**Day 8 Task: Basic Git & GitHub for DevOps Engineers.**

**What is Git?**

Git is a version control system that allows you to track changes to files and coordinate work on those files among multiple people. It is commonly used for software development, but it can be used to track changes to any set of files.

With Git, you can keep a record of who made changes to what part of a file, and you can revert back to earlier versions of the file if needed. Git also makes it easy to collaborate with others, as you can share changes and merge the changes made by different people into a single version of a file.

**What is Github?**

GitHub is a web-based platform that provides hosting for version control using Git. It is a subsidiary of Microsoft, and it offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. GitHub is a very popular platform for developers to share and collaborate on projects, and it is also used for hosting open-source projects.

**What is Version Control? How many types of version controls we have?**

Version control is a system that tracks changes to a file or set of files over time so that you can recall specific versions later. It allows you to revert files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.

There are two main types of version control systems: centralized version control systems and distributed version control systems.

1. A centralized version control system (CVCS) uses a central server to store all the versions of a project's files. Developers "check out" files from the central server, make changes, and then "check in" the updated files. Examples of CVCS include Subversion and Perforce.
2. A distributed version control system (DVCS) allows developers to "clone" an entire repository, including the entire version history of the project. This means that they have a complete local copy of the repository, including all branches and past versions. Developers can work independently and then later merge their changes back into the main repository. Examples of DVCS include Git, Mercurial, and Darcs.

**Why we use distributed version control over centralized version control?**

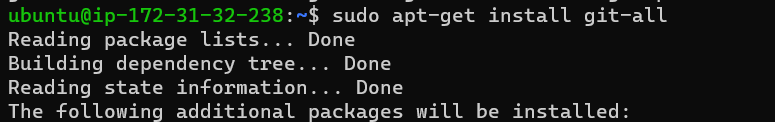
1. Better collaboration: In a DVCS, every developer has a full copy of the repository, including the entire history of all changes. This makes it easier for developers to work together, as they don't have to constantly communicate with a central server to commit their changes or to see the changes made by others.
2. Improved speed: Because developers have a local copy of the repository, they can commit their changes and perform other version control actions faster, as they don't have to communicate with a central server.
3. Greater flexibility: With a DVCS, developers can work offline and commit their changes later when they do have an internet connection. They can also choose to share their changes with only a subset of the team, rather than pushing all of their changes to a central server.
4. Enhanced security: In a DVCS, the repository history is stored on multiple servers and computers, which makes it more resistant to data loss. If the central server in a CVCS goes down or the repository becomes corrupted, it can be difficult to recover the lost data.

Overall, the decentralized nature of a DVCS allows for greater collaboration, flexibility, and security, making it a popular choice for many teams.

**Task:**

* Install Git on your computer (if it is not already installed). You can download it from the official website at <https://git-scm.com/downloads>

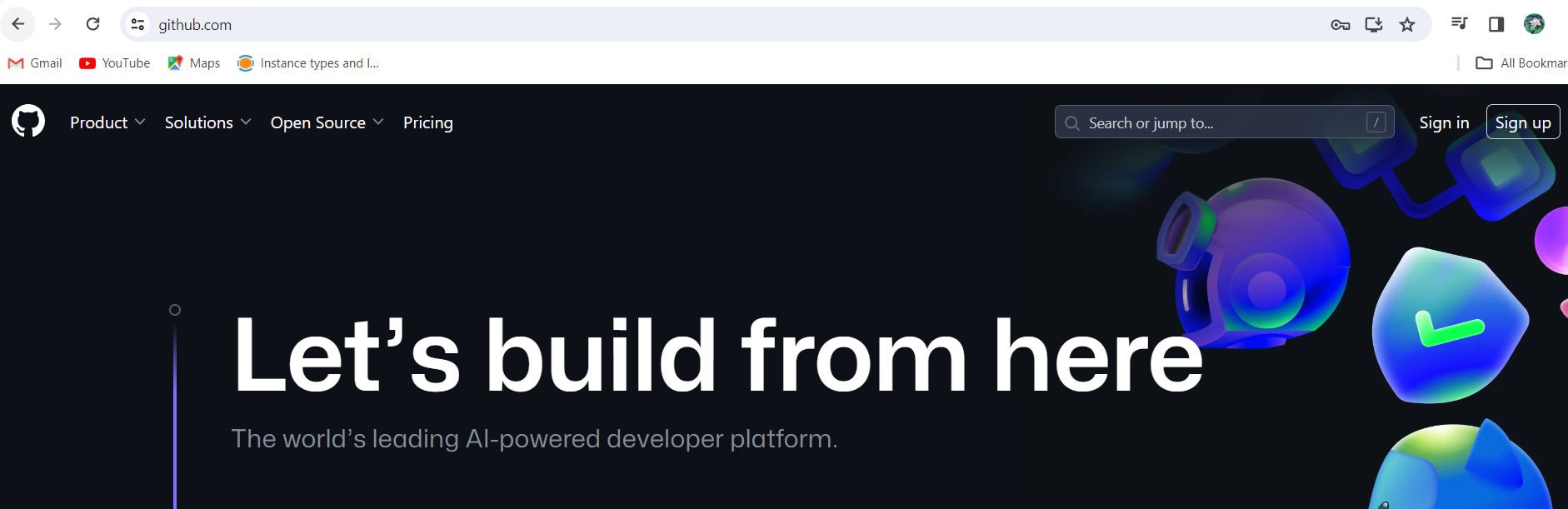






**TASK: create free account on github**

To create a free account on GitHub, follow these steps:



1. Go to the GitHub website at <https://github.com/>.
2. Click on the **"Sign up"** button in the top-right corner of the page.
3. Enter your desired **username, email address** and **password**.
4. Choose to sign up for a **free individual account**.
5. Complete the **CAPTCHA** verification to prove you are not a robot.
6. Click on the **"Create account"** button.
7. GitHub will send you a verification email to confirm your email address. Open the email and click on the link to verify your account.
8. Once your account is verified, you can customize your profile and start using GitHub.

**TASK : Create a new repository on GitHub and clone it to local machine, Make some changes to a file in the repository and commit them to the repository using Git also Push the changes back to the repository on GitHub**

Steps to create a new repository on GitHub, clone it to your local machine, make changes to a file, commit the changes, and push them back to the repository on GitHub,

1. Log in to your GitHub account.
2. Click on the **"+"** icon in the top right corner of the page and select **"New repository"**.
3. Give your repository a name and a brief description, select whether it should be public or private, and then click on **"Create repository"**.
4. On your local machine, open a terminal or command prompt and navigate to the directory where you want to clone the repository.
5. Copy the repository's URL by clicking on the green **"Clone or download"** button on the GitHub repository page.
6. Run the following command, replacing "**repository URL**" with the Actual-URL you just copied, This will clone the repository to your local machine.

