

```
$ git bundle verify ../commits.bundle
The bundle contains 1 ref
71b84daaf49abed142a373b6e5c59a22dc6560dc refs/heads/master
The bundle requires these 1 ref
9a466c572fe88b195efd356c3f2bbeccdb504102 second commit
../commits.bundle is okay
```

If the bundler had created a bundle of just the last two commits they had done, rather than all three, the original repository would not be able to import it, since it is missing requisite history. The `verify` command would have looked like this instead:

```
$ git bundle verify ../commits-bad.bundle
error: Repository lacks these prerequisite commits:
error: 7011d3d8fc200abe0ad561c011c3852a4b7bbe95 Third commit - second repo
```

However, our first bundle is valid, so we can fetch in commits from it. If you want to see what branches are in the bundle that can be imported, there is also a command to just list the heads:

```
$ git bundle list-heads ../commits.bundle
71b84daaf49abed142a373b6e5c59a22dc6560dc refs/heads/master
```

The `verify` sub-command will tell you the heads as well. The point is to see what can be pulled in, so you can use the `fetch` or `pull` commands to import commits from this bundle. Here we'll fetch the `master` branch of the bundle to a branch named `other-master` in our repository:

```
$ git fetch ../commits.bundle master:other-master
From ../commits.bundle
* [new branch]      master      -> other-master
```

Now we can see that we have the imported commits on the `other-master` branch as well as any commits we've done in the meantime in our own `master` branch.

```
$ git log --oneline --decorate --graph --all
* 8255d41 (HEAD, master) Third commit - first repo
| * 71b84da (other-master) Last commit - second repo
| * c99cf5b Fourth commit - second repo
| * 7011d3d Third commit - second repo
|/
* 9a466c5 Second commit
* b1ec324 First commit
```

So, `git bundle` can be really useful for sharing or doing network-type operations when you don't have the proper network or shared repository to do so.