Stop cherry-picking, start merging, Part 10: Web-based workflow for Azure DevOps (formerly VSTS)

devblogs.microsoft.com/oldnewthing/20180323-00

March 23, 2018

Raymond Chen

If you use Visual Studio Team Services (VSTS) Azure DevOps as your repository hosting service, then a lot of the workflows described in this series can be performed completely from the Web interface.

Find the merge base

This is the one step that doesn't have a convenient button to click in Azure DevOps.

Option 1: Use special knowledge.

If you have special knowledge of the way your team manages branches, you may be able to figure out the merge base by using your powerful brain. In the Windows division, we have internal Web sites that let you look up various information from the repo, and you may be able to use that to find a suitable merge base. For example, build numbers are established by the master branch, so a safe (though perhaps not optimal) merge base is the commit on the master branch whose build number is the minimum of the build numbers of the two branches you are working with.

Option 2: Ask Azure DevOps to look it up.

This is very annoying to do the first time, and moderately annoying to do subsequent times.

One-time setup: Take the repo clone URL and append `/vsts/info` to it. Visit that Web site, and you will get some JSON back. Look in the `repository` section of the JSON and suck out the `url` property.

```
{
  "serverUrl": "https://dev.azure.com/fabrikam-fiber-inc",
  "collection": {
    ...
  },
  "repository": {
    "id": "2f3d611a-f012-4b39-b157-8db63f380226",
    "name": "FabrikamCloud",
    "url": "https://dev.azure.com/fabrikam-fiber-inc/_apis/
            git/repositories/2f3d611a-f012-4b39-b157-8db63f380226",
    ...
    "project": {
      ...
    },
    ...
  }
}
```

This URL will not change, so you need to look this up only once. (I inserted line breaks for readability. It will be one long line.)

Next, if one or more of the things from which you want to get the merge base is a branch, you need to convert it to a commit ID. Go to your repo on Azure DevOps, find the branch on the *Branches* tab, go to the *Commit* column and hover over the commit ID. A little document icon will appear: Click it to copy the hash to the clipboard.

Finally, ask Azure DevOps to calculate the merge base between the two commits.

```
https://dev.azure.com/fabrikam-fiber-inc/_apis/git/
    repositories/2f3d611a-f012-4b39-b157-8db63f380226/
    commits/fe17a84cc2dfe0ea3a2202ab4dbac0706058e41f/mergebases?
    otherCommitId=0360c963d7d86d040e9c33bba836feab14da4ad3&
    api-version=4.1-preview
```

The first part of the URL is the service URL we obtained during the one-time setup. Substitute the two commit IDs (in boldface). The merge-base operation is symmetric, so it doesn't matter what order you list them in. (Note that this is a preview API, so eventually I'll have to update the URL to the release version.)

The result will be some more JSON:

```
{
    "count": 1,
    "value": [{
        "commitId": "be67f8871a4d2c75f13a51c1d3c30ac0d74d4ef4"
    }]
}
```

There is your merge base.

That was a real pain. Maybe I can convince our engineering team to create a tiny little Web page that lets people type in two commits or branch names, and it can do the grunt work of finding the merge base.

Create the patch branch

Go to your repo on Azure DevOps and enter the commit ID for the merge base into the *Commits* page. You are now looking at the commit. Go to the "…" menu (next to the *Search in branches*) and click *New branch*. Create your branch.

Prepare the patch branch

Apply the changes to the patch branch. For example, if you need to edit a file, you can go to a file and click *Edit*. If you need to cherry-pick, you can go to the commit you want to cherry-pick and select *Cherry-pick* from the "…" menu. Note that Azure DevOps will not cherry-pick directly into a branch; it will create a temporary branch and a pull request from the temporary branch into the target branch. You can make the temporary branch your new patch branch to avoid an extraneous commit.

Now that you have the patch branch ready to be merged into the two targets, you can create the pull requests.

Viewing the proposed result of a pull request

Some parts of the workflow include the step "Verify that the merge results in no code change." To do this in Azure DevOps, you can go to the "…" menu and select *View merge commit*. This shows the changes that will be introduced into the target branch by the pull request. (The commit will not have the final commit message, but that's okay because we're interested only in the files.) If there are no changes, then Azure DevOps will show an empty diff.

Building the proposed result of a pull request

You may want to build the proposed result of the pull request in order to test the result before committing it. Get the commit hash for the proposed merge commit, say by copying it from the web page by clicking the "copy" icon next to the hash. Let's say that the commit hash is `xyz`.

Fetch the commit and then check it out.

```
git fetch origin xyz
git checkout xyz
```

If you are using VFS for Git, then you can skip the fetch step because VFS for Git downloads git objects on demand.

Note that you now have a detached head. If you make any commits (say, to fix a problem), your commit won't go into any named branch. You'll have to cherry-pick them back into the patch branch.

Ensuring that the result is a merge

On the pull request page, when you are about to complete the pull request, make sure to uncheck the *squash* button.

Fortunately, Azure DevOps lets you preload the check boxes on the completion dialog, so click the Complete button, uncheck the *squash* box, and then cancel the completion. Now you won't forget to uncheck the box when you finally decide to complete the pull request.

That's the end of the series for now. I hope you now understand the merge-recursive algorithm and three-way merge algorithm enough to apply these principles to your own scenarios where you may be tempted to cherry-pick changes between branches that will eventually merge.

Raymond Chen

Follow