Development - Azure DevOps Process Flow (Quality Assurance)

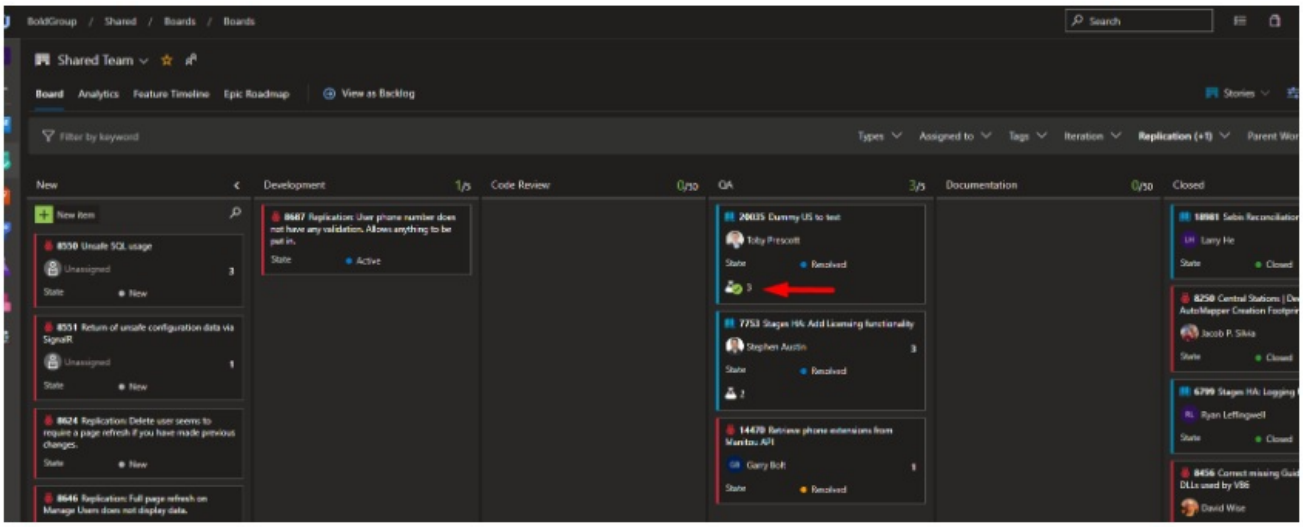
12/29/2023 4:24 pm EST

Overview

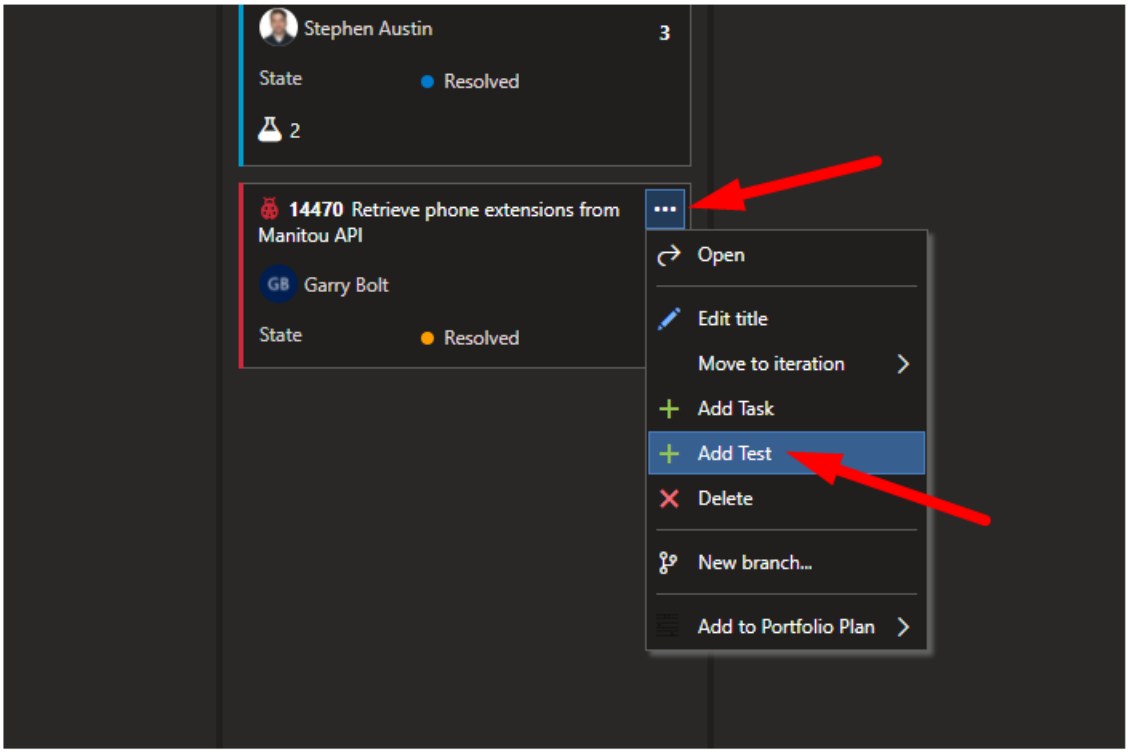
This is intended to give a start to finish look at the typical QA process using our Agile methodology in Azure DevOps platform. This should allow QA testers to quickly become proficient at managing their items on the various boards in a consistent manner.

Test Planning

At the end of Sprint Planning, developers will be adding their tasks to the individual work items. QA will need to begin adding one or more Test Cases to each work item. These can be added from the Stories Board quickly and easily. They should be added to all work items in the New and Development states. It is easy to tell which ones need test cases by looking for the beaker on the card.



You can easily add the shell of a test to a work Item through the menu on that work item.



Test Explorer is fantastic for documenting test steps.

Every work item should have one or more tests attached.

Ideally, these are created in the first few days of the sprint so that we can quickly move the work items through the process. In some cases, we don't have enough information to real craft the tests. These can be added once the item lands in QA. At this stage they should be set to a "Design" state while they are being defined.

NOTE: Doing it this way also creates a Test Plan for that Sprint. We are going to be using "Release-based Test Plans" going forward. They will either be a maintenance or feature release.

This breaks each work item into individual Test Suites with separate test cases under that.

The hierarchy flows as follows for Release Testing (Each release will have its own test Plan!) [Dev Managers]

These release plans can be exported to Excel for easy inclusion in our Release Checklist process

- Maintenance Release (This will be a top level Test Plan)
    - Static End-to-End Regression Suite
    - KanBan/Bug Testing Suite
- Feature Release
    - Static End-to-End Regression Suite
    - Project Epic/Feature Testing Suite
    - KanBan/Bug Testing Suite

For our purposes, we should use the following definition:

Static Test Suite = End-to-End Regression
Requirement-based Test Suite = Project Epic/Feature Testing
Query-based Test Suite = KanBan/Bug Testing

Test Points record the actual results of a unique combination of test case, test suite, configuration, and tester. This provides the ability to generate reports and historical metrics.

The use of defining configurations allows testing the same single Test Case in many ways without having to duplicate by creating individual Test Points for each configuration! Test Parameters also works similarly.

Pull Requests and Building Branches

As a developer completes work, they will move the work item to the Code Review swim lane. Once the code has been reviewed, they will move it to the QA swim lane. QA can monitor this swim lane on the Stories Board or monitor the Pull Requests they are assigned to. If monitoring the Stories Board, care will have to be taken on User Stories to ensure that all of the individual User Stories for a given Feature are completed before beginning testing. This is so that we can be efficient in our testing as well as prevent incomplete work from being merged into Master prematurely. Bugs however can be pulled directly from the Stories Board safely.

It is also helpful to switch to the Features Storyboard and monitor the QA swim lane as well.

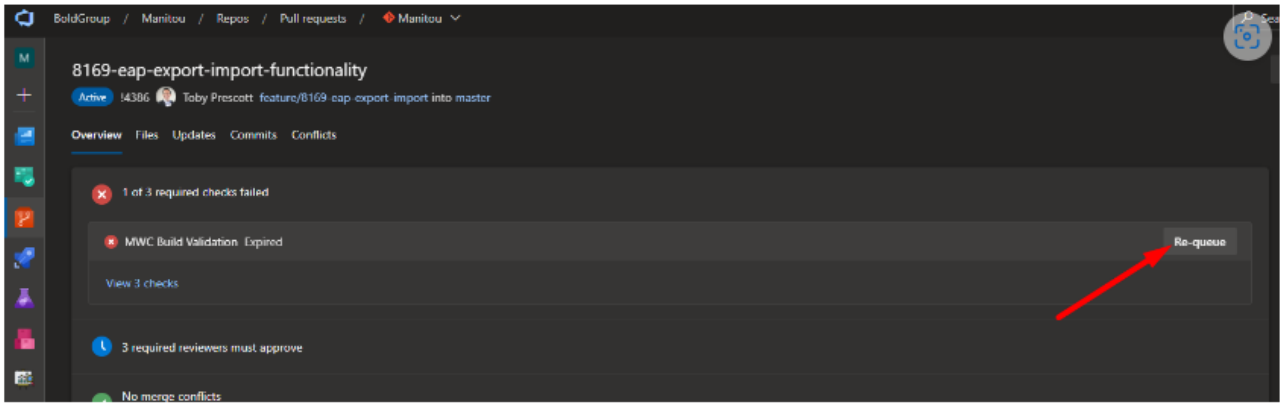
When viewing Pull Requests assigned to you, make sure you keep in mind which repository you are viewing.

Pull Request Checklist

1. The work item is marked as Resolved (QA column on the Board)
2. There are dev notes with proper templates under the Solution Summary section
3. That the PR has been approved by a Team Lead
4. All Comments are marked as Resolved
5. That there are no Merge Conflicts

Once you have decided on which Pull Request you are planning to test, it is time to build that branch.

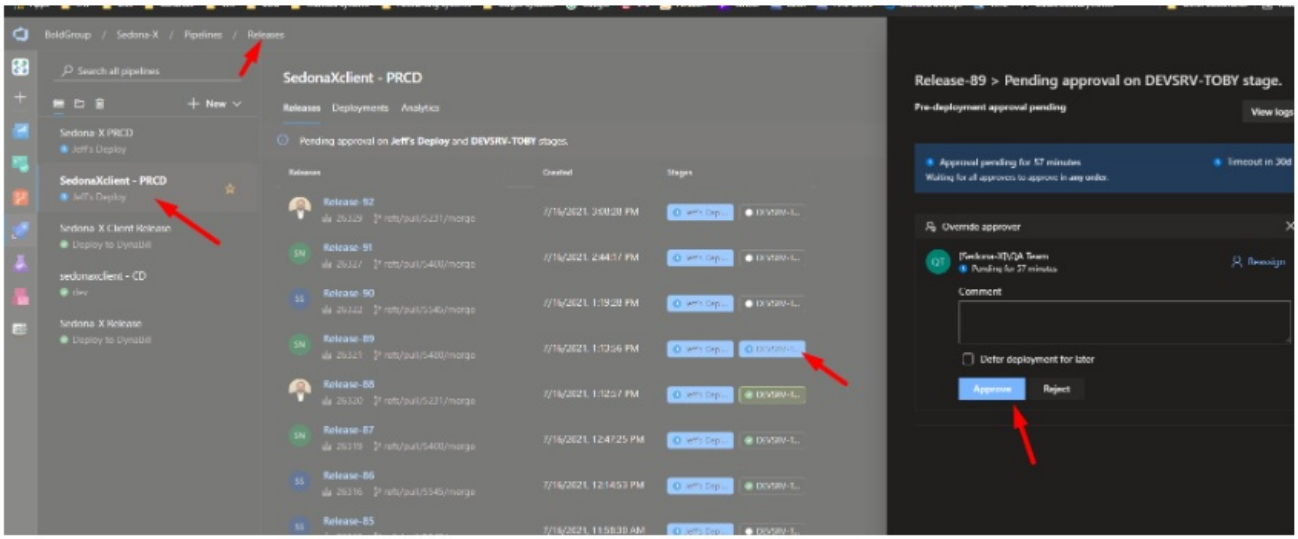
Once the PR is ready, we can kick off a build. This should be done from the Pull Request because that build involves a PRE-merge with master and so the output that QA will test is Master as it sits NOW + the code change in question.



> This removes the need to "merge master into the branch" unless there is a merge conflict. That makes this change much more portable to be cherry-picked into other branches safely.

Deploying Builds

Depending on the project, there are Release Pipelines that can be used to deploy to the appropriate environment. You may have to approve/reject releases to get to the one you intend to test. But then you simply approve the release to deploy. These release pipelines for QA are called PRCD for clarity.

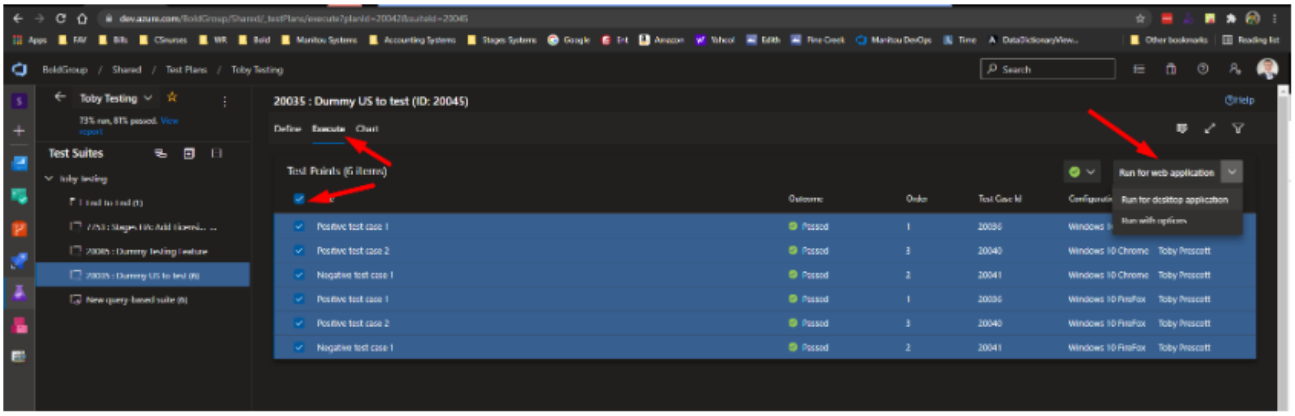


> Some project simply build to an output location and the deployment for testing must be done manually.

Executing Test Cases

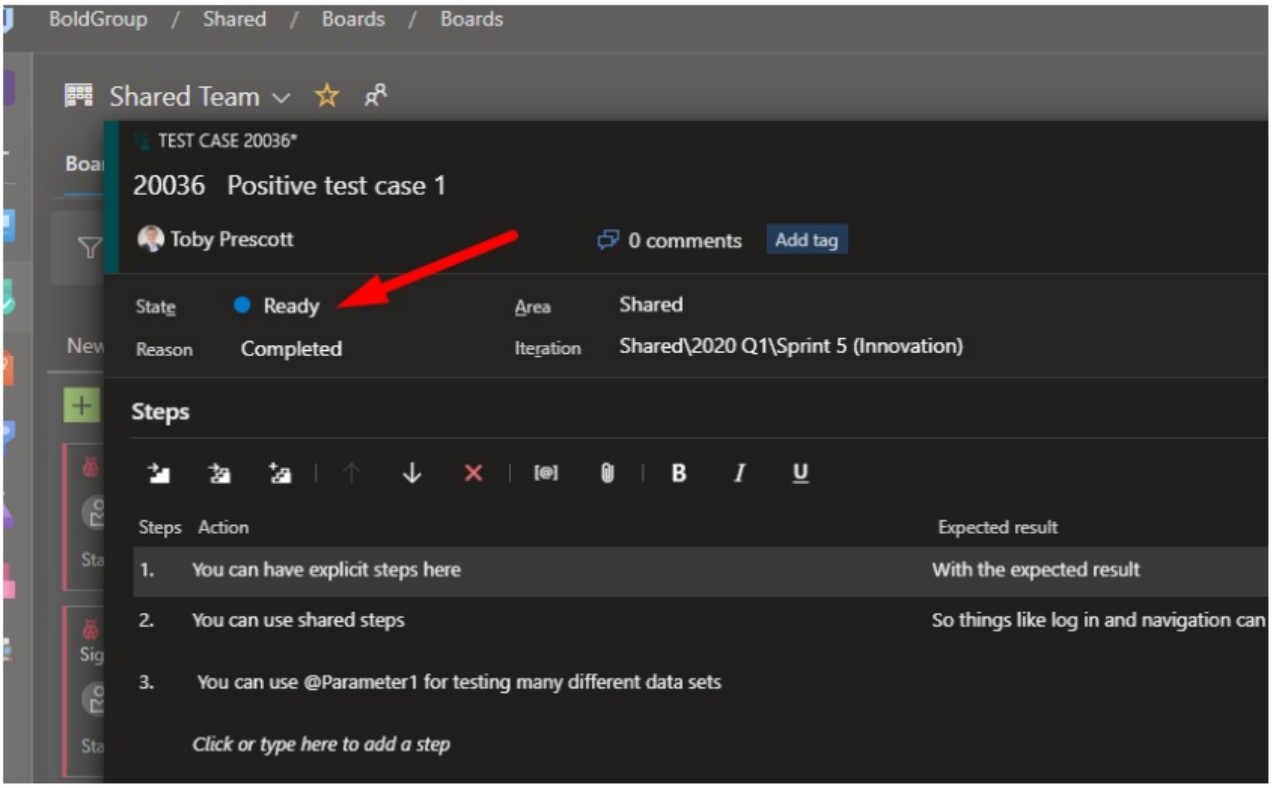
Once the build/deploy finishes, it is time to run the previously defined Test Plans. This is done from the Test Plans view in DevOps. Open the Test Plan built for the Sprint and select a given Test Suite. Select all of the Test Cases and click one

of the Run buttons.



This will launch (I selected the web application option) a window where each step can be marked as success or fail. You can also capture screenshots, etc. Complete all of the tests. Note you switch tests using the drop-down or the next button. Once finished testing you can Save and close the test window.

It is easy to see the final results for the test cases and more details can be found in Test Run Explorer.
The final step is to move the State of the Test Case to Ready to indicate that it has been validated and is ready to be used for release testing.



Work Flow on DevOps Boards

This section focuses on the management of the work items through the flow. QA will be testing Bugs directly but for User Stories will only test the completed Feature. The Feature should have links to all of the associated User Stories. When the Feature passes, QA may have to go back through and move all the User Stories on the board as well. When you know what item(s) you are going to start testing, open the item and set the Tested By to yourself. This will let everyone know what items are being tested currently.
Once work has been tested, there are two paths it can take. First, it can pass testing and move QA --> Documentation stage. Second, it could fail testing and need to be sent back for the developer to clean up. We will look at both paths below.

Passed Testing

The easiest is the path where the work was completed correctly and passed all the QA tests. This means that it is ready to be merged into Master and released whenever we determine to cut the next patch. The following things should happen in this case:

- The QA tester should Approve the Pull Request. This creates the Audit Trail of activity on the Pull Request.
- QA can then Complete the Pull Request which would do the actual merge from the branch into Master. There should not be any merge conflicts but if there is, message the Developer for help right away. If the Pull Request was completed successfully, you can delete the branch. The code is now safely in master.
- Make sure all the test cases are marked as passed on the work time
- Create/Update any Technical Documentation needed
- Add the release notes (and related Technical Documentation links if applicable)
- Last step is to move the work item on the Stories Board QA --> Documentation to let everyone know it is ready for final review. You can then move to testing the next work item in QA.

Failed Testing

This path requires more involvement with the developer. We were using Issues for this but feel that a more direct interaction has proven to resolve things faster and more completely. For this the QA tester should add a comment on the work item with their finding. Then they should directly reach out to the developer and let them know what they found.

Using the DevOps Testing tools allows capturing screenshots and video that gets attached to the failed test point. The dev can review this as well.

If needed, a screen sharing session and demo to the dev is most valuable here. Last they can move the work item back to Development if it will be a time consuming fix. If it is a quick fix, then it can be left in QA. We do not want to develop bad habits of these going back and sitting unnoticed with a developer for extensive periods of time. The goal is to resolve the issue that was found in QA quickly and continue to move things through the pipeline. Only if the issue is large enough in scope that the developer is going to have to significantly rework things should it be sent back. This signifies that QA is moving on to something else.

To summarize, QA moves things from QA --> Documentation in the normal flow of work.

Additional DevOps Test Plan Resources

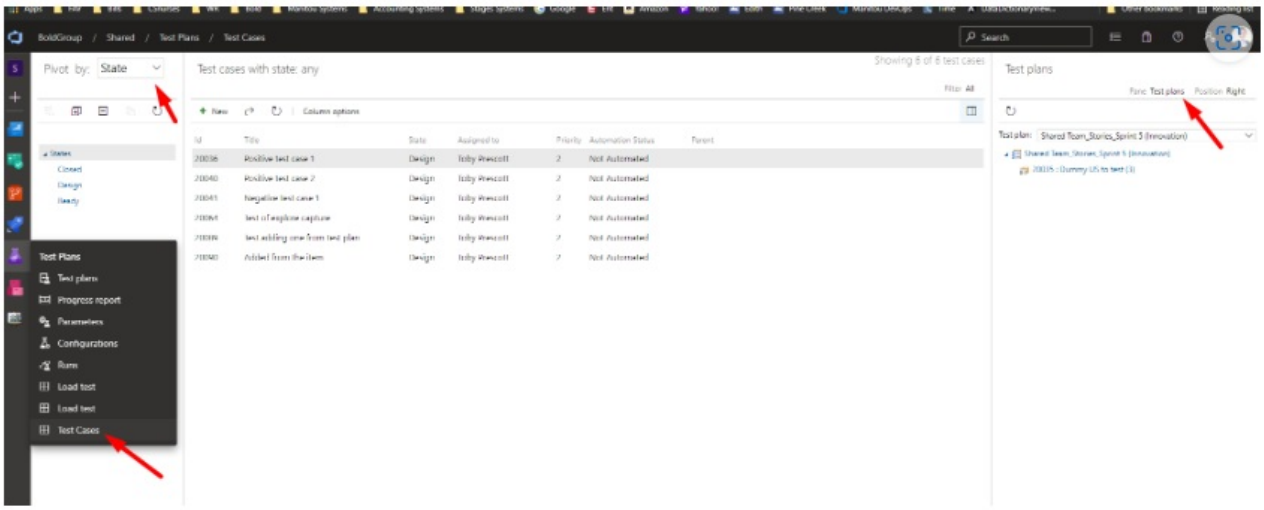
https://docs.microsoft.com/en-us/azure/devops/test/?view=azure-devops

https://docs.microsoft.com/en-us/azure/devops/test/overview?view=azure-devops#planned-manual-testing

https://marketplace.visualstudio.com/items?itemName=ms.vss-exploratorytesting-web#supportedbrowsers

https://www.modernrequirements.com/blogs/documenting-azure-devops-test-plans/

To manage/find Test Cases easily, There is a Test Case Explorer.



It lets you view data in a lot of different ways and filters. You can also drag/Drop to move/add/clone into other test plans easily. This also provides a way to find "orphaned" tests and quickly tie them into Test Plans/Suites.