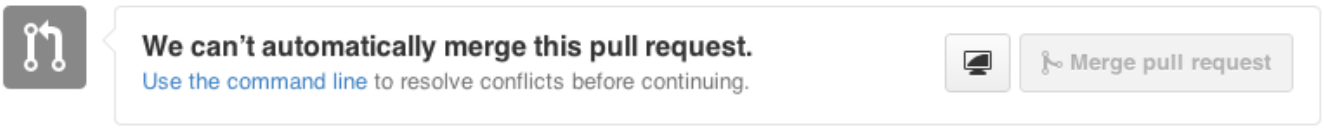
Resolving Merge Conflicts

Case: Adding a new feature to the source

So you followed the workflow described above, create a fork and maybe even a new branch to avoid collisions.

Now you go to make a pull request and it says "Cannot automatically merge"



Gimme a break! We went through all of that and there's still going to be merge conflicts? Welcome to source control, my friend.

Not to worry! These things can usually be easily resolved if you are following the feature-branching advice given above.

(You are creating separate branches for each of your features, aren't you? This minimizes the size of potential conflicts by isolating each set of related changes. See below.

Pull from Upstream

On your local copy, pull from upstream to grab any changes to the upstream repo:

```
$ git pull upstream master
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 4 (delta 3), reused 4 (delta 3), pack-reused 0
Unpacking objects: 100% (4/4), done.
From https://github.com/nds-org/ndslabs-specs
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> upstream/master
Auto-merging clowder/clowder.json
CONFLICT (content): Merge conflict in clowder/clowder.json
Automatic merge failed; fix conflicts and then commit the result.
```

Notice the line starting with **CONFLICT**, which indicates that you will manually need to fix this file.

Locate Offending Conflicts

Open the file in your favorite editor and you should see the conflict(s) surrounded by > and <:

```
.....
        {
            "name": "RABBITMQ_EXCHANGE",
            "value": "clowder",
            "canOverride": false
        },
        {
<<<<<<< HEAD
            "name": "TOOLMANAGER_URI",
            "value": "localhost:8082",
=======
            "name": "TOOLMGR_URI",
            "value": "http://localhost:8082",
>>>>>>> 807f37cbf505e06469a7b54429be5e3a60dd26d3
            "label": "ToolServer address",
            "canOverride": true
        }
    ],
    "ports": [
        {
.....
```

<<<<<<< HEAD indicates the HEAD of your current local working copy ("your" modifications)

>>>>>>> COMMIT SHA HASH indicates the SHA hash of the conflicting commit ("their" modifications)

Resolve Any Conflicts

Edit each file appropriately to resolve the merge conflict(s).

Simply remove any excess bits and retain only the file that you wish to commit:

```
.....
        {
            "name": "RABBITMQ_EXCHANGE",
            "value": "clowder",
            "canOverride": false
        },
        {
            "name": "TOOLMANAGER_URI",
            "value": "http://localhost:8082",
            "label": "ToolServer address",
            "canOverride": true
        }
    ],
    "ports": [
        {
.....
```

Tell Git that you have Resolved the Conflicts

Now re-add any conflicting files to git's index.

This will mark the conflict as resolved, and stage the files for commit.

Now you should be able to commit to your local copy, then push to your personal fork:

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
You have unmerged paths.
  (fix conflicts and run "git commit")
Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   clowder/clowder.json
no changes added to commit (use "git add" and/or "git commit -a")


$ git add clowder/clowder.json

$ git commit -a -m "Fixed merge conflict"
[master e09f5a2] Fixed merge conflict
 1 file changed, 1 insertion(+), 1 deletion(-)


$ git push origin master
Username for 'https://github.com': your-git-username
Password for 'https://your-git-username@github.com':
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 366 bytes | 0 bytes/s, done.
Total 4 (delta 3), reused 0 (delta 0)
To https://github.com/your-git-username/ndslabs-specs.git
   c241a36..e09f5a2  master -> master
```

Verify that Conflict is Resolved

You should now see that your outstanding Pull Request, if you made one, has been updated to include your newly pushed commit.

Hopefully the indicator changed from gray (conflict) to green (mergeable), and your conflict has been resolved:

NOTE: You may need to perform these steps multiple times to work out all merge conflicts.

With a fully resolved branch, pulling should yield the following messages:

```
$ git pull upstream master
From https://github.com/nds-org/ndslabs-specs
 * branch            master     -> FETCH_HEAD
Already up-to-date.
```