Preparing, Building and Deploying EVSS Components for Dev Test in INT

GitHub and Jira Version – August 3 2021

# Overview

This SOP describes the process to prepare for a new EVSS Sustainment Release. This includes adding one or more components that will be updated and deploying that component into the VA INT(1) environment for testing.

The following steps are included in this process:

1. Create Jira Work Item
2. Prepare Release Definitions
3. Build Release Definitions
4. Add Component(s) to Release
5. Review and Approve Changes in GitHub
6. Commit Code to GitHub (*for Developers, see companion “EVSS Developer GitHub Flow SOP”*)
7. Building Components
8. Deploying Components

In order to accomplish these goals, the following prerequisites are required:

* GitHub EC and GitHub.com access and basic familiarity
* Basic familiarity with GitHub Pull Requests usage
* Authenticated access to the CI Jenkins server and basic familiarity with Jenkins
  + See <https://jenkins.io/doc/book/getting-started/> if needed
* A username and password for each WebLogic Admin server console for the environment and tier to which artifacts will be deployed (i.e. “WebLogic” and password.)
* The WebLogic console MUST NOT be locked for configuration changes (at deployment time.)

# Create Jira Work Item

The first step is to create a Jira work item which will be used by commits related to preparing the release.

The following process should be used:

1. Create a Jira work item which will be used for version changes in POM files.
   1. Set the Summary to: “Prepare Development Projects for EVSS R<X.Y> Sprint/Release” where <X.Y> is the Release number (for example, at this time the current Sustainment Release is “17.18”.)
   2. Set the work item type to: Task
   3. Set the Component field to: Development
   4. Set the Environment to “CM, INT1”
   5. Set the due date to the Tuesday when Development ends
   6. Set the Release Notes to include: “Description: Prepare components in R<X.Y> for development/test builds” where <X.Y> is the Release number
   7. Set the label to: CM
   8. Set the Description to the same value as the Summary
   9. Set the Story Points to: 8
   10. Set the Estimated time to: 1d
2. Save the draft work item, then change the Status to Start Working and save again.
   1. Note the Jira work item number to be used later (below.)

**NOTE**: You can also clone the work item from the previous release as a starting point / short-cut if desired.

# Prepare Release Definitions

The Release Definition component includes the master “index” of every EVSS component and its version in each release. Management of the contents in this file are generally automated through scripts run in Jenkins (although hand editing is also possible.)

**NOTE**: There are two release definition packages for each release: the ‘main’ package used by most services and webparts components and the true baseline for the release is names ‘R17.X.Y’. The second one used by a few webparts components, and the Portal tier is named ‘R17.X.WLS.Y’. The components using this are behind on some TRM packages and must use older versions of several components for compatibility (but this list should decrease over time.)

At the start of development for each release, new branches both release-definitions components must be created. To start, these will be identical to the previous except having a new version number.

The following procedure should be used to prepare release-definitions for a new version:

1. Login to the EVSS CI environment Jenkins console at the URL:  
   <http://vaausessapp88.aac.va.gov:8001/jenkins/>
2. Navigate to the “998-Stream Management” tab.
3. Select the “Sustainment\_Prepare\_Release\_Definition” project.
4. Select “Build with Parameters” from the menu on the left-hand side of the screen.
5. Enter the number for the work item associated with preparing the release in the WORK\_ITEM input field (created above.)
6. Enter the name for current Production Sustainment Release in the CURRENT\_VERSION input field.
7. Enter the name for this new Sustainment Release in the NEW\_VERSION input field.
   1. Release version names should follow the naming convention R17.X.Y, where X is the sprint number and Y will be 0 for the first iteration of this sprint and increment by one (1) for each additional iteration.
8. Ensure the “TEST\_MODE” field is set to “NO” unless you are testing this process (for the developer only!)
9. Click the “BUILD” button.
   1. Within a few seconds, a new build job will begin executing. The new job number will be displayed on the left-hand pane with a blinking circular icon next to it and a build number.
   2. This job will update the create two new branches for release definitions and set the versions to a new SNAPSHOT. It will also set the current Production release as the “previous” release that came before this new one.
      1. For example, if R17.22.0 is currently in Production and you are preparing for R17.23.0 then new branches for R17.23.0 (from R17.22.0) and R17.23.WLS.0 (from R17.22.WLS.0) will be created and updated appropriately.
10. Click the hyperlink on the left-hand column to open this build job.
    1. The job overview screen will be displayed.
11. On the left-hand column, choose (click) the “Console Output” task to monitor progress of this job.
    1. The browser will navigate to the Jenkins job console output screen and a running log of output from the deployment job will be displayed. This will update automatically as new operations are performed.
12. Once the build finishes:
    1. If the status is reported as: “Finished: FAILURE” investigate the problem and correct it.

# Building Release Definitions

After new release definitions component branches are created you must start a build for each using the following process.

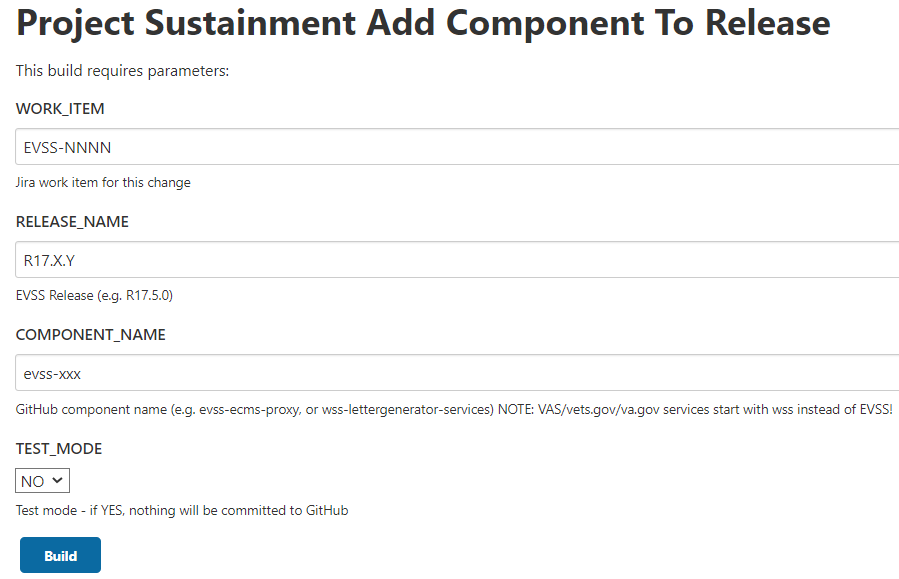
1. Login to the EVSS CI environment Jenkins console at the URL:  
   <http://vaausessapp88.aac.va.gov:8001/jenkins/>
2. Navigate to one of the “10-Dependnecy-CUT” tab.
3. Select the “cut-evss-release-definition” project.
4. Select “Build with Parameters” from the menu on the left-hand side of the screen.  
     
   Graphical user interface, text, application, email

   Description automatically generated
5. Leave the JKD\_VERSION set to the default value.
6. Leave the MAVEN\_GOALS set to the default value.
7. Scroll through the list of branches and locate the “R17.X.Y’ branch for the target release.   
     
   **NOTE** this sometimes will have a prefix like “origin/branches/R17.X.Y”. This list is automatically loaded from GitHub and should always include all possible choices.
8. Click the “BUILD” button.
   1. Within a few seconds, a new build job will begin executing. The new job number will be displayed on the left-hand pane with a blinking circular icon next to it and a build number.
9. Click the hyperlink on the left-hand column to open this build job.
   1. The job overview screen will be displayed.
10. On the left-hand column, choose (click) the “Console Output” task to monitor progress of this job.
    1. The browser will navigate to the Jenkins job console output screen and a running log of output from the deployment job will be displayed. This will update automatically as new operations are performed.
11. Once the build finishes:
    1. If the status is reported as: “Finished: FAILURE” investigate the problem and correct it or notify the developer(s) to address the issue.
12. Once complete repeat steps 4-11 for the “R17.X.WLS.Y” release-definitions branch.
    1. For example, the 17.22.0 release required builds of “R17.22.0” and “R17.22.WLS.0” branches of the release definitions.

# Add Component(s) to Release

Our overall process keeps each component set to a new SNAPSHOT version, so they are generally always ready for development work. However, to support our build and deployment automation we need to track which components are actually being updated. This is done by noting the new SNAPSHOT version for such a component in the release-definitions index.

Use the following process to add a component to a release. This can be repeated as many times as needed when multiple components are to be updated int that release (a normal occurrence.)

1. Login to the EVSS CI environment Jenkins console at the URL:  
   <http://vaausessapp88.aac.va.gov:8001/jenkins/>
2. Navigate to the “998-Stream Management” tab.
3. Select the “Sustainment Add Component To Release” project.
4. Select “Build with Parameters” from the menu on the left-hand side of the screen.  
     
   
5. Enter the number for the work item associated with preparing the release in the WORK\_ITEM input field (created above.)
6. Enter the name for this new Sustainment Release in the RELEASE\_NAME input field.
   1. Release version names should follow the naming convention R17.X.Y, where X is the sprint number and Y will be 0 for the first iteration of this sprint and increment by one (1) for each additional iteration.
7. Enter the name for the *component repository in GitHub* in the COMPONENT\_NAME input field.
   1. For example, use “evss-framework” for the framework component from GitHub EC, or “wss-form526-services” for the 526 services component from GitHub.com.
   2. If insure, browse and locate the desired component in GitHub.com (https://github.com/department-of-veterans-affairs) or GitHub EC  
      (https://github.ec.va.gov/EPMO) to confirm the precise component name.
8. Ensure the “TEST\_MODE” field is set to “NO” unless you are testing this process (for the developer only!)
9. Click the “BUILD” button.
   1. Within a few seconds, a new build job will begin executing. The new job number will be displayed on the left-hand pane with a blinking circular icon next to it and a build number.
10. Click the hyperlink on the left-hand column to open this build job.
    1. The job overview screen will be displayed.
11. On the left-hand column, choose (click) the “Console Output” task to monitor progress of this job.
    1. The browser will navigate to the Jenkins job console output screen and a running log of output from the deployment job will be displayed. This will update automatically as new operations are performed.
12. Once the build finishes:
    1. If the status is reported as: “Finished: FAILURE” investigate the problem and correct it or notify the developer(s) to address the issue.
13. Once complete repeat steps 4-12 for any other components you wish to add.
    1. **NOTE**: You can also come back later and add additional components at any time before finalizing the release.

# Review and Approve Changes in GitHub

Adding components to the release (above) resulted in changes to each component. These changes were made in a (new) temporary branch and a Pull Request (PR) was submitted to merge these changes into the master branch. Those PRs need to be approved and merged into the master branch, after which the temporary branch for each component can be deleted.

Follow the order in which components appear in the release-definitions parent pom file. The order is important for some of them, and this file lists the correct order.

**CM stops here; remaining steps performed by development team**

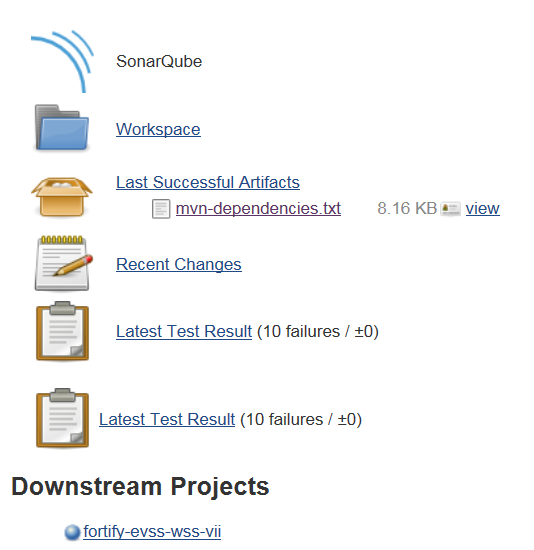
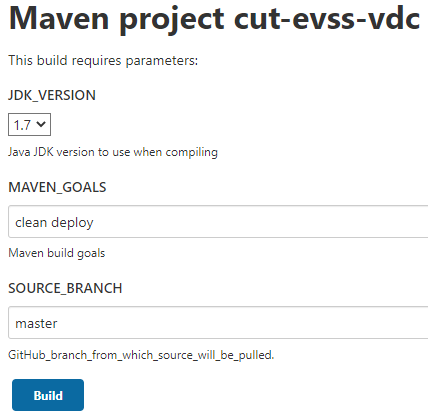
# Commit Code to GitHub

Before building and deploying a code change, that change must be checked in to GitHub and merged into the master branch.

The process used by our Development team is based on GitHub Flow and is documented in a companion SOP entitled “*EVSS Developer GitHub Flow SOP*”. Please consult that document for details.

# Building Components

Builds are started automatically when the master branch for a component is updated in GitHub. They can also be done manually through the EVSS CI Jenkins server. The following procedure should be used:

1. Login to the EVSS CI environment Jenkins console at the URL:  
   <http://vaausessapp88.aac.va.gov:8001/jenkins/>
2. Navigate to one of the “CUT” tabs depending on which component you plan to build:
   1. For dependency projects not directly associated with any particular application/tier, select the “10-Dependency-CUT” tab.
   2. For Partner service libraries, select the “20-Partner-CUT” tab.
   3. For EBN/Services tier projects, select the “30-Services-CUT” tab.
   4. For SEP/Portal tier projects, select the “40-Portal-CUT” tab.
   5. For VDC/Webparts tier projects, select the “50-WebParts-CUT” tab.
   6. For VAS/VA.gov Services tier projects, select the “990-VETS.GOV-CUT” tab.
3. Select the “cut-xxx” project you intend to build. For example, to build the vdc (evss-vdc) project, go to the “50-WebParts-CUT” tab and select the “cut-evss-vdc” project.
4. Select “Build with Parameters” from the menu on the left-hand side of the screen.  
     
   
5. Leave all settings at the default value.
6. Click the “BUILD” button.
   1. Within a few seconds, a new build job will begin executing. The new job number will be displayed on the left-hand pane with a blinking circular icon next to it and a build number.
7. Click the hyperlink on the left-hand column to open this build job.
   1. The job overview screen will be displayed.
8. On the left-hand column, choose (click) the “Console Output” task to monitor progress of this job.
   1. The browser will navigate to the Jenkins job console output screen and a running log of output from the deployment job will be displayed. This will update automatically as new operations are performed.
9. Once the build finishes:
   1. Maven dependency information collected during the build will be available from the Project overview page.
   2. Following completion of the CUT build, a Fortify job will automatically be initiated to scan the code base for security issues. The Fortify jobs and resulting reports can be found under the “91-Fortify” tab in Jenkins. A link to this job is also available on the Project overview page.
   3. If the status is reported as: “Finished: FAILURE” investigate the problem and correct it or notify the developer(s) to address the issue.

# Deploying Components

Deploying each EVSS artifact follows a mostly consistent process regardless of which component you are working on. However, there are small differences in some cases. This document will show one basic example that applies everywhere, using common-services as the sample component.

## Deploying Common-Services

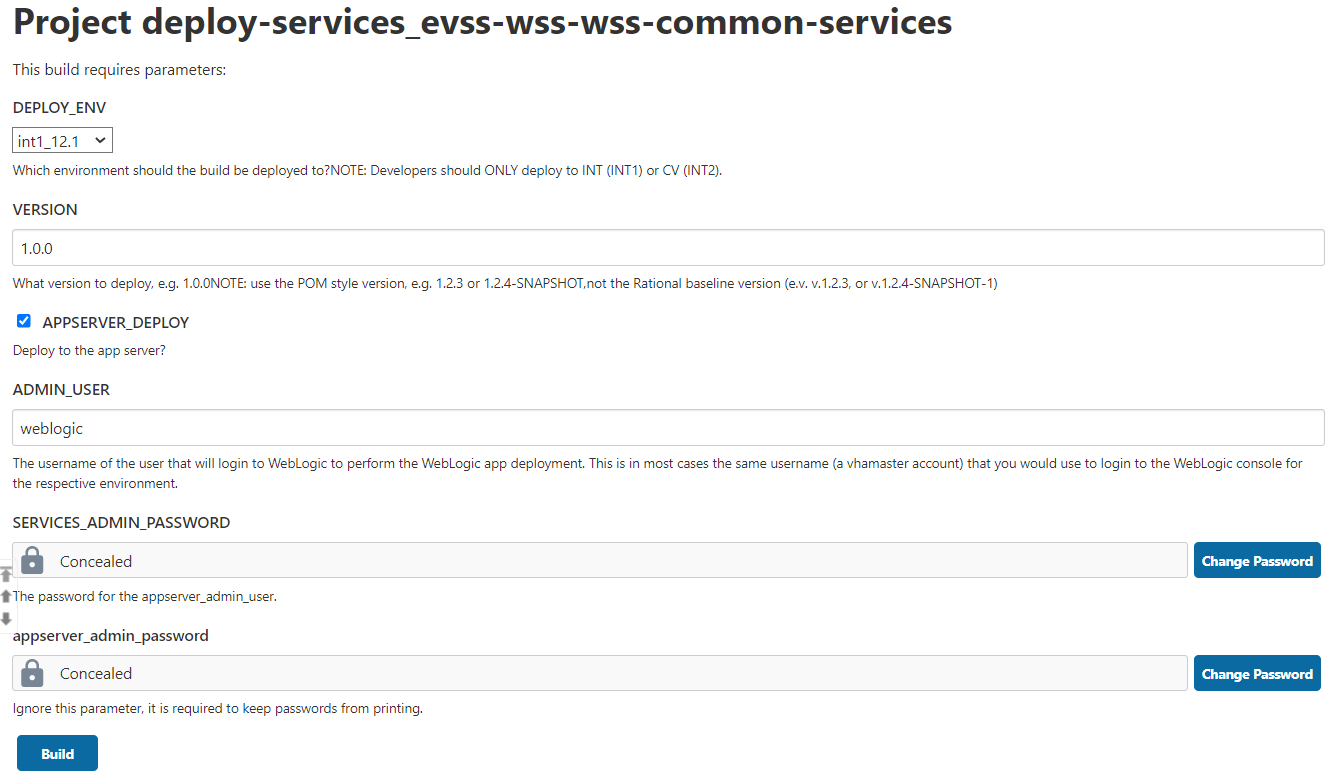
There are two main tasks for the deployment:

1. Deploying artifacts
2. Validating the deployment

Each is described in detail below. Common problems are described at the end in case something goes wrong.

### Deploying Artifacts

|  |  |
| --- | --- |
| Jenkins Deployment Project | deploy\_services-evss-wss-wss-common-services |
| Deployed Artifacts | common-services-web |
| EVSS Application Tier | Services |
| Deployment Version | *X.Y.Z-SNAPSHOT* |

1. Open the (new) EVSS VA CI Jenkins server Deployment task tab in a web browser.
   1. URL: http://vaausessapp88.aac.va.gov:8001/jenkins/view/92-Deploy/
2. Choose (click) the *Jenkins Deployment Project* from the “Name” column of available deployments.
   1. Your browser should navigate to Jenkins project overview screen.
3. On the left-hand column choose the “Build with Parameters” link from the Jenkins task menu.
   1. Your browser should navigate to the Build Parameters screen with input options for customizing this deployment.  
      
4. Leave the “DEPLOY\_ENV” field set to “INT1\_12.1” (this is the default.)
5. Type the *Deployment Version* into the “VERSION” input field.
   1. This is version in the POM file for the component you intend to deploy, such as 1.2.4-SNAPSHOT. Internal development builds should ALWAYS be “SNAPSHOT” builds!
6. Leave the “APPSERVER\_DEPLOY” box CHECKED (this is the default.)
7. Leave the “ADMIN\_USER” field with the default value which is correct for the INT1 environment.
8. Leave the “SERVICES\_ADMIN\_PASSWORD” field with the default value which is correct for the INT1 environment.
9. Leave the “APPSERVER\_ADMIN\_PASSWORD” field with the default value which is correct for the INT1 environment.
10. Click the BUILD button.
    1. Within a few seconds, a new build job will begin executing. The new job number will be displayed on the left-hand pane with a blinking circular icon next to it and a build number.
11. Click the hyperlink on the left-hand column to open this build job.
    1. The job overview screen will be displayed.
12. On the left-hand column, choose (click) the “Console Output” task to monitor progress of this job.
    1. The browser will navigate to the Jenkins job console output screen and a running log of output from the deployment job will be displayed. This will update automatically as new operations are performed.
    2. A sequence of steps associated with deploying the builds artifacts will execute and results will be reported. At the end, elapsed time and a status message will be displayed.
13. If the status is reported as: “Finished: SUCCESS” the deployment should be completed successfully.
14. If the status is report as “Finished: FAILURE” a problem occurred that will need to be corrected (see Common Problems and Resolution section below) or the deployment will need to be done manually.

### Validating the Deployment

To confirm successful deployment, do the following:

1. Login to the WebLogic Admin Console for the appropriate *EVSS Application Tier*.
2. Verify that the *Deployed Artifacts* are deployed and in a running state on the managed nodes/cluster.

## Common Problems and Resolutions

1. Deployment deferred until managed server becomes available.  
     
   If a managed server node is not responding the WebLogic deployment tool “wldeploy” will report an exception when attempting to deploy an artifact and log a message that the deployment is deferred until that node becomes available. While this generates a long exception message/backtrack in the log, it is NOT a fatal error and WILL NOT cause the overall deployment to fail. However, it can be disconcerting so we are documenting this to tell you what to look for and not to worry.  
     
   You can recognize this type of error by message such as the following:

|  |
| --- |
| [wldeploy] Task 12 deferred: [Deployer:149026]deploy application sep-service-14.6 on Cluster001.  [wldeploy] Target state: deploy deferred on Cluster Cluster001  [wldeploy] java.rmi.RemoteException: [Deployer:149145]Unable to contact "ManagedServer003". Deployment is deferred until "ManagedServer003" becomes available.  [wldeploy] at weblogic.deploy.service.internal.transport.UnreachableHostException.writeReplace(UnreachableHostException.java:47)  … Many more backtrace lines … |

As noted above, this should be considered a “warning”, but does not cause the deployment to fail. When the node is brought back online it should receive the updated component via the normal WebLogic synchronization.   
  
It is normal for ManagedServer002 to be down in our INT2 environment, so you should expect to see this error in the logs.

1. Artifact not undeployed:  
     
   When switching to Jenkins deployments from manual deployments it is not unusual to have the deployment fail due to the inability undeploy previous artifacts.  
     
   You can recognize this type of error by message such as the following:

|  |
| --- |
| weblogic.application.ModuleException: Context path '/iw-wsprovider' is already in use by the module: sep-service-14.6.2-20180119.162724-1.war application: sep-service-14.6.2-20180119.162724-1  at weblogic.servlet.internal.WebAppModule.validateContextPath(WebAppModule.java:1365)  … Many more backtrace lines … |

If this occurs, the resolution is to login to the WebLogic admin console and manually stop, then undeploy the Artifacts for this project, the start the Jenkins deployment over.

1. Artifact not targeted to cluster:  
     
   When switching to Jenkins deployments from manual deployments it is not unusual to have the deployment fail due to the automated deployment targeting cluster nodes differently that was done in the manual deployment.  
     
   You can recognize this type of error by message such as the following:

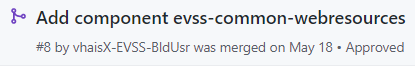
|  |
| --- |
| Deployer:149139]Application "sep-theme" is targeted to cluster "Cluster001", and provided the target for deploy operation is "ManagedServer001". The deploy operation on a subset of the targeted clustered servers is not supported.  at weblogic.utils.StackTraceDisabled.unknownMethod() |

If this occurs, the resolution is to login to the WebLogic admin console and manually stop, then undeploy the Artifacts for this project, the start the Jenkins deployment over.  
  
Be sure NOT to leave the configuration locked, or the deployment will fail again!

# Appendix 1: Approving and Merging Changes in GitHub

For each component updated one of the steps above (except release-definitions), jenkins created a temporary branch and submitted a PR to merge the version changes into the master branch in GitHub. Those PRs need to be approved and merged into the master branch, after which the temporary branch for each component can be deleted. Follow these steps to complete this process.

1. Graphical user interface, text, chat or text message

   Description automatically generatedNavigate to the component repository in GitHub (ex. evss-common-webresources, as shown here.)
2. Click on the “Pull Requests” link/tab near the top of the page.
3. Select the PR with the description “Add component XXX ” from the list.
4. From the PR overview page, click the “Add your review” link to review and approve this PR.  
     
   Graphical user interface, text, application, email

   Description automatically generated
5. Verify the changes in this PR. You should expect a single one line change in each component:
   1. The release-definitions version in the (ONE) parent pom should have the version changed to a new SNAPSHOT version for this release (e.g. R17.24.0-SNAPSHOT for R17.24.0).
6. Approve and Submit your review for the update.
7. Upon return to the PR overview page, click the “Merge pull request” and then “Confirm merge” buttons to complete the process and merge the changes into the master branch.
8. Confirm you see the message “Pull request successfully merged and closed”. If not, resolve this problem before continuing!
9. Click the “Delete branch” button to delete the temporary branch used to create these changes, since it is no longer needed.