SmartGit

Contents

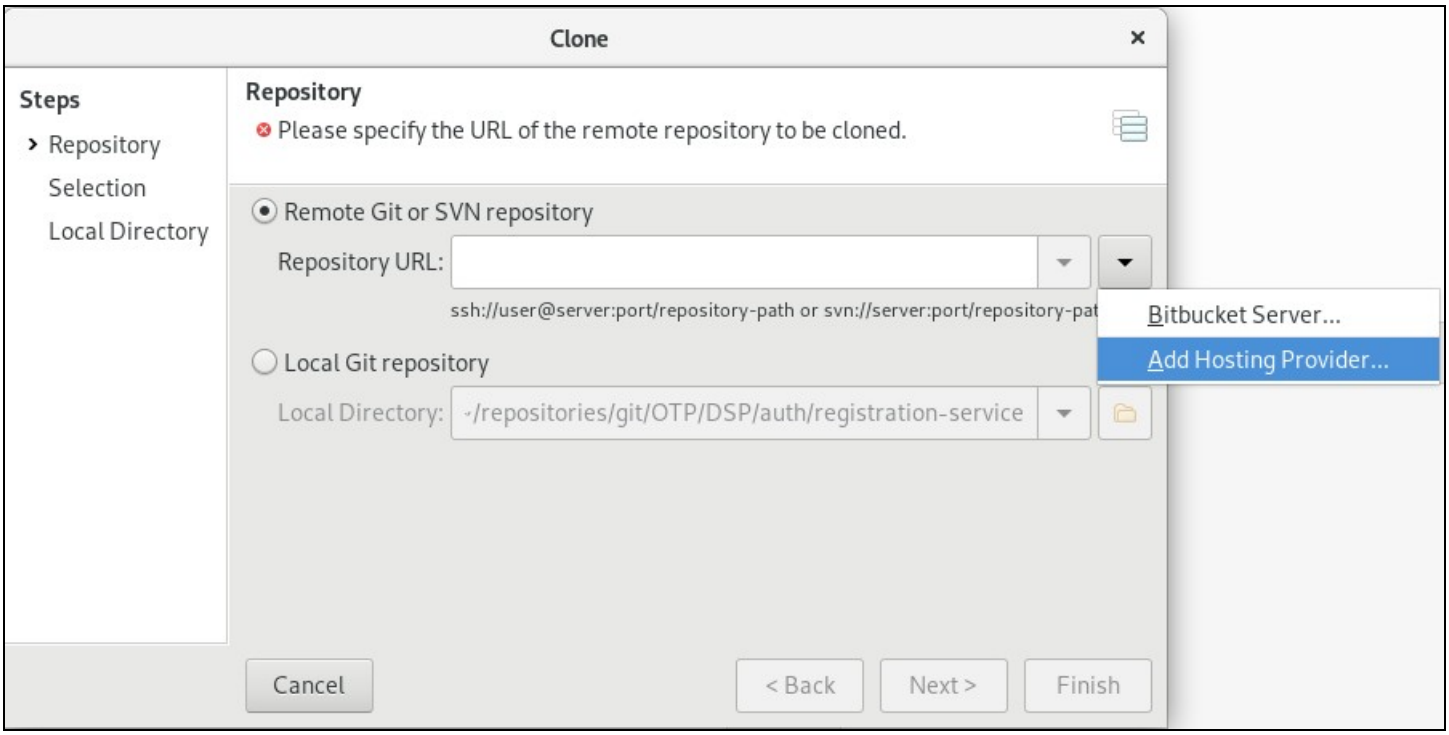
Új BitBucket repo klónozása

A SmartGit sok féle GIT szolgáltató integrációját tartalmazza. Lehet?ség van rá, hogy egy Git szolgáltatót mint pl a BitBucket felvegyünk mint **Host Provider**, ami kés?bb lehet? teszi, hogy klónozásnál egyszer?en válogathatunk az elérhet? repók listájából anélkül, hogy tudnánk mi az adott repo clone URL-je. Fel fogunk venni egy BitBucket szolgáltatót.

Állítsuk be, hogy ne legyen SSL verifikáckó. Ezt a saját user-ünk alatt kell futtatni:

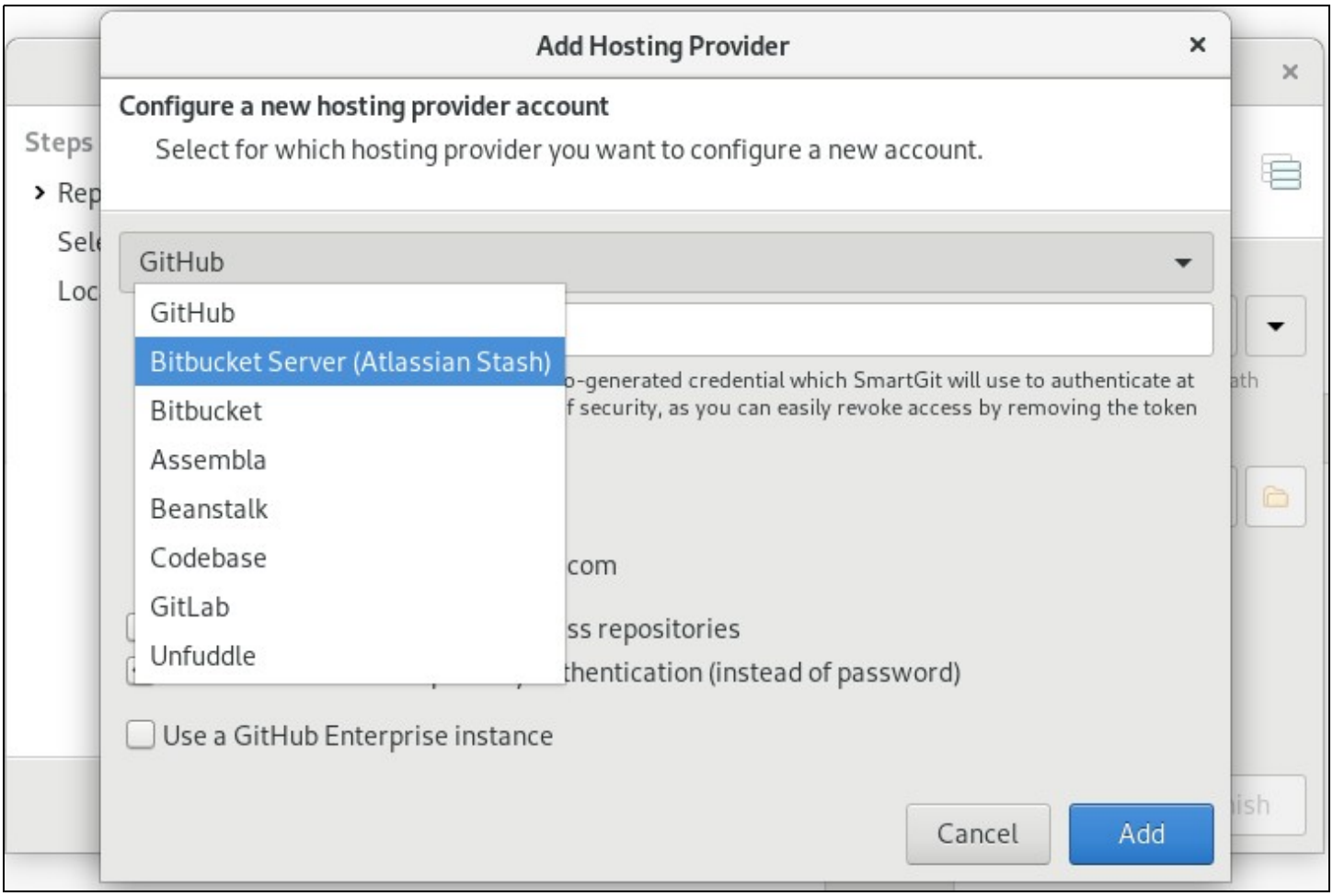
```
$ git config --global http.sslVerify false
```

Válasszuk a Repository -> Clone lehet?séget.

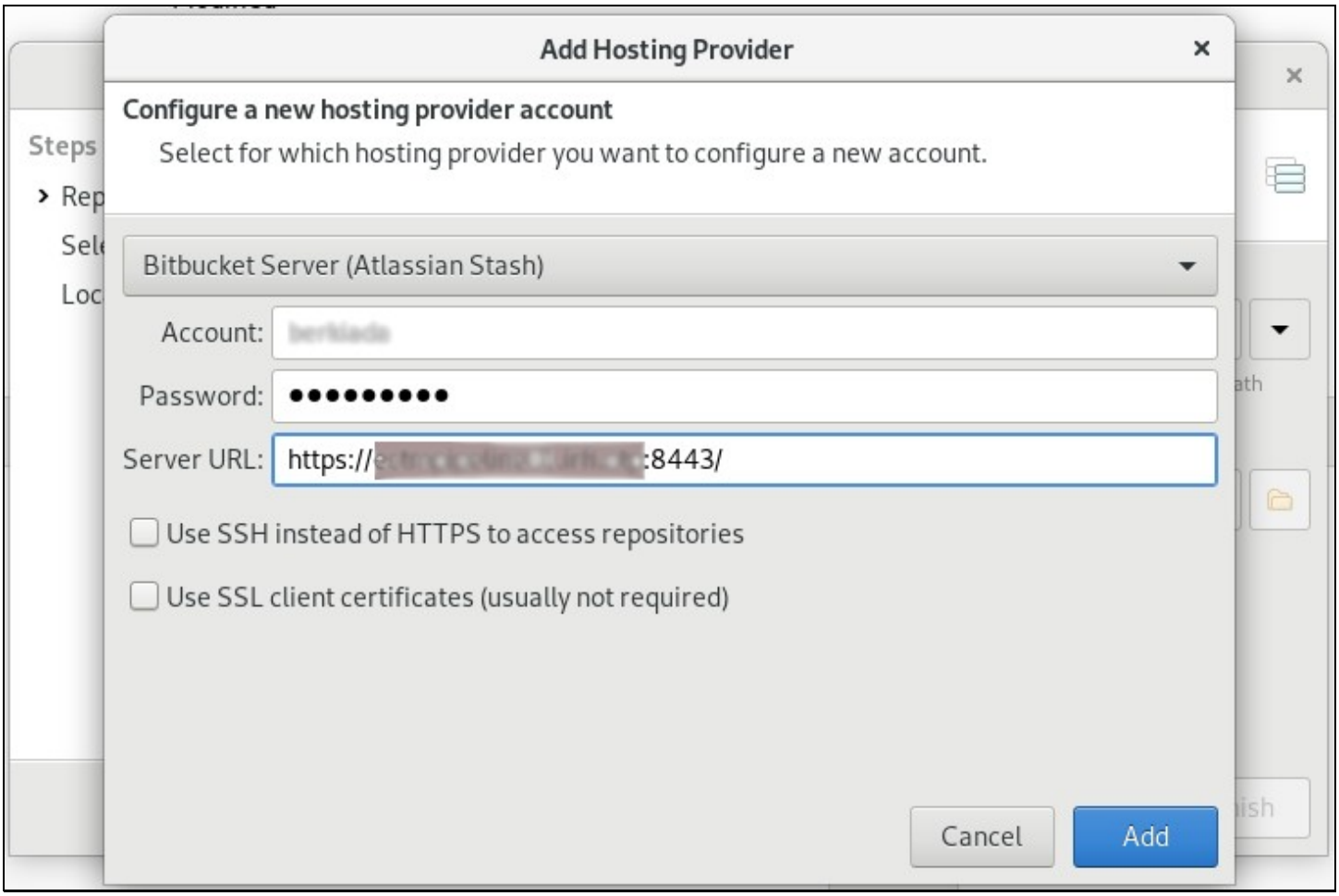


Kattintsunk a **Repository URL** melletti mez? kis fekete nyílra, majd válasszuk az **Add hosting provider** lehet?séget.

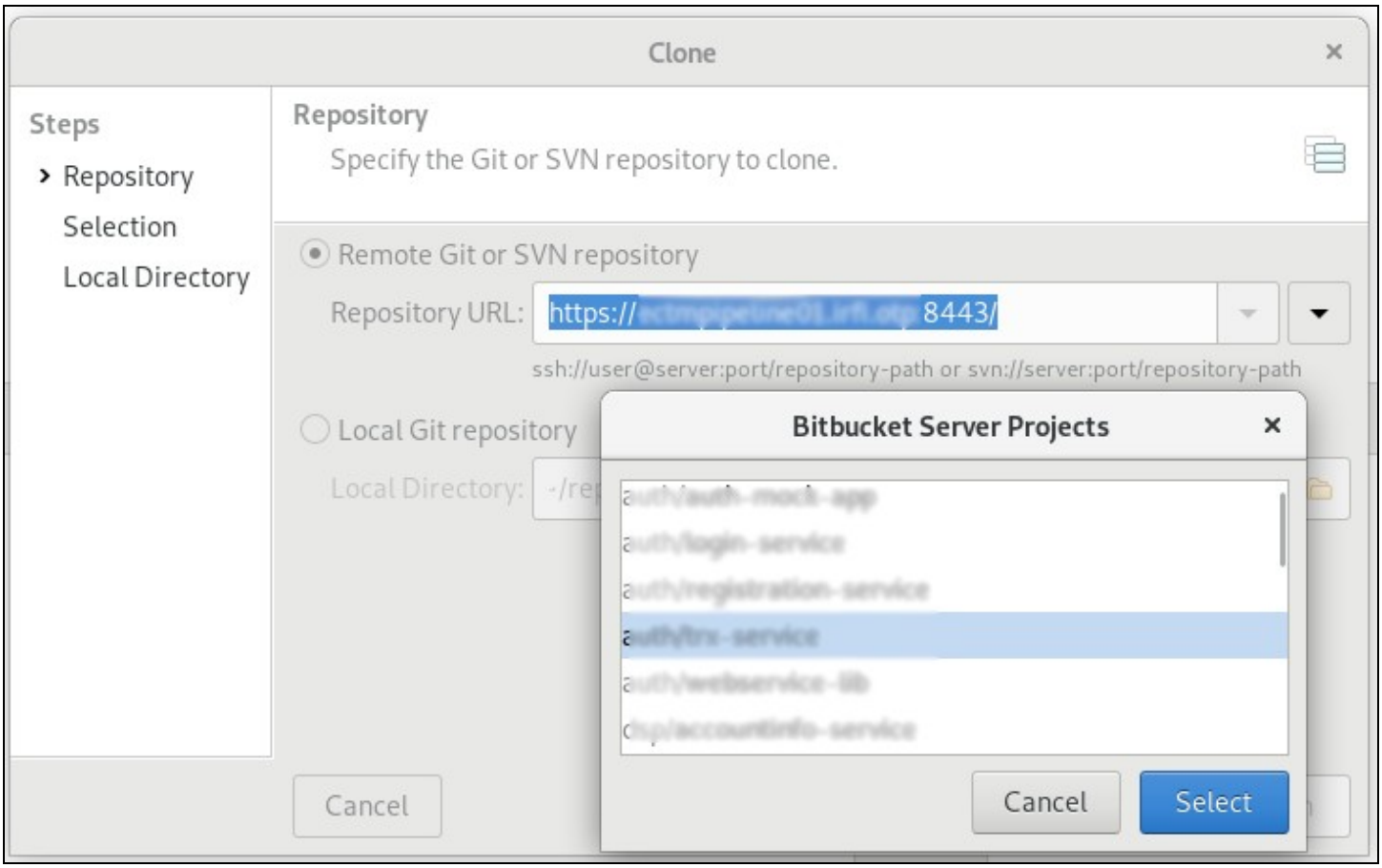
Itt válasszuk a BitBucket server (Atlassian Stash) lehet?séget.



Adjuk meg a felhasználó nevünket és a jelszót, és a szerver base URL-jét, vagyis csak a hoszt nevet és a portot (ha nem szabványos a port):

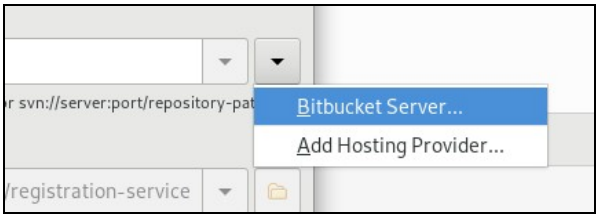


Ekkor be fog jelentkezni a SmartGit a Bitbucket-be és az összes projektet ki fogja listázni, amihez van hozzáférésünk:

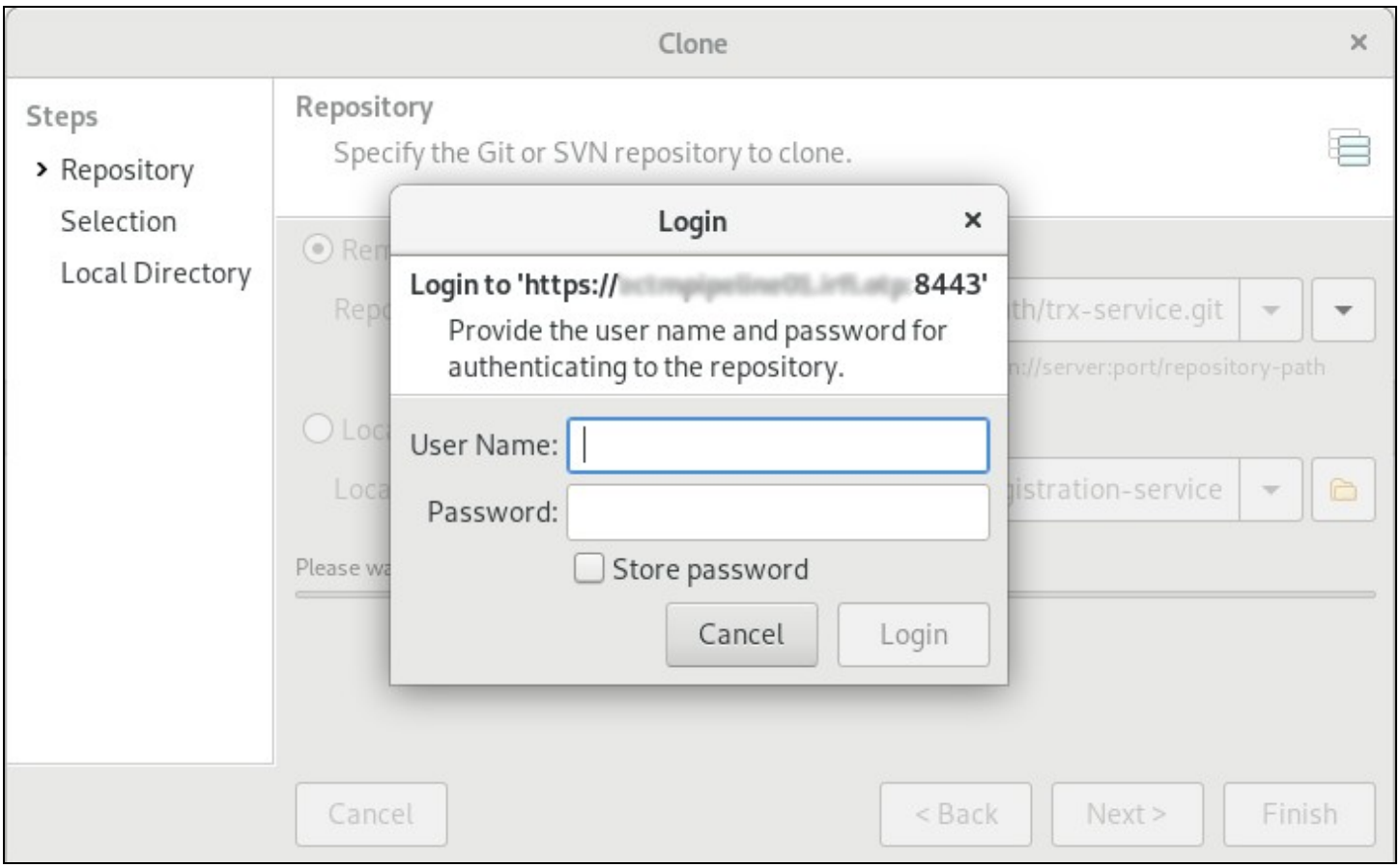


Nincs más dolgunk, mint hogy kiválasszuk a szükséges repo-t a listáról.

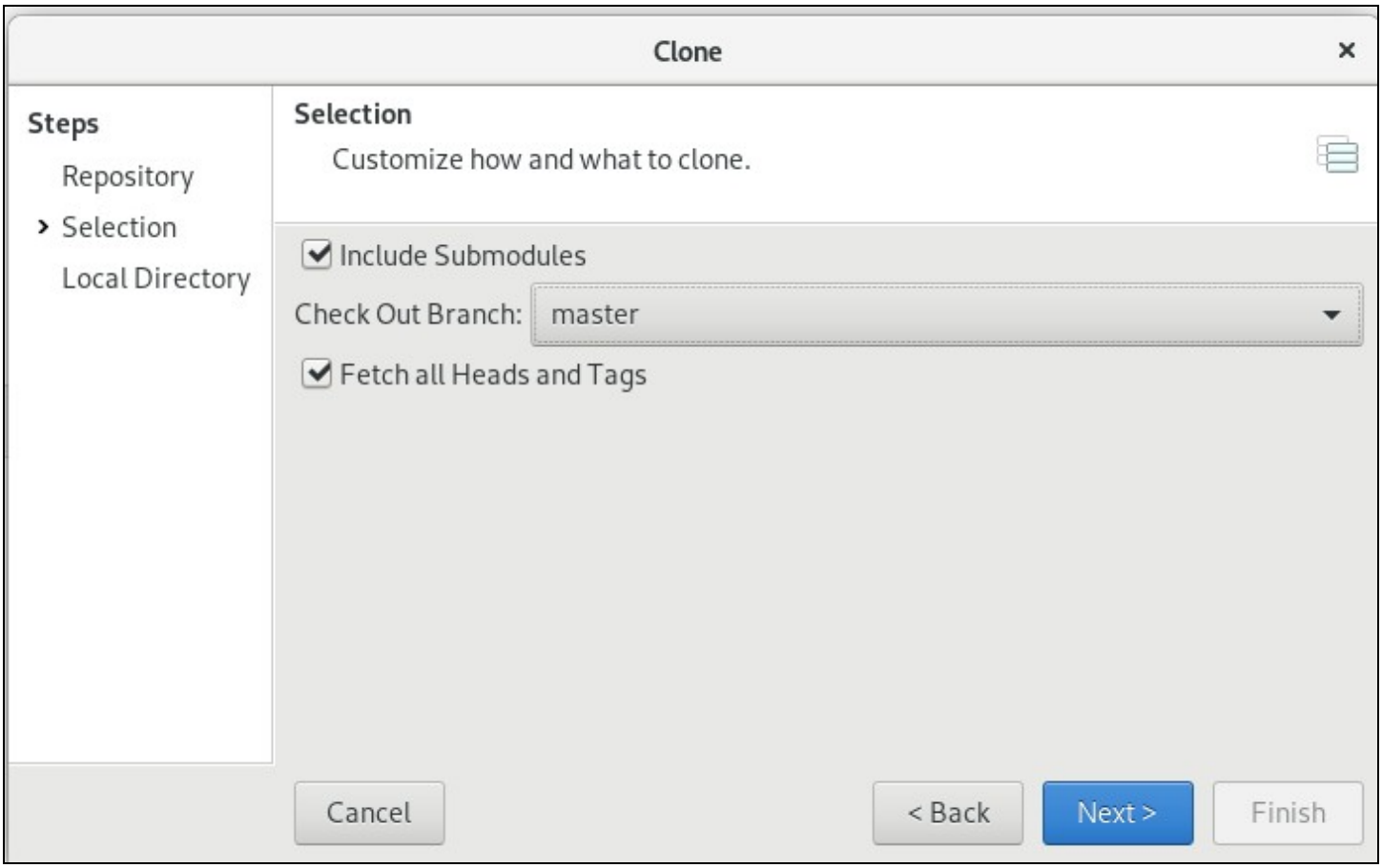
A jöv?ben ez a BitBucket provider azonnal elérhet? lesz, nem lesz rá szükség hogy a szerveren kiválasszuk hogy mit akarunk klónozni, csak a Repository URL melletti kis fekete nyílra kell kattintani, és a lenyíló listában a BitBucket server-t kell választani, és a megnyíló ablakban már válogathatunk az elérhet? repository-k között.



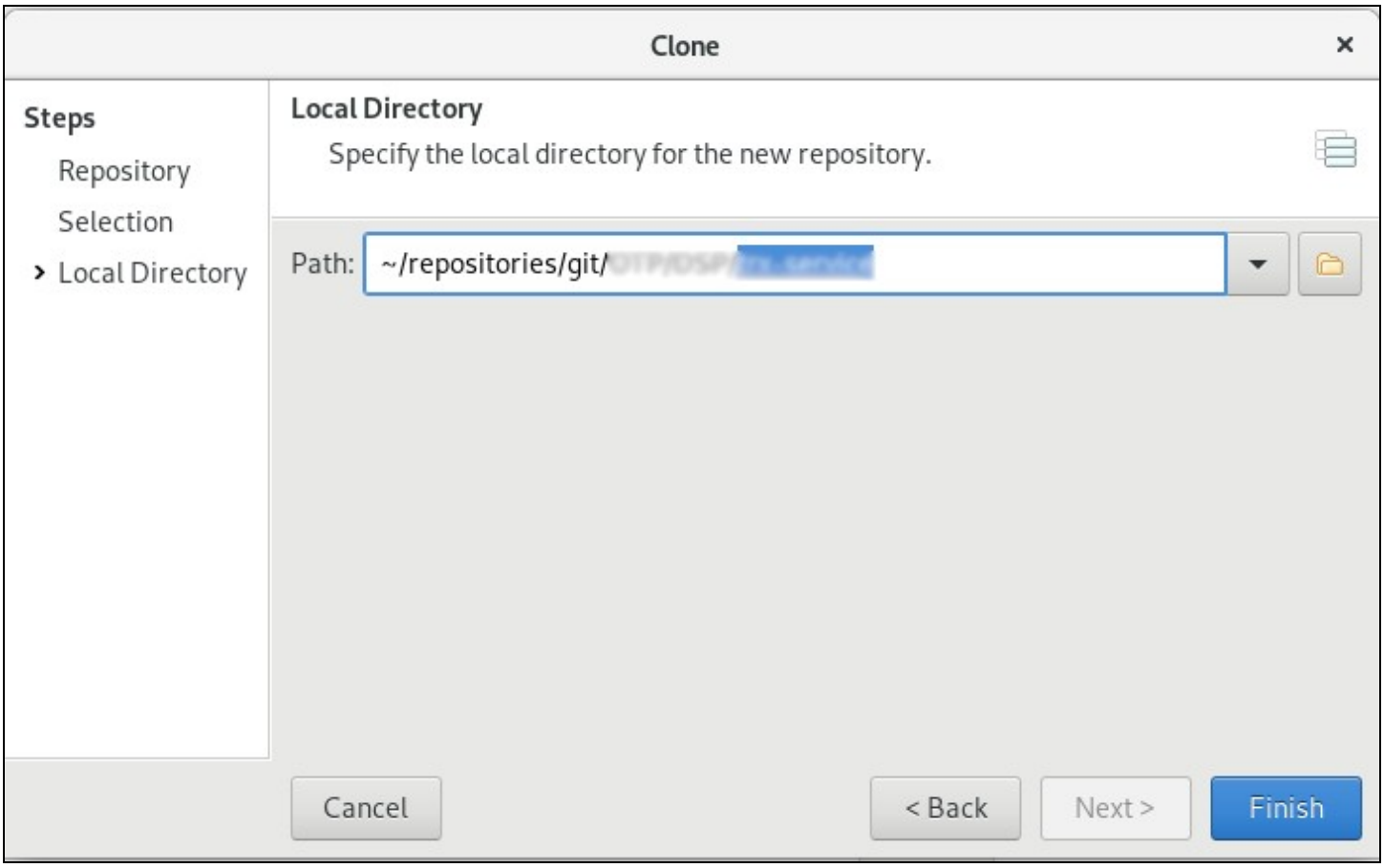
A következ? oldalon adjuk meg a felhasználó nevet és a jelszót:



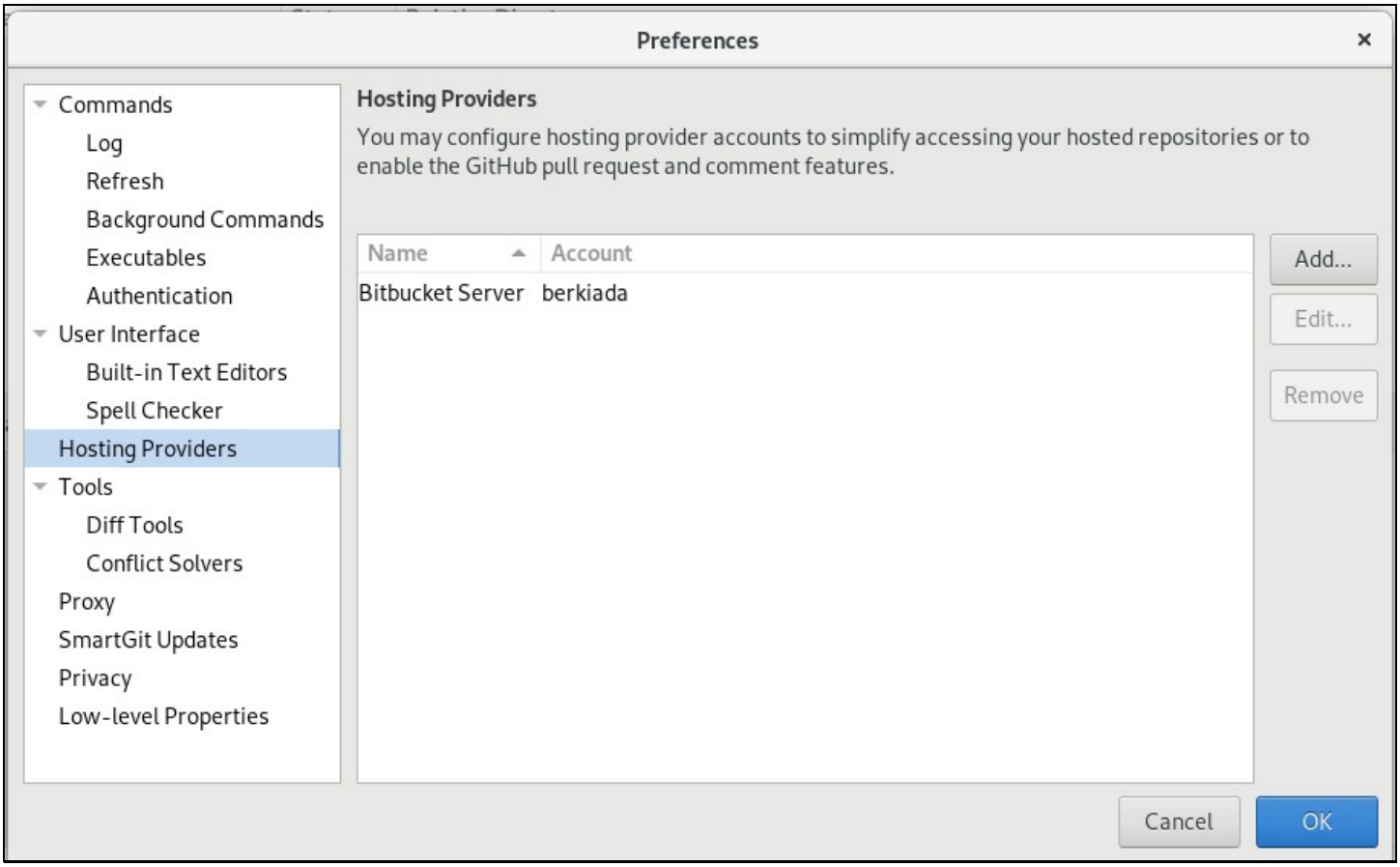
Válasszuk ki, hogy mit akarunk checkout-olni a kezdésként:



Majd válasszuk ki a lokális mappát a repositroy-nak. A SmartGit a korábbi választásaink alapján általában tökéletes ajánlatot fog adni:



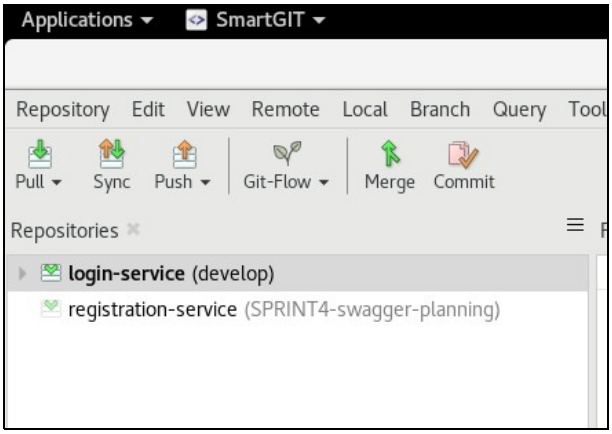
A már felvett Hosting Provider-ek listáját megtekinthetjük az **Edit -> Preferences -> Hosting Providers** menüpont alatt:



Felület bemutatása

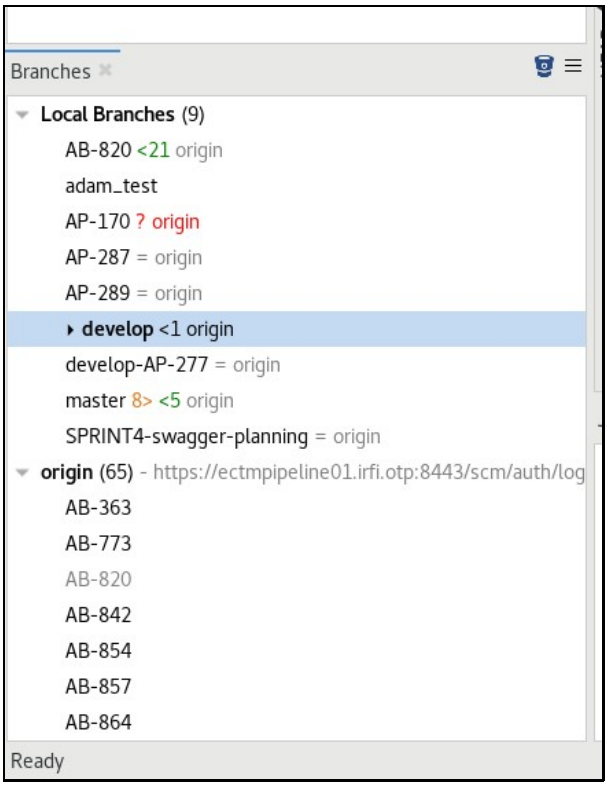
Branch-ek kezelése

Az oldal tején láthatjuk a repository választót.



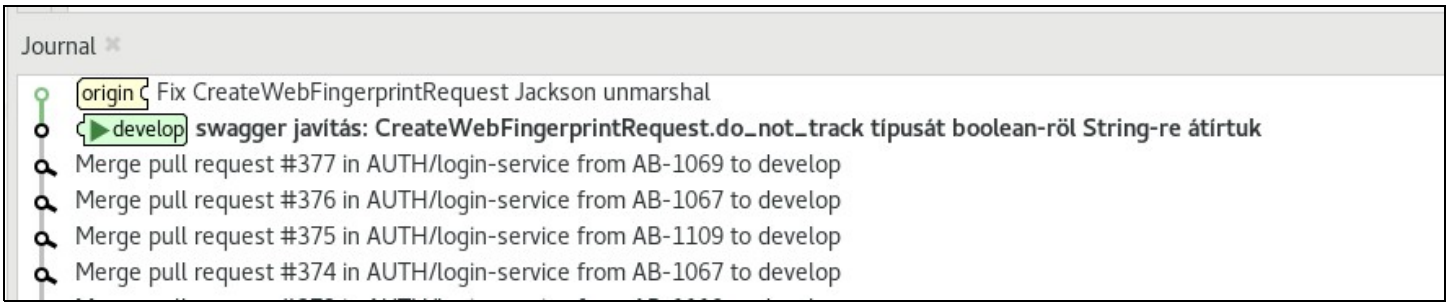
Egyszerre mindig csak egy repository lehet kiválasztva, amin éppen dolgozunk. Nem lehet úgy repository-t váltani, hogy van módosított fájl a munkaterületen.

Az **origin** mutatja a távoli repository-t a **Local Branches** mutatja a helyi branch-eket. Az a lokális ág van jelenleg checkout-olva, amire a kis fekete nyíl mutat. (**>**)



Egyszerre mindig csak egy aktív branch lehet kiválasztva, amin éppen dolgozunk. Nem lehet úgy branch-et váltani, ha van módosított fájl a munkaterületen.

A képerny? alján láthatjuk az aktuálisan checkout-olt branchen hogy hol tart a HEAD pointer a lokális és a távoli repository-ban.



A sárga (origin) mutatja, hogy a távoli repository hol tart, és a zöld mutatja, hogy a lokális változatunkban hol tart a HEAD pointer. A zöld mez?be bele van írva a branch neve is (a példában develop). Ha a sárga és a zöld mez? egybe esik, akkor a lokális repoban ugyan az van mint a központiban. A távoli állapotot a rapidSVN onnan tudja (sárga mez?) hogy nem PULL-ozta csak FETCH-elte a távoli tartalmat, vagyis anélkül hogy letöltötte volna a lokális repoba a változást, csak lekérdezte, hogy mi változott. A rapidSVN periodikusan magától is futtat FATCH-t.

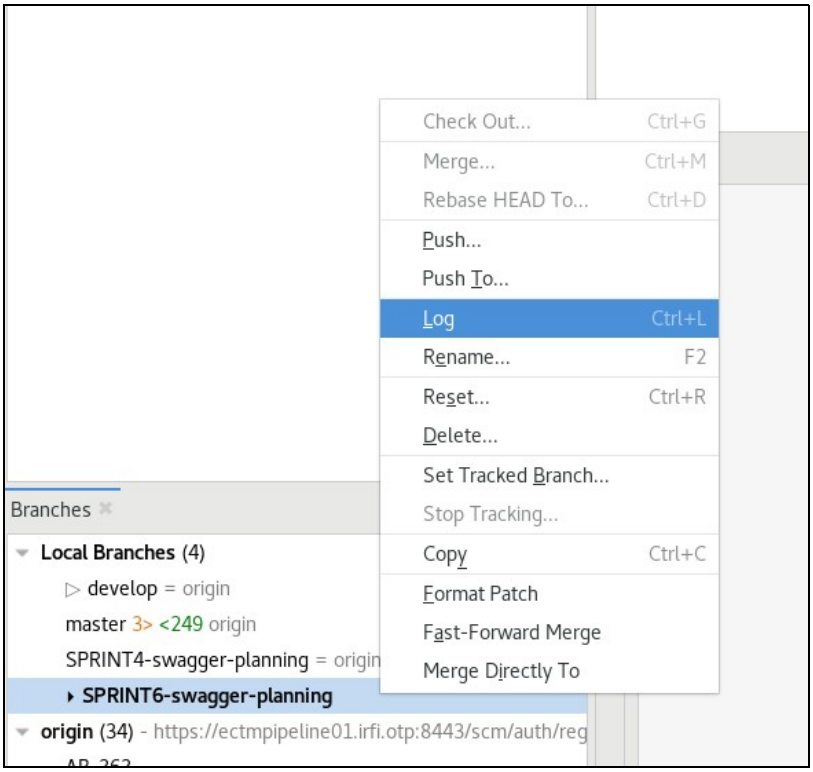
Új branch létrehozása

Lokális branch létrehozása

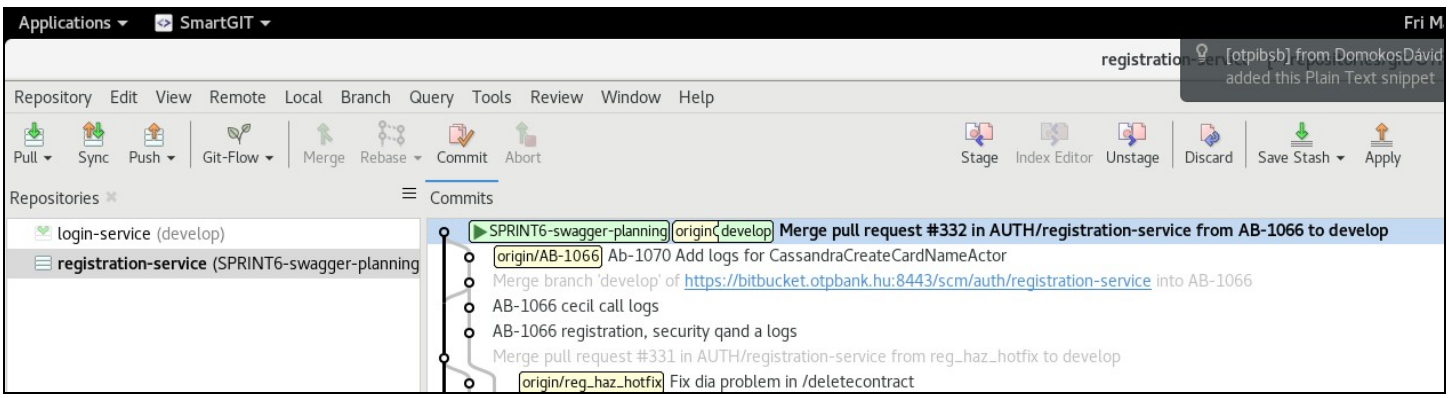
Új branch létrehozásához checkout-oljuk azt a lokális branch-et amib?l le akarunk ágazni, majd a fenti menüb?l válasszuk a Branch -> Add Branch lehet?séget.

Adjuk meg az új branch nevét, és válasszuk az "Add branch and checkout" lehet?séget. Ekkor lokálisan létrejön az új branch-ünk, ami checkout-olva is van. A távoli repóban (origin) ez a branch még nem létezik.

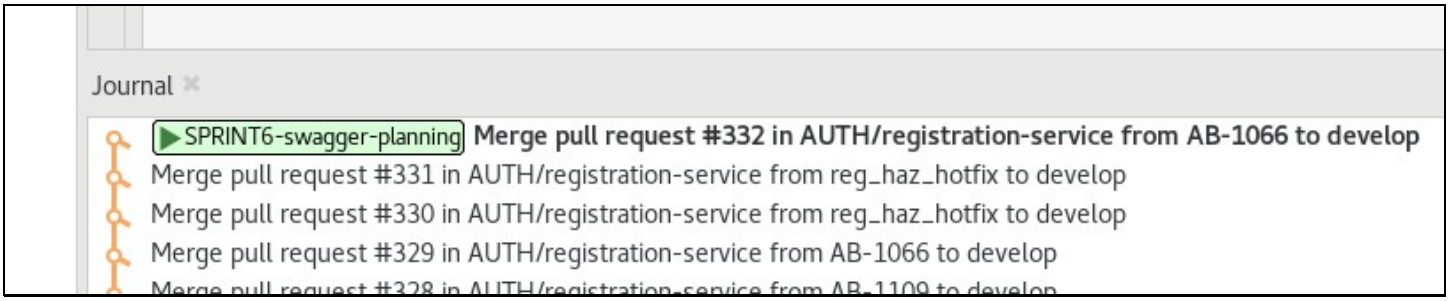
Kattintsunk az új lokális branch-re és nézzük meg a log-ját:



Az új branch-et a develop ágból hoztuk létre. Láthatjuk, hogy a HEAD-je egybe esik a develop ág HEAD-jével. Azoban az új branchnek (Sprint6-swagger-planning) nincs megfelel?je a távoli repóban (sárga origin), mivel ez még csak egy lokális branch, ezt PUSH-olni kell hogy a távoliban is megjelenjen.

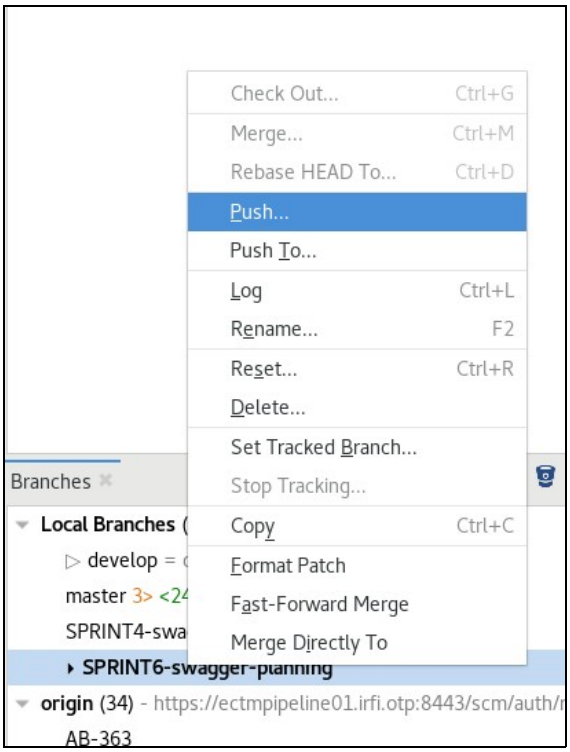


A f? képerny? alján, ahol az aktuálisan checkout-olt branch commit log-ját mutatja, láthatjuk, hogy a branch még csak lokálisan létezik, ezt jelzi a narancssárga vonal.

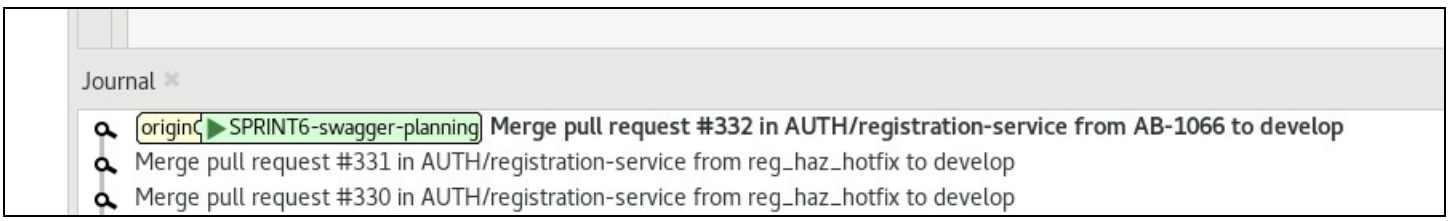


Lokális branch PUSH-olása

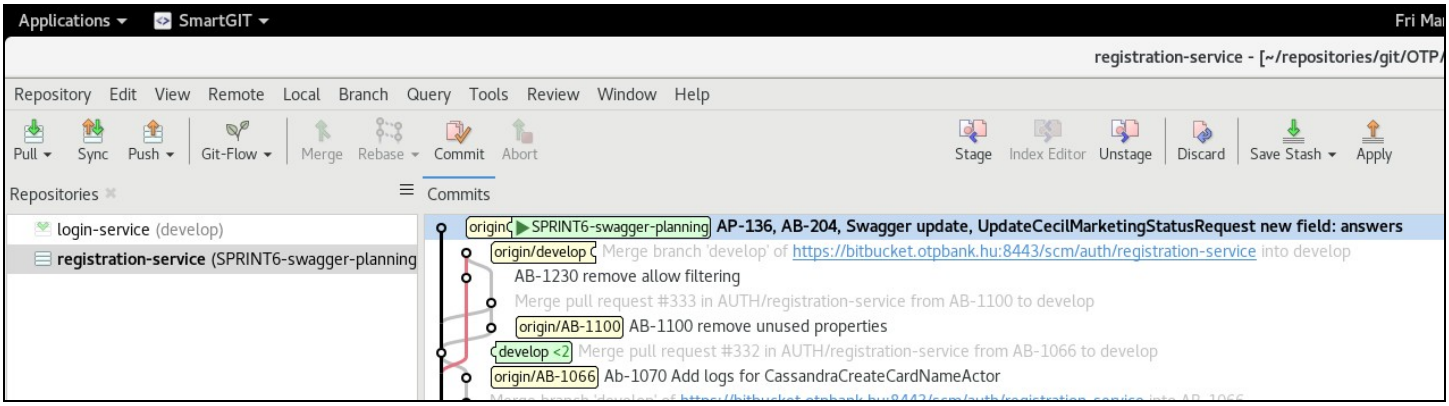
A lokális branch-et PUSH-olni kell a távoli branch-be. Jobb klikk a lokális branch-re, majd PUSH.



Ekkor a f? képerny?n lév? log historyban, ami az aktuálisan checkout-olt branch-et mutatja, a távoli sárga és a lokális zöld címke egy vonalba került, ezzel jelezve, hogy a lokális és a távoli branch már azonos.



Miután már bármit PUSH-oltunk az új branch-be, a log-ban látszik, hogy már külön életet él a develop-hoz képest:



A képen látható hogy a SPRINT6-swagger-planning branchben a lokális és a távoli ugyan azon a HEAD verzión van, mert a sárga és a zöld címke egybe esik. Azt is láthatjuk, hogy a pirossal jelölt develop ágban a lokális repónk le van maradva, mert a zöld lokális HEAD-et már több commit-al megel?zi a sárga távoli HEAD muató.

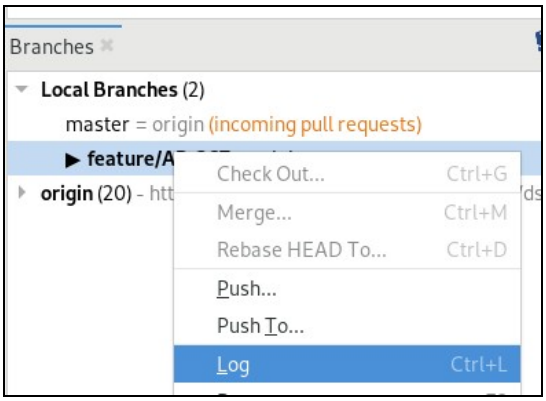
Pull Request-ek kezelése (BitBucket)

- **Incoming pull**: requests are those which other users are requesting to pull from their repositories. They are displayed in a separate category called Pull Requests in the Branches view.
- **Outgoing pull**: requests are those which you have sent to other users/repositories, requesting them to pull your changes. They are display directly below the local (or if it does not exist), the remote branch in the Branches view.

Pull request készítése

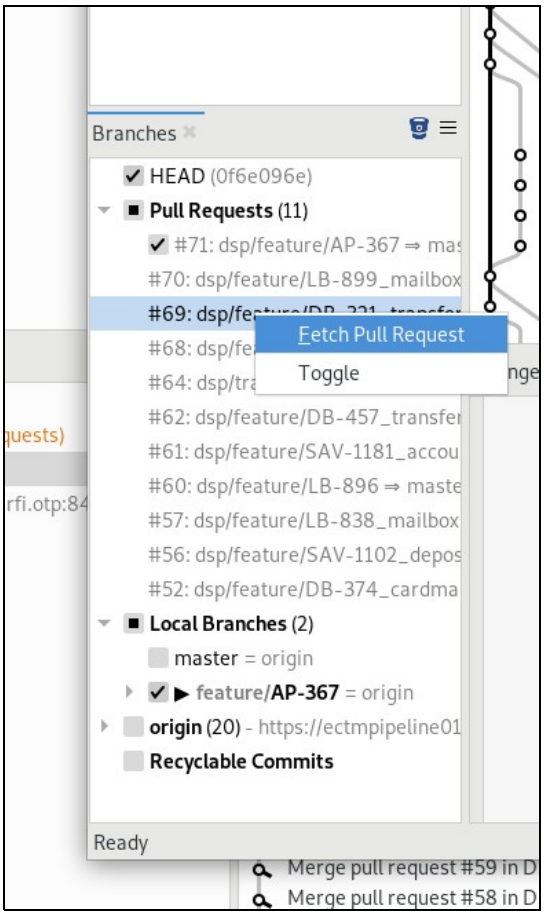
Pull request megtekintése

Checkout-olni kell azt a branch-et (vagy a master-t) aminek a pull request-jeit meg akarjuk nézni.



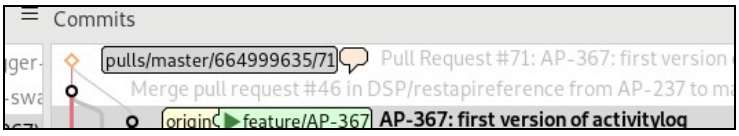
Láthatjuk, hogy a SmartGit jelzi is, hogy a master ágon vannak Pull request-ek.

A megnyíló log ablakban láthatjuk, hogy a PULL request-ek külön vannak gy?jtve. A példában 11 pull request van. Ahhoz hogy meg tudjuk tekinteni a PULL request tartalmát és az arra adott comment-eket, vagy hogy mi is commentelni tudjunk, le kell Fetch-elni kell a pull requestet. Kattintsunk rá a megtekinteni kívánt pull request nevére, majd válasszuk a **Fetch pull request** lehet?séget.



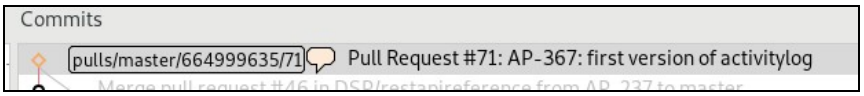
Ekkor a már lehozott pull request el?tt meg fog jelenni egy Checkbox, ezzel jelezve, hogy már hozzá tudjuk adni ill ki tudjuk szedni az egyesített log nézetb?l.

Ha be van pipálva a checkbox a már lehozott (Fetch-elt) pull request sorában, akkor az külön megjelenik az egyesített log fában:

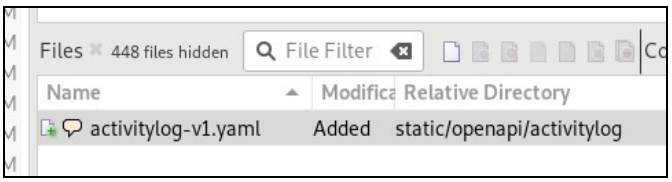


Ha volt hozzá komment is, akkor egy narancssárga kis buborék jelenik meg a neve mellett.

Válasszuk ki az egyesített log fában a pull request sorát:

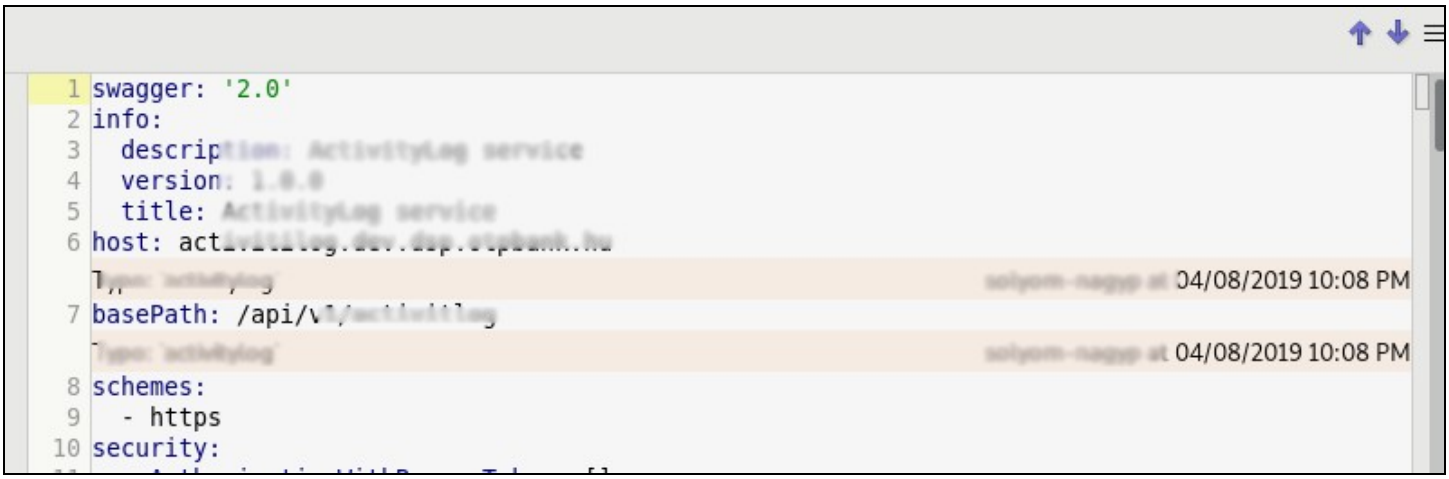


Ekkor a bal oldali fájl listázóban meg fognak jelenni a Pull request-hez tartozó fájlok:

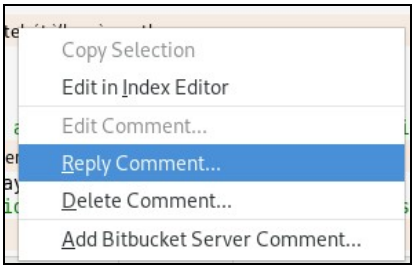


A narancssárga buborék jelzi, hogy az adott fájlhoz van comment.

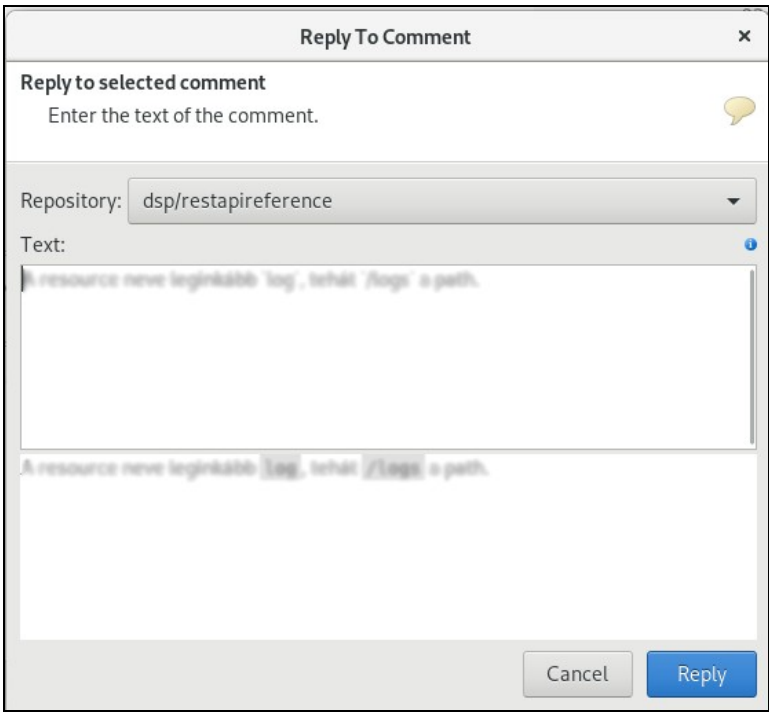
Ha kiválasztunk egy fájlt, akkor az kommentekkel együtt meg fog jelenni a bal alsó ablakban:



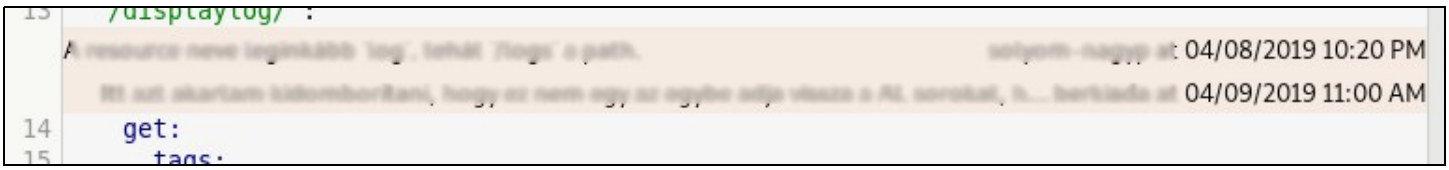
A kommentekre lehet válaszolni, vagy a fájl tetsz?leges pontjára új comment-et beszúrni. Ha válaszolni akarunk, akkor jobb click a kommentre, majd Replay comment:



A megnyíló ablak ki lesz már eleve töltve az eredeti comment tartalmával, amire válaszolni akarunk. Ezt töröljük ki és írjuk be a fels? mez?be a választ:

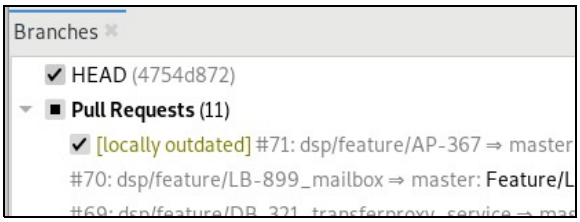


A válasz meg fog jelenni az eredeti comment alatt:

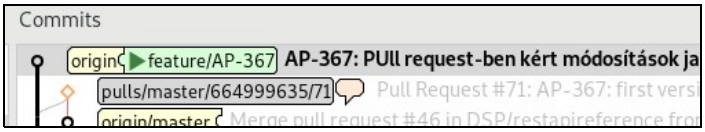


Pull request frissítése

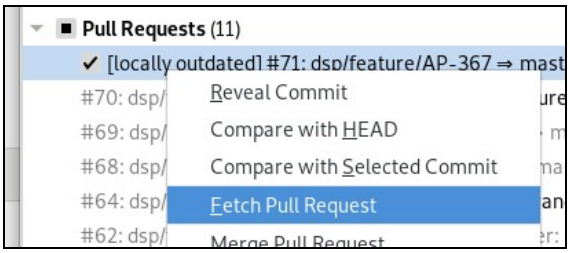
Ha egy commentezett pull request-ünket frissíteni akarunk, akkor nincs más dolgunk, mint hogy ugyan arra a branch-re PUSH-oljuk be ugyan azokat a fájlokat módosítva, amire a PULL request volt kérve. Ekkor a bitBucket észre fogja venni, és frissíteni fogja az eredeti request-et. Ha a PUSH után megnyitjuk a LOG-ját a repository-nak, akkor a pull request felsorolásban, látni fogjuk, hogy a lokális másolata a pull request-nek már elavult.



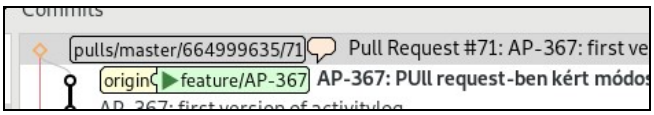
Az egyesített log fában láthatjuk, hogy a branch-ünk már egyel el?rébb jár mint a pull request:



Külön frissíteni kell a pull request adatokat:

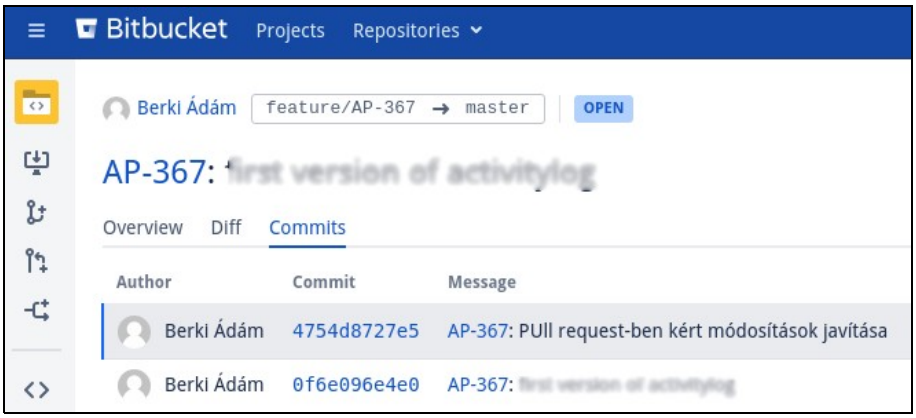


Ekkor láthatjuk az egyesített log fában, hogy megint a pull request kerül a commit-unk elé:



Innent?l kezdve nem lehet megnézni az el?z? kommenteket.

A bitbucket-ben látható, hogy mind a két commit-ot hozzákapcsolta az eredeti pull request-hez:

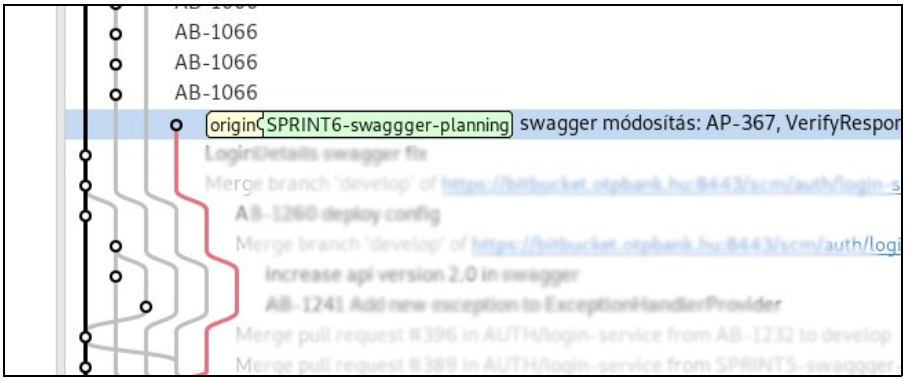


Pull request elfogadása

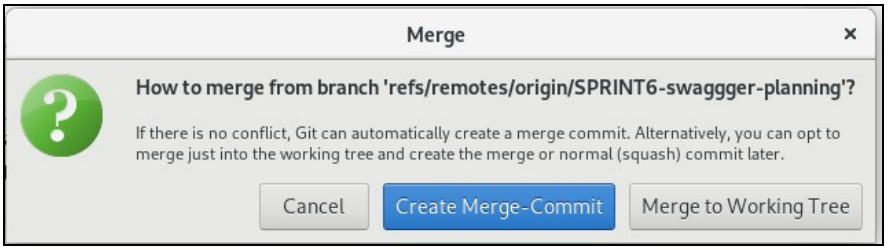
Merge feature branch to master

Ha van egy branch-ünk amit egy másik branch tetejére akarunk rakni (tipikusan egy feature branch-et a develop-ra vagy master-re, akkor els?ként inden lokális változtatást push-olni kell a távoli szerverre, majd bele kell állni (checkout) abba az ágba, ahova merge-ölni akarjuk a feature branch-et, az esetünkben ez a develop branch lesz.

Nyissuk meg a log fát a develop ágon (jobb click -> logs). A bal oldali pipálós listában pipáljuk be azt a branch-et, amit mergelni szeretnénk a developre. Majd az egyesített log fában keressük meg. Mivel a develop-ban állunk, egy még nem merge-ölt feature branch vége a leveg?ben fog lógni:



Kattintsunk rá, hogy legyen kijelölve.
Ez után válasszuk a fels? f? menüben a Merge lehet?séget. Ekkor a felugró ablakban szövegesen is ki lesz írva, hogy mit hova akarunk merge-ölni.



Két lehet?ségünk van:

- Azonnal készítünk az egyesített állapotból egy úgynevezett merge-commit-ot. Ezt akkor tehetjük meg ha nincs conflict.
- Csak a munkaterületre merge-öli rá a smartGit a feature branch-et, és majd mi végezzük el a commit-ot miután feloldottuk a konfliktust.

Általában az els? lehet?ség megfelel?. Ez után láthatjuk majd, hogy a f? képerny?n a branch history-ban, hogy a lokális reponk megel?zi a távoli head-et.

Azt is láthatjuk, hogy automatikusan adott commit üzenetet a smartGit: "Merge remote-tracking branch 'origin/XXXX' into develop".

Most ha nyomunk egy PUSH-t, akkor fel is fogja küldeni az origin-ba az új merge-commit-ot.