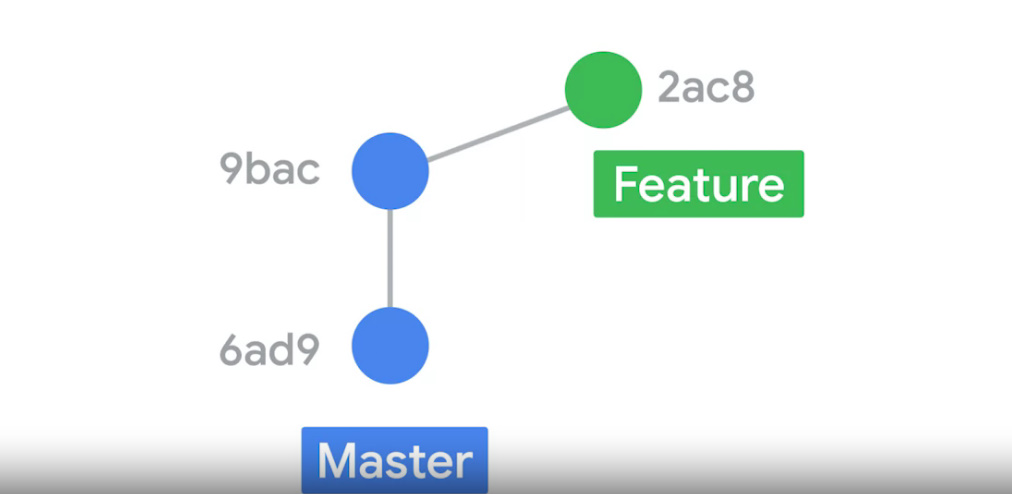
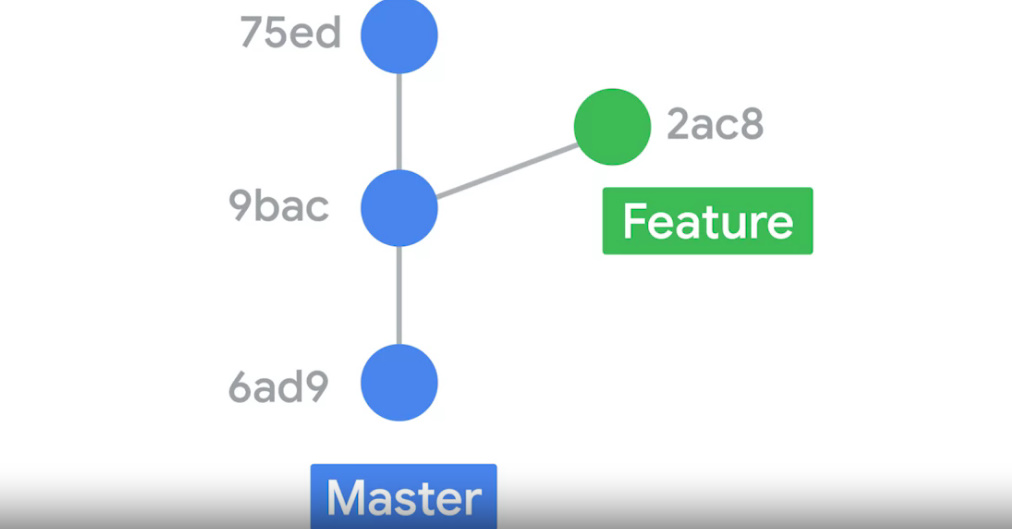
* Odata ce branch creata a fost testata si finalizata, putem sa-i dam merge in master branch. Folosim git merge, dar mai e si alta metoda.
* git rebase – alterntiva pentru git merge
* Rebase – inseamna sa schimbam commitul de baza care e folosit de branch
* Cand cream un branch, git mereu stie care e ultimul commit trimis pentru ambele brnanches
* Cand doar una din ele are noi modificari cand incercam sa dam merge, git poate folosi **Fast Forward** merge si sa aplice modificarile



Insa, cand ambele au modificari cand incercam sa dam merge, git va incerca sa le combine cu **Three Way Merge** si sa creeze un nou commit

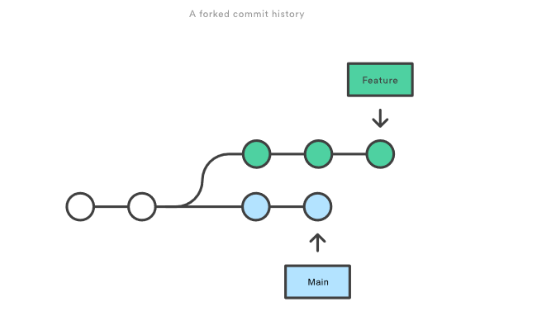


**Rebase**

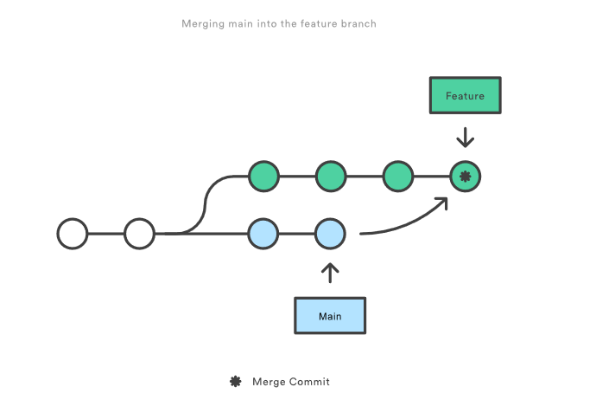
Git merge pur si simplu creaza un nou commit in varf la branch.

De ex:

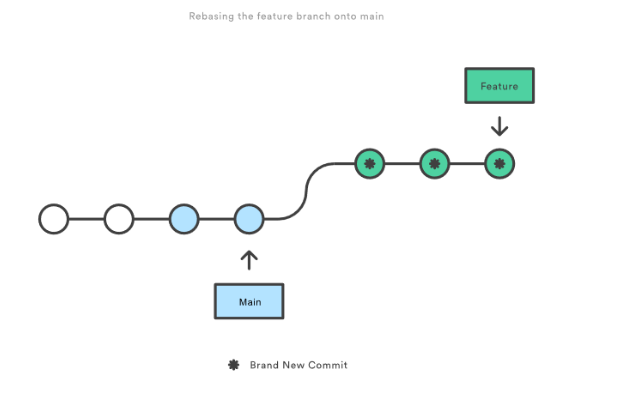
Sa zicem ca noi cream un branch Feature care porneste de la Main branch. Noi facem ceva modificari acolo, insa in acest timp cineva da merge la Main si Main are alt commit ca cel mai nou. Deci, branch Feature ramane in urma.

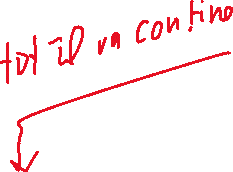


Daca dam merge in Feature branch, adica git merge Main, ultimul commit din Feature se va combina cu cel din Main, si daca va reusi, se va crea un commit nou in Feature



Totusi, git rebase ne va da un rezultat asemenator, dar va face putin altceva. El nu va crea pur si simplu un nou commit combinand cel mai recent commit din Feature cu cel mai recent din Main, dar va adauga noul commit cel mai recent din Main in branch,**cu totul cu cele noi de mai inainte de el ce tot nu au fost puse in child branch,**dar nu va fi primul, ci dupa cel care a fost pana atunci commit de baza in Feature, apoi va rescrie istoria, si anume prin combinarea noului commit curent(si cele noi de pana la el) din Main cu fiecare din Feauture, adica si ultimul commit din Feature, si penultimul...si primul.





Deci, daca in feature erau 3 commits(fara cel din main), si in main 2 noi, utilizand merge vor fi deja 5, astfel incat primele 3 raman cum erau si al 4 si al 5 va contine modificarile facute in Main in commiturile cel mai noi. Insa, git rebase tot va avea 5, dar va adauga cele mai nou commituri din branch Main dupa fostul commit de baza care nu mai e de baza in Feature,dar a fost cand am creat branch sau dupa ultimul merge, apoi pe baza lui va crea cate un nou commit din cele din istorie, stergand fiecare commit din istorie si inlocuindu-l cu unul nou, care contine tot ce a avut acel commit din Feature plus ceea ce se afla deja in ultimul commit din Main ce a fost pus in acel branch inaintea acelor commituri ce vor fi rescrise, dar dupa commitul care a fost cel mai nou cand am facut acele commituri in branch parinte. Asa ca si cum branch a noastra Feature a inceput chiar de la commiturile cele mai noi acum in Main,si apoi dupa el au mers urmatoarele commituri din istorie, desi in realitate nu a fost asa

* Rebase e bun pentru a pastra o istorie liniara a proeictului. Asa orice branch va parea ca a inceput cu cel mai recent commit din Main(si cele de sub el noi),asa cum el va fi adaugat dupa fostul cel mai recent commit de la care a inceput brnahc, indiferent de cate ori a ramas in urma. Asa istoria ramane curata si e usor de facut apoi analize. E important sa o folosim pentru a integra deodata schimbarile facute de altii si in commiturile anterioare, caci daca vom avea nevoie sa revenim la vreun commit anterior, si el nu are acele modificari, poate fi greu de le pus.
* **Git da merge(la rebase) automat unde poate**, dar In cazul in care apare un merge conflict, va trebui sa il rezolvam manual pentru fiecare commit din istorie.Dupa aceea vom da git add si git rebase –continue dupa ce terminam
* Git rebase va permite ca sa dam merge in Main branch pe baza la test branch fara sa fie, la sigur, vreun merge conflict, asa cum test branch la sigur deja are ca commit de baza cel mai recent din Main care e combinat cu commiturile din branch ce se afla dupa locul in care a fost pus el in test branch.

Ex:

Main

commit Main 1

commit Main 2



New Test Branch

commit Main 2

commit Test 1

commit Test 2

**Modificam Main**

Main

commit Main 1

commit Main 2

commit Main 3

commit Main 4

git rebase Main in Test, si deci vom avea:



Test



commit Main 2

commit Main 3

commit Main 4

commit Test 1 + Main 4

commit Test 2 + Main 4

**Putem folosi git fetch + git rebase Nume**