গিট ও গিটহাব

জাকির হোসাইন

অদম্য প্রকাশ

ভূমিকা

আমরা যারা প্রোগ্রামিং করি, তারা দেখা যায় কোনো একটি আইডিইতে কোড লিখছি। ফাইল আকারে কোডগুলো সেভ হচ্ছে আমার লোকাল পিসিতে। কোনো এক সফটওয়্যার বানানো একদিনের কাজ নয়, অনেক দিন, মাস বা বছরও লেগে যায়। প্রতিদিন আমরা নির্দিষ্ট পরিমাণ কোড করতে পারি। ভাবুন তো, এক মাস আগে কী কোড করেছিলাম সেটা কি আমরা মনে রাখতে পারব? বা এক সপ্তাহ আগেও কী করেছি আমরা ভুলে গেছি হয়তো। আবার ধরুন, একটা প্রজেক্টে কাজ করছে অনেকজন সফটওয়্যার ইঞ্জিনিয়ার, কোনো প্রজেক্টে ৫ জন, ১০ জন, ২০ জনও কাজ করছে। সবার কোডগুলো এক জায়গায় কীভাবে আসে? পুরো ব্যাপারটা হয় গিট দিয়ে। গিট হচ্ছে এমন একটি প্রক্রিয়া, যা দিয়ে সবগুলো সমস্যার সমাধান করা যায়। তা ছাড়া ধরুন, আপনার একাধিক পিসি, আবার অফিসে বাসায় বসে সেইম প্রজেক্টে কাজ করেন। সব জায়গায় কোডের সিনক্রোনাইজেশন কীভাবে হবে? গিট দিয়েই সম্ভব। গিট সিস্টেম ব্যবহার করার জন্য অনেকগুলো প্ল্যাটফর্ম আছে, যেমন গিটহাব, বিটবাকেট, গিটল্যাব ইত্যাদি। আপনি চাইলে নিজেও একটি গিট সিস্টেম তৈরি করতে পারেন।

গিট কী? যদি বুকিশ সংজ্ঞা চিন্তা করি, গিট একটি সফটওয়্যার, যেটা একটা ফাইলসেটের মধ্যে পরিবর্তন ট্র্যাক করে হিসাব রাখে এবং সফটওয়্যার ডেভেলপারদের কোডের মধ্যে কোলাবরেশন করতে সাহায্য করে। গিট প্ল্যাটফর্মগুলো ব্যবহার করে আমরা আমাদের সফটওয়্যারের কোড খুব সিকিউরডওয়েতে স্টোর করতে পারি, পুল করতে পারি, মার্জ করতে পারি।

প্রোগ্রামিং জানার সঙ্গে সঙ্গে এখন গিট জানা একটি অপরিহার্য স্কিল। আমরা কিন্তু এখন যখন কোনো সফটওয়্যার প্রোগ্রামারের ইন্টারভিউ নিই, আমরা জিজ্ঞেসও করি না আপনি গিট পারেন কি না, আমরা ধরে নিই এটা ব্যাসিক স্কিল। এটা জানতেই হবে, কোনো অলটারনেটিভ নেই। গিট আমরা যত সহজে বলে ফেললাম এর চেয়ে বড় কিছু, আরও গভীর কিছু। গিট সম্পর্কে জানতে, গিটকে নিজের স্কিলসেটে যোগ করতে এই বইটি হতে পারে আপনার দুর্দান্ত সঙ্গী।

জাকির হোসাইন আমার এক যুগেরও পুরোনো বন্ধু। তিনি তার ভ্রমণ প্রোগ্রামিংবিষয়ক লেখার জন্য বাংলাদেশের আইটি কমিউনিটিতে বেশ জনপ্রিয় একটি মুখ। তিনি গিট নিয়ে লিখেছেন শুনে বেশ আনন্দিত হয়েছি। বাংলাদেশে গিটের চমৎকার একটি বইয়ের অভাব ছিল। তিনি সেই অভাবটি মোচন করবেন বলে আমাদের ধারণা। জাকির হোসাইন ও তার গিট নিয়ে বইয়ের জন্য শুভকামনা।

শুভেচ্ছান্তে, মোশাররফ হোসেইন রুবেল

প্রতিষ্ঠাতা, কিউটেক সল্যুশন লিমিটেড, ঢাকা, বাংলাদেশ

লেখকের কথা

বর্তমানে গিট ছাড়া যে কোন সফটওয়্যারের প্রজেক্ট ম্যানেজমেন্ট বলা যায় প্রায় অসম্ভব। টেক ইন্ড্রাস্ট্রিতে গিট জানা থাকা গুরুত্বপূর্ণ একটা স্কিল হিসেবে দেখা হয়। সফটওয়্যার ডেভেলপমেন্ট সম্পর্কিত যেকোনো ইন্টার্ভিউতে গিট জানা আছে কিনা তা জিজ্ঞেস করা হয়। কোন প্রজেক্টে গিট ব্যবহার করে প্রজেক্ট ম্যানেজ করলে প্রোডাক্টিভিটি এবং ইফিসিয়েন্সি বহুগুণ বেড়ে যায়।

এই বই থেকে গিট সম্পর্কে জানা যাবে। জানা যাবে কিভাবে গিট ব্যবহার করে যে কোন ধরনের প্রজেক্ট ম্যানেজ করা যায়। যারা একবার গিটের মজা পেয়ে যায়, তারা পরবর্তীতে দেখা যায় সাধারণ টেক্সট ডকুমেন্টও গিট ব্যবহার করে!

বইটি গল্পের বইয়ের মতো না। পড়ার পাশাপাশি প্র্যাকটিস করতে হবে। কেউ যদি বইটি পড়ার পাশাপাশি গিট কমান্ডগুলো প্র্যাকটিস করে, তাহলে সহজেই গিট সম্পর্কে জানতে পারবে।

সূচিপত্র

অধ্যায় ১

ভার্সন কন্ট্রোল

এই অধ্যায়েয় আমরা যা শিখব:

- ভার্সন কন্ট্রোল সম্পর্কে ধারণা
- লোকাল ভার্সন কন্ট্রোল সিস্টেম
- সেন্ট্রালাইজড ভার্সন কন্ট্রোল সিস্টেম
- ডিস্ট্রিবিউটেড ভার্সন কন্ট্রোল সিস্টেম
- VCS-এ যে সুবিধাগুলো থাকা উচিত

ভার্সন কন্ট্রোল সম্পর্কে ধারণা

একটা সফটওয়্যার তৈরি করতে প্রচুর সময় লাগে। আবার একটা স্ট্যান্ডার্ড সফটওয়্যার ডেভেলপমেন্টে অনেক ডেভেলপার কাজ করে। একজন ডেভেলপার একটা সফটওয়্যারে কাজ করলে খুব একটা সমস্যা হতো না। সাধারণত কোডগুলোতে কমেন্ট লিখে বুঝতে পারত কেন কোন কোড লিখেছে বা কোন কোডের কী কাজ। ভার্সনের জন্য নির্দিষ্ট সময় পর পর একটা কপি অন্য কোথাও রাখলেই কাজ সহজ হয়ে যায়। কিন্তু যখন একের অধিক ডেভেলপার একটা প্রজেক্টে কাজ করে, সমস্যাটা তখনই তৈরি হয়। কে কোন ফাইলে কাজ করে, কোন ফিচারে কাজ করে এসব ট্র্যাক রাখার একটা সিস্টেমের দরকার হয়। আবার সবার কোডগুলো মার্জ করার দরকার পড়ে। তো এই যে সফটওয়্যারের কোড বা সোর্সগুলোর ট্র্যাক রাখা এবং ম্যানেজ করার জন্য যে সিস্টেমের দরকার হয়, তাই হচ্ছে ভার্সন কন্ট্রোল সিস্টেম।

লোকাল ভার্সন কন্ট্রোল সিস্টেম

ভার্সন কন্ট্রোলের না থাকলে আমরা কীভাবে কোন প্রজেক্টে কাজ করতাম? প্রতিদিন বা নির্দিষ্ট দিন পর পর পুরো প্রজেক্টের একটা কপি অন্য একটা ডিরেক্টরিতে বা কোনো রিমুভাল ডিস্কে রাখতাম। এর অসুবিধা হচ্ছে প্রজেক্ট বড় হলে প্রতিদিন এর কপি রাখতে প্রচুর স্পেসের দরকার হয়। এ ছাড়া যদি কোনো কারণে ভুল ফাইল কপি

করি, তাহলে অনেক ডেটাই হারিয়ে যাবে। পরবর্তী সময়ে এই সমস্যা সমাধানের জন্য একটা ভার্সন কন্ট্রোল সিস্টেম তৈরি করেছে যেখানে একটা ডেটাবেজের মাধ্যমে ফাইলগুলোর ট্র্যাক রাখা হতো। একটা ফাইলের বিভিন্ন ভার্সন টাইমস্ট্যাপ অনুযায়ী সেভ রাখত। এ ধরনের ভার্সন কন্ট্রোলকে বলা যায় লোকাল ভার্সন কন্ট্রোল সিস্টেম।

সেন্ট্রালাইজড ভার্সন কন্ট্রোল সিস্টেম

লোকাল ভার্সন কন্ট্রোল সিস্টেমে একজন ডেভেলপার কাজ করলে তা-ও মোটামুটি কাজ করা যায়। কিন্তু একের অধিক ডেভেলপার কাজ করতে গেলে দুইজনের কাজ সমন্বয় করা দুষ্কর হয়ে দাঁড়ায়। এই সমস্যা সমাধানের জন্য ডেভেলপাররা নতুন ধরনের ভার্সন কন্ট্রোল সিস্টেম তৈরি করে নিল, যাকে বলা যায় সেন্ট্রালাইজড ভার্সন কন্ট্রোল সিস্টেম। এখানে একটা সেন্ট্রাল লোকেশনে মূল প্রজেক্টের একটা কপি থাকে। যেখানে প্রজেক্টের সবগুলো ভার্সন থাকে। কেউ যদি কাজ করতে চায়, এই সেন্ট্রাল লোকেশন থেকে কপি নিয়ে কাজ করে আবার সেন্ট্রাল লোকেশনে সাবমিট করতে হয়। এ ছাড়া কে কোন ফাইলে কী ধরনের অ্যাকসেস পাবে, একজন এডমিন তা ঠিক করে দিতে পারে। তাই প্রজেক্ট রক্ষণাবেক্ষণ করা আগেকার লোকাল ভার্সন কন্ট্রোল সিস্টেম থেকে অনেক বেশি সুবিধা পাওয়া যায় এই সিস্টেমে।

ডিস্ট্রিবিউটেড ভার্সন কন্ট্রোল সিস্টেম

সেন্ট্রালাইজ ভার্সন কন্ট্রোল সিস্টেমেও কিছু ইস্যু রয়ে যায়। যেমন কোনো কারণে সেন্ট্রাল লোকেশন বা সার্ভার ক্র্যাশ করলে পুরো ডেটা হারানোর ভয় থাকে। সার্ভার ডাউন থাকলে সবাইকে অপেক্ষা করতে হয়। এসব সমস্যা সমাধান করার জন্য ডেভেলপাররা আরও বেটার ভার্সন কন্ট্রোল সিস্টেম তৈরি করেছে। যাকে বলা যায় ডিস্ট্রিবিউটেড ভার্সন কন্ট্রোল সিস্টেম। ডিস্ট্রিবিউটেড ভার্সন কন্ট্রোলগুলোতে সবার কাছেই পুরো রিপোজিটরির একটা ক্লোন থাকে। সেন্ট্রাল লোকেশনের ওপর নির্ভর বা অন্য কারও ওপর নির্ভর না করেই যে কেউ কাজ করতে পারে। এমনকি অফলাইনেও কাজ করতে পারে। পরে যেকোনো সময় অন্যদের সঙ্গে সিঙ্ক্রোনাইজ করে নিতে পারে।

বর্তমানের প্রায় সব ভার্সন কন্ট্রোল সিস্টেম বা VCS গুলো সফটওয়্যার ডেভেলপমেন্টের ক্ষেত্রে কোডের প্রতিটা পরিবর্তন ট্র্যাক রাখতে সাহায্য করে। আবার কোন পরিবর্তনটা কেন করা হয়েছে, তা কমেন্ট আকারে লিখে রাখার সুযোগ করে দেয়। কোনো কারণে ভুল করলে আগের কোনো ভার্সনে ফিরে যাওয়ার সুবিধা থাকে। ভার্সন কন্ট্রোল সিস্টেমকে সোর্স কন্ট্রোল সিস্টেমও বলা হয়। কয়েকটি জনপ্রিয় ভার্সন কন্ট্রোল সিস্টেম হচ্ছে Git, Mercurial, SVN, CVS ইত্যাদি।

সবগুলো VCS-এর উৎপত্তি হচ্ছে বিভিন্ন প্রোগ্রামের সোর্স কোডগুলো সুন্দর মতো ম্যানেজ করার লক্ষ থেকে। প্রোগ্রামার বা ডেভেলপাররা যখন একটা সফটওয়্যার ডেভেলপ করে, তখন অনেক কোড লিখতে হয়। আগের কোড বাদ দিতে হয়। নতুন কোনো ফিচার যুক্ত করার জন্য লিখতে হয় নতুন কোড। দেখা গেল নতুন কিছু যুক্ত করতে গিয়ে আগের কোডের সঙ্গে সামঞ্জস্যপূর্ণ না হওয়াতে পুরো প্রোগ্রামই কাজ করা বন্ধ করে দিল। যদি কোনো ভার্সন কন্ট্রোল সিস্টেম না থাকত, তখন আগের ভার্সনে ফিরে যাওয়া যেত না। এ ছাড়া প্রোগ্রামের বাগ ফিক্স করাসহ অন্যান্য কিছুর ট্র্যাক রাখার জন্যও VCS ব্যবহার করা হয়।

ধরে নিচ্ছি আমাদের কাছে একটা ফাইল রয়েছে। ওই ফাইলের প্রথম ভার্সনে আমরা কিছু যুক্ত করেছি। এরপর দ্বিতীয় ভার্সনে ওই ফাইলটিতে আরও কিছু পরিবর্তন এনেছি। তৃতীয় ভার্সনে আরও কিছু পরিবর্তন এনেছি। VCS-এর কাজ হচ্ছে সব ভার্সনের ট্র্যাক রাখা। এবং আমরা চাইলে ফাইলটির যেকোনো ভার্সনে ফিরে যেতে পারি।

VCS-এ যে সুবিধাগুলো থাকা উচিত

যেকোনো স্ট্যান্ডার্ড ভার্সন কন্ট্রোল থেকে যে সুবিধাগুলো আমরা আশা করতে পারিঃ

হিস্টরিঃ প্রজেক্টের যেকোনো ফাইলের সম্পূর্ণ এডিট হিস্টোরি। প্রজেক্টে একটা ফাইল তৈরি থেকে শুরু করে নতুন কন্টেন্ট যোগ করা, কোওনা কন্টেন্ট মুছে ফেলা, পুরো ফাইল ডিলেট করাসহ সব ধরনের পরিবর্তনগুলোর ইতিবৃত্ত থাকবে। এ ছাড়া এই পরিবর্তনগুলো কে করেছে, তার নাম এবং কখন পরিবর্তনগুলো করেছে, তাও থাকবে। এ ছাড়া প্রতিটা এডিট কেন করেছে, তার একটা বিবরণও থাকবে, পরবর্তী যেকোনো সময় যে কেউ বুঝতে পারে এডিটগুলো কী কারণে করা হয়েছে। এভাবে সম্পূর্ণ হিস্টোরির ট্র্যাক থাকার ফলে যেকোনো সময় প্রজেক্টের আগের কোনো অবস্থায় ফিরে যাওয়া যায়। প্রতিটা পরিবর্তন ট্রেস করার সক্ষমতা আরেকটা গুরুত্বপূর্ণ ফিচার ভার্সন কন্ট্রোল সিস্টেমগুলোর। সম্পূর্ণ হিস্টোরি যেকোনো বাগ ট্রেস করতে সাহায্য করে। কোনো সমস্যা দেখা দিলে এই সমস্যাটা খুঁজে বের করার জন্য ট্রেসিং গুরুত্বপূর্ণ ভূমিকা পালন করে।

ব্রাঞ্চিং এবং মার্জিংঃ যে ভার্সন কন্ট্রোল সিস্টেমের দরকারি ফিচার হচ্ছে এই ব্রাঞ্চিং এবং মার্জিং। এই ফিচার দুইটি ছাড়া VCS চিন্তা করা যায় না। একটা প্রজেক্টে একজন ডেভেলপার কাজ করুক বা একের অধিক, ব্রাঞ্চিং এবং মার্জিং ফিচার ডেভেলপমেন্ট প্রসেসকে অনেক সহজ করে দেয়। যে কেউ চাইলে নতুন একটা ব্রাঞ্চ তৈরি করে কাজ করতে পারে। এই নতুন ব্রাঞ্চে চাইলে নিজের মতো করে কোড করতে পারে। কোডগুলো সঠিকভাবে কাজ করলে মূল ব্রাঞ্চের সঙ্গে মার্জ করতে পারে অথবা ব্রাঞ্চ ডিলেট করে দিতে পারে।

অধ্যায় ২

গিট

এই অধ্যায়ে আমরা যা শিখবঃ

- গিট সম্পর্কে ধারণা
- গিটের সুবিধাসমূহ
- গিট রিপোজিটরির তিন অবস্থা
- গিট সম্পর্কে ধারণা
- গিট কাদের জন্য
- গিট যেভাবে শিখব

গিট সম্পর্কে ধারণা

যে কয়েকটি ভার্সন কন্ট্রোল সিস্টেম সফটওয়্যার রয়েছে, সেগুলোর মধ্যে সবচেয়ে জনপ্রিয় হচ্ছে গিট। এটি একই সঙ্গে আধুনিক, সবচেয়ে বেশি ব্যবহৃত এবং সবচেয়ে বেশি সুবিধা সংবলিত ভার্সন কন্ট্রোল সিস্টেম। গিটের উৎপত্তিটা সুন্দর।

২০০০ সালের দিকে সোর্স কন্ট্রোল সিস্টেমের জন্য ওই সময় দারুণ একটা সোর্স কোড ম্যানেজমেন্ট সিস্টেম ছিল BitKeeper SCM নামে। যা ছিল ডিস্ট্রিবিউটেড ভার্সন কন্ট্রোল সিস্টেম। এবং এটা ছিল একটা প্রোপ্রাইটরি সফটওয়্যার। যার কমিউনিটি ভার্সন ছিল ফ্রি। আর ২০০২ থেকে ২০০৫ পর্যন্ত লিনাক্স কার্নেলের সোর্স কোডের ভার্সন কন্ট্রোল করার জন্য এই BitKeeper SCM ব্যবহার করা হয়েছিল। লিনাক্স কার্নেল হচ্ছে একটা ওপেনসোর্স সফটওয়্যার কিন্তু বিটকিপার হচ্ছে ক্লোজড সোর্স প্রোপ্রাইটরি সফটওয়্যার। তখন অনেকেই বলছিল যদি বিটকিপার নিজেদের রুল পরিবর্তন করে যদি ফ্রি না দেয় ভবিষ্যতে তখন কী হবে? এবং ফাইনালি তাই ঘটল। ২০০৫ সালের এপ্রিলে কমিউনিটি ভার্সন তুলে দেওয়া হলো।

ফ্রি ভার্সন তুলে দেওয়ার পর লিনাক্স কার্নেলের ভার্সন কন্ট্রোল করার জন্য একটা সিস্টেমের দরকার পড়ল। ফ্রি অন্যান্য ওপেনসোর্স যে ভার্সন কন্ট্রোলগুলো ছিল, সেগুলো লিনাস টরভল্ডসের পছন্দ হয়নি। এরপর ২০০৫ সালের এপ্রিলে সে নিজেই একটা ভার্সন কন্ট্রোল সিস্টেম লিখে ফেলল। পরবর্তী সময়ে যার নাম দেওয়া হলো গিট। বলে রাখা ভালো যে লিনাস টরভল্ডস হচ্ছে লিনাক্স কার্নেলের জনক। যে লিনাক্স কার্নেল লিখে সবার জন্য উন্মুক্ত করে দিল। তার তৈরি ভার্সন কন্ট্রোল সিস্টেমও যে ফ্রি এবং ওপেন সোর্স হবে তা তো বোঝাই যায়। গিট ফ্রি এবং ওপেন সোর্স।

গিটের সুবিধাসমূহ

গিটে অনেক সুবিধা পাওয়া যায়। যেমনঃ

ডিস্ট্রিবিউটেড

গিট হচ্ছে ডিস্ট্রবিউটেড ভার্সন কন্ট্রোল সিস্টেম। তার আগে জেনে নিই ডিস্ট্রিবিউটেড ভার্সন কন্ট্রোল সিস্টেম কী।

আগের ভার্সন কন্ট্রোলগুলোতে কাজ করার জন্য প্রজেক্টের একটা মাস্টার কপি থাকত। যাকে বলা যায় সেন্ট্রাল কোড রিপোজিটরি মডেল। কোনো ফাইলে কাজ করার জন্য মাস্টার ফাইল থেকে একটা কপি নিয়ে কাজ করে আবার সেন্ট্রাল রিপোজিটরিতে সাবমিট করতে হতো। অন্য আরেকজন ইউজার যদি মাস্টার ফাইলে কোনো পরিবর্তন আনে, তা নিজের ভার্সনে রিফ্লেক্ট করতে হলে প্রথমে ওই ইউজারকে সেন্ট্রাল কপিতে সাবমিট করতে হতো। এরপর সেন্ট্রাল কপি থেকে নিজের কপিতে আনার জন্য পুল করতে হতো।

গিট কাজ করে অনেক আধুনিক এবং স্মার্টভাবে। সেন্ট্রাল রিপোজিটরির পরিবর্তে প্রতিটা ইউজার তার নিজস্ব রিপোজিটরিতে কাজ করে। গিট প্রতিটা রিপোজিটরির পরিবর্তনগুলো ট্র্যাক করে। এরপর একজন ইউজার চাইলে এই পরিবর্তনগুলো অন্য যেকোনো রিপোজিটরির সঙ্গে বিনিময় করতে পারে বা অন্য কোনো রিপোজিটরির পরিবর্তনগুলোর সঙ্গে নিজের রিপোজিটরির মার্জ করতে পারে।

গিট প্রজেক্টের ক্লোন সবার কাছেই থাকে। মানে হচ্ছে সবার কাছেই মূল সোর্স কোডের কপি থাকে। এমনকি প্রজেক্টের শুরু থেকে সবগুলো পরিবর্তনের হিস্টোরি থাকে। অন্য কারও ওপর ডিপেনডেন্ট না হয়েই প্রজেক্টে কাজ করতে পারে। আর এই ডিস্ট্রিবিউটেড সুবিধার জন্য সেন্ট্রাল কপির সঙ্গে কমিউনিকেট করতে হয় না। তাই দ্রুত কাজ করা যায়। কাজ করার জন্য নেটওয়ার্কের দরকার হয় না। অফলাইনেও কাজ করা যায়।

স্ন্যাপশট

গিট কাজও করে একটু ভিন্ন ভাবে। যেমন অন্যান্য মেজর ভার্সন কন্ট্রোল সিস্টেম পরিবর্তনগুলো ফাইলে সেভ করে রাখে। গিট পরিবর্তনগুলোর একটা স্ন্যাপশট নিয়ে রাখে। বলতে পারি যতবারই আমরা কমিট করি, ততবার গিট ফাইলগুলোর স্টেটের একটা ছবি তুলে রাখে। ফাইলগুলো কী অবস্থায় ছিল, ওই অবস্থার স্ন্যাপশট তুলে রাখে এবং একটা রেফারেন্স সেভ করে রাখে। এতে সুবিধা হচ্ছে যদি কোনো ফাইলে পরিবর্তন না দেখে, তাহলে দ্বিতীয়বার আর স্ন্যাপশট নিতে হয় না। আগের স্ন্যাপশটের রেফারেন্স নাম্বারের সঙ্গে লিংক করে দেয়।

অফলাইন সুবিধা

আরেকটা দারুণ সুবিধা হচ্ছে গিট রিপোজিটরিতে কাজ করতে ইন্টারনেটের প্রয়োজন হয় না। অফলাইনেও কাজ করা যায়। সব কাজ লোকালি করা যায়। এরপর সুযোগ হলে রিমোট রিপোজিটরিতে আপলোড করা যায়। অনেক বড় প্রজেক্টে অনেক ডেভেলপারের সঙ্গে কাজ করতে গেলেও কারও জন্য কাউকে অপেক্ষা করতে হয় না।

সিকিউরিটি

গিট রিপোজিটরির সবকিছু SHA1 হ্যাশ অ্যালগরিদম ব্যবহার করে সিকিউর করা হয়। এখানে সবকিছু বলতে যেকোনো ফাইল, একটা ফাইলের সঙ্গে আরেকটা ফাইলের সম্পর্ক, ডিরেক্টরিগুলো, প্রতিটা ভার্সন, কমিট ইত্যাদি। এর ফলে যেকোনো ম্যালেশিয়াস পরিবর্তন থেকে কোডকে প্রটেক্ট করা সহজ হয়।

গিট রিপোজিটরির তিন অবস্থা

গিট রিপোজিটরিতে ফাইলগুলো তিন অবস্থায় থাকতে পারে।

- Modified
- Staged
- Committed

Modified: এখানে মডিফাইড মানে হচ্ছে ফাইলের ডেটা আমরা পরিবর্তন করেছি কিন্তু এখনো কমিট করিনি।

Staged: Staged মানে হচ্ছে কোনো কোনো ফাইলে পরিবর্তন হয়েছে, সেগুলো গিটে যুক্ত করেছি কিন্তু এখনো কমিট করিনি এমন অবস্থা।