SOAL:

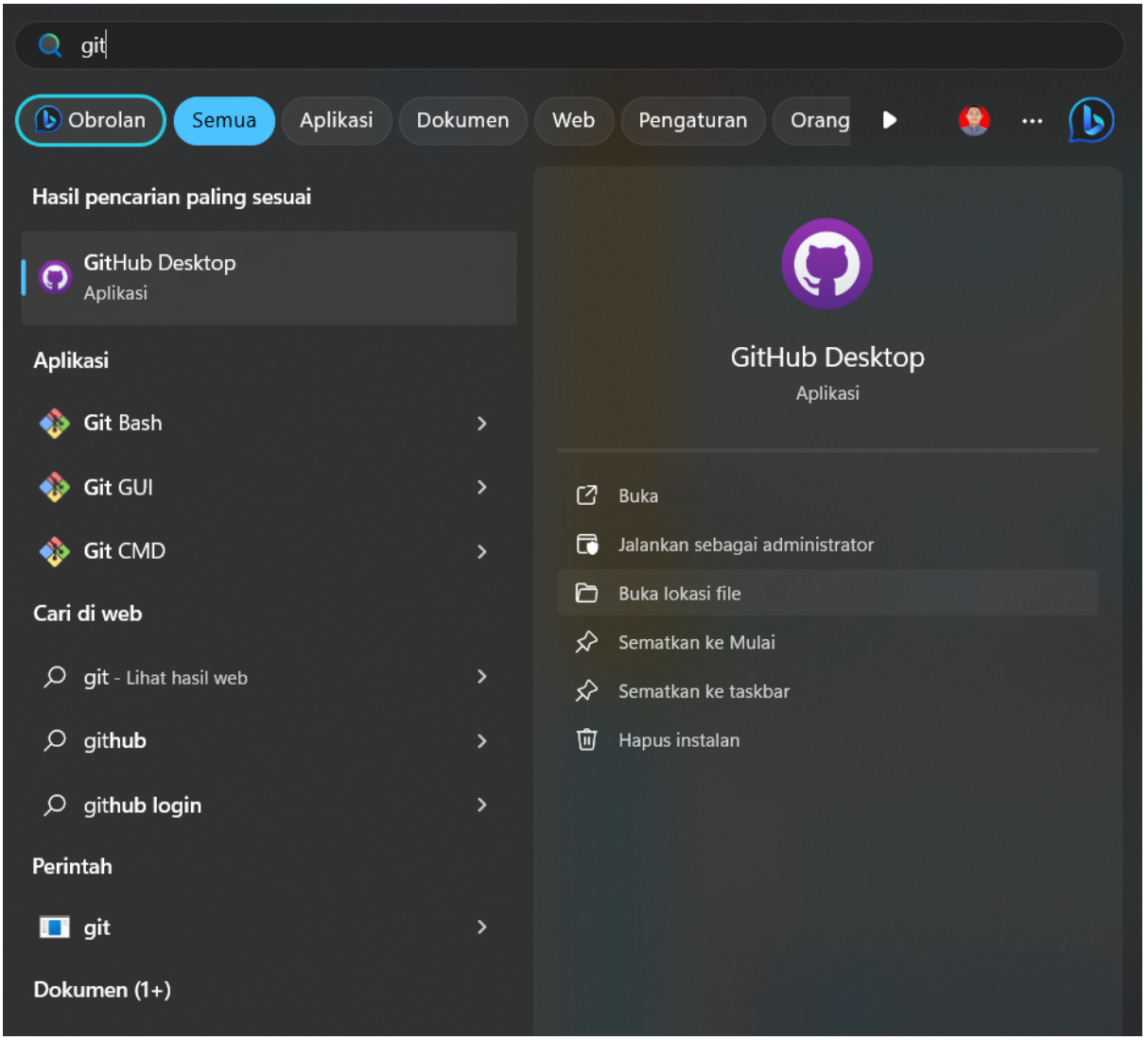
Silahkan teman teman melakukan proses instalasi beberapa tools dibawah ini :

A. Git
B. Visual Studio Code
C. Browser

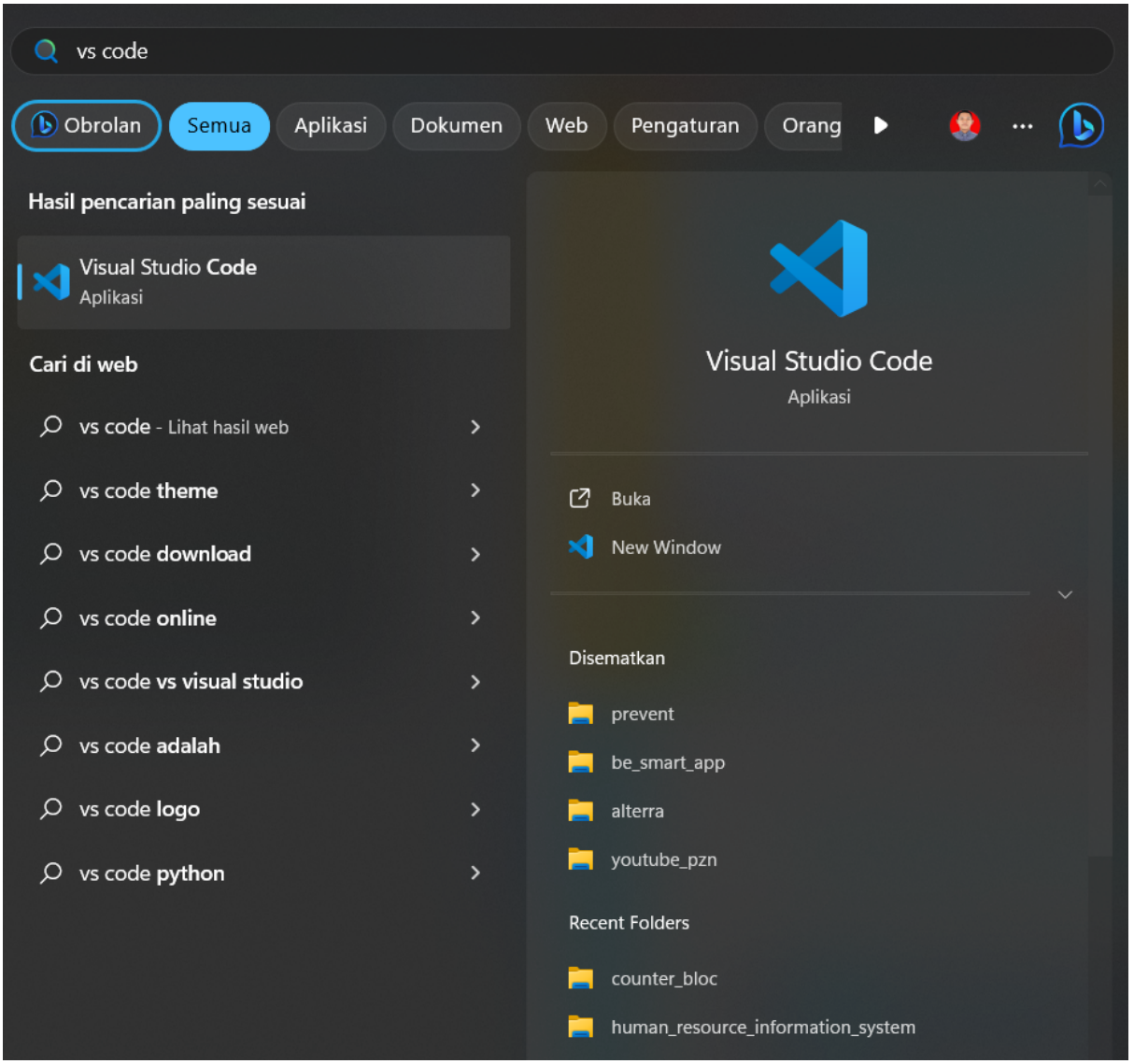
Setelah melakukan instalasi, buat sebuah summary untuk apa yang sudah dipelajari dalam materi pengenalan software engineer sebelumnya pada google docs.

JAWAB:

A. Git



B. Visual Studio Code



C. Browser

Summary Full Stack Developer Career Path:

1. **Introduction Full Stack Web/Mobile Developer**
   A. Scope Penting Full Stack Development

- Front End Development

  Membangun antarmuka pengguna yang menarik dan interaktif menggunakan HTML, CSS, dan JavaScript. Menggunakan framework dan pustaka front-end, seperti React, Angular, Vue.js, atau jQuery, untuk mempercepat pengembangan dan meningkatkan efisiensi.

  Menghubungkan komponen front-end dengan layanan back-end melalui API (Application Programming Interface) untuk berkomunikasi dengan server dan database. Menyelaraskan data dan tampilan antara sisi depan dan sisi belakang aplikasi.

- Back-End Development

  Membangun server dan aplikasi yang berfungsi sebagai "otak" dari aplikasi, menerima permintaan dari sisi depan, memproses data, dan memberikan respons yang sesuai. Menggunakan bahasa pemrograman server-side seperti Node.js, Python, Ruby, Java, PHP, atau C#.

- Database Management

  Mendesain dan mengelola basis data untuk menyimpan, mengambil, dan memanipulasi data aplikasi. Menggunakan teknologi database seperti MySQL, PostgreSQL, MongoDB, atau Firebase.

- Integration of Front End and Back End

  Menghubungkan komponen front-end dengan layanan back-end melalui API (Application Programming Interface) untuk berkomunikasi dengan server dan database. Menyelaraskan data dan tampilan antara sisi depan dan sisi belakang aplikasi.

- Version Control and Collaboration

  Menggunakan sistem pengendalian versi, seperti Git, untuk mengelola perubahan kode dan kolaborasi dalam tim pengembang. Memastikan kode terus berkembang dengan aman dan sesuai dengan tujuan proyek.

- Mobile Development

  Beberapa Pengembang Full Stack juga memiliki kemampuan untuk mengembangkan aplikasi mobile menggunakan framework seperti React Native, Flutter.

B. Dasar Dasar Front End Web Development
   - HTML
   - CSS
   - Javascript

C. Dasar Dasar Back End Development
   - Bahasa Pemrograman Server Side
   - Server Framework
   - Database Management

D. Dasar Dasar Database Management System
   - Database Management System
   - Tipe Database
   - Bahasa Query

E. Dasar Dasar Mobile Development
   - Platform Mobile
   - Integrated Development Environment (IDE)

2. Skillset Full Stack Web/Mobile Developer

A. Pengembangan Aplikasi End to End

Pengembangan Aplikasi End to End merupakan pendekatan pengembangan perangkat lunak yang mencakup keseluruhan siklus pembuatan aplikasi, dari tahap perencanaan hingga tahap pengujian dan implementasi. Tujuannya adalah untuk menghasilkan aplikasi yang lengkap, fungsional, dan siap digunakan oleh pengguna akhir.

B. Tahap - Tahap Pengembangan Aplikasi End to End

- Perencanaan dan Analisis

  Tahap awal ini melibatkan pengumpulan kebutuhan dan pemahaman mendalam tentang tujuan aplikasi, sasaran pengguna, dan lingkungan operasional. Analisis kebutuhan dan riset pasar dilakukan untuk mengidentifikasi fitur utama yang harus dimasukkan dalam aplikasi.

- Desain

  Proses desain melibatkan merancang antarmuka pengguna (UI) dan pengalaman pengguna (UX) yang intuitif dan menarik. Pengembang juga merencanakan arsitektur aplikasi, termasuk pemilihan teknologi, database, dan framework yang sesuai.

- Pengembangan Front-End

  Pada tahap ini, tim pengembang bekerja pada bagian depan aplikasi, menggunakan HTML, CSS, dan JavaScript untuk membuat tampilan dan interaksi yang menarik bagi pengguna. Pengembang mungkin juga menggunakan framework front-end seperti React, Angular, atau Vue.js untuk mempercepat pengembangan.

- Pengembangan Back-End

  Tahap ini melibatkan pengembangan sisi server dan logika bisnis aplikasi. Pengembang menggunakan bahasa pemrograman server-side seperti Node.js, Python, Ruby, atau Java, dan menggunakan framework back-end seperti Express.js, Flask, atau Ruby on Rails.

- Integrasi dan Pengujian

  Bagian depan dan belakang aplikasi harus diintegrasikan melalui API (Application Programming Interface) sehingga mereka dapat berkomunikasi dan berbagi data. Pengujian aplikasi dilakukan untuk memastikan semua fitur berfungsi dengan benar dan mengidentifikasi dan memperbaiki bug yang mungkin ada.

- Pemeliharaan dan Peningkatan

Setelah diluncurkan, aplikasi harus dipelihara dengan memperbaiki bug dan menangani perubahan lingkungan atau kebutuhan bisnis. Peningkatan terus menerus dilakukan untuk memperbarui fitur, meningkatkan kinerja, dan memastikan aplikasi tetap relevan dalam waktu yang berlanjut.

C. Kolaborasi Efektif Dengan Version Control

- Version Control

  Version control (pengendalian versi) adalah sistem yang memungkinkan pengembang perangkat lunak untuk melacak perubahan pada kode sumber aplikasi selama pengembangan. Ini memungkinkan kolaborasi yang efisien di antara anggota tim, terutama ketika banyak orang bekerja pada proyek yang sama. Berikut 2 Version Control yang cukup populer yaitu:

  - Git
  - Mercurial

- Beberapa Manfaat Version Control Untuk Berkolaborasi
  - Rekam Perubahan

    Setiap kali pengembang membuat perubahan pada kode, sistem version control merekam detail perubahan tersebut.

  - Pencatatan Riwayat

    Version control memungkinkan tim untuk melihat riwayat lengkap dari semua perubahan yang terjadi pada proyek dari awal hingga saat ini.

  - Pemecahan Konflik

    Ketika dua atau lebih pengembang melakukan perubahan pada area kode yang sama, version control membantu mengidentifikasi dan menyelesaikan konflik.

  - Pemulihan dengan mudah

Version control memungkinkan pengembang untuk memulihkan kode ke versi sebelumnya jika ada masalah atau bug yang terjadi, sehingga mengurangi resiko kehilangan pekerjaan.

- Penggunaan Version Control untuk Berkolaborasi
  - Inisialisasi Proyek

    Tim memulai proyek dengan membuat repository version control. Repository ini akan menyimpan semua kode sumber, file, dan perubahan yang dilakukan selama pengembangan.

  - Pengembangan Paralel

    Setiap anggota tim akan memiliki salinan repository pada komputernya sendiri. Mereka dapat bekerja secara paralel, membuat perubahan.

  - Branching

    Version control memungkinkan pembuatan cabang (branch) yang terpisah dari kode utama. Ini memungkinkan tim untuk mengisolasi perubahan dan fitur yang sedang dikembangkan.

  - Merge

    Setelah fitur atau perubahan selesai, cabang dapat digabungkan kembali ke cabang utama (biasanya disebut sebagai "merge").

  - Pull Request

    Di beberapa platform version control seperti GitHub, GitLab, dan Bitbucket, pull request adalah mekanisme yang memungkinkan pengembang untuk mengajukan perubahan mereka untuk ditinjau oleh anggota tim lain sebelum digabungkan ke cabang utama.

3. Tools Set Full Stack Web/Mobile Developer

Sebagai Pengembang Full Stack Web dan Mobile, Anda akan memerlukan kombinasi dari berbagai alat dan teknologi untuk secara efisien

membangun dan mengelola aplikasi. Berikut adalah beberapa set alat yang penting yang mungkin digunakan oleh Pengembang Full Stack:

- IDE - Code Editor
  - Visual Studio Code

- Version Control - Repository
  - GitHub
  - GitLab
  - Bitbucket

- Version Control - Git Tools
  - Sourcetree
  - GitLens

- Database Management System (DBMS)
  - PostgreSQL
  - MySQL
  - Oracle
  - MongoDB
  - Redis

- Application Programming Interface (API)
  - Postman
  - Swagger

- Tests dan Debugging
  - Jest
  - Mocha
  - JUnit

- Mobile Development
  - Framework React Native
  - Framework Flutter

- Layanan Cloud
  - AWS
  - Google Cloud
  - Azure

- CI/CD
  - Jenkins
  - Circleci

- Desain UI/UX
  - Adobe XD
  - Figma
  - Sketch

Summary SDLC & Design Thinking Implementation:

1. **What is SDLC**

SDLC (Siklus Hidup Pengembangan Perangkat Lunak) adalah rangkaian proses yang terstruktur dan metodologi yang digunakan untuk mengembangkan perangkat lunak dari awal hingga selesai. SDLC terdiri dari serangkaian tahap yang saling terkait dan dilakukan secara berurutan untuk memastikan bahwa pengembangan perangkat lunak berjalan dengan baik dan sesuai dengan kebutuhan dan tujuan yang ditentukan.

- Siklus SDLC
  1. Perencanaan dan Analisis

     Tahap pertama ini melibatkan identifikasi masalah atau kebutuhan bisnis yang perlu diselesaikan oleh perangkat lunak. Para pemangku kepentingan berinteraksi untuk mengumpulkan persyaratan dan menentukan ruang lingkup proyek.

     Rencana keseluruhan untuk proyek perangkat lunak dibuat. Rencana ini mencakup alokasi sumber daya, jadwal waktu, dan definisi tugas dan tanggung jawab anggota tim. misal: Seorang pelanggan ingin memiliki aplikasi yang melibatkan transaksi uang. Dalam hal ini,

persyaratannya harus jelas seperti transaksi apa yang akan dilakukan, bagaimana melakukannya, dalam mata uang apa akan dilakukan, dll.

2. Desain Produk

   Di tahap ini, perangkat lunak dirancang secara rinci berdasarkan persyaratan yang telah dikumpulkan. Desain mencakup arsitektur sistem, antarmuka pengguna, dan desain database.

3. Pengembangan Produk

   Tahap ini melibatkan implementasi perancangan perangkat lunak yang telah disetujui sebelumnya. Para pengembang menulis kode untuk menghasilkan produk perangkat lunak yang berfungsi.

4. Pengujian Produk

   Setelah perangkat lunak dikembangkan, tahap pengujian dilakukan untuk memastikan bahwa perangkat lunak berfungsi sesuai dengan persyaratan yang telah ditentukan. Pengujian mencakup verifikasi fungsionalitas, kinerja, keamanan, dan kualitas keseluruhan perangkat lunak.

5. Penerapan Produk

   Tahap ini melibatkan implementasi perancangan perangkat lunak yang telah disetujui sebelumnya. Para pengembang menulis kode untuk menghasilkan produk perangkat lunak yang berfungsi.

6. Pemeliharaan Produk

   Setelah perangkat lunak diimplementasikan, pemeliharaan dilakukan untuk memperbaiki bug, meningkatkan fitur, dan menjaga perangkat lunak agar tetap sesuai dengan perubahan kebutuhan bisnis.

- Manfaat Penggunaan SDLC Dan Kesimpulan
  - Prediktabilitas dan Pengendalian Proyek
  - Peningkatan Kualitas Perangkat Lunak
  - Pengelolaan Risiko yang Lebih Baik

- Efisiensi Tim dan Kolaborasi
- Memenuhi Kebutuhan Pengguna
- Penghematan Biaya dan Waktu
- Meningkatkan Pengawasan dan Evaluasi
- Peningkatan Dokumentas

Kesimpulan:

Dengan menggunakan SDLC secara efektif, organisasi dapat meningkatkan keberhasilan dan efisiensi dalam mengembangkan aplikasi, memastikan pengiriman produk berkualitas tepat waktu, dan memberikan nilai yang lebih besar bagi pelanggan dan stakeholder.

2. Model-Model Software Development Life Cycle (SDLC)

- Waterfall Model

  Waterfall model adalah model SDLC yang linier dan berurutan. Setiap tahap dalam model ini harus selesai sebelum memulai tahap berikutnya. Cocok untuk proyek dengan persyaratan yang jelas dan stabil.

- V-Shaped Model

  Model V-Shaped adalah model yang terkait erat dengan model waterfall, tetapi menekankan pada pengujian. Tahapan pengujian diwakili oleh garis miring "V", yang berarti bahwa setiap tahap pengembangan memiliki tahapan pengujian yang sesuai. Cocok untuk proyek dengan fokus pada kualitas tinggi.

- Prototype Model

  Model Prototype adalah model pengembangan perangkat lunak yang bertujuan untuk menciptakan prototipe atau contoh awal sebelum mengembangkan versi finalnya. Model ini fokus pada pemahaman kebutuhan pengguna dan mengumpulkan umpan balik untuk memastikan bahwa perangkat lunak akhir sesuai dengan ekspektasi dan persyaratan pengguna.

- Spiral Model

  Model ini menggabungkan elemen model spiral dengan pendekatan inkremental. Setiap siklus spiral membangun pada inkremental sebelumnya, menghasilkan perangkat lunak yang semakin berkembang dengan fitur yang lebih banyak setiap siklusnya.Cocok untuk proyek besar dan kompleks dengan banyak risiko.

- Iterative Incremental Model

  Model ini melibatkan pengulangan siklus pembangunan dan peningkatan perangkat lunak dalam tahapan-tahapan kecil. Setiap iterasi menambahkan lebih banyak fitur hingga produk akhir mencapai tingkat kesempurnaan yang diinginkan. Cocok untuk proyek dengan waktu dan anggaran yang terbatas.

- Big Bang Model

  Model Big Bang adalah model yang kurang terstruktur, di mana semua tahapan pengembangan dilakukan tanpa perencanaan yang detail. Pengembangan dimulai tanpa melakukan analisis dan perencanaan yang mendalam. Cocok untuk proyek kecil atau prototyping.

- Agile Model

  Model Agile adalah pendekatan kolaboratif dan iteratif yang berfokus pada pengiriman perangkat lunak secara berkala dan inkremental. Tim bekerja dalam sprint (iterasi singkat) dan selalu terbuka untuk perubahan persyaratan pengguna. Cocok untuk proyek dengan lingkungan yang dinamis dan persyaratan yang berubah-ubah.

Kesimpulan:

Setiap model SDLC memiliki kelebihan dan kelemahan tergantung pada jenis proyek dan kebutuhan organisasi. Pemilihan model SDLC yang tepat sangat penting untuk mencapai keberhasilan proyek pengembangan perangkat lunak.

3. **Design Thinking Implementation (Steps Design Thinking)**
    - Empathize: Understand User Needs
    - Define: Define the Problem
    - Ideate: Generate Ideas
    - Prototype: Build Quick and Iterative Solutions
    - Test: Gather User Feedback
    - Implement: Develop the Software

Summary Basic Git & Collaborating Using Git

1. **Terminal and IDE**
    - Sejarah Singkat Terminal

        Dengan perkembangan teknologi dan perangkat lunak, terminal tetap menjadi alat penting bagi para pengembang perangkat lunak, administrator sistem, dan pengguna teknis lainnya. Meskipun antarmuka grafis semakin canggih dan populer, terminal tetap memberikan fleksibilitas dan kekuatan untuk melakukan tugas-tugas khusus dan otomatisasi dalam lingkungan komputer modern.

    - Command Line Dasar

Command	Windows	Linux / macOS	Description
List Files	`dir`	`ls`	List files and directories in the current folder
Change Directory	`cd`	`cd`	Change current working directory
Make Directory	`mkdir`	`mkdir`	Create a new directory
Remove Directory	`rmdir`	`rm -r`	Remove a directory
Delete Files	`del`	`rm`	Delete files
Copy Files	`copy`	`cp`	Copy files
Move / Rename	`move`	`mv`	Move or rename files/directories

Command	Windows	Linux / macOS	Description
Display File	`type`	`cat`	Display file content
Display Text	`echo`	`echo`	Display text or enable/disable echoing of commands
List Processes	`tasklist`	`ps`	List running processes
Terminate Process	`taskkill`	`kill`	Terminate processes

Command	Windows	Linux / macOS	Description
Print Directory	N/A	`pwd`	Print the current working directory
Create File	N/A	`touch`	Create an empty file or update timestamp
Change Permissions	N/A	`chmod`	Change file permissions
Change Ownership	N/A	`chown`	Change file ownership
Search in Files	N/A	`grep`	Search for patterns in files
Display Manual	N/A	`man`	Display the manual page for a command
Execute as Admin	N/A	`sudo`	Execute a command with administrative privileges
Ping	N/A	`ping`	Send ICMP echo requests to check network connectivity
Network Config	`ipconfig`	`ifconfig`	Display network interface configurations
Secure Shell	N/A	`ssh`	Connect to a remote host using SSH

2. Installing, initializing and committing GIT

- Memahami Version Control Git

  Kontrol versi adalah metode yang digunakan untuk melacak dan mengelola perubahan dalam kode sumber atau berkas proyek. Git merupakan salah satu sistem kontrol versi terdistribusi yang paling populer dan kuat. Berikut adalah langkah-langkah untuk memahami kontrol versi dan Git.

- Apa itu GIT?

  Git adalah sistem kontrol versi terdistribusi yang memungkinkan pengembang perangkat lunak untuk melacak perubahan dalam kode mereka, berkolaborasi dengan anggota tim, dan mengelola revisi kode secara efektif. Topic ini akan mempelajari menginstal Git pada berbagai sistem operasi, menginisialisasi repositori Git baru, dan melakukan commit pertama.

- Dasar Dasar Command GIT

- git init
  Menginisialisasi direktori sebagai repositori Git kosong.

- git clone
  Menduplikasi repositori Git yang sudah ada ke direktori lokal.

- git status
  Menampilkan status perubahan yang belum di commit di repositori lokal.

- git add
  Menambahkan perubahan ke area persiapan (staging area) untuk disiapkan menjadi commit.

- git commit
  Membuat commit dari perubahan yang sudah di-staging dan menambahkan pesan commit.

- git push
  Mengirimkan commit ke repositori jarak jauh(remote repository).

- git pull
  Mengambil commit terbaru dari repositori jarak jauh dan menggabungkannya ke repositori lokal.

- git branch
  Menampilkan daftar cabang (branch) yang ada di repositori dan menunjukkan cabang aktif.

- git checkout
  Beralih ke cabang lain atau ke commit tertentu.

- git merge

Menggabungkan perubahan dari satu cabang ke cabang aktif.

- git log
  Menampilkan daftar commit beserta riwayatnya dalam repositori.

- git remote
  Menampilkan daftar repository jarak jauh yang terhubung dengan repositori lokal.

- git fetch
  Mengambil informasi terbaru dari repositori jarak jauh tanpa menggabungkan perubahan.

- git diff
  Menampilkan perbedaan antara versi yang sudah di-staging dengan versi sebelumnya.

- git reset
  Mengembalikan file yang sudah di-staging ke direktori kerja sebelumnya.