Modul Pemrograman Perangkat Bergerak #004

Materi : Menggunakan Version Control System (Git) di Android Studio.

Version Control System (VCS)

Version Control System (**VCS**) atau dapat disebut juga **Source Code Management** (**SCM**) adalah software yang berguna untuk membantu memanage perubahan pada source code. Dapat dianggap juga sebagai database yang merekam perubahan kode yang ada. VCS dapat digunakan untuk membantu mengembalikan kode ke semula dengan membandingkan kode dengan versi sebelumnya yang telah tersimpan, yang dapat berguna ketika terdapat kode yang membuat error, terutama jika bekerja dalam tim (lebih dari 1 programmer). Selain itu menggunakan VCS, memudahkan kita dalam menggabungkan kode (**merge**) antar programmer ataupun membuat versi kode yang berbeda (**branch**) jika dibutuhkan.

Beberapa VCS yang banyak digunakan, antara lain **Git** dan **Mercurial** yang merupakan tipe Distributed VCS (**DVCS**), yang dapat digunakan secara local ataupun menggunakan server, lalu **SVN/Subversion** yang merupakan tipe Centralized, yang menggunakan server.

Git adalah DVCS yang gratis dan open source, baik untuk project kecil maupun besar. **Git** semakin terkenal penggunaannya dengan munculnya beberapa server git gratis seperti **github**, **gitlab**, maupun **bitbucket**.

VCS yang didukung oleh Android Studio saat ini adalah CVS, Git, Mercurial, dan Subversion.

Tempat penyimpanan / database untuk kode pada VCS dikenal dengan sebutan **repository**. Selain itu terdapat istilah seperti commit, push, pull, fetch, dll.

VCS/SCM:

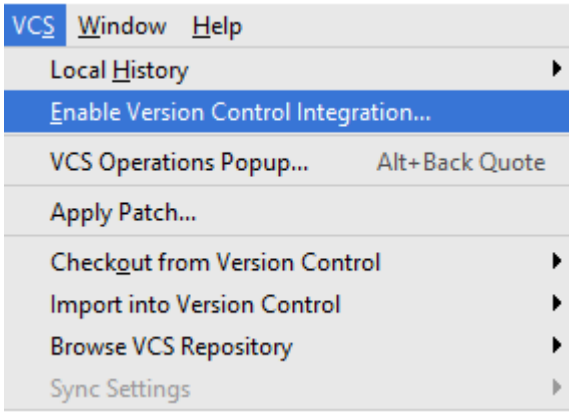
- **Git**
  https://git-scm.com
- **Mercurial/Hg**
  https://www.mercurial-scm.org
- **SVN/Subversion**
  https://subversion.apache.org
  https://www.visualsvn.com

Server Git Gratis:

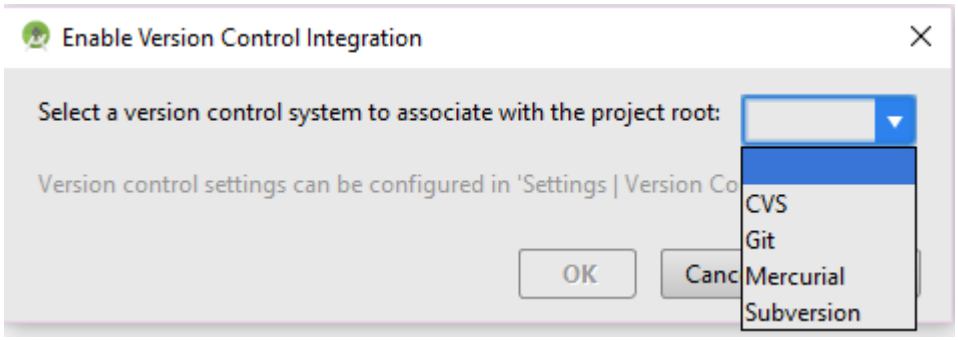
- **GitHub**
  https://github.com
- **GitLab**
  https://gitlab.com
- **Bitbucket**
  https://bitbucket.org

Praktek Git di Android Studio

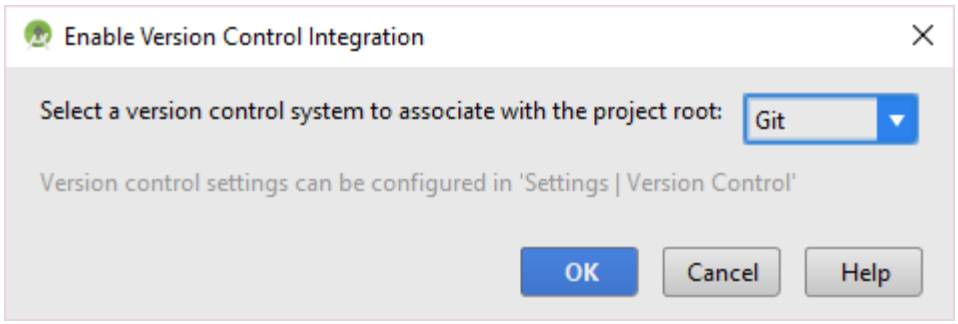
1) Buat Project Baru dengan Nama Aplikasi **HelloGit** dan Company Domain **learn.smktelkom-mlg.sch.id**, pilih template **Empty Activity** (yang lain default).
2) Aktifkan **Git** dengan cara :
   - ❖ Pilih pada menu **VCS** → **Enable Version Control Integration…** .



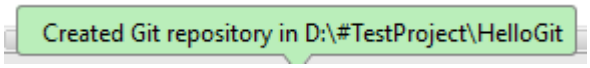
- ❖ Pada dialog yang muncul, klik pada DropDown.



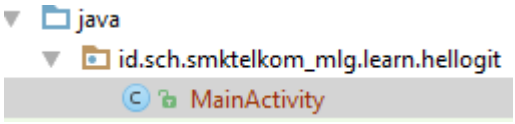
Kemudian pilih **Git**, tekan OK.



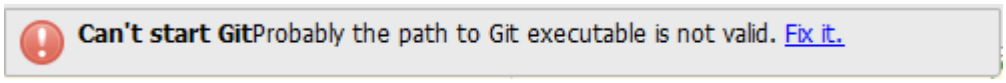
- ❖ Maka akan muncul pesan Created Git repository in {folder Project} , yang berarti Git telah dibuat dan diaktifkan.



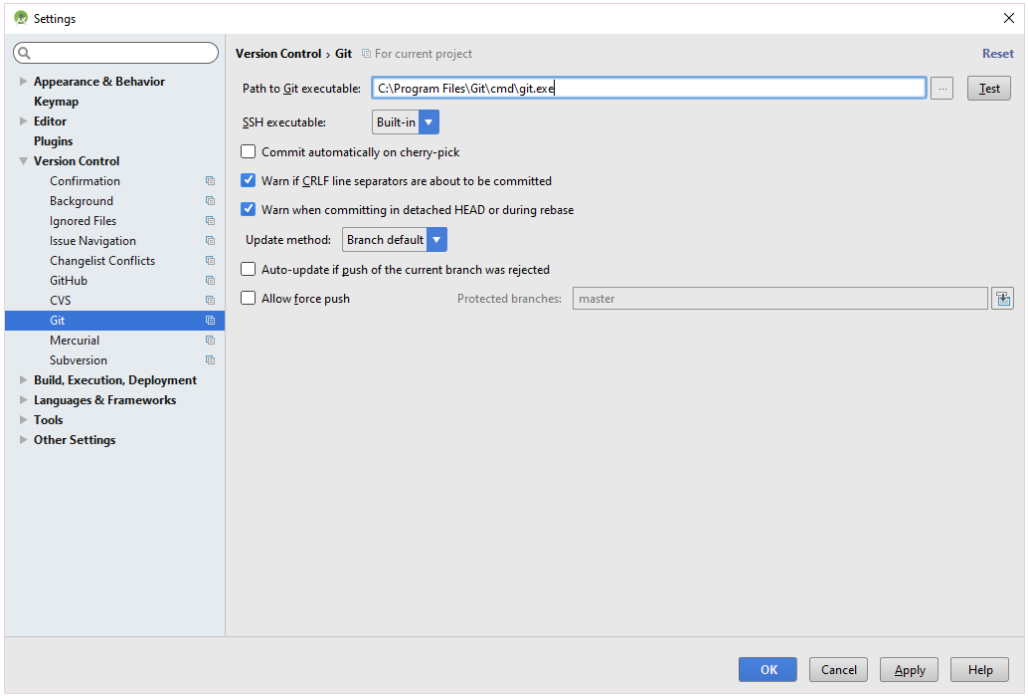
Selain itu dapat pula terlihat melalui warna dari nama file yang ada, yang semula hitam berubah menjadi merah, yang berarti bahwa file tersebut belum ditambahkan pada **repository** Git (di luar repository).



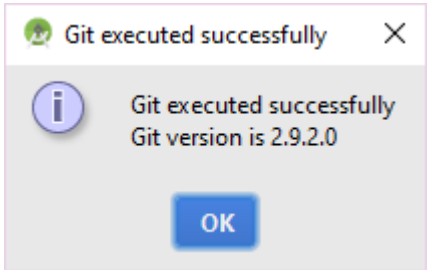
Jika muncul error **Can't start Git** setelah menekan OK, berarti path dari Git belum tersetting dengan benar. Hal ini dapat terjadi apabila kita menginstall Git setelah menginstall Android Studio, sehingga path Git tidak terdeteksi secara otomatis.



Untuk memperbaikinya tekan **Fix it**, kemudian Window Setting dari Git akan terbuka. Isi **Path to Git executable** dengan path dari file **git.exe** di PC masing-masing. Secara default path file git.exe berada di **C:\Program Files\Git\cmd\git.exe**.

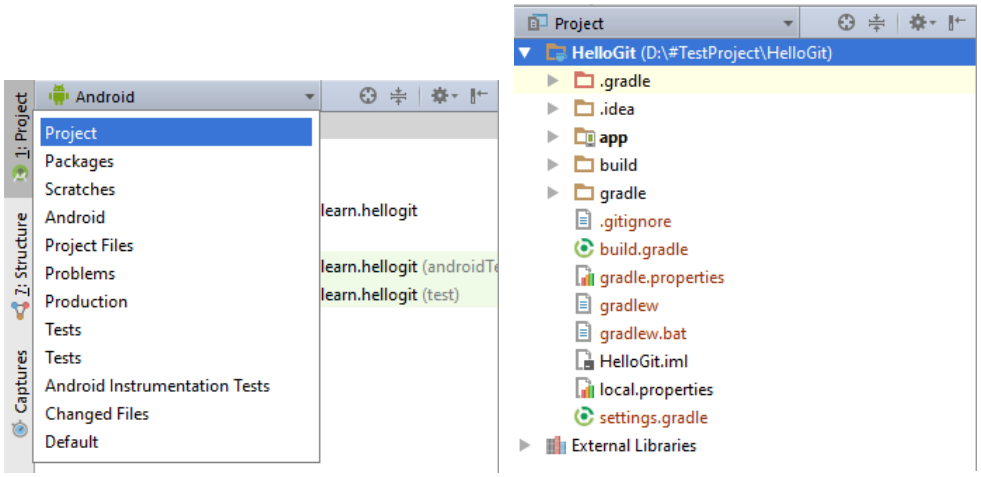


Tekan **Test** untuk memastikan path sudah benar. Jika sudah benar, akan muncul dialog yang menyatakan Git sukses dijalankan. Tekan Ok pada Dialog.

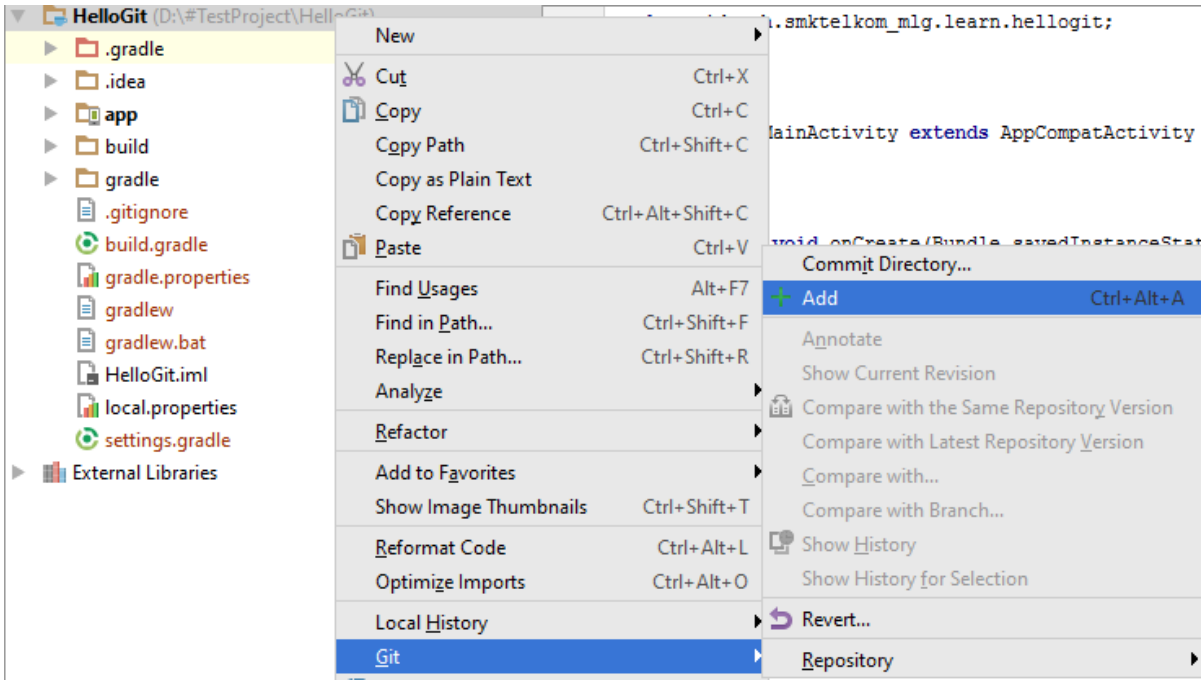


Tekan Ok pada Window Setting untuk menyimpan konfigurasi Git. Kemudian ulangi kembali langkah mengaktifkan Git.

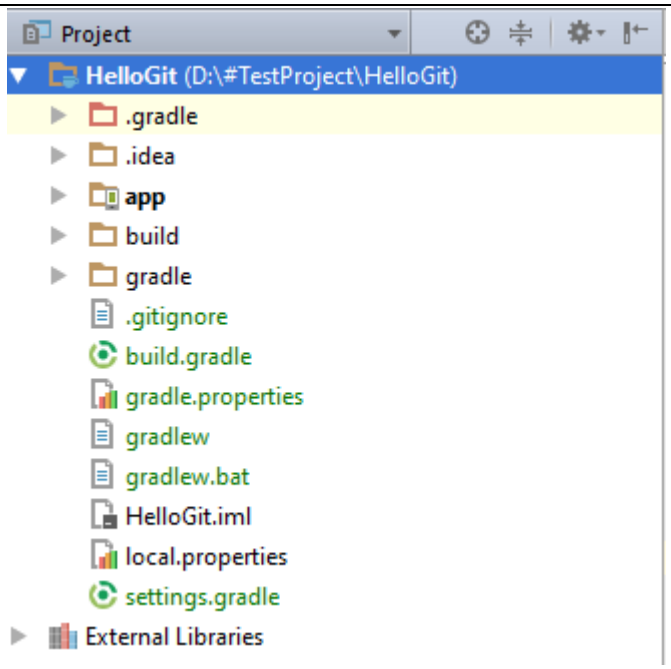
3) Selanjutnya kita perlu menambahkan (**Add**) semua file pada Project kita ke dalam repository Git, dengan cara:
   ❖ Rubah mode view Project Explorer pada bagian kiri atas dari **Android** ke **Project**.



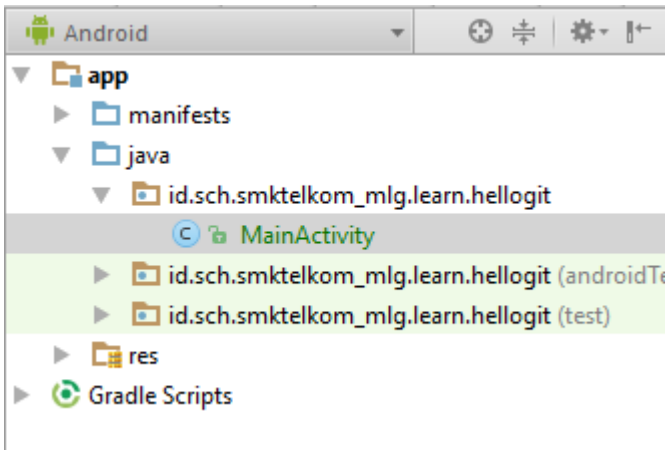
❖ Klik kanan pada folder Project (**HelloGit**), pilih **Git** → **Add**.



Jika berhasil, maka warna nama file akan berubah dari merah ke hijau, yang berarti file sudah ditandai untuk ditambahkan ke dalam repository.

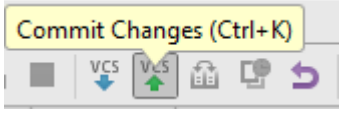


❖ Rubah kembali mode view Project Explorer dari **Project** menjadi **Android**.

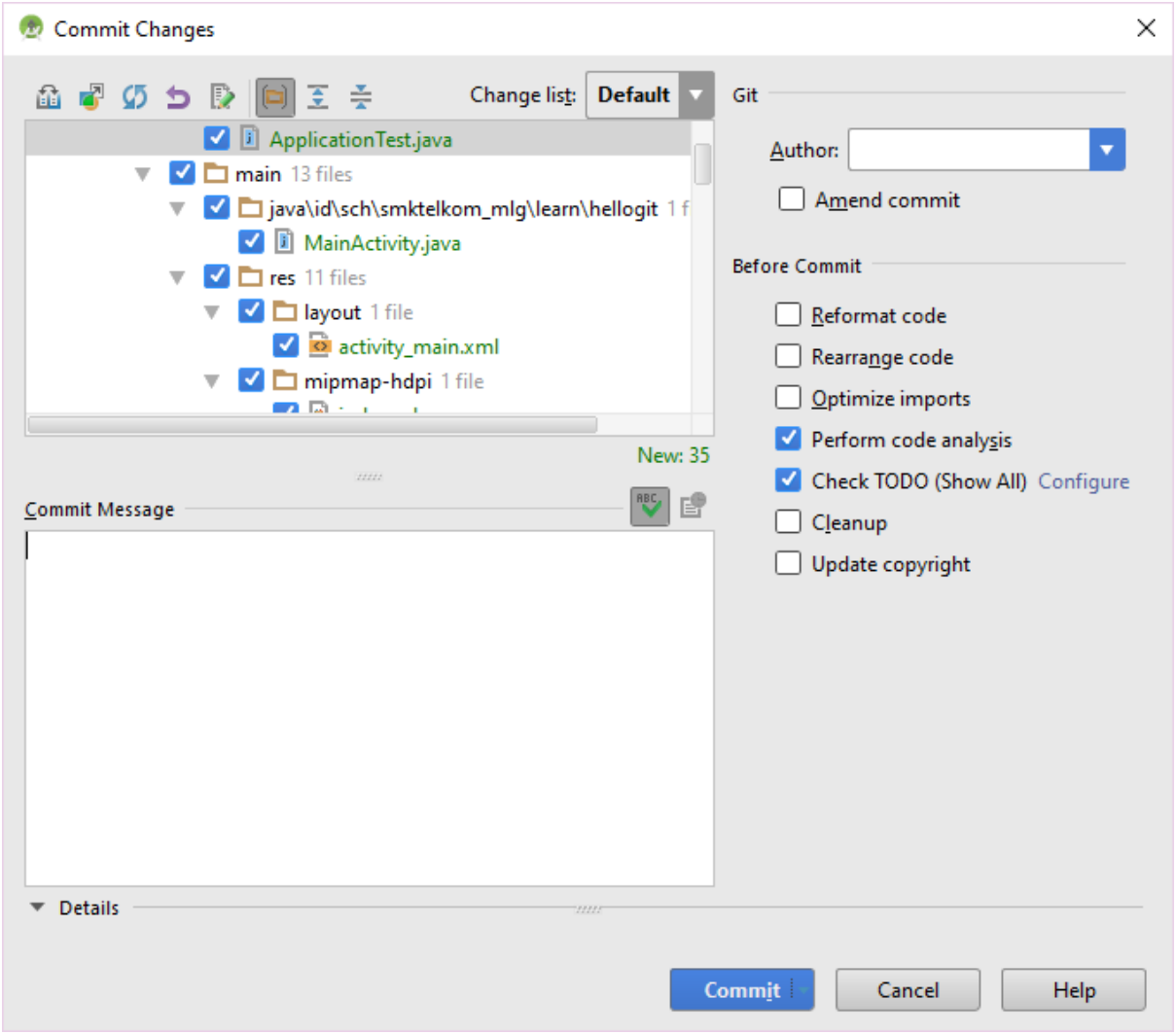


4) File sudah ditandai, tetapi belum disimpan ke dalam repository. Menyimpan di VCS disebut dengan **commit**. Untuk melakukan **commit** kita lakukan dengan cara :

❖ Klik pada icon **Commit Changes**  yang terdapat pada toolbar atas.

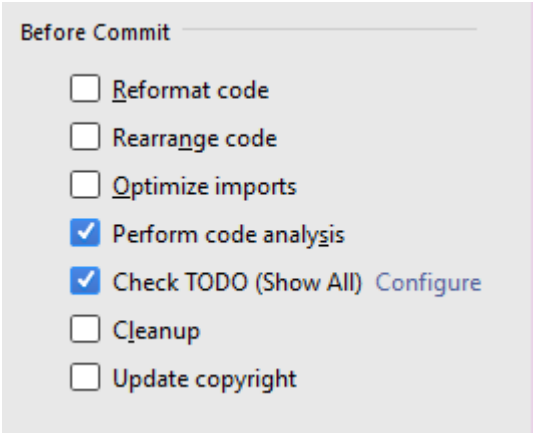


❖ Akan muncul dialog seperti di bawah ini.



❖ Isikan pada bagian **Commit Message**, keterangan tentang commit yang dilakukan, misalnya merubah …, menambahkan …, revisi/versi ke…, dsb. Commit Message sangat penting karena memudahkan kita mengidentifikasi perubahan apa yang sudah kita lakukan pada saat melakukan commit, karena itu tuliskan keterangan sejelas mungkin. Pada Commit Message dapat juga digunakan simbol-simbol yang ditempatkan sebelum apa yang kita tulis untuk memudahkan mengidentifikasi apa yang kita lakukan, misalnya tanda + untuk ketika menambahkan, tanda - untuk ketika mengurangi, tanda * untuk ketika memodifikasi, dsb. Untuk kali ini sebagai percobaan kita ketikkan pesan **first commit**, karena ini adalah commit pertama dari project kita.

❖ Pada sebelah kiri terdapat beberapa opsi **Before Commit**. Opsi-opsi tersebut akan dijalankan sebelum kita melakukan commit.



Opsi pada Before Commit:

- **Reformat code**
  Berguna untuk mengatur (secara otomatis) format penulisan kode sesuai dengan style penulisan yang kita gunakan (yang sudah diatur sebelumnya pada Setting Code Style dari Project). Pengaturan style dapat dilakukan melalui menu **File → Settings… → Editor → Code Style**. Selanjutnya pilih file yang ingin diatur stylenya, misalnya untuk style pada file Java, pilih Java kemudian rubah stylenya. Reformat code juga dapat dijalankan terpisah melalui menu Code.

- **Rearrange code**
  Berguna untuk mengatur (secara otomatis) posisi (baris) kode agar lebih rapi (sesuai dengan Setting Arrangement di Code Style dari Project). Rearrange code juga dapat dijalankan terpisah melalui menu Code.

- **Optimize imports**
  Berguna untuk menghapus (secara otomatis) import-import yang tidak digunakan. Optimize imports juga dapat dijalankan terpisah melalui menu Code.

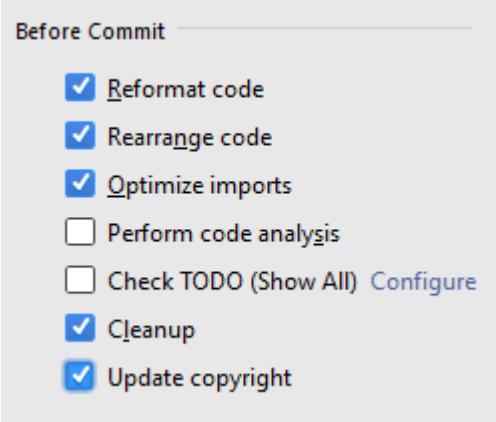
- **Perform code analysis**
  Berguna untuk menjalankan fasilitas pengecekan kode (Code Inspection) pada file yang akan dicommit. Akan melakukan pengecekan kode dan memberikan peringatan jika ada kode yang dianggap tidak bagus penulisan / implemetasinya.

- **Check TODO**
  Berguna untuk melakukan pengecekan kode yang perlu diisi (sesuai Setting TODO). Akan memberikan peringatan jika ada TODO yang belum diisi.

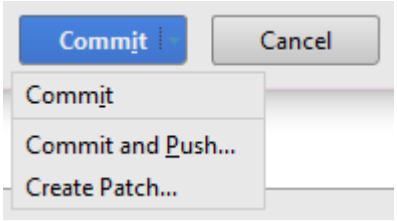
- **Cleanup**
  Berguna untuk melakukan perbaikan otomatis terhadap kode (sesuai setting Cleanup inspection) pada file yang akan dicommit. Mirip seperti Reformat Code.

- **Update copyright**
  Berguna untuk mengupdate lisensi copyright pada tiap file yang akan dicommit.

Kali ini kita akan mengaktifkan opsi selain **Perform code analysis** dan **Check TODO**, karena kedua opsi ini akan mencegah proses commit ketika ada kode yang dianggap belum sesuai.

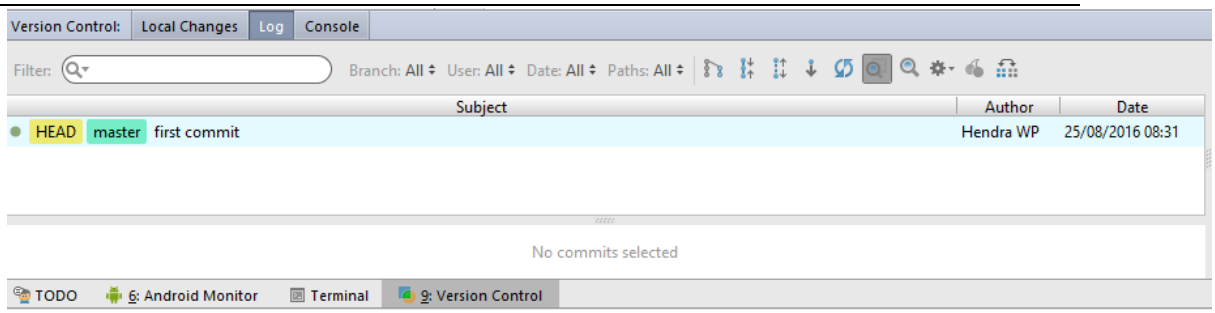


❖ Selanjutnya kita tekan tombol **Commit** untuk melakukan commit. Pada saat kita mengarahkan cursor ke tombol commit akan terdapat 3 opsi, **Commit** yang berguna untuk menyimpan pada repository lokal, **Commit and Push…**, yang berguna untuk menyimpan pada repository lokal sekaligus melakukan upload pada server git, dan **Create Patch…**, yang berguna untuk membuat patch. Pilih **Commit** karena kita ingin menyimpan pada repository lokal di Project kita.

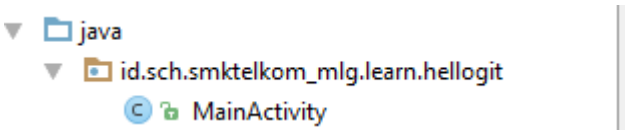


❖ Ketika kita lakukan commit pertama kali pada Git akan muncul dialog untuk mengisi nama dan email yang akan digunakan sebagai identitas orang yang melakukan commit (Author). Isi nama dan email yang benar sesuai identitas, dan tekan commit. Jika berhasil akan terdapat pesan dengan format **{jumlah file} file commited : {pesan commit}**.

Jika terdapat pesan error, klik pada tab **Version Control** yang terdapat pada bagian bawah, kemudian pilih tab Log. Lihat apakah hasil commit sudah muncul (identifikasi berdasarkan pesan commit yang tadi dituliskan). Jika sudah muncul maka commit sudah berhasil (error dapat diabaikan).



Selain itu dapat pula dicek dari warna nama file yang sudah dicommit, jika warna sudah berubah menjadi hitam, maka file telah berhasil dicommit.

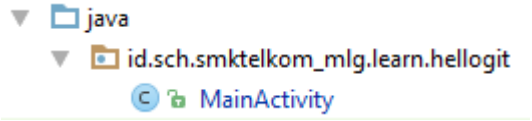


5) Selanjutnya kita akan coba merubah kode pada file **MainActivity.java** dan menggunakan Git untuk mengecek perubahan kode yang ada. Tambahkan kode pada method **OnCreate** seperti di bawah ini.

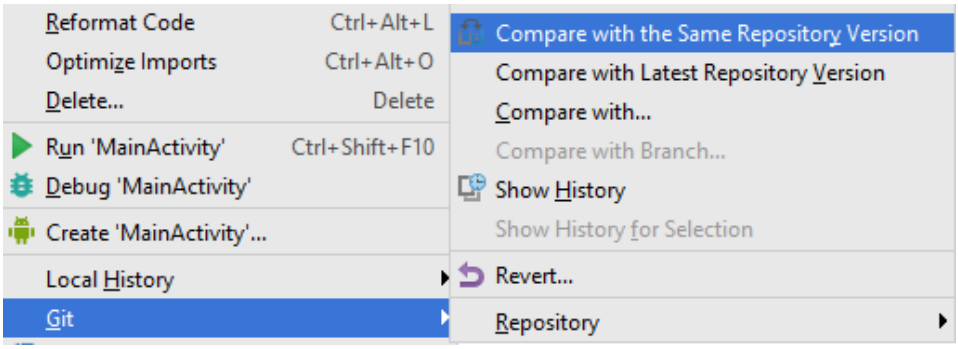
```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    int x=5;
}
```

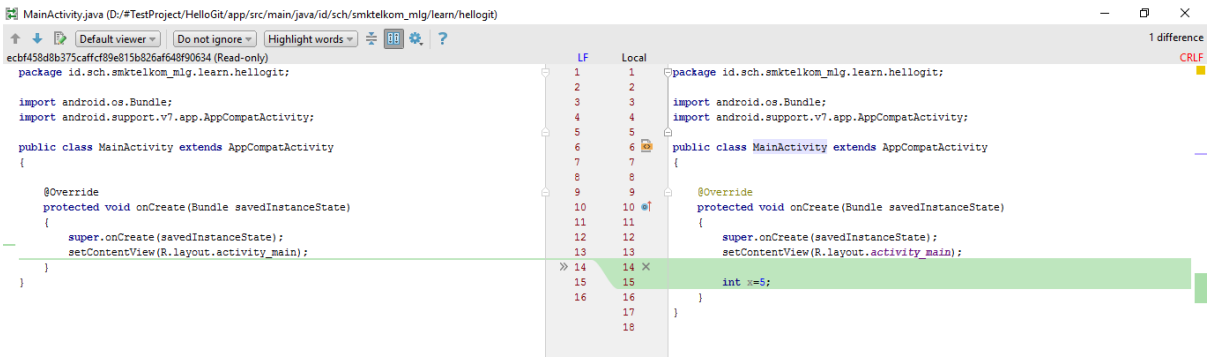
Perhatikan bahwa warna nama file **MainActivity.java** berubah menjadi biru, yang berarti file telah mengalami perubahan.



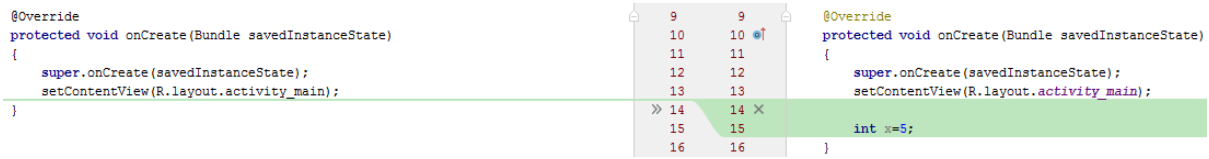
6) Kemudian lakukan klik kanan pada file **MainActivity.java** dan pilih **Git → Compare with the Same Repository Version**.



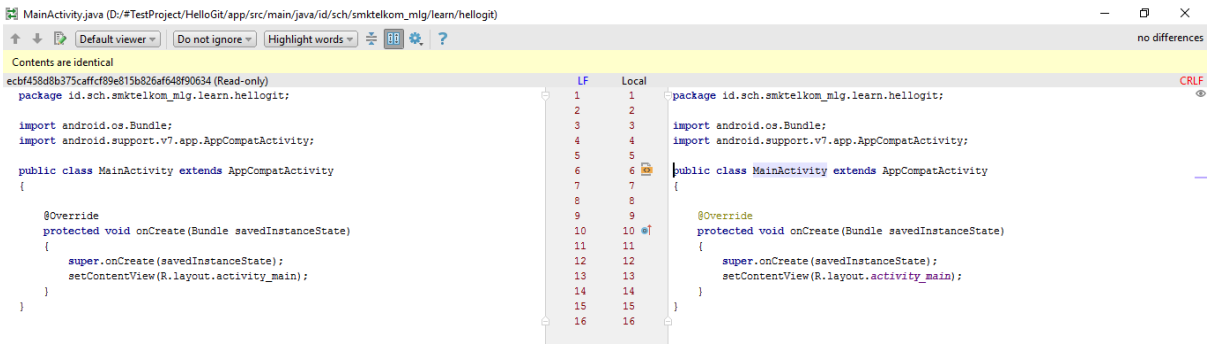
7) Maka akan muncul window yang terbagi 2, pada bagian kiri muncul kode terakhir yang telah kita simpan pada repository dan pada bagian kanan muncul kode yang kita miliki sekarang.



Perhatikan blok warna hijau yang menunjukkan kode yang berubah.



8) Selanjutnya kita akan mengembalikan kode yang kita miliki sehingga kembali seperti kode yang kita simpan pada repository. Caranya adalah dengan melakukan klik pada tanda ✕ (Revert) atau pada tanda » (Replace). Kali ini coba klik pada tanda ✕ (Revert). Maka kode di sebelah kiri dan kanan akan menjadi sama (jika sama akan muncul pesan **Contents are identical**).

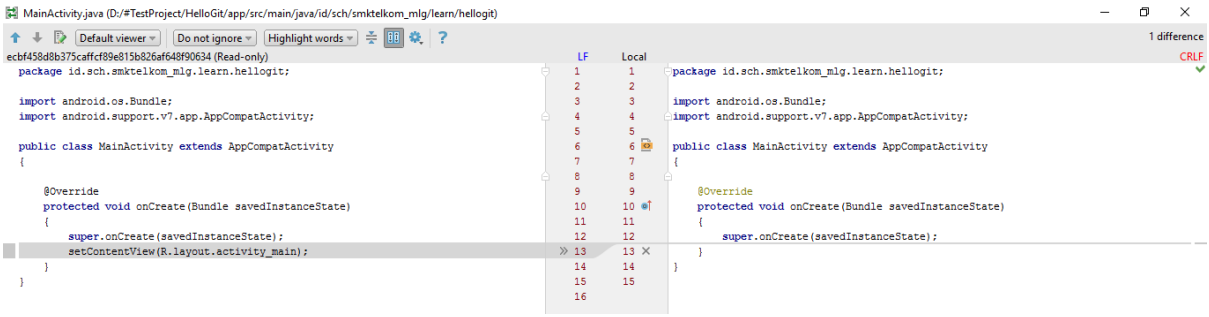


Tutup Window, dan dapat dilihat bahwa kode kita sudah kembali seperti kode yang kita simpan pada repository.

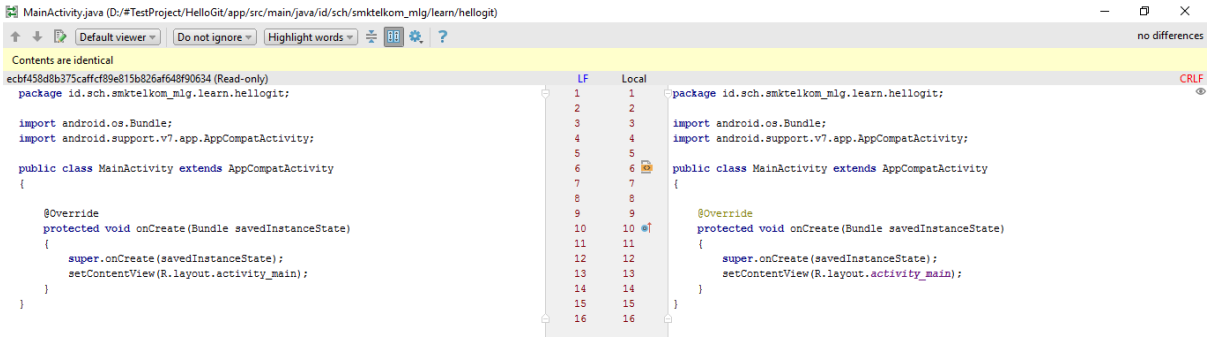
9) Selanjutnya kita coba hapus kode yang ada dan kita coba kembalikan ke semula menggunakan Git. Pertama, hapus kode **setContentView** pada method **OnCreate**.

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
}
```

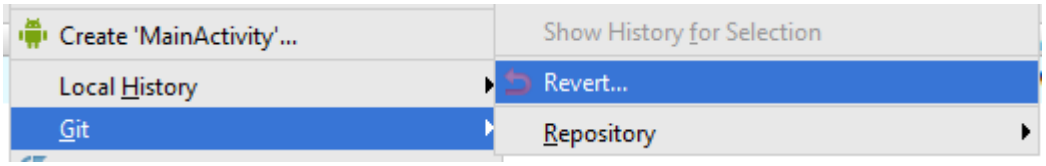
10) Kemudian lakukan hal yang sama dengan sebelumnya. Lakukan klik kanan pada file **MainActivity.java** dan pilih **Git → Compare with the Same Repository Version**.



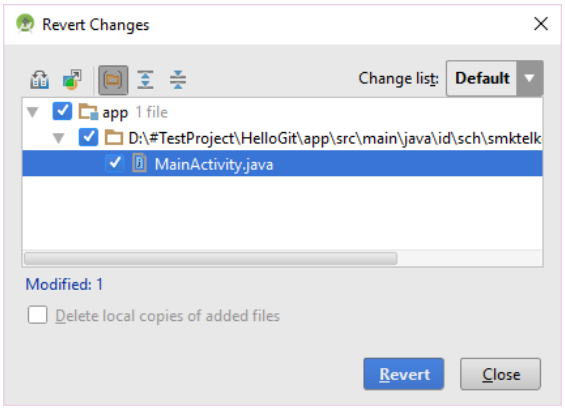
11) Selanjutnya sama seperti sebelumnya, kita dapat mengembalikan kode seperti semula melakukan klik pada tanda ✕ (Revert) atau pada tanda » (Replace). Kali ini coba klik pada tanda » (Replace).



Jika kita ingin mengembalikan secara otomatis terutama utuk lebih dari 1 file, maka kita dapat menggunakan perintah **Git → Revert**.

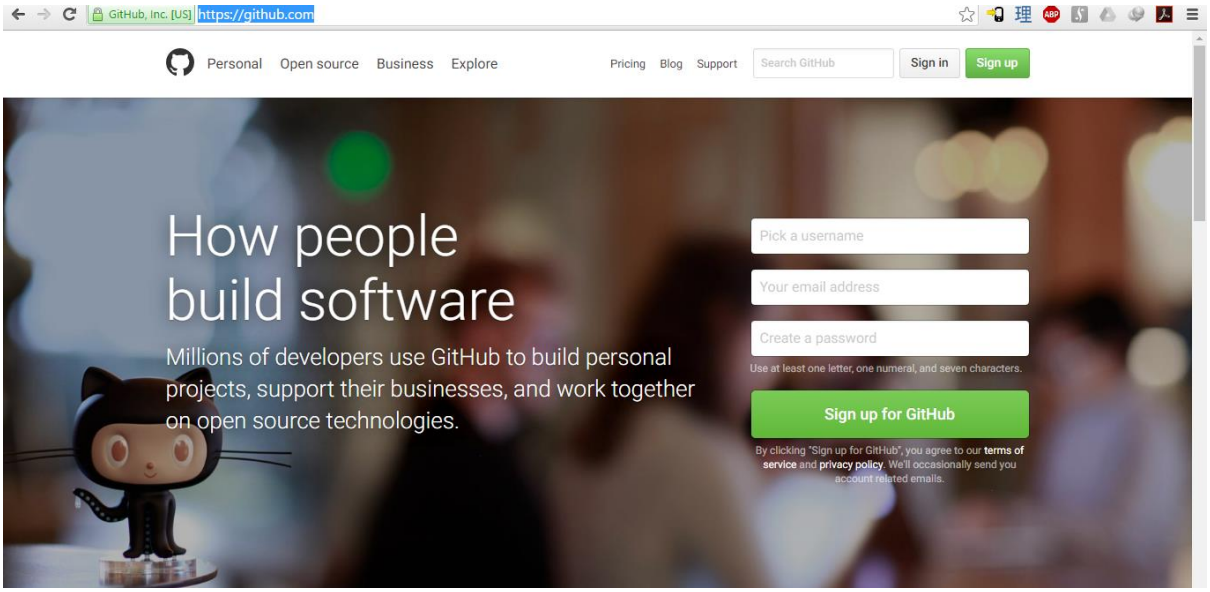


Lalu pada Dialog yang muncul pilih file yang ingin dikembalikan dan tekan Revert.

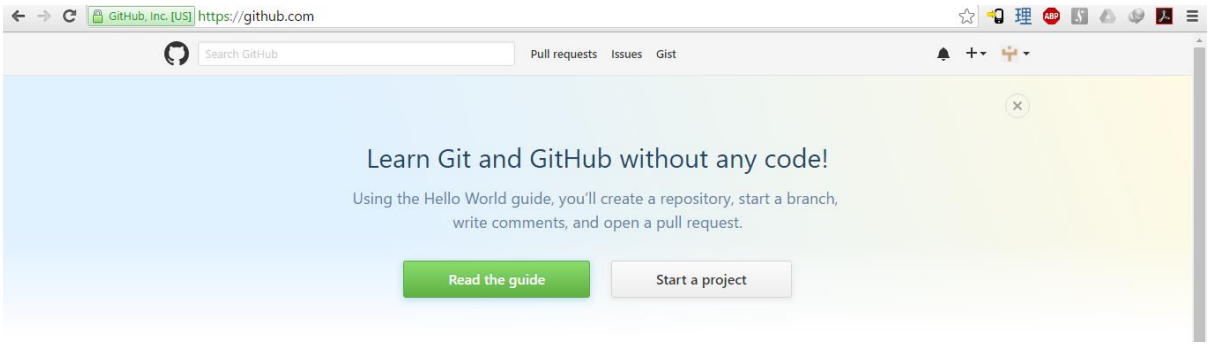


Maka file akan kembali sesuai dengan kode pada repository terakhir.

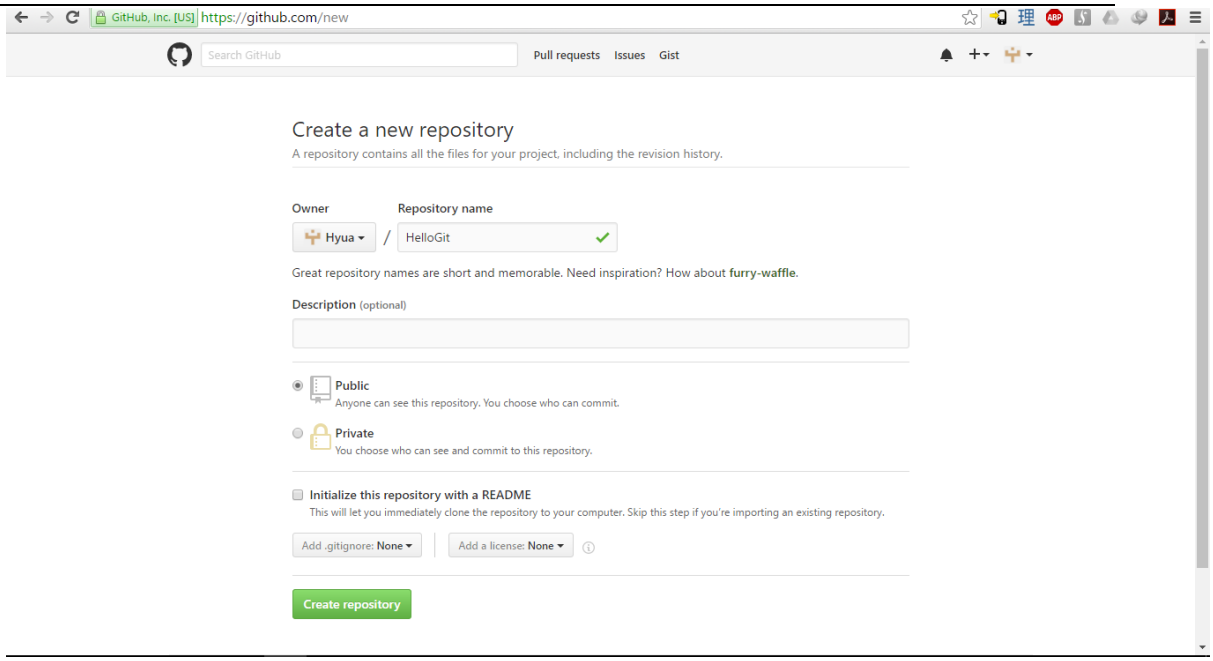
12) Selanjutnya kita akan coba melakukan upload (push kode) pada server **GitHub**. Sebelum dapat melakukan push kode pada server GitHub, kita harus terlebih dahulu mempunyai account pada GitHub. Buat account (**sign up**) pada website GitHub (https://github.com) jika belum mempunyai account. Kemudian lakukan **sign in** dengan account yang telah dimiliki.



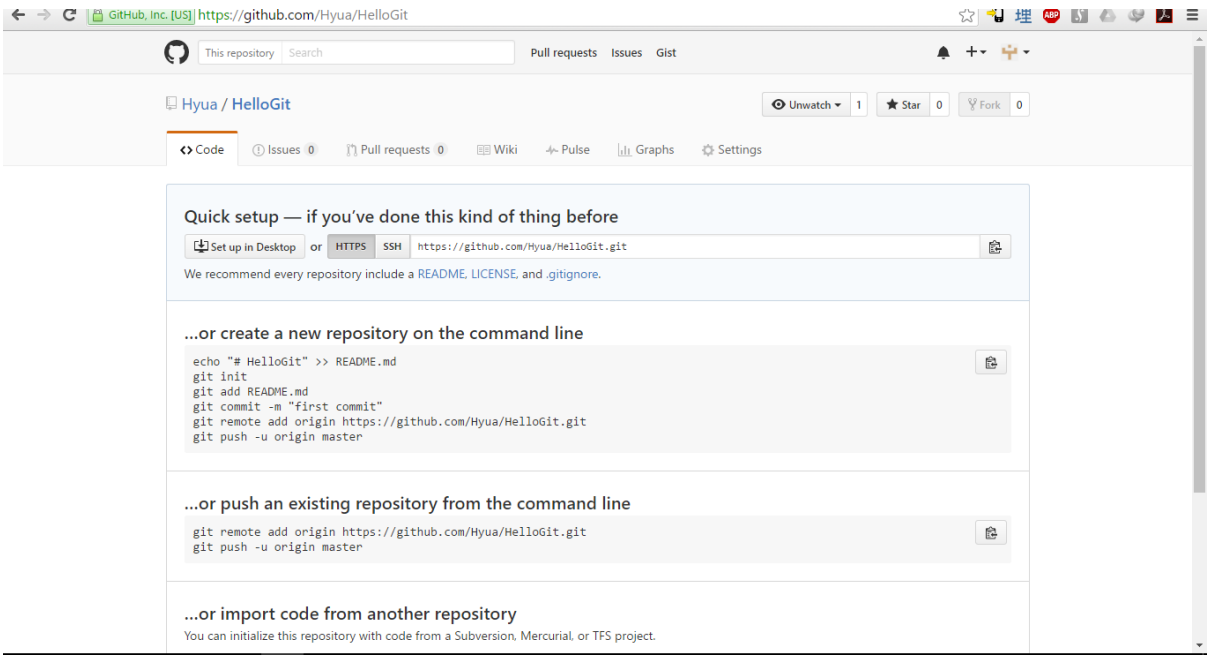
13) Setelah melakukan **sign in** buat project baru dengan menekan **Start a project**.



14) Pada halaman pembuatan repository baru, isi nama repository dengan **HelloGit** sesuai nama project kita. Kemudian klik **Create repository** untuk membuat repository. Perhatikan bahwa di sini kita menggunakan hak akses **Public** yang berarti repository yang kita buat dapat dilihat oleh siapa saja. Kita menggunakan hak akses Public karena pada github, repository dengan hak akses ini gratis, sedangkan untuk hak akses Private kita harus membayar. Kita juga dapat menambahkan deskripsi jika diperlukan.

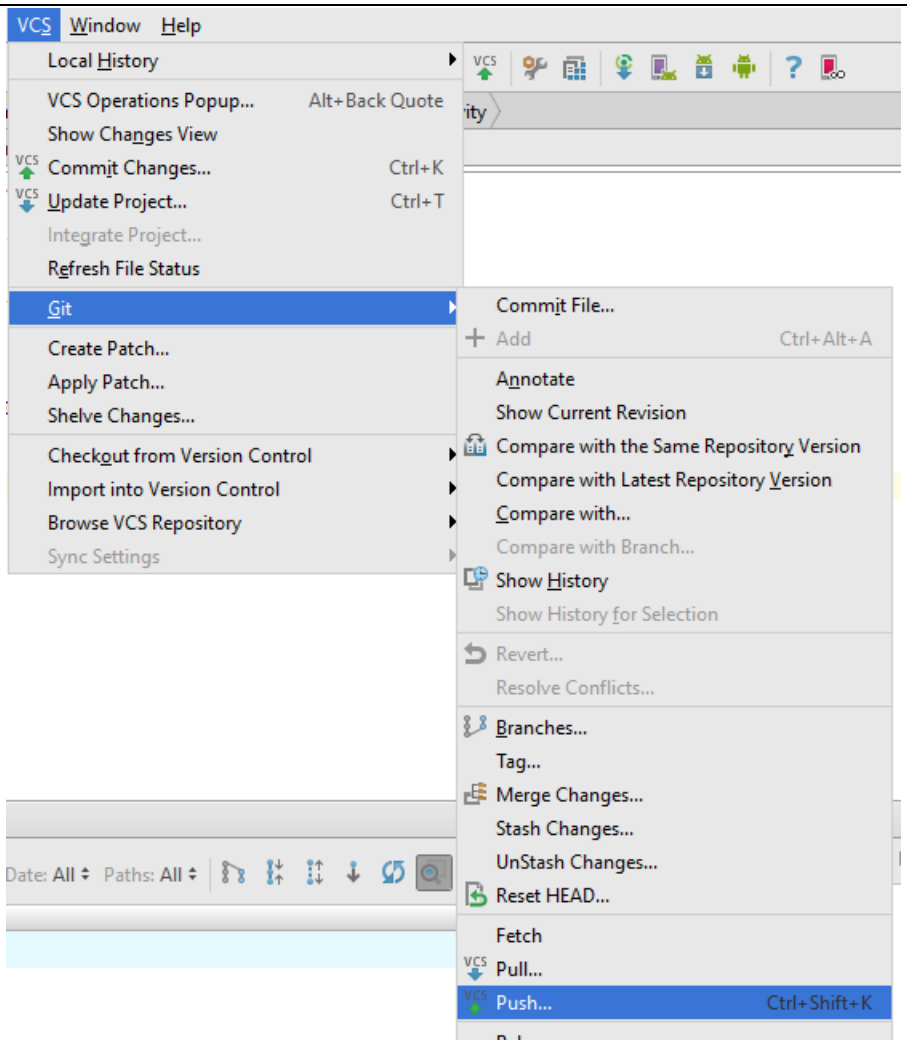


15) Setelah repository kita terbentuk, sekarang kita dapat mulai melakukan upload kode kita ke GitHub dari Android Studio.

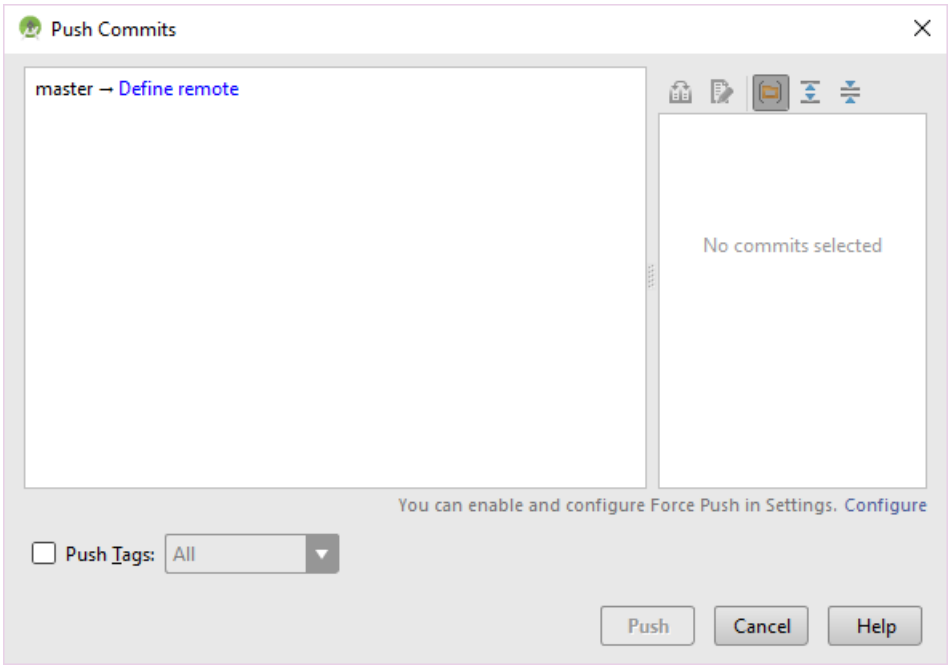


Pertama, copy alamat dari project GitHub milik kita, misalnya **https://github.com/Hyua/HelloGit.git**.

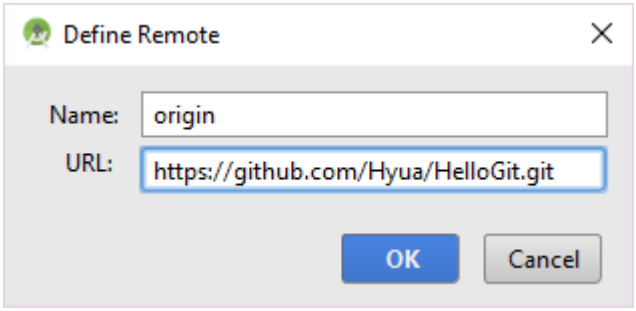
16) Kembali ke Android Studio, pilih menu **VCS → Git → Push…** untuk melakukan upload kode ke server.



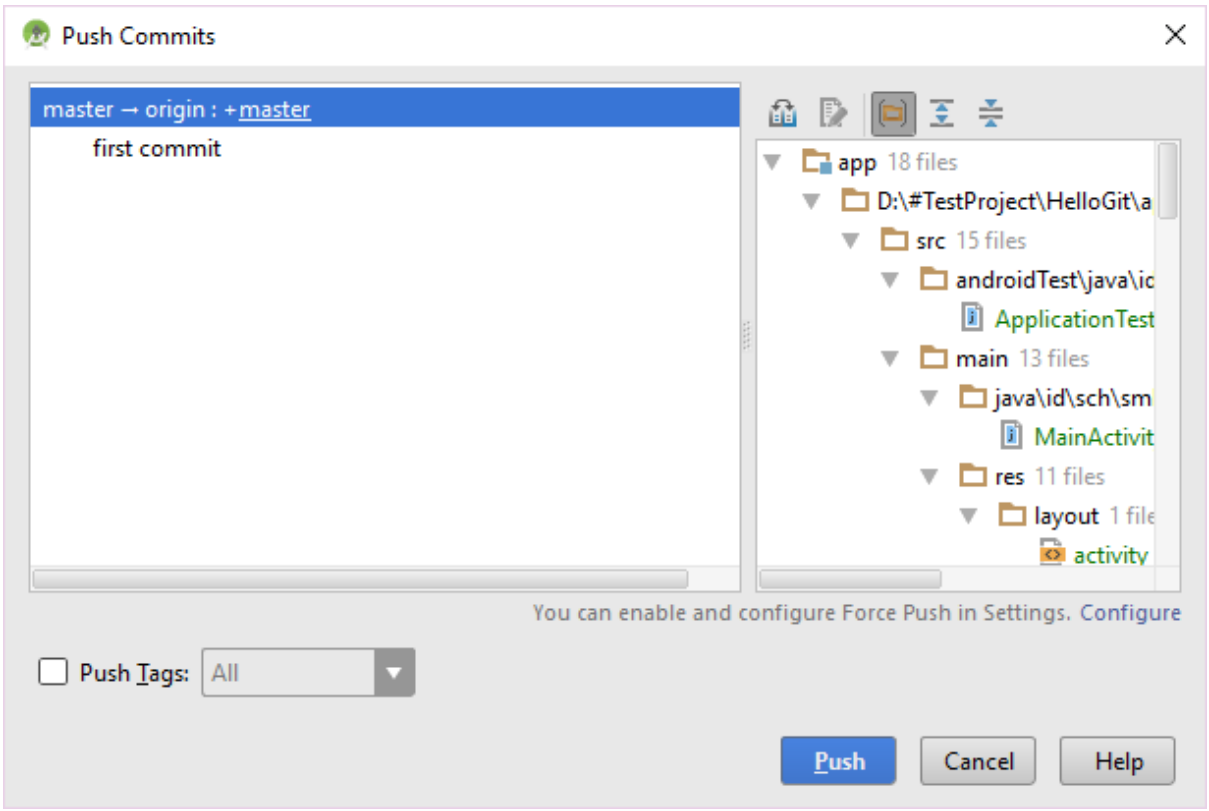
17) Pada Dialog (Push Commits) yang muncul, klik pada **Define remote** menghubungkan dengan GitHub.



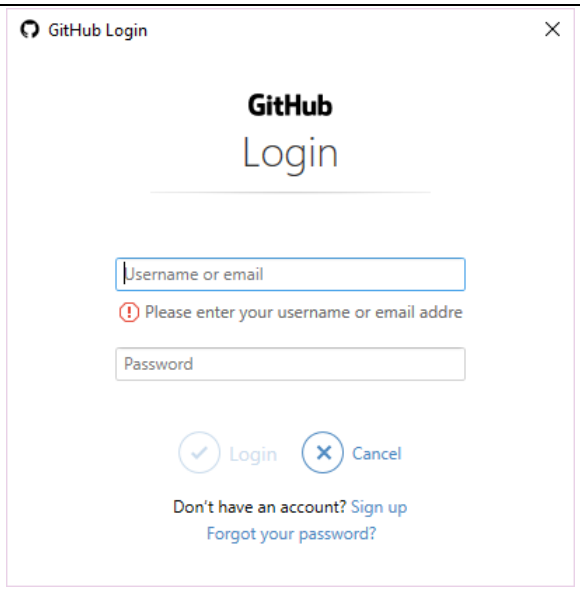
18) Pada dialog (Define Remote) yang tampil, lakukan paste pada bagian URL untuk mengisinya dengan alamat project kita (HelloGit) pada GitHub, kemudian tekan OK.



19) Tunggu hingga proses pengecekan URL. Jika berhasil maka kita akan dapat melihat daftar commit kita seperti di bawah ini. Tekan **Push** untuk melakukan push ke GitHub.

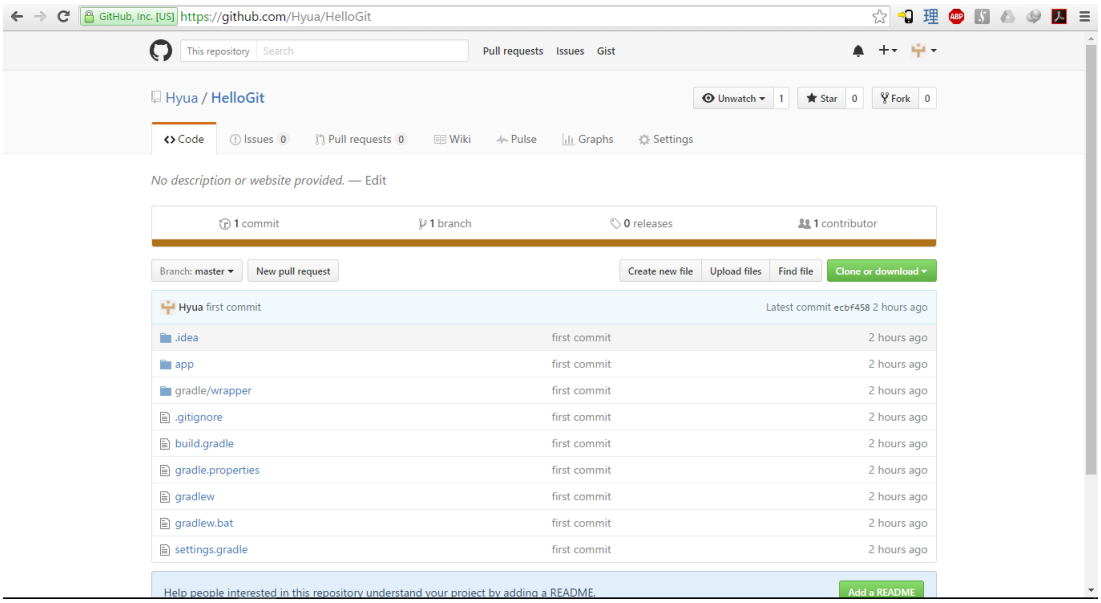


20) Pada Window Login GitHub yang muncul lakukan login sesuai dengan account yang dimiliki (yang digunakan untuk Project **HelloGit** pada GitHub).

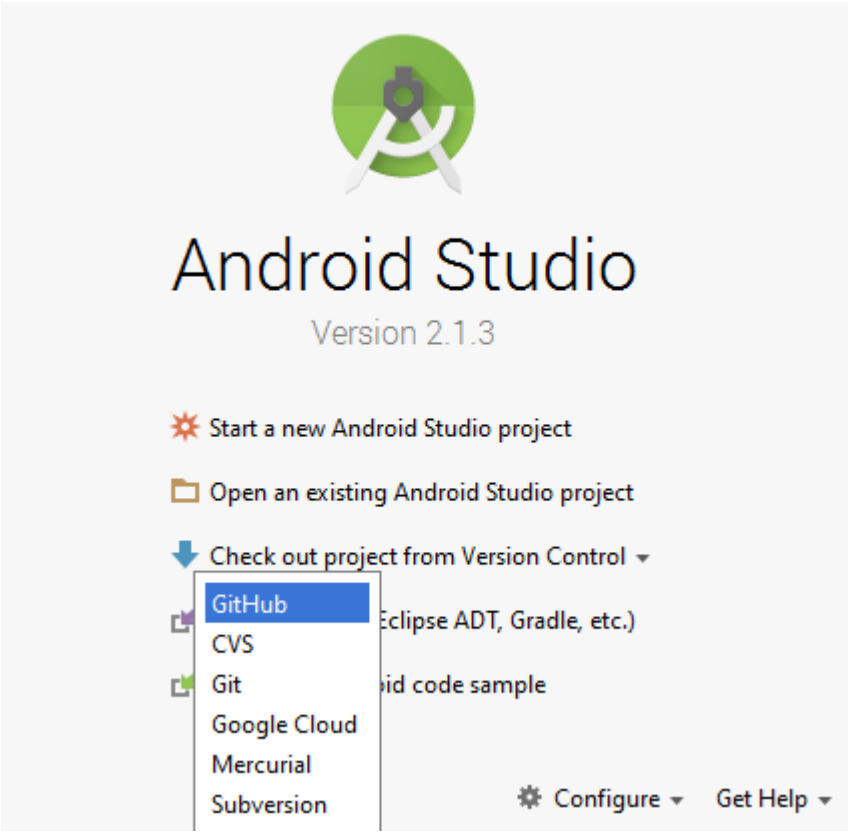


21) Kembali ke Android Studio dan tunggu hingga proses push selesai. Jika berhasil akan muncul pesan **Push successful** di bagian bawah.

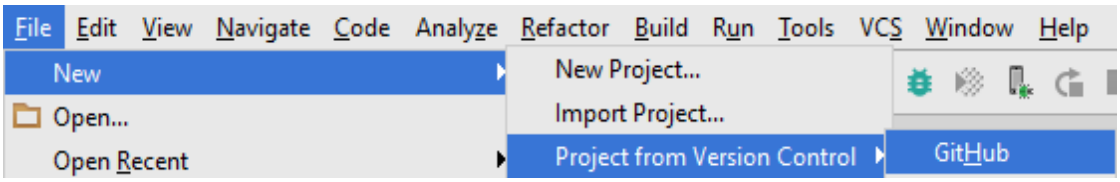
22) Buka halaman project kita pada GitHub (di browser). Kita dapat melihat bahwa file pada project kita telah terupload di GitHub.



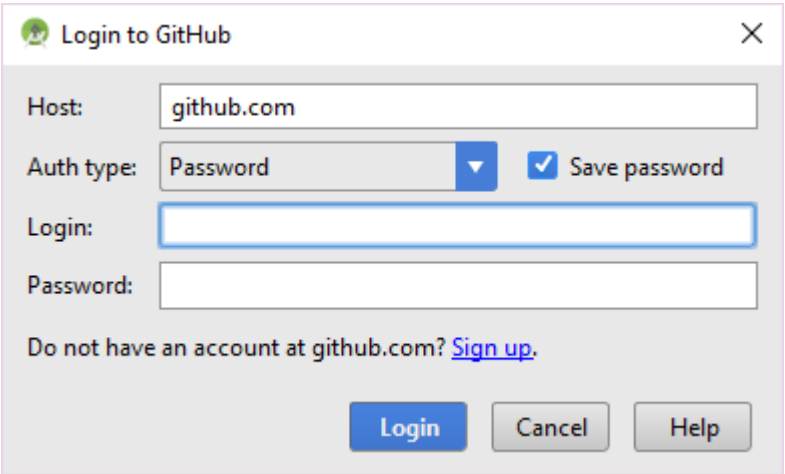
23) Jika kita ingin mendownload project kita pada PC yang lain (Clone), dapat dilakukan dengan memilih **Check out project from Version Control** (pada Halaman Awal Android Studio) kemudian pilih **GitHub**.



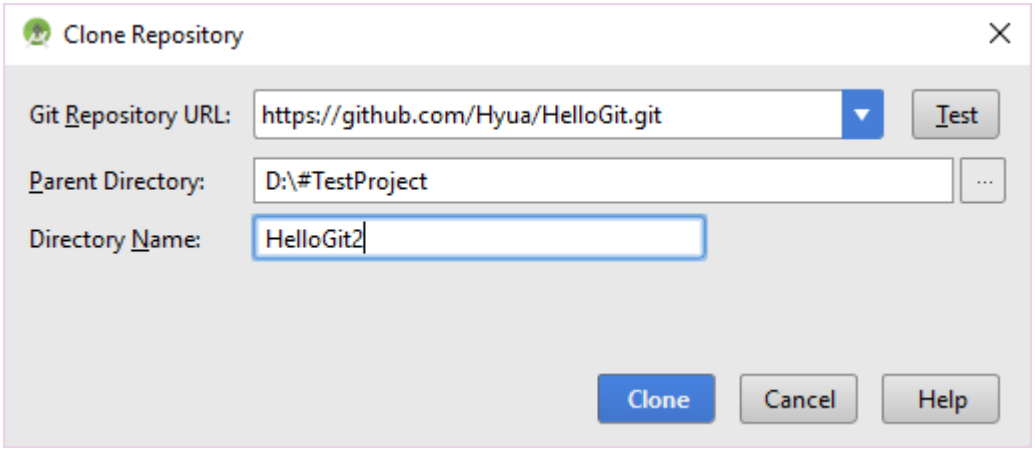
Atau melalui menu **File → New → Project from Version Control → GitHub** (pada saat sudah membuka Project lain).



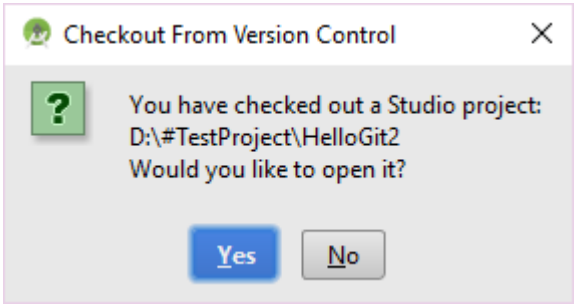
24) Kemudian isi dialog Login yang muncul sesuai account pada GitHub dan tekan Login.



25) Pada dialog (**Clone Repository**) yang muncul, isikan URL dari project kita (**Git Repository URL**) dan sesuaikan **Parent Directory** dan **Directory Name** dari Project yang akan dibuat di PC kita. Lalu tekan **Clone**. Tunggu hingga proses clone selesai.



26) Jika sudah berhasil melakukan clone, maka akan muncul Dialog (**Checkout From Version Control**), Tekan **Yes** untuk membuka Project tersebut.



Project dari Github telah di download (clone) dan terbuka di PC yang lain.
Dengan cara ini kita dapat melakukan kolaborasi kode dalam tim (lebih dari 1 orang).

-------- Selamat Mengerjakan --------