JOT2012 Demo2-ohje

Ville Isomöttönen, Jonne Itkonen

19. syyskuuta 2012

Sisältö

1 Esitiedot

Demo 2:ssä oletetaan, että olet osallistunut Demo1:een ja harjoittelut siellä käsittelyt asiat. Demo 2:ssa harjoittelemme tehtävien hallintaa komentoriviltä käyttäen Bugs Everywhere -ohjelmaa.

2 Bugs Everywhere

Bugs Everywhere (BE) on yksinkertainen ohjelma, jota voi käyttää yhtälailla tehtävien ja bugien hallintaan. Se täydentää hajautettua versionhallintaa. Tällä kurssilla käytämme sitä Git-versionhallinnan yhteydessä (toki sitä voisi käyttää muidenkin versionhallintaohjelmistojen kanssa: Arch, Bazaar, Darcs, Mercurial, Monotone).

BE:tä voi käyttää yksinkertaisesti komentoriviltä. Tehtäviä (tai bugeja) voidaan luoda, niille voidaan kiinnittää tekijä, ja niiden tilaa voidaan muuttaa ja seurata.

Ota etäyhteys jalava-koneeseen ja siirry siellä demo 1:ssä luomaasi hakemistoon, jossa lokaali git-varastosi sijaitsee.

3 Harjoittelua

Tee seuraava harjoitus jälleen parisi kanssa. Alla käytetään parista nimiä Alice ja Bob.

1. Alice luo BE-varaston antamalla komennon `be init`. Tämä luo hakemistoosi varaston (.be-hakemiston) tehtävien hallintaa varten, minkä voit todeta komennolla `ls -la`. Listan BE:n tarjoamista työkaluista saat näkyville komennossa `be help`. Saat tarkemmin apua yksittäisestä be-komennosta komennolla `be help KOMENTO`.

Alice luo nyt uuden tehtävän kommennolla `be new KUVAUS`. `KUVAUS` on lyhyt kuvaus tehtävästä. Jos kuvaus koostuu useammasta kuin yhdestä sanasta, on se annettava lainausmerkeillä " tai hipsuilla ' ympäröitynä. Esimerkiksi

```
$ be new "Uusi tehtävä"
Created bug with ID 81a/821
$ _
```

Huomaa, että kuvaus ei yksilöi tehtävää, vaan BE luo jokaiselle tehtävälle yksilöivän tunnisteen (*id*). Yllä tämä tunniste on `81a/821`. Tätä tunnistetta käytetään viitattaessa tehtävään.

Alice haluaa myös määrittää tehtävän vaativuuden. Hän antaa komennon

```
be severity serious /821
```

(huomaa, kuinka tehtävään voi viitata lyhyesti id:n jälkiosalla). Komennolla

```
be list --severity=serious
```

löytyvät nyt kaikki hankalat tehtävät.

Jotta Alicen luoma tehtävä tulee muidenkin projektin osallisten nähtäville, hän sanoo, kuten git-versionhallinnan yhteydessä, `git commit`. Tällöin tieto luodusta tehtävästä on luodussa be-varastossa, joka on tässä tapauksessa git-versiohallinnan alaisuudessa.

Vielä komento `git push origin`, kuten viime demoissa opimme, ja lokaalin git-varaston tiedot siirtyvät etäkoneelle, jolloin myös tieto luodusta tehtävästä siirtyy etäkoneelle.

2. Bob, joka tässä tapauksessa on vastuussa tehtävien jakamisesta, haluaa ensin päivittää tietoonsa ilmoitetut tehtävät. Tällöin Bobin tarvitsee noutaa etäkoneelta nykyiset versiotiedot omalle koneelleen. Bob toimii kuten edellisissä demoissa ja antaa komennon `git fetch -v origin`, ja tämän perään `git merge origin/master`, jolloin etäkoneelta tullut projektin tila ja Bobin omalla koneella näkyvä tila saadaan vastaamaan toisiaan. Samalla siirtyvät etäkoneella tiedossa olevat tehtävät Bobille. Nyt myös Bobilla on lokaali BE-varasto, josta löytyvät Alicen lisäämät tehtävät.

3. Bob voi nyt listata tehtävät komennolla `be list` ja tarkastella niitä tarkemmin komennolla `be show ID`, missä ID viittaa halutun tehtävän tunnisteeseen, kuten esimerkissämme edellä `81a/821`.

   Bob päättää ottaa tehtävän `81a/821` hoitaakseen, joten hän suorittaa komennon:

   ```
   be assign "Bob Hacker" 812/821
   ```

   Huomaa, että kiinnitettäessä tehtäviä henkilöille on yksinkertaisinta käyttää tunnisteena henkilöiden oikeaa nimeä.

   Bob päivittää tehtävälistan taas etäkoneelle komennoilla:

   ```
   git commit
   git push origin master
   ```

Tylsän helppoa, eikö?

Myös `be commit` toimii, mutta sillä voi olla ongelmia selvitä ääkkösistä ja muista ASCII-merkistön ulkopuolisista merkeistä, joten suosittelemme toistaiseksi `git commit` -komennon käyttöä tehtäviä versioitaessa.

4. Kun Bob on saanut tehtävän tehtyä, hän muuttaa sen tilan tehdyksi komennolla

   ```
   be status fixed /821
   ```

   Jälleen `git commit` ja `git push origin`, jotta muuttuneet tiedot saatetaan projektin muiden jäsenten tietoon.

5. Alice hakee nyt projektin tiedot etäkoneelta tutuilla git-komennoilla. (Mitkäs ne olivatkaan?) Tämän jälkeen hän asettaa tehtävän tilan suljetuksi (*closed*), ja tallentaa muutokset lokaaliin tehtävävarastoon ja etäkoneelle. (Mitenkäs tämä tapahtuikaan?)

6. Anna komento `be help` ja tutustu niin moneen komentoon kuin ehdit.

7. Laita tehtävät harjoitustyön vaiheille oman harjoitustyönne git-varastoon. Harjoitustyötä varten voitte kloonata itsellenne varaston ykkösdemojen ohjeiden mukaisesti, mutta käyttäen harjoitustyölle varattua YouSource-projektia https://yousource.it.jyu.fi/jotharkat2012/harkka.

   Kun nyt lisäät tehtäviä BEhen, lisää niiden tavoitteeksi (*target*) `Vaihe1`. Nämä tehtävät siis tulee suorittaa (*status=fixed*), jotta BE tulkitsee tavoitteen `Vaihe1` suoritetuksi. (Aloita esimerkiksi kirjoittamalla `be help target`. Tavoitteet esitetään riippuvuuksina, joten lue myös `be help depend`.)

4 Vinkkejä

Jos vim tuntuu menevän jumiin, varsinkin tallennusvaiheessa, olet todennäköisesti antanut näppäinkomennon `C-s`, eli `Control+s`. Tämä jäädyttää unix-yhteyden. Yhteyden saa taas toimintaan näppäinkomennolla `C-q`.