DOMANDE CONCORSO PUBBLICO PER TITOLI ED ESAMI PER UN POSTO PER IL PROFILO PROFESSIONALE DI COLLABORATORE TECNICO DI VI LIVELLO PROFESSIONALE CON CONTRATTO DI LAVORO A TEMPO INDETERMINATO PER ATTIVITA' DI GESTIONE DEL SISTEMA AUDIO-VIDEO E DI VIDEO-REGISTRAZIONE DEGLI EVENTI, REGIA E MONTAGGIO DI RIPRESE, PUBBLICAZIONE E ARCHIVIAZIONE DEI CONTENUTI, GESTIONE DELLE PAGINE WEB DEL GGI E SUPPORTO TECNICO DELLA PARTE HARDWARE E SOFTWARE DI PC E STAMPANTI.

BANDO N. TI/GGI/C6/26031 – PROVA ORALE

BUSTA N. 1

1 - Il candidato descriva come gestire la produzione di video post evento con particolare attenzione alla fruibilità da parte di soggetti diversamente abili.

2 - Il candidato descriva a sua scelta un software per elaborazione di immagini digitali.

3 - Lettura e traduzione di un breve testo in lingua inglese

she's done two commits on the topic branch she just pushed, she can run this:

```
$ git request-pull origin/master myfork
The following changes since commit 1edee6b1d61823a2de3b09c160d7080b8d1b3a40:
Jessica Smith (1):
        Create new function

are available in the git repository at:

  https://githost/simplegit.git featureA

Jessica Smith (2):
      Add limit to log function
      Increase log output to 30 from 25

 lib/simplegit.rb |   10 ++++++++++-
 1 files changed, 9 insertions(+), 1 deletions(-)
```

This output can be sent to the maintainer—it tells them where the work was branched from, summarizes the commits, and identifies from where the new work is to be pulled.

On a project for which you're not the maintainer, it's generally easier to have a branch like `master` always track `origin/master` and to do your work in topic branches that you can easily discard if they're rejected. Having work themes isolated into topic branches also makes it easier for you to rebase your work if the tip of the main repository has moved in the meantime and your commits no longer apply cleanly. For example, if you want to submit a second topic of work to the project, don't continue working on the topic branch you just pushed up—start over from the main repository's `master` branch:

```
$ git checkout -b featureB origin/master
  ... work ...
$ git commit
$ git push myfork featureB
$ git request-pull origin/master myfork
  ... email generated request pull to maintainer ...
$ git fetch origin
```

Now, each of your topics is contained within a silo—similar to a patch queue—that you can rewrite, rebase, and modify without the topics interfering or interdepending on each other, like so:

DOMANDE CONCORSO PUBBLICO PER TITOLI ED ESAMI PER UN POSTO PER IL PROFILO PROFESSIONALE DI COLLABORATORE TECNICO DI VI LIVELLO PROFESSIONALE CON CONTRATTO DI LAVORO A TEMPO INDETERMINATO PER ATTIVITA' DI GESTIONE DEL SISTEMA AUDIO-VIDEO E DI VIDEO-REGISTRAZIONE DEGLI EVENTI, REGIA E MONTAGGIO DI RIPRESE, PUBBLICAZIONE E ARCHIVIAZIONE DEI CONTENUTI, GESTIONE DELLE PAGINE WEB DEL GGI E SUPPORTO TECNICO DELLA PARTE HARDWARE E SOFTWARE DI PC E STAMPANTI.

BANDO N. TI/GGI/C6/26031 – PROVA ORALE

BUSTA N. 2

1 - Il candidato descriva come organizzare in termini di hardware e software una sala multimediale adatta allo streaming.

2 - Il candidato descriva a sua scelta un software per elaborazione di video digitali.

3 - Lettura e traduzione di un breve testo in lingua inglese

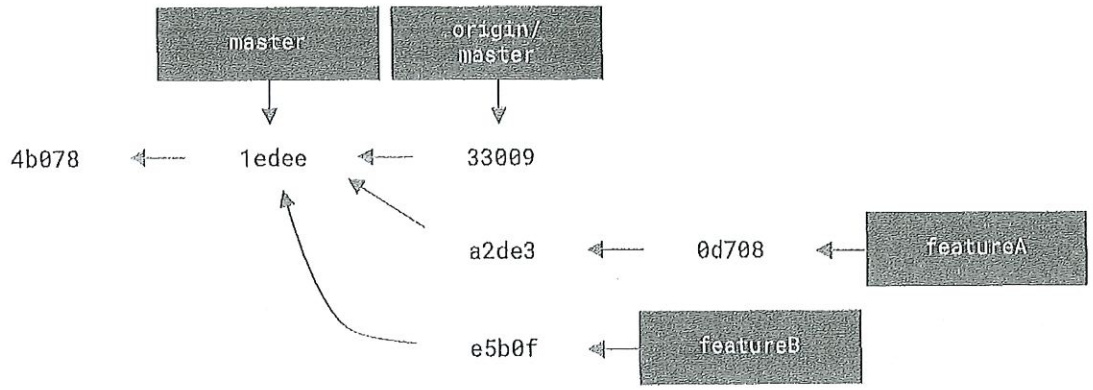

Figure 69. Initial commit history with `featureB` *work*

Let's say the project maintainer has pulled in a bunch of other patches and tried your first branch, but it no longer cleanly merges. In this case, you can try to rebase that branch on top of `origin/master`, resolve the conflicts for the maintainer, and then resubmit your changes:

```
$ git checkout featureA
$ git rebase origin/master
$ git push -f myfork featureA
```

This rewrites your history to now look like Commit history after `featureA` work.

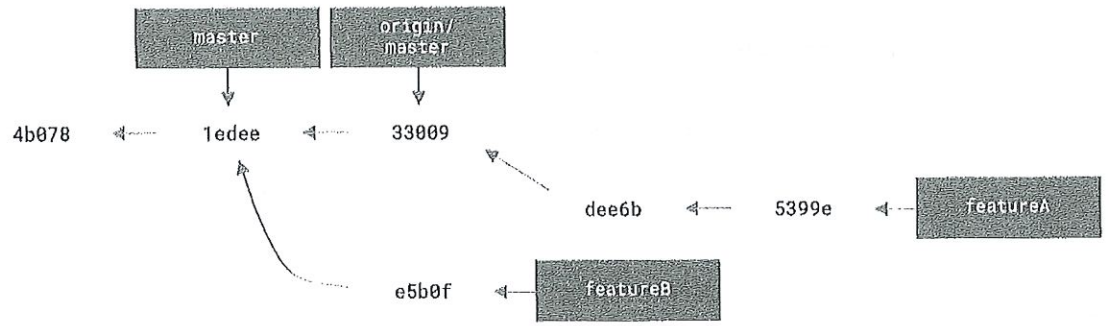

Figure 70. Commit history after `featureA` *work*

Because you rebased the branch, you have to specify the `-f` to your push command in order to be able to replace the `featureA` branch on the server with a commit that isn't a descendant of it. An alternative would be to push this new work to a different branch on the server (perhaps called `featureAv2`).

Let's look at one more possible scenario: the maintainer has looked at work in your second branch and likes the concept but would like you to change an implementation detail. You'll also take this opportunity to move the work to be based off the project's current `master` branch. You start a new branch based off the current `origin/master` branch, squash the `featureB` changes there, resolve any conflicts, make the implementation change, and then push that as a new branch:

DOMANDE CONCORSO PUBBLICO PER TITOLI ED ESAMI PER UN POSTO PER IL PROFILO PROFESSIONALE DI COLLABORATORE TECNICO DI VI LIVELLO PROFESSIONALE CON CONTRATTO DI LAVORO A TEMPO INDETERMINATO PER ATTIVITA' DI GESTIONE DEL SISTEMA AUDIO-VIDEO E DI VIDEO-REGISTRAZIONE DEGLI EVENTI, REGIA E MONTAGGIO DI RIPRESE, PUBBLICAZIONE E ARCHIVIAZIONE DEI CONTENUTI, GESTIONE DELLE PAGINE WEB DEL GGI E SUPPORTO TECNICO DELLA PARTE HARDWARE E SOFTWARE DI PC E STAMPANTI.

BANDO N. TI/GGI/C6/26031 – PROVA ORALE

BUSTA N. 3

1 - Il candidato descriva come gestire un evento misto (in presenza e online) permettendo la possibilita' di domande e commenti da remoto.

2 - Il candidato descriva a sua scelta un software per elaborazione di presentazioni (slide) digitali.

3 - Lettura e traduzione di un breve testo in lingua inglese

```
$ git checkout -b featureBv2 origin/master
$ git merge --squash featureB
  ... change implementation ...
$ git commit
$ git push myfork featureBv2
```

The `--squash` option takes all the work on the merged branch and squashes it into one changeset producing the repository state as if a real merge happened, without actually making a merge commit. This means your future commit will have one parent only and allows you to introduce all the changes from another branch and then make more changes before recording the new commit. Also the `--no-commit` option can be useful to delay the merge commit in case of the default merge process.

At this point, you can notify the maintainer that you've made the requested changes, and that they can find those changes in your `featureBv2` branch.

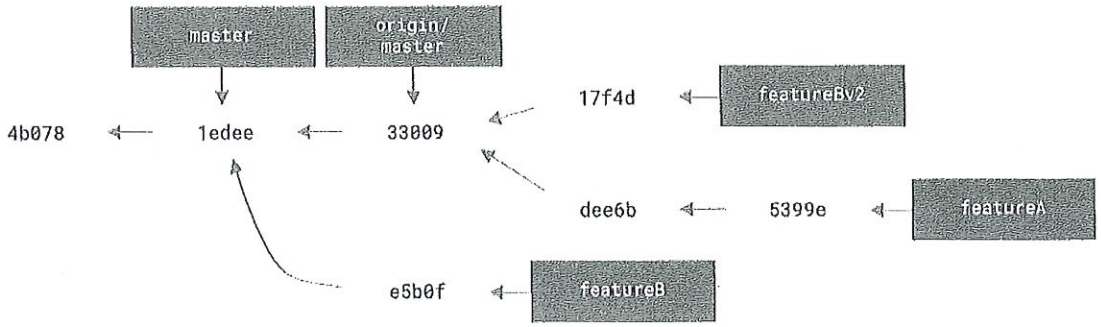

Figure 71. Commit history after `featureBv2` *work*

Public Project over Email

Many projects have established procedures for accepting patches—you'll need to check the specific rules for each project, because they will differ. Since there are several older, larger projects which accept patches via a developer mailing list, we'll go over an example of that now.

The workflow is similar to the previous use case—you create topic branches for each patch series you work on. The difference is how you submit them to the project. Instead of forking the project and pushing to your own writable version, you generate email versions of each commit series and email them to the developer mailing list:

```
$ git checkout -b topicA
  ... work ...
$ git commit
  ... work ...
$ git commit
```

Now you have two commits that you want to send to the mailing list. You use `git format-patch` to generate the mbox-formatted files that you can email to the list—it turns each commit into an