**GitHub Enterprise** is a version of GitHub designed for organizations that need a more secure, scalable, and customizable Git hosting solution. It provides the core features of GitHub, such as code repositories, pull requests, and collaboration tools, but with enhanced options to meet the needs of enterprises. GitHub Enterprise can be deployed on-premises or used in a cloud-based setup, allowing flexibility in how organizations manage their development environments.

**Key Features of GitHub Enterprise:**

1. **Self-Hosting or Cloud Hosting**:
   * **GitHub Enterprise Server**: The on-premises version of GitHub Enterprise, which organizations can host on their own infrastructure or private cloud.
   * **GitHub Enterprise Cloud**: The cloud-hosted version, which is fully managed by GitHub and hosted on GitHub’s infrastructure (using the same architecture as GitHub.com).
2. **Security**:
   * **SAML Authentication & Single Sign-On (SSO)**: GitHub Enterprise integrates with enterprise identity providers for single sign-on (SSO), allowing easier management of user access across systems.
   * **Advanced Security Features**: Includes two-factor authentication (2FA), security advisories, and audit logs for compliance purposes.
   * **Branch Protection Rules**: Enforces rules on branches, such as requiring reviews or CI checks before merging.
   * **Security Alerts and Vulnerability Scanning**: Automated alerts for vulnerable dependencies within repositories.
3. **Scalability and Performance**:
   * GitHub Enterprise is optimized for large organizations with extensive teams and codebases.
   * High availability setups and disaster recovery options are available to ensure reliability in enterprise environments.
4. **Customization**:
   * **Webhooks & API Integrations**: GitHub Enterprise supports custom integrations using webhooks, GitHub Apps, and its REST API, enabling organizations to tailor the platform to their workflows.
   * **Custom Branding and User Interface**: Allows enterprises to customize the look and feel of their GitHub instance to match their corporate identity.
5. **Collaboration and Code Management**:
   * **GitHub Actions**: Integrated CI/CD pipelines directly into the GitHub workflow.
   * **Pull Requests, Issues, Discussions**: Facilitates collaboration with built-in tools for code reviews, bug tracking, and feature discussions.
   * **Code Scanning**: Analyzes code for security vulnerabilities directly within the repository.
6. **Compliance and Auditing**:
   * **Audit Logs**: Detailed logs of user actions, repository changes, and system access to meet compliance needs.
   * **Role-Based Access Control (RBAC)**: Granular access control to ensure the right users have the appropriate permissions for various repositories and actions.
7. **Enterprise Support**:
   * GitHub Enterprise comes with enterprise-grade support from GitHub’s team, including options for 24/7 support, training, and more.
   * **Managed Services**: For users who prefer a fully managed experience, GitHub offers managed services to help with setup, maintenance, and troubleshooting.

**Deployment Options:**

* **GitHub Enterprise Server**: Installed on-premises or within a private cloud. The organization has full control over hardware, networking, and maintenance.
* **GitHub Enterprise Cloud**: A fully managed, cloud-hosted solution where GitHub handles scaling, infrastructure, and updates.

**Use Cases for GitHub Enterprise:**

1. **Large-Scale Organizations**: Enterprises with hundreds or thousands of developers who need a centralized platform for managing code, collaboration, and CI/CD.
2. **Regulatory Compliance**: Companies in industries like finance, healthcare, or government that require strict security, privacy, and audit capabilities.
3. **Collaboration at Scale**: Teams that work on large projects across multiple locations and require advanced version control, security, and integration capabilities.
4. **Custom Workflows**: Organizations needing tailored GitHub features, integrations, and configurations specific to their development environment.

**Pricing:**

GitHub Enterprise has a subscription-based pricing model that depends on the number of users and whether the organization opts for cloud or self-hosted solutions. Pricing for GitHub Enterprise Server generally involves a licensing fee, along with an option for paid support services, while GitHub Enterprise Cloud typically involves a per-user or per-seat fee.

**Conclusion:**

GitHub Enterprise is designed to provide a seamless and secure Git experience for large organizations with complex needs. Whether you're managing a large-scale project with extensive security and compliance requirements or simply looking for advanced collaboration tools, GitHub Enterprise offers a robust and flexible solution.

**🚀 Key Differences: GitHub Enterprise vs GitHub.com**

| **Feature** | **GitHub.com (Free/Pro/Team)** | **GitHub Enterprise** |
| --- | --- | --- |
| **Authentication** | Standard OAuth, personal access tokens, and 2FA. | **SAML single sign-on (SSO)**, **SCIM provisioning**, and **Enterprise Managed Users (EMU)**. |
| **Access Control** | Limited to repository-level permissions. | **Organization-wide policies**, **IP allow lists**, and **role-based access controls (RBAC)**. |
| **Compliance** | Basic security features for public repos. | **Audit logs**, **SOC 2**, **GDPR**, **HIPAA** compliance, and **data residency controls**. |
| **Repository Management** | Basic settings for branches, merges, and webhooks. | **Required reviews**, **protected branches**, and **custom repository roles**. |
| **Deployment Controls** | Basic CI/CD via GitHub Actions. | Enhanced deployment security with **environment protection rules**, **deploy keys**, and **secrets scanning**. |
| **Administration** | Limited org-level settings. | **Enterprise Account**, centralized billing, and **Admin Center** with advanced user management. |
| **Support** | Community support and basic help. | **24/7 support**, **SLA guarantees**, and dedicated account managers. |
| **Integration** | Limited integrations. | **Webhooks**, REST/GraphQL APIs, and advanced integrations with Jira, Azure, and more. |
| **Security** | Dependabot alerts and basic security features. | **Advanced Security** with **code scanning**, **secret scanning**, and **dependency reviews**. |
| **Backup & Disaster Recovery** | Limited data recovery options. | Automated backups, **business continuity solutions**, and **high availability**. |
| **Compliance & Auditing** | Basic audit history. | **Audit logs**, **compliance reports**, and **custom security policies**. |

INITIALIZING UR DIRECTORY :

git config --global user.email “raman@example.com”

git config --global user.name “raman.khanna”

286 mkdir raman

287 ls

288 cd raman/

289 ls

290 git status

291 git init

292 git status

ADDING A FILE :

ls -la

295 cd .git/

296 ls

297 cd ..

298 ls

299 pwd

300 touch file1

301 ls

302 git status

303 git add .

304 git status

COMMITING TO REPO :

git status

308 git commit -m "first file commited"

309 git status

310 git log

vi file1

317 ls

318 cat file1

319 git status

320 git add file1

321 git status

322 git commit -m " added content in file1"

323 git status

git log

REMOTE REPO :

86 git remote add origin https://github.com/ramannkhanna2/myrepo.git

87 git push origin master

88 git status

--add a file on github manually and try to pull it

331 git diff 4950ad7174b02ffd8cb158ecd0b12bc327e61890 526591fe5cc497fbbef2e19c70f775539190df64

341 git pull origin master

342 ls

343 cat file-remote

git clone https://github.com/admingagan/test.git

**What is a Branch?**

In GitHub (and Git in general), a **branch** is like a separate "workspace" where you can work on a specific task or feature without affecting the main code. Imagine you're writing a book, and you want to try a new chapter idea. Instead of messing up the original manuscript, you make a copy of the page and write the new chapter there. If you like it, you can add it back to the original manuscript later.

* **Main Branch**: The main or default branch in Git is often called main (or master). This is like your book’s main content that everyone is reading.
* **Feature Branch**: A branch you create for working on new features, bug fixes, or experiments.

**How to Create a Branch?**

You create a branch when you want to work on something new, like adding a new feature or fixing a bug.

1. **Using GitHub Web Interface:**
   * Go to the repository (the project) you’re working on.
   * In the top-left corner, click the dropdown that says **Branch: main**.
   * Type the name of your new branch (e.g., feature/add-login-page).
   * Press **Enter**, and your new branch is ready!
2. **Using Git on Your Computer (Git CLI):**
   * If you’re working with Git on your computer, run this command:

bash

CopyEdit

git checkout -b feature/add-login-page

* + This creates and switches to your new branch.

BRANCHING :

git branch

348 git branch 2ndbranch

349 git branch

350 ls

351 cat file1

352 git status

353 git switch 2ndbranch

354 git --help

355 clear

356 git checkout 2ndbranch

357 git branch

358 ls

359 cat file

360 cat file1

361 vi file1

362 cat file1

363 git status

364 git add .

365 git status

366 git commit -m "changed file1 contents on 2ndbranch to test"

367 ls

368 git status

369 git checkout master

370 ls

371 cat file1

372 git checkout 2ndbranch

373 cat file1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*