# GIT -н талаар ойлголттой болох нь

**GIT гэж юу вэ?**

Git гэдэг бол програмын сорс код болон ямар нэг файлд орсон өөрчлөлтийн түүхийг бичиж үлдээх зорилготой тархсан хувилбар удирдах систем юм. Linux Kernel-н сорс кодыг удирдах зорилгоор Linux Kernel-г хөгжүүлсэн Linus Trovalds гэдэг хүн анх үүсгэн хөгжүүлсэн. Түүнээс хойш маш олон том том прожектууд GIT хувилбар удирдах системийг ашиглан хөгжиж байна.

Git ашиглаж буй хэрэглэгч бүр өөрийн ажлын замд бүх түүхийг агуулсан байх ба үндсэн Repository-тай яг адилхан Repository-г өөр дээрээ үүсгэнэ. Тийм учраас сүлжээний асуудал зэргээс хамаарч үндсэн Repository-д хандаж чадахгүй орчин байлаа ч түүхийг харах, өөрчлөлт оруулах зэрэг ихэнхи ажлаа хийх боломжтой байдаг. Энэ нь тархмал систем гэж нэрлэгдэх болсон нэг шалтгаан юм.

**GIT үүсэх болсон шалтгаан.**

Ерөнхийдөө 2 шалтгаанаар GIT үүссэн гэж тайлбарласан байхыг уншсан. Нэг нь файл өөрчлөх, нөгөө нь багаар ажиллах гэсэн хоёр зүйл GIT ашиглахгүй бол нэлээд төвөгтэй болж ирсэн юм байна. Жишээ нь бидний өөрчлөлт хийж буй файл бидэнд аль ч үеийнхээр хэрэг болж болно үүнийг гараар тухайн бүрд өөрчлөлт хийсэн он сар өдрөөр нэрлээд хадгалж явна гэдэг нэлээд төвөгтэй зүйлийг GIT дээр маш хялбар хүссэн үеийнхээ хувилбараар авах боломжтой байдаг. Мөн нэг файл дээр багаараа буюу хоёроос дээш хүн зэрэг өөрчлөлт хийх боломжийг олгож байгаа юм. Өнөөг хүртэлх бусад хувилбар удирдах систем дээр хоёр дахь хүн нь эхний хүний хийсэн өөрчлөлтийг дараад хадгалах асуудал нэлээд гардаг байсан байна.

**GIT-н давуу тал**

GIT－н хамгийн том давуу тал нь тархмал систем гэдэг нь юм. Өөрийн компьютер дээрээ засвар болон түүхийг агуулсан үндсэн хуудсыг хуулаад авчих боломжтой юм. Энэ нь өөрөөр хэлбэл үндсэн фолдерийг өөр дээрээ хуулаад чөлөөтэй засвар хийж болно гэсэн үг юм. Remote-Repository-г clone хийж аваад Local-Repository үүсгээд үүн дээрээ засвар өөрчлөлтүүдээ хийгээд явна гэсэн үг.

**GIT дээр юу юу хийж чадах вэ?**

**Файлын өөрчлөлтийн түүхийг удирдах боломжтой.**

GIT ашиглавал хэдэн сарын хэдэнд хэн ямар өөрчлөлт хийсэн гэсэн өөрчлөлтийн түүхийг удирдаж чадах учир файлын нэрийг нэг нэгээр нь өөрчилж хадгалах шаардлага байхгүй. Файлын тоо нэмэгдэхгүй учраас өөрийн орчин нь цэмцгэр байна.

**Өмнөх файл руу буцаах боломжтой.**

Файлыг өөрчилсөний дараа яалт ч үгүй өмнөх файл руу буцаах хэрэгтэй тохиолдол гардаг. GIT ашиглавал өөрчлөлтийн түүхийг бүгдийг үлдээж явж байгаа учраас хэдийд ч аль үеийн файл руу буцаах боломжтой.

**Git-р удирдах боломжтой зүйл нь** зөвхөн програмын сорс код биш эксэл файл, зураг зэрэг маш олон файлыг удирдах боломжтой. Тэгэхээр зөвхөн IT инженерүүд биш бусад салбарын файл дээр их ажилладаг баг хамт олон ч GIT ашиглавал давуу тал ихтэй гэсэн үг.

**Интернэт орчин ашиглаад** багийн гишүүдтэйгээ файлын өөрчлөлтийн түүх зэргийг адилхан харах боломжтой. Олон хүнтэй ажиллахад гардаг бэрхшээлээс зайлсхийх боломжтой.

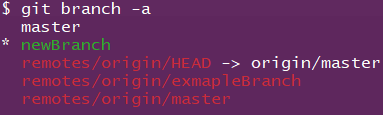
**GIT дээр ашиглагдах үндсэн нэр томъёо болон үндсэн үйл ажиллагаа**

GIT ашиглахад үндсэн гурван зүйлийг татаж авч суулгах юм уу, аккаунт нээх хэрэгтэй болдог байгаа. Нэр томъёо, үйл ажиллагааны талаар ярихын өмнө энд товч дурдаад дараа арай дэлгэрэнгүй яръя.

1. Bitbucket веб сервис — Үүний оронд GitHub зэрэг өөр веб сервис байсан ч болно. Remote Repository буюу үндсэн Repository байрладаг веб сервис юм.
2. SourceTree — GIT-н үйлдлүүдийг windows дээр хялбар ашиглах боломжтой болгосон хэрэгсэл. Үүний оронд маш олон түүл байгаа. Алийг нь сонгосон ч болно байх. Бусдыг нь хэрэглэж үзээгүй болохоор сайн мэдэхгүй байна.
3. Тэгээд мэдээж GIT-ээ татаж аваад суулгах шаардлагатай.

**Git Flow Branching Model**

Branch үүсгэх:

git-flow нь Vincent Driessen гэдэг хүний гаргасан [A successful Git branching model] гэдэг бранч моделийг дэмждэг түүл буюу командын нэр юм. Ерөнхийдөө модел болон түүл аль алиных нь нэршилээр ашиглагддаг. git-flow-д үүрэг рол нь тодорхой 5 төрлийн бранч байдаг. Зарим тохиолдолд 6 төрөл байдаг. Эдгээр бранчуудын хооронд шилжиж тохирсон бранчаа сонгож хөгжүүлэлтээ хийгээд явдаг. Бранч шинээр үүсгэх, мерж хийх зэрэгт дүрэм зохиож өгснөөр олон хүн хөгжүүлэлтийн баг байлаа ч бранчийг илүү ойлгоход хялбар байдлаар барьж чадах ба амжиллтай биш мерж хийхээс зайлсхийж чадна. git-flow бранчийн төрлүүдтэй танилцацгаая.

**Master** : Бүтээгдэхүүн болгон нийлүүлэхэд бэлэн сорс

**Develop:**Хөгжүүлэлт хийгдэж буй хамгийн сүүлийн версион (Integration test хийгдэж байгаа үеийн сорс гэж ойлгож болно)

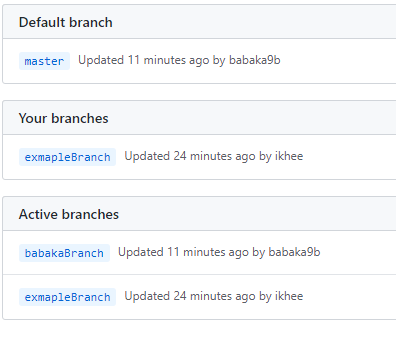
**feature:** функцийн хөгжүүлэлт (хөгжүүлэлт болон unit test-н үед хэрэглэх бранч гэж ойлгож болно) Develop branch-с салбарлаж, Develop branch руу нийлж байгаа branch-г хэлж байгаа. Яг хэзээ Develop branch-тайгаа нийлэхийг Прожект менежер нь шийдэж өгнө. Ерөнхийдөө feature branch дээр хийгдэж буй ажил алдаагүй, код бичиглэлийн хувьд төгс гүйцэтгэлтэй болсон хойно Develop branch-руу нийлүүлдэг.

**release:**Release хийхийн тулд бэлдсэн branch(Setting файл доторх версионы мэдээллийг засварлах гэх мэт) Develop branch-с салбарлуулаад develop болон мастер branch-д мерж хийж болно. Release хийхэд бэлэн болсон хойно бранч салгана гэдэг хоёр салаа болчих учраас Прожект менежер хэзээ яаж нийлүүлэхээ дахин тодорхойлж өгөх хэрэгтэй.

**hotfix:** Програм ашиглагдаж эхэлснээс хойш гэнэтийн засвар шаардлагатай үед hotfix branch үүсгэж засвар хийнэ. Master branch-с салбарлуулж Master branch руу нийлүүлнэ. Develop.с салбарлуулах юм бол хөгжүүлэлт хийгдэж байхдаа Release хийгдэх тул заавал Master branch-с салбарлуулна.

**GIT дээр ашиглагдах нэр томъёо**

1. **Remote-Repository (Алсын агуулах гэж орчуулсан байдаг юм билээ)**

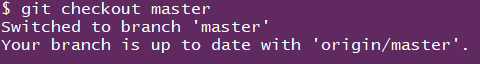


Server дээрх Repository буюу GitHub, Bitbucket зэрэг веб сервис дээр байрлах үндсэн Repository. Энэ Repository-г өөрийн компьютер тэр чигт нь хуулж аваад өөрийн хуулж авсан Repository дээрээ өөрчлөлт шинэчлэлт хийгээд дараа нь үндсэн Repository-руугаа хийнэ. Шууд Remote Repository-руу өөрчилөлт шинэчлэлт хийгээд байдаггүй.

1. **Local-Repository**

Өөрийн компьютер дээрээ хуулж авчирсан Repository. Remote Repository-г өөрийн компьютерийн ямар нэг замд clone хийж авчирсан гэсэн үг. Эндээ хөгжүүлэлт хийгээд явнаа гэсэн үг.

1. **Remote-Branch**



Серверт байрлах бранч. Repository маань Branch-с бүрдэж байгаа. Master Branch, Development Branch гэх мэт

1. **Local-Branch**

Өөрийн компьютер дээрх бранч

1. **index**

Тухайн сорс болон тухайн файлыг working-tree файлаас index рүү хийнэ. index-д орсон үеийн байдлаар локал repository -руу коммит хийгдэж орно.

1. **working-tree/working-copy**

Clone хийгдсэн замд ямар өөрчлөлт орно тэр бүх өөрчлөлт working-tree -рүү автоматаар ороод явна. stage/add хийвэл index-рүү орно.

1. **stash**

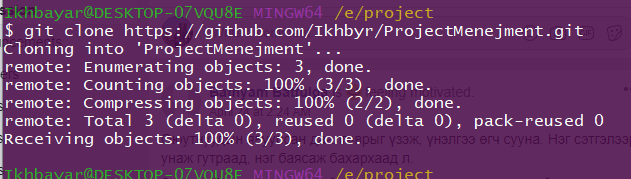
index -д байгаа файлыг түр хадгалах хэсэг

**GIT-н үндсэн үйл ажиллагаа**

1. **Fork**

Remote дээр Remote Repository-г хуулбарлана. Fork хийх үеийн сорс, бранчийн бүтэц хуулагдана. Fork-с өмнөх түүхэн мэдээлэл хуулбарлагдахгүй. Энэ үйлдлийг хийж үзээгүй учир яг сайн мэдэхгүй байна.

1. **Clone**



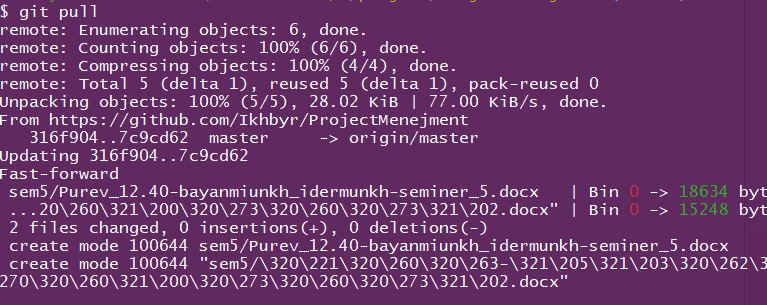
Локал руу бранчийг хуулбарлана. Clone хийх remote branch болон repository-г зааж өгнө. Мөн Clone хийсэн зүйлээ байрлуулах Local замаа зааж өгнө. Clone хийж Local-Repository-оо үүсгэнэ.

1. **Merge**

Development Branch-ээ Master Branch руугаа нэгтгэхэд энэ үйлдлийг ашиглана. Branch-н тухай доор бага зэрэг дэлгэрүүлж бичсэн байгаа.

1. **Fetch**

Remote branch-н статусыг remote tracking branch-руу хуулбарлана.  
мерж хийхгүй тул өөрчлөлтийг шалгахын тулд pull хийхийн оронд fetch хийсэн нь илүү дээр байдаг. SourceTree дээр fetch нь автоматаар хийгддэг.

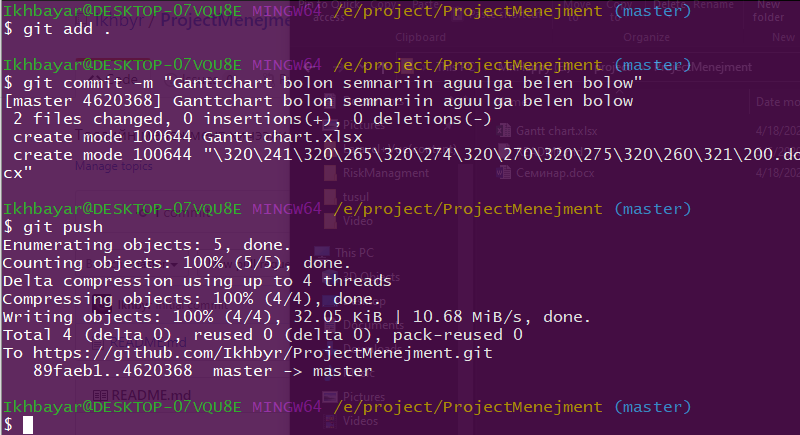
1. **Pull**

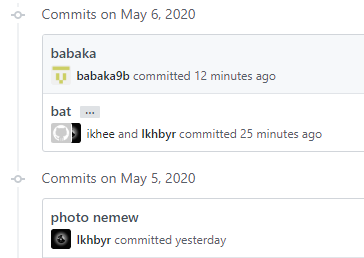
fetch + merge үйлдэл юм. Remote branch-н агуулгын өөрчлөлтийг хяналгүй нэг дор Local-Repository -руугаа мерж хийх тохиолдолд энэ үйлдлийг ашиглана. Багаар ажиллаж байлаа гэж бодоход багийн бусад гишүүд ч Remote-Branch дээр өөрийн хийсэн өөрчлөлтүүдээ оруулж нэгтгээд явж байх учир тэдгээр өөрчлөлтүүдийг тухай бүрт нь өөрийн локал руу Pull хийж нэгтгээд өөрийн Repository-оо сүүлийн байдлаар шинэчлээд явна гэсэн үг.

1. **add/stage**

Засвар хийсэн файлаа коммит хийх үед энэ үйлдлийг ашиглана. эхлээд stage хийлгүй коммит хийх боломжгүй. Мөн stage хийсний дараа хийсэн тухайн файл дээрх өөрчлөлт коммит хийгдэхгүй.

1. **Commit**



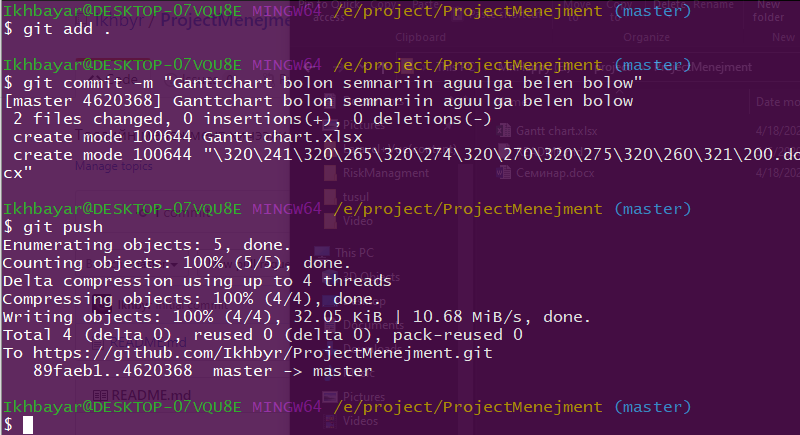


Commit хийгдсэн байдал

add/stage хийгдсэн өөрчлөлтийг локал бранч руу хийнэ.

1. **Push**

Локал бранчийн агуулгыг remote branch руу оруулж өгнө.



1. **stash(stash save)**

index болон working tree -д байгаа өөрчлөлтийг stash гэж нэрлэгдэх бүс рүү зөөнө. энэ үйлдлийг хийх юм бол index болон working tree нь local branch-тэй адилхан болно. (өөрчлөлтийг хийхээс өмнөхтэй адилхан болно.) stash хийх бүрд stash-рүү тус тусдаа хийгдэнэ. (өмнө нь хийгдсэн зүйл дээр давхар бичихгүй мөн мерж хийгдэхгүй )  
stash хийсний дараа хадгалахгүй байж болдог автоматаар хадгалагддаг учраас. Мөн working tree-с шууд stash хийх боломжгүй бөгөөд эхлээд add/stage хийгээд index дотор байгаа үед stash хийх боломжтой.

1. **stash apply**

Хадгалсан статусаа Working Tree -рүү буцаах үйлдэл. Хэт олон stash хийх юм бол conflict үүсэх магадлал өндөр тул нэг дор их олныг хийхгүй байх нь зөв. Шинэ файлын тухайд зөвхөн working tree -рүү биш index рүү мөн хийдэг***.***

**GIT суулгах нь**

Linux дээр ажилладаг GIT-с гадна Windows дээр ажилладаг Git for Windows болон Mac OS дээр суулгахад зориулсан GIT байгаа учраас дуртай орчин дээрээ ашиглах боломжтой.

**Git Bash**

Git for Windows –г суулгавал Git Bash гэдэг түүл давхар суугддаг. Git Bash бол Windows－н команд промпттэй төстэй түүл бөгөөд Linux-д их хэрэглэгддэг ls гэх мэт командуудыг хэрэглэх боломжтой. GIT-г ашиглах аппликешн бол биш бөгөөд GIT-г ашиглах шелл скриптийг windows дээр хэрэглэх түүл юм. Анх Linux систем дээр хийгдсэн GIT-г Windows дээр ажиллуулахад хэрэгтэй юм байна гэж ойлгосон. Бүр нарийн сайн судлаагүй.

**Windows дээр ашиглах боломжтой GIT түүл**

GIT-н үйлдлүүдийг ашиглах боломжтой аппликешн. CUI(Command User Interface) ашиглаад GIT-г ашиглаж болох боловч эдгээр түүлийг хэрэглэсэнээр CUI ашиглахгүйгээр илүү хялбар Windows аппликешний меню дараад ашиглаад явчих боломжтой юм. Дээр source tree гэж дурдсан бөгөөд pull, push, commit, clone гэх мэт үйлдлүүдээ энэ аппликешн дээр хийгээд явах боломжтой юм. SmartGit, GitKraken, Fork, GitExtensions, Magit, TortoiseGit, tig, SourceTree гэх мэт тоо томшгүй олон ийм төрлийн аппликешн байдаг. Миний хувьд SourceTree-г ашиглаж байсан байгаа. Мөн Монголд байхдаа CUI дээр сорсоо pull хийж аван тест хийдэг байсанаа санаад байгаа юм.

**GIT-д ашиглах веб сервис**

Веб сервисийн талаар ярихын өмнө GIT болон GitHub нь адилхан зүйл мөн үү? гэсэн асуултанд хариулъя. Яалт ч үгүй хамт яригддаг, нэр нь төстэй болохоор хүмүүс их андуурдаг боловч адилхан зүйл биш юм. GitHub бол Git механизмыг ашиглан дэлхий даяарх хүмүүс програмын код болон бусад датаг хадгалах зориулалттай веб сервис юм. Ажиллуулж байгаа компани нь GitHub компани. GitHub-тай адил web service хэд хэд байдаг бөгөөд жишээ нь Bitbucket юм. Bitbucket бол Atlassian компанийн ажиллуулдаг WebService юм. Миний хувьд Bitbucket дээр ажиллаж үзсэн. Гэхдээ нэг их ялгаагүй байх гэж бодож байна.