Modul 07

Version Control, Git and App Deployment

TUJUAN PEMBELAJARAN

1. Mampu memahami dan menjelaskan tentang Version Control menggunakan Git
2. Mampu memahami dan menjelaskan tentang proses *app deployment*

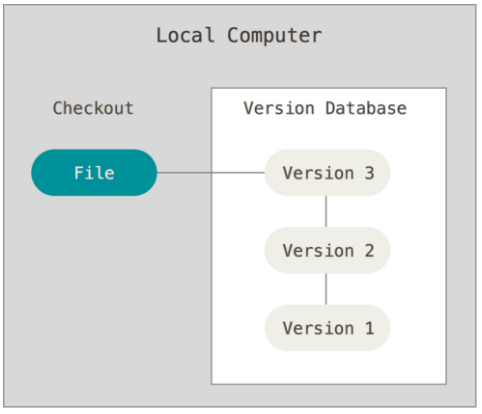
HARDWARE & SOFTWARE

1. PC (*Personal Computer)* dengan akses Internet
2. JavaScript
3. Visual Studio Code atau IDE lainnya yang mendukung JavaScript
4. NPM
5. Express.JS
6. Git dan GitHub

URAIAN MATERI

A. Version Control

Version control, juga dikenal sebagai kontrol versi atau pengendalian versi, adalah sebuah sistem yang merekam perubahan-perubahan dari sebuah berkas atau sekumpulan berkas dari waktu ke waktu sehingga pengembang dapat menilik kembali versi khusus suatu saat nanti.



Gambar 1. Sistem Version Control Lokal

Dengan adanya version control, pengembang perangkat lunak dan tim manajemen proyek dapat melacak dan mengelola perubahan dalam kode sumber atau berkas proyek. Hal ini memungkinkan tim pengembangan untuk bekerja secara kolaboratif pada proyek, mengelola perubahan, dan memantau evolusi kode sumber dengan cara yang terstruktur dan terdokumentasi.

a. Tujuan Version Control

Version control bertujuan untuk mencapai beberapa hal yang penting dalam pengembangan perangkat lunak, termasuk:

1) Pelacakan Perubahan: Mampu melacak semua perubahan yang dibuat pada proyek seiring waktu berjalan
2) Kolaborasi: Memungkinkan berbagai anggota tim bekerja bersama pada proyek yang sama tanpa menyebabkan konflik.
3) Pemulihan: Memungkinkan pemulihan kode sumber ke titik sebelumnya jika terjadi kesalahan atau masalah.
4) Pengujian: Memungkinkan pengujian dan pemantauan perubahan kode sumber.

b. Konsep Dasar

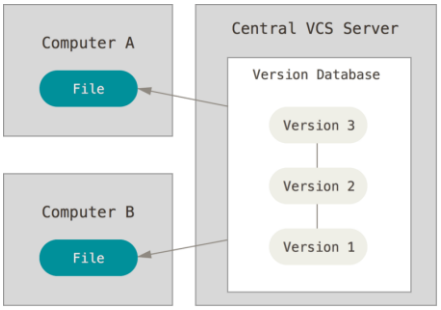
Berikut adalah beberapa hal penting tentang version control:

1) Repository: adalah tempat penyimpanan pusat di mana semua versi kode sumber disimpan. Terdapat dua jenis repository: pusat (centralized) dan terdistribusi (distributed).
2) Commit: Proses menyimpan perubahan ke dalam repository. Setiap commit memiliki pesan yang menjelaskan perubahan yang dilakukan.
3) Branch atau cabang adalah garis perkembangan independen dalam kode sumber, memungkinkan untuk mengembangkan fitur atau memperbaiki *bug* secara terisolasi.
4) Merge: Proses menggabungkan perubahan dari satu branch ke branch lainnya.
5) Conflict Resolution: Ketika dua perubahan yang bertentangan dikombinasikan, konflik harus dipecahkan.

c. Sistem Version Control

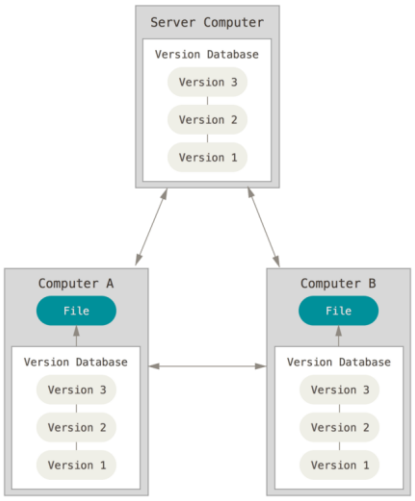
Ada dua jenis sistem version control yang umum digunakan:

1) Sistem Kontrol Versi Terpusat (Centralized Version Control System, CVCS): Misalnya, Subversion (SVN). Dalam CVCS, ada satu repositori pusat yang digunakan untuk menyimpan semua perubahan.



Gambar 2. Sistem Kontrol Versi Terpusat

2) Sistem Kontrol Versi Terdistribusi (Distributed Version Control System, DVCS): Misalnya, Git dan Mercurial. Dalam DVCS, setiap anggota tim memiliki salinan lengkap repositori, memungkinkan kerja terdistribusi dan offline.



Gambar 3. Sistem Kontrol Versi Terdistribusi

d. Manfaat Version Control

Adapun manfaat dari version control secara umum adalah sebagai berikut

1) Meningkatkan kolaborasi dan koordinasi dalam tim pengembangan.
2) Meningkatkan keamanan kode sumber dengan menyimpan versi yang dapat dipulihkan.
3) Memfasilitasi manajemen proyek dengan pelacakan perubahan dan pemantauan evolusi kode sumber.
4) Meningkatkan transparansi dalam pengembangan perangkat lunak.

B. Git

Git adalah salah satu sistem kontrol versi terdistribusi yang sangat popular dan banyak digunakan oleh pengembang perangkat lunak dan tim pengembangan untuk mengelola dan melacak perubahan dalam kode sumber sebuah proyek aplikasi.

Gambar 4. Beberapa *command* **git** yang umum digunakan

Dokumentasi git lebih lengkapdapat diakses melalui link berikut ini https://git-scm.com/docs atau https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F. Berikut adalah beberapa langkah-langkah umum dalam menggunakan Git:

1. Inisialisasi Repositori Git
   Untuk memulai penggunaan Git, perlu dilakukan inisialisasi repositori Git di dalam direktori proyek. Hal ini dapat dilakukan dengan perintah berikut:
   `git init`
2. Menambahkan dan Melacak Berkas
   Untuk melacak berkas dalam repositori, programmer harus menambahkannya dengan perintah `git add`. Misalnya, untuk melacak semua perubahan di direktori saat ini:
   `git add .`
3. Proses *Commit*
   Setelah menambahkan perubahan, maka perlu melakuan proses *commit* untuk menyimpan perubahan tersebut ke dalam repositori dengan perintah berikut:
   `git commit -m "Pesan commit Anda di sini"`
4. Branching
   Cabang (branch) dibuat untuk mengembangkan fitur atau memperbaiki bug tanpa memengaruhi cabang utama (biasanya disebut sebagai '**master**' atau '**main**'). Gunakan perintah **`git branch`** untuk melihat daftar cabang dan **`git checkout`** untuk beralih ke cabang yang berbeda
5. Merging
   Setelah selesai dengan perubahan di cabang yang berbeda, programmer dapat menggabungkan (merge) perubahan tersebut kembali ke cabang utama dengan menggunakan **`git merge`**
6. Pemantauan Status
   Hal ini dapat dilakukan dengan menggunakan perintah **`git status`** untuk memeriksa status berkas di direktori aplikasi dan apakah ada perubahan yang belum di-*commit*.
7. Penggunaan Remote Repositori
   Git mendukung kerja kolaboratif dengan menggunakan *remote repository*, seperti yang di-host di GitHub, GitLab, atau Bitbucket. Hal ini dapat dilakukan dengan menambahkan remote repository dengan perintah **`git remote add`**, dan mengirim perubahan ke remote repository dengan perintah **`git push`**
8. Sinkronisasi dengan Remote Repository
   Untuk mengambil perubahan dari remote repository ke repositori lokal, gunakan perintah **`git pull`**
9. *Conflict Resolution*
   Terkadang, saat menggabungkan perubahan, programmer mungkin akan menghadapi konflik. Konflik ini dapat diselesaikan dengan mengedit berkas terkait dan kemudian melakukan *commit* lagi.

10. *Time Travel* dan Pemulihan
    Git memungkinkan programmer untuk kembali ke titik sebelumnya dalam sejarah repositori dengan perintah **`git reset`**, **`git reflog`**, dan **`git checkout`**

C. App Deployment

Deployment aplikasi adalah proses meng-transfer perangkat lunak dari lingkungan pengembangan atau ujicoba (lokal) dan memasangnya di lingkungan produksi agar dapat diakses dan digunakan oleh pengguna akhir (*end-user*). *Deployment* merupakan tahap kritis dalam pengembangan perangkat lunak yang memerlukan perencanaan yang baik, otomatisasi, dan pemantauan yang cermat untuk memastikan bahwa aplikasi dapat berjalan dengan lancar dan andal di lingkungan produksi. Kesalahan dalam pengimplementasian dapat memiliki dampak serius terhadap kinerja dan ketersediaan aplikasi. Berikut adalah hal umum terkait tentang *deployment* aplikasi:

1. **Tujuan Deployment**
   Deployment aplikasi bertujuan untuk menjadikan aplikasi perangkat lunak siap digunakan dalam lingkungan produksi. Hal ini mencakup langkah-langkah untuk mengonfigurasi, menguji, dan menyiapkan semua sumber daya yang diperlukan, sehingga aplikasi dapat berjalan dengan baik dan efisien.
2. **Lingkungan Produksi**
   Lingkungan produksi adalah lingkungan di mana aplikasi akan digunakan oleh pengguna akhir. Hal ini dapat berupa server fisik, pusat data, atau infrastruktur cloud seperti AWS, Azure, atau Google Cloud. Lingkungan produksi harus dirancang dan diatur agar dapat menangani beban pengguna yang nyata.
3. **Tahap *Deployment***
   a. *Pre-Deployment*: tahap perencanaan yang melibatkan identifikasi sumber daya, konfigurasi perangkat keras dan perangkat lunak, serta penyusunan rencana untuk pengujian.
   b. *Deployment Process*: proses pemindahan aplikasi dari lingkungan pengembangan atau ujicoba ke lingkungan produksi. Hal ini termasuk pemasangan perangkat lunak, konfigurasi, dan migrasi data jika diperlukan.
   c. *Post-Deployment*: melibatkan pengujian lanjutan untuk memastikan bahwa aplikasi berfungsi dengan baik di lingkungan produksi. Ini juga mencakup pemantauan kinerja dan perbaikan jika ada masalah.
4. **Continuous Deployment dan *Continuous Integration* (CI/CD)**
   Dalam konteks pengembangan modern, praktik *Continuous Integration (CI)* dan *Continuous Deployment (CD)* sangat penting. CI berfokus pada mengintegrasikan perubahan kode secara teratur ke dalam repositori bersama dan menjalankan otomatisasi

pengujian, sementara CD berfokus pada otomatisasi pengimplementasian perangkat lunak setelah lolos dari pengujian CI.

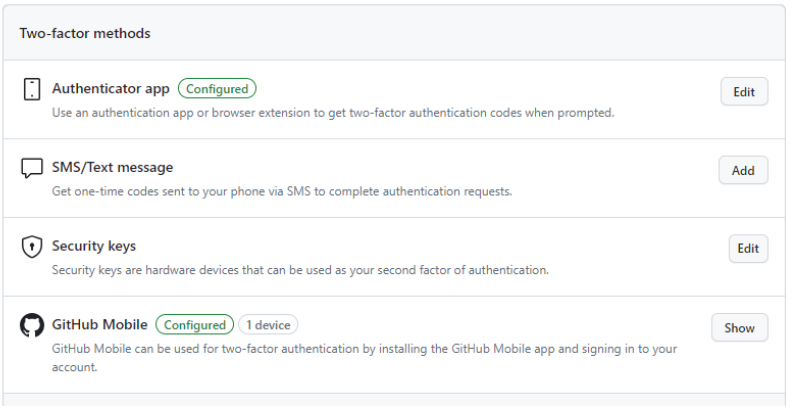
5. **Konfigurasi dan Manajemen Sumber Daya**
   Hal ini mencakup mengkonfigurasi server, jaringan, database, serta aspek keamanan seperti firewall dan izin akses. Manajemen sumber daya adalah tugas yang mencakup alokasi sumber daya, penskalaan otomatis, dan manajemen beban.
6. **Migrasi Data**
   Jika aplikasi melibatkan basis data, *deployment* seringkali melibatkan migrasi data dari lingkungan pengembangan ke lingkungan produksi. Hal ini perlu dilakukan dengan hati-hati untuk memastikan konsistensi dan integritas data.
7. **Pemantauan dan Manajemen Kinerja**
   Setelah pengimplementasian, aplikasi perlu dipantau secara berkala untuk memastikan kinerja yang baik dan mendeteksi masalah dengan cepat. Ini melibatkan penggunaan alat pemantauan dan manajemen kinerja.
8. **Skalabilitas dan Pemulihan Bencana**
   Rancangan *deployment* harus mempertimbangkan skenario peningkatan beban (skalabilitas) dan juga strategi pemulihan bencana untuk menjaga ketersediaan aplikasi.

LATIHAN

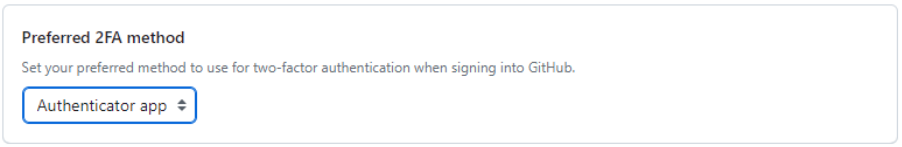
A. Version Control Menggunakan Git

a. Registrasi Github

1. Silakan lakukan instalasi akun Github melalui link berikut ini https://github.com/signup. Disarankan menggunakan email institusi atau kampus seperti @student.unp.ac.id
2. Setelah pembuatan akun, silakan login. Disarankan juga untuk melakukan pengaturan **Two-factor authentication.** Anda dapat melakukannya dengan mengakses *profile picture* pada bagian kanan atas, lalu pilih menu **Setting**
3. Selanjutnya pilih menu **Password and Authentication**, lalu pilih bagian **Two-factor authentication.** Anda dapat memilih berbagai jenis metode autentikasi seperti gambar berikut



4. Disarankan memilih menggunakan **Authenticator app** atau **GitHub Mobile**. Opsi lainnya juga diperbolehkan. Perlu diperhatikan, jika anda memilih SMS/Tex message pastikan bahwa nomor handphone yang anda gunakan selalu aktif.
5. Jika anda memilih **authenticator app,** anda diharuskan mendownload aplikasi autentikasi seperti **Google Authenticator** dan **Microsoft Authenticator.** Silakan download aplikasi ini melalui PlayStore atau AppStore di smartphone anda. Setelah mendownload aplikasi autentikasi, silakan login aplikasi tersebut menggunakan akun gmail (Google) atau outlook (Microsoft).
6. Setelah itu, silakan scan barcode yang tampil di halaman GitHub menggunakan aplikasi autentikator yang anda pilih. Akan muncul kode angka dan silakan masukan kode ini di halaman GitHub. Lalu, selesaikan proses aktivasi autentikasi tersebut.
7. Jika anda mengaktifkan beberapa aplikasi autentikator, maka anda dapat memilih metode autentikasi apa yang diinginkan ketika anda login ke GitHub melalui bagian **Preferred 2FA method**



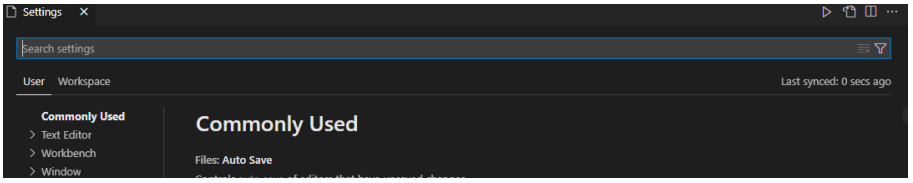
8. Autentikasi ini digunakan untuk melindungi akun anda sehingga setiap anda masuk anda akan diminta untuk memasukan kode autentikasi. Anda dapat mengakses kode autentikasi ini sesuai dengan jenis metode autentikasi yang anda pilih

b. Pengaturan Git Commit untuk Version Control

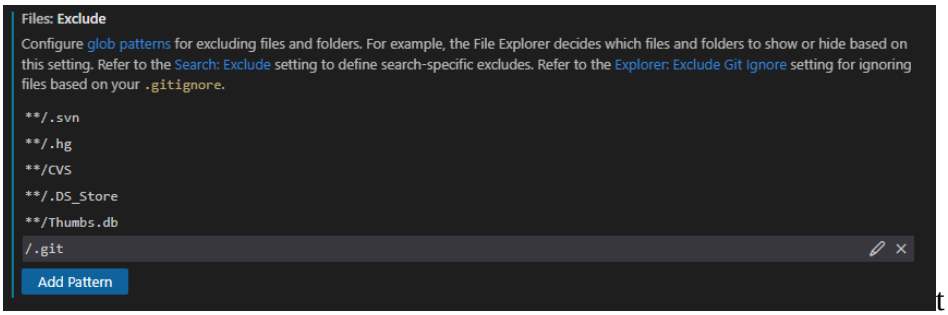
1. Silakan Download Git melalui link berikut ini https://git-scm.com/. Downloadlah sesuai sistem operasi anda
2. Lakukan instalasi Git. Setelah selesai, silakan ketikan perintah berikut **`git -version`** melalui terminal di visual studio code untuk memastikan instalasi telah berhasil

```
PS C:\Users\Randi Proska Sandra\Desktop\node-course\web-servers> git --version
git version 2.42.0.windows.2
```

3. Selanjutnya silakan pastikan anda berada pada direktori web-server (aplikasi yang anda kembangkan pada modul sebelumnya) dan ketikan perintah berikut ini **`git init`** melalui terminal. Perhatikanlah bagian explorer di visual studio code, bahwa folder-folder pada direktori web-server anda berubah menjadi warna hijau
4. Ketika anda melakukan perintah **`git init`** akan keluar informasi yang menyatakan bahwa git repository telah di inisialisasi
5. Klik menu File – Preferences – Setting pada visual studio code anda dan akan muncul tampilan berikut ini



6. Drag ke bawah atau cari bagian **`File:Exclude`** seperti gambar dibawah ini, dan hapuslah tanda bintang ** pada bagian **`**/.git`** sehingga menjadi **`/.git`**



7. Perhatikan pada bagian explorer, bahwa folder baru .git telah bertambah di direktori aplikasi web-server anda
8. Langkah tadi hanya untuk mengecek bahwa direktori tersebut telah ada dalam aplikasi anda. Silakan kembalikan lagi tanda ** sehingga folder tersebut tersembunyi
9. Selanjutnya jalankan perintah git status untuk mengecek status tracking folder, akan keluar info terkait semua folder dan file berwarna merah
10. Pada proses version control ini, kita tidak akan memasukan folder dan file **`node_modules`** karena folder ini berisi library yang kita butuhkan dan bukan bagian utama dari source code aplikasi yang selalu kita edit. Buatlah file baru dalam folder web-server dan beri nama **`.gitignore`** (tanda titik . wajib ada).
11. Masukanlah teks berikut **`node_modules/`** pada file **`.gitignore`**
12. Jalankan lagi **`git status`** dan perhatikan bahwa sekarang direktori dan file yang belum di track hanya tersisa

```
.gitignore
package-lock.json
package.json
public/
src/
```

`templates/`

13. Sedangkan folder **`node_modules`** akan berwarna abu-abu yang berarti folder tersebut akan dikecualikan
14. Selanjutnya pada terminal, ketik perintah **`git add src/`**
15. Jalankan lagi git status dan pastikan bahwa tidak ada direktori atau file yang belum di track (berwarna merah)
16. Jika masih ada jalankan perintah berikut **`git add .`**
17. Jalankan kembali git status dan pastikan bahwa tidak ada lagi yang berwarna merah atau belum ditrack.
18. Selanjutnya ketikan perintah **`git commit -m "Init commit"`** untuk menyimpan perubahan yang telah dilakukan dan perhatikan bahwa semua direktori sekarang tidak berwarna hijau lagi
19. Setelah proses ini, setiap kali anda mengedit baris kode pada file aplikasi anda, folder atau file tersebut akan berwarna orange yang mengindikasikan bahwa telah terjadi perubahan. Anda harus menjalankan perintah **`git status, git add .`** dan **`git commit -m "Init commit"`** kembali untuk melakukan sinkronisasi perubahan
20. Ketikan juga perintah berikut ini **`git commit -m "Remove unnecessary console.log call"`** untuk menghapus beberapa baris console.log yang tidak penting
21. Apabila semuanya telah selesai di commit menjadi proyek git dan tidak ada perubahan pada proyek aplikasi anda, maka setiap kali anda menjalankan perintah **`git commit`** akan keluar informasi berikut yang mengindikasikan bahwa tidak perubahan pada repository aplikasi anda

```
PS C:\Users\Randi Proska Sandra\Desktop\node-course\web-servers> git commit -m "Init commit"
On branch master
nothing to commit, working tree clean
```

c. Membuat SSH Key

1. Silakan buka terminal pada visual studio code dan pastikan anda berada pada direktori web-server
2. Silakan ketik perintah berikut ini untuk menampilkan list konten pada direktori ssh **`Get-ChildItem -Force -File ~/.ssh`**
3. Selanjutnya masukan perintah berikut ini **`ssh-keygen -t rsa -b 4096 -C`** **`randiproska@gmail.com`** untuk membuat RSA SSH Key pair. Silakan ganti email dengan email anda.
4. Hal ini akan meng-generate SSH key, biarkan semua nilai (valuenya) default dengan menekan enter hingga selesai atau tampil lebih kurang seperti gambar berikut

```
The key's randomart image is:
+---[RSA 4096]----+
|          ...+   |
|    .  .o.o* .. .|
|   . .. .ooo=.o=.|
| E... .. oo+ ++o=|
|..+o =  S +.. o=.|
|.o  B . .o.  o...|
|o    + +..    o .|
|.   .   o        |
|                 |
+----[SHA256]-----+
```

5. Selanjutnya jalankan lagi perintah **`Get-ChildItem -Force -File ~/.ssh`** dan perhatikan bahwa sebuah RSA key telah di generate dengan dua nama file yaitu **`id_rsa`** dan **`id_rsa.pub`**

```
    Directory: C:\Users\Randi Proska Sandra\.ssh


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/14/2023      8:30            120 config
-a----        10/18/2023     14:16           3389 id_rsa
-a----        10/18/2023     14:16            748 id_rsa.pub
```

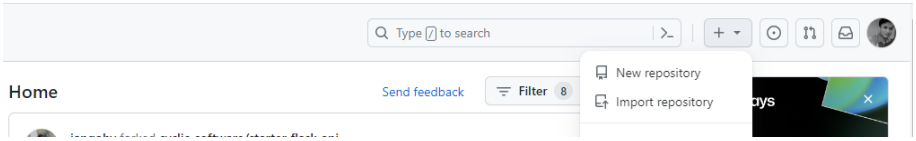
6. RSA SSH Key ini akan digunakan oleh github nantinya. Kemudian jalankan perintah berikut ini satu persatu

```
$env:SSH_AUTH_SOCK = ""
$env:SSH_AGENT_PID = ""
Start-Process ssh-agent -Wait -NoNewWindow
```

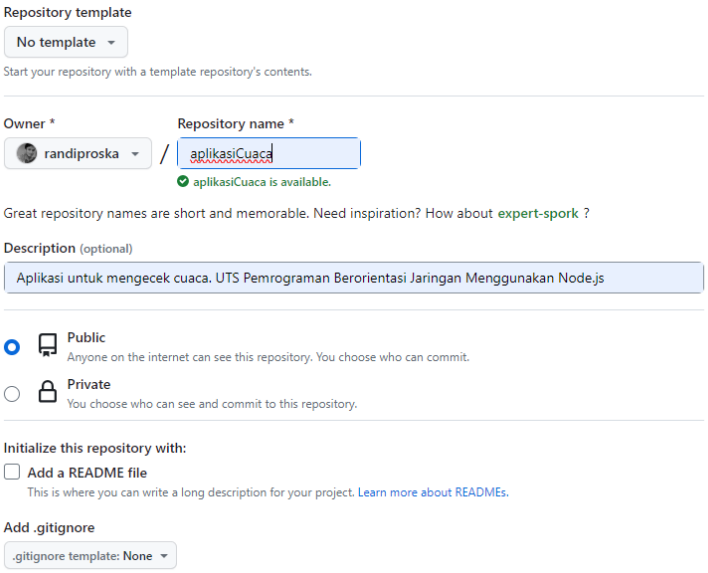
7. Kemudian jalankan perintah berikut ini **`ssh-add 'C:\Users\Randi Proska Sandra\.ssh\id_rsa'`** untuk menambahkan SSH private key ke SSH agent di Windows. Silakan sesuaikan direktori dengan direktori lokasi file id_rsa anda sesuai gambar pada nomor 5

d. Membuat Github Repository dan Push Aplikasi ke Github

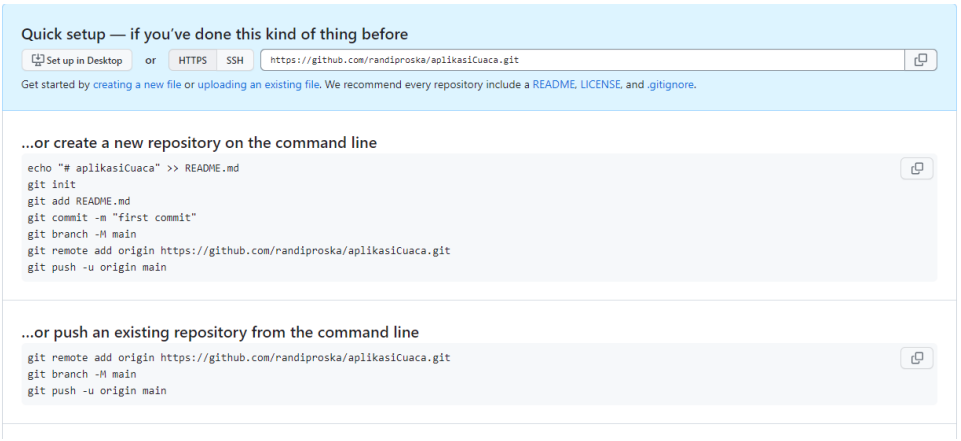
1. Silakan login pada akun github anda dan buat repository baru dengan mengklik ikon tambah, lalu pilih new repository pada bagian kanan atas



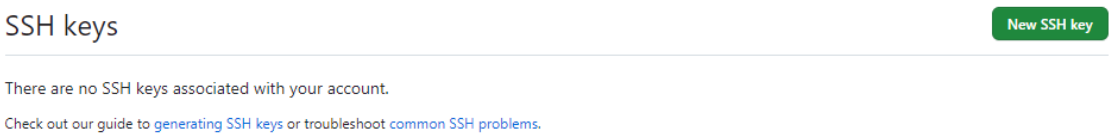
2. Silakan sesuai pengaturan seperti gambar berikut. Cukup ubah sesuai gambar (repository name, description, public), selebihnya biarkan default



3. Lalu klik Create Repository dan akan tampil tampilan seperti berikut ini yang berisi informasi tentang command-command untuk setup repository



4. Pada langkah selanjutnya kita akan memasukan aplikasi yang dibuat ke repository GitHub. Silakan masukan perintah berikut pada terminal di visual studio code **`git remote add origin https://github.com/randiproska/aplikasiCuaca.git`**
   Silakan sesuai nama akun dan nama aplikasi anda
5. Selanjutnya buka kembali github, klik ikon gambar anda kanan atas dan pilih setting, lalu SSH dan GPG Keys. Kemudian, pilih New SSH Key

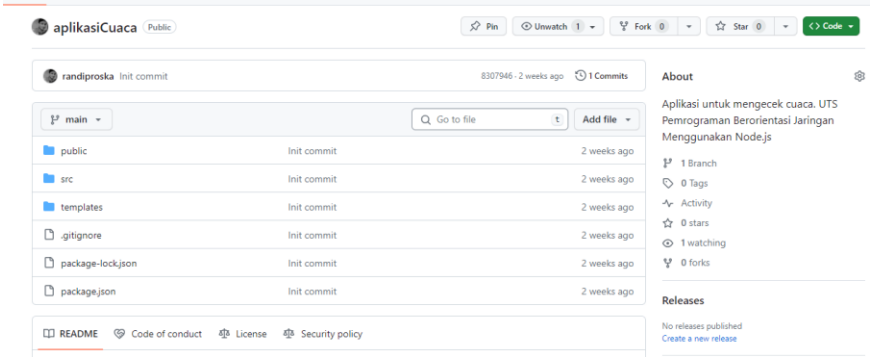


6. Lalu pada tampilan berikut silakan diisi judul SSH key

7. Silakan ketik perintah berikut **`cat ~/.ssh/id_rsa.pub`** pada terminal di visual studio code.
8. Copy kan semua kode yang muncul pada langkah 7 ke bagian key seperti pada gambar di langkah 6. Pastikan bahwa tidak ada satu karakter pun tertinggal.
9. Lakukan tes koneksi antara windows dan GitHub anda dengan memasukan perintah berikut pada terminal di visual studio code **`ssh -T git@github.com`**
10. Lalu, tekan **enter** dan pilih **yes**
11. Selanjutnya jalankan **`git branch -M main`** dan kemudian **`git push -u origin main`** seperti gambar berikut ini

```
PS C:\Users\Randi Proska Sandra\Desktop\node-course\web-servers> git branch -M main
PS C:\Users\Randi Proska Sandra\Desktop\node-course\web-servers> git push -u origin main
Enumerating objects: 37, done.
Counting objects: 100% (37/37), done.
Delta compression using up to 8 threads
Compressing objects: 100% (35/35), done.
Writing objects: 100% (37/37), 1.54 MiB | 770.00 KiB/s, done.
Total 37 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To github.com:randiproska/aplikasiCuaca
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

12. Perhatikanlah bahwa sekarang aplikasi anda telah terupload ke akun GitHub anda

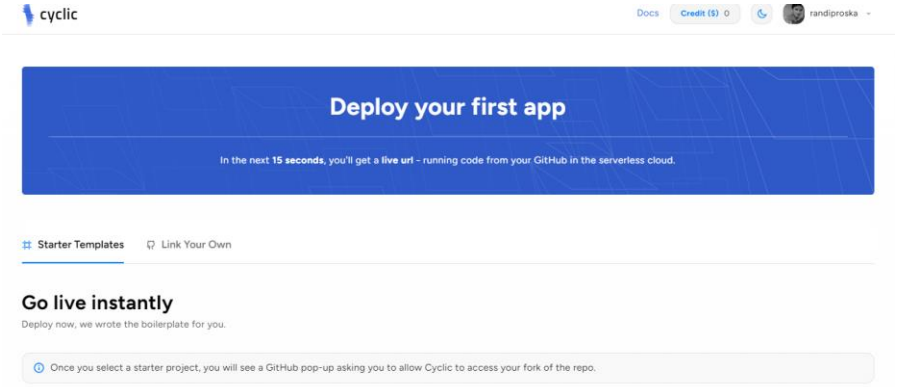


B. App Deployment menggunakan Cyclic

a. Registrasi Cyclic

Cyclic adalah platform cloud yang menyediakan cara mudah untuk menyebarkan (*deploy*) dan mengelola (*manage*) aplikasi yang dikembangkan oleh developer. Ikuti langkah-langkah berikut untuk registrasi

1. Silakan kunjungi halaman berikut ini https://www.cyclic.sh/, kemudian lakukan **sign up** menggunakan akun github dan setujui autentikasinya (jika diminta)
2. Setelah registrasi selesai akan tampil tampilan seperti berikut ini



3. Jika tampilan berbeda, silakan akses url berikut https://app.cyclic.sh/

b. Pengubahan File Sebelum Deployment

1. Silakan buka kembali direktori web-server anda melalui visual studio code.
2. Buka file package.json yang berada didalam direktori web-server dan gantilah kode bagian script menjadi seperti berikut ini

```
5   "main": "index.js",
    ▷ Debug
6   "scripts": {
7     "start": "node src/app.js"
8   },
9   "keywords": [],        You, 2 weeks ago • Init co
```

3. Untuk menguji bahwa aplikasi tetap berjalan normal, silakan jalankan perintah **`npm run start`** dan akses **`localhost:3000`** pada web browser anda
4. Lalu buka file **src/app.js,** dan tambahkan baris kode berikut **`const port = process.env.PORT || 3000`** seperti pada gambar

```
const app = express()
const port = process.env.PORT || 3000
//Define paths for Express config
const publicDirectoryPath = path.join(__dirname, '../public')
const viewPath = path.join(__dirname, '../templates/views')
const partialsPath = path.join(__dirname, '../templates/partials')
```

5. Pada kode app.listen bagian akhir, ubah menjadi seperti gambar berikut

```
112 app.listen(port, () => {
113     console.log('Server berjalan pada port '+ port)
114 })
```

6. Lalu buka lagi file **public/js/app.js** dan ubahlah url pada bagian **fetch** menjadi seperti gambar berikut ini

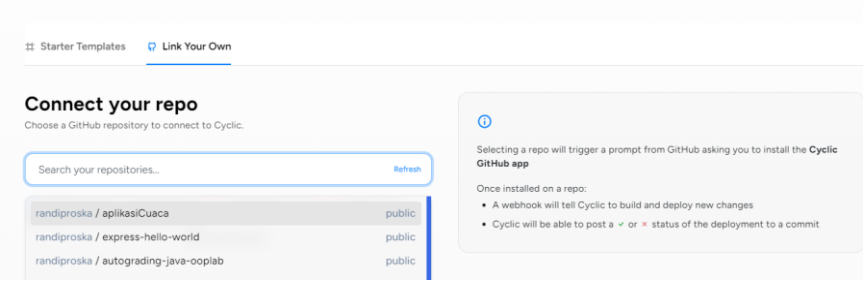
```
23     fetch('/infocuaca?address='+ location).then((response)=>{
24     response.json().then((data)=>{
25             if(data.error){
```

7. Jalankan kembali **npm run start** seperti langkah nomor 3 untuk menguji bahwa tidak ada masalah dengan program anda.
8. Jika telah selesai jalankan kembali perintah git dibawah ini **satu persatu** untuk mengupdate versi terbaru aplikasi anda ke github repository

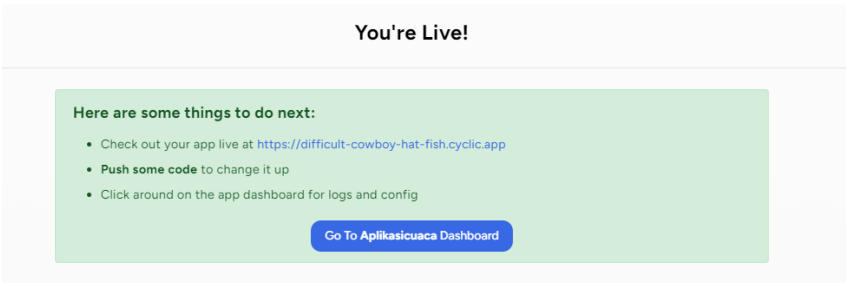
```
git status
git add .
git commit -m "Init commit"
git commit -m "Remove unnecessary console.log call"
git branch -M main
git push -u origin main
```

c. App Deployment

1. Pada gambar bagian a nomor 2, silakan pilih **Link Your Own**, lalu connectkan akun GitHub (jika diminta)
2. Jika akun telah terkoneksi maka akan muncul list repository anda seperti gambar berikut dan silakan pilih nama aplikasi yang telah di upload ke GitHub melalui proses version control



3. Nama repository pada modul ini adalah aplikasiCuaca, lalu pada tampilan selanjutnya silakan klik **Connect Cyclic.** Lalu tunggu proses selesai.
4. Setelah selesai, akan muncul informasi seperti gambar berikut



5. Silakan klik url tersebut dan website anda telah online melalui hosting cyclic

REFERENCES

1. Chacon, S., & Straub, B. (2014, November 9). *Pro Git* (2nd ed.). Apress.
2. Casciaro, M., & Mammino, L. (2020, July). *Node. Js Design Patterns* (3rd ed.) [Packtpub]. Packt Publishing. https://www.packtpub.com/product/nodejs-design-patterns-third-edition/9781839214110
3. Pasquali, S. (2013, November). *Mastering Node.js* [Packtpub]. Packt Publishing. https://www.packtpub.com/product/mastering-nodejs/9781782166320
4. Panchal, M. (2022, May 4). *What Is ExpressJS In Node JS? – Backend Framework For Web Apps.* Excellent Webworld. Retrieved October 1, 2023, from https://www.excellentwebworld.com/what-is-expressjs-in-node-js/
5. MDN (Mozilla Developer Network). (2023, July 3). *What is a web server?*. Mozilla Developer. Retrieved October 1, 2023, from https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server
6. OpenJS Foundation. (n.d.). *Express routing* (StrongLoop/IBM, Trans.). Expressjs. Retrieved October 1, 2023, from https://expressjs.com/en/guide/routing.html