## repository အသစ် ဖန်တီးခြင်း

directory အသစ် တစ်ခုတည်ဆောက်ပါ။ ပြီးလျှင် terminal မှာ အောက်ကလို ရိုက်ပါ။  
git init  
ဆိုပြီး git repository တစ်ခုကို စတင်ပါ။

## repository ကို checkout လုပ်ခြင်း

copy အနေနဲ့ ပွားပြီးတော့ အလုပ်လုပ်မယ်ဆိုရင်တော့ အောက်က command ကို အသုံးပြုရပါတယ်။  
git clone /path/to/repository  
တကယ်လို့ server နဲ့ အလုပ်လုပ်တာဆိုရင်တော့ အောက်ကလို အသုံးပြုရပါတယ်။  
git clone username@host:/path/to/repository

## အလုပ်လုပ်ပုံ

သင့် local repository မှာ git က ထိန်းသိမ်းဆောက်ရွက်တဲ့ ဆင့်ပွား ၃ ခု ရှိပါတယ်။ ပထမ တစ်ခုကတော့ Working Directory ပါ။ သူကတော့ လက်ရှိ file တွေကို သိမ်းထားပေးပါတယ်။ ဒုတိယ တစ်ခုကတော့ Index ပါ။ သူကတော့ staging area ပါ။ နောက်ဆုံးကတော့ HEAD ပါ။ နောက်ဆုံး သင် commit လုပ်ထားတဲ့ နေရာကို ညွှန်ပြထားပါတယ်။

## add နှင့် commit

ပြင်ဆင်ထားတာများကို ထည့်သွင်းလိုလည်း အောက်ပါ အတိုင်း အသုံးပြုနိုင်သည်။  
git add <filename>  
git add \*  
ဒီ အဆင့်က ပထမ အဆင့်ပါ။ ပြောင်းလဲထားသည်များကို commit လုပ်လိုလျှင်  
git commit -m "Commit message"  
ပြင်ဆင်ထားသည်များကို **HEAD** တွင် မှတ်ထားပြီးသားဖြစ်သည်။ သို့သော် remote repository ပေါ်သို့ မရောက်သေးပါ။

## pushing changes

သင်ပြင်ဆင်ထားတာတွေက အခု သင့်ရဲ့ local working copy နေရာက **<HEAD>** မှာ ရောက်နေပါပြီ။  
ပြောင်းလဲထားတဲ့ code တွေကို remote repository ဆီပို့ဖို့ အတွက်  
git push origin master  
*master* ကို သင့်ရဲ့ စိတ်ကြိုက် branch ပြောင်းနိုင်ပါတယ်။ သင် ပို့ချင်တဲ့ branch တစ်ခုခုကိုပေါ့။

တကယ်လို့ သင်က cloned မလုပ်ချင်ဘူး။ လက်ရှိ ရှိထားတဲ့ code ကို remote server နဲ့ ချိတ်ဆက်ချင်တယ်ဆိုရင်တော့  
git remote add origin <server>  
အခု သင်ရဲ့ code ကို သင်ရွေးချယ်ထားတဲ့ remote server ပေါ်တင်လို့ရပါပြီ။

## branching

Branches တွေကို develop အဆင့်တွေမှာ အသုံးပြုပါတယ်။ *master* branch ကတော့ repository အသစ်ဖန်တီးတဲ့အခါမှာ ပုံသေပါလာပါတယ်။ တကယ်လို့ development အတွက် အြခား branches တွေခွဲထွက်ပြီး အကုန်ပြီးစီးမှသာ master branch နဲ့ ပြန်ပေါင်းပါတယ်။

"feature\_x" ဆိုတဲ့ branch အသစ်ဆောက်ပြီး ထို branch ကို ပြောင်းလဲ အသုံးပြုပုံ  
git checkout -b feature\_x  
master branch ကို ပြန်ပြောင်းပုံ  
git checkout master  
branch ကို ဖျက်ခြင်း  
git branch -d feature\_x  
branch ကို remote repository ကို push လုပ်လိုလျှင်  
git push origin <branch>

## အသစ်ဆွဲချခြင်း နှင့် ပေါင်းစည်းခြင်း

local repository ကို remote repository တွင် အသစ် commit လုပ်ထားသည်များကို ဆွဲချလိုလျှင်  
git pull  
*pull* လို့ ခေါ်တဲ့အခါမှာ working directory ထဲသို့ နောက်ဆုံး ပြောင်းထားသည်များကို *fetch* နှင့် *merge* လုပ်သွားပါမယ်။  
လက်ရှိ အလုပ်လုပ်နေသော branch (ဥပမာ။။ master) ထဲကို အခြား branch မှ လာပေါင်းလိုလျှင်  
git merge <branch>  
ထိုအခါ git က အလိုအလျှောက် ပြင်ထားသည်များကို ပေါင်းပေးပါလိမ့်မယ်။ သို့သော်လည်း အမြဲတန်း အဆင်မပြေပါ။ တစ်ခါတစ်လေ *conflicts* ဖြစ်တာလေးတွေ ရှိတတ်ပါတယ်။ ထို အခါ *conflicts* ဖြစ်တာကို ကိုယ့်ဘာသာကိုယ် ပေါင်းဖို့ လုပ်ဆောင်ရပါတော့တယ်။ *conflicts* ဖြစ်နေတဲ့ file များကို git တွင်ဖော်ထားပါတယ်။ ထို file များကို ကိုယ့်ဘာသာကိုယ် ပြင်ဆင်ပြီးရင် ပြန်ပြီးတော့ ထည့်ပေးဖို့ လိုပါတယ်။  
git add <filename>  
မပေါင်းခင်မှာ အရင်ဆုံး ပြောင်းထားတဲ့ code တွေကို ယှဉ်ကြည့်နိုင်ပါတယ်။  
git diff <source\_branch> <target\_branch>

## tagging

software version တစ်ခုကို ထုတ်လိုက်တိုင်းမှာ tag ကို အသုံးပြု အကြံပြုပါတယ်။ ဒါသုံးခြင်းက SVN မှာ ပါပြီးသား အယူအဆ တစ်ခုပါ။ သင့်အနေနဲ့ tag နာမည် အသစ် *1.0.0* ဆိုပြီး ပေးနိုင်ပါတယ်။  
git tag 1.0.0 1b2e1d63ff  
*1b2e1d63ff* ကတော့ သင် tag လုပ်ချင်တဲ့ commit id ရဲ့ ပထမ စာလုံး ၁၀ လုံး ပါ။ သင့် အနေနဲ့ commit id ကို အောက်ကလို ရိုက်ပြီး ရယူနိုင်ပါတယ်။  
git log  
သင့်အနေနဲ့ commit id ရဲ့ အခြားစာလုံး အနည်းငယ်ကို အသုံးပြုနိုင်ပါတယ်။ သို့ပေမယ့် အဲဒီ id ဟာ ဘယ်သူမှ အသုံးမပြုထားပဲ ကင်းရှင်းဖို့ သို့မဟုတ် unique ဖြစ်ဖို့ လိုပါတယ်။

## local ပြောင်းလဲခြင်းများကို အစားထိုးခြင်း

တကယ်လို့ တခုခုမှားသွားပြီး local ထဲက file နဲ့ ပြန် အစားထိုးချင်ရင်  
git checkout -- <filename>  
အခု ပြောင်းလဲလိုက်တာက သင့်ရဲ့ working tree ထဲက HEAD ထဲ က နောက်ဆုံး content မှာ သွားပြီးတော့ အစားထိုးသွားမှာပါ။ ပြောင်းထားတာတွေကိုတော့ index အနေနဲ့ ထည့်ပြီး files အသစ်တွေကိုလည်း သိမ်းထားပါလိမ့်မယ်။

အကယ်၍ သင့်အနေနဲ့ local changes နှင့် commits တွေကို ဖျက်ချပြီး နောက်ဆုံး server နှင့် master branch က အကုန် ဆွဲယူပြီး အစားထိုးချင်ရင်  
git fetch origin  
git reset --hard origin/master

## အသုံးဝင်သော အရာများ

GUI ပါဝင်သော git  
gitk  
အရောင်များဖြင့် git output  
git config color.ui true  
commit များကို စာ တကြောင်းစီဖြင့် ကြည့်ရန်။  
git config format.pretty oneline  
interactive adding ကို အသုံးပြုခြင်း  
git add -i

## link နှင့် resources

### **graphical clients**

* [GitX (L) (OSX, open source)](http://gitx.laullon.com/)
* [Tower (OSX)](http://www.git-tower.com/)
* [Source Tree (OSX, free)](http://www.sourcetreeapp.com/)
* [GitHub for Mac (OSX, free)](http://mac.github.com/)
* [GitBox (OSX)](https://itunes.apple.com/gb/app/gitbox/id403388357?mt=12)

### **အသုံးပြုပုံ လမ်းညွှန်များ**

* [Git Community Book](http://book.git-scm.com/)
* [Pro Git](http://progit.org/book/)
* [Think like a git](http://think-like-a-git.net/)
* [GitHub Help](http://help.github.com/)
* [A Visual Git Guide](http://marklodato.github.com/visual-git-guide/index-en.html)