Git

গিট ইন্সটল করার পর আমাদের ইমেইল এবং নাম সেট করতে হবে। তার জন্য নিচের দুইটা কমান্ড লিখতে হবেঃ

|  |  |
| --- | --- |
|  | git config --global user.name "Your Name"    git config --global user.email"Your Email" |
|  | ইউজার নেম এবং ইমেইল ঠিক মত সেট আপ করা হয়েছে কিনা, তা দেখতে চাইলে কমান্ড লাইন বা টারমিনালে নিচের কমান্ড লিখলে দেখাবেঃ  git config --global user.name    git config --global user.email  এরপর গিট ইনিশিয়ালাইজ করার জন্য লিখবঃ  git init  তাহলে আমাদের  git-push-demo ফোল্ডারে একটা লোকাল গিট রিপোজিটোরি তৈরি হবে। এবার কাজ হচ্ছে আমাদের এই রিপোজিটোরিতে আমাদের প্রজেক্টের ফাইল গুলো যুক্ত করা। তার জন্য লিখবঃ  git add .  git add . মানে হচ্ছে সব গুলো ফাইল গিট রিপোজিটোরিতে যুক্ত করা। আমাদের প্রজেক্টে মাত্র একটা ফাইল রয়েছে। |
|  |  |
|  |  |

এড করার পর commit করব। তার জন্য লিখবঃ

git commit -m "first commit"

এবার আমরা এ প্রজেক্টটি গিটহাবে পুশ করার জন্য প্রস্তুত। গিটহাবে যেখানে আমরা পুশ করব, তাকে বলে রিমোট রিপোজিটোরি। লোকাল রিপোজিটোরি থেকে রিমোটে পুশ করার জন্য আগে আমাদের রিমোট রিপোজিটোরির URL টা যুক্ত করতে হবে। প্রত্যেকটা গিটহাব প্রজেক্টের জন্য একটা করে url পাবো। যেমন আমাদের git-push-demo <https://github.com/jakirseu/git-push-demo.git> । jakirseu হচ্ছে আমার ইউজার নেম। রিমোট রিপোজিটোরি যুক্ত করার জন্য লিখতে হবেঃ

git remote add origin <https://github.com/shahriarmurol/test.git>

git push -u origin master

এই বার আমাদের গিটহাবের ইউজার নেম চাইবে। এরপর চাইবে পাসওয়ার্ড। পাসওয়ার্ড দিয়ে এন্টার চাপলে আমাদের প্রজেক্টটি গিটহাবে আপলোড হয়ে যাবে। সব কিছু ঠিক মত হলে সাকসেস মেসেজ দেখাবে।

Repository download

git clone <https://github.com/shahriarmurol/test.git>

আর একটা বিষয়! পুল। সার্ভার থেকে নিজ ফোল্ডারে প্রজেক্ট ডাউনলোড করার জন্য বা নিজ লোকাল প্রজেক্ট আপডেট করার জন্য ব্যবহার করা হয় pull

git pull <https://github.com/shahriarmurol/test.git>

//not understand

আমরা কিছু কাজ করেছি, তা অদরকারি। সে কাজ গুলো কমিট করতে চাচ্ছি না। সে কাজ গুলো ফেলে দেওয়ার জন্য লিখতে পারিঃ

git stash

------------------------ git for team work -------------------------------

**Git Branching & Merging:**

গিটের উপরের কাজ গুলো সিম্পল। একজন একটি প্রজেক্টে একজন ডেভেলপার কাজ করলে সাধারণত উপরের কমান্ড গুলো বেশি ব্যবহৃত হয়।

কিন্তু টিমে কাজ করলে আরো কিছু কমান্ড জানা লাগে। কারণ যখন আমরা টিমে কাজ করি, তখন একই প্রজেক্টে অনেক জন ডেভেলপার এক সাথে কাজ করি। দেখা যায় এক জন এক এক ফিচারে কাজ করে। আর এ থেকেই এসেছে Branch এর ধারণা। এক এক জন এক এক ব্রাঞ্চে কাজ করে। আবার যখন এক একটা ফিচার এর কাজ কমপ্লিট হয়ে যায়, তখন সব কিছু আবার তা মূল প্রজেক্টের সাথ যুক্ত করতে হয়। একে বলা হয় Merge।

নতুন একটা ব্রাঞ্চ তৈরি করার জন্য লিখতে হয়:

|  |  |
| --- | --- |
|  | git branch branch\_name |

git এ ডিফল্ট ব্রাঞ্চ একটা তৈরি থাকে, তা হচ্ছে master। একটা ব্রাঞ্চ তৈরি করার পর master ব্রাঞ্চ থেকে ঐ ব্রাঞ্চে সুইচ করতে হয়। ব্রাঞ্চ তৈরি করার পর ঐ ব্রাঞ্চে সুইচ করার জন্য লিখতে হয়ঃ

|  |  |
| --- | --- |
|  | git checkout branch\_name |

ডেভেলপারদের home এবং about দুইটা পেইজ তৈরি হয়ে গেলে বা menu এবং sidebar এর কাজ হয়ে গেলে আমরা ব্রাঞ্চ গুলো মার্জ করে নিব। মার্জ করে এভাবেঃ

git merge branch\_name

একটা ব্রাঞ্চের কাজ শেষ। মূল প্রজেক্টের সাথে মার্জ করার পর আমরা চাইলে ঐ ব্রাঞ্চ ডিলেট করে দিতে পারি। ডিলেট করার জন্য লিখতে হয়ঃ

|  |  |
| --- | --- |
| 1 | git branch -d branch\_name |

যেমন আমরা about ব্রাঞ্চ ডীলেট করার জন্য লিখবঃ

|  |  |
| --- | --- |
| 1 | git branch -d about |
|  |  |

**jQuery Animations - The animate() Method**

The jQuery **animate()** method is used to **create custom animations**.

Syntax:

$(selector).animate({params},speed,callback);

$("button").click(function(){

$("div").animate({left: '250px'});

});