Development - Azure DevOps Process Flow (Developers)

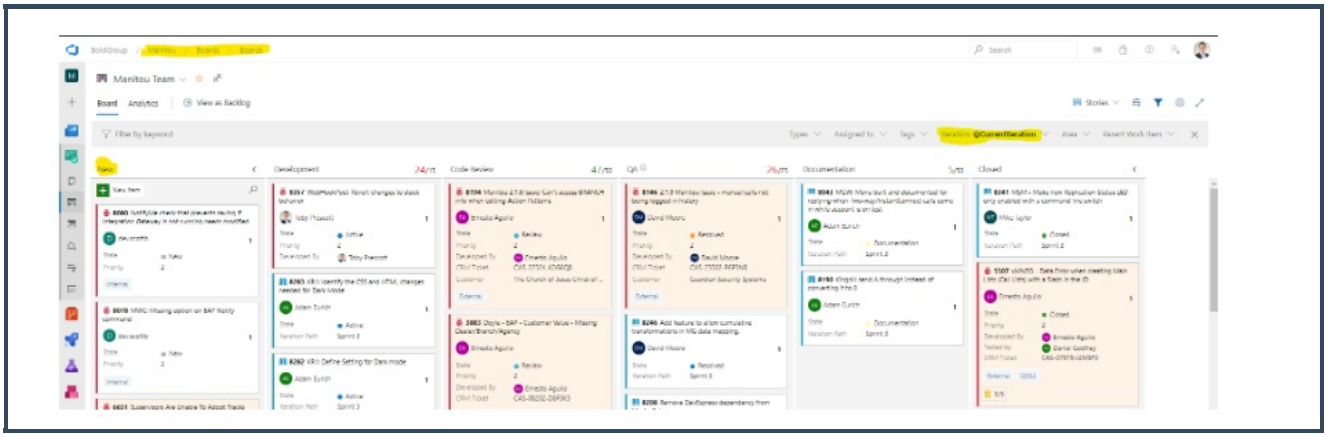
12/29/2023 1:31 pm EST

Overview

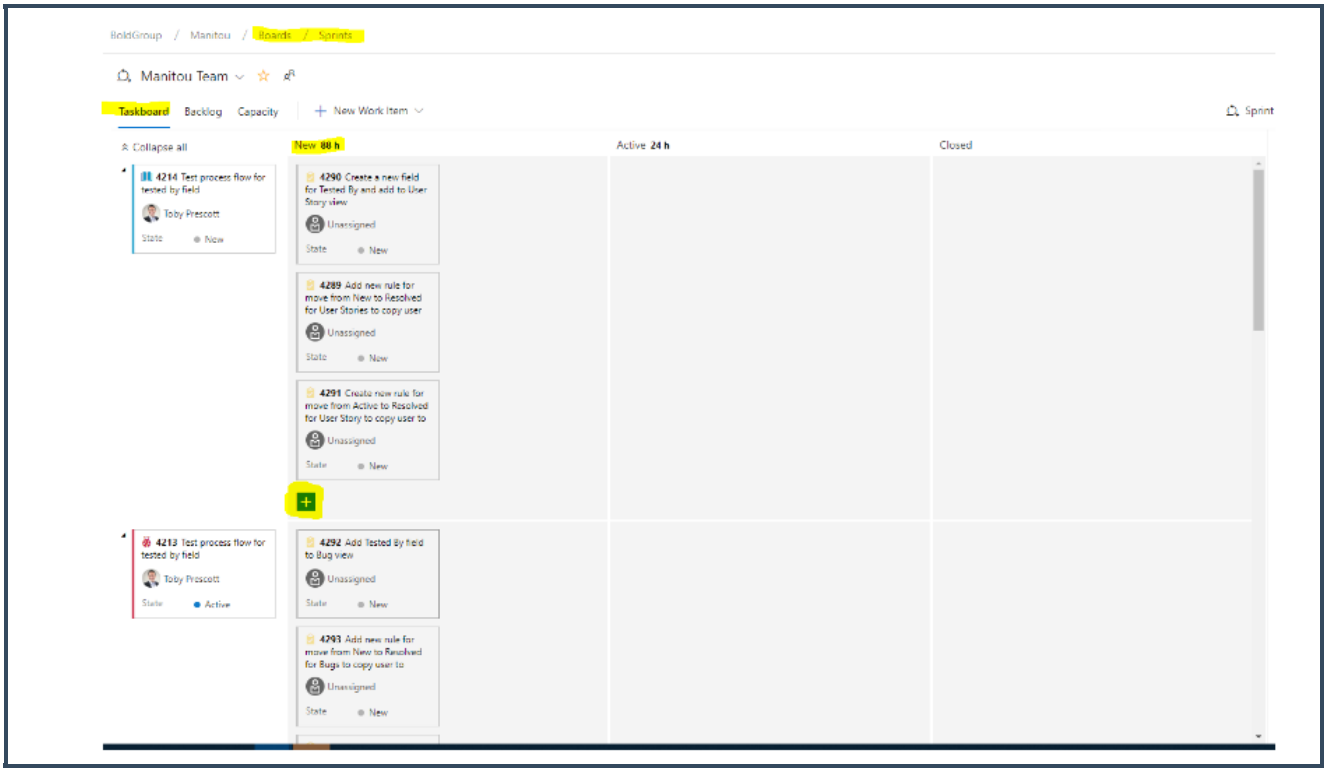
This is intended to give a start-to-finish look at the typical developer process using our Agile methodology in Azure DevOps platform. This should allow developers to quickly become proficient at managing their items on the various boards in a consistent manner.

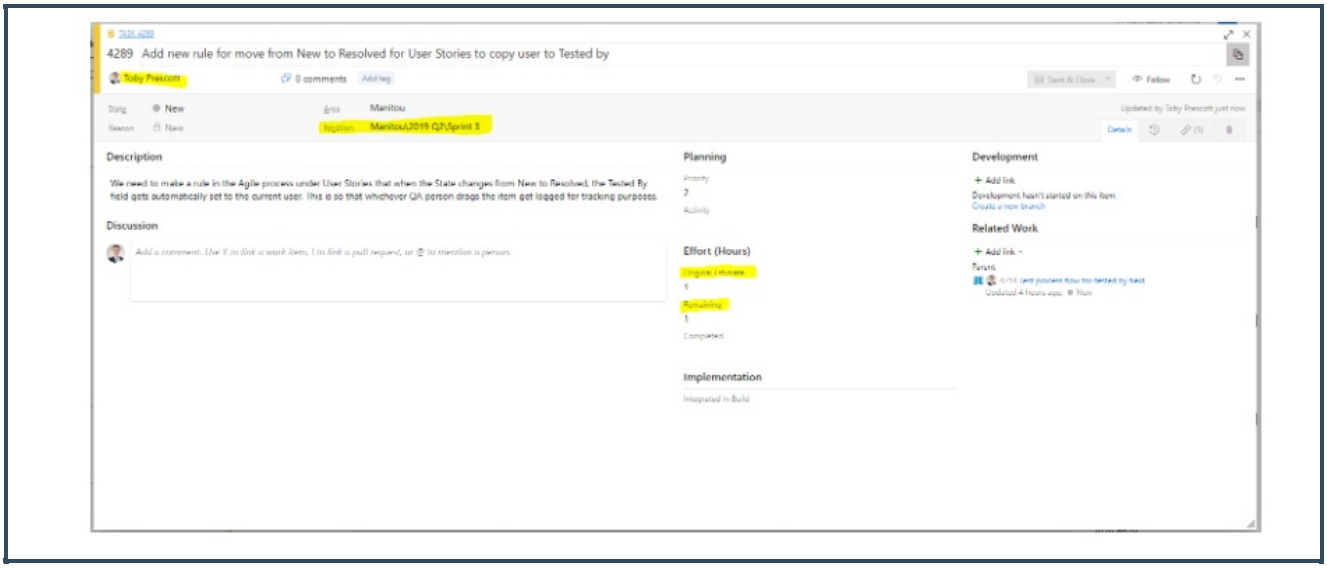
Sprint Planning

At the end of Sprint Planning, each developer will have one or more work items assigned to them. These are either User Stories or Bugs. The state will be new to begin. These can be seen easily from the Stories Boards view in DevOps



Following the Sprint Planning meeting, each developer is expected to spend time digesting and understanding how they intend to accomplish this work item and will add tasks to each work item that reflect the estimated time involved (Overall and Remaining). This should be as detailed as possible to allow accurate tracking of the burn-down chart. The best place to manage this is on the Sprint Board using the Taskboard view grouped by Stories. There you can easily add these tasks under the New section.





Make sure that these tasks are assigned to you if you will be doing the work. QA will be adding tasks as well for testing.

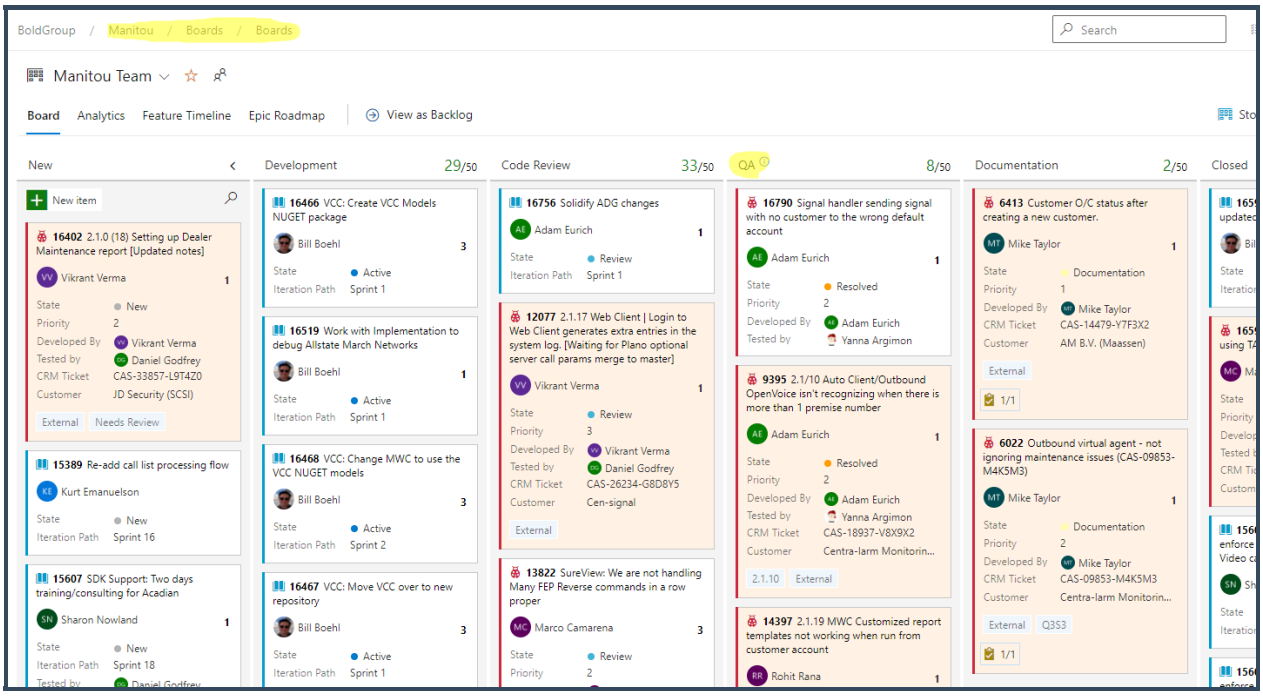
Work Flow on DevOps Boards

As a developer begins working on a specific work item they should do two things.

1. Move the Work Item on the Stories Board.
   Here the flow for the developer moves New --> Development in the swim-lanes at the start of work.
   Once the work is completed by the developer, the Solution Summary section should be completed on the work item using a standard format to provide consistent instructions for QA and Documentation.

2. The item can then be moved Development --> Code Review in the swim-lanes to let other developers know the work is ready for review. They can then move the next item to the Development state.

3. Once Code Review is complete, the item can be moved Code Review --> QA to let the QA team know that they can begin work on this item.



4. Move the Tasks on the Sprint Board.

   At the same time the developer is moving items on the Stories Board, they are also managing individual task on the Sprint Board. This drives the daily burn-down chart as well as let others know where they are in completing the overall item. This has two components:

   - Moving the state of the task.

   - Adjust the hours completed and remaining

     Both of these can be done directly from the taskboard view. The hours remaining gives a nice drop-down on the card itself. It is important to do this at least once per day but more frequent improves smoothness of the burn-down chart. Once all tasks are moved to Closed, then the work item should be moved to the next state on the Stories Board.

Code Review and Pull Requests

Once work has been performed it is time to do a Code review and QA the final work. To begin a Pull Request must be created. Again this can be done in many places but the ideal place is from the work item to keep all the links clean in

DevOps.

This starts with the Code Review portion. One or more other developers should be added as reviewers when creating the Pull Request. The title should include the work item number as well for ease later. Lastly the target branch should be Master for Bugs and Features. User Stories should target the Feature branch where all the work will be gathered and integrated.

The other developers will review the changes in the Pull Request and either Approve or add comments. Comments can be added directly to the code line by hovering over the line number. They can also pull the branch and do testing directly if applicable.

Once the required developers (one team lead and one team member) have approved the Pull Request, the developer doing the work should move the work item Code Reveiw --> QA on the board. This signifies that the item is ready for testing.

Once QA has built and tested the branch, they will approve the Pull Request and Complete it. This will merge the code into Master where it will be released in the next available update. Once this happens, it is safe for the original developer to delete the branch. This is quickly done from the Pull Request. Additionally, the change can be Cherry-Picked into another branch if necessary. This would be for cases where we are doing a Hot-Fix for a released version but need the fix to be put into Master as well.

Pull Request Rules

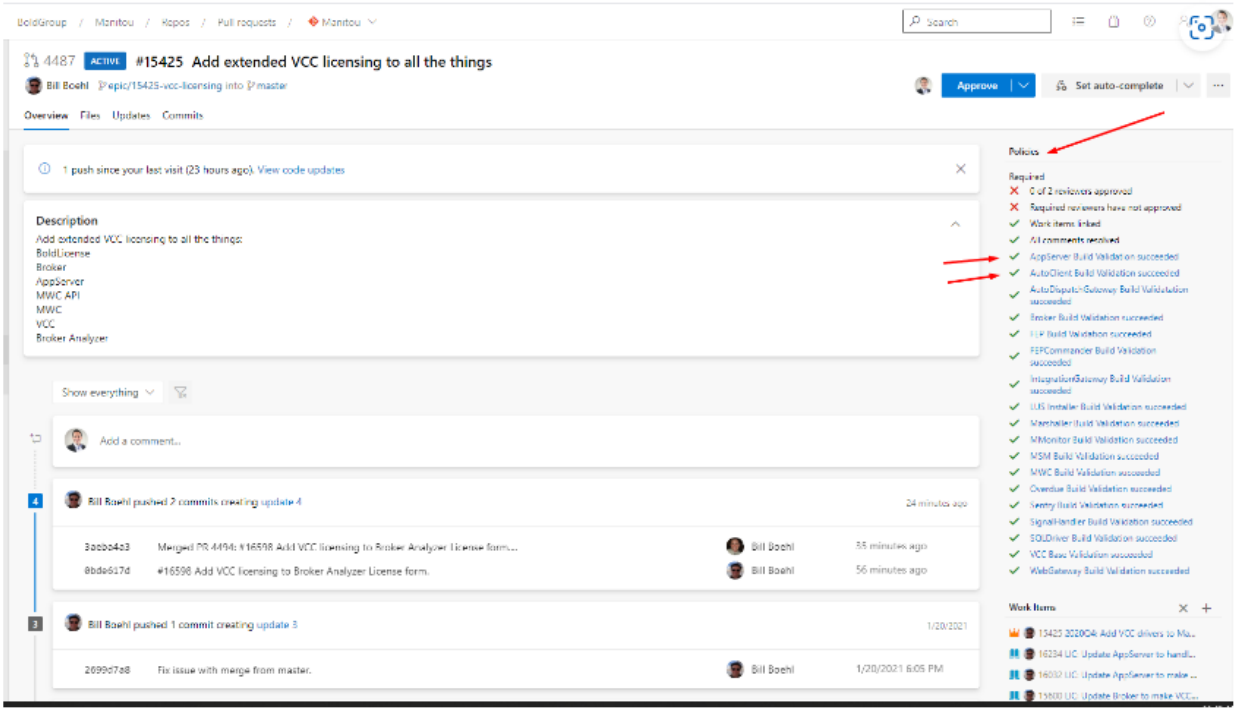
To help ensure our process is followed and high-quality output, several pull request rules should be implemented for changes going into the master branch. These can be applied per project/repo and provide the critical checks and balances for this process to become stable and efficient.

- PRs must have all comments resolved before the completion
- PRs must have required Developer reviewer approval. Ideally, this is a Team Lead for the group
- PRs must have required QA approval
- PRs must have successful build Validation. This will lead to automated testing as we build out
- PRs must have linked work items for traceability

While it is possible to need to override these policies, doing so will require a reason to be entered for tracking purposes. We can do this on a case-by-case basis within reason.

Build Validation

This provides a powerful step in the process and brings huge value to our organization. This requires a build pipeline(s) that runs whenever a PR is created or updated. The unique aspect of this is that it does a PRE-merge with master from the work item branch. This is a critical point to understand. This means that the build output (that should be used by QA for testing) is NOT the code in the branch at that moment. Instead, it is a perfect representation of what Master would look like IF the PR was completed but without the danger of accepting that commit and then testing. This is done by DevOps creating a virtual branch that has Master + Changes and building from that one. But using this powerful tool, we know that what we are testing is the final result and we do not need to test again after the PR has been completed.


BoldGroup / Manitou / Repos / Pull requests / Manitou
Search
4487 ACTIVE #15425 Add extended VCC licensing to all the things
Bill Boehl epic/15425-vcc-licensing into master
Approve
Set auto-complete
Overview Files Updates Commits
1 push since your last visit (23 hours ago). View code updates
Description
Add extended VCC licensing to all the things:
BoldLicense
Broker
AppServer
MWC API
MWC
VCC
Broker Analyzer
Show everything
Add a comment...
Bill Boehl pushed 2 commits creating update 4
24 minutes ago
3ddb2423 Merged PR 4494: #16598 Add VCC licensing to Broker Analyzer License form...
Bill Boehl 55 minutes ago
8bde617d #16598 Add VCC licensing to Broker Analyzer License form.
Bill Boehl 56 minutes ago
Bill Boehl pushed 1 commit creating update 3
1/20/2021
2699d7a8 Fix issue with merge from master.
Bill Boehl 1/20/2021 6:05 PM
Policies
Required
0 of 2 reviewers approved
Required reviewers have not approved
Work items linked
All comments resolved
AppServer Build Validation succeeded
AutoClient Build Validation succeeded
AutoDispatchGateway Build Validation succeeded
Broker Build Validation succeeded
FEP Build Validation succeeded
FEPCommander Build Validation succeeded
IntegrationGateway Build Validation succeeded
LDS Installer Build Validation succeeded
Marshaller Build Validation succeeded
MMonitor Build Validation succeeded
MSM Build Validation succeeded
MWC Build Validation succeeded
Overdue Build Validation succeeded
Sentry Build Validation succeeded
SignalHandler Build Validation succeeded
SOLDriver Build Validation succeeded
VCC Basic Validation succeeded
WebGateway Build Validation succeeded
Work Items