersifat pribadi.



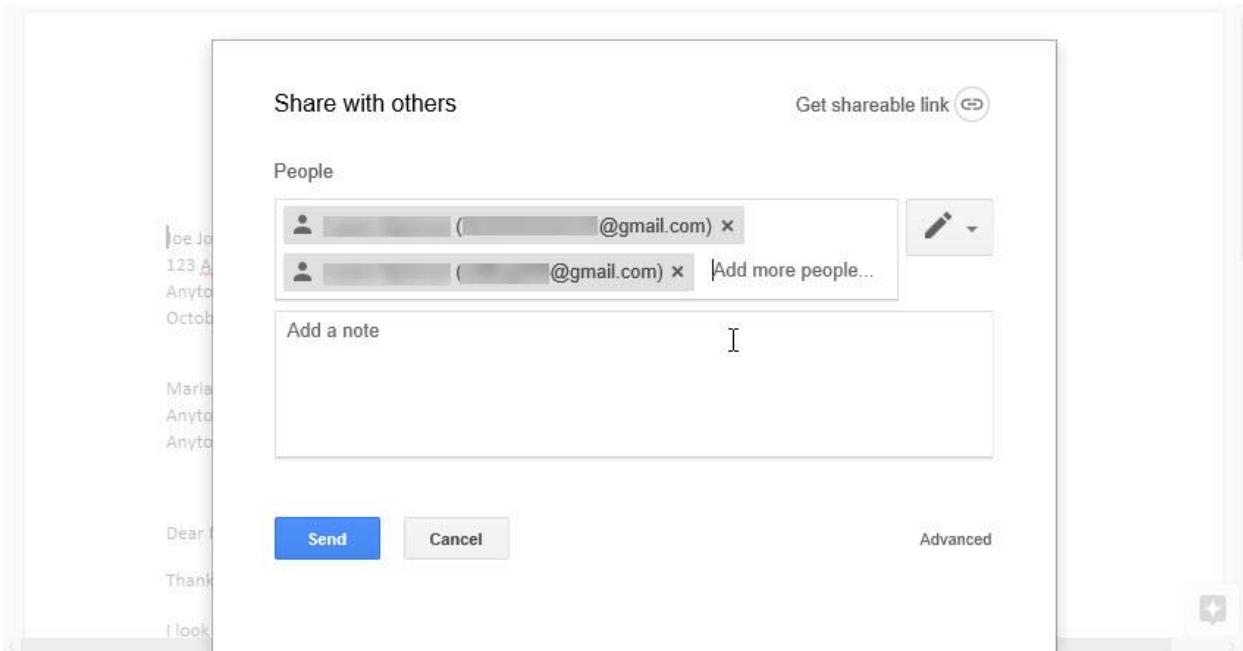
Lewatkan kursor pada tombol **Share** untuk melihat pengaturan yang ada.

Untuk mengubah pengaturannya, klik tombol **Share**. Muncul layar **Share with others**.



Dari layar **Share with Others**, Anda bisa memasukkan sejumlah nama atau alamat email kepada siapa Anda ingin membagikan dokumen di Google Docs.

Ketikkan nama orang dari daftar kontak Anda atau alamat email orang yang Anda ingin membagikan dokumen dengannya.

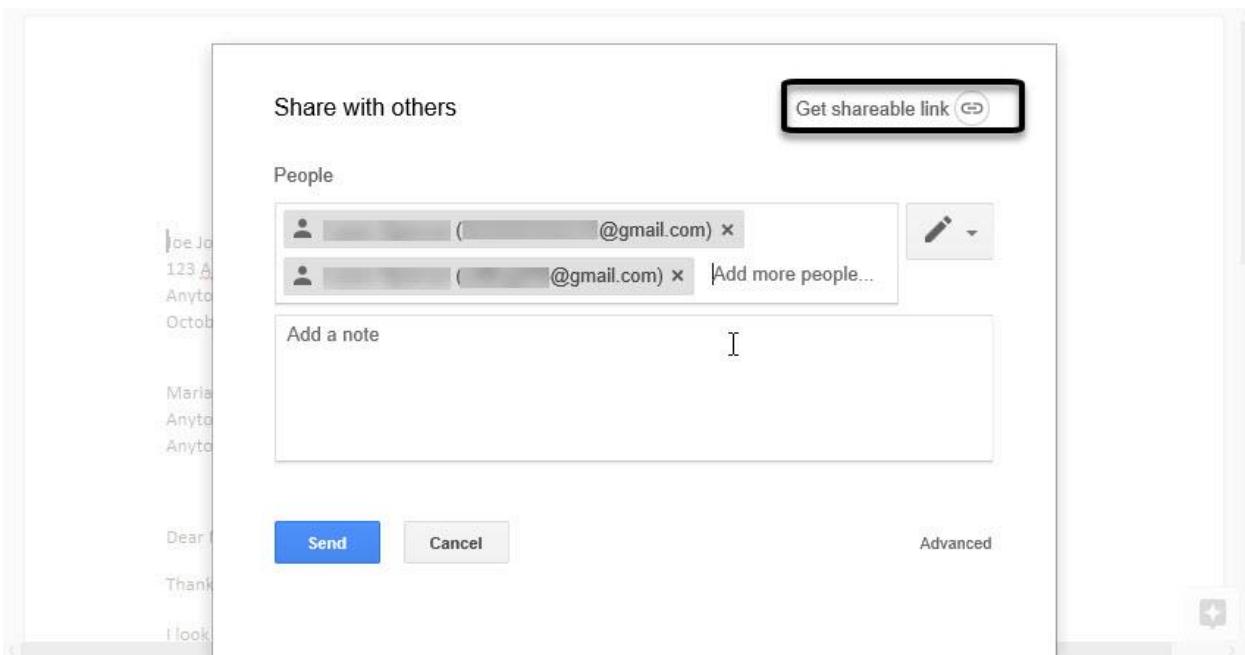


Klik tombol **Send** setelah Anda memasukkan beberapa nama atau alamat email. Lanjutkan menambah nama dan alamat email siapa saja yang Anda ingin membagikan dokumen dengan mereka. Anda juga bisa menambahkan catatan untuk mereka yang berkolaborasi di kotak **Add a note**.

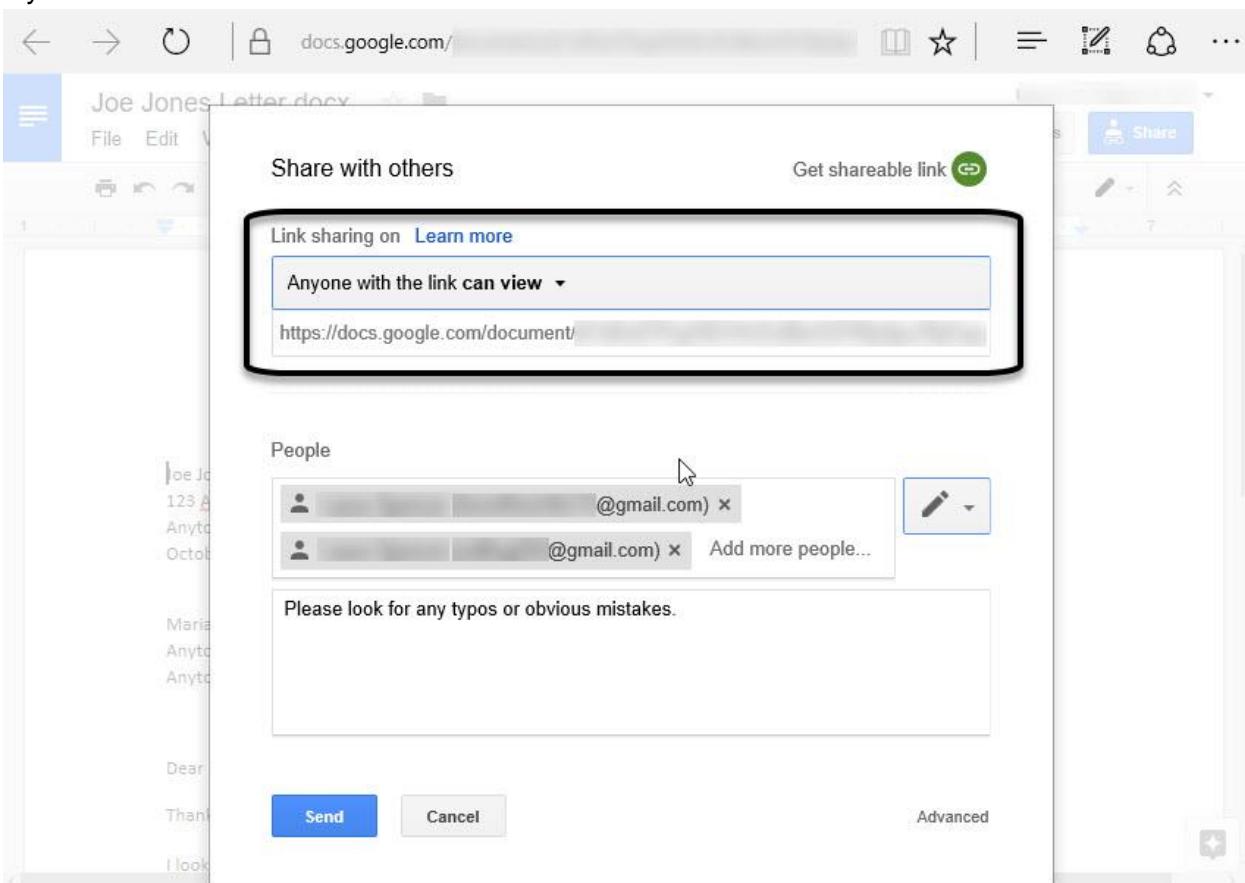
Klik tombol **Send** di sebelah kiri bawah layar begitu Anda selesai menambahkan kolaboratornya.

Langkah 2. Bagaimana Menghidupkan Tautan Opsi Visibilitas

Anda juga bisa menggunakan layar **Share with others** di Google Docs untuk menghidupkan opsi visibilitas. Jika tautan opsi visibilitas diaktifkan, semua orang yang punya tautan ke dokumen Anda akan bisa mengaksesnya.



Klik tautan di sudut kanan atas untuk mendapatkan tautan yang bisa dibagikan.
Untuk mengaktifkan opsi **Link Sharing**, klik ikon tautan di sudut kanan atas layar. Akan muncul layar berisi tautan.

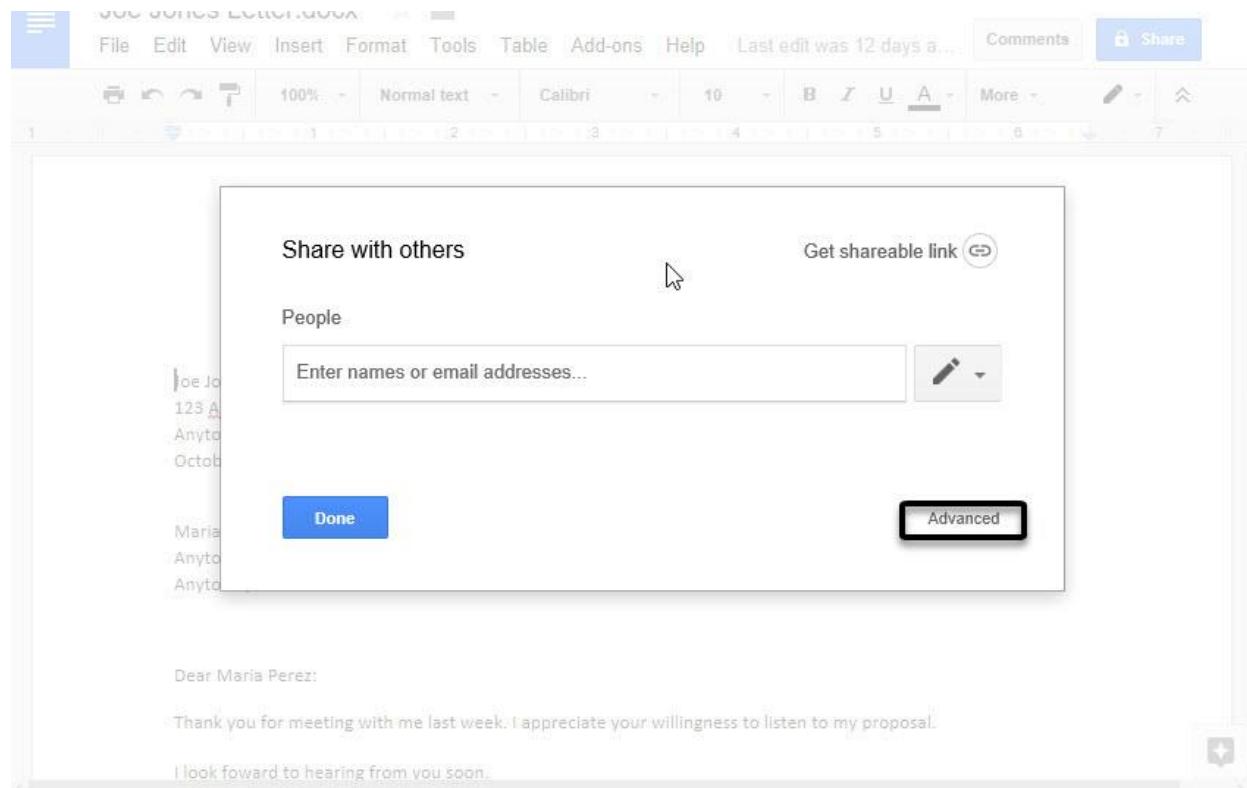


Bagikan tautannya untuk membagikan dokumen yang dimaksud.
Sistemnya menyediakan tautan yang bisa Anda bagikan dengan orang lain.

Catatan: Penting untuk diingat bahwa mengaktifkan opsi tautan berbagi mengurangi intensitas kontrol yang Anda miliki terhadap siapa saja yang bisa melihat dokumennya. Salah satu dari mereka yang punya akses bisa membagikan tautan dengan orang lain, dan orang lain tersebut bisa mengakses dokumen Anda.

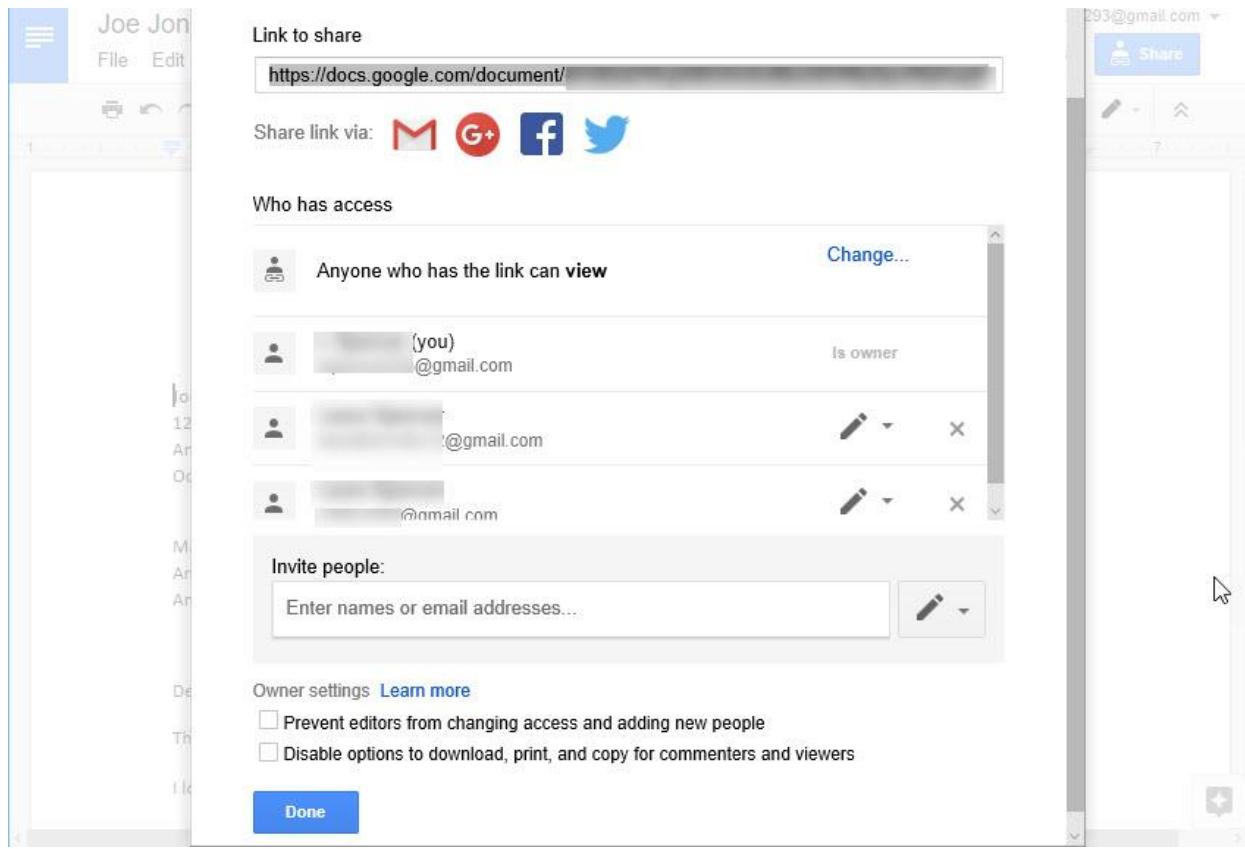
Langkah 3. Menggunakan Advanced Sharing Options di Google Docs.

Anda bisa mendefinisikan lebih jauh siapa yang bisa melihat dokumen Anda. Mulai dari layar **Share with others**.



Klik tautan **Advanced** untuk membuka menu **Sharing Settings**.

Lalu, pilih **Advanced**. Di menu **Sharing settings**, Anda akan bisa mengubah siapa saja yang bisa melihat dokumennya.

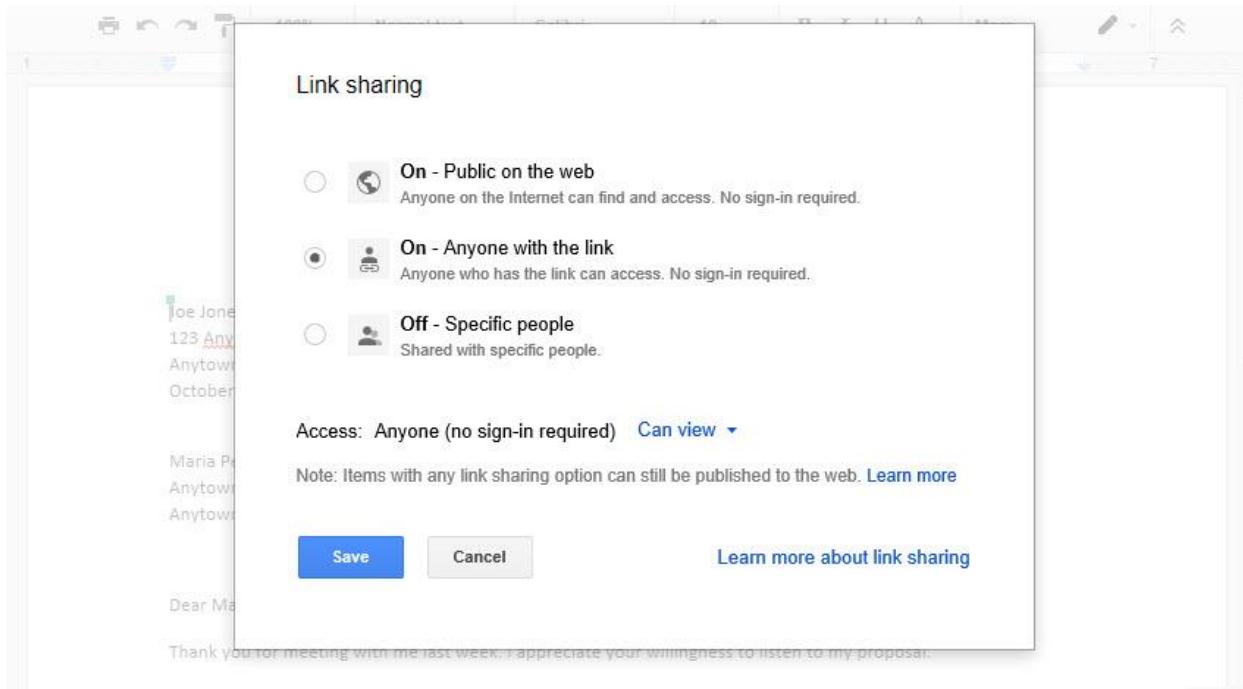


Daftar orang yang bisa mengakses dokumen.

Perhatikan **Owner settings** di bagian bawah halaman. Pengaturan-pengaturan tersebut hanya tersedia bagi pemilik dokumen. Pengaturan tersebut memungkinkan Anda membatasi apa yang bisa dilakukan para kolaborator terhadap dokumennya.

Langkah 4. Mendefinisikan Opsi Link Sharing

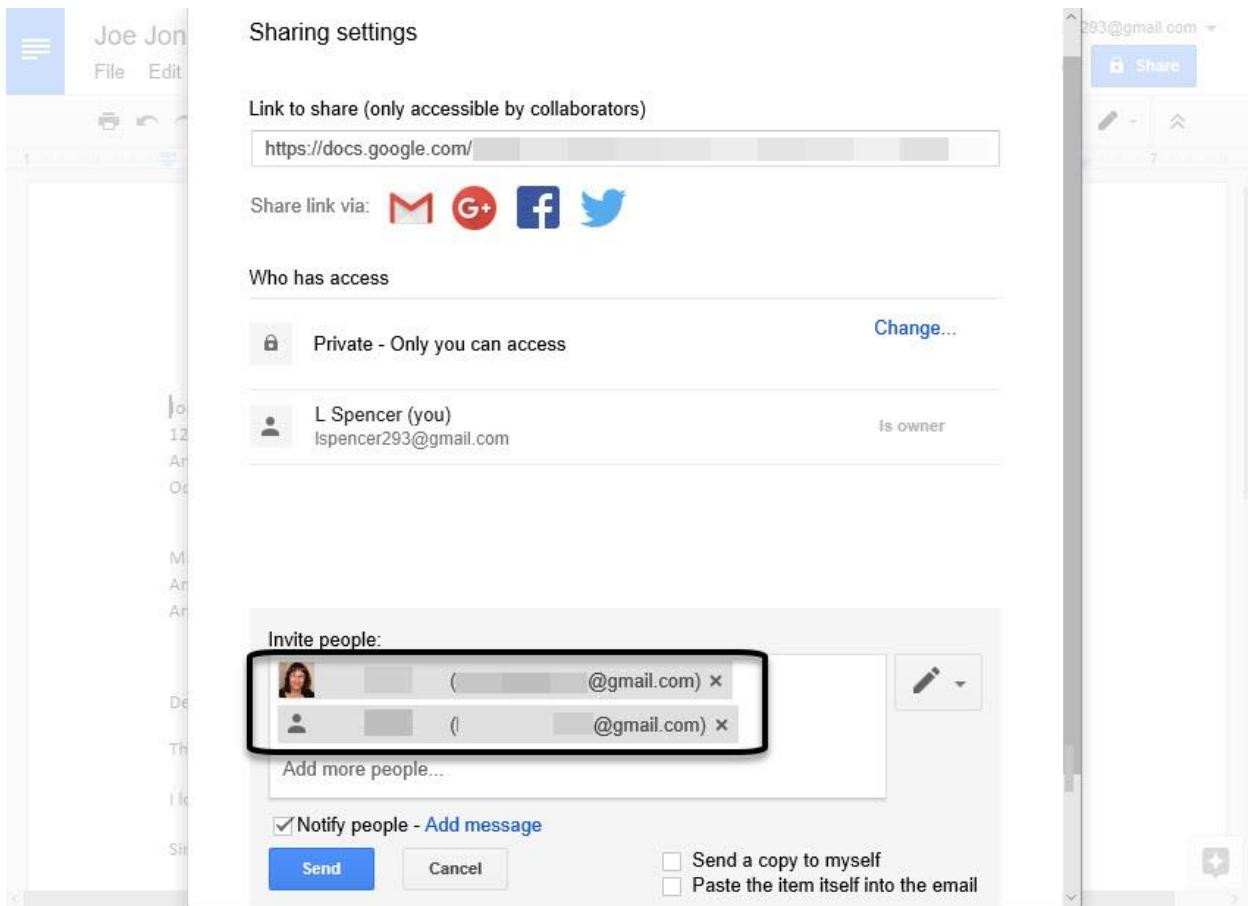
Dari menu **Sharing settings**, pilih **Change** untuk menampilkan opsi-opsi **link sharing**.



Dari menu **Sharing settings**, pilih Change untuk menampilkan pelbagai opsi lainnya. Berikut adalah penjelasan bagi tiap opsi tersebut:

1. Specific People

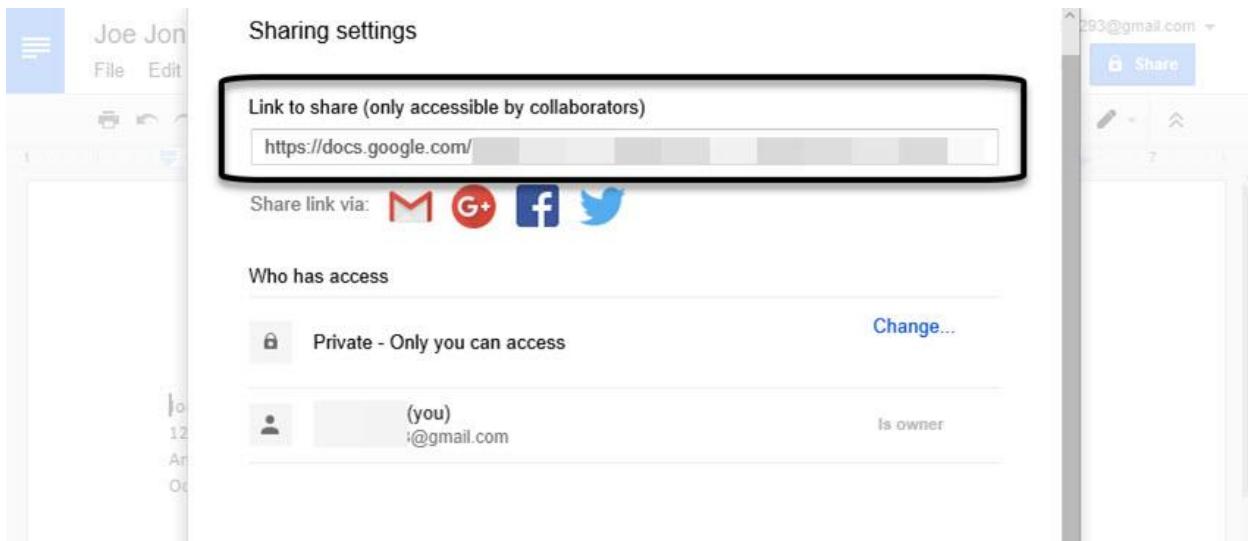
Untuk memungkinkan orang tertentu melihat dokumennya, pilih **Save**, yang selanjutnya akan mengembalikan Anda ke menu **Sharing settings**. Ketik alamat email orang-orang yang akan memiliki akses terhadap dokumen di bagian bawah menu, pada **Invite people**.



Dari menu **Sharing settings** di Google Docs, masukkan saja alamat-alamat email orang yang butuh akses terhadap dokumen Anda.

2. Anyone With the Link

Untuk memberikan akses kepada sekelompok besar orang yang ditentukan, pilih **Anyone with the link** dari opsi Visibility. Pilih **Save** untuk kembali ke menu **Sharing settings**. Dari menu **Sharing settings**, Anda bisa menyorot tautan ke dokumen dan membagikannya melalui email, media sosial, atau pesan teks.



Dari menu **Sharing settings**, sorotlah tautan yang akan dibagikan. Copy dan paste tautan tersebut untuk didistribusikan kepada siapa saja yang butuh akses terhadap dokumen yang dimaksud.

3. Public on the Web

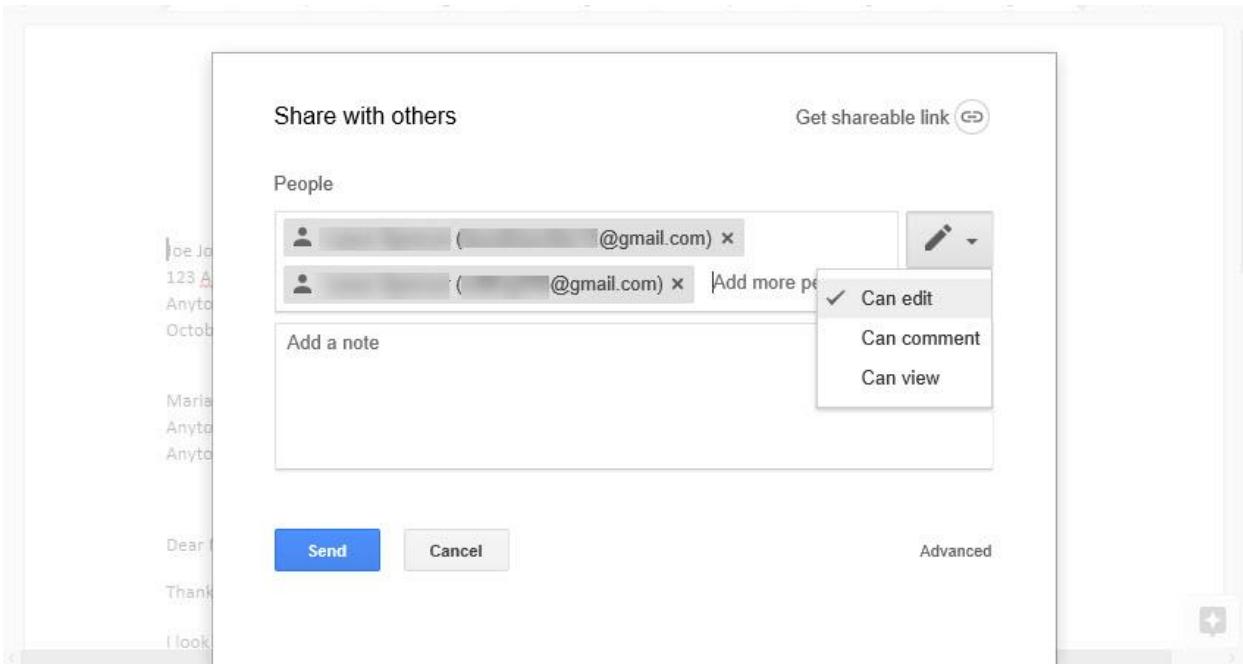
Ingin tahu cara membagikan dokumen di Google Docs kepada khalayak umum di web? Mudah saja di Google Docs. Untuk memberikan akses kepada siapa saja untuk menempelkan suatu dokumen Google Docs di halaman webnya, pilih **Public on the web** dari opsi Visibility. Pilih **Save** untuk kembali ke menu Sharing settings. Dari menu Sharing settings, Anda bisa menyorot tautan ke dokumen dan membagikannya di media sosial untuk memberi akses pada siap saja ke dokumen Anda.

2. Collaborator Options

Di bagian ini, pelajari bagaimana membagikan dokumen Google Docs dengan lebih banyak kontrol dengan cara mengatur para kolaboratornya. Ada empat **Collaborator Options** yang memberikan tingkat akses berbeda terhadap dokumen Anda bagi orang-orang tertentu. Pemilik dokumen bisa mengotorisasi orang lain untuk melihat, berkomentar, mengedit, atau menjadi pemilik dokumen.

Untuk melihat opsi **Collaborator**, pilih tombol **Share** di kanan atas dokumen Anda. Pilih **Advanced**. Di menu **Sharing settings**, undang orang-orang atau kelompok orang tertentu dengan menggunakan alamat email atau grup kontak email.

Ketika memasukkan alamat-alamat email, opsi **Collaborator** akan nampak. Jika Anda mengklik panah arah ke bawah di dekat ikon pensil di sebelah tiap nama. Di sini, Anda bisa memilih opsi: **Can edit, can comment**, atau **can view**.



Begitu Anda mengklik opsi tanda panah di sebelah pensil, Anda bisa mengubah opsi Collaborator dari menu drop-down.

Berikut adalah penjelasan tiap opsi:

1. Can Edit

Opsi Collaborator ini memungkinkan tiap orang mengedit dokumennya. Para kolaborator bisa bekerja pada suatu dokumen secara simultan pada waktu yang sama, sehingga tidak butuh untuk saling mengemailkan perubahan ke sana ke mari.

Tiap editor yang mengerjakan dokumen akan terlihat sebagai ikon di kanan atas layar, di sebelah tombol **Comments**. Tiap kolaborator akan memiliki warna yang berbeda di sekeliling ikonnya, dan pada bagian dokumen yang sedang mereka edit akan tampak kursor yang warnanya sama dengan ikon tadi.

2. Can Comment

Dengan opsi Collaborator ini, setiap orang yang kepadanya dibagikan dokumen Google Docs, akan bisa melihat dan mengomentari dokumen, tetapi tidak akan mampu mengubah dokumennya dengan cara apapun. Opsi ini bermanfaat untuk mendapatkan umpan balik bagi suatu proyek, tapi tidak dikehendaki adanya perubahan informasi tanpa sepengertuan Anda.

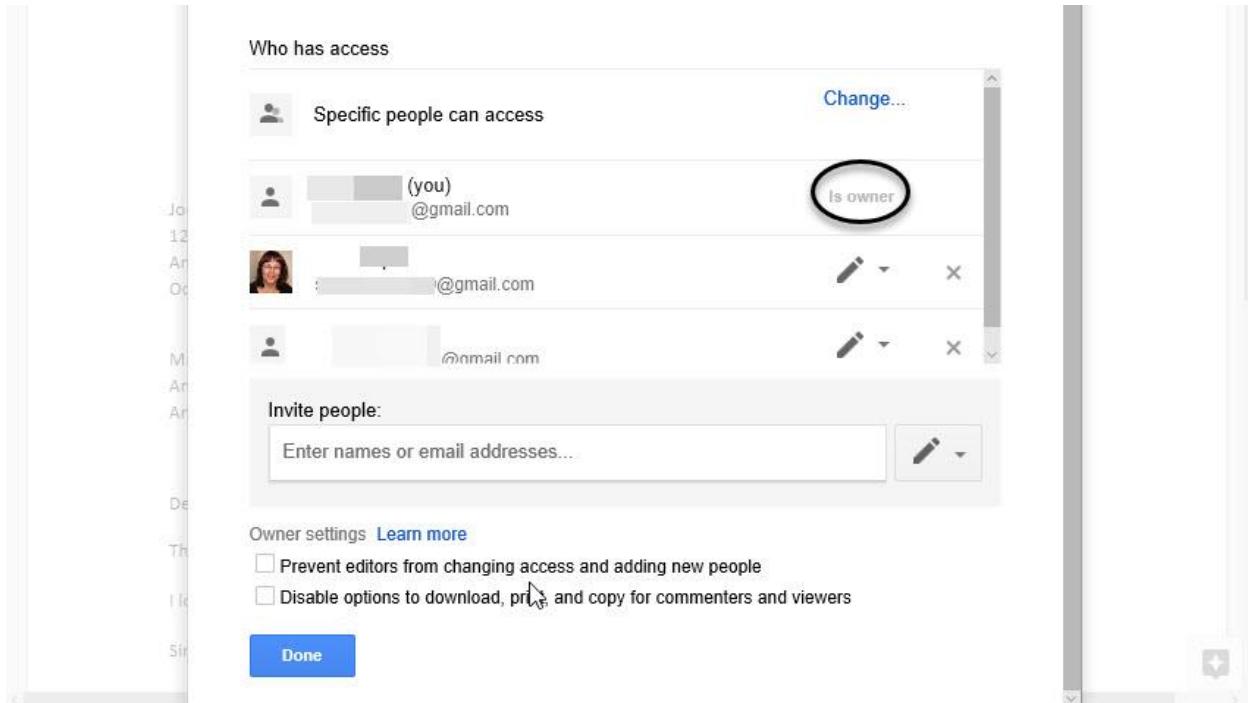
Untuk berkomentar pada suatu dokumen, sorot teks yang akan dikomentari dan klik kanan untuk memilih **Comment**. Anda juga bisa menggunakan jalan pintas keyboard **Ctrl-Alt-M**, atau tombol **Comments** di kanan atas layar, di sebelah **Share**. Pemilik dokumen akan melihat bagian yang disorot dan kotak komentar di sebelah kanan layar mereka.

3. Can View

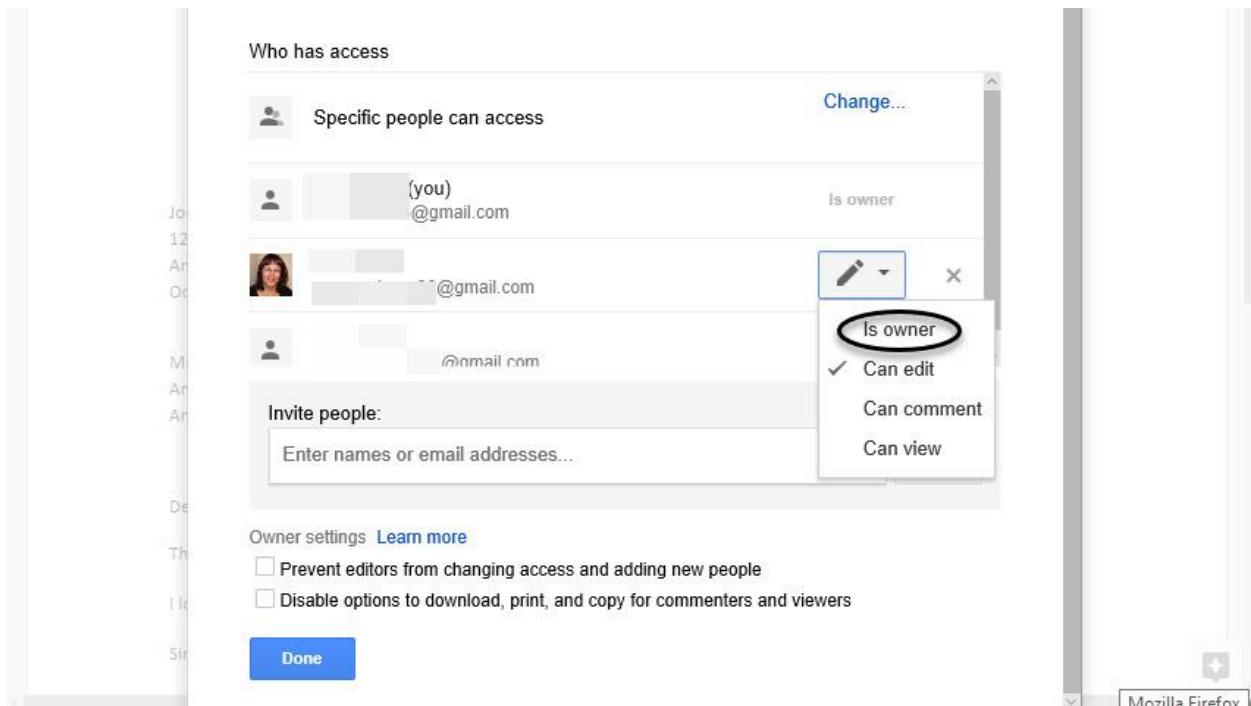
Untuk memungkinkan orang lain bisa melihat saja suatu dokumen, pilih opsi **Can view**. Opsi ini berguna khususnya untuk membagikan templat yang akan digunakan oleh orang lain. Dengan mengatur opsi Collaborator menjadi **Can view**, mereka akan dipaksa untuk membuat salinan

(Make a copy) suatu dokumen agar bisa mengubahnya, ini memastikan tak seorangpun bisa mengubah templat secara langsung.

Opsi Colaborator keempat, **Owner**, hanya tersedia jika Anda sudah membagikan dokumen di Google Docs dengan orang lain. Setelah dokumennya dibagikan, nama pemilik akan muncul di daftar **Who has access** di menu **Sharing settings**.



Anda bisa mentransfer penetapan Is owner kepada kolaborator lainnya di Google Docs. Opsi owner terakhir ini memungkinkan Anda mentransfer kepemilikan dokumen kepada orang lain. Ini muncul di menu drop-down di sebelah nama kolaborator jika Anda la sang pemilik dokumen.



Pilih menu **Is owner** untuk mengubah kepemilikan dokumen di Google Docs.

Secara default pemilik dokumen adalah yang membuat dokumen tersebut. Jika kepemilikan dokumen dipindahkan pada orang lain, ada beberapa hal yang tidak bisa lagi dilakukan oleh pemilik dokumen, yaitu:

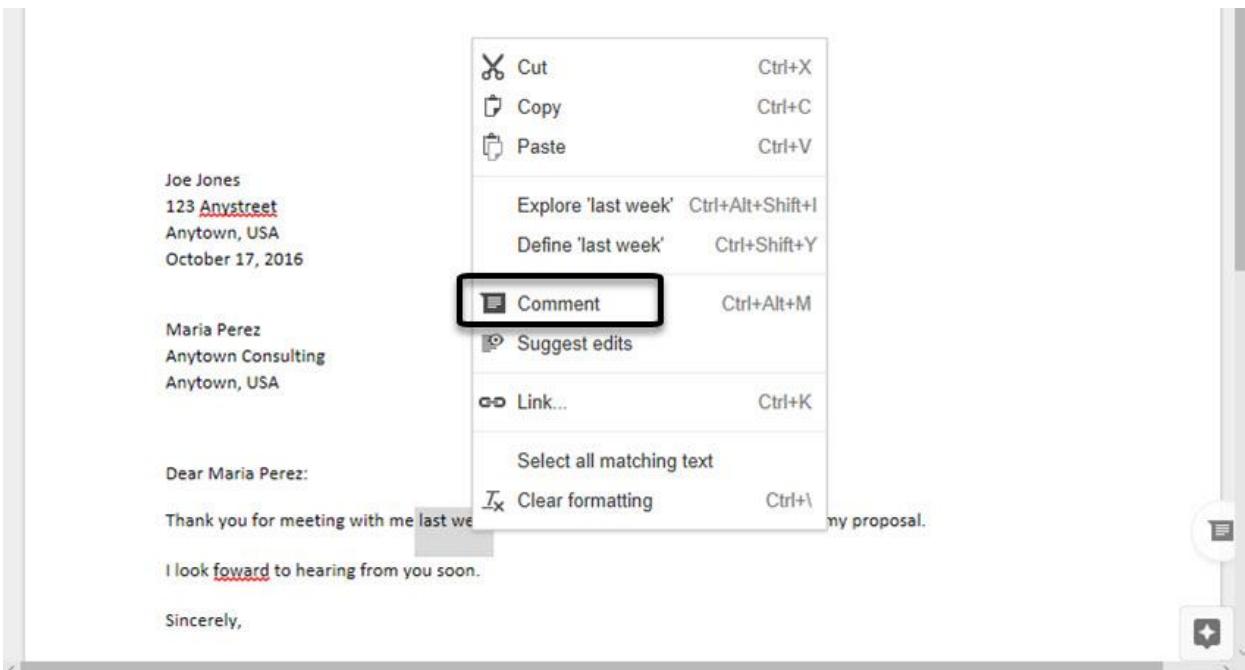
- menghapus kolaborator
- membagikan dengan orang lain sebanyak yang Anda suka
- mengubah opsi visibilitas
- memungkinkan kolaborator mengubah hak-hak akses tertentu bagi orang lain
- secara permanen menghapus sesuatu dari Google Drive Setelah dihapus, tidak ada yang bisa mengaksesnya, termasuk kepada mereka yang dokumennya dibagikan kepadanya.

3. Menambahkan Komentar

Kolaborator yang diatur **Can comment** atau **Can edit** bisa meninggalkan komentar pada dokumen yang dibagikan di Google docs. Komentar yang ada mungkin memerintahkan pemilik dokumen untuk mengubah kesalahan ejaan sederhana, atau bisa membuka dialog tentang bagian tertentu suatu dokumen.

Langkah 1. Menambahkan Komentar

Untuk menambahkan komentar pada suatu dokumen Google Docs yang dibagikan, sorot bagian dokumen yang Anda ingin komentari. Bisa jadi yang disorot adalah kata tertentu, kalimat tertentu, atau seluruh paragraf. Setelah menyorot bagian yang dimaksud, klik kanan dan pilih **Comment**.



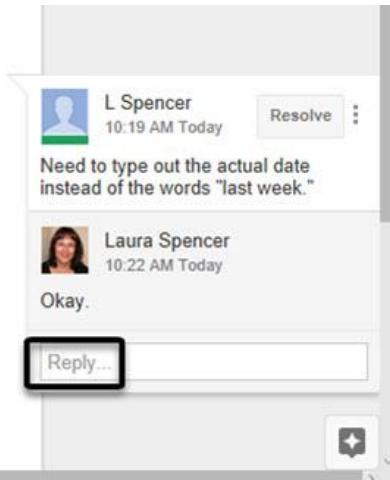
Begini Anda selesai menyorot teks yang akan dikomentari, klik kanan dan pilih **Comment**. Suatu kotak teks dengan nama Anda akan muncul di sisi kanan jauh layar sebelum Anda memasukkan komentar.



Suatu kotak teks seperti contoh di atas akan muncul di sisi kanan jauh dokumen tersebut. Ketik di dalam kotak dan klik **Comment**.

Jika ada yang menulis komentar pada dokumen Anda, Anda bisa memulai dialog dengan membalas komentar mereka. Di bawah komentar, akan muncul kotak teks bertuliskan **Reply**. Di sini, Anda bisa mengetikkan respon Anda.

ng with me **last week**. I appreciate your willingness to listen to my proposal.
ring from you soon.

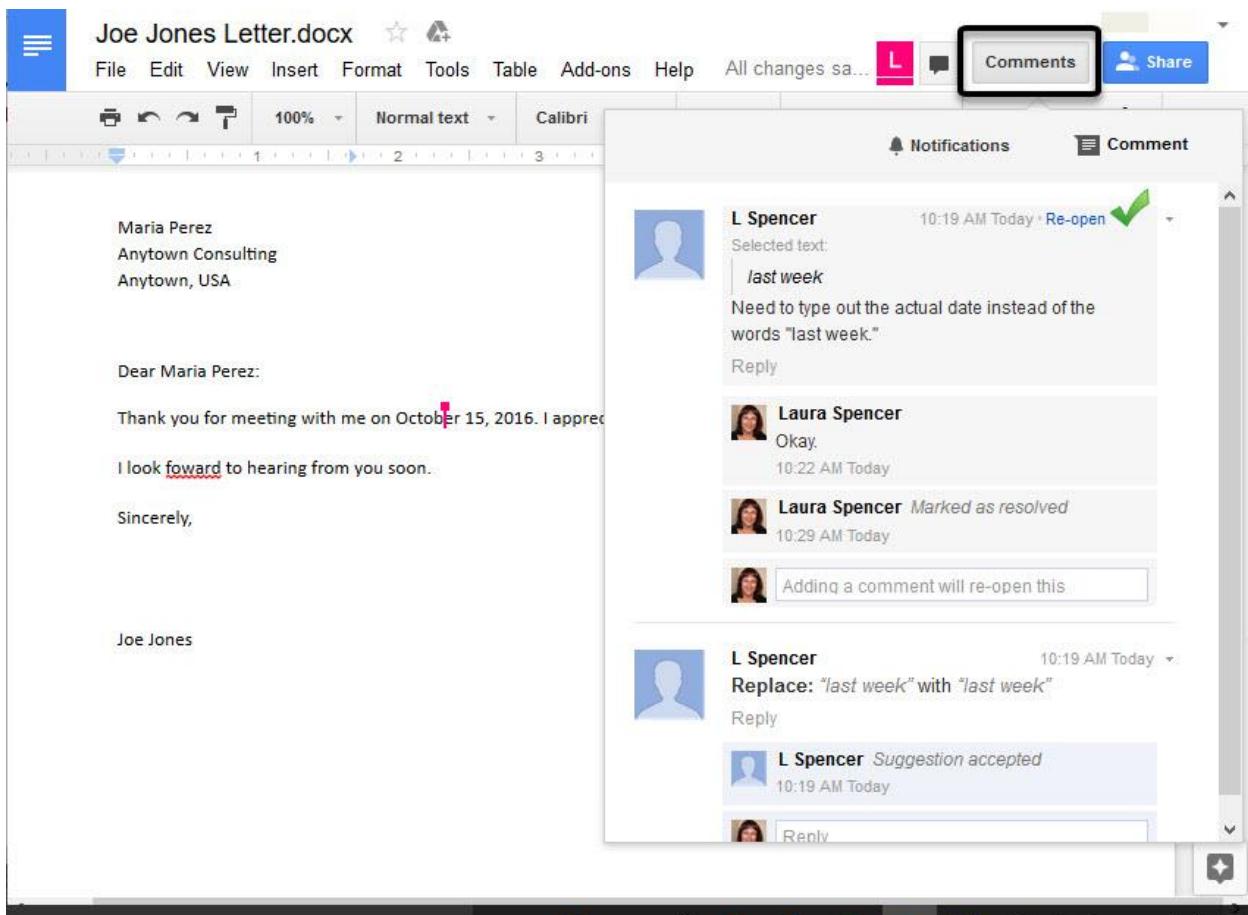


Untuk membalas komentar, klik kotak komentarnya dan ketikkan di dalam kotak teks di bagian bawah.

Jika komentarnya adalah tentang perbaikan sederhana yang tidak butuh respon, Anda bisa melakukan penyesuaian yang diperlukan terhadap dokumen dan menunjukkan bahwa komentar tersebut telah dituntaskan dengan mengklik tombol **Resolve** di sebelah komentar. Dengan menuntaskan suatu komentar, maka komentar tersebut akan disembunyikan. Untuk membuka lagi komentar tertentu, klik tombol **Comment** di sisi kanan atas dokumen.

Langkah 2. Membuka Komentar yang Telah Dituntaskan (Resolved Comment)

Meskipun suatu komentar telah dituntaskan, Anda masih bisa melihatnya lagi. Klik tombol **Comments** di bagian kanan atas dokumen Anda.



Gunakan tautan **Re-open** untuk membuka kembali komentar yang telah dituntaskan di dokumen Google Docs yang dibagikan.

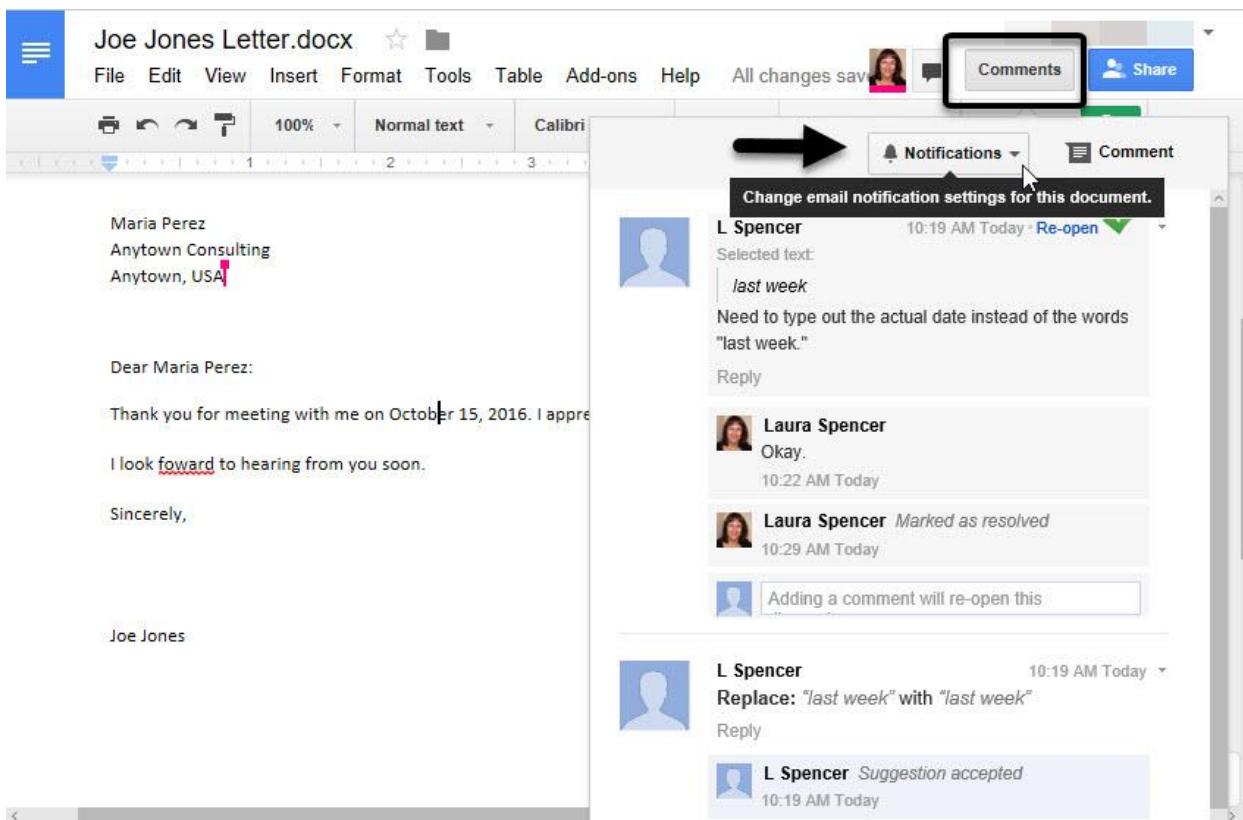
Untuk membuka lagi komentar yang telah dituntaskan, klik tautan biru **reopen** di sebelah dokumen. Komentar akan ditambahkan kembali ke dokumen.

4. Memberikan Notifikasi pada Kolaborator

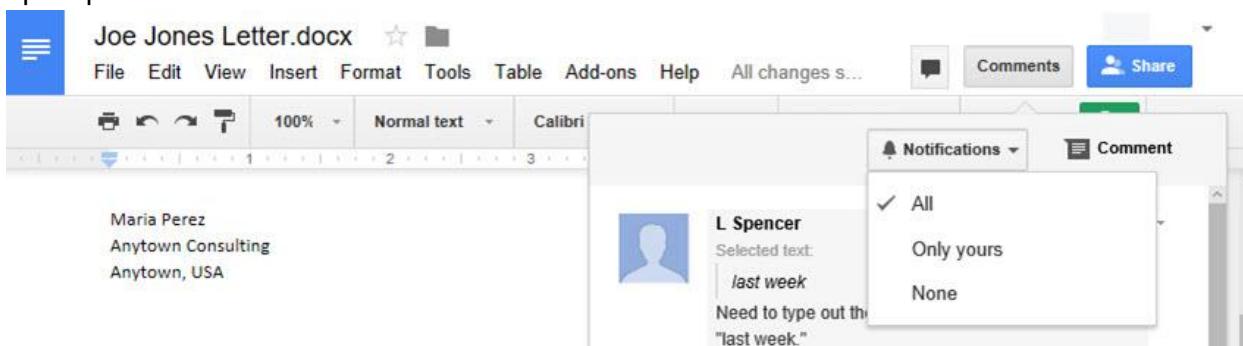
Di bagian ini, kami sampaikan juga cara membagikan dokumen Google Docs dengan notifikasi yang sesuai. Anda bisa secara otomatis memberikan notifikasi pada para kolaborator melalui email ketika ada perubahan pada dokumen yang dibagikan. Fitur ini bermanfaat jika Anda mengerjakan suatu dokumen dan butuh kolaborator Anda tahu apa saja yang Anda edit.

Langkah 1. Aktifkan Notifikasi

Buka dokumen Anda. Pastikan dokumen Anda dibagikan minimal dengan satu orang lainnya. Klik tombol **Comments** di sisi kanan atas dokumen. Klik tombol **Notifications** di bawahnya.



Klik tombol **Notifications** untuk menampilkan menu drop-down notifikasi. Setelah Anda mengklik tombol **Notifications**, akan muncul menu drop-down yang menampilkan opsi-opsi notifikasi Anda.



Pilih opsi notifikasi Anda dari menu drop-down tersebut. Anda bisa mengubah opsi notifikasi dari menu drop-down di dokumen Google Docs yang dibagikan, dengan pengaturan ini:

1. All

Untuk mengirimkan notifikasi email kepada semua orang yang kepadanya dibagikan dokumen yang dimaksud, pilih opsi **All**. Ini adalah pilihan yang bagus apabila dokumennya ditulis bersama oleh suatu tim.

2. Only Yours

Untuk mengirim email notifikasi hanya kepada diri Anda sendiri ketika ada komentar masuk atau dokumennya dimodifikasi, pilih opsi **Only yours**. Ini adalah pilihan yang bagus terutama jika Anda adalah penulis utama dan butuh tinjauan umpan balik, tapi peninjau tidak harus saling melihat komentar yang lain.

3. None

Untuk mematikan notifikasi email pilih opsi **None**. Ini adalah pilihan yang bagus jika tidak penting bagi Anda melihat dan mengedit komentar secepatnya.

Langkah 2. Melihat Notifikasi Email

Jika Anda menghidupkan notifikasi email, Anda akan menerima email yang meringkas notifikasi begitu notifikasinya dibuat. Buka kotak masuk Anda untuk melihat notifikasinya. Lihat contoh email berikut:

The screenshot shows a Gmail inbox interface. On the left, there's a sidebar with 'COMPOSE' at the top, followed by a list of labels: 'Inbox' (which is selected and highlighted in red), 'Starred', 'Sent Mail', 'Drafts (2)', 'To Do', and 'More labels'. Below the labels is a search bar with a magnifying glass icon and a 'L' icon. In the center, there's a message card for a reply from 'Laura Spencer' to a comment on 'Joe Jones Letter.docx'. The message content is: 'Need to type out the actual date instead of the words "last week."'. Below this, another message from 'Laura Spencer' says 'Okay.' and 'Marked as resolved'. At the bottom of the message card are 'Reply' and 'Open' buttons. At the very bottom of the inbox screen, there are icons for 'People', 'Mails', and 'Calls'. The top of the window has a toolbar with standard email controls like back, forward, and delete, along with a 'More' dropdown and a settings gear icon. The status bar at the top right says '1 of 1'.

Bukalah kotak masuk email Anda untuk melihat notifikasi email dari dokumen Google Docs yang Anda bagikan.

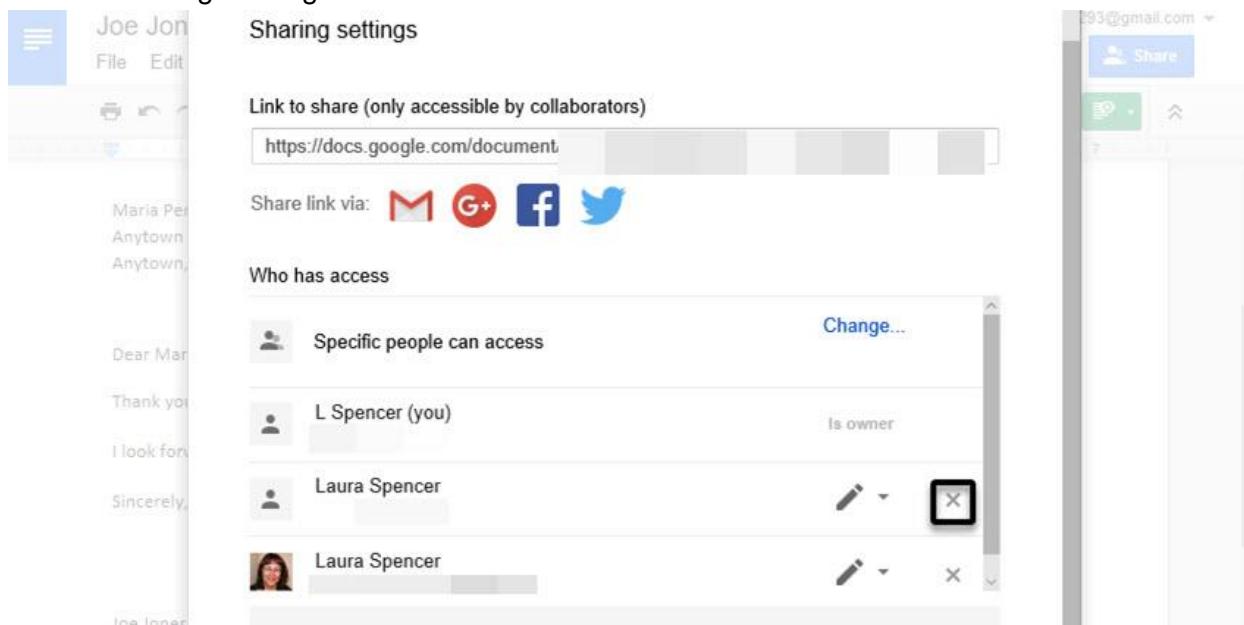
5. Menghapus Kolaborator

Terkadang Anda mungkin butuh untuk menghapus seorang kolaborator dari suatu dokumen. Ini bisa terjadi apabila ada anggota tim yang meninggalkan tim dan tidak lagi butuh akses dokumen. Berikut adalah cara menghapus kolaborator dari suatu dokumen yang dibagikan di Google Docs.

Catatan: Hanya pemilik dokumen yang bisa menambah atau menghapus kolaborator dari suatu dokumen.

Pertama, pastikan Anda mengerjakan suatu dokumen yang ada kolaboratornya.

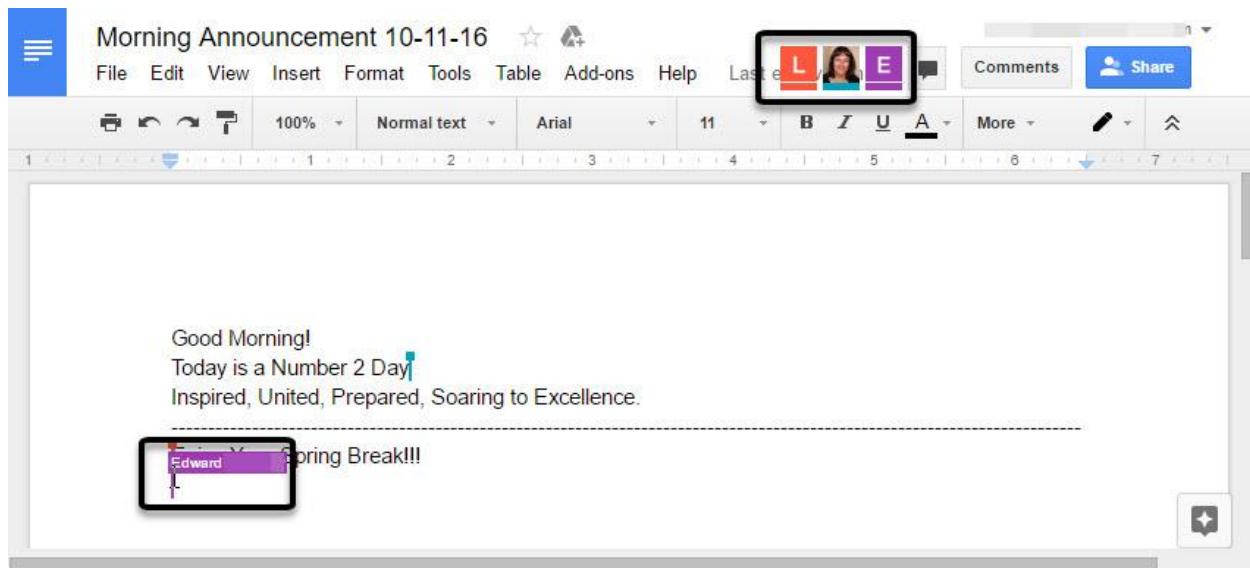
Mulai dari menu **Sharing settings** di dalam dokumen Anda. Anda akan melihat tanda **x** di sebelah masing-masing nama dalam daftar **Who has access**.



Klik tanda **x** untuk menghapus kolaborator tertentu dari dokumen yang dibagikan tersebut. Untuk menghapus salah satu kolaborator dari dokumen yang dibagikan tersebut, klik tanda **x** di sebelah nama orang tersebut. Nama orang itu akan menghilang dari daftar.

6. Contoh Kolaborasi

Mengedit di waktu yang sama di Google Docs memungkinkan Anda berkolaborasi dengan siapa saja, di mana saja, tanpa harus bingung atau ribet. Berikut ini adalah dari suatu sekolah. Mereka menggunakan Googe Docs sebagai sarana Pengumuman Paginya. Para guru bisa menambah pengumuman ke halam tersebut untuk even-even khusus, dan mereka bisa menggunakan proyektor untuk menampilkan pengumumannya di waktu bersamaan ketika kelas akan dimulai. Ini akan menghemat waktu dan gangguan yang mungkin ada. Di bagian atas halaman Anda bisa melihat guru lain yang sedang mengedit atau menampilkan pengumuman, dan bahkan bisa melihat langsung bagian dokumen yang mereka edit. Pengumuman pagi akan tampak seperti ini apabila dikerjakan bersama oleh para guru tersebut.



Di sini Anda bisa melihat seperti apa jika berbagai orang berkolaborasi pada dokumen yang sama di saat yang sama di Google Docs. Suatu ikon muncul di puncak dokumen bagi tiap orang yang mengerjakannya. Kursor di dokumen warnanya persis dengan warna ikon orang yang mengedit dokumen.

Tugas

1. Setelah anda mencoba layanan SaaS berupa google suite. Buatlah tabel perbandingan dengan layanan SaaS lain
2. Buatlah kesimpulan mengenai perbedaan antara SaaS dan Paas

Pertemuan 2 - Pengenalan Software as A services

Tujuan

4. Memahami pemanfaatan software as a service
5. Memanfaatkan software as a service untuk online collaborative
6. Dapat membangun layanan software as a service dengan model deployment private

Teori

SaaS (software as a service atau perangkat lunak berbentuk layanan) adalah suatu model penyampaian aplikasi perangkat lunak oleh suatu vendor perangkat lunak yang mengembangkan aplikasi web yang diinangi dan dioperasikan (baik secara mandiri maupun melalui pihak ketiga) untuk digunakan oleh pelanggannya melalui Internet. Pelanggan tidak mengeluarkan uang untuk memiliki perangkat lunak tersebut melainkan hanya untuk menggunakan. Pelanggan menggunakan perangkat lunak tersebut melalui antarmuka pemrograman aplikasi yang dapat diakses melalui web dan seringkali ditulis menggunakan layanan web atau REST.

Salah satu aplikasi yang memiliki karakteristik Software as a Services adalah Etherpad, dimana pengguna dapat menggunakan aplikasi secara realtime dan dapat berkolaborasi bersama pengguna lain.

Praktik

Persiapan

Seluruh proses praktik dapat dilakukan dengan menggunakan platform katacoda. Praktik ini menggunakan workspace ubuntu yang tersedia pada platform tersebut. Sebelum memulai anda harus login terlebih dahulu ke email gmail atau github, kemudian klik [ubuntu playground katacoda](#) untuk dapat memulai masuk ke sistem.

Langkah 1 - Membuat etherpad user

Pertama kita membuat user etherpad pada system dengan menggunakan perintah :

```
sudo adduser --system --home=/opt/etherpad --group etherpad
```

Langkah 2 - Instalasi Library

Untuk melakukan proses instalasi etherpad, ada beberapa kebutuhan library dan dependency.

```
sudo apt-get update  
sudo apt-get install gzip git-core curl python libssl-dev build-  
essential abiword python-software-properties
```

Langkah 3 - Instalasi Node JS

Download source code nodejs :

```
curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
```

Instalasi nodejs dengan perintah :

```
sudo apt-get install nodejs
```

Gunakan perintah berikut untuk melakukan pengecekan :

```
node -v
```

Langkah 4 - Instalasi Etherpad

Download source code Etherpad melalui git :

```
git clone git://github.com/ether/etherpad-lite.git
```

Setelah selesai masuk ke dalam direktori etherpad-lite dengan perintah :

```
cd etherpad-lite
```

Jalankan Etherpad dengan menggunakan perintah

```
bin/run.sh
```

Pengujian

Untuk menguji hasilnya dan mengakses etherpad lakukan dengan cara berikut ini:



1. Klik tombol + dan pilihlah select port to view seperti gambar diatas.
2. Setelah terbuka tab browser baru, tuliskan 9001 pada kolom isian port.
3. Setelah itu anda dapat mulai menggunakan etherpad

Tugas

1. Gambarkan model infrastruktur yang anda persiapkan untuk layanan SaaS ?
2. Buatlah layanan Software as a service dengan menggunakan owncloud

Pertemuan 3 - Implementasi Layanan Business Process as a Services.

Tujuan

1. Mahasiswa dapat memahami pemanfaatan teknologi software as a service untuk proses bisnis
2. Mahasiswa dapat merancang, membangun, dan implementasi langsung layanan software as a service yang dimanfaatkan untuk proses bisnis

Teori

Apache OFBiz. Multi-platform (Java) OFBiz (yang merupakan singkatan Terbuka untuk Bisnis) adalah paket bisnis perusahaan Apache Foundation. Yang dirilis dibawah lisensi Apache 2.0. Apache OFBiz ™ merupakan produk open source untuk otomatisasi proses perusahaan yang mencakup komponen framework dan aplikasi bisnis untuk ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), E-Bisnis / E-Commerce, SCM (Supply Chain Management) , MRP (Manufacturing Resource Planning), MMS / EAM (Maintenance Management System / Enterprise Asset Management), POS (Point Of Sale).

Apache OFBiz memberikan landasan dan titik awal untuk solusi perusahaan yang handal, aman dan scalable. Menggunakannya out-of-the-box, menyesuaikan atau menggunakannya sebagai kerangka kerja untuk melaksanakan kebutuhan bisnis yang paling menantang. Apache OFBiz dilisensikan di bawah Lisensi Apache Versi 2.0 . Proyek Apache OFBiz merupakan bagian dari The Apache Software Foundation .

Apache OFBiz adalah suatu framework common data model business process atau aplikasi ERP dan dibangun dengan Pemrogaman Java. Semua aplikasi dibuat dengan menggunakan arsitektur yang umum, menggunakan komponen-komponen data, logic dan process. Apache OFBiz menawarkan fleksibilitas desain dan akses terhadap kode program sehingga memudahkan melakukan modifikasi. Apache OFBiz dibangun pada teknologi open source dan standar seperti Java, Java Enterprise Edition (JEE), XML dan SOAP

Apache OFBiz datang dengan berbagai fungsi yang meliputi: Accounting (perjanjian, faktur, vendor manajemen, buku besar umum), Asset Maintenance, Catalogue dan Manajemen Produk, Fasilitas dan Warehouse Management, Manufaktur, Order Processing, Inventory Management, otomatis pengisian ulang saham dll, Content Management System (CMS), Orang-orang dan Group Management, Project Management, Sales Force Automation, Kerja Manajemen Usaha, Electronic Point Of Sale (EPOS), Ecommerce, Sumber daya manusia (SDM)

Praktik

Persiapan

Seluruh proses praktik dapat dilakukan dengan menggunakan platform katacoda. Praktik ini menggunakan workspace ubuntu yang tersedia pada platform tersebut. Sebelum memulai anda harus login terlebih dahulu ke email gmail atau github, kemudian klik [ubuntu playground katacoda](#) untuk dapat memulai masuk ke sistem

Berikut ini adalah langkah untuk instalasi **Apache OFBiz 16.0**.

Langkah 1 - Install JAVA

Java dibutuhkan untuk dapat melakukan instalasi apacha ofbiz, jalankan perintah berikut pada terminal anda:-

```
$ sudo add-apt-repository ppa:webupd8team/java  
$ sudo apt-get update
```

Install Java :-

```
$ sudo apt-get install oracle-java8-installer
```

Setelah instalasi, lakukan pengecekan pada sistem.

```
gurinder@gurinder:~/apache-ofbiz-16.11.03$ java -version  
java version "1.8.0_131"  
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)  
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
```

Untuk memeriksa versi yang terinstal dan beralih ke versi Java-8-oracle untuk OFBiz jalankan perintah berikut:

```
$ sudo update-alternatives --config java
```

Perintah ini akan mendaftar semua versi yang terinstal dengan tanda aktif saat ini dan menyediakan dialog untuk beralih. Ada 3 pilihan untuk alternatif java.

```
gurinder@gurinder:~/apache-ofbiz-16.11.03$ sudo update-alternatives --config java  
[sudo] password for gurinder:  
There are 3 choices for the alternative java (providing /usr/bin/java).  


| Selection | Path                                           | Priority | Status      |
|-----------|------------------------------------------------|----------|-------------|
| 0         | /usr/lib/jvm/java-7-oracle/jre/bin/java        | 1082     | auto mode   |
| 1         | /usr/lib/jvm/java-7-oracle/jre/bin/java        | 1082     | manual mode |
| 2         | /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java | 1081     | manual mode |
| * 3       | /usr/lib/jvm/java-8-oracle/jre/bin/java        | 1081     | manual mode |



Press <enter> to keep the current choice[*], or type selection number: █


```

Langkah 2 - Edit .bashrc file

Lakukan proses editing pada JAVA_HOME yang terletak pada file .bashrc.

```
$ vi .bashrc  
$ export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

Setelah selesai mengedit anda dapat merefresh dengan perintah

```
$ source ~/.bashrc
```

Langkah 3 - instalasi OFBiz

Kami menggunakan 16.11.03 (versi stabil), Unduh Apache OFBiz di [sini](#). Catatan: Pilih versi stabil di menu 'releases'.

Langkah 4 - Quick START

Untuk menginstal dan menyalakan OFBiz dengan cepat, ikuti petunjuk di bawah ini:

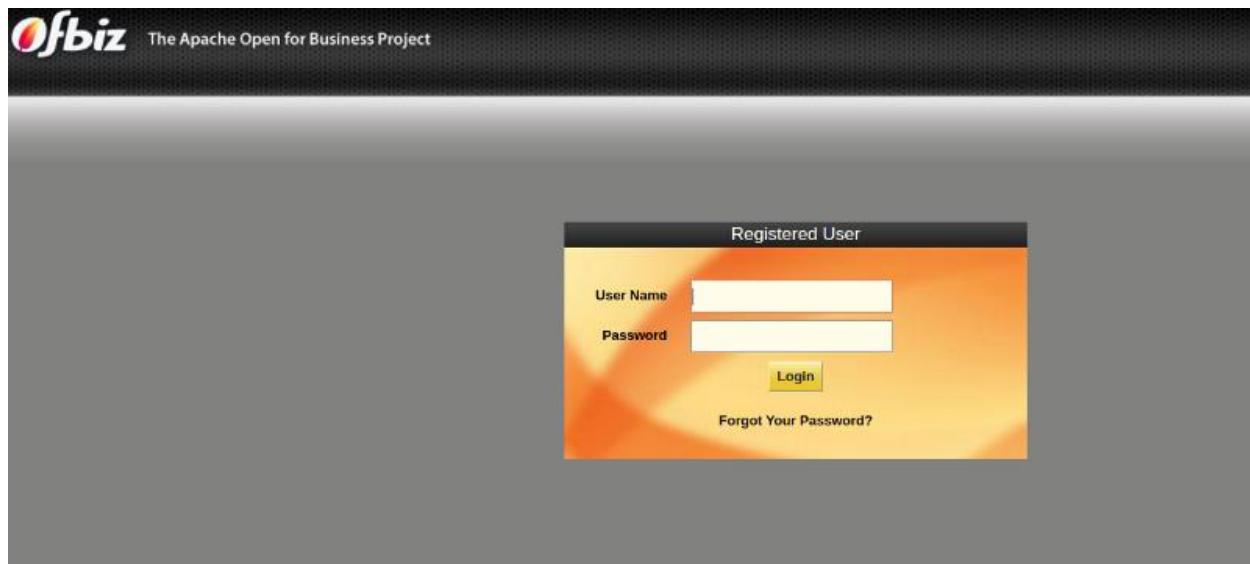
```
$ ./gradlew cleanAll loadDefault
```

Langkah 5 - Start OFBiz

```
$ ./gradlew ofbiz
```

Untuk melihat hasil instalasi dapat merujuk pada link berikut:

<https://localhost:8443/ordermgr/control/main>



USERNAME:- admin

PASSWORD:- ofbiz

Tugas

1. Bersama kelompok, Setelah masuk ke dalam sistem buatlah model proses bisnis untuk penjualan perangkat komputer. Referensi cara pembuatan dapat merujuk link [berikut ini](#)

Pertemuan 4 : Pengenalan Docker

Tujuan

1. Mahasiswa dapat membandingkan perbedaan antara virtualisasi dan teknologi container
2. Mahasiswa dapat mengenal dan mengelola docker container
3. Mahasiswa dapat memahami implementasi docker container

Teori

Container kemudian datang dan membawa beberapa perubahan. Dengan container, sebuah program ‘diikat’ beserta library-nya, file konfigurasi, dan seluruh hal yang dibutuhkannya. Perbedaan yang sangat terlihat dibandingkan dengan virtualisasi adalah container memiliki ukuran file yang jauh lebih kecil karena tidak perlu menyiapkan sistem operasi secara penuh. Dalam hal ini, pengembang biasa menyebutnya sebagai ‘lightweight’ platform. Aplikasi yang berjalan menggunakan container pun jauh lebih cepat dan lebih efisien.

Docker adalah salah satu platform yang dibangun berdasarkan teknologi container. Docker merupakan sebuah project open-source yang menyediakan platform terbuka untuk developer maupun sysadmin untuk dapat membangun, mengemas, dan menjalankan aplikasi dimanapun sebagai sebuah wadah (container) yang ringan. Dengan sangat populernya docker, sebagian orang sering menganggap docker adalah sebutan lain untuk container.

Praktik

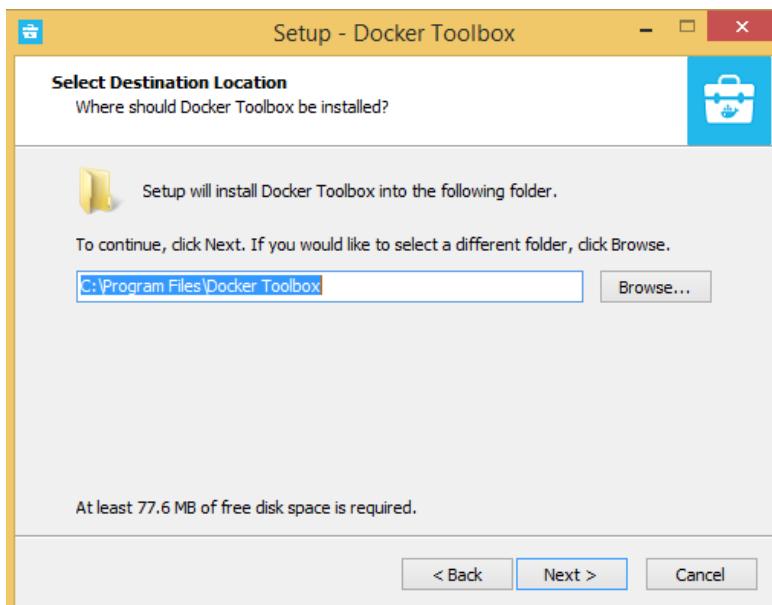
Persiapan

Download docker toolbox di [website docker](#)

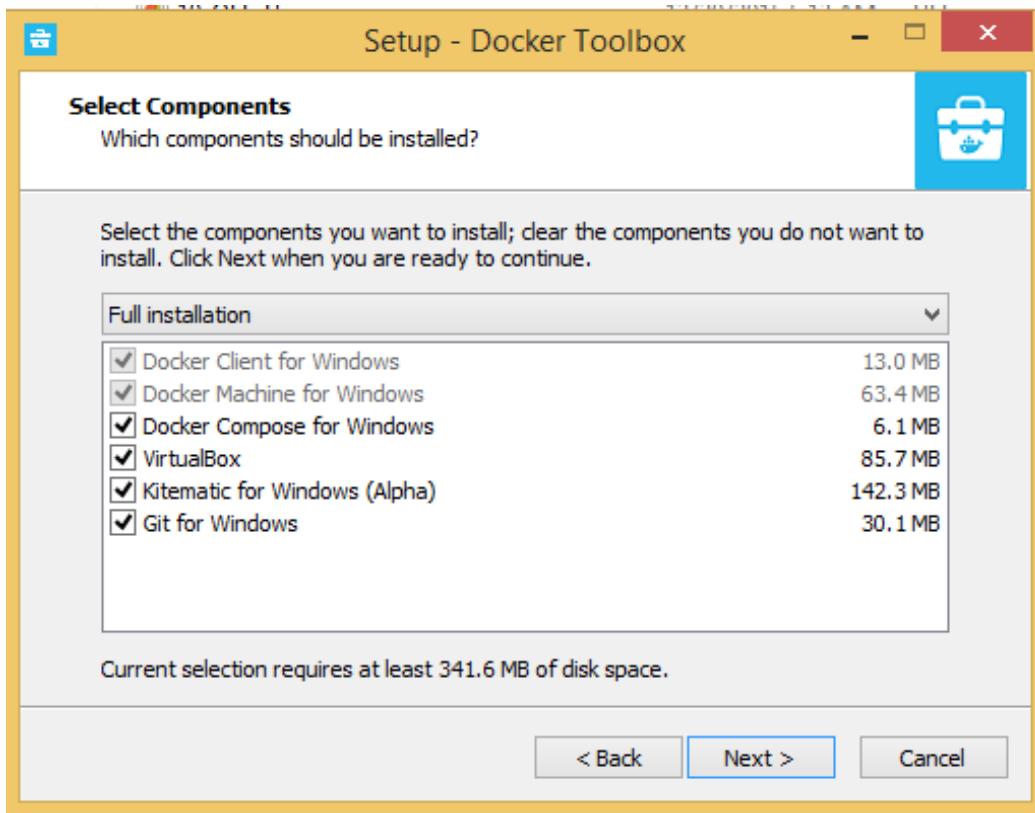
Instalasi Windows



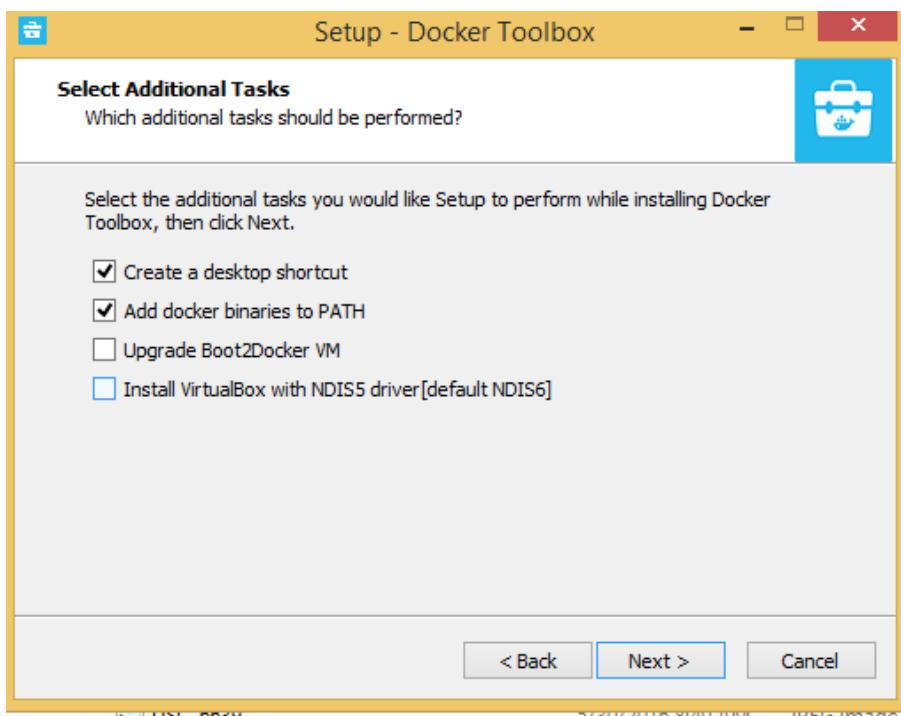
Tampilan awal proses install, pilih “Next” untuk melanjutkan instalasi Docket Toolbox



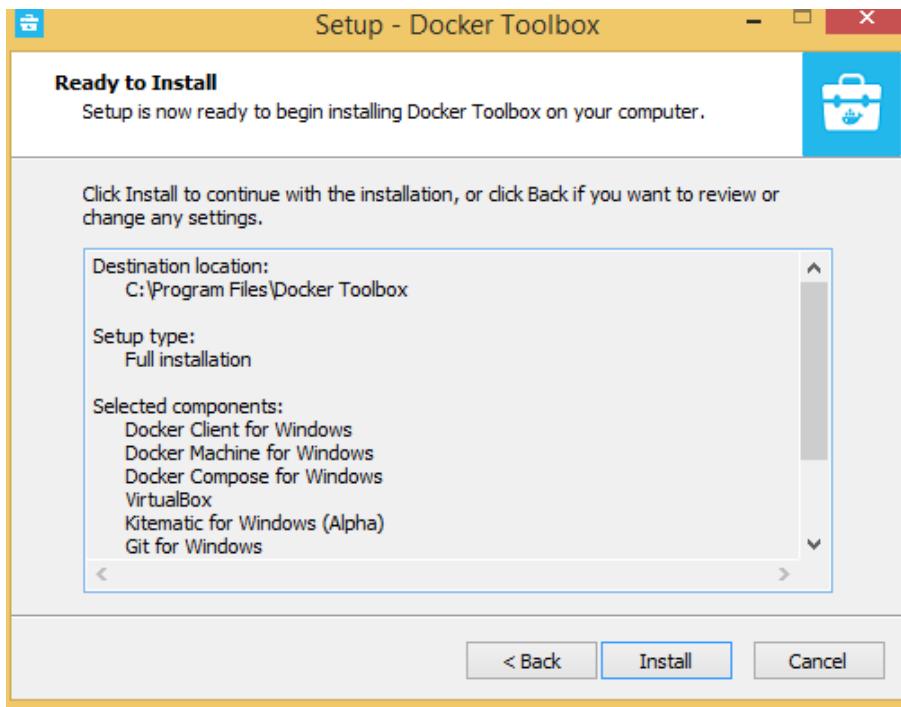
Mengatur lokasi dimana Docker Toolbk akan diinstall, Default ada di C:\Program Files. Jika tidak ingin untuk merubah lokasi installasinya, biarkan default.



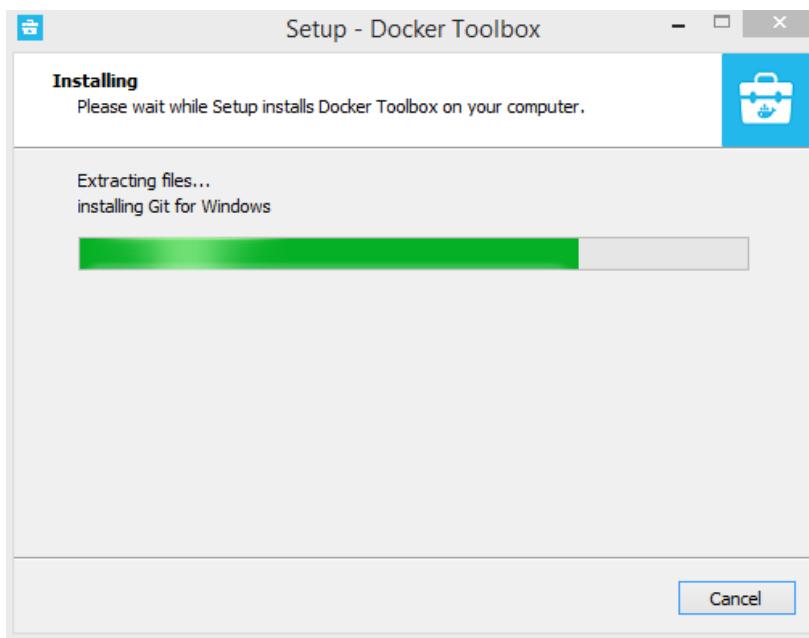
Komponen-komponen diatas adalah komponen untuk menjalankan Docker Toolbox. Komponen yang akan di install dapat dipilih apa saja yang akan diinstall. Rekomended untuk menginstall semua komponen (jika komponen belum ada di Komputer) .



Kemudian pilih centang Create a desktop shortcut, add docker binaries, dan upgrade boot2docker.



Setelah seluruh proses anda lewati, selanjutnya adalah proses instalasi docker-toolbox



name	date modified	type	size
Docker Quickstart Terminal	3/15/2017 7:13 AM	Shortcut	
Kitematic (Alpha)	3/15/2017 7:13 AM Location: bash (C:\Program Files\Git\bin)	Shortcut	

Setelah proses instalasi selesai, anda akan menemukan shortcut docker quickstart terminal dan Kitematic. Buka Dockter terminal dengan klik 2x shortcut Docker Quickstart di desktop atau windows explorer anda.

A screenshot of a terminal window titled "Docker Quickstart Terminal". The window shows a command-line interface with the following text:

```
Running pre-create checks...
<default> Default Boot2Docker ISO is out-of-date, downloading the latest release
<default> Latest release for github.com/boot2docker/boot2docker is v17.03.0-ce
<default> Downloading C:\Users\nugroho\.docker\machine\cache\boot2docker.iso from https://github.com/boot2docker/boot2docker/releases/download/v17.03.0-ce/boot2docker.iso...
```

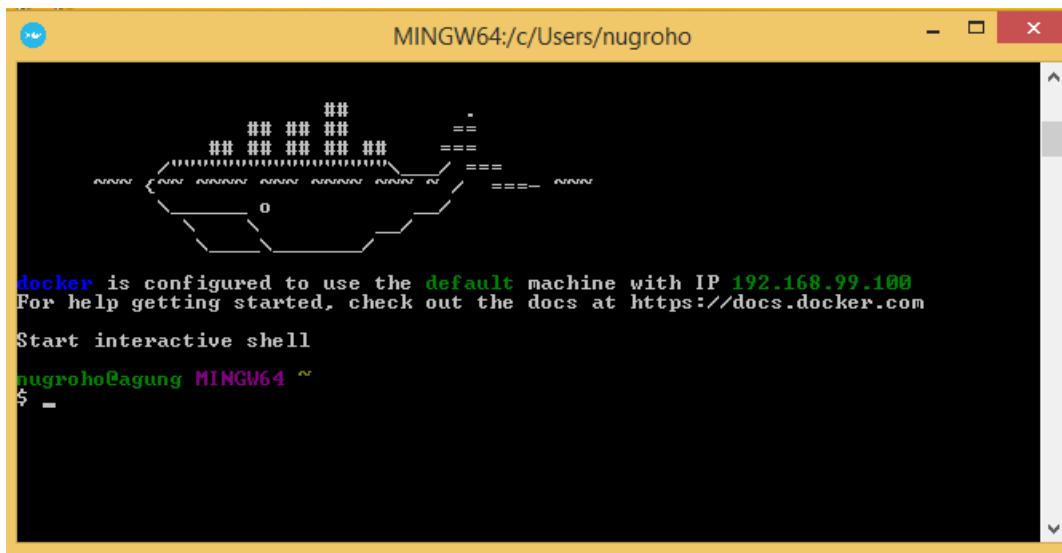
Pada proses ini, secara otomatis akan mendownload image boot2docker.iso yang terletak di [github.com](https://github.com/boot2docker/boot2docker/releases/download/v17.03.0-ce/boot2docker.iso). Dalam proses ini, jika ada keterbatasan koneksi dapat dilakukan bypass dengan

menutup jendela docker quickstart tersebut dan copy manual file image. Download manual di <https://github.com/boot2docker/boot2docker/releases/> , kemudian replace file boot2docker.iso di direktori c:\users\nama-user\docker\machine\cache

Setelah proses ini selesai, jalankan kembali docker quickstart, anda akan mendapatkan tampilan seperti gambar dibawah.



Setelah seluruh proses verifikasi selesai, anda sudah dapat menggunakan docker.



Menjalankan Container

Dengan Docker, semua container berawal dari docker image. Docker image merupakan sistem operasi yang didalamnya terdapat library / runtime / aplikasi tertentu. Image ini memiliki semua yang dibutuhkan untuk menjalankan proses sehingga host tidak perlu melakukan konfigurasi atau dependensi (ketergantungan terhadap packet atau library).

Untuk menjalankan container, dapat menggunakan Docker image yang buat sendiri (Build) atau menggunakan Docker image yang telah disediakan oleh Docker dan komunitas. Ketersediaan image dapat anda lihat [registry.hub.docker.com](https://hub.docker.com) atau dapat menggunakan perintah :

```
docker search <nama-image>
```

contoh :

```
docker search alpine
```

NAME	DESCRIPTION
STARS	OFFICIAL AUTOMATED
alpine	A minimal Docker image based on Alpine Lin...
1981	[OK]
anapsix/alpine-java	Oracle Java 8 (and 7) with GLIBC 2.23 over...
194	[OK]
frolvlad/alpine-glibc	Alpine Docker image with glibc (~12MB)
67	[OK]
container4armhf/armhf-alpine	Automatically built base images of Alpine ...
53	[OK]

Secara detail anda dapat melihat nama image, deskripsi singkat, stars, official dan automated. Stars menunjukkan popularitas dari image tersebut, sementara official adalah versi stable yang dikeluarkan langsung oleh pembuat / pengembang aplikasi / sistem operasi. Direkomendasikan untuk memilih Docker image berdasarkan kedua kriteria ini.

Selanjutnya, jalankan container secara background menggunakan image yang tersedia atau yang telah anda pilih setelah proses searching image. Gunakan perintah `docker run <options>`

`<image-name>` untuk menjalankan container. Secara default, Docker akan menjalankan perintah menggunakan foreground mode, sehingga untuk menjalankan di mode background, anda perlu menambahkan opsi `-d`.

Pada langkah ini, anda tidak memiliki Docker image di komputer anda, sehingga akan mendownload langsung dari Docker registry. Jika anda menjalankan container untuk kedua kalinya, maka local image yang akan digunakan.

Untuk melihat Docker images di host, anda dapat gunakan perintah :

```
docker images
```

Untuk menjalankan container gunakan perintah :

```
docker run -d nginx:alpine
```

Tips : Agar mudah diingat, anda dapat menggunakan opsi `--name` sehingga container tersebut mudah untuk dikenali.

Dalam proses ini akan menjalankan image alpine sebagai container, karena Docker image alpine belum terdapat di local.

```
$ docker run -d nginx:alpine
time="2017-03-16T13:07:42+07:00" level=info msg="Unable to use system
certificat
e pool: crypto/x509: system root pool is not available on Windows"
Unable to find image 'nginx:alpine' locally
alpine: Pulling from library/nginx
709515475419: Pull complete
5142c3ee45c6: Pull complete
6393d8407bbe: Pull complete
0816ea01673a: Pull complete
Digest:
sha256:aa0daf2b17c370a1da371a767110a43b390a9db90b90d2d1b07862dc81754d6
1
Status: Downloaded newer image for nginx:alpine
dc6228dbc73b49472a50ec4c7da4a60562c0d32059f6056d31d6c3c962eb3531
```

Melihat proses docker container

Docker ps dapat digunakan untuk melihat listing proses container yang berjalan, contoh implementasi :

```
$ docker ps
time="2017-03-16T13:08:59+07:00" level=info msg="Unable to use system certificat
```

```

e pool: crypto/x509: system root pool is not available on Windows"
CONTAINER ID        IMAGE               COMMAND                  CREATED          NAMES
STATUS              PORTS
dc6228dbc73b      nginx:alpine       "nginx -g 'daemon ..."   7 seconds ago
Up 6 seconds           80/tcp, 443/tcp             gifted_spence

```

Untuk melihat lebih detail terkait volume, network, dan detail lain dari Docker container yang telah berjalan, anda dapat gunakan perintah :

```
docker inspect <container id / friendly name>
```

Sementara untuk melihat catatan sistem log untuk container anda dapat menggunakan perintah :

```
docker logs <container id / friendly name>
```

Binding port

Container yang berjalan terkadang perlu untuk dapat di akses secara eksternal, untuk memudahkan Docker dapat melakukan binding port untuk host dan container. Dalam contoh dibawah ini, melakukan binding pada port <host port>:<container port> dengan menggunakan opsi -p. --name adalah opsi untuk memberikan friendly name (tag) pada container, sementara itu nginx:alpine adalah image yang digunakan untuk membuat container.

```

$ docker run -d --name nginxPort -p 8080:8080 nginx:alpine
time="2017-03-16T13:36:03+07:00" level=info msg="Unable to use system
certificat
e pool: crypto/x509: system root pool is not available on Windows"
8c06240903b6b55100e3d2dfb810f51f9c1433f28103c114120db9dd62ab4a65

```

Setelah berjalan, anda dapat melakukan pengecekan dengan melihat proses dari Docker. Pada hasil dibawah, terlihat ada 2 proses container yang sedang berjalan.

```

$ docker ps
time="2017-03-16T13:36:08+07:00" level=info msg="Unable to use system
certificat
e pool: crypto/x509: system root pool is not available on Windows"
CONTAINER ID        IMAGE               COMMAND                  CREATED          NAMES
CREATED
STATUS              PORTS
NAMES
8c06240903b6      nginx:alpine       "nginx -g 'daemon ..."   5
seconds ago
Up 5 seconds           80/tcp, 443/tcp, 0.0.0.0:8080->8080/tcp
nginxPort

```

```
dc6228dbc73b      nginx:alpine      "nginx -g 'daemon ..."   27
minutes ago
Up 27 minutes      80/tcp, 443/tcp
gifted_spence
```

Docker juga memungkinkan container dapat berjalan sekaligus pada 1 lingkungan host, sehingga container port akan di konfigurasi secara default.

```
$ docker run -d --name nginxdynamic -p 8080 nginx:alpine
time="2017-03-16T14:00:00+07:00" level=info msg="Unable to use system
certificat
e pool: crypto/x509: system root pool is not available on Windows"
a7b1ac9f0255f73066f04fa543fe9d5700e396c91041733ac11fcbec0ef6d312
```

Binding Direktori

Containers secara desain bersifat stateless. Seluruh data yang kita inginkan untuk tetap ada setelah container berhenti dijalankan harus tersimpan pada mesin host. Untuk mengatasi persoalan ini, dapat dilakukan dengan cara binding host direktori ke dalam container.

```
mkdir bindingdata
docker run -d --name nginxMapDir -v "$PWD/bindingdata":/usr/share/nginx
nginx:alpine
```

Membangun image container

Docker memungkinkan untuk membuat image sendiri dengan base Docker image yang tersedia. Untuk membuat diperlukan file Dockerfile. Berikut ini langkah untuk membuat image tersebut, dalam contoh ini, seluruh file kebutuhan diletakkan pada direktori imageku.

```
mkdir imageku
cd imageku
notepad Dockerfile
(save pada direktori imageku)
```

Rename file Dockerfile.txt menjadi Dockerfile

```
mv Dockerfile.txt Dockerfile
```

Isi file Dockerfile dengan konten ini :

```
FROM nginx:alpine
COPY index.html /usr/share/nginx/html/index.html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Penjelasan

- FROM : mendefenisikan base image Docker yang digunakan, dalam contoh ini menggunakan nginx:alpine
- COPY melakukan proses duplikasi file index.html pada direktori imageku ke dalam root direktori nginx di dalam container
- EXPOSE : Mendefenisikan port yang digunakan
- CMD : Eksekusi perintah command pada lingkungan container

Selanjutnya, buatlah file index.html untuk menampilkan “hello world”.

Setelah seluruh file tersebut tersedia, anda dapat membuat image dengan format penamaan nama-aplikasi:versi, contoh nginx-ku:v3 :

```
$ docker build -t nginx-ku:v3 .
time="2017-03-16T14:51:39+07:00" level=info msg="Unable to use system certificate pool: crypto/x509: system root pool is not available on Windows"
Sending build context to Docker daemon 3.072 kB
Step 1/4 : FROM nginx:alpine
--> 0ae090dba3ab
Step 2/4 : COPY index.html /usr/share/nginx/html/index.html
--> Using cache
--> b706c55d765e
Step 3/4 : EXPOSE 80
--> Using cache
--> 18b6f4c157fe
Step 4/4 : CMD nginx -g daemon off;
--> Running in 16fe4c7a7015
--> 3882535cfcc67
Removing intermediate container 16fe4c7a7015
Successfully built 3882535cfcc67
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.
```

Langkah terakhir, image tersebut dapat anda jalankan sebagai container.

```
$ docker run -d -p 80:80 nginx-ku:v3
time="2017-03-16T14:51:46+07:00" level=info msg="Unable to use system certificate pool: crypto/x509: system root pool is not available on Windows"
6499dd762c5a2c59e6783c6af225b940242b9e620691226fb97884cd8b0a3c
```

Untuk ujicoba, buka halaman web browser anda dan akses ke 192.168.99.100

Pertemuan 5 - Data as a Service dengan Posgresql

Tujuan

1. Mahasiswa dapat memahami model cloud data as a service
2. Mahasiswa dapat implementasi penggunaan SQL pada layanan data as a service

Teori

Sistem manajemen database relasional adalah komponen utama bagi banyak situs web dan aplikasi. Mereka memberikan sebuah cara terstruktur untuk menyimpan data dan mengakses informasi dengan cara yang terstruktur.

PostgreSQL, atau Postgres, adalah sebuah sistem manajemen database relasional (RDBMS) yang memberikan sebuah implementasi bahasa query SQL. Sistem database ini merupakan pilihan populer bagi banyak proyek baik proyek kecil maupun besar karena memiliki banyak fitur canggih seperti transaksi yang reliabel dan konkurensi tanpa read locks.

Praktik

Persiapan

Seluruh proses praktik dapat dilakukan dengan menggunakan platform [elephantsql.com](https://www.elephantsql.com). Praktik ini menggunakan workspace yang tersedia pada platform tersebut. Sebelum memulai anda harus login terlebih dahulu ke email gmail atau github, kemudian klik [login elephantsql.com](https://www.elephantsql.com) untuk dapat memulai masuk ke sistem.

Membuat instance Posgresql

1. Setelah anda login, klik **Create New Instance** sehingga muncul seperti gambar dibawah ini:

Plan Region Configure Confirm

Select a plan and name - Step 1 of 4

Name	<input type="text" value="Name to describe your instance"/>
Plan	<input type="text" value="Tiny Turtle (Free)"/>
Tags	<input type="text"/>

Tags are used to separate your instances between projects. This is primarily used in the project listing view for easier navigation and access control.

Tags allow admins to [manage team members access](#) to different groups of instances.

Plan



Tiny Turtle

See the [plan page](#) to learn about the different plans.

[Cancel](#) [Select Region](#)

2. Lalu klik **select region** - pilih provider yang anda inginkan - lalu klik **Confirm**

Plan Region Configure Confirm

Select a region and data center - Step 2 of 4

Data center	<input type="text" value="US-East-1 (Northern Virginia)"/>
-------------	--

aws

[« Back](#) [Cancel](#) [Confirm](#)

3. Setelah langkah diatas, seluruh detail akan ditampilkan, dan kemudian klik **create instance**
4. Untuk melihat detail, klik link nama instances, dan akan mendapatkan tampilan seperti gambar. Untuk masuk ke administrasi query, klik menu **browser**.

Membuat dan Menghapus Tabel

Setelah dapat terhubung ke sistem database PostgreSQL, kita dapat melakukan beberapa tugas sederhana. Pertama, kita akan membuat sebuah tabel untuk menyimpan beberapa data. Mari kita buat sebuah tabel yang mendeskripsikan daftar peralatan sebuah taman bermain.

Sekarang kita akan membuat tabel seperti berikut ini:

```
CREATE TABLE playground (
    equip_id serial PRIMARY KEY,
    type varchar (50) NOT NULL,
    color varchar (25) NOT NULL,
    location varchar(25) check (location in ('north', 'south', 'west',
    'east', 'northeast', 'southeast', 'southwest', 'northwest')),
    install_date date
);
```

Kita telah membuat tabel playground untuk menyimpan data peralatan yang dimiliki. Tabel ini dimulai dengan sebuah ID yang bertipe serial. Tipe ada ini adalah sebuah auto-incrementing integer. Kita dapat memberikan konstrain primary key untuk kolom ini yang artinya isi kolom harus unik dan tidak null. Untuk dua kolom equip_id dan install_date kita tidak memberikan panjangnya. Hal ini karena beberapa tipe kolom tidak memerlukan panjang maksimum. Kita lalu membuat agar kolom type dan color tidak dapat dikosongkan. Kita juga membuat sebuah kolom location dan sebuah konstrain untuk membatasi nilainya dari 8 opsi. Kolom terakhir adalah kolom install_date untuk memberitahu kapan kita memasangnya.

Add, Query, dan Delete Data Di Table

Setelah memiliki sebuah tabel, kita dapat memasukkan beberapa data ke dalamnya. Kita memasukkan data dengan memanggil tabel yang ingin ditambahkan, lalu memberikan nama kolom dan data untuk tiap kolom tersebut.

```
INSERT INTO playground (type, color, location, install_date) VALUES
('slide', 'blue', 'south', '2014-04-28');
INSERT INTO playground (type, color, location, install_date) VALUES
('swing', 'yellow', 'northwest', '2010-08-16');
```

Ada beberapa hal yang perlu diperhatikan saat memasukkan data untuk meminimalkan kesalahan. Pertama, nama kolom tidak boleh diberi tanda ", namun isi kolom tersebut perlu diberikan. Hal lain yang perlu dilakukan adalah memastikan untuk tidak memberikan data kolom equip_id secara manual. Hal ini karena isinya akan dibuat secara otomatis.

Kita dapat melihat data yang sudah dimasukkan dengan perintah:

```
SELECT * FROM playground;

Output equip_id | type | color | location | install_date
-----+-----+-----+-----+
 1 | slide | blue | south | 2014-04-28
 2 | swing | yellow | northwest | 2010-08-16
(2 rows)
```

Disini, kita dapat melihat bahwa kolom equip_id secara otomatis sudah ada dan data yang lain sukses ditambahkan.

Untuk menghapus data dari tabel gunakan perintah:

```
DELETE FROM playground WHERE type = 'slide';
```

Jika kita ingin melihat isi tabel kita lagi, kita akan menemukan bahwa seluncuran (slide) sudah tidak ada:

```
SELECT * FROM playground;

Output equip_id | type | color | location | install_date
-----+-----+-----+-----+
 2 | swing | yellow | northwest | 2010-08-16
(1 row)
```

Bagaimana Cara Menambah dan Menghapus Kolom dari Tabel

Jika ingin memodifikasi sebuah tabel setelah dibuat untuk menambah kolom baru, kita dapat melakukannya dengan mudah.

```
ALTER TABLE playground ADD last_maint date;
```

Untuk melihat informasi tabel kembali, gunakan perintah (perhatikan kolom terakhir yang ditambahkan):

```
SELECT * FROM playground;
```


equip_id	type	color	location	install_date	last_maint
2	swing	yellow	northwest	2010-08-16	

(1 row)

Kita juga dapat menghapus sebuah kolom dengan mudah. Menghapusnya dengan cara:

```
ALTER TABLE playground DROP last_maint;
```

Bagaimana Cara Mengupdate Data di Tabel

Kita tahu bagaimana menambah data ke dalam sebuah tabel dan bagaimana menghapusnya, tapi kita belum membahas bagaimana untuk memodifikasi data yang sudah ada. Misalnya, kita ingin mengambil data dengan tipe "swing" dan mengganti warnanya menjadi merah (red) (ini akan mengganti seluruh data yang memiliki tipe "swing"):

```
UPDATE playground SET color = 'red' WHERE type = 'swing';
```

Kita dapat memeriksa bahwa perintah berhasil dengan:

```
SELECT * FROM playground;
```


equip_id	type	color	location	install_date
2	swing	red	northwest	2010-08-16

(1 row)

TUGAS

Tuliskan perintah pada database serta berikan contoh penggunaannya:

- Membuat sebuah collection (mahasiswa)
- Memasukkan dokumen baru kedalam sebuah collection
- Mengupdate dokumen yang ada dalam sebuah collection

- Memasukkan dokumen baru ataupun mengupdate dokumen yang ada dalam sebuah collection
- Menghapus dokumen baru suatu collection
- Drops atau menghapus sebuah collection

Pertemuan 6 - Data as a Service dengan NoSQL MongoDB

Tujuan

1. Mahasiswa mampu memahami perbedaan SQL dan noSQL
2. Mahasiswa dapat mengimplementasikan NoSQL dengan model database MongoDB
3. Mahasiswa mampu mengelola database NoSQL MongoDB

Teori

Sebuah record di MongoDB adalah sebuah dokumen. MongoDB menyimpan data dalam bentuk dokumen, yang strukturnya terdiri dari field dan pasangan nilai. Dokumen ataupun struktur MongoDB mirip dengan objek JSON yang terdiri dari field dan pasangan nilainya. Secara formal, dokumen MongoDB adalah dokumen BSON. BSON adalah representasi biner dari JSON dengan jenis informasi tambahan. Dalam dokumen, nilai field bisa berupa salah satu tipe data BSON, termasuk dokumen-dokumen lainnya, array, dan array dokumen.

Berikut contoh dokumen mongoDB yang sederhana

```
{
  name: "sue",           ← field: value
  age: 26,               ← field: value
  status: "A",            ← field: value
  groups: [ "news", "sports" ] ← field: value
}
```

Berikut contoh dokumen mongoDB yang lebih kompleks

```
{
  "_id" : ObjectId("54c955492b7c8eb21818bd09"),
  "address" : {
    "street" : "2 Avenue",
```

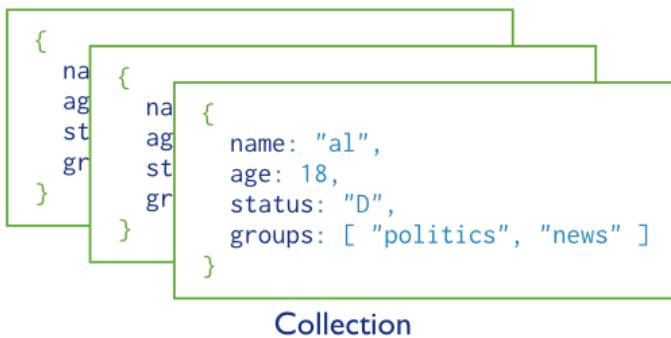
```

    "zipcode" : "10075",
    "building" : "1480",
    "coord" : [ -73.9557413, 40.7720266 ],
  },
  "borough" : "Manhattan",
  "cuisine" : "Italian",
  "grades" : [
    {
      "date" : ISODate("2014-10-01T00:00:00Z"),
      "grade" : "A",
      "score" : 11
    },
    {
      "date" : ISODate("2014-01-16T00:00:00Z"),
      "grade" : "B",
      "score" : 17
    }
  ],
  "name" : "Vella",
  "restaurant_id" : "41704620"
}

```

MongoDB menyimpan dokumen dalam collection. Collection analog (mirip) dengan tabel dalam database relasional. Collection adalah sekelompok dokumen terkait yang memiliki satu set indeks yang sama. Collection tidak memerlukan dokumen untuk memiliki skema yang sama.

Dalam MongoDB, dokumen yang disimpan dalam collection harus memiliki field `_id` unik yang bertindak sebagai primary key.



Capped collection adalah collection yang sudah ditentukan ukurannya, secara otomatis akan menggantikan record yang lama apabila telah mencapai batas maksimum yang telah ditetapkan.

Praktik

Menginstall mongodb

1. Buat file repo untuk mongodb sehingga kita bisa menginstall MongoDB secara langsung, menggunakan perintah yum

```
vim /etc/yum.repos.d/mongodb-org-3.0.repo
```

2. Isikan dengan script berikut:

```
[mongodb-org-3.0]
name=MongoDB Repository
baseurl=http://repo.mongodb.org/yum/redhat/$releasever/mongodb-
org/3.0/x86_64/
gpgcheck=0
enabled=1
```

3. Untuk menginstall mongoDB pada versi yang terakhri dan stabil gunakan perintah berikut:

```
sudo yum install -y mongodb-org
```

4. Kita harus mengkonfigurasi SELinux agar MongoDB dapat berjalan pada sistem yang berbasis RedHat. Sebelumnya kita harus menginstal policycoreutils-python agar bisa menggunakan perintah 'semanage'

```
yum -y install policycoreutils-python
```

5. Membuka akses ke port yang digunakan, secara default MongoDB menggunakan port 27017 untuk SELinux. Untuk itu kita perlu menjalankan perintah berikut:

```
semanage port -a -t mongod_port_t -p tcp 27017
```

6. Merubah konfigurasi SELinux menjadi permissive mode dalam file /etc/selinux/config:

```
vim /etc/selinux/config
```

```
SELINUX=enforcing
```

dirubah menjadi:

```
SELINUX=permissive
```

Setelah semua proses telah selesai, selanjutnya kita akan menjalankan MongoDB:

```
service mongod start
```

7. Agar MongoDB selalu start pada saat sistem berjalan, kita bisa menggunakan perintah berikut:

```
chkconfig mongod on
```

8. Untuk menghentikan MongoDB, kita bisa menggunakan perintah berikut:

```
service mongod stop
```

9. Untuk merestart MongoDB, kita bisa menggunakan perintah berikut:

```
service mongod restart
```

10. Untuk menganalisa setiap tahapan ataupun menganalisa jika terjadi error maka kita bisa membuka log file dengan perintah berikut:

```
cat /var/log/mongodb/mongod.log | more
```

Memulai dengan mongo shell

1. Untuk memulai menggunakan mongoDB pertama kali, kita bisa menggunakan mongodb melalui perintah shell, berikut perintah untuk menggunakan mongo shell

```
mongo
```

2. Perintah dasar:

Nama Perintah	Diskripsi
> help	Menampilkan informasi help

> show dbs	menampilkan semua database yang ada
> db	menampilkan database yang sedang digunakan
> db.help()	menampilkan informasi help untuk methods database
> use <database>	Menggunakan (mengaktifkan) database tertentu, atau digunakan untuk membuat database yang baru
> show collections	Menampilkan daftar semua collection pada database yang sedang digunakan
> db.createCollection(name, options)	Membuat collection baru. Biasanya digunakan untuk membuat capped collection.
> db.<collection>.help()	Menampilkan help pada methods collection. "<collection>" bisa berupa nama dari collection yang ada ataupun collection yang tidak ada.
> show users	Menampilkan daftar user dari database yang sedang digunakan
> show profile	Menampilkan 5 operasi yang sering digunakan dalam 1 milisecond atau lebih
> db.<collection>.find()	Mencari (menampilkan) semua dokumen pada collection tertentu
> Exit	Keluar dari mongo shell

Mengimport data

1. Download dataset dari link berikut, kemudian simpan dengan nama “primer-dataset.json”, simpan file tersebut dalam direktori “/home/<user>/” (direktori user yang digunakan untuk login). contoh: “/home/student/”
<https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/dataset.json>
2. Menggunakan perintah mongoimport untuk memasukkan dokumen ke dalam collection restaurants pada database. Jika collection sudah pernah ada sebelumnya didalam database test, maka operasi tersebut akan mendrop terlebih dahulu collection restaurants.

```
mongoimport --db test --collection restaurants --drop --file primer-dataset.json
```

Membuat database baru menggunakan mongo shell

1. Untuk membuat database baru kita bisa menggunakan perintah “use”, sama seperti kita mengaktifkan database

```
# mongo  
> use mydb
```

Untuk mengecek database yang ada digunakan perintah

```
> show dbs
```

2. Membuat uncapped collection, dengan ukuran maksimum 5 MB dengan menggunakan perintah berikut

```
> db.createCollection( "inventory", { size : 5242880} )
```

Untuk mengecek, digunakan perintah

```
> show collections
```

3. Memasukkan (insert) record (dokumen) ke dalam collection

```
> db.inventory.insert(  
  {  
    _id: 1,  
    item: "ABC1",  
    details: {  
      model: "14Q3",
```

```

        manufacturer: "XYZ Company"
    },
    stock: [ { size: "S", qty: 25 }, { size: "M", qty: 50 } ],
    category: "clothing"
}
)

```

Jika benar maka akan menampilkan hasil berikut

```
WriteResult({ "nInserted" : 1 })
```

4. Jika kita ingin memasukkan beberapa dokumen sekaligus, kita membuat suatu variable yang menampung array dari setiap dokumen. Untuk memisahkan setiap dokumen kita tambahkan tanda koma

```

> var mydocuments = [
    {
        _id: 2,
        item: "ABC2",
        details: { model: "14Q3", manufacturer: "M1 Corporation" },
        stock: [ { size: "M", qty: 50 } ],
        category: "clothing"
    },
    {
        _id: 3,
        item: "MNO2",
        details: { model: "14Q3", manufacturer: "ABC Company" },
        stock: [ { size: "S", qty: 5 }, { size: "M", qty: 5 }, { size:
    "L", qty: 1 } ],
        category: "clothing"
    },
    {
        _id: 4,
        item: "IJK2",
        details: { model: "14Q2", manufacturer: "M5 Corporation" },
        stock: [ { size: "S", qty: 5 }, { size: "L", qty: 1 } ],
        category: "houseware"
    }
];

```

Kemudian kita memasukkan variable dokumen dengan perintah insert

```
> db.inventory.insert( mydocuments );
```

Jika benar maka akan menampilkan hasil berikut

```
BulkWriteResult({  
    "writeErrors" : [ ],  
    "writeConcernErrors" : [ ],  
    "nInserted" : 3,  
    "nUpserted" : 0,  
    "nMatched" : 0,  
    "nModified" : 0,  
    "nRemoved" : 0,  
    "upserted" : [ ]  
)}
```

5. Untuk mereview dokumen-dokumen yang telah kita masukkan ke dalam collection kita bisa menggunakan perintah berikut

```
> db.inventory.find()
```

6. Memodifikasi/mengedit (update) record (dokumen) ke dalam collection, contoh perintah berikut akan mengupdate dokumen yang memiliki item “MNO2”

```
> db.inventory.update(  
    { item: "MNO2" },  
    {  
        $set: {  
            category: "apparel",  

```

Contoh perintah berikut akan mengupdate multiple dokumen yang memiliki kategori “clothing”

```
> db.inventory.update(  
    { category: "clothing" },  
    {  
        $set: { category: "apparel" },  
        $currentDate: { lastModified: true }  
    },  
    { multi: true }  
)
```

7. Menghapus (delete) record (dokumen) dari collection. Perintah berikut akan menghapus dokumen yang memiliki item “ABC1”

```
> db.inventory.remove( { item : "ABC1" } )
```

Perintah berikut akan menghapus semua dokumen, tanpa menghapus collection.

```
> db.inventory.remove( {} )
```

Perintah berikut akan menghapus collection secara keseluruhan, kosong maupun berisi

```
> db.inventory.drop()
```

Tugas

Tuliskan perintah pada database serta berikan contoh penggunaannya:

1. Membuat sebuah collection (mahasiswa)
2. Memasukkan dokumen baru kedalam sebuah collection
3. Mengupdate dokuman yang ada dalam sebuah collection
4. Memasukkan dokumen baru ataupun mengupdate dokumen yang ada dalam sebuah collection
5. Menghapus dokumen baru suatu collection
6. Drops atau menghapus sebuah collection

Pertemuan 7 - Pengantar Platform as a Services

Tujuan

1. Mahasiswa dapat membedakan antara SaaS dan PaaS
2. Mahasiswa dapat membuat aplikasi diatas Platform as a service
3. Mahasiswa dapat memahami workflow dari PaaS

Teori

Heroku merupakan salah satu layanan cloud yang menyediakan layanan platform as a service (PaaS) yang mendukung beberapa bahasa pemrograman. Heroku adalah salah satu platform cloud pertama yang diciptakan sejak juni 2007, dimana pada awal Heroku hanya mendukung bahasa pemrograman ruby. Namun sekarang Heroku mendukung bahasa pemrograman Java, Node.js, Scala, Clojure, Python, PHP dan Go. Heroku dikatakan sebagai platform polyglot karena memungkinkan pengembang membangun, menjalankan, dan menskala aplikasi dengan cara yang sama di semua bahasa. Pengembang yang akan melakukan deploy aplikasi cukup melakukan konfigurasi aplikasi menurut bahasa pemrograman yang dipakai untuk dikembangkan, dijalankan dan dikelola tanpa kompleksitas membangun dan memelihara infrastruktur untuk pengembangan dan peluncuran aplikasi.

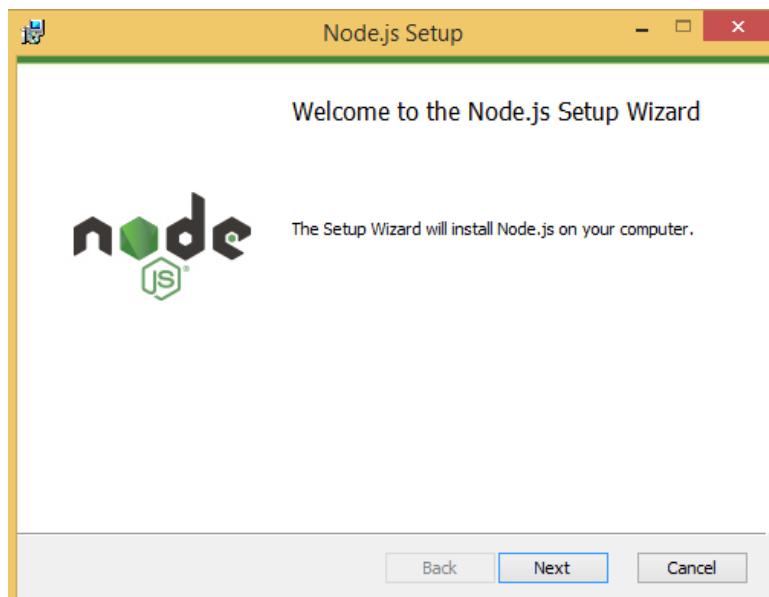
Praktik

Pertama, Mendaftar akun pengguna di Heroku dengan mengakses pranala berikut : <https://signup.heroku.com/>, lalu mengisi atribut pendaftaran sekaligus mencantumkan email sehingga nanti akan ada konfirmasi pada email untuk konfigurasi password akun Heroku tersebut.

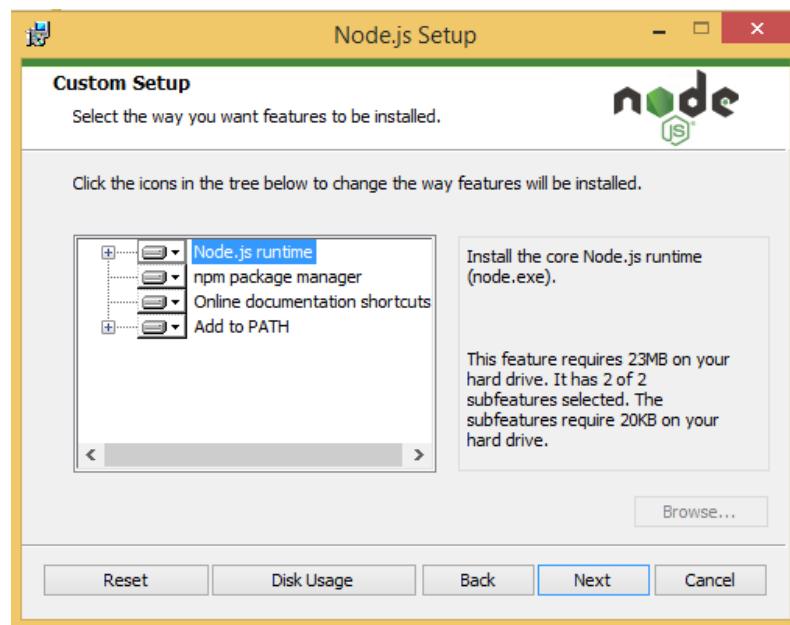
Kemudian terlihat tampilan dashboard Heroku seperti berikut :

The screenshot shows the Heroku dashboard interface. At the top, there's a navigation bar with the Heroku logo, a search bar labeled 'Jump to Favorites, Apps, Pipelines, Spaces...', and user profile icons. Below the navigation is a 'Personal' dropdown menu and a 'New' button. The main content area has a message 'You don't have any apps yet' and a sub-message 'Every app and pipeline you create or become a collaborator on will appear here'. A 'Create new app' button is located below this. Further down, there's a section titled 'Looking for help getting started?' with a sub-message 'Get started by reading one of our language guides in the Dev Center' and a 'Choose a language guide... ▾' button.

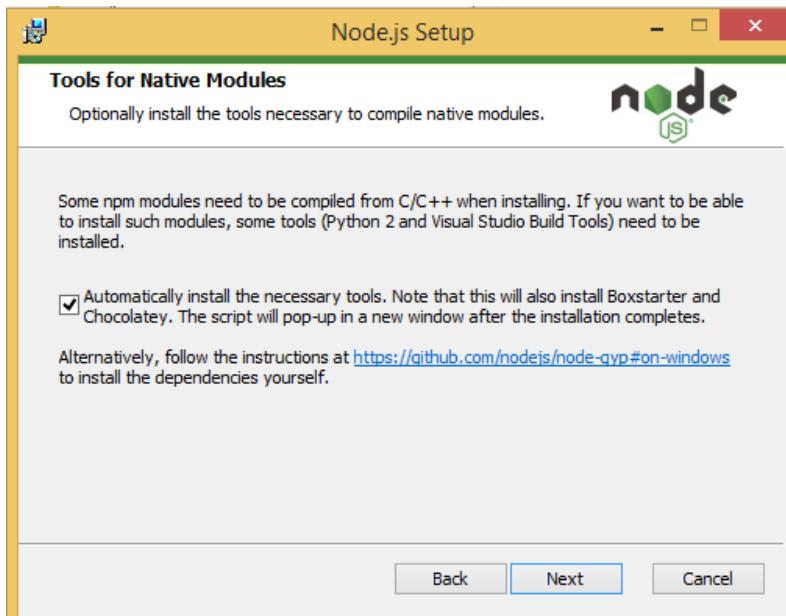
Kedua, Memasang node.js pada perangkat komputer dengan mengunduh file node-v10.13.0-x64.msi. kemudian muncul tampilan berikut:



Berikutnya terdapat agreement klik centang, lalu next. Pada custom setup tidak perlu diubah. Setelah itu klik next.



Kemudian pilih centang pada automatically install the necessary tools nya



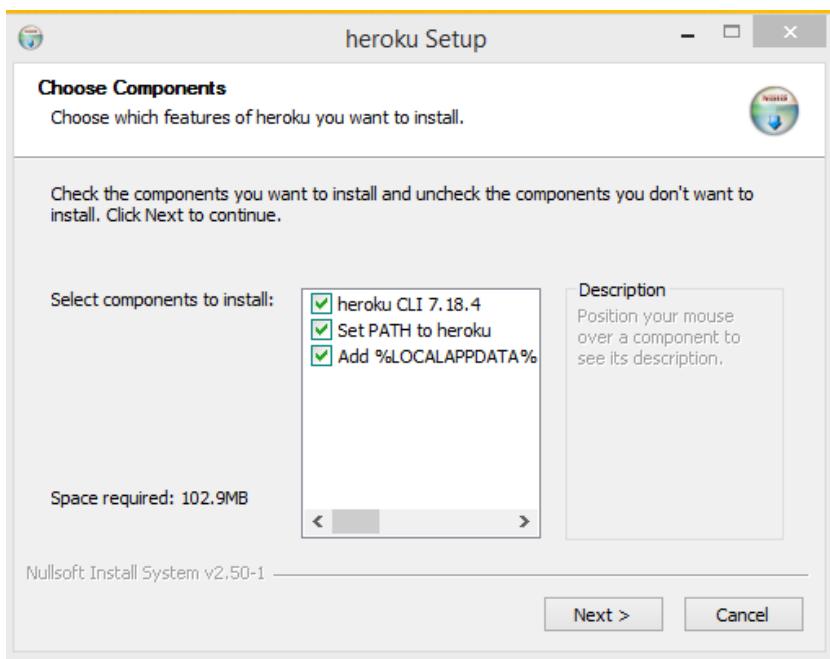
Setelah itu melakukan pengecekan apakah instalasi berjalan atau tidak, dengan perintah di cmd seperti berikut :

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

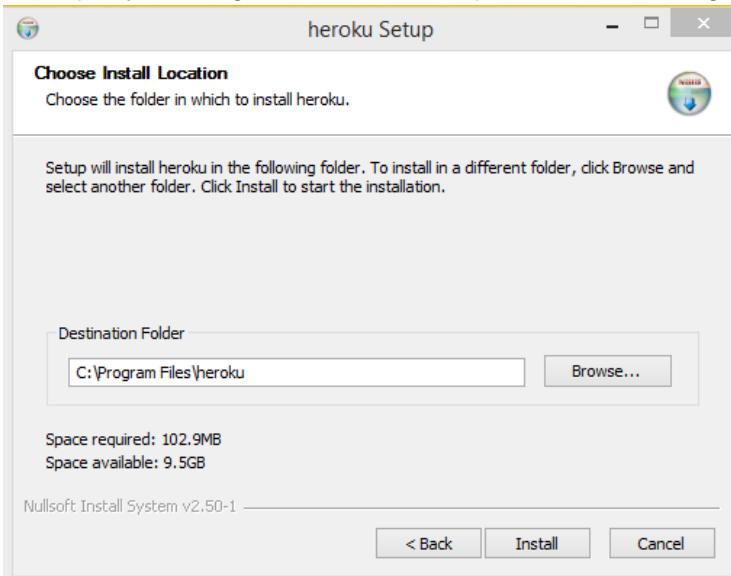
C:\Users\tamvan>node -v
v10.13.0

C:\Users\tamvan>npm -v
6.4.1
```

Untuk melakukan instalasi Heroku pada perangkat computer, Centang semua komponen yang akan diinstall pada tab choose components



Selanjutnya, mengarahkan instalasi pada direktori Program Files yang berada di disk C:



Setelah proses instalasi selesai cek instalasi dengan perintah di cmd:

heroku -v

Ketiga, Mengunduh aplikasi web

Clone project dengan git bash dengan perintah berikut:

```
git clone https://github.com/chessaAditirta/web_project
```

Catatan

Ubah nama folder web_project dengan nama diikuti nim karena nama folder dan nama aplikasi nanti digunakan sebagai subdomain aplikasi di Heroku. Contoh format : Chessa155610037

Keempat, memasang http-server dengan npm dengan perintah :

```
npm install http-server -g
```

Kelima, Mengarahkan cursor menuju ke direktori projek dan menjalankan aplikasi di lokal dengan perintah seperti pada gambar di bawah ini :

```
C:\Users\tamvan>cd chessa155610037  
C:\Users\tamvan\chessa155610037>http-server -p 8080 -a localhost -c 0  
Starting up http-server, serving ./  
Available on:  
  http://localhost:8080  
Hit CTRL-C to stop the server
```

http-server adalah potongan code di atas untuk menggunakan fungsi dengan http-server yang sebelumnya diinstalasi dan -p 8080 adalah port aplikasi yang berjalan. Setelah itu -a localhost adalah address atau alamat akses aplikasi. Kemudian yang terakhir -c 0 yang merupakan cache time dalam satuan detik yang berarti cache time bernilai 0.

Keenam, Melakukan inisiasi npm dengan perintah npm init -y yang akan memunculkan file bernama package.json yang berisi nama aplikasi, deskripsi aplikasi, dependensi aplikasi dan lain sebagainya seperti file definisi aplikasi sejenis seperti pom.xml milik maven dan build.gradle milik gradle.

```
C:\Users\tamvan\chessa155610037>npm init -y  
Wrote to C:\Users\tamvan\chessa155610037\package.json:  
  
{  
  "name": "chessa155610037",  
  "version": "1.0.0",  
  "description": "Tayo^2 Programming",  
  "main": "modernizr-custom.js",  
  "scripts": {  
    "test": "echo \\"$Error: no test specified\\" && exit 1"  
  },  
  "repository": {  
    "type": "git",  
    "url": "git+https://github.com/chessaAditirta/web_project.git"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "bugs": {  
    "url": "https://github.com/chessaAditirta/web_project/issues"  
  },  
  "homepage": "https://github.com/chessaAditirta/web_project#readme"  
}
```

Lalu, install node static dengan perintah npm install node-static --save

```
C:\Users\tamvan\chessa155610037>npm install node-static --save
+ node-static@0.7.11
updated 1 package and audited 6 packages in 2.835s
found 0 vulnerabilities
```

Kemudian cek kembali dengan perintah npm init -y maka akan ada tambahan dependensi yaitu node static

```
C:\Users\tamvan\chessa155610037>npm init -y
Wrote to C:\Users\tamvan\chessa155610037\package.json:

{
  "name": "chessa155610037",
  "version": "1.0.0",
  "description": "Tayo^2 Programming",
  "main": "modernizr-custom.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/chessaAditirta/web_project.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/chessaAditirta/web_project/issues"
  },
  "homepage": "https://github.com/chessaAditirta/web_project#readme",
  "dependencies": {
    "node-static": "^0.7.11"
  },
  "devDependencies": {}
}
```

Ketujuh, Membuat index.js untuk mengorganisasi jalannya aplikasi dengan node.js Sebelumnya tambahkan perintah "start": "node index.js", dan hapus beberapa code sehingga pada package.json akan terlihat seperti gambar di bawah:

```
C:\Users\tamvan\chessa155610037>npm init -y
Wrote to C:\Users\tamvan\chessa155610037\package.json:

{
  "name": "chessa155610037",
  "version": "1.0.0",
  "description": "Tayo^2 Programming",
  "main": "modernizr-custom.js",
  "scripts": {
    "start": "node index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "node-static": "^0.7.11"
  },
  "devDependencies": {}
}
```

Kemudian buat file index.js dengan code seperti dibawah,

```
var static = require('node-static');
var file = new static.Server();

require('http').createServer(function(request, response) {
  request.addListener('end', function() {
    file.serve(request, response);
  }).resume();
}).listen(process.env.PORT || 5000);
```

Terakhir jalankan perintah node index.js pada cmd untuk menjalankan web statis dengan index.js. Jika tidak menemukan error log buka pada **http://localhost:5000** untuk tes aplikasi di local.

Kedelapan, Menghubungkan direktori pada git lalu git local dihubungkan ke Heroku
Sebelum melakukan inisiasi git hapus folder .git bila tidak menemukan temukan dengan show hidden folder pada view tab.

Menjalankan perintah git init untuk inisialisasi git untuk menggantikan .git yang tadi dihapus karena milik repository sebelumnya.

```
C:\Users\tamvan\chessa155610037>git init
Initialized empty Git repository in C:/Users/tamvan/chessa155610037/.git/
```

Menjalankan perintah heroku login untuk masuk ke akun Heroku masing-masing melalui cmd

```
C:\Users\tamvan\chessa155610037>heroku login  
heroku: Enter your login credentials  
Email: masihsendiri999@gmail.com  
Password: *****  
Logged in as masihsendiri999@gmail.com
```

Membuat aplikasi diheroku dengan format perintah heroku create name-app. Seperti perintah sebelumnya nama folder dan nama aplikasi yang digunakan adalah format namaNim maka perintahnya menjadi seperti berikut:

```
C:\Users\tamvan\chessa155610037>heroku create chessa155610037  
Creating chessa155610037... done  
https://chessa155610037.herokuapp.com/ | https://git.heroku.com/chessa155610037.git
```

Ada dua alamat yang tertera pada hasil pembuatan aplikasi dimana yang kiri adalah domain aplikasi dan kanan adalah alamat repo git yang terintegrasi ke Heroku. Dengan alamat repo tersebut dapat dimungkinkan untuk melakukan git clone app dengan alamat tersebut dengan syarat telah login ke Heroku atau perintah Heroku login. Pembuktian dapat dilihat bila saudara menjalankan perintah *git remote -v*.

```
C:\Users\tamvan\chessa155610037>git remote -v  
heroku https://git.heroku.com/chessa155610037.git (fetch)  
heroku https://git.heroku.com/chessa155610037.git (push)
```

Gambar diatas menunjukkan push dan fetch beralamat di git.heroku, maka jangan takut kehilangan kode program bila setelah di push nanti karena sudah tersimpan di cloud. Saudara tinggal melakukan clone app dengan git clone alamat tersebut.

Sembilan, Push proyek ke Heroku sekaligus deploy aplikasi

Salah satu keunggulan Heroku adalah mendeploy aplikasi dan push code programnya dengan sekali push karena git local saudara telah terintegrasi ke akun Heroku saudara. Oleh karena itu istilah di Heroku bukan deploy apps melainkan push apps. Sebelum melakukan push perlu diingat bergitu push apps dijalankan Heroku pertama kali akan menjalankan perintah npm install dimana direktori aplikasi kita akan ditambahkan direktori dengan nama node_modules. Mekanisme tersebut terjadi karena kita menggunakan ekosistem node.js dimana node_modules adalah direktori yang berisi plugin, ekstensi atau kebutuhan resource aplikasi node.js. Hal ini seperti mekanisme SDK pada pemrograman Android studio. Singkatnya kita perlu melakukan ignore terhadap direktori node_modules tersebut agar aplikasi web statis bisa dijalankan untuk itu perlu membuat file .gitignore. Jadi buatlah file **.gitignore** didalam direktori source anda, kemudian isi konten .gitignore dengan ini : **/node_modules**. Kemudian jalankan git add

```
C:\Users\tamvan\chessa155610037>git add .
warning: LF will be replaced by CRLF in about.html.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in index.html.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in modernizr-custom.js.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in package-lock.json.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in package.json.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in styles/main.css.
The file will have its original line endings in your working directory.
```

Kemudian lakukan perintah commit dengan format

```
git commit -m "isi bebas lepas tanpa batas"
```

```
C:\Users\tamvan\chessa155610037>git commit -m "1.1.3:Web apps push to Heroku"
[master (root-commit) c98d0ae] 1.1.3:Web apps push to Heroku
```

Terakhir jalankan perintah dengan format:

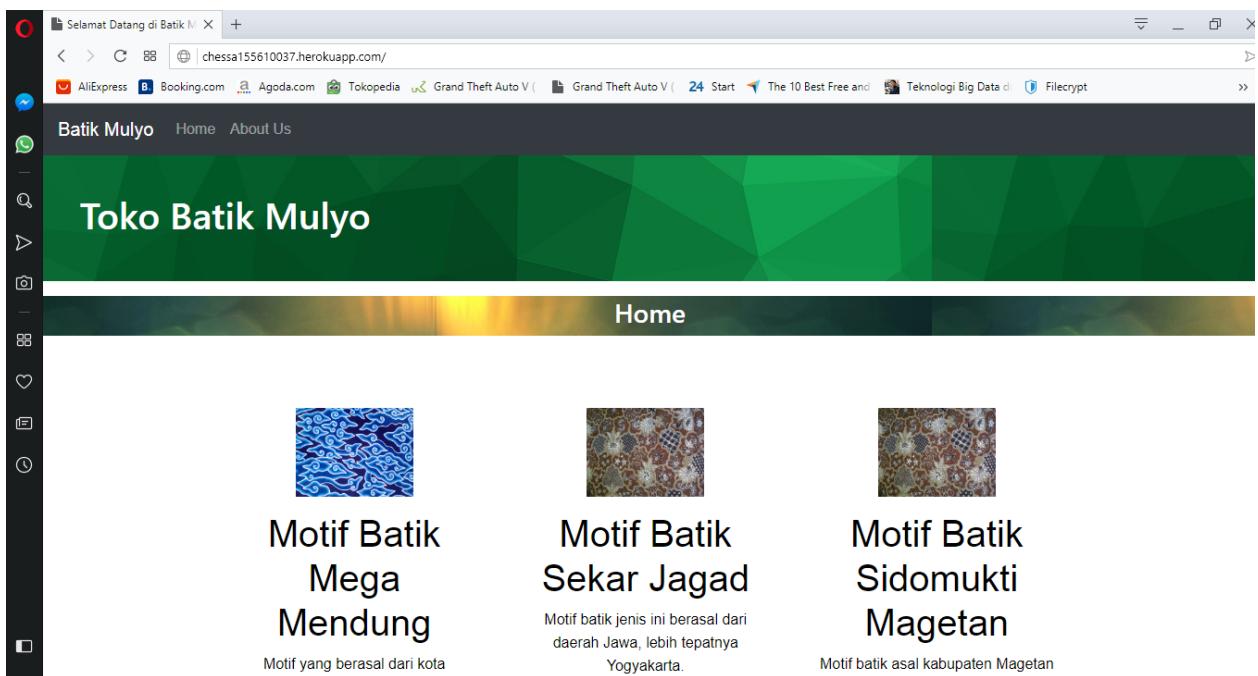
```
git push heroku master
```

```
C:\Users\tamvan\chessa155610037>git push heroku master
Counting objects: 28, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (25/25), done.
Writing objects: 100% (28/28), 2.91 MiB | 243.00 KiB/s, done.
Total 28 (delta 0), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Node.js app detected
remote:
remote: -----> Creating runtime environment
remote:
remote:     NPM_CONFIG_LOGLEVEL=error
remote:     NODE_ENV=production
remote:     NODE_MODULES_CACHE=true
remote:     NODE_VERBOSE=false
remote:
remote: -----> Caching build
remote:         - node_modules
remote:
remote: -----> Pruning devDependencies
remote:     audited 6 packages in 0.806s
remote:     found 0 vulnerabilities
remote:
remote:
remote: -----> Build succeeded!
remote: -----> Discovering process types
remote:     Procfile declares types      -> (none)
remote:     Default types for buildpack -> web
remote:
remote: -----> Compressing...
remote:     Done: 20.9M
remote: -----> Launching...
remote:     Released v3
remote:     https://chessa155610037.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/chessa155610037.git
 * [new branch]      master -> master
```

Catatan

Jangan mengetik perintah `git push origin master`. Ingat nama remote kita adalah `heroku` coba kembali ke halaman yang membahas `git remote -v`.

Untuk melihat hasilnya, jalankan perintah `heroku open` untuk membuka aplikasi pada browser.



Aplikasi yang telah di push ke Heroku dan memperoleh alamat
<https://chessa155610037.herokuapp.com/>

Tugas

Buat aplikasi web statis menurut imajinasi masing-masing memiliki beberapa halaman web dan salah satu halamannya tersusun atas nama, nim serta hal yang menunjukkan profil saudara seperti halaman web mengenai tentang , profil atau About pada umumnya.

Pertemuan 8 - Platform as a service Docker di Openshift

Tujuan

1. Mahasiswa dapat memahami pemanfaatan docker pada platform as a service openshift
2. Mahasiswa dapat membuat aplikasi, deploy, dan mengelola container docker melalui platform openshift.

Teori

OpenShift Origin, atau dikenal sebagai OKD sejak Agustus 2018 merupakan proyek komunitas yang digunakan untuk basis openshift online, openshift dedicated, dan openshift container platform. Dibangun dengan menggunakan teknologi container docker dan kubernetes sebagai manajemen kluster. Openshift origin juga merupakan secara fungsionalitas mendukung manajemen ekosistem pengembangan aplikasi dan devops.

OpenShift memungkinkan kemampuan untuk dengan mudah menerapkan kode aplikasi web secara langsung menggunakan pustaka dari image yang dibuat sebelumnya atau kita dapat membuat image Docker kita sendiri. Dengan fitur seperti persistent volume, kami tidak terbatas hanya menjalankan aplikasi bawaan cloud, tetapi juga mendeploy basis data dan banyak jenis aplikasi.

Minishift merupakan versi ringan dari openshift yang ditujukan untuk kemudahan dalam mempelajari platform as a service di komputer/laptop.

Praktik

Installing Minishift on Ubuntu Linux (VirtualBox)

Disarankan sistem linux ubuntu sudah menggunakan virtualbox 5.1.x, instalasi juga dapat dilakukan dengan mudah di windows. Virtualbox terbaru dapat di download pada link https://www.virtualbox.org/wiki/Linux_Downloads

Untuk binari dari minishift dapat di download melalui

<https://github.com/minishift/minishift/releases>

Downloads

minishift-1.1.0-darwin-amd64.tgz	3.86 MB
minishift-1.1.0-darwin-amd64.tgz.sha256	65 Bytes
minishift-1.1.0-linux-amd64.tgz	3.66 MB
minishift-1.1.0-linux-amd64.tgz.sha256	65 Bytes
minishift-1.1.0-windows-amd64.zip	3.67 MB
minishift-1.1.0-windows-amd64.zip.sha256	65 Bytes
Source code (zip)	
Source code (tar.gz)	

Unzip file ke folder yang telah ditentukan lalu jalankan perintah berikut:

```
$ minishift start --vm-driver=virtualbox
```

Konfigurasi path untuk openshift

```
jga@novatec-minishift:~/minishift$ ./minishift oc-env
export PATH="/home/jga/.minishift/cache/oc/v1.5.1:$PATH"
# Run this command to configure your shell:
# eval $(minishift oc-env)
jga@novatec-minishift:~/minishift$ export
PATH="/home/jga/.minishift/cache/oc/v1.5.1:$PATH"
```

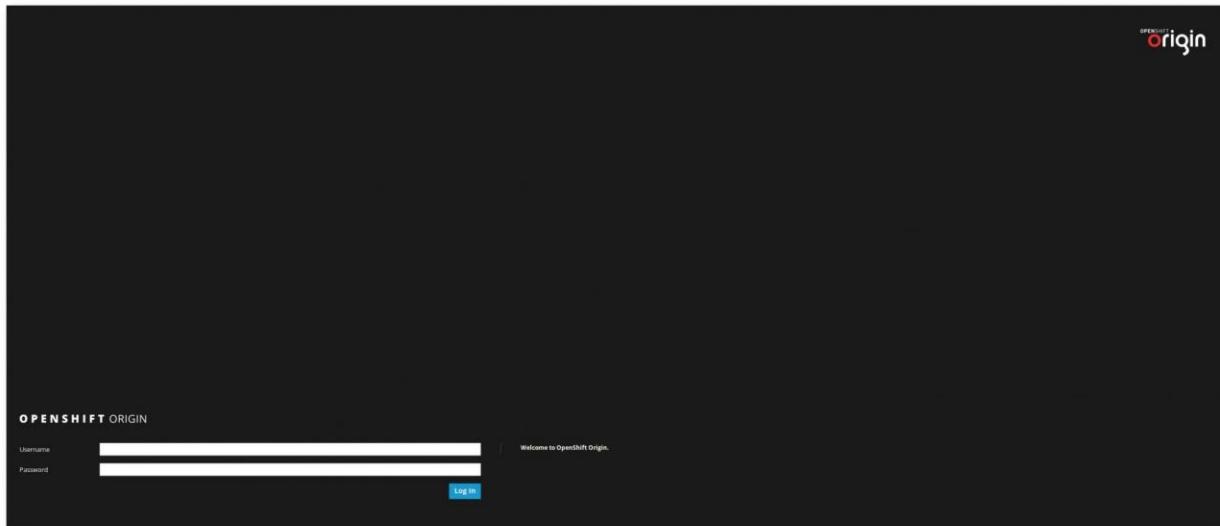
Terhubung OpenShift Origin (Minishift)

Setelah proses instalasi untuk dapat terhubung ke minishift gunakan perintah:

```
$ minishift console --url
```

Contoh:

```
jga@novatec-minishift:~/minishift$ ./minishift console --url
https://192.168.42.208:8443
jga@novatec-minishift:~/minishift$
```



Secara default login dengan username dan password: **developer/developer**

Sekarang login sebagai admin dengan menggunakan CLI:

```
oc login https://192.168.42.208:8443
```

Gunakan user dan password: *developer/developer*.

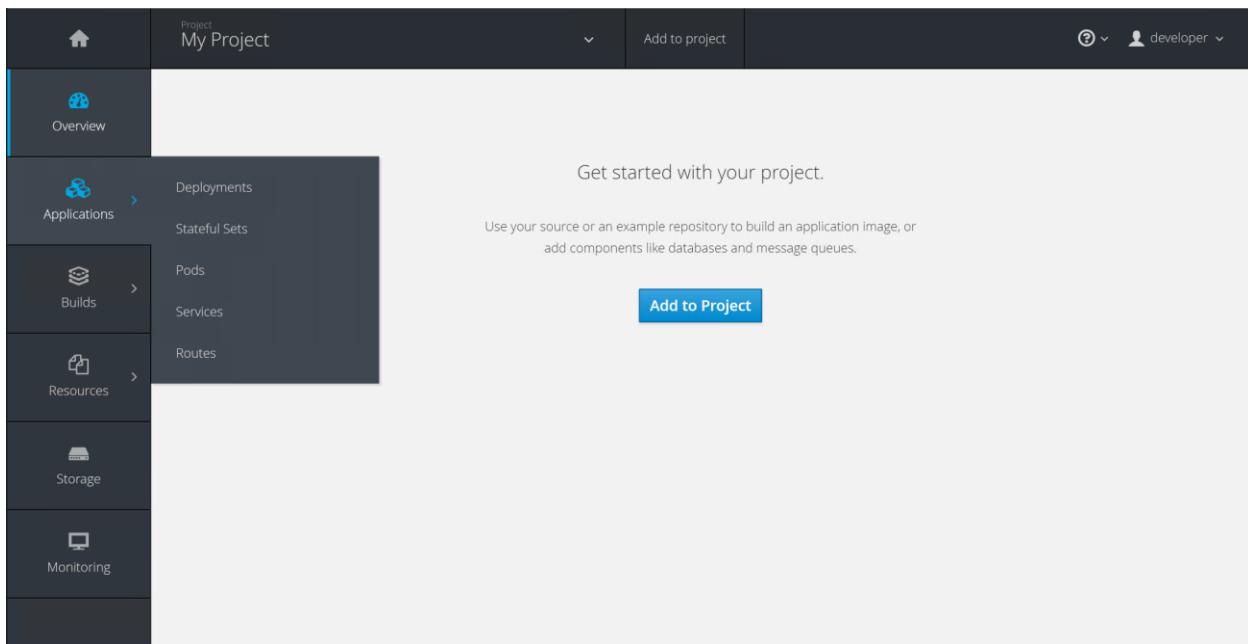
```
jga@novatec-minishift:~/minishift$ oc login https://192.168.42.208:8443
Authentication required for https://192.168.42.208:8443 (openshift)
Username: developer
Password:
Login successful.

You have one project on this server: "myproject"

Using project "myproject".
jga@novatec-minishift:~/minishift$ oc status
In project My Project (myproject) on server https://192.168.42.208:8443

You have no services, deployment configs, or build configs.
Run 'oc new-app' to create an application.
jga@novatec-minishift:~/minishift$
```

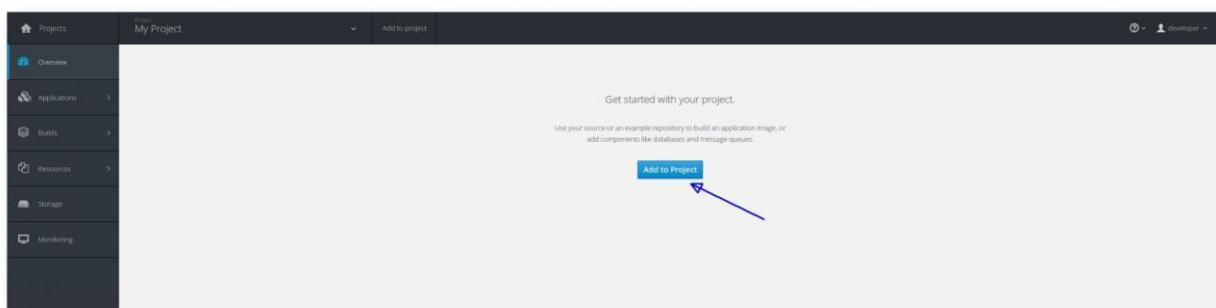
Untuk menggunakan GUI atau tampilan grafis



Deploy aplikasi

Dalam contoh ini akan menggunakan aplikasi helloworld berbasis java yang diambil dari github: [gshipley/book-helloworld](https://github.com/gshipley/book-helloworld)

Setelah itu aplikasi ini akan di install kedalam openshift. Pertama, buatlah project dengan klik **Add to Project**.



Pilih Java Template dan WildFly:

The screenshot shows the OpenShift Origin interface with the title "OPENSHIFT ORIGIN" at the top. Below it, the URL "My Project > Add to Project" is visible. There are three tabs: "Browse Catalog" (which is active), "Deploy Image", and "Import YAML / JSON". A search bar with the placeholder "Filter by name or description" is present. The main area is titled "Languages" and contains six categories: Java (selected), JavaScript, Perl, PHP, Python, and Ruby. Below this, under "Technologies", there are three categories: "Continuous Integration & Deployment" (with a sub-note about automating build, test, and deployment), "Data Stores" (with a note about managing collections of data), and "Uncategorized".

Anda akan mendapatkan tampilan seperti berikut:

This is a detailed view of the WildFly builder image card. At the top, the title "Java" is displayed above a search bar with the placeholder "Filter by name or description". The card features the WildFly logo and the text "WildFly". Below this, the section "BUILDS SOURCE CODE" is shown with the subtext: "Build and run WildFly 10.1 applications on CentOS 7. For more information about using this builder image, including OpenShift considerations, see <https://github.com/openshift-s2i/s2i-wildfly/blob/mast...>". A dropdown menu for "Version" is set to "10.1 — latest", with a "Select" button below it.

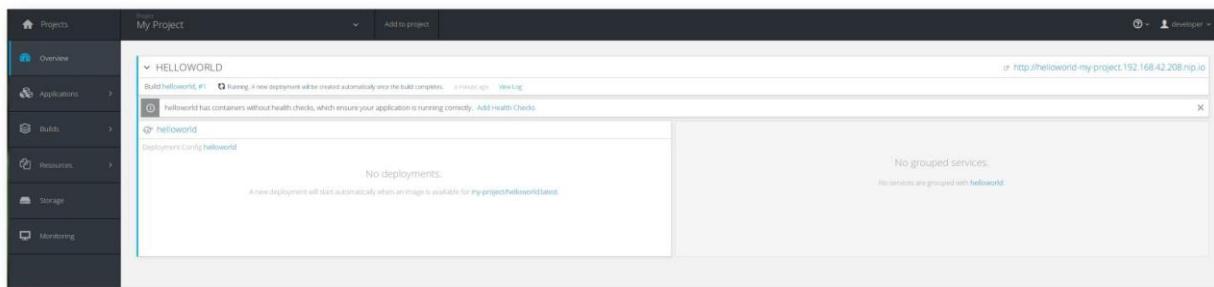
Selanjutnya copy paste repository berikut <https://github.com/jmgalvez/book-helloworld.git>

The screenshot shows a configuration interface for a WildFly application. At the top, there's a navigation bar with links: My Project > Add to Project > Catalog > WildFly. Below the navigation is a WildFly logo and the text "WildFly Version: 10.1". The main area contains a form with fields for "Name" (set to "helloworld") and "Git Repository URL" (set to "https://github.com/jmgalvez/book-helloworld.git"). A note below the URL says "Identifies the resources created for this application." and "Sample repository for wildfly: https://github.com/bparees/openshift-jee-sample.git Try It ↑". There's also a link to "Show advanced options for source, routes, builds, and deployments.". At the bottom of the form are two buttons: "Create" (in blue) and "Cancel".

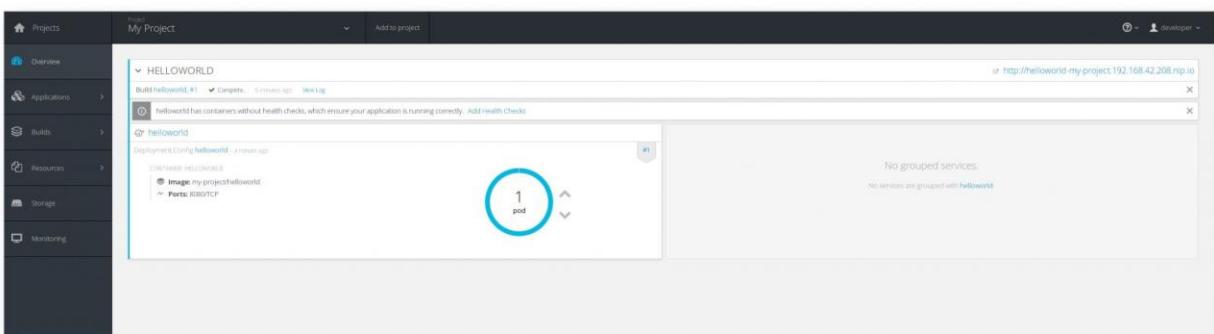
Setelah klik CREATE maka akan muncul tampilan seperti berikut :

The screenshot shows a confirmation page after creating an application. The navigation bar at the top includes "My Project > Add to Project > WildFly > Next Steps". The main content area starts with the message "Application created. Continue to overview.". Below this, there are two sections: "Manage your app" and "Making code changes". The "Manage your app" section provides instructions for using the web console or command line tools, mentioning the "oc" tool and providing a terminal window with commands like "oc login", "oc project my-project", and "oc status". The "Making code changes" section informs the user that a GitHub webhook trigger has been created for the "helloworld" build config and provides a URL for setting it up: "https://github.com/jmgalvez/book-helloworld/settings/hooks".

Pada proses ini S2I telah dieksekusi dan membutuhkan waktu beberapa menit untuk melakukan proses pembuatan image dan kompilasi source code. Jika telah selesai akan muncul notifikasi: *A new deployment will be created automatically once the build completes.*



Setelah selesai anda akan melihat indikator berwarna biru yang menunjukkan 1 pod dan dalam tahap ini apps sudah tersedia.



Tampilan aplikasi di browser



Pertemuan 9 - Source Control Management dengan Git

Tujuan

1. Mahasiswa dapat memahami workflow dari source control manajemen
2. Mahasiswa mampu menggunakan dan mengelola source code di layanan git
3. Mahasiswa dapat berkolaborasi dalam proyek aplikasi dengan layanan git

Teori

Git adalah salah satu sistem pengontrol versi (*Version Control System*) pada proyek perangkat lunak yang diciptakan oleh Linus Torvalds. Pengontrol versi bertugas mencatat setiap perubahan pada file proyek yang dikerjakan oleh banyak orang maupun sendiri. Git dikenal juga dengan *distributed revision control* (VCS terdistribusi), artinya penyimpanan database Git tidak hanya berada dalam satu tempat saja.

Semua orang yang terlibat dalam pengkodean proyek akan menyimpan database Git, sehingga akan memudahkan dalam mengelola proyek baik online maupun offline. Dalam Git terdapat merge untuk menyebut aktivitas penggabungan kode. Sedangkan pada VCS (Version Control System) yang terpusat, database disimpan dalam satu tempat dan setiap perubahan disimpan ke sana.

VCS terpusat memiliki beberapa kekurangan:

- Semua tim harus terkoneksi ke jaringan untuk mengakses source-code;
- Tersimpan di satu tempat, nanti kalau server bermasalah bagaimana?

Karena itu, Git hadir untuk menutupi kekurangan yang dimiliki oleh VCS terpusat.

Praktik

1. Cara Install Git di Linux

Instalasi Git pada Distro keluarga Debian dapat menggunakan perintah apt.

```
sudo apt install git
```

atau

```
sudo apt-get install git
```

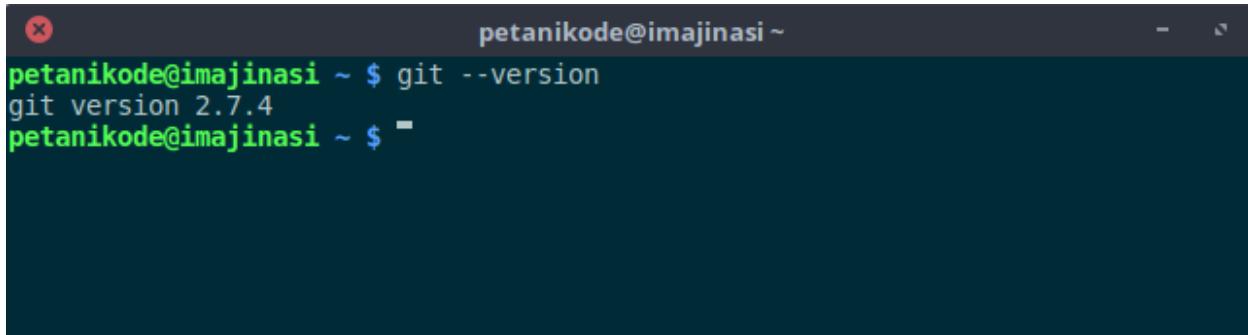
Pada Fedora:

```
yum install git
```

Setelah itu, coba periksa versi yang terinstal dengan perintah:

```
git --version
```

Pada komputer saya, versi yang terinstal adalah versi 2.7.4.



```
petanikode@imajinasi ~
petanikode@imajinasi ~ $ git --version
git version 2.7.4
petanikode@imajinasi ~ $ -
```

2. Cara Install Git di Windows

Instalasi Git di Windows memang tidak seperti di Linux yang ketik perintah langsung terinstal. Kita harus men-download dulu, kemudian melakukan ritual *next>next>finish*.

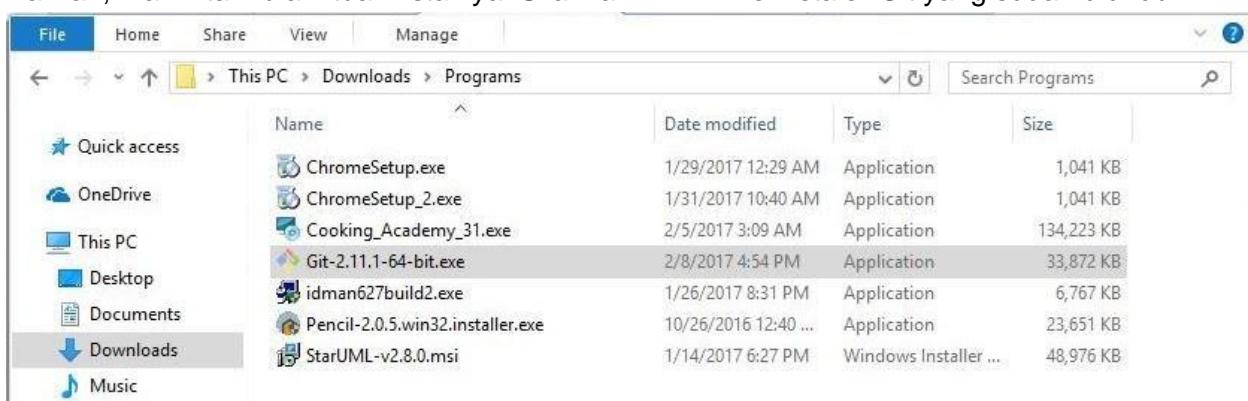
Tapi dalam ritual tersebut, ada pilihan yang harus diperhatikan agar perintah git dapat dikenali di CMD.

Download Git

Silahkan buka website resminya Git (git-scm.com). Kemudian unduh Git sesuai dengan arsitektur komputer kita. Kalau menggunakan 64bit, unduh yang 64bit. Begitu juga kalau menggunakan 32bit.

Langkah-langkah Install Git di Windows

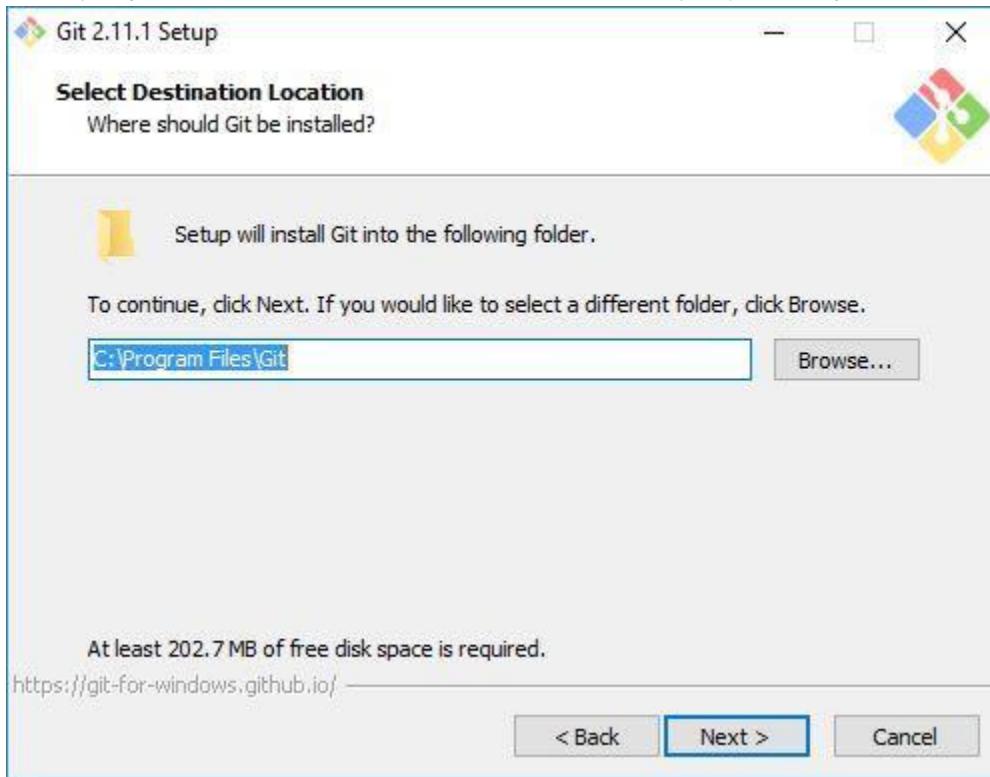
Baiklah, mari kita mulai ritual instalnya. Silahkan klik 2x file instalator Git yang sudah diunduh.



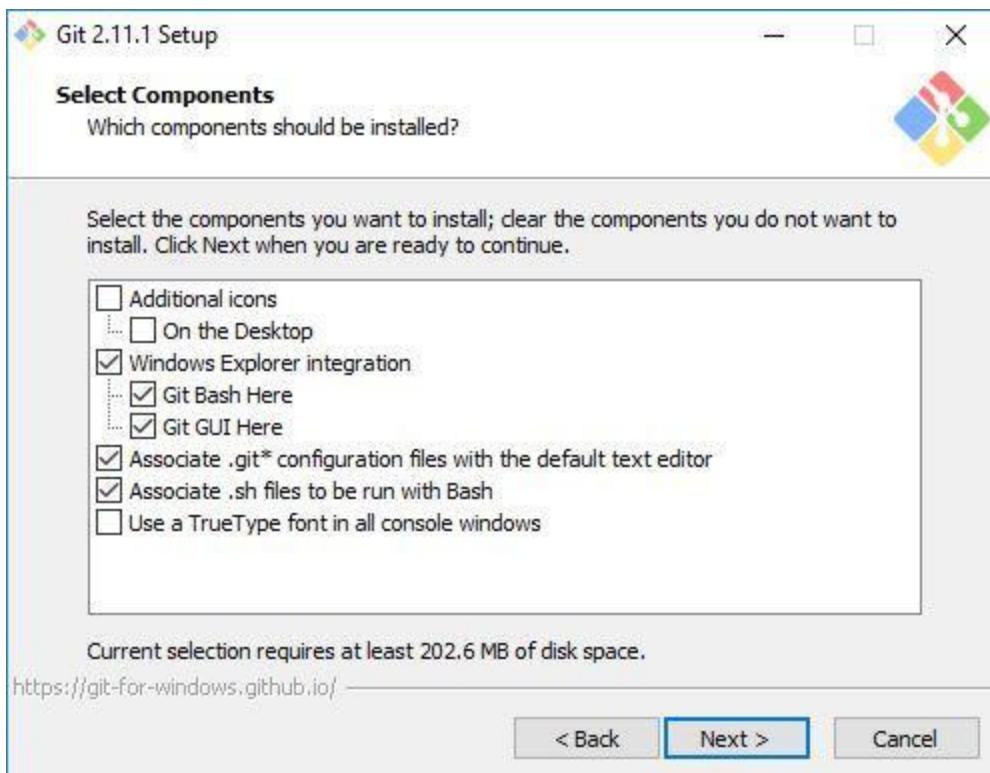
Maka akan muncul infomasi lisensi Git, klik *Next >* untuk melanjutkan.



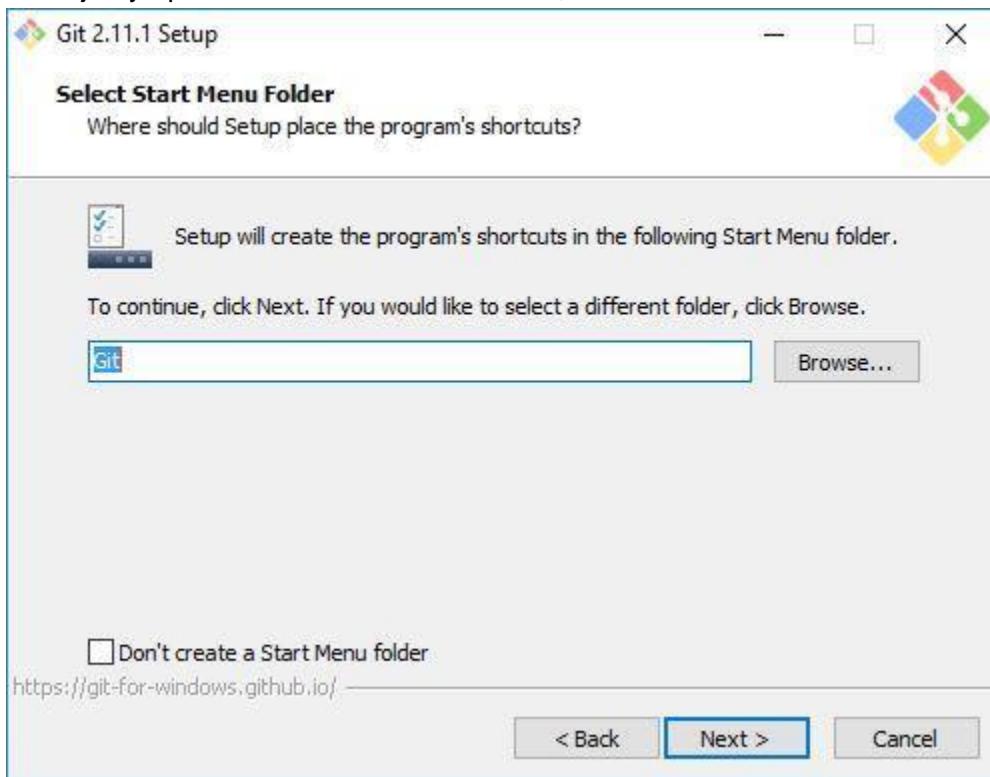
Selanjutnya menentukan lokasi instalasi. Biarkan saja apa adanya, kemudian klik **Next >**.



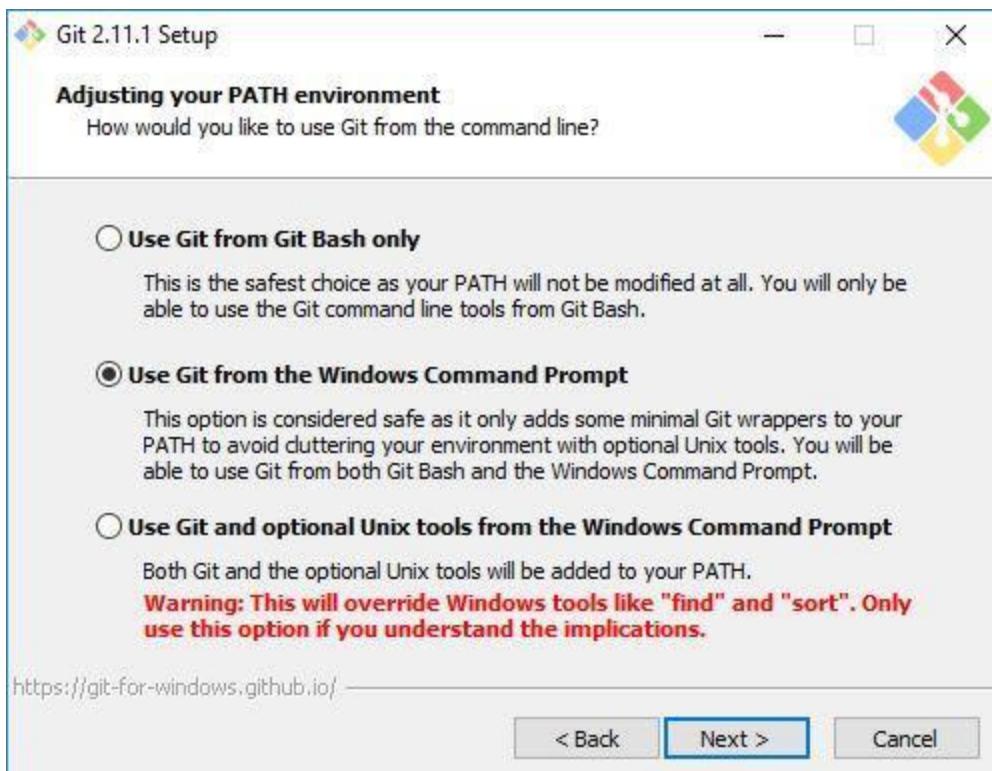
Selanjutnya pemilihan komponen, biarkan saja seperti ini kemudian klik **Next >**.



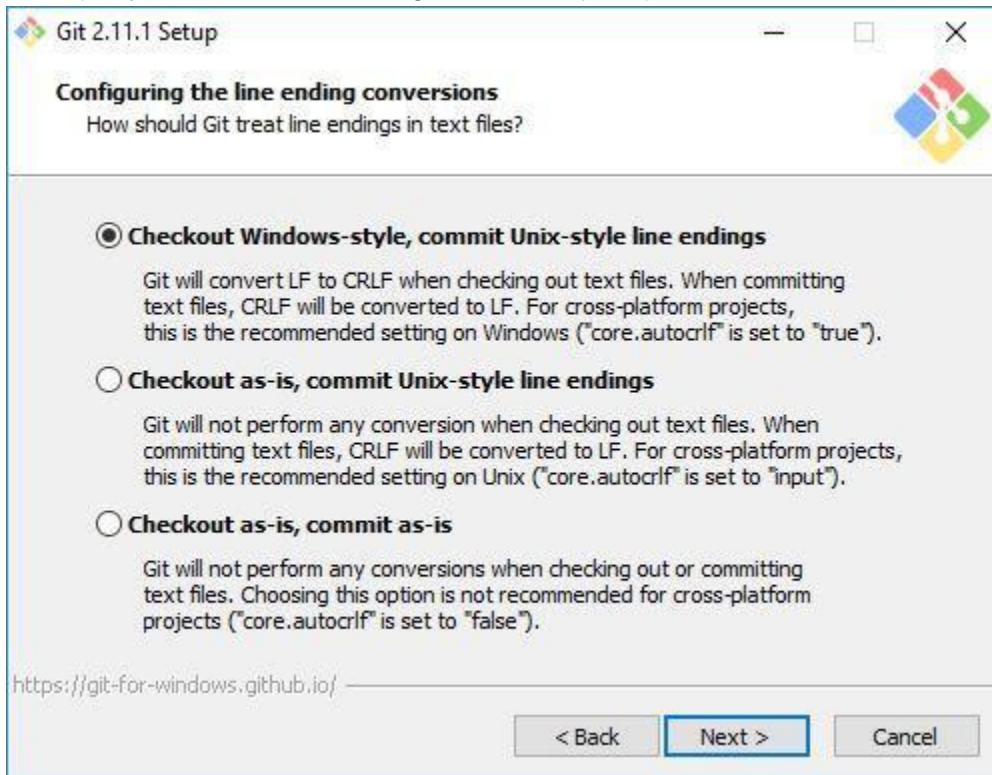
Selanjutnya pemilihan direktori start menu, klik *Next >*.



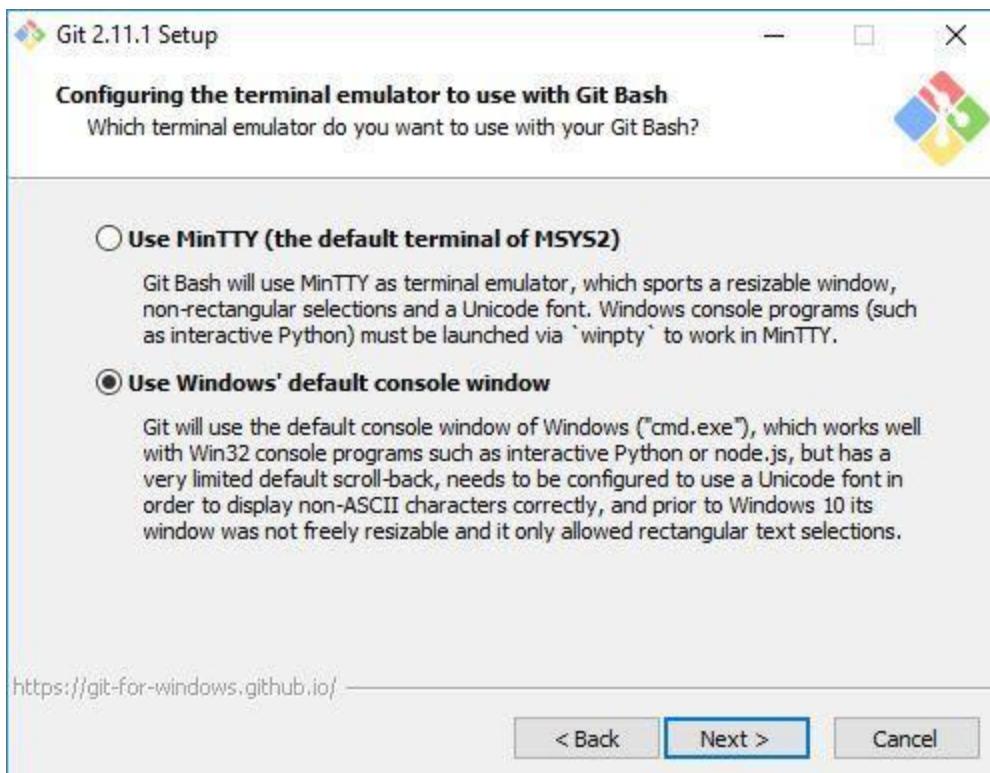
Selanjutnya pengaturan *PATH Environment*. Pilih yang tengah agar perintah git dapat di kenali di *Command Prompt* (CMD). Setelah itu klik *Next >*.



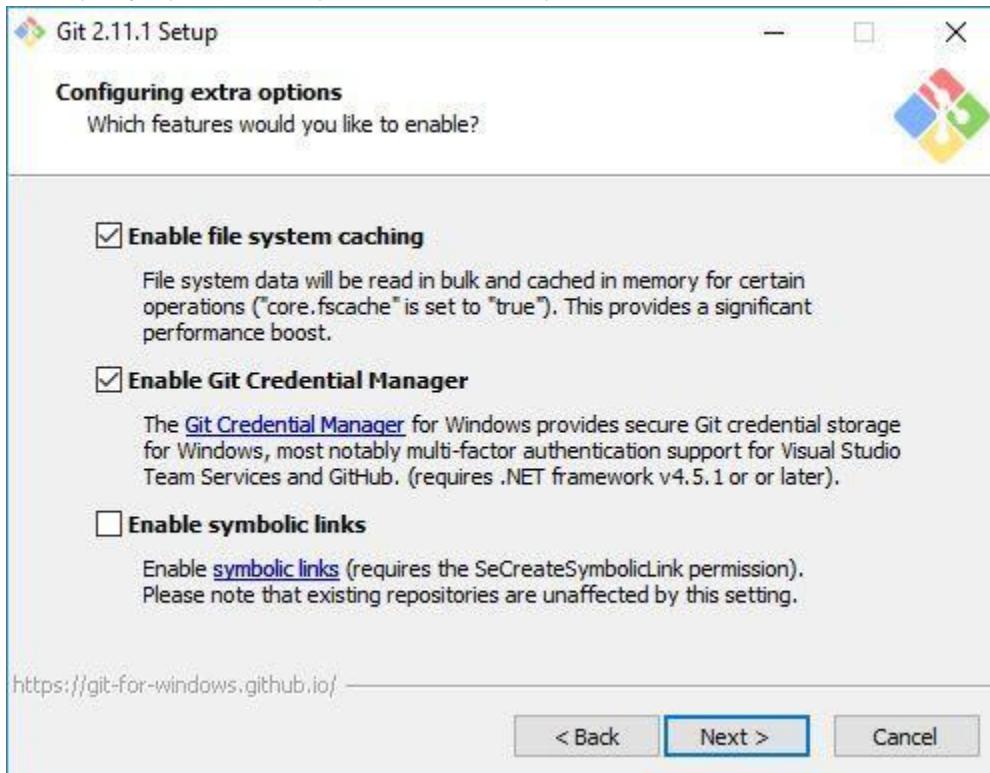
Selanjutnya konversi *line ending*. Biarkan saja seperti ini, kemudian klik *Next >*.



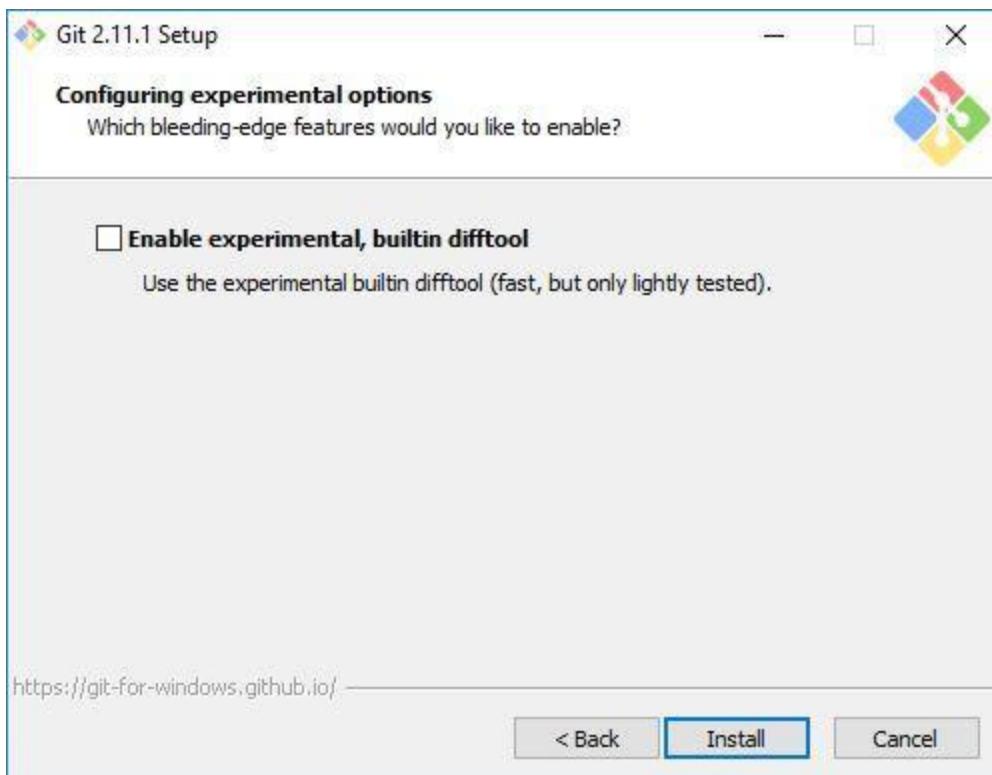
Selanjutnya pemilihan emulator terminal. Pilih saja yang bawah, kemudian klik *Next >*.



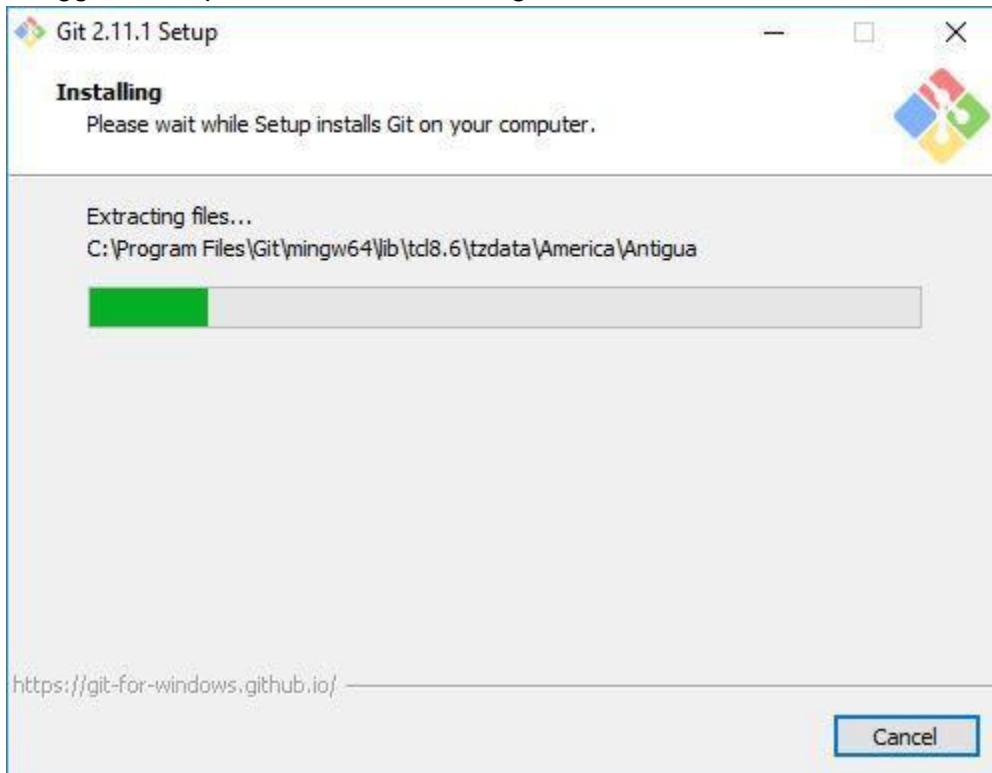
Selanjutnya pemilihan opsi ekstra. Klik saja *Next >*.



Selanjutnya pemilihan opsi eksperimental, langsung saja klik *Install* untuk memulai instalasi.



Tunggu beberapa saat, instalasi sedang dilakukan.



Setelah selesai, kita bisa langsung klik *Finish*.



Selamat, Git sudah terinstal di Windows. Untuk mencobanya, silahkan buka CMD atau PowerShell, kemudian ketik perintah git --version.

A screenshot of a Microsoft Windows Command Prompt window. The title bar says "Command Prompt". The window shows the following text:

```
c:\ Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Development>git --version
git version 2.11.1.windows.1

C:\Users\Development>
```

3. Konfigurasi Awal yang Harus Dilakukan

Ada beberapa konfigurasi yang harus dipersiapkan sebelum mulai menggunakan Git, seperti *name* dan *email*.

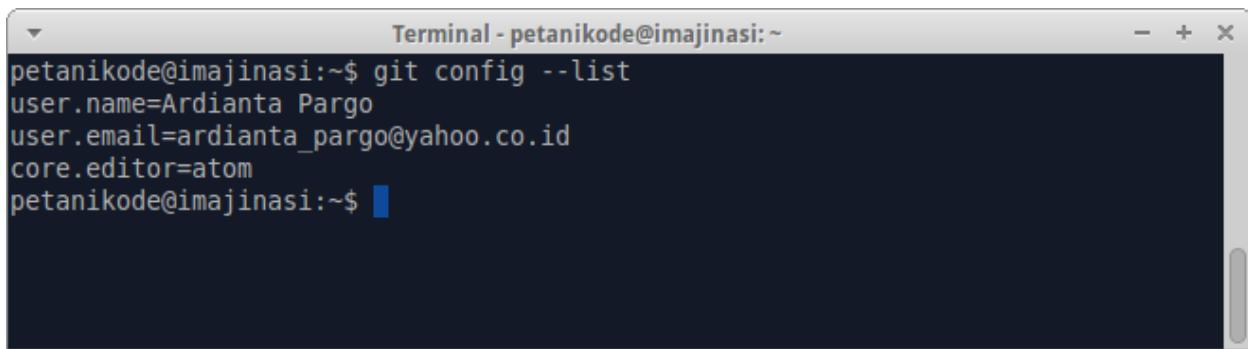
Silahkan lakukan konfigurasi dengan perintah berikut ini.

```
git config --global user.name "Petani Kode"
git config --global user.email contoh@petanikode.com
```

Kemudian periksa konfigurasinya dengan perintah:

```
git config --list
```

Apabila berhasil tampil seperti gambar berikut ini, berarti konfigurasi berhasil.



```
Terminal - petanikode@imajinasi:~  
petanikode@imajinasi:~$ git config --list  
user.name=Ardianta Pargo  
user.email=ardianta_pargo@yahoo.co.id  
core.editor=atom  
petanikode@imajinasi:~$
```

Konfigurasi `core.editor` bersifat opsional. Sedangkan `name` dan `email` wajib.

Jika kamu memiliki akun Github, Gitlab, Bitbucket atau yang lainnya...maka `username` dan `email` harus mengikuti akun tersebut agar mudah diintegrasikan.

Membuat Repositori

Pembuatan repositori dapat dilakukan dengan perintah `git init` nama-dir. Contoh:
`git init proyek-01`

Perintah tersebut akan membuat direktori bernama `proyek-01`. Kalau direktorinya sudah ada, maka Git akan melakukan inisialisasi di dalam direktori tersebut.

Perintah `git init` akan membuat sebuah direktori bernama `.git` di dalam proyek kita. Direktori ini digunakan Git sebagai database untuk menyimpan perubahan yang kita lakukan.

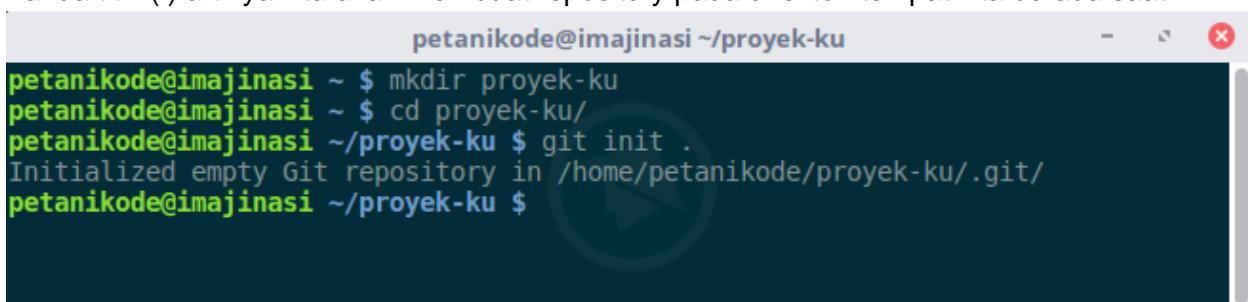
Hati-hati!

Kalau kita menghapus direktori ini, maka semua rekaman atau catatan yang dilakukan oleh Git akan hilang.

Contoh-contoh lain

Perintah berikut ini akan membuat repositori pada direktori saat ini (*working directory*).
`git init .`

Tanda titik(.) artinya kita akan membuat repository pada direktori tempat kita berada saat ini.



```
petanikode@imajinasi ~/proyek-ku  
petanikode@imajinasi ~ $ mkdir proyek-ku  
petanikode@imajinasi ~ $ cd proyek-ku/  
petanikode@imajinasi ~/proyek-ku $ git init .  
Initialized empty Git repository in /home/petanikode/proyek-ku/.git/  
petanikode@imajinasi ~/proyek-ku $
```

Perintah berikut ini akan membuat repositori pada direktori `/var/www/html/proyekweb/`.
`git init /var/www/html/proyekweb`

.gitignore

.gitignore merupakan sebuah file yang berisi daftar nama-nama file dan direktori yang akan diabaikan oleh Git.

Perubahan apapun yang kita lakukan terhadap file dan direktori yang sudah masuk ke dalam daftar .gitignore tidak akan dicatat oleh Git.

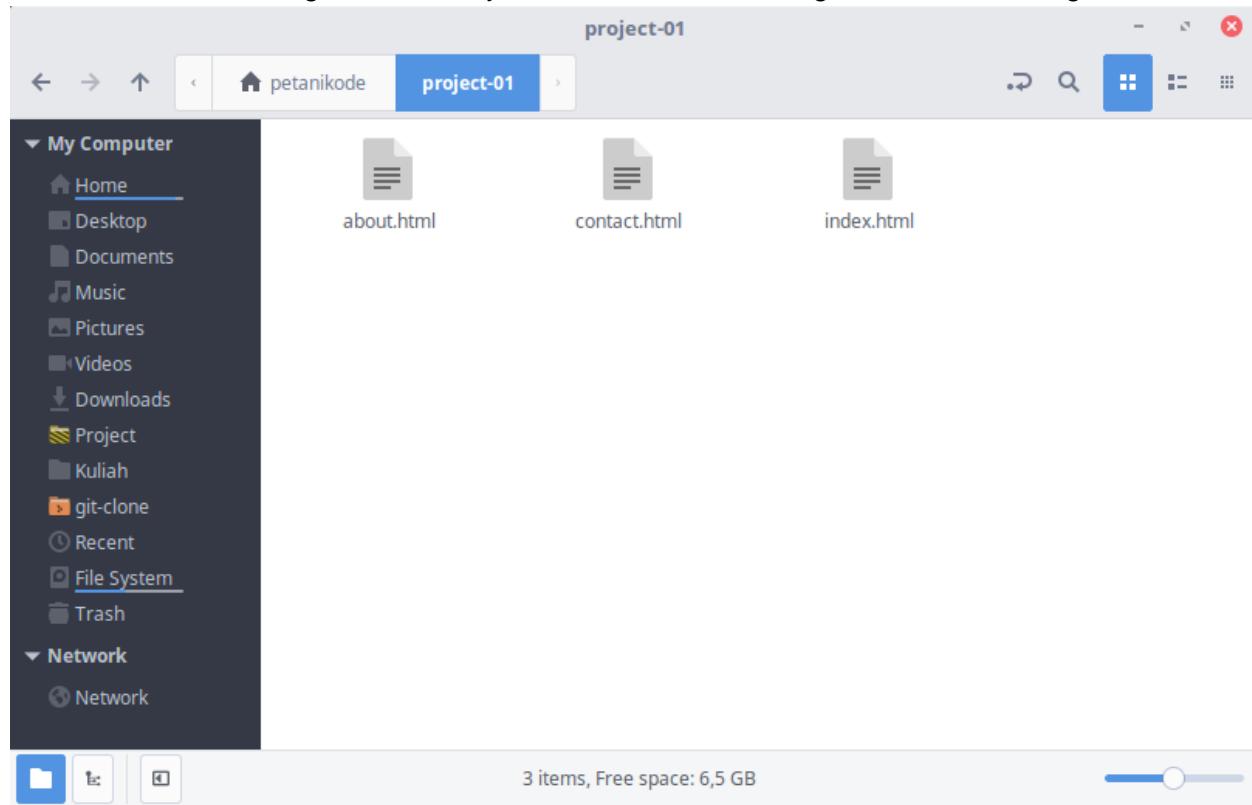
Cara menggunakan .gitignore, buat saja sebuah file bernama .gitignore dalam root direktori proyek/repositori.

```
File: .gitignore
/vendor/
/upload/
/cache
test.php
```

Pada contoh file .gitignore di atas, saya memasukan direktori vendor, upload, cache dan file test.php. File dan direktori tersebut akan diabaikan oleh Git. Pembuatan file .gitignore sebaiknya dilakukan di awal pembuatan repositori.

Simpan Perubahan Revisi dengan Git Commit

Kita sudah membuat repositori kosong. Belum ada apa-apa di sana. Sekarang coba tambahkan sebuah file baru. Sebagai contoh, saya akan menambahkan tiga file HTML kosong.



Setelah ditambahkan, coba ketik perintah git status untuk melihat status repositorinya.

```
petanikode@imajinasi ~/project-01
petanikode@imajinasi ~/project-01 $ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    about.html
    contact.html
    index.html

nothing added to commit but untracked files present (use "git add" to track)
petanikode@imajinasi ~/project-01 $
```

Berdasarkan keterangan di atas, saat ini kita berada cabang (*branch*) *master* dan ada tiga file yang belum ditambahkan ke Git.

Tiga Kelompok Kondisi File dalam Git

Sebelum kita membuat revisi, kita akan berkenalan dulu dengan tiga kondisi file dalam Git.

1. Modified

Modified adalah kondisi dimana revisi atau perubahan sudah dilakukan, tetapi belum ditandai dan belum disimpan di *version control*. Contohnya pada gambar di atas, ada tiga file HTML yang dalam kondisi *modified*.

2. Staged

Staged adalah kondisi dimana revisi sudah ditandai, tetapi belum disimpan di *version control*. Untuk mengubah kondisi file dari *modified* ke *staged* gunakan perintah `git add nama_file`.

Contoh:

```
git add index.html
```

3. Committed

Committed adalah kondisi dimana revisi sudah disimpan di *version control*. perintah untuk mengubah kondisi file dari *staged* ke *committed* adalah `git commit`.

Membuat Revisi Pertama

Baiklah, sekarang kita akan sudah tahu kondisi-kondisi file dalam Git. Selanjutnya, silahkan ubah kondisi tiga file HTML tadi menjadi *staged* dengan perintah `git add`.

```
git add index.html
git add about.html
```

```
git add contact.html
```

Atau kita bisa melakukannya seperti ini:

```
git add index.html about.html contact.html
```

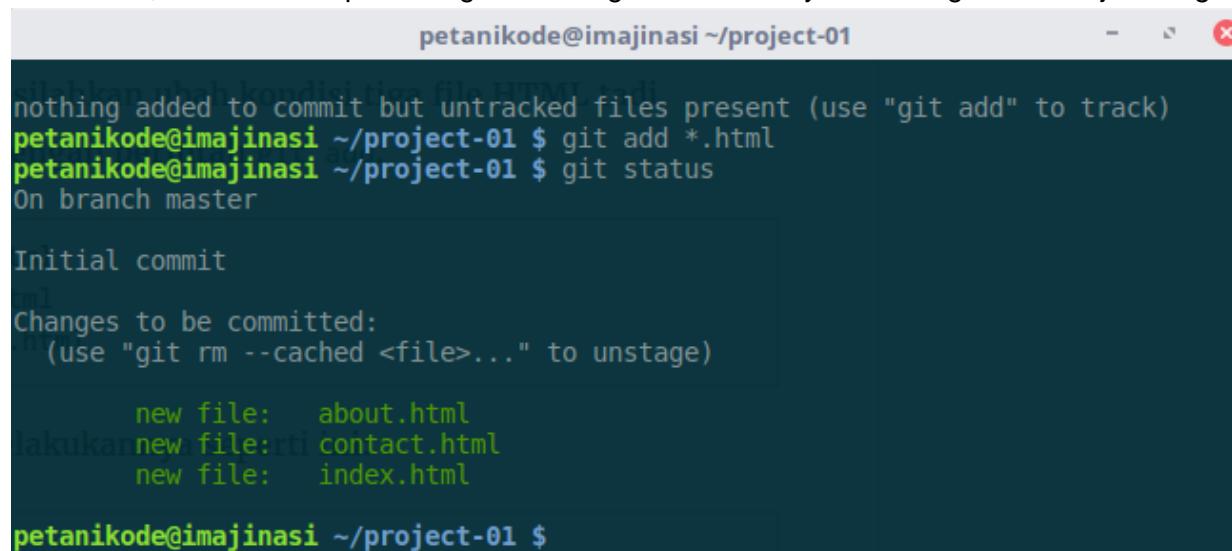
atau:

```
git add *.html
```

Atau seperti ini (semua file dan direktori):

```
git add .
```

Setelah itu, cobalah ketik perintah git status lagi. Kondisi filenya sekarang akan menjadi *staged*.



```
petanikode@imajinasi ~/project-01
nothing added to commit but untracked files present (use "git add" to track)
petanikode@imajinasi ~/project-01 $ git add *.html
petanikode@imajinasi ~/project-01 $ git status
On branch master
Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

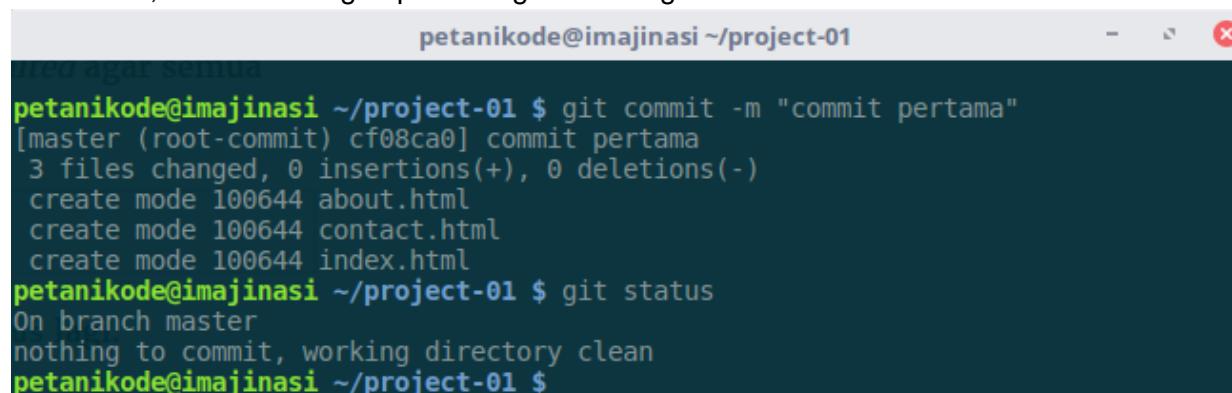
    new file:  about.html
    new file:  contact.html
    new file:  index.html

petanikode@imajinasi ~/project-01 $
```

Setelah itu, ubah kondisi file tersebut ke *committed* agar semua perubahan disimpan oleh Git.

```
git commit -m "Commit pertama"
```

Setelah itu, coba cek dengan perintah git status lagi.



```
petanikode@imajinasi ~/project-01
[commit pertama]
petanikode@imajinasi ~/project-01 $ git commit -m "commit pertama"
[master (root-commit) cf08ca0] commit pertama
 3 files changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 about.html
  create mode 100644 contact.html
  create mode 100644 index.html
petanikode@imajinasi ~/project-01 $ git status
On branch master
nothing to commit, working directory clean
petanikode@imajinasi ~/project-01 $
```

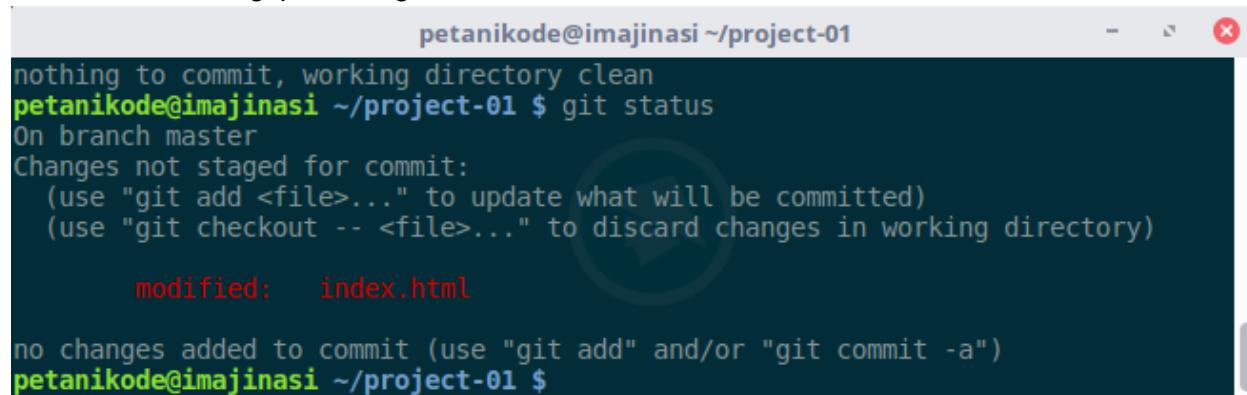
Selamat, revisi pertama sudah kita buat. Selanjutnya cobalah untuk membuat revisi kedua.

Membuat Revisi kedua

Ceritanya ada perubahan yang akan kita lakukan pada file index.html. Silahkan modifikasi isi file index.html. Sebagai contoh saya mengisinya seperti ini.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Belajar Git - Project 01</title>
  </head>
  <body>
    <p>Hello Semua, Saya sedang belajar Git</p>
  </body>
</html>
```

Setelah itu ketik lagi perintah git status.



A screenshot of a terminal window titled "petanikode@imajinasi ~/project-01". The window shows the command "git status" being run. The output indicates there are no changes to commit, the working directory is clean, and the user is on the "master" branch. It then lists a change: "modified: index.html". Below this, it says "no changes added to commit (use "git add" and/or "git commit -a")". The prompt ends with "\$".

```
petanikode@imajinasi ~/project-01
nothing to commit, working directory clean
petanikode@imajinasi ~/project-01 $ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
petanikode@imajinasi ~/project-01 $
```

Terlihat di sana, file index.html sudah dimodifikasi. Kondisinya sekarang berada dalam *modified*. Lakukan *commit* lagi seperti revisi pertama.

```
git add index.html
git commit -m "ditambahkan isi"
```

Dengan demikian, revisi kedua sudah disimpan oleh Git. Mungkin anda belum tahu maksud dari argumen *-m*, argumen tersebut untuk menambahkan pesan setiap menyimpan revisi.



Sekarang Git sudah mencatat dua revisi yang sudah kita lakukan. Kita bisa ibaratkan revisi-revisi ini sebagai *checkpoint* pada Game. Apabila nanti ada kesalahan, kita bisa kembali ke *checkpoint* ini.

Pertemuan 10 - Docker Images

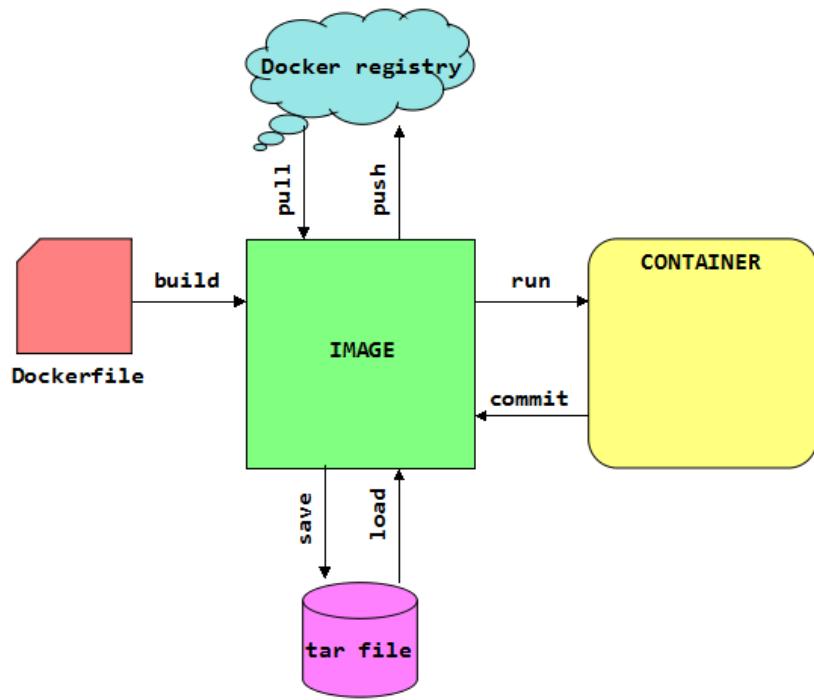
Tujuan

1. Mahasiswa dapat memahami cara kerja docker images
2. Mahasiswa dapat memaketkan aplikasi yang dibuat kedalam docker images

Teori

Docker adalah sebuah aplikasi yang bersifat open source yang berfungsi sebagai wadah/container untuk mengepak/memasukkan sebuah software secara lengkap beserta semua hal lainnya yang dibutuhkan oleh software tersebut dapat berfungsi. Pengaturan software beserta file/hal pendukung lainnya akan menjadi sebuah Image (istilah yang diberikan oleh docker). Kemudian sebuah instan dari Image tersebut kemudian disebut Container. Jika pernah belajar tentang OOP/Java – maka analogi hubungan Container dengan Image adalah seperti Object dengan Class-nya. Sehingga jika object adalah instance-of class maka container adalah instance-of image.

Dalam praktik dasarnya ada beberapa perintah Docker yang sering digunakan yaitu build, push, pull, run, commit. Perintah-perintah tersebut akan dibahas secara singkat dan akan dilengkapi kemudian saat diperlukan. Di bawah ini terdapat illustrasi sederhana dari docker workflow – alur kerja docker yang dasar.



Praktik

Menjalankan Image

Suatu container Docker diluncurkan dengan menjalankan image. Ada beberapa cara untuk menjalankan container yang mempengaruhi seberapa mudah pengelolaan semua container yang ada. Ketika container mulai berjalan, biasanya yang dijalankan adalah perintah yang ditetapkan di Dockerfile. Berikut adalah Dockerfile untuk container hello-world:

```

FROM
scratch
COPY hello
/
CMD
["/hello"]

```

Perintahnya sekedar menjalankan biner "hello" yang disalin ke root container ketika membuat image-nya.

Foreground versus Detached

Suatu container bisa berjalan di foreground yang dihambatnya sampai prosesnya keluar dan container berhenti berjalan. Di mode foreground, container mencetak keluarannya ke konsol dan bisa membaca masukan standar. Di mode detached (ketika Anda memberikan -d flag), kontrol akan seketika itu juga kembali dan containernya.

Menjalankan Container Tanpa Nama

Cara termudah menjalankan suatu container adalah: docker run <image id or name>. Ketika Anda menjalankan suatu container dengan perintah ini, Docker akan menetapkan suatu nama yang tersusun atas dua kata acak. Misalnya: docker run hello-world. Jika Anda sudah punya image hello-world, Docker akan menjalankannya. Jika belum, Docker akan menarik dari repositori resmi Docker di DockerHub lalu menjalankannya. Keluarannya akan terlihat sebagaimana berikut:

```
Hello from Docker!
```

```
This message shows that your installation appears to be working correctly.
```

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate work-flows, and more with a free Docker ID:

<https://cloud.docker.com/>

For more examples and ideas, visit:

<https://docs.docker.com/engine/userguide/>

Program hello keluar setelah menampilkan pesan, yang akan menghentikan proses yang berjalan di dalam container dan mengakhiri berjalannya container. Containernya masih tidak ke mana-mana jika Anda

masih ingin terhubung dengannya, memeriksa log, atau yang lain. Untuk melihat containernya, Anda bisa menjalankan perintah berikut:

```
docker ps -a --format "table {{.ID}}\t{{.Status}}\t{{.Names}}"
```

CONTAINER ID	STATUS	NAMES
8e2e491accb5	Exited (0) 2 minutes ago	clever_liskov

Akan saya jelaskan lebih lanjut tentang bagaimana senarai container dan semua pilihan terkait. Untuk saat ini, mari berfokus pada bagian Nama. Docker secara otomatis menghasilkan nama "clever_liskov", dan saya harus menggunakan atau menggunakan ID container untuk merujuk ke container ini untuk tujuan seperti memulai ulang, menghapusnya, atau mengeksekusi suatu perintah.

Menjalankan Container Bernama

Menggunakan ID container atau nama yang dihasilkan secara otomatis terkadang tidak nyaman. Jika Anda sering berinteraksi dengan suatu container yang sering Anda buat ulang, container itu akan mendapatkan ID dan nama otomatis yang berbeda. Namanya juga akan acak. Docker mengizinkan Anda menamai container-container tersebut apabila dijalankan dengan memberikan argumen perintah baris "--name". Dalam kasus sederhana, yang Anda hanya punya satu container untuk tiap imagnenya, Anda bisa menamai container dengan nama image: `docker run --name hello-world hello-world`.

Sekarang, jika kita lihat prosesnya (saya menghapus **clever_liskov** sebelumnya) kita akan melihat bahwa container itu bernama hello-world:

```
docker ps -a --format "table {{.ID}}\t{{.Names}}"
CONTAINER ID      NAMES
f6fe77b3b6e8      hello-world
```

Ada sejumlah manfaat container yang bernama:

- Anda punya nama yang stabil untuk container yang digunakan secara interaktif maupun dalam skrip.
- Anda bisa memilih nama yang bermakna.
- Anda bisa memilih nama yang pendek supaya nyaman ketika bekerja secara interaktif.
- Dengan melakukan itu Anda akan terhindar dari tidak sengaja memiliki banyak container untuk image yang sama (selama Anda selalu menyediakan nama yang sama).

Mari kita lihat pilihan terakhir. Jika saya coba menjalankan perintah yang sama dengan nama "hello-world" yang sama, saya akan mendapat pesan kesalahan yang jelas:

```
docker run --name hello-world hello-world
docker: Error response from daemon: Conflict. The container name
"/hello-world" is already in use by container
f6fe77b3b6e8e77ccf346c32c599e67b2982893ca39f0415472c2949cacc4a51.
You have to remove (or rename) that container to be able to reuse
that name.
See 'docker run --help'.
```

Menjalankan Auto-Remove Image

Secara default container selalu ada. Tapi terkadang ada yang tidak Anda butuhkan. Daripada menghapus secara manual container yang sudah keluar, Anda bisa membuatnya pergi dengan sendirinya. Argumen perintah baris `--rm` lah yang melakukannya: `docker run --rm hello-world`.

Menjalankan Perintah yang Berbeda

Secara default, Docker menjalankan perintah yang ditetapkan di Dockerfile untuk membuat image (atau langsung titik masuknya jika tidak ditemukan perintah apapun). Anda selalu bisa mengesampingkannya dengan memberikan perintah Anda sendiri di akhir perintah `run`. Mari menjalankan `ls -la` di image **busybox** (image **hello-world** tidak punya `ls` yang bisa dieksekusi):

```
docker run busybox ls -la
total 44
drwxr-xr-x  18 root  root      4096 Mar 18 17:06 .
drwxr-xr-x   18 root  root      4096 Mar 18 17:06 ..
-rwxr-xr-x  1 root   root       0 Mar 18 17:06 .dockerenv
drwxr-xr-x  2 root   root    12288 Mar  9 00:05 bin
drwxr-xr-x  5 root   root      340 Mar 18 17:06 dev
drwxr-xr-x  2 root   root      4096 Mar 18 17:06 etc
drwxr-xr-x  2 nobody nogroup  4096 Mar  9 00:05 home
dr-xr-xr-x   85 root  root       0 Mar 18 17:06 proc
drwxr-xr-x  2 root   root      4096 Mar  9 00:05 root
dr-xr-xr-x   13 root  root       0 Mar 18 17:06 sys
drwxrwxrwt  2 root   root      4096 Mar  9 00:05 tmp
drwxr-xr-x  3 root   root      4096 Mar  9 00:05 usr
drwxr-xr-x  4 root   root      4096 Mar  9 00:05 var
```

Berinteraksi dengan Host

Container Docker menjalankan berbagai proses yang terisolasi di dunianya sendiri. Tetapi seringkali penting dan bermanfaat untuk memberikan akses ke hostnya.

Memasukkan Variabel Environment ke suatu Container

Container Docker tidak bisa secara otomatis mewarisi environment proses host yang menjalankannya. Anda harus secara eksplisit menyediakan variabel-variabel environment ke container ketika Anda menjalankannya dengan tanda perintah baris -e. Anda bisa memasukkan berbagai variabel environment. Berikut adalah suatu contoh:

```
docker run --rm -it -e ENV_FROM_HOST="123" busybox
/ # env
HOSTNAME=8e7672bce5a7
SHLVL=1
HOME=/root
ENV_FROM_HOST=123
TERM=xterm
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
PWD=/
/ #
```

Baris pertama menjalankan container busybox, memasukkannya ke variabel **ENV_FROM_HOST** dan di dalam container yang menjalankan env menunjukkan bahwa **ENV_FROM_HOST** telah diatur dengan benar.

Anda bisa menggunakan variabel environment host juga. Ini akan mengatur sejumlah variabel environment host dan menggunakannya di perintah run:

```
$ export VAR_1=1
$ export VAR_2=2
$ docker run --rm -it -e VAR_1="$VAR_1" -e VAR_2="$VAR_2" busybox
```

Di dalam container, sekarang bisa dilihat:

```
/ # env | grep
VAR
VAR_1=1
VAR_2=2
```

Menaikkan Direktori Host

Salah satu interaksi paling bermanfaat adalah menaikkan direktori host. Ini akan memungkinkan beberapa kasus pemakaian yang menarik:

- Berbagi ruang penyimpanan antar container yang berjalan di host yang sama.
- Menampilkan dan menyunting file dengan environment dan tools host, serta menggunakan file yang ada di dalam container.
- Persistensi tingkat host yang melampaui masa aktif suatu container.

Berikut saya membuat satu file di host: \$ echo "Yeah, it works!" > ~/data/1.txt

Lalu saya menjalankan image **busybox** yang menaikkan **~/data directory** ke **/data** di dalam containernya dan menampilkan isi file di layar:

```
$ docker run --rm -v ~/data:/data busybox cat /data/1.txt  
Yeah, it works!
```

Di sini saya menggunakan perintah cat /data/1.txt.

Mengekspor Port ke Host

Jika Anda mengekspor suatu port di Dockerfile Anda dengan **EXPOSE**, maka hanya akan bisa diakses ke container docker lainnya. Untuk menjadikannya bisa diakses di host, Anda harus menggunakan argumen perintah baris -p. Sintaksnya adalah -p <host port>: <exposed container port>.

Berikut adalah menjalankan image **nginx**, yang mengekspor port 80 dan menggunakan argumen perintah baris -p untuk menjadikannya terlihat di host pada port 9000.

```
docker run --name nginx --rm -d -p 9000:80 nginx
```

Pertemuan 11 - Kubernetes

Tujuan

1. Mahasiswa dapat memahami cara kerja dari orkestrasi container kubernetes
2. Mahasiswa dapat membuat layanan yang memanfaatkan kubernetes

Teori

Kubernetes adalah sebuah cluster management open source yang di gunakan untuk mengelola docker. Aplikasi ini berasal dari aplikasi internal yang digunakan Google untuk mengelola cluster. Secara bisnis, Kubernetes adalah senjata andalan Google untuk mendongkrak peringkatnya di pasar cloud hosting atau google cloud platform. Kubernetes berfungsi sebagai mesin untuk menjadwalkan dan menjalankan docker pada server fisik atau virtual server. Kubernetes memberikan infrastruktur kontainer-sentris maksudnya semua aplikasi berjalan dalam kontainer atau docker.

Istilah—Istilah pada kubernetes:

1. Pod
Adalah grup container instance. Kita bisa menjalankan beberapa container (misalnya aplikasi web + Database) dalam satu pod. Antar container dalam satu pod bisa saling mengakses dengan menggunakan alamat localhost. Bisa dikatakan pod seperti laptop yang kita pakai coding.
2. Node
Adalah penyebutan dari mesin-mesin yang di gunakan. Mesin ini bisa saja mesin virtual (seperti VPS atau Instance) atau fisik.
3. Service
Merupakan mekanisme untuk mengekspos pod kita ke dunia luar. Aplikasi kita yang berjalan dalam pod tidak memiliki alamat IP yang tetap. Agar bisa diakses oleh aplikasi lain atau oleh user, kita perlu alamat IP yang tetap. Service menyediakan alamat IP yang tetap, yang nantinya akan kita arahkan ke pod kita dengan menggunakan selector.
4. Label
Adalah seperangkat informasi metadata untuk mencari pod tertentu.
5. App-Test
Kita buat label app yang isinya adalah nama aplikasi. Semua container, pod, dan service yang menjadi bagian dari aplikasi test kita beri label app=test.
6. Stage-production
Label stage bisa kita gunakan untuk menentukan berbagai konfigurasi environment deployment aplikasi kita, misalnya development, testing, performance test, security test, dan production
7. Jenis-frontend

kita bisa membuat label jenis aplikasi, misalnya frontend, cache, database, fileserver, dan sebagainya.

8. Selector

Adalah filtering menggunakan label. Misalnya kita ingin mencari semua instance database untuk aplikasi test yang berjalan di production.

Praktik

Instalasi minikube

Download dan **rename** menjadi **minikube.exe** :

<https://storage.googleapis.com/minikube/releases/v0.19.1/minikube-windows-amd64.exe>

Download command file kubectl :

<http://storage.googleapis.com/kubernetes-release/release/v1.6.3/bin/windows/amd64/kubectl.exe>

Setelah kedua file tersebut di download, pindahkan ke drive C:

Menjalankan Minikube

Jalankan powershell dengan mode administrator atau tekan ctrl+shift+enter setelah memilih powershell.

Cek Versi minikube :

```
PS C:\> .\minikube.exe version
minikube version: v0.19.1
```

Melihat ketersediaan kubernetes dan versinya :

```
PS C:\> .\minikube get-k8s-versions
The following Kubernetes versions are available:
- v1.7.0-alpha.2
- v1.6.4
- v1.6.3
- v1.6.0
- v1.6.0-rc.1
- v1.6.0-beta.4
- v1.6.0-beta.3
```

Menjalankan minikube dengan menggunakan versi kubernetes 1.6.3 dan berjalan pada virtualbox.

```
PS C:\> .\minikube start --kubernetes-version="v1.6.3" --vm-driver="virtualbox" --v=3 --alsologtostderr
```

Jelaskan proses apa yang terjadi pada saat perintah ini dijalankan ?

Melihat status minikube

```
PS C:\> .\minikube.exe status
minikube: Running
localkube: Running
```

Melihat informasi kluster kubernetes

```
PS C:\> .\kubectl.exe cluster-info
Kubernetes master is running at https://192.168.99.100:8443
```

Untuk proses debug dan melakukan diagnosis masalah pada cluster gunakan perintah 'kubectl cluster-info dump'.

Melihat versi kubectl

```
PS C:\> .\kubectl version
```

Melihat ip address untuk minikube

```
PS C:\> .\minikube.exe ip
192.168.99.100
```

Menjalankan dashboard interface untuk minikube

```
PS C:\> .\minikube.exe dashboard
```

Melihat informasi node yang berjalan

```
PS C:\> .\kubectl.exe get nodes
NAME      STATUS     AGE      VERSION
minikube  Ready      1h       v1.6.3
```

Menjalankan container hello-nginx dengan image nginx

```
PS C:\> .\kubectl.exe run hello-nginx --image=nginx --port=80
deployment "hello-nginx" created
```

Melihat kondisi pods (group container)

```
PS C:\> .\kubectl.exe get pods
NAME                  READY   STATUS        RESTARTS
AGE
hello-nginx-3322088713-170vl  0/1    ContainerCreating   0
13m
```

Melihat deskripsi pods

```

PS C:\> .\kubectl.exe describe pods hello-nginx-3322088713-170v1
Name:           hello-nginx-3322088713-170v1
Namespace:      default
Node:           minikube/192.168.99.100
Start Time:    Sat, 03 Jun 2017 05:40:10 +0700
Labels:         pod-template-hash=3322088713
                run=hello-nginx
Annotations:   kubernetes.io/created-
by={"kind":"SerializedReference","apiVersion":"v1","reference": {"kind"
:"ReplicaSet
","namespace":"default","name":"hello-nginx-
3322088713","uid":"6f847e67-47e4-11e7-9725-080027be4...
Status:        Running
IP:            172.17.0.4
Controllers:   ReplicaSet/hello-nginx-3322088713
Containers:
  hello-nginx:
    Container ID:
      docker://59a5071bba6d612b90d4e5aa1336d9634b651e382ac3957d204dae096c54b
      c27
      Image:          nginx
      Image ID:
        docker://sha256:958a7ae9e56979be256796dabd5845c704f784cd422734184999cf
        91f24c2547
      Port:          80/TCP
      State:         Running
      Started:      Sat, 03 Jun 2017 05:48:26 +0700
      Ready:         True
      Restart Count: 0
      Environment:  <none>
      Mounts:
        /var/run/secrets/kubernetes.io/serviceaccount from default-
        token-c3fdc (ro)

```

Expose container hello-nginx agar dapat diakses melalui node kubernetes

```

PS C:\> .\kubectl.exe expose deployment hello-nginx --type=NodePort
service "hello-nginx" exposed

```

Melihat service yang berjalan

```

PS C:\> .\kubectl.exe get services
NAME      CLUSTER-IP  EXTERNAL-IP  PORT(S)      AGE
hello-nginx  10.0.0.133  <nodes>      80:31115/TCP  41s
kubernetes  10.0.0.1     <none>       443/TCP      2h

```

Memberikan akses ke container agar dapat diakses

```
PS C:\> .\minikube.exe service --url=true hello-nginx
http://192.168.99.100:31115
```

Scaling container

```
PS C:\> .\kubectl.exe scale --replicas=3 deployment/hello-nginx
deployment "hello-nginx" scaled
```

Identifikasi hasil scaling container

```
PS C:\> .\kubectl.exe get deployment
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
hello-nginx  3          3          3           1          47m
```

Pertemuan 12 - Konsep IaaS dengan Aplikasi Virtualisasi

Tujuan

1. Mahasiswa dapat menggunakan teknologi virtualisasi dengan virtualbox
2. Mahasiswa dapat memahami dan mengimplementasikan network virtual dengan model sederhana

Teori

Virtualisasi adalah sebuah teknik yang saat ini banyak diterapkan untuk memenuhi kebutuhan TI yang semakin tinggi namun diikuti dengan tuntutan untuk mengefisiensikan biaya yang digunakan semaksimal mungkin. Virtualisasi adalah teknologi yang telah diterapkan secara luas saat ini dengan dampak peningkatan operasional dan finansial yang positif. Virtualisasi adalah konsep dimana akses ke sebuah hardware seperti server diatur sehingga beberapa operating system (guest operation system) dapat berbagi sebuah hardware. Tujuan dari virtualisasi adalah kinerja tingkat tinggi, ketersediaan, keandalan, ketangkasan, atau untuk membuat dasar keamanan dan manajemen yang terpadu.

Virtualisasi memungkinkan kita untuk berbagi hardware untuk digunakan beberapa sistem operasi. Virtualisasi dapat membuat sebuah tempat penyimpanan tunggal yang besar terlihat menjadi beberapa tempat penyimpanan dengan ukuran yang lebih kecil.

Setiap perusahaan memiliki tujuan yang berbeda mengapa menerapkan virtualisasi, salah satu tujuan yang ingin dicapai perusahaan dapat merupakan salah satu dari hal berikut :

- Memungkinkan semua device yang terhubung dengan jaringan untuk mengakses aplikasi melalui jaringan, bahkan jika aplikasi tidak pernah dirancangan untuk dapat bekerja di device tersebut.
- Isolasi beban perkerjaan atau aplikasi yang satu dengan yang lainnya untuk meningkatkan keamanan dan kemudahan pengelolaan lingkungan.
- Isolasi aplikasi dari sistem operasi, memungkinkan aplikasi untuk tetap berfungsi meskipun dirancang untuk sistem operasi dengan tipe yang berbeda
- Isolasi aplikasi dari sistem operasi, memungkinkan sebuah aplikasi untuk bekerja di sistem operasi yang asing
- Meningkatkan jumlah orang yang dapat didukung oleh aplikasi, dengan mengijinkan untuk menjalankan aplikasi dari mesin-mesin yang berbeda secara bersamaan
- Mengurangi waktu yang diperlukan untuk menjalankan aplikasi, dengan memisahkan data atau aplikasi itu sendiri dan menyebar pekerjaan di beberapa sistem
- Mengoptimalkan penggunaan sistem tunggal
- Meningkatkan keandalan atau ketersediaan dari aplikasi atau beban kerja dengan pengulangan

Cloud computing bisa dianggap sebagai perluasan dari virtualisasi. Perusahaan bisa menempatkan aplikasi atau sistem yang digunakan di internet, tidak mengelolanya secara

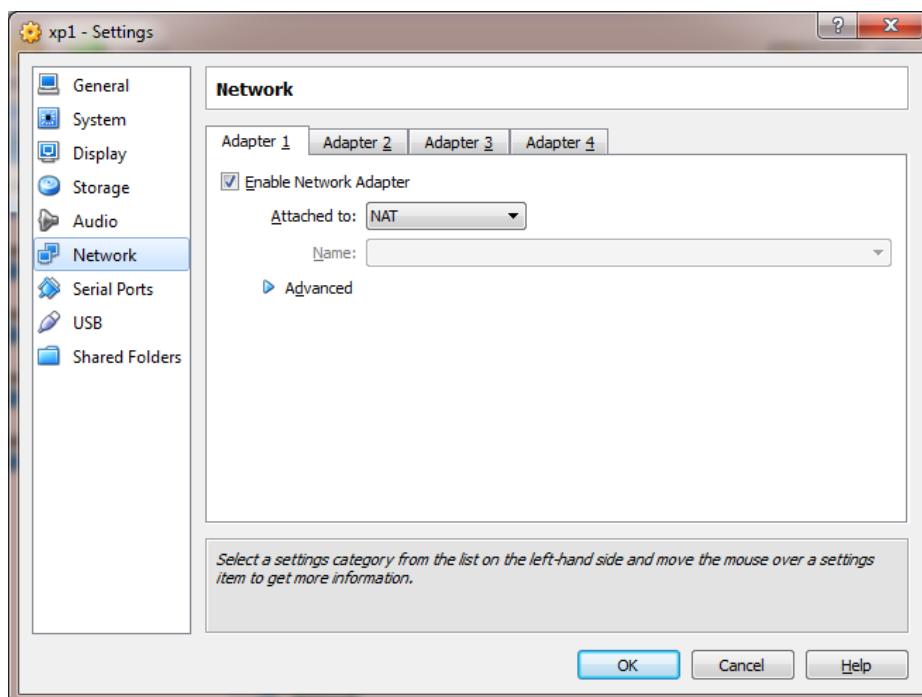
internal. Contoh cloud computing untuk versi public adalah layanan-layanan milik Google seperti Google Docs dan Google Spreadsheet. Adanya kedua layanan tersebut meniadakan kebutuhan suatu aplikasi office untuk pengolah kata dan aplikasi spreadsheet di internal perusahaan. Contoh cloud computing untuk keperluan non public adalah Amazon EC2 (Amazon Elastic Compute Cloud). Amazon menyediakan komputer induk, kita bisa mengirim dan menggunakan sistem virtual dan menggunakannya dalam jangka waktu dan biaya sewa tertentu.

Praktik

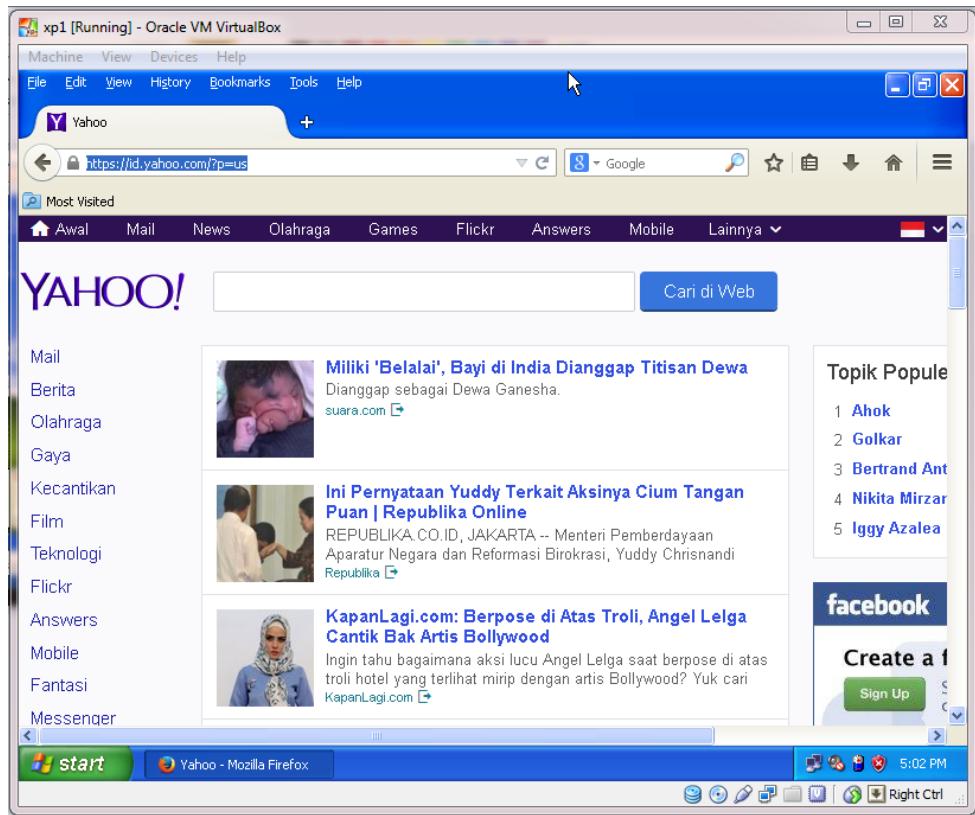
Dalam praktik ini menggunakan 3 Virtual machine yang masing-masing terinstall beberapa sistem operasi yaitu, Debian/Ubuntu, CentOS, Windows XP/Windows 7.

Mode jaringan Network Address Translation (NAT)

- Gunakan VM dengan OS Windows XP/Windows 7, VM harus dalam kondisi off, kemudian rubah konfigurasi jaringan pada VM menjadi **NAT**. Setelah konfigurasi selesai, tekan tombol OK, kemudian start VM. Catatan: Pastikan komputer sudah terkoneksi ke internet.

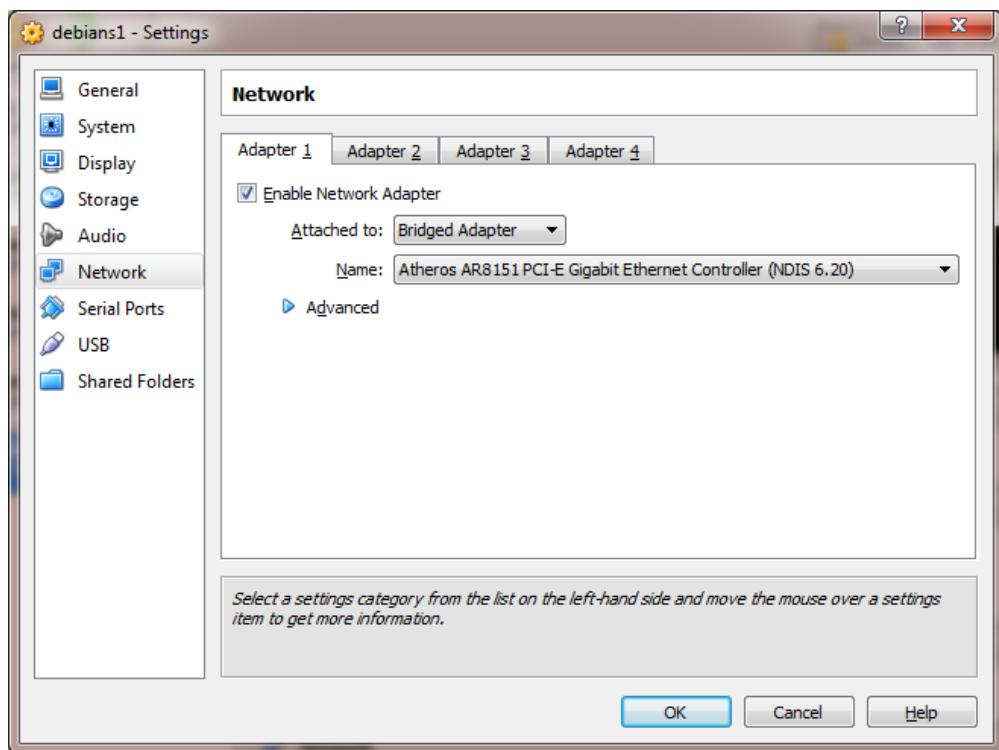


- Setelah masuk ke operating sistem, buka aplikasi browser dan mencoba untuk terkoneksi dengan internet. Jika bisa mengakses website maka konfigurasi jaringan berhasil. Setelah berhasil maka OS bisa dishutdown (untuk menghemat memory).



Mode jaringan Bridged networking

- Gunakan VM dengan OS Linux Debian/Ubuntu, VM harus dalam kondisi off, kemudian rubah konfigurasi jaringan pada VM menjadi **Bridged Adapter**. Kemudian pilih interface yang terhubung ke internet. Jika anda terhubung ke internet dengan menggunakan Ethernet maka pilih koneksi bridge dengan Ethernet adapter. Jika anda terhubung ke internet dengan menggunakan wireless maka pilih koneksi bridge dengan wireless adapter. Setelah konfigurasi selesai, tekan tombol OK, kemudian start VM. Catatan: Pastikan komputer sudah terkoneksi ke internet.



- Start Guest operating system. Login dengan menggunakan user: root
- Konfigurasi interface dengan DHCP. Dengan menggunakan perintah berikut

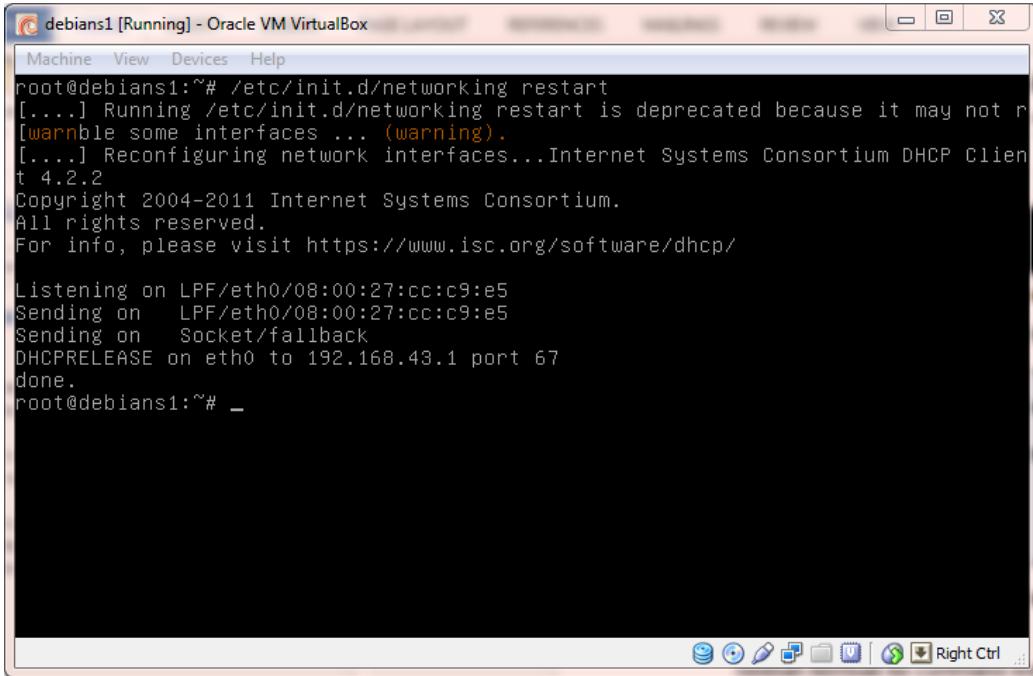
```
vi /etc/network/interfaces
```

Kemudian edit/modifikasi kontennya dengan menekan tombol [Insert], menjadi seperti berikut:

```
auto eth0
iface eth0 inet dhcp
```

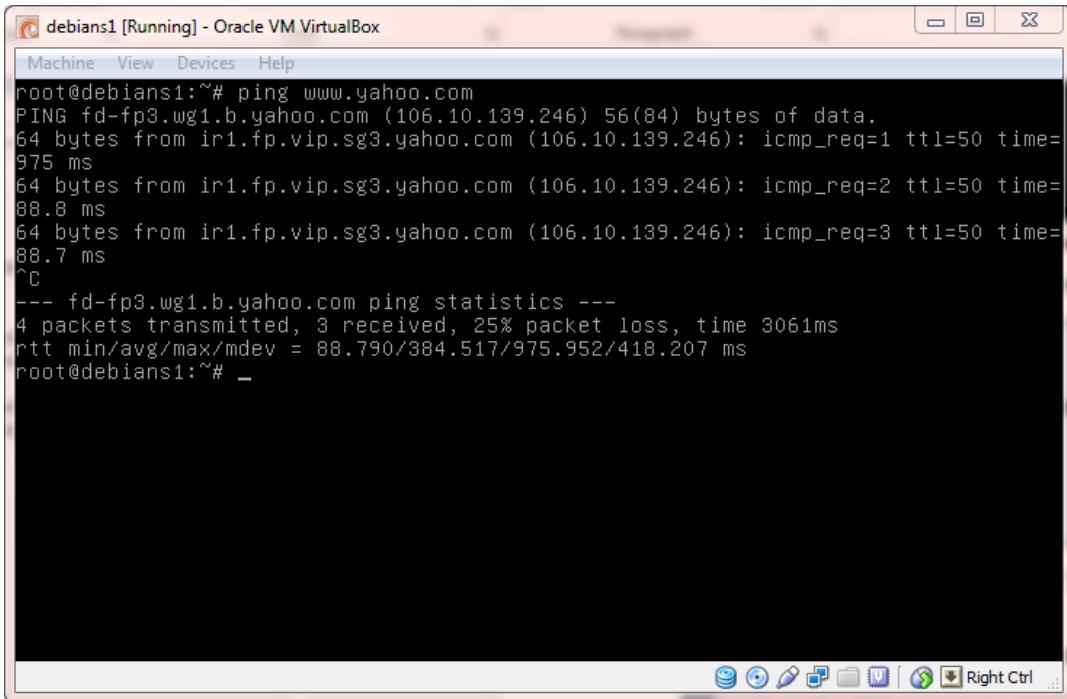
Setelah selesai mengedit file “interfaces”, tekan tombol “[esc]” diikuti dgn tombol “[:][w][q][!]”

Setelah kembali ke command line, restart interface dengan menggunakan perintah berikut.



```
Machine View Devices Help
root@debians1:~# /etc/init.d/networking restart
[....] Running /etc/init.d/networking restart is deprecated because it may not r
[warnble some interfaces ... (warning).
[....] Reconfiguring network interfaces...Internet Systems Consortium DHCP Client
t 4.2.2
Copyright 2004-2011 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Listening on LPF/eth0/08:00:27:cc:c9:e5
Sending on LPF/eth0/08:00:27:cc:c9:e5
Sending on Socket/fallback
DHCPRELEASE on eth0 to 192.168.43.1 port 67
done.
root@debians1:~# _
```

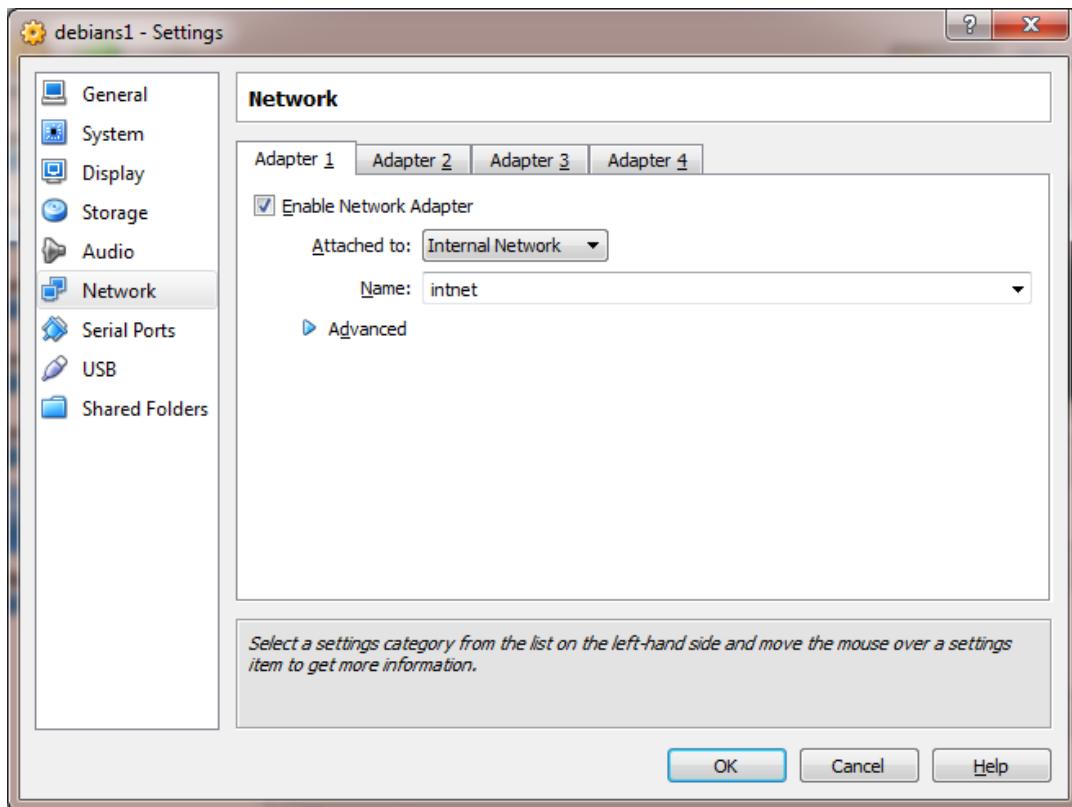
- Tahapan berikutnya yaitu mengecek koneksi ke internet, dengan menggunakan perintah “ping”. Apabila koneksi berhasil dan mendapatkan replay, maka konfigurasi telah berhasil dan berjalan dengan baik. Untuk menghentikan proses replay, tekan **ctrl + c**. Setelah berhasil maka OS bisa dishutdown (untuk menghemat memory).



```
Machine View Devices Help
root@debians1:~# ping www.yahoo.com
PING fd-fp3.wg1.b.yahoo.com (106.10.139.246) 56(84) bytes of data.
64 bytes from ir1.fp.vip.sg3.yahoo.com (106.10.139.246): icmp_req=1 ttl=50 time=
975 ms
64 bytes from ir1.fp.vip.sg3.yahoo.com (106.10.139.246): icmp_req=2 ttl=50 time=
88.8 ms
64 bytes from ir1.fp.vip.sg3.yahoo.com (106.10.139.246): icmp_req=3 ttl=50 time=
88.7 ms
^C
--- fd-fp3.wg1.b.yahoo.com ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3061ms
rtt min/avg/max/mdev = 88.790/384.517/975.952/418.207 ms
root@debians1:~# _
```

Mode jaringan Internal networking

- Gunakan dua VM dengan OS Linux Debian/Ubuntu dan OS CentOS, VM harus dalam kondisi off, kemudian rubah konfigurasi jaringan pada dua VM tersebut menjadi **Internal Network**. Setelah konfigurasi selesai, klik tombol OK, kemudian start VM.



- Login ke masing-masing OS

```

debians1 [Running] - Oracle VM VirtualBox
Machine View Devices Help

Debian GNU/Linux 7 debians1 tty1
debians1 login: root
Password:
Last login: Wed Apr 1 00:41:45 EDT 2015 on tty1
Linux debians1 3.2.0-4-686-pae #1 SMP Debian 3.2.65-1+deb7u2 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@debians1:~# 

```

- Konfigurasi alamat IP secara static dari interface dari masing-masing OS. Untuk OS debian/ubuntu, gunakan IP: 192.168.101.25, dengan menggunakan perintah berikut:

```
# vim /etc/network/interfaces
```

Edit pada bagian berikut dengan menekan tombol [Insert].

```

auto eth0
iface eth0 inet static
address 192.168.101.25
netmask 255.255.255.0
gateway 192.168.101.1

```

- Setelah selesai mengedit tekan tombol [esc] diikuti “:wq!” Kemudian Restart interface dengan menggunakan perintah:

```
# /etc/init.d/networking restart
```

- Untuk OS CentOS, gunakan IP: 192.168.101.26, dengan menggunakan perintah berikut:

```
# vim /etc/sysconfig/network-scripts/ifcfg-eth0
```

Edit pada bagian berikut dengan menekan tombol [Insert].

```

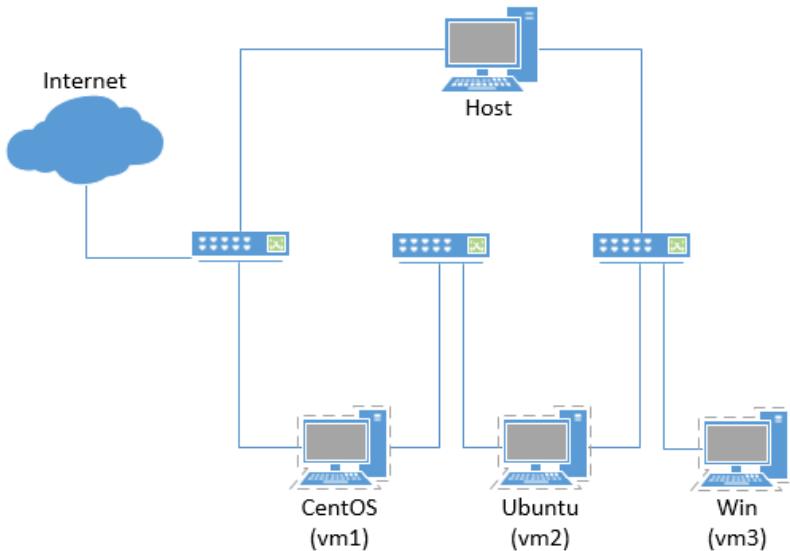
DEVICE=eth0
TYPE=Ethernet
BOOTPROTO=static
ONBOOT=yes
NETWORK=192.168.101.0
IPADDR=192.168.101.26
NETMASK=255.255.255.0

```

- Setelah selesai mengedit tekan tombol [esc] diikuti “:wq!” Kemudian Restart interface dengan menggunakan perintah:
`# service network restart`
- Mengecek IP pada masing OS dengan perintah ‘ifconfig’, kemudian melakukan ping ke OS yang lain. Jika setelah menjalankan perintah ping mendapatkan respon reply, maka koneksi dan konfigurasi berhasil.
- Menjalankan perintah ping dari OS debian ke OS CentOS. Lakukan sebaliknya untuk membutikan koneksi jaringan.
- Setelah selesai melakukan testing dan berhasil, maka shutdown masing-masing OS tersebut dengan menggunakan perintah
`# shutdown -h now`

Tugas

Buatlah topologi jaringan seperti berikut, catatan: IP bebas



Pertemuan 13 - Pengantar MbaaS pada platform firebase

Tujuan

1. Mahasiswa dapat mengenal cara kerja platform Firebase
2. Mahasiswa dapat implementasi function as a service di platform firebase

Teori

Firebase Realtime Database adalah database yang di-host di cloud. Data disimpan sebagai JSON dan disinkronkan secara realtime ke setiap klien yang terhubung. Ketika Anda membuat aplikasi lintas-platform dengan SDK Android, iOS, dan JavaScript, semua klien akan berbagi sebuah instance Realtime Database dan menerima update data terbaru secara otomatis.

Kemampuan utama

Realtime	Sebagai ganti permintaan HTTP biasa, Firebase Realtime Database menggunakan sinkronisasi data—setiap kali data berubah, semua perangkat yang terhubung akan menerima update dalam waktu milidetik. Memberikan pengalaman yang kolaboratif dan imersif tanpa perlu memikirkan kode jaringan.
Offline	Aplikasi Firebase tetap responsif bahkan saat offline karena SDK Firebase Realtime Database menyimpan data ke disk. Setelah koneksi pulih, perangkat klien akan menerima setiap perubahan yang terlewat dan melakukan sinkronisasi dengan status server saat ini.
Dapat Diakses dari Perangkat Klien	Firebase Realtime Database dapat diakses secara langsung dari perangkat seluler atau browser web; server aplikasi tidak diperlukan. Keamanan dan validasi data dapat diakses melalui Aturan Keamanan Firebase Realtime Database yang merupakan kumpulan aturan berbasis ekspresi dan dijalankan ketika data dibaca atau ditulis.

Menskalakan di beberapa database	Dengan Firebase Realtime Database pada paket harga Blaze, Anda dapat mendukung kebutuhan data aplikasi Anda pada skala tertentu dengan membagi data Anda di beberapa instance database di project Firebase yang sama. Menyederhanakan autentikasi dengan Firebase Authentication pada project Anda dan mengautentikasi pengguna di instance database Anda. Mengontrol akses ke data di tiap database dengan Aturan Firebase Realtime Database khusus untuk tiap instance database.
----------------------------------	--

Bagaimana cara kerjanya?

Firebase Realtime Database memungkinkan Anda untuk membuat aplikasi kolaboratif dan kaya fitur dengan menyediakan akses yang aman ke database, langsung dari kode sisi klien. Data disimpan di drive lokal. Bahkan saat offline sekalipun, peristiwa realtime terus berlangsung, sehingga pengguna akhir akan merasakan pengalaman yang responsif. Ketika koneksi perangkat pulih kembali, Realtime Database akan menyinkronkan perubahan data lokal dengan update jarak jauh yang terjadi selama klien offline, sehingga setiap perbedaan akan otomatis digabungkan.

Realtime Database menyediakan bahasa aturan berbasis ekspresi yang fleksibel, atau disebut juga Aturan Keamanan Firebase Realtime Database, untuk menentukan metode strukturisasi data dan kapan data dapat dibaca atau ditulis. Ketika diintegrasikan dengan Firebase Authentication, developer dapat menentukan siapa yang memiliki akses ke data tertentu dan bagaimana mereka dapat mengaksesnya.

Realtime Database adalah database NoSQL, sehingga memiliki pengoptimalan dan fungsionalitas yang berbeda dengan database terkait. API Realtime Database dirancang agar hanya mengizinkan operasi yang dapat dijalankan dengan cepat. Hal ini memungkinkan Anda untuk membangun pengalaman realtime yang luar biasa dan dapat melayani jutaan pengguna tanpa mengorbankan kemampuan respons. Oleh karena itu, perlu dipikirkan bagaimana pengguna mengakses data, kemudian [buat struktur data sesuai dengan kebutuhan tersebut](#).

Praktik

Firebase CLI

Memulainya kita ada bantuan yang [Firebase](#) sediakan untuk ini. Kita perlu [Node.js](#) dengan [NPM](#). Dan kita pasang menggunakan perintah npm install -g firebase-tools

```
c:\Users\dede>npm install -g firebase-tools
[.....] - fetchMetadata: sill mapToRegistry uri https://registry.npmjs.org/uuid
```

Saat ini versinya seperti pada gambar dibawah.

```
D:\app\firebasehost>firebase --version  
3.4.0  
D:\app\firebasehost>
```

Firebase Login

Kita perlu *login* ke [Google](#) lewat perintah firebase Login. Ia menggunakan *login* kita yang ada di *browser chrome*, jadi sebaiknya *login* juga disana. Ini dilakukan sekali saja, karena *token auth* biasanya disimpan.

```
D:\app\firebasehost>firebase login  
? Allow Firebase to collect anonymous CLI usage information? Yes  
  
Visit this URL on any device to log in:  
https://accounts.google.com/o/oauth2/auth?client_id=[REDACTED]-fg  
enid%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudplatformproje  
_type=code&state=246912241&redirect_uri=http%3A%2F%2Flocalhost%3A96  
  
Waiting for authentication...  
  
+ Success! Logged in as [REDACTED]@gmail.com  
D:\app\firebasehost>
```

Firebase Init

Untuk memulainya kita buat dan masuk ke *folder* tempat kita akan menaruh *file website*-nya. Dari situ kita jalankan perintah firebase init.

```
D:\app\firebasehost>firebase init  
  
##### ##### ##### ##### ##### ##### ##### #####  
## ## ## ## ## ## ## ## ## ## ## ## ## ##  
##### ##### ##### ##### ##### ##### ##### #####  
## ## ## ## ## ## ## ## ## ## ## ## ## ##  
## ## ## ## ## ## ## ## ## ## ## ## ## ##  
  
You're about to initialize a Firebase project in this directory:  
  
D:\app\firebasehost  
  
? Are you ready to proceed? (Y/n)
```

Setelah pilihan untuk melanjutkan, kita akan memilih Hosting: Configure and deploy Firebase Hosting sites. Karena kita memang menggunakan ini untuk *hosting*.

```
D:\app\firebasehost>firebase init

#####
##      ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##      ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
#####  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##      ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##      ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##

You're about to initialize a Firebase project in this directory:

D:\app\firebasehost

? Are you ready to proceed? Yes
? What Firebase CLI features do you want to setup for this folder?
  (*) Database: Deploy Firebase Realtime Database Rules
>(*) Hosting: Configure and deploy Firebase Hosting sites
```

Karena kita memulai tanpa membuat projek dahulu di-[Firebase Console](#) maka kita pilih Create a new project. Kalau kita sudah membuat project di-[Firebase Console](#) maka projek itu akan muncul disini.

```
D:\app\firebasehost>firebase init

#####
##      ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##      ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
#####  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##      ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##      ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##

You're about to initialize a Firebase project in this directory:

D:\app\firebasehost

? Are you ready to proceed? Yes
? What Firebase CLI features do you want to setup for this folder? Database: Deploy Firebase Realtime Database Rules
, Hosting: Configure and deploy Firebase Hosting sites

--- Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? What Firebase project do you want to associate as default?
 [don't setup a default project]
> [create a new project]
```

Pilihan ini kita cukup melewatkannya saja dengan menekan tombol *enter*. Yang terjadi [Firebase](#) akan membuatkan kita file database.rules.json, ini bisa dikonfigurasi kemudian hari.

```
D:\app\firebasehost>firebase init

#####
##  ##  #####  #####  #####  ##  #####  #####
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
#####  ##  #####  #####  #####  ##  #####  #####
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##  ######  ##  #####  #####  ##  #####  #####
You're about to initialize a Firebase project in this directory:

D:\app\firebasehost

? Are you ready to proceed? Yes
? What Firebase CLI features do you want to setup for this folder? Database: Deploy Firebase Realtime Database Rules, Hosting: Configure and deploy Firebase Hosting sites

== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? What Firebase project do you want to associate as default? [create a new project]

== Database Setup

Firebase Realtime Database Rules allow you to define how your data should be
structured and when your data can be read from and written to.

? What file should be used for Database Rules? (database.rules.json)
```

Selanjutnya kita akan ditanyakan *folder* mana yang akan digunakan untuk *file* kita. Secara default nama *folder*-nya public.

```
D:\app\firebasehost>firebase init

#####
##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##  ##  ##  ##  ##  ##  ##  ##  ##  ##
#####  ##  #####  ##  #####  ##  #####  ##  #####
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##

You're about to initialize a Firebase project in this directory:

D:\app\firebasehost

? Are you ready to proceed? Yes
? What Firebase CLI features do you want to setup for this folder? Database: Deploy Firebase Realtime Database Rules
, Hosting: Configure and deploy Firebase Hosting sites

== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? What Firebase project do you want to associate as default? [create a new project]

== Database Setup

Firebase Realtime Database Rules allow you to define how your data should be
structured and when your data can be read from and written to.

? What file should be used for Database Rules? database.rules.json
+ Database Rules for undefined have been downloaded to database.rules.json.
Future modifications to database.rules.json will update Database Rules when you run
firebase deploy.

== Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? (public)
```

Karena di tulisan ini kita tidak membuat *web* dengan [javascript framework](#) maka kita pilih N. Konfigurasi ini bisa di atur secara manual, seperti halnya konfigurasi untuk *web-server rewrite*.

```
D:\app\firebasehost>firebase init

#####
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  #
#####  ##  #####  ##  #####  ##  #####  ##  #####
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  #
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##

You're about to initialize a Firebase project in this directory:

D:\app\firebasehost

? Are you ready to proceed? Yes
? What Firebase CLI features do you want to setup for this folder? Database: Deploy Firebase Realtime Database Rules, Hosting: Configure and deploy Firebase Hosting sites

== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? What Firebase project do you want to associate as default? \[create a new project\]

== Database Setup

Firebase Realtime Database Rules allow you to define how your data should be
structured and when your data can be read from and written to.

? What file should be used for Database Rules? database.rules.json
+ Database Rules for undefined have been downloaded to database.rules.json.
Future modifications to database.rules.json will update Database Rules when you run
firebase deploy.

== Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? public
? Configure as a single-page app (rewrite all urls to /index.html)? \(y/N\)
```

Selesai sudah proses ini. Langkah selanjutnya kita perlu ke [Firebase Console](#)-nya.

```
D:\app\firebasehost>firebase init

#####
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##

You're about to initialize a Firebase project in this directory:

D:\app\firebasehost

? Are you ready to proceed? Yes
? What Firebase CLI features do you want to setup for this folder? Database: Deploy Firebase Realtime Database Rules
, Hosting: Configure and deploy Firebase Hosting sites

== Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? What Firebase project do you want to associate as default? [create a new project]

== Database Setup

Firebase Realtime Database Rules allow you to define how your data should be
structured and when your data can be read from and written to.

? What file should be used for Database Rules? database.rules.json
+ Database Rules for undefined have been downloaded to database.rules.json.
Future modifications to database.rules.json will update Database Rules when you run
firebase deploy.

== Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? public
? Configure as a single-page app (rewrite all urls to /index.html)? No
+ Wrote public/404.html
+ Wrote public/index.html

i Writing configuration info to firebase.json...
i Writing project information to .firebaserc...

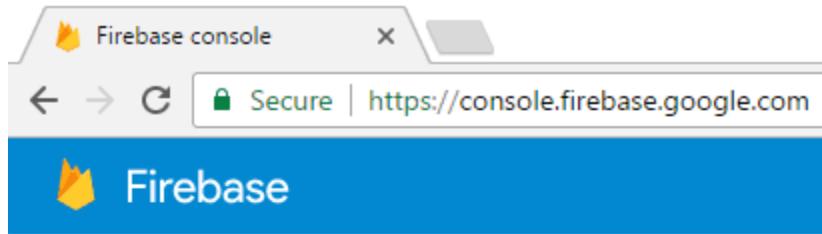
+ Firebase initialization complete!

Project creation is only available from the Firebase Console
Please visit https://console.firebaseio.google.com to create a new project, then run firebase use --add

D:\app\firebasehost>
```

Console Firebase

Tampaknya firebase-tools belum bisa membuatkan project di-[Firebase Console](#). Jadi kita perlu ke sana. Dan membuat project baru.



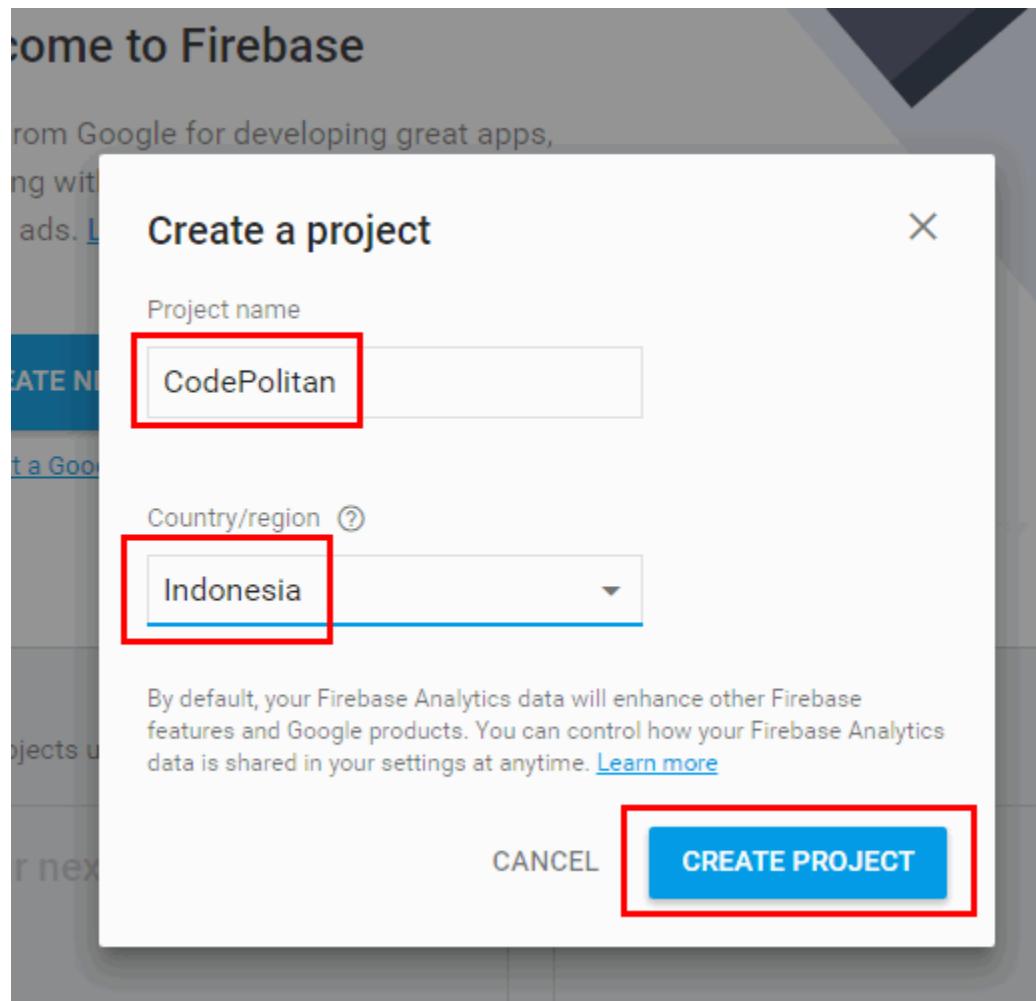
Welcome to Firebase

Tools from Google for developing great apps,
engaging with your users, and earning more through
mobile ads. [Learn more](#)

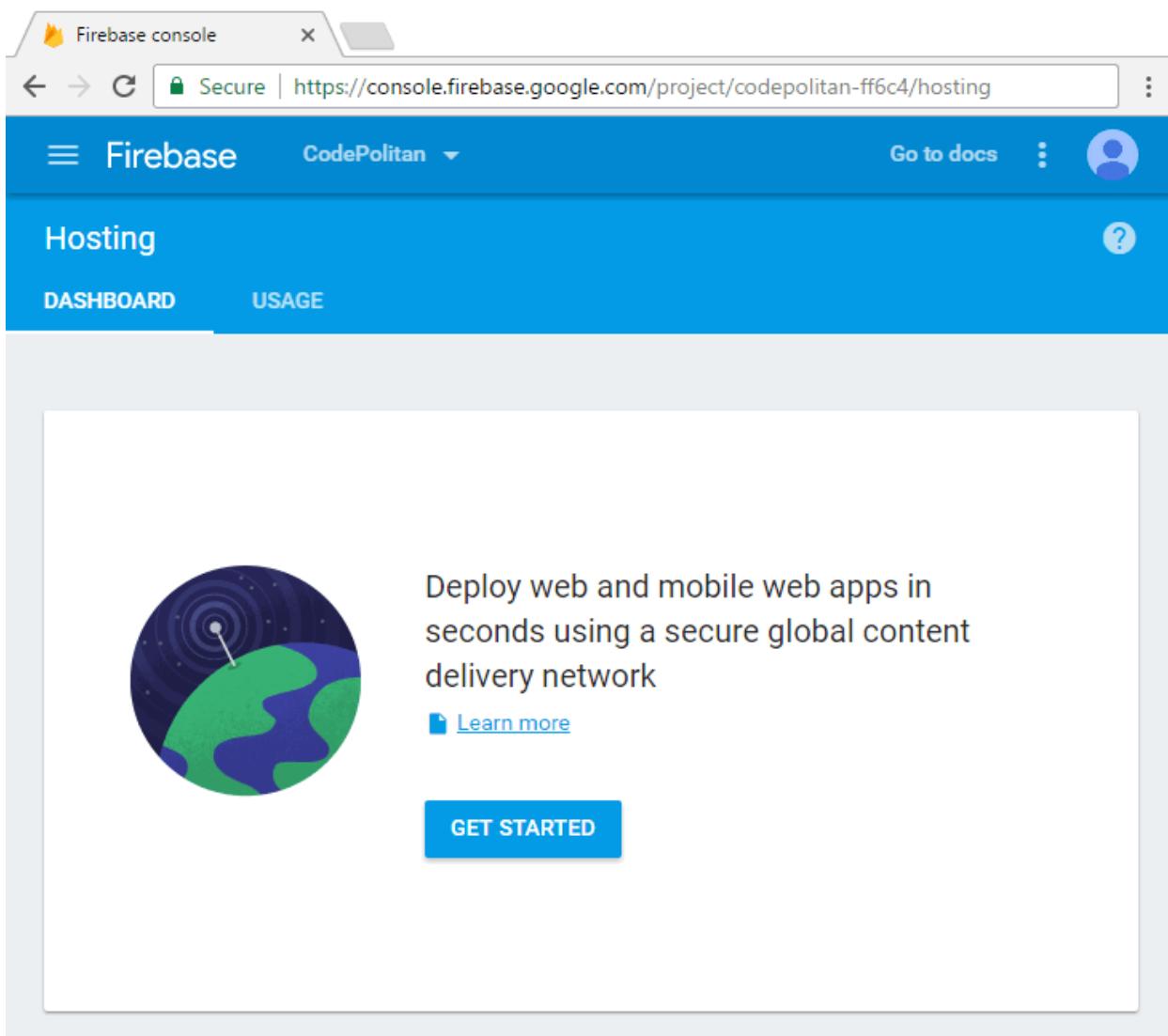
[CREATE NEW PROJECT](#)

[or import a Google project](#)

Untuk tulisan ini kita buat projek dengan nama CodePolitan dinegara Indonesia.



Kalau kita pilih menu *hosting* maka kita akan masuk ke *dashboard*, terlihat seperti dibawah. Dari sini kita tidak perlu melakukan apa-apa lagi disini.



Firebase Use

Projeknya sudah siap. Sekarang saatnya kita akan menyambungkan dengan yang kita sudah *init* tadi. Gunakan perintah `firebase use --add`, maka akan muncul projek yang kita buat tadi di [Firebase Console](#) dalam hal ini codepolitan-50731.

```
D:\app\firebasehost>firebase use --add  
? Which project do you want to add?  
> codepolitan-50731
```

Ada pertanyaan soal *alias*, cukup kita masukkan codepolitan biar sederhana.

```
D:\app\firebasehost>firebase use --add  
? Which project do you want to add? codepolitan-50731  
? What alias do you want to use for this project? (e.g. staging) codepolitan
```

Setelah itu projek kita diselaraskan dan kita konfigurasi selesai.

```
D:\app\firebasehost>firebase use --add
? Which project do you want to add? codepolitan-50731
? What alias do you want to use for this project? (e.g. staging) codepolitan

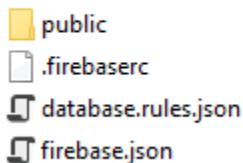
Created alias codepolitan for codepolitan-50731.
Now using alias codepolitan (codepolitan-50731)

D:\app\firebasehost>
```

Firebase Deploy

Pada saat kita menginisiasi *folder* tadi. Proses tersebut akan membuat beberapa *file*. Ini termasuk *file* [HTML](#) dalam *folder public* dan beberapa [JSON](#) *file* (ini tempat konfigurasinya).

▶ This PC ▶ Data (D:) ▶ app ▶ firebasehost



Sekarang bukan jaman pakai [FTP](#) lagi, jadi kita akan gunakan perintah firebase deploy untuk me-copy *file* di komputer kita ke server. Selesai itu kita akan melihat nama *sub-domain* dari [website](#) kita itu.

```
D:\app\firebasehost>firebase deploy
=== Deploying to 'codepolitan-50731'...

i  deploying database, hosting
+  database: rules ready to deploy.
i  hosting: preparing public directory for upload...
Uploading: [=====] 56%+  hosting: public folder uploaded successfully
+  hosting: 5 files uploaded successfully
i  starting release process (may take several minutes)...

+ Deploy complete!

Project Console: https://console.firebaseio.google.com/project/codepolitan-50731/overview
Hosting URL: https://codepolitan-50731.firebaseio.com

D:\app\firebasehost>
```

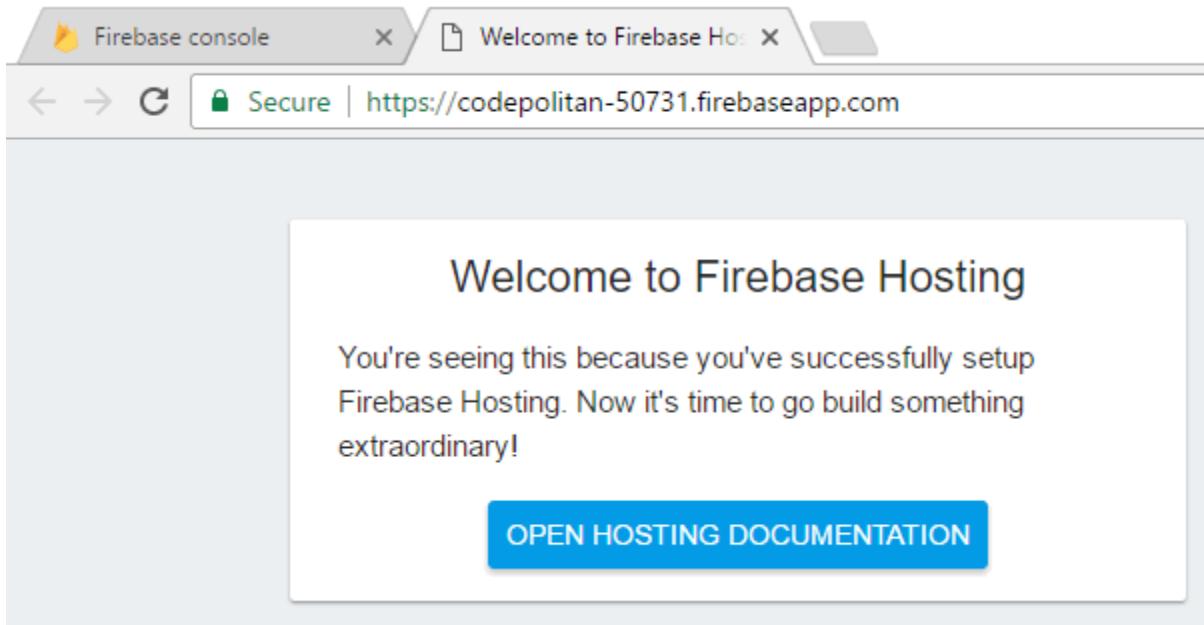
Dan bila kita kembali ke-[Firebase Console](#), kita akan melihat *Deployment history* yang berisi catatan pen-deploy-an kita.

The screenshot shows the Firebase Hosting dashboard. At the top, there's a header with the Firebase logo, a project name "CodePolitan", and navigation links for "Go to docs" and a user profile. Below the header, the word "Hosting" is prominently displayed, followed by "DASHBOARD" and "USAGE".

In the "Domain" section, there's a large button labeled "CONNECT DOMAIN". Below it is a table with one row, showing a connected domain "codepolitan-50731.firebaseio.com" with a "Type" of "Default".

The "Deployment history" section shows a single deployment entry. It includes columns for "Status" (marked as "Current" with a star icon), "Time" (Feb 25, 2017, 12:44 PM), "Deploy" (represented by a user icon and email address [REDACTED]@gmail.com), and "Files" (2 files). At the bottom of this section are pagination controls: "Rows per page: 10", "1-1 of 1", and navigation arrows.

Kalau kita ke alamat codepolitan-50731.firebaseio.com kita melihat seperti pada gambar. Dan tandanya kita sudah berhasil.



Lalu bagaimana kita me-update-nya. Pastinya kita lakukan itu dahulu di komputer kita. Seperti contoh kita sekarang masukkan file tambahan seperti file [javascript](#), [CSS](#), [image](#) dan lainnya kedalam *folder public*.

Name	Date modified
index.html	25-Feb-17
404.html	25-Feb-17
images	25-Feb-17
assets	25-Feb-17

Dari situ kita cukup gunakan perintah firebase deploy lagi. Maka maka kita ia akan menganalisis dan mengupload ke server. Mudah bukan?

```
D:\app\firebasehost>firebase deploy
=== Deploying to 'codepolitan-50731'...

i  deploying database, hosting
+  database: rules ready to deploy.
i  hosting: preparing public directory for upload...
Uploading: [=====] 56%+  hosting: public folder uploaded successfully
+  hosting: 57 files uploaded successfully
i  starting release process (may take several minutes)...

+  Deploy complete!

Project Console: https://console.firebaseio.google.com/project/codepolitan-50731/overview
Hosting URL: https://codepolitan-50731.firebaseio.com
```

Kembali ke alamat codepolitan-50731.firebaseio.com kita lagi untuk mengecek apakah *deploy*-nya sudah berhasil atau tidak.

Pertemuan 14 - Proyek Kelompok

Tujuan

1. Mahasiswa dapat berkolaborasi dengan tim untuk membangun layanan diatas cloud
2. Mahasiswa dapat berkolaborasi dengan tim untuk membangun platform as a service

Proyek (pilih salah satu)

1. Membuat aplikasi yang berjalan diatas container docker. Ada pemisahan antara backend dan frontend pada container berbeda
2. Membuat aplikasi web yang sudah implementasi CRUD pada platform Heroku
3. Membuat aplikasi web yang sudah implementasi CRUD pada platform openshift
4. Membangun layanan dengan kubernetes

Referensi

<https://docs.docker.com/>

<https://kubernetes.io/docs/home>

<https://git-scm.com/docs/gittutorial>

<https://learn.openshift.com/>

<https://www.petanikode.com/git-untuk-pemula/>

<https://www.codepolitan.com/belajar-menggunakan-database-postgresql-di-ubuntu-1604-59f6e0c63f5e4>

<https://www.codepolitan.com/web-static-dengan-firebase-hosting-58b36787c3f00>

<https://firebase.google.com/docs/database/?hl=id>

<https://www.excellent.co.id/product-services/vmware/keuntungan-teknologi-virtualisasi-cloud-computing/>