এক পলকে গিট (Git) ও গিটহাব (GitHub)—পর্ব ২/৩

medium.com/প্রোগ্রামিং-পাতা/এক-পলকে-গিট-git-ও-গিটহাব-github-পর্ব-২-৩-73d252cad40b

June 5, 2018

এক পলকে গিট (Git) ও গিটহাব (GitHub)—পর্ব ২/৩
*আগের পর্ব না দেখে থাকলে অবশ্যই সেখান থেকে পড়া শুরু করবেন। কারন এই লেখাটা আগের লেখার ধারাবাহিকতায় লেখা*medium.com

*গিটহাবের সাথে লিঙ্ক করাঃ*medium.com

গত পর্বে কিভাবে কমিট করবেন সেপর্যন্ত দেখিয়েছিলাম, এবার বাকি অংশে আরেকটু অ্যাডভান্স কাজ দেখাবো…

ফাইল মডিফাই করে আবার কমিট করাঃ

এখন আমার একটা ভার্শন তৈরী হয়ে গেলো। কিন্তু আমার প্রোজেক্টে আরো অনেক কাজ আছে। আমি চাচ্ছি `friend-lists.txt` ফাইলে সব ফ্রেন্ডদের ফোন নাম্বার সেইভ করে রাখতে। বর্তমানে ফাইলটা এরকম আছেঃ

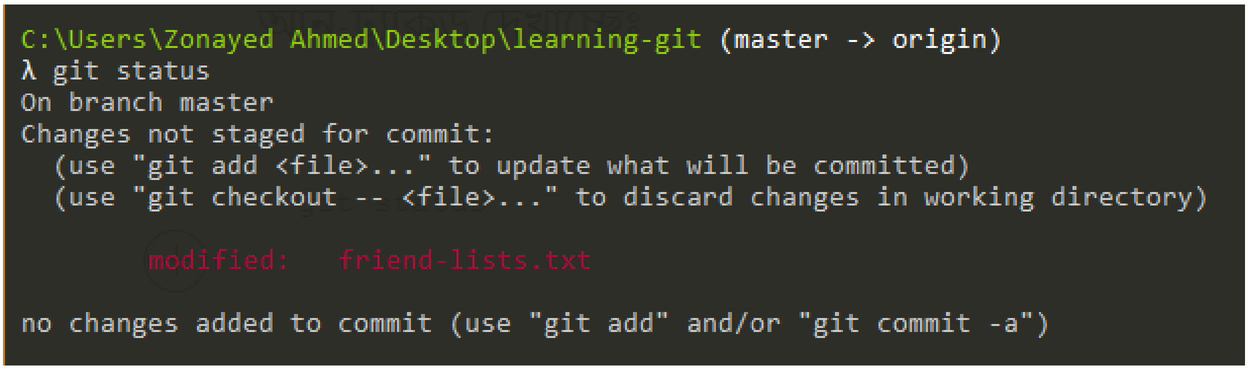
```
Dibakar Sutradhar
S M Shahadat Hossain
Reduanul Houque Munna
Ar Rolin
Niraj Paudel
Tanvir Faisal Moon
Sagar Neupane
Yadav Lamechane
```

এখানে এই টেক্সট ফাইল এডিট করা আর কোনো কোড এডিট করা একই কথা। আমি টেক্সট ফাইলের সাহায্যে দেখাচ্ছি যাতে সবার বুঝতে সুবিধা হয়। এখন আমি সবার সাথে ফোন নাম্বার অ্যাড করবো।

```
Dibakar Sutradhar - +88018XXXXXXXX
S M Shahadat Hossain - +88018XXXXXXXX
Reduanul Houque Munna - +88018XXXXXXXX
Ar Rolin - +88018XXXXXXXX
Niraj Paudel - +9718XXXXXXXX
Tanvir Faisal Moon - +88018XXXXXXXX
Sagar Neupane - +9718XXXXXXXX
Yadav Lamechane - +9718XXXXXXXX
```

এখন টেক্সট এডিটর বা কোড এডিটর যেটাই ইউজ করে ফাইল মডিফাই করলাম সেটাতে সেইভ দিয়ে `git status` চেক করলে দেখবেন ফাইল এটা মডিফাইড এবং আন-ট্র্যাকড দেখাচ্ছেঃ

```
git status
```



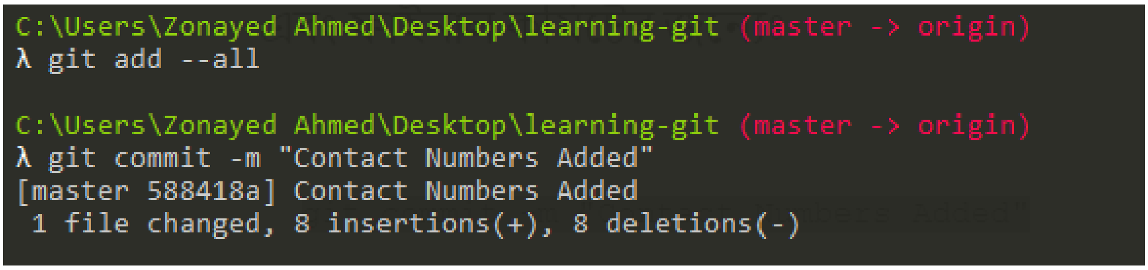
```
C:\Users\Zonayed Ahmed\Desktop\learning-git (master -> origin)
λ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   friend-lists.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

এখন এই আন-ট্র্যাকড ফাইলটাকে স্টেজিং এ নিয়ে ফাইনাল কমিট করে দিতে চাচ্ছি

```
git add --all
```

এবং ফাইনাল কমিটের জন্যঃ

```
git commit -m "Contact Numbers Added"
```



```
C:\Users\Zonayed Ahmed\Desktop\learning-git (master -> origin)
λ git add --all

C:\Users\Zonayed Ahmed\Desktop\learning-git (master -> origin)
λ git commit -m "Contact Numbers Added"
[master 588418a] Contact Numbers Added
 1 file changed, 8 insertions(+), 8 deletions(-)
```

পুনরায় মডিফাই করে কমিট করাঃ

এতক্ষণ ধরে হয়তো নিচের এই জিনিসটা লক্ষ্য করেছেন। `master` সাম্থিং...

```
(master -> origin)
```

আমার কমান্ড লাইনে কখনো লাল কখনো সাদা। লাল হয় যদি কোনো ফাইল ট্র্যাক না করা থাকে, আর সাদা হয় সবকিছু ট্র্যাক(কমিট) করা থাকলে। যাই হউক এই `master` হলো বর্তমান ব্রাঞ্চের নাম। অর্থাৎ বর্তমানে আপনি মাস্টার ব্রাঞ্চ বা বর্তমান ওয়ার্কিং ডিরেক্টরিতে আছেন। এটাই আপনার প্রোজেক্টের বর্তমান ভার্শন।

এখন আমাদের ডেভেলপমেন্ট এ অনেক সময় দেখা যায় পূর্বের ভার্শনে ফিরে যেতে হয়। একটা একটা করে ফিচার ডেভেলপড করার পর একটা সময় এসে কোনো প্রব্লেম দেখা দেয় যেটা পূর্বের কোনো ভার্শনে ঠিকঠাক কাজ করতো কিন্তু এখন সেটা কাজ করছে না। সেক্ষেত্রে গিট এ ট্র্যাক করা থাকলে আপনি সহজেই আপনার সেই ভার্শনে ফিরে যেতে পারবেন আর কোড চেক করতে পারবেন, চাইলে আপনার প্রোজেক্ট রানও করতে পারবেন। ঠিক ঐসময় আপনার প্রোজেক্ট যেরকম ছিলো সেরকমটাই দেখবেন।

আমরা এখন ইচ্ছাকৃতভাবেই `friend-lists.txt` ভিতরে অ্যাড করা ফোন নাম্বারগুলো মুছে দিয়ে কমিট করবো আরেকটা। মুছে ফেলার পর ফাইলটা এরকম হবেঃ

```
Dibakar Sutradhar
S M Shahadat Hossain
Reduanul Houque Munna
Ar Rolin
Niraj Paudel
Tanvir Faisal Moon
Sagar Neupane
Yadav Lamechane
```

এখন এটা সেইভ করে স্টেজিং এ অ্যাড করে কমিট করে দিবোঃ

```
git add --all
git commit -m "Contact numbers removed"
```

কমিট লগ চেক করাঃ

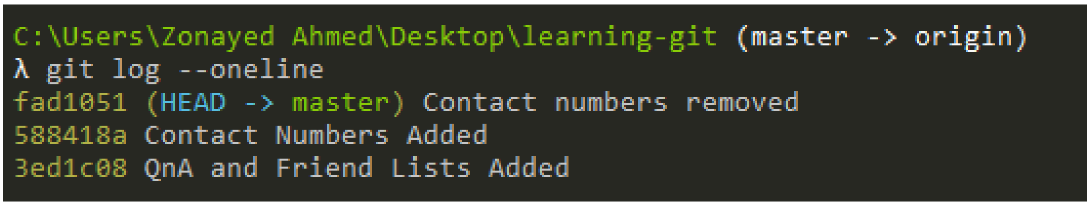
(আর্টিফিশিয়াল) হায় হায়! ফোন নাম্বার সব গেলো!!! কি হবে এখন? হ্যাঁ গিট দিয়ে তো ট্র্যাক করেই রেখেছি সব। কোন কমিটে জানি ফোন নাম্বারগুলো রেখেছিলাম? হ্যাঁ সেটা দেখতে চাচ্ছি। সব কমিটের লগ দেখতে চাইলেঃ

```
git log
```

এখানে তিনটা কমিট আছে। সাথে ডিটেইলস সহ, কমিটের ম্যাসেজ দেখে ইজিলিই বুঝতে পারবেন কোন কমিটে কি করা হয়েছিলো। আর সাথে কিছু এলোমেলো নাম্বার আছে। এগুলো ইউজ করে আমরা পূর্বের ভার্শনগুলোয় ফিরে যেতে পারবো।

এই লগ টা আরো সুন্দর করে কম্প্যাক ভার্শনে দেখতে চাইলে উপরের কমান্ডটা এভাবেও দেওয়া যাবেঃ

```
git log --oneline
```



```
C:\Users\Zonayed Ahmed\Desktop\learning-git (master -> origin)
λ git log --oneline
fad1051 (HEAD -> master) Contact numbers removed
588418a Contact Numbers Added
3ed1c08 QnA and Friend Lists Added
```

এখানে সুন্দর করে ছোটো করে প্রয়োজনীয় সব দেখাচ্ছে। এখন এইখানের শর্টকাট এলোমেলো ইউনিক কমিট আইডিগুলোও শর্ট করে দেওয়া হয়েছে। এই শর্ট ভার্শনগুলোও ইউজ করে আগের কাঙ্ক্ষিত ভার্শনে যেতে পারবেন।

পূর্বের ভার্শনে ফিরে যাওয়াঃ

এখন আমরা যে কমিটে ফোন নাম্বার গুলো অ্যাড করেছিলাম সে কমিটে ফিরে যেতে চাচ্ছি। আমার এখানে সেই কমিটটা হলো এটাঃ

```
588418a Contact Numbers Added
```

এখন এই ভার্শনে ফিরে যেতে চাইলে গিটের আরেকটা কমান্ড একভাবে ইউজ করতে হবে

```
git checkout 588418a
```

এখানে শেষেরটা হচ্ছে কমিট আইডি। আপনার আইডি ভিন্ন হবে। এখন এই কমান্ড রান করলে আপনার প্রোজেক্ট `master` ব্রাঞ্চ থেকে আগের এই কমিটের ভার্শনে ফিরে যাবে। তবে অবশ্যই মাস্টার ব্রাঞ্চে থাকাকালে সবকিছু আপনার ট্র্যাক করা থাকতে হবে। কোনো ফাইল/ফোল্ডার আন-ট্র্যাকড থাকলে বা আন-কমিটেড থাকলে আপনি চেক-আউট করতে পারবেন না। এখন কমান্ড লাইনে `master` এর জায়গায় কমিট আইডিটা দেখবেন। সাথে দেখবেন লেখা HEAD detached at আপনার কমিট আইডি।

এখন আপনার ফাইল চেক করে দেখুন আগের সেই ভার্শনে ফিরে আসছে। friend-lists.txt তে সবার ফোন নাম্বারগুলো রয়েছেঃ

```
Dibakar Sutradhar - +88018XXXXXXXX
S M Shahadat Hossain - +88018XXXXXXXX
Reduanul Houque Munna - +88018XXXXXXXX
Ar Rolin - +88018XXXXXXXX
Niraj Paudel - +9718XXXXXXXX
Tanvir Faisal Moon - +88018XXXXXXXX
Sagar Neupane - +9718XXXXXXXX
Yadav Lamechane - +9718XXXXXXXX
```

এখন আপনার বর্তমান ওয়ার্কিং ডিরেক্টরি আগের একটা ভার্শনে রয়েছে। কিন্তু আপনি যদি মাস্টার ব্রাঞ্চে যেতে চান তাহলে আবার চেক-আউট দিতে হবে এভাবেঃ

```
git checkout master
```

ব্রাঞ্চ তৈরীঃ

আমি আগেই ব্রাঞ্চ (branch) এর কথা বলেছিলাম। তবে ব্রাঞ্চ কে আরো স্পেসেফিকলি বললেঃ- ব্রাঞ্চ আসলে আপনার করা কমিটগুলোই, কিন্তু সেই কমিটগুলোর একটা ইউনিক নাম থাকবে। আপনি সেই কমিটে চেক-আউট করতে চাইলে ব্রাঞ্চ এর নাম দিয়েই চেক-আউট করতে পারবেন। আগের সেই আশ্চর্য টাইপের কমিট আইডি লাগবে না।

আমি এখন আমার এই প্রোজেক্টে নতুন কিছু ট্রাই করতে চাচ্ছি। তবে আমি মেইন প্রোজেক্টে বা মাস্টার ব্রাঞ্চে সেটা এখনি আনতে চাচ্ছি না। বলতে পারেন আমি এখন এক্সপেরিমেন্টাল কিছু একটা করবো। তারপর ভালো লাগলে মাস্টার ব্রাঞ্চে নিয়ে আসবো।

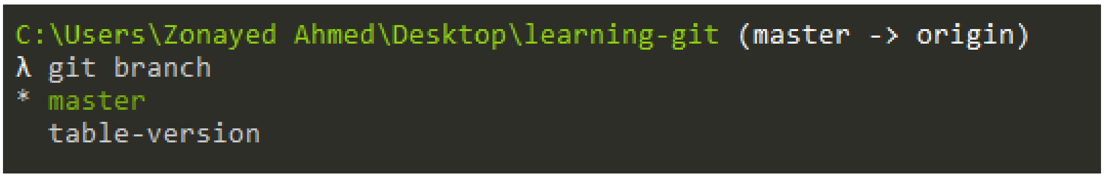
এইজন্যে আমরা নতুন একটা ব্রাঞ্চ তৈরী করবো `table-version` নাম দিয়েঃ

```
git branch table-version
```

এখন আপনার এই `table-version` নামে একটা ব্রাঞ্চ তৈরী হয়ে যাবে। আপনি যে ব্রাঞ্চ থেকে এই নতুন ব্রাঞ্চ তৈরী করবেন, নতুন ব্রাঞ্চে সেই ভার্শনটাই থাকবে। আমার ক্ষেত্রে আমি `master` ব্রাঞ্চ থেকে `table-version` ব্রাঞ্চ তৈরী করেছি। আর তাই `table-version` এ আমার বর্তমানে `master` ব্রাঞ্চ এ থাকা প্রোজেক্টের ভার্শনটাই যাবে। মানে এখন `master` আর `table-version` এর প্রোজেক্ট পুরোপুরি সেইম।

আপনি চাইলে আপনার প্রোজেক্টে থাকা সবগুলা ব্রাঞ্চ এর লিস্ট ও দেখতে পারবেনঃ

```
git branch
```



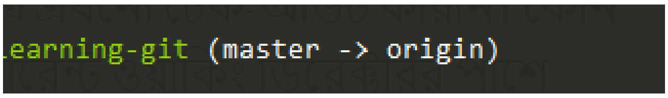
```
C:\Users\Zonayed Ahmed\Desktop\learning-git (master -> origin)
λ git branch
* master
  table-version
```

ব্রাঞ্চ এ চেক-আউট করাঃ

আমরা ব্রাঞ্চ তৈরী করেছি, কিন্তু সেই ব্রাঞ্চ এ এখনো চেক-আউট করিনি। কোন ব্রাঞ্চ এ আছি তা আপনার কমান্ড লাইনে কারেন্ট

ওয়ার্কিং ডিরেক্টরির পাশে দেখলেই বুঝবেন। আমরা আমাদের প্রোজেক্টে এখন `master` ব্রাঞ্চেই আছি।

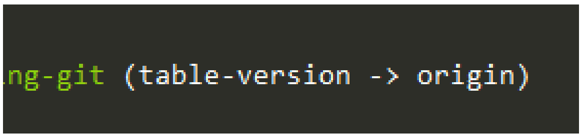
এখন নতুন ক্রিয়েট করা ব্রাঞ্চে চেক-আউট করা ঠিক আগের অন্য কোনো কমিটে চেক-আউট করার মতোই। শুধুমাত্র এক্ষেত্রে আমরা ব্রাঞ্চ এর নাম দিয়েই চেক-আউট করতে পারবোঃ



```
earning-git (master -> origin)
```

```
git checkout table-version
```

এখন দেখবেন আপনার ব্রাঞ্চ `table-version` এ চলে গেছেঃ

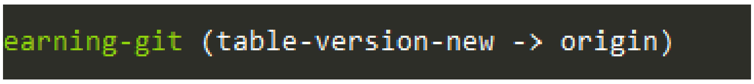


```
ng-git (table-version -> origin)
```

এখানেও একটা ছোট্ট শর্টকাট টেকনিক আছে। আপনি যদি চান নতুন ব্রাঞ্চ তৈরী করে সাথে সাথে সেই ব্রাঞ্চে চেক-আউট করতে, সেটা একলাইনের কমান্ডেই করতে পারবেনঃ

```
git checkout -b table-version-new
```

দেখুন আমরা নতুন একটা ব্রাঞ্চ `table-version-new` নামে তৈরী করেছি এবং সাথে সাথে সেই ব্রাঞ্চে চেক-আউট করে ফেলেছি



```
earning-git (table-version-new -> origin)
```

যাই হোক এখন আমরা `table-version` এ কিছু মডিফাই করে তারপর সেগুলো মাস্টারে মার্জ করবো। তাই `git checkout table-version` দিয়ে আমরা আমাদের কাঙ্ক্ষিত ব্রাঞ্চে চলে যাবো। অবশ্যই কাজ করার সময় খেয়াল করবেন কোন ব্রাঞ্চে আছেন। কষ্ট করে কারেন্ট ওয়ার্কিং ডিরেক্টরির ডান পাশে দেখলেই পাবেন কোন ব্রাঞ্চে আছেন সেটা।

নতুন ব্রাঞ্চে মডিফিকেশনঃ

এখন আমরা আমাদের এই নতুন `table-version` ব্রাঞ্চে নতুন কিছু ট্রাই করবো। বর্তমানে আমাদের প্রোজেক্টের `friend-lists.txt` ফাইল এই অবস্থায় আছেঃ

```
Dibakar Sutradhar
S M Shahadat Hossain
Reduanul Houque Munna
Ar Rolin
Niraj Paudel
Tanvir Faisal Moon
Sagar Neupane
Yadav Lamechane
```

এখন আমরা এই নামগুলো একটা টেবিলের ভিতরে নিয়ে দেখি কেমন লাগেঃ

```
===========================
|| Dibakar Sutradhar     ||
===========================
|| S M Shahadat Hossain  ||
===========================
|| Reduanul Houque Munna ||
===========================
|| Ar Rolin              ||
===========================
|| Niraj Paudel          ||
===========================
|| Tanvir Faisal Moon    ||
===========================
|| Sagar Neupane         ||
===========================
|| Yadav Lamechane       ||
===========================
```

ধরে নিলাম আমার কাজের এই ভার্শনটা আমার ভালো লেগেছে, এখন আমি এটা মাস্টার ব্রাঞ্চে বা মেইন প্রোজেক্টে নিয়ে যেতে চাই। কিন্তু তার আগে আপনার এই পরিবর্তনগুলো বর্তমান ব্রাঞ্চে কমিট করতে হবে। কারণ আপনি যতক্ষণ পর্যন্ত কোনো কিছু কমিট না করবেন, গিট সেগুলাকে কাউন্টই করবে না। কমিট করার জন্যেঃ

```
git add --all
git commit -m "Table added"
```

ব্যাস কমিট হয়ে গেলো আমার নতুন পরিবর্তনগুলো। এখন আমি এই `table-version` এ থাকা কাজগুলো মেইন `master` ব্রাঞ্চে নিতে চাচ্ছি। সেজন্যে প্রথমে `master` ব্রাঞ্চে চেক-আউট করতে হবেঃ

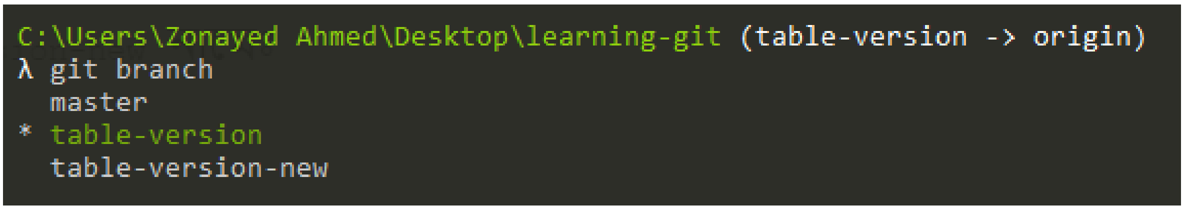
```
git checkout master
```

ব্রাঞ্চের নাম যেহেতু `master` , তাই এটা লিখেই চেক-আউট করতে পারবেন। এখন খেয়াল করুন আপনার `master` ব্রাঞ্চে যাওয়ার পর আপনার প্রোজেক্টের সেই আগের ভার্শনটাই রয়ে গেছে। নতুন `table-version` এ করা কাজ এখানে আসে নাই। আপনি যদি `table-version` এ করা কাজ ফেলে দিতে চাইতেন, তাহলে জাস্ট `master` চেক-আউট করে চলে আসলেই হচ্ছে, কোথাও কোনো লেখা বা কোডে হাত দিতে হবে না।

মনে করি নতুন ব্রাঞ্চে করা কাজ আমার ভালো লাগে নাই, বা এটা আমি রাখতে চাচ্ছি না। তাহলে জাস্ট সেই ব্রাঞ্চটাকে এভয়েড করে `master` এ চেক-আউট দিলেই চলবে বা চাইলে ব্রাঞ্চ ডিলেটও করে দিতে পারবেন। তবে আমরা `table-version` টা রাখবো। কিন্তু এর সাথে কিন্তু আমরা আরেকটা ব্রাঞ্চ তৈরী করেছিলাম `table-version-new` নামেঃ

ব্রাঞ্চ এর লিস্ট দেখতেঃ

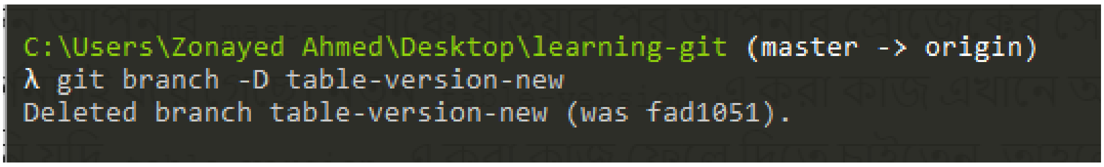
```
git branch
```



```
C:\Users\Zonayed Ahmed\Desktop\learning-git (table-version -> origin)
λ git branch
  master
* table-version
  table-version-new
```

আমরা `table-version-new` ব্রাঞ্চ ডিলেট করবো এখনঃ

```
git branch -D table-version-new
```



```
C:\Users\Zonayed Ahmed\Desktop\learning-git (master -> origin)
λ git branch -D table-version-new
Deleted branch table-version-new (was fad1051).
```

এখন এই ব্রাঞ্চ ডিলেট হয়ে যাবে, আর সেই সাথে ব্রাঞ্চে কোনো মডিফিকেশন থাকলে সেগুলোও ডিলেট হয়ে যাবে।

ব্রাঞ্চ মাস্টারে মার্জ করাঃ

এখন মাস্টারে চেক-আউট করার পরে দেখবেন মাস্টার আগের ভার্শনেই আছে। এখন আমরা `table-version` এ করা মডিফিকেশনগুলো মাস্টারে আনতে চাচ্ছি। সেটা একদম ইজি। মাস্টার ব্রাঞ্চে থাকা অবস্থায় এই কমান্ড দিলেই অটোম্যাটিক মার্জ হয়ে যাবেঃ

```
git merge table-version
```

সেই সাথে `table-version` এর কমিটটাও গিট অটোম্যাটিক অ্যাড করবে। গিট লগ দেখলে সেটাই দেখতে পাবেনঃ

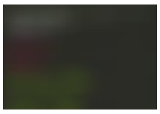
```
git log --oneline
```

একটা কমিটের সাথে আরেকটা কমিটের পার্থক্য দেখাঃ

এখন আমরা যদি আমাদের বর্তমানের কমিটের সাথে আগের কমিটের পার্থক্য দেখতে চাই, কী কী কোড পরিবর্তন হয়েছে, কোথায় কোড অ্যাড করা হয়েছে, কোথায় ডিলেট করা হয়েছে, এগুলোও সব দেখতে পারবো গিটের কমান্ডের সাহায্যেঃ

ধরি, আমরা Contact numbers removed আর Contact Numbers Added এই দুইটা কমিটের পার্থ্যক্যগুলো দেখতে চাচ্ছি। তাহলে এই দুটোরই কমিট আইডি লাগবে। কমিট আইডি গিট লগ দিয়ে ইজিলিই বের করতে পারবেন। এখানে `git diff` এর সাথে উক্ত দুইটা কমিটের আইডি পাস করতে হবে এভাবেঃ

```
git diff fad1051 588418a
```



আমি আসলে সবগুলো একসাথে কপি পেস্ট করেছিলাম, তাই সবগুলা লাইনই একবার রিমুভ আরেকবার অ্যাড দেখাচ্ছে, গিট এখানে অনেক স্মার্ট। গিট লাইন বাই লাইন, ক্যারেক্টার বাই ক্যারেক্টার দেখাবে। এখানে আসলে শুধুমাত্র ফোন নাম্বার গুলো অ্যাড করা হয়েছে এরকম দেখাবে।

এখানে উক্ত দুইটা কমিটে কোন ফাইলে এবং ঠিক কি কি রিমুভ (লালগুলো) এবং অ্যাড(সবুজগুলো) করা হয়েছে সেগুলো দেখানো হচ্ছে।

এখানে আরো লক্ষ্য করুন আমি `git diff` এর সাথে প্রথম আর্গুমেন্ট এ মোস্ট রিসেন্ট কমিট এবং পরেরটায় সেই কমিটের আগের কমিটের আইডি দিয়েছি। মানে প্রথম নতুনটা আর পরে পুরোনোটা দিয়েছে। এটার মানে হচ্ছে আমি প্রথমটার সাথে দ্বিতীয়টার পার্থ্যক্য দেখতে চাচ্ছি। এক্ষেত্রে দ্বিতীয়টা অর্থাৎ পুরোনোটার অনুসারে অ্যাডেড বা রিমুভড কোডগুলো দেখাবে। সেই সাথে কমিট আইডি প্রথমে পুরোনোটা এবং পরে নতুনটা দিলে ঠিক উল্টোটা দেখতে পাবেন। নতুনটার অনুসারে দেখাবে। কয়েকবার নিজে কমান্ড দিয়ে দেখলেই বুঝতে পারবেন।

এখন আমার প্রোজেক্ট আমি বাইরে সবার সাথে শেয়ার করতে চাই। এজন্যে আমাদের একটা হোস্ট প্রোভাইডার লাগবে, যে গিট ফ্রেন্ডলি এবং আমাকে গিটের সুবিধাগুলোসহ আমাকে ফ্রীতে হোস্ট প্রোভাইড করবে। হ্যাঁ এরকম একটা প্রোভাইডার হচ্ছে গিটহাব। আরো অনেক আছে, কিন্তু আজকে আমি গিটহাবেই কিভাবে কি করবেন সব দেখাবো। এখানে যেহেতু ইউজার ইন্টারফেস আছে, তাই আপনি পরে চাইলেও অন্য কোনো প্রোভাইডারের সার্ভিসও ইউজ করতে পারবেন। তবে গিটহাব অনেক পপুলার, কিন্তু তারপরেও কেন অন্য

প্রোভাইডার লাগবে? সেটার কারণ হচ্ছে গিটহাব ফ্রীতে প্রাইভেট রিপোজটরি করতে দেয় না। আপনি ফ্রীতে আনলিমিটেড রিপোজটরি করতে পারবেন। আপনার কোড যে কেউ চাইলে দেখতে পারবে। আর প্রাইভেট রিপোতে শুধুমাত্র সেই প্রোজেক্টের কন্ট্রিবিউটররাই দেখতে পারবে। তো এটাই গিটহাবের অনেক বড় অসুবিধা। এজন্যে অনেকে বিটবাকেট ও ইউজ করে, কারণ বিটবাকেট স্মল টিমের জন্যে ফ্রীতে প্রাইভেট রিপোর অ্যাক্সেস দেয়। তাছাড়া গিটল্যাবও আছে অনেক ভালো লেভেলের। আর যদি আপনি স্টুডেন্ট হয়ে থাকেন কোন ইউনিভার্সিটির, এই সিরিজের শেষের দিকে গিটহাবে প্রাইভেট রিপোর অ্যাক্সেস ফ্রীতে কিভাবে নিবেন সেটার একটা অভিজ্ঞতা শেয়ার করবো।