**গিট এন্ড গিটহাব নোট:**

**১. ইন্সটলেশন:**

প্রথমে গিটহাবে একাউন্ট খুলতে হবে এবং লোকাল মেশিনের সাথে গিটহাবের কানেকশন করানোর জন্য গিট সফটওয়ার লোকাল মেশিনে ইন্সটল করতে হবে।

**২. গিটহাব কনফিগারেশন:**

ইন্সটল হয়ে গেলে এবার কনফিগার করার পালা। এর মাধ্যমে গিট কে বোঝানো হয় যে আমরা কোন ইমেইল ও কোন ইউজার নেম ব্যবহার করে গিট অপারেট করবো। এর সাথে গিটহাবে লগইন করার ইউজারনেম ও ইমেইলের কোন সম্পর্ক নেই। তবে গিট ও গিটহাবের ইমেইল ও ইউজার নেম কমন হওয়া ভালো। তবে একটি বিষয় অবশই মনে রাখতে হবে যে গিট কনফিগার করতে হবে রুট ডিরেক্টোরিতে অর্থাৎ টার্মিনাল অন করেই কনফিগার করতে হবে। কোন রকম ডাইরেক্টরি পরিবর্তন করা যাবে না।

**২.১ কনফিগার করার কমান্ডলাইনস:**

ভার্সন চেক: *git - -version*

ইউজারনেম কনফিগার করা: *git config* - -*global user.name “UserName”*

ইউজার ইমেইল কনফিগার করা: *git config* - -*global user.email “user@email.com”*

কনফিগার বিবরণ দেখা: *git config* - -*list*

**৩. গিটহাবের রিপোজিটরি ক্লোন করা:**

গিটহাবের কোন ফোল্ডারকে লোকাল মেশিনে ক্লোন করা বা ডাউনলোট করার জন্য ক্লোন পদ্ধতি ব্যবহার করা হয়। এর জন্য প্রথমে লোকাল মেশিনের যে স্থানে বা ফোল্ডারের মধ্যে রিমোট মেশিন বা গিটহাবের ফোল্ডার/রিপোজিটরি ক্লোন করা হবে আগে সেখানে ওয়ার্কিং ডাইরেক্টরি সেট করতে হবে।

**৩.১ ক্লোনের সাথে সম্পর্কিত কমান্ডলাইনস:**

ওয়ার্কিং ডাইরেক্টরি টার্গেট/চেঞ্জ করা: *git cd ‘Name of the Directory/’ [এখানে cd অর্থ হলো change Directory]*

ওয়ার্কিং ডাইরেক্টরিতে গিটহাব রিপো ক্লোন করা: *git clone ‘HTTPS path of github repository’*

অপারেশন স্ট্যাটার দেখা: *git status*

দেখা যাবে যে গিটহাবের ফোল্ডারটি হুবহু আমদের লোকাল মেশিনের টার্গেট ডাইরেক্টরিতে এসে গেছে। এবার ওয়ার্কিং ডাইরেক্টোরি টি ঐ নতুন ক্লোন ফোল্ডারে নিতে হবে কারণ এবার সব কাজ ঐ ফোল্ডারের মধ্যেই হবে।

ওয়ার্কিং ডাইরেক্টরি টার্গেট/চেঞ্জ করা: *git cd ‘Name of the cloned directory/’*

ওয়ার্কিং ডাইরেক্টরির মধ্যে কন্টেন্ট দেখা: *ls*

Initiator ফোল্ডারসহ ওয়ার্কিং ডাইরেক্টরির মধ্যে কন্টেন্ট দেখা: *ls -a*

**৪. ক্লোন ফোল্ডারের মধ্যে পরিবর্তন করা গিটকে ট্রাক করা বা রেকর্ড করার কমান্ড দেওয়া:**

সবসময় মনে রাখতে হবে যে ক্লোন ফোল্ডারটি রিমোট রিপোজিটরি এর একটি প্রতিকল্প মাত্র। এখানে কোন নতুন ফাইল বানালে সেটা Untracked অবস্থায় যাবে সৃষ্ট কোন ফাইলে পরিবর্তন আনলে সেটা প্রথমে Modified অবস্থায় থাকবে। এর পর লোকাল রিপোতে Add করলে সেটা Staged অবস্থায় যাবে। এর পর Commit করলে লোকাল রিপোতে সেভ হয়ে যাবে। এখন দেখা যাক কিভাবে পরিবর্তনটা গিটের মেমোরিতে ধরে রাখবো।

**৪.১ সম্পর্কিত কমান্ডলাইনস:**

যাবতীয় কাজ শেষ করার পর ফাইলগুলো স্টেজে তোলা: *git add ‘file name’ or ‘.’ (dot-that refers everything)*

স্টেজে তোলা কাজ কে লোকাল রিপোতে সেভ করা: *git commit -m “Meaningful message”*

অপারেশন স্ট্যাটার দেখা: *git status*

ওয়ার্কিং ডাইরেক্টরির মধ্যে কন্টেন্ট দেখা: *ls*

**৫. লোকাল রিপোতে সেভ করার পর সেগুলো রিমোট রিপোতে তুলে দেওয়া:**

আমরা যে কাজগুলো করলাম সেগুলো করলাম আমাদের লোকাল রিপোতে। কিন্তু সেগুলো তো রিমোট রিপোতে সেভ করা হলো না। এটা কেবল আমাদের লোকাল রিপোতেই আছে। এখন সব পরিবর্তন আমাদেরকে লোকাল রিমোট রিপো বা গিটহাবের ঐ ফোল্ডারে সেব করতে হবে। তাহলে আসুন যানা যাক কমান্ড টি।

**৫.১ সম্পর্কিত কমান্ডলাইনস:**

রিমোট রিপোতে সব কিছু সেভ করা: *git push origin ‘branch name’*

[এখানে push দিয়ে বলা হচ্ছে যে লোকাল রিপোর সমস্ত পরিবর্তন ধাক্কা দাও। কোথায় দেবে? Origin এর কোন ব্রাঞ্চে। এখানে origin অর্থ হলো গিটহাবের ঐ অরিজিনাল বা মূল রিপোজিটরি যেটাকে লোকাল মেশিনে ক্লোন করা হয়েছে। আর কোন প্রজেক্টের মূল নিয়ন্ত্রক বাঞ্চ হলো main ব্রাঞ্চ। যে যে ব্রাঞ্চ থেকে পুশ করবে সে সেই ব্রাঞ্চ এর নাম লিখবে।]

**৬. লোকাল মেশিনের ফোল্ডার কে লোকাল রিপো বানানো:**

ধরা যাক আমরা আমাদের পিসি তে একটি ফোল্ডারে আমাদের সব কাজ রেখেছি। এখন ভাবলাম আমরা এই ফোল্ডারটি গিটহাবে রিপোজিটর হিসাবে আপলোড করে দেই যাতে করে দুনিয়ার সব প্রান্ত থেকে সেটাকে একসেস করা যায়। এক্ষেত্রে আমাদের ফোল্ডারকে আগে গিট রিপোজিটরি বানাতে হবে। সেজন্য প্রথমে গিট ব্যাস বা টার্মিনাল থেকে ওয়ার্কিং ডাইরেক্টরি চেঞ্জ করে আমাদের টার্গেট ফোল্ডার কে present working directory বানাতে হবে। কারণ আমরা এই ফোল্ডারকে নিয়ে অপারেশনে যাবো।

**৬.১ সম্পর্কিত কমান্ডলাইনস:**

লোকাল ফোল্ডার কে ওয়ার্কিং ডাইরেক্টরি বানানো: *cd ‘Target Folder Name’*

লোকাল ফোল্ডারকে লোকাল/গিট রিপোজিটরি বানানো: *git init [init দিয়ে বোঝায় ফোল্ডারটি গিট রিপো হিসাবে ইনিশিয়েট করা হলো।]*

লোকাল রিপো চেক: *ls –a [দেখা যাবে লিস্টে ..git নামে একটি ফোল্ডার আছে তার মানে এটা সফলভাবে লোকাল গিটে পরিণত হয়েছে।]*

এখন আমাদের প্রয়োজনীয় কাজ সব করে সমস্ত পরিবর্তন সেভ করার জন্য ৪.১ এর নির্দেশনা ফলো করতে হবে। তাহলে সব পরিবর্তন লোকাল রিপোতে সেভ হলো। কিন্তু রিমোট রিপো বা গিটহাভে তো কিছুই করা হয়নি। এর জন্য পরবর্তী নির্দেশনা মনোযোগ দিয়ে দেখতে হবে।

**৭. লোকাল রিপোকে রিমোট রিপো বানানো**

এখন আমরা লোকাল রিপোকে রিমোট রিপো হিসাবে আপলোড করবো গিটহাবে। প্রথমে গিটহাবে গিয়ে একটি প্যারেন্ট ফোল্ডার বানাতে হবে যার মধ্যে এরকম অনেক লোকাল রিপো বা প্রজেক্ট ফোল্ডার সেভ হবে। প্যারেন্ট ফোল্ডার বা রিপোজিটরি বানানোর পরে সেই রিপোর HTTPS path কপি করে নিতে হবে। এবার বাকি কাজ গিট ব্যাশ বা টার্মিনালেই করতে হবে।

**৭.১ সম্পর্কিত কমান্ডলাইনস:**

প্রথমে লোকাল রিপোর সাথে রিমোট রিপোর কানেকশন পাথ তৈরী করতে হবে।

পাথ তৈরী: *git remote add origin ‘HTTPS path of github repository’*

পাথ তৈরি হলো কিনা চেক করা: *git remote –v*

**৭.২ এবার ব্রাঞ্চ কনফিগার করার পালা:-**

ব্রাঞ্চ চেক করা: *git branch*

*[দেখা যাবে যে সিস্টেমটি master নামের ব্রাঞ্চ এ চলছে। এখন আমাদের কে main ব্রাঞ্চ এ যেতে হবে অথবা master ব্রাঞ্চ এর নাম পরিবর্তন করে main ব্রাঞ্চ নাম দিতে হবে। কারণ রিপোর মূল নিয়ন্ত্রণ কিন্তু main ব্রাঞ্চ এর মাধ্যমে হয়।]*

ব্রাঞ্চ এর নাম পরিবর্তন করা: *git branch -M main* *[master ব্রাঞ্চ এর নাম main ব্রাঞ্চ হয়ে গেলো]*

তাহলে এখন আমরা লোকাল রিপোর main branch এ আছি। এখন সময় হলো লোকাল রিপো কে রিমোট রিপোতে হোস্ট করা বা গিটহাব সাইটে আপলোড করা।

লোকাল রিপোকে রিমোট রিপোতে পুশ করা: *git push origin main*

আমরা এখানে -u নামের একটি ফ্লাগ ব্যবহার করতে পারি। তখন কমান্ড হবে এরকম *git push -u origin main* ।এর ফলে কি হবে? পরবর্তীতে কেবল *git push* লিখলেই হবে *origin main* লেখা লাগবে না।

**৮. ব্রাঞ্চ অপারেশন:**

একটি প্রোজেক্টে সাধারণত অনেক ধরণের ডেভেলপার কাজ করে। কিন্তু গিটহাবে তো রিপজিটরি একটা তার মধ্যে প্রোজেক্ট ও একটা তাহলে সবাই কি আলাদা আলাদা রিপো বানাবে? এমন টা নয়, প্রোজেক্ট ডেভেলপমেন্ট এর এক এক শাখা ঐ একটা প্রোজেক্টের আলাদা আলাদা ভারসন ভার্চুয়ালি তৈরি করে এবং তাদের নিজস্ব পরিবর্তন সেভ করে। এবং কাজ শেষে সব শাখা গুলো এক সাথে মার্জ করে একটি মেইন ফাইল তৈরী হয়। মূল কথা হলো ফাইল আলাদা আলাদা তৈরি হয় না। একটাই ফাইলের আলাদা আলাদা ভার্সন তৈরী হয়। এটাই মূলত ব্যাঞ্চের সৌন্দর্য।

**৮.১ সম্পর্কিত কমান্ড লাইনস:**

চলমান ব্রাঞ্চ চেক করা: *git branch*

চলমান ব্রাঞ্চ এর নাম পরিবর্তন করা: *git branch -M ‘New Name of current branch’*

নতুন ব্যাঞ্চ তৈরি করা: *git checkout -b ‘New branch Name’*

ব্রাঞ্চ পরিবর্তন করা: *git checkout ‘branch name’*

ব্রাঞ্চ ডিলিট করা: *git branch -d ‘Name of target branch’*

ব্রাঞ্চ ক্রিয়েট করা মানে হলো লোকাল রিপোতে ক্রিয়েট করা। এটা তখনও রিমোট রিপোতে আপডেট হয়নি। এখন নতুন ব্রাঞ্চে গিয়ে কাজ করে সব পরিবর্তন *add, commit* এবং সবশেষে *push* করলে ব্রাঞ্চ আপডেট হয়ে যাবে। তবে এখন পুশ কিন্তু *git push origin main* নয় বরং *git push origin ‘Current branch name’*

**৯. Merge অপারেশন:**

ধরা যাক কোন প্রোজেক্টে main branch ছাড়া দুইটি আলাদা ব্রাঞ্চ যেমন homePc branch এবং officePc branch রয়েছে। এখন দুইটি ব্রাঞ্চ একই ফাইলে তারা আলদা আলাদা কাজ করেছে। প্রোজেক্ট শেষে তারা তাদের কাজ এবার main branch এর সাথে মিলিয়ে দিতে চায়। এক্ষেত্রে ব্যবহার করা হয় merge branch operation । এই কাজটি দুই ভাবে করা যায় প্রথমে দেখব কিভাবে ম্যানুয়ালি অর্থাৎ গিটহাবে গিয়ে করতে হয় সেটা।

**৯.১ গিটহাব থেকে merge করার উপায়:**

ধরা যাক আমি homePc branch এ আছি এখন আমি আমার ব্রাঞ্চ এর কাজ কে main branch এর সাথে মিশানোর জন্য main branch যে দেখাশোনা করছে তাকে pull request দেব। সেক্ষেত্রে গিটহাবে গিয়ে নিজের ব্রাঞ্চ সিলেক্ট করে pull request create করতে হয়। এর পর main branch থেকে সেটা যাচাই করা হয় এবং দেখা হয় automatic merge সম্ভব কিনা। যদি সম্ভব হয় তাহলে automatic merge বাটনে ক্লিক করে merge করতে হয় আর যদি automatic merge সম্ভব না হয় তাহলে বিশেষ অংশ পরিবর্তনের জন্য উক্ত শাখার ডেভলপারকে কিছু অংশ চেঞ্জ করতে বলা হয় কমেন্টের মাধ্যমে।

সফল ভাবে মার্জ করলে রিমোট রিপোতে মার্জ হবে অর্থাৎ রিমোট রিপোর main branch update হবে ঠিকই কিন্তু লোকাল রিপোতে main branch update হবে না। সেক্ষেত্রে লোকাল রিপোতে আপডেট করার জন্য গিট ব্যাশে গিয়ে আগে ব্রাঞ্চটা চেঞ্জ করে main branch এ নিতে হবে। তারপর কমান্ড চালাতে হবে। রিমোটের কোন চেঞ্জ লোকালে আনার জন্য সবসময় pull command লিখতে হবে।

কমান্ড হলো *git pull origin main*

**৯.২ গিট টার্মিনাল থেকে merge করার উপায়:**

একই কাজ গিট ব্যাশ থেকে করা যায়। যে ব্রাঞ্চ থেকে মার্জ করতে চাই সেই ব্রাঞ্চ এ গিয়ে আগে main branch এর সাথে পার্থক্য কোথায় সেটা বের করতে হবে। এই কাজ গুলো জটিল ধাপে আগায়।

**প্রথমত**: নিজের ব্রাঞ্চে কাজ করার পর সেগুলো add এবং commit করতে হবে। Push করা যাবে না। push করলে ম্যানুয়ালি গিটহাবে হিয়ে পুল রিকোয়েস্ট এর মাধ্যমে merge করতে হবে।

**দ্বিতীয়ত**: commit করা অবস্থায় main branch এর সাথে পার্থক্য দেখে নিতে হবে। Main branch এর সাথে পার্থক্য দেখার কম্যান্ড: *git diff main* [main না হয়ে অন্য যে কোন ব্রাঞ্চ হতে পারে। মূলত যার সাথে পার্থক্য দেখতে চাই সেই ব্রাঞ্চ এর নাম]

**তৃতীয়ত:** এখন main branch কে আমার সাথে merge করে নিতে হবে প্রথমে। তাহলে কম্যান্ড লিখতে হবে *git merge main* যদি কোন কনফ্লিক্ট থাকে তাহলে merge হবে না বরং কনফ্লিক্ট টা আগে সংশোধন করতে বলবে। সেক্ষেত্রে কোড এডিটর থেকে conflict টা মিমাংশা করতে হবে তার পর পুনরায় add ও commit করতে হবে। অতঃপর conflict টা মিমাংশা করার পর পুনরায় merge এর কম্যান্ড টা লিখতে হবে। মার্জ হয়ে গেলে পুনরায় add ও commit ও push command দিয়ে রিমোটে আপডেট করে দিতে হবে। এক্ষেত্রে main branch টা আমার ব্রাঞ্চ এ মিলিয়ে নিলাম। এখন main branch টা আমার ব্রাঞ্চ কে নিজের মধ্যে মিলিয়ে নিক।

**সর্বশেষ:** ঠিক একই কাজ এবার main branch এ গিয়ে করতে হবে। main branch এ গিয়ে যে ব্রাঞ্চ থেকে কেবল merge করা হয়েছে ঠিক সেই ব্রাঞ্চকে main branch এর সাথে মার্জ করতে হবে। তাহলে কম্যান্ড লিখতে হবে *git merge homePc* । মার্জ হয়ে গেলে add ও commit ও push command দিয়ে রিমোটে আপডেট করে দিতে হবে। এক্ষেত্রে কোন কনফ্লিক্ট জন্মায় না, কারন কনফ্লিক্ট পূর্বের ব্রাঞ্চ থেকে আগেই নিষ্পত্তি করা হয়েছে।

**১০. পরিবর্তন কে ফিরিয়ে আনা:**

ধরা যাক আমার কোড বেস-এ একটি পরিবর্ন এনে আমি add করে ফেলেছি। আমি এখন চাচ্ছি যে আমার add করা ভুল হয়ে গেছে আমি add এর আগের মুহূর্তে ফেরত যেতে চাই। উল্লেখ্য, কোন পরিবর্তন add না করা পর্যন্ত ঐ পরিবর্তন গিট Modified হিসাবে দেখায় আর add করে দিলে **Modified** দেখায়। তাহলে আমরা সবুজ **Modified** কে আবার লাল এর পর্যায়ে আনার জন্য কম্যান্ড লিখতে পারি।

**১০.১ Add হওয়া পরিবর্তন কে Add এর আগমুহূর্তে নিয়ে আসা:**

কোন পরিবর্তনকে unstaged করার command: *git reset ‘Name of the file’*

সব ফাইল একবারে এক স্টেজ আগে রিসেট করার জন্য: *git reset*

তবে মনে রাখতে হবে যে, এই কম্যান্ড মুহূর্তটা চেঞ্জ করবে অর্থাৎ একটি কম্যান্ড আগে নিয়ে যাবে সময়টা।

**১০.২ এক commit এর পিছনে আসা:**

কাজ শেষ করে add এর তাঁর পর commit করে দেওয়ার পরে যদি মনে পড়ে যে ভুল হয়েছে যা সংশোধন করা দরকার তাহলে নিচের কমান্ড লিখতে হবে: *git reset HEAD~1*

এক্ষেত্রে একটি বিষয় মাথায় রাখতে হবে যে এই কমান্ড সর্বশেষ একটি commit মুহূর্তকে পিছনে নিয়ে আসবে এবং add এর আগে এনে রাখবে। ফলে প্রয়োজনীয় সংশোধন এনে পুনরায় add এবং commit করতে হবে। এই কমান্ডে গিট ব্যাশ পরিবর্তিত হবে কিন্তু কোড বেস এর কোন পরিবর্তন হবে না।

**১০.৩ একাধিক commit এর পিছনে আসা:**

ধরাযাক আমরা একদিনে ১০ টা কমিট করেছি। এর মধ্যে ৯ নং কমিটে আমরা কিছু ডিলিট করে ফেলেছি আবার ১০ কমিটে গিয়ে কিছুটা এড করেছি। এখন মনে হচ্ছে আমরা ১০ম কমিট টা ভুল করেছি। আমাদের ৯ কমিট এ কিছু কোড ডিলিট করেছি যা ফেরত আনতে হবে। এখন যদি ১০.২ এর কমান্ড দেই তাহলে গিট ব্যাশ এক কমিট পেছনে যাবে কিন্তু কোড এডিটরে কোণ পরিবর্তন আসবে না। কিন্তু আমাদের তো ৯ নং কমিট তো বাতিল করাই লাগবে এবং কোড ফেরত ও আনা লাগবে সেক্ষেত্রে আমাদের একটি প্রোসেস ফলো করতে হবে।

কমিট লগ দেখা: *git log*

এবার আমাদের কোন কমিটে ফেরত যেতে চাই সেই কমিটের হ্যাশ কোড কপি করতে হবে এবং

কমান্ড দিতে হবে: *git reset ‘hash code’*

এখন গিট ব্যাশ তো ৯ নং কমিটের আগে নিতেই পারলাম কিন্তু কোড বেস এর পরিবর্তন হলো না। এক্ষেত্রে HARD Reset ব্যবহার করতে হবে। কমান্ড দিতে হবে *git reset - -HARD ‘hash code’*  
এখন দেখা যাবে ব্যাশ ও পেছনে গেছে এবং কোড এডিটরে ও সব ফেরত এসেছে।