Development Workflow

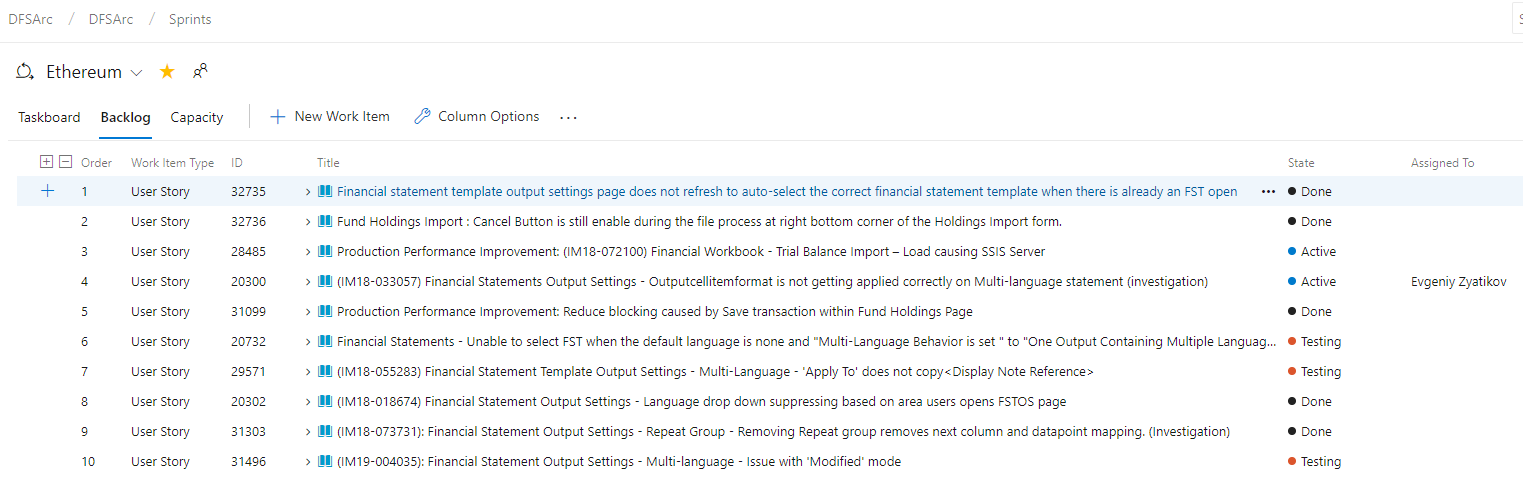
This document is intended to help standardize the development workflow across the ArcSuite products. It is anticipated that developers will need to be able to jump in to any other product's workflow at any time. Standardizing this process will improve our efficiency as we will not distracted by a different workflow every time we step into a different ArcSuite product sprint/code base. It can also set a standard for new DFIN development projects.

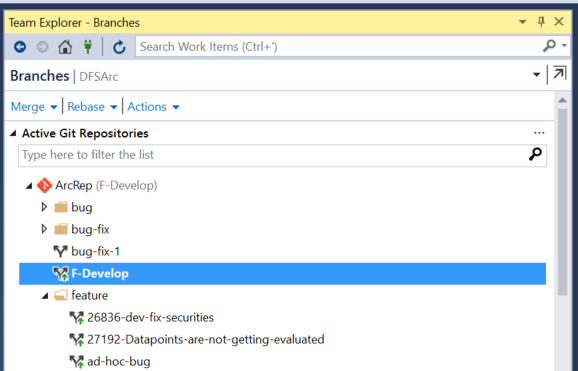
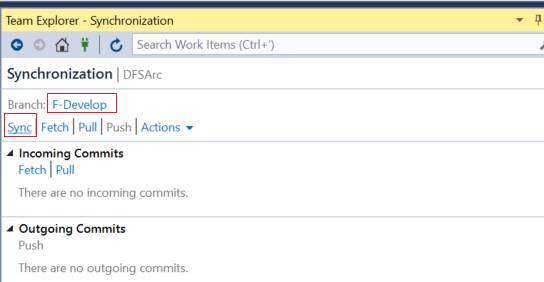
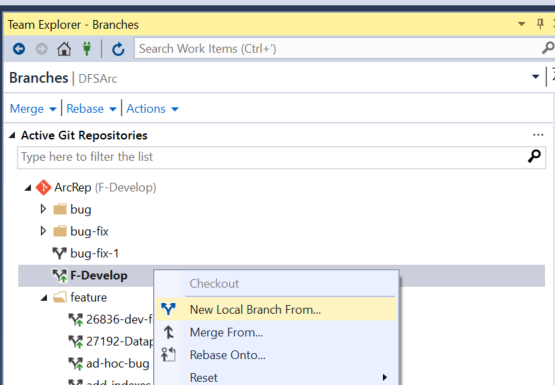
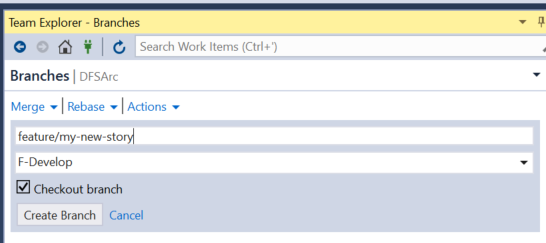
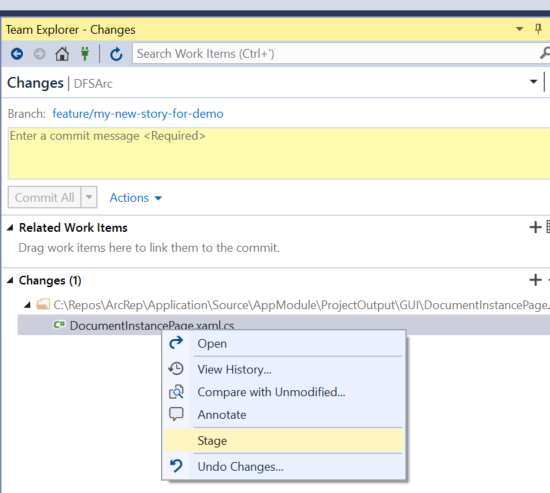
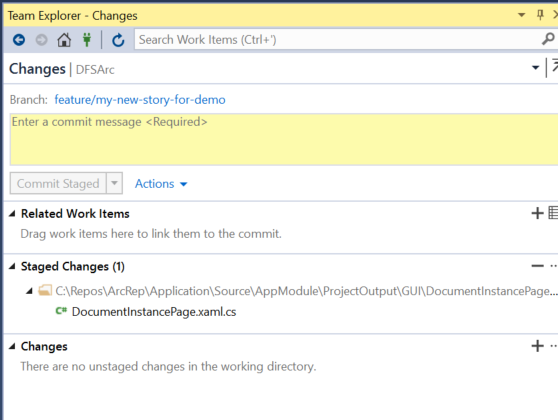
This is currently a living document. Feel free to comment below for clarifications/discussion items.

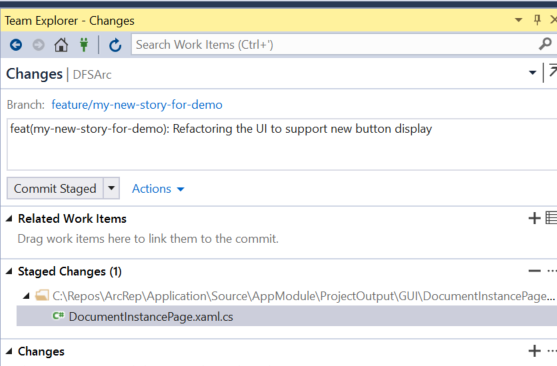
### Developer

(note: UI screenshots are from VS2015 but functionality should be similar)

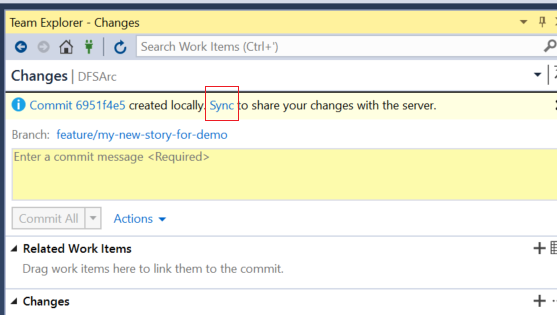
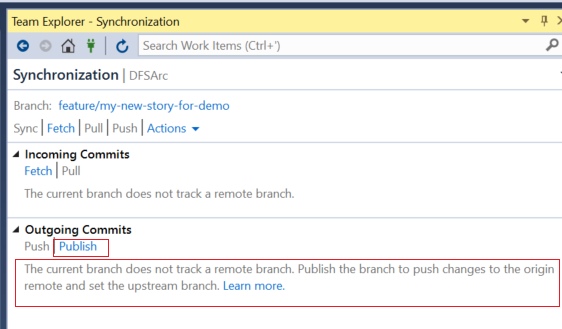
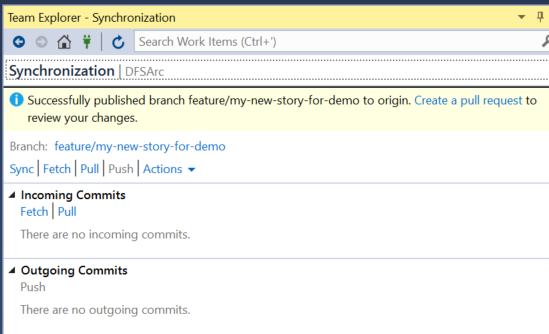
1. Review Sprint Backlog
   1. For Example: <https://dfsarc.visualstudio.com/DFSArc/_sprints/backlog/Ethereum>

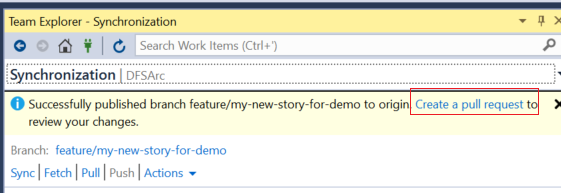
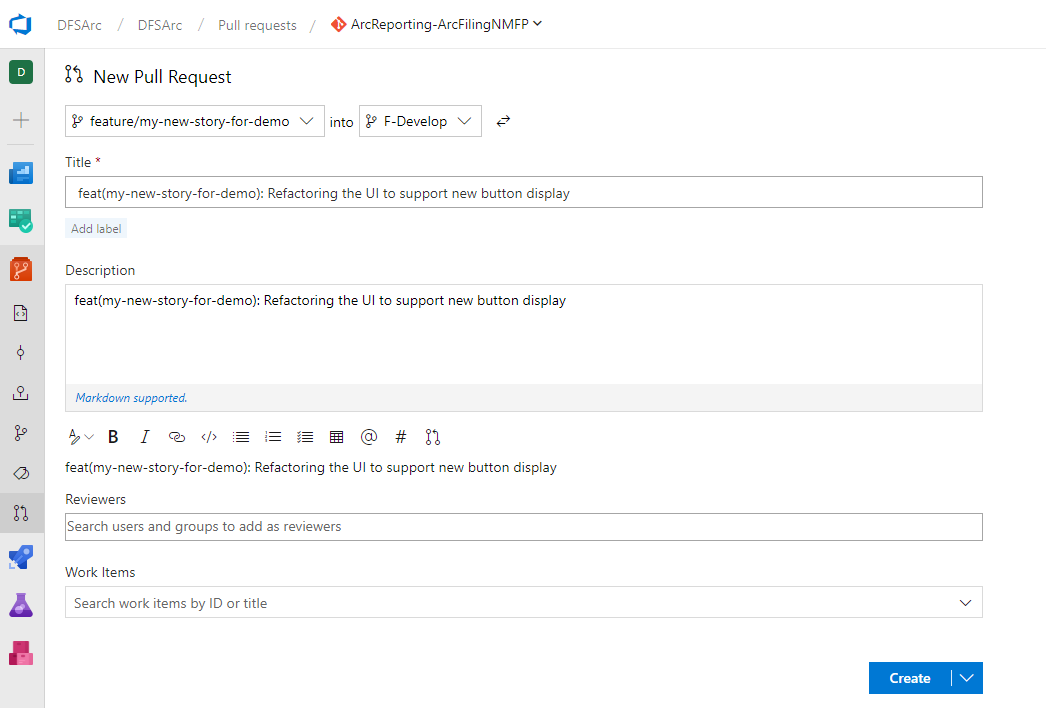
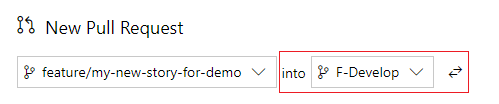
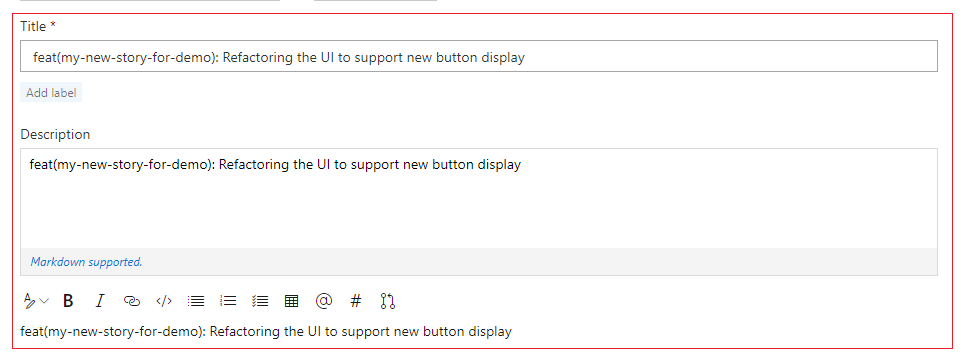


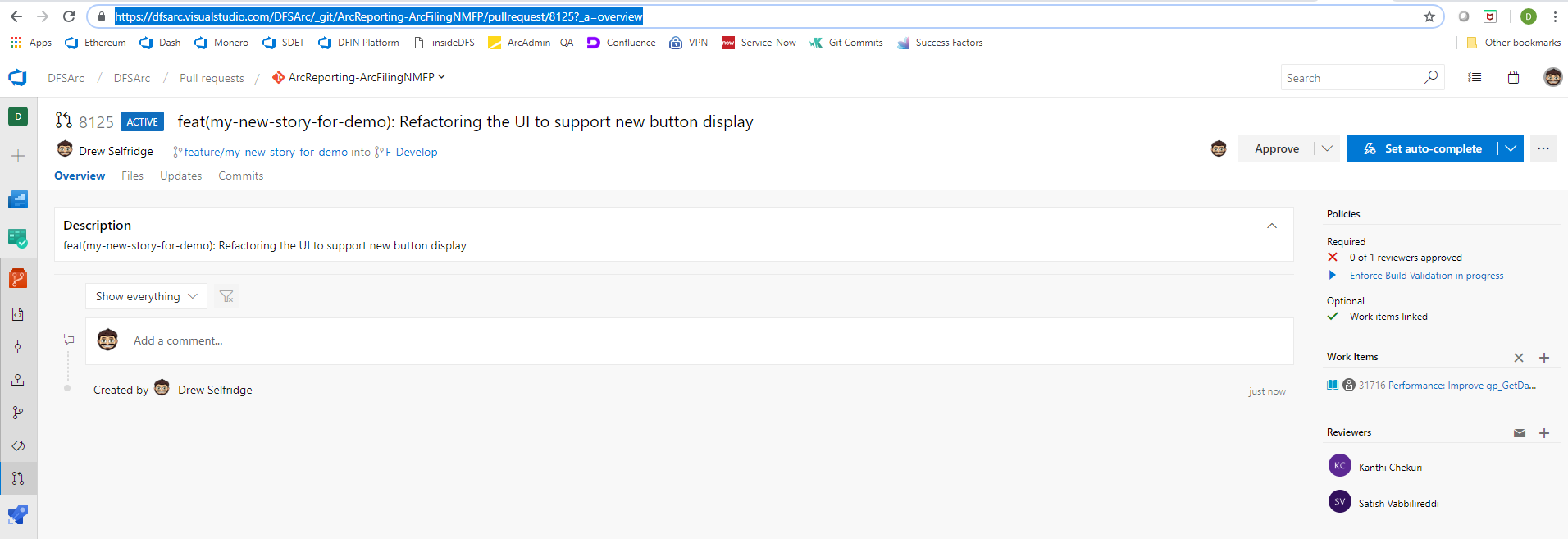
1. Identify story to start - make it Active
2. Review story and create tasks as necessary to get started.
   1. Name Task “Dev: [some work]”. Make it Active.
3. **Checkout** the **F-Develop** branch and **Sync** to get the latest
   1. 
   2. 
4. Create new Local Branch from F-Develop
   1. 
   2. 
5. Begin coding on new local branch
6. Mark any applicable tasks as Active.
7. When unit of work is complete, commit changes to local branch and sync and Publish/Push local branch/code to server
   1. Stage changed files
   2. 
   3. 
   4. Commit the changes

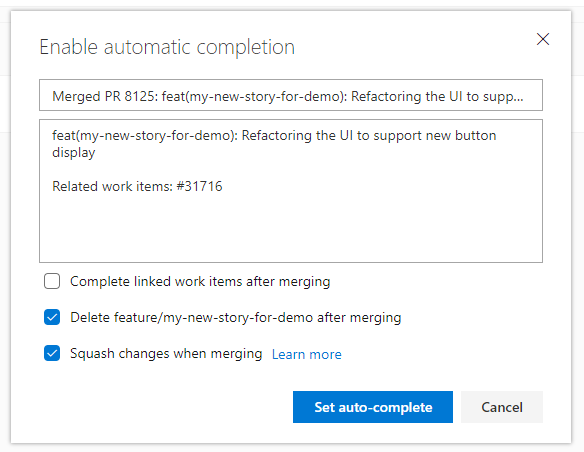


[GIT Commit comment standards](http://karma-runner.github.io/1.0/dev/git-commit-msg.html)

* 1. 
  2. Sync changes with the server
  3. 
     1. In VS2017, it is called “Push” not Publish
  4. Publish/Push branch to server
  5. 

1. Create new PR to get this branch’s code merged into F-Develop:
   1. 
   2. 
   3. Select correct branch to merge into (F-Develop)
   4. 
   5. Enter appropriate title and description
   6. 
   7. Assign reviewers
   8. 
   9. Associate work item - User Story and Task(s)
   10. 
   11. Review files and changes (at bottom of PR) that will be included.
   12. Click Create.
   13. Copy URL of PR and message reviewers in Slack with the URL and request for review. Tag reviewers, requesting they review and approve. Include topic, maybe link to US, etc.



* 1. Set PR to Auto-complete
     1. 
     2. 
     3. DO NOT check box to “Complete linked work items after merging”
     4. Decide if you want to keep your branch for further changes on this feature (or create just a new one) - select “Delete after merging” as needed
     5. DO check box for “Squash Changes…”
  2. Set auto-complete.

1. Note: Auto-Complete will auto-complete and merge the code into the destination branch after approvals are obtained and all other branch policies are satisfied.

### Reviewer

1. Click on link from PR and review code.
2. Add any necessary comments.
3. Once satisfied, click on Approve.

**Note**: if all branch policies on the PR are satisfied, then the PR can be completed. If Auto-Complete was not enabled (see above), then the Developer must go back into the PR and complete it. Completing the PR will merge the code into the destination branch.

**Note:** for the F-Develop branch, only one approver is required, and this approver can be any peer on the team, or dev lead.

### CI/CD

1. After code is merged into F-Develop, a build is automatically triggered.
2. At noon and midnight everyday, a Release to Development is automatically triggered

### 

### Developer

1. Confirm (if possible) task is working as expected in Dev.
2. Change task to “Closed” if applicable.

**Rinse and Repeat :** Developer can create another branch to either work on further tasks on the same story, or take up a new story if the current story is complete.

### Testing

When stories and tasks are completed, unit tested and deployed to dev, the developer marks the User Story in “Testing” state. If team does not have QA, mark item as Done.

[more testing content here]

### Staging

When code must be moved to staging for UAT, a lead developer will submit a PR to merge all changes for the sprint from F-Develop into the staging branch.

The PR can be set to Auto-complete



Do NOT delete source branch, obviously, as this the F-Develop branch.

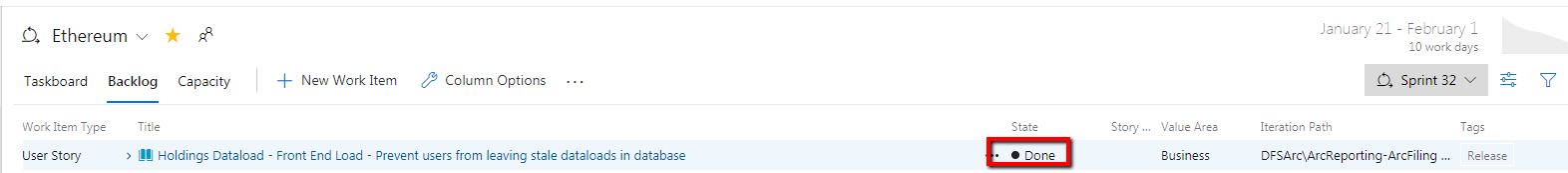
Two approvers are required for merges into the staging branch. Usually dev leads are involved in reviewing the code being merged, to verify work that has been completed thus far in the Sprint.

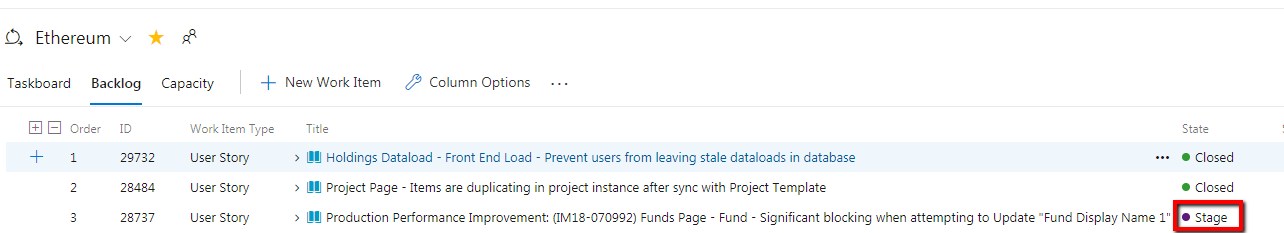
After PR is complete and code is merged, a staging build will auto-trigger. This build will wait for UAT to trigger a release.

Notify UAT team that stage build is running.

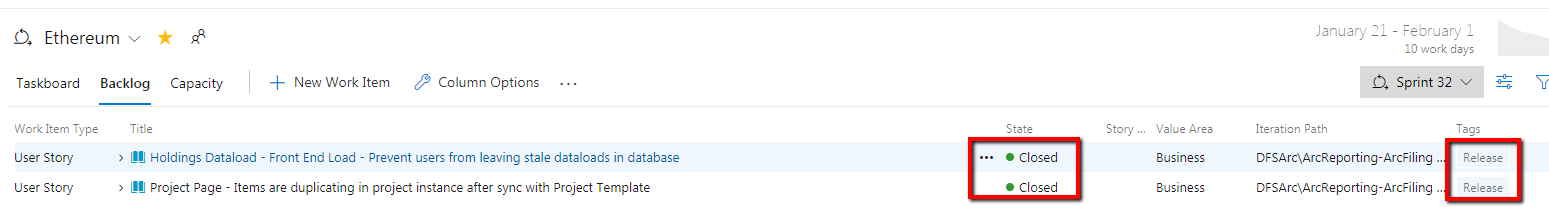
### UAT Process

1. UAT team review the product backlog which is already prioritized and pick up IM ticket at the top to reproduce the bug in stage environment. UAT team will create a “UAT: Reproduce Issue” task under story and provide detailed steps / screenshots to reproduce the production bug and close the task. This helps the team to understand and reproduce the production bugs.
2. UAT team participates in the sprint planning session to go over the items are that scheduled for the sprint and provide inputs / suggestions / steps to reproduce the problem
3. UAT team will create a task under each story to validate the user story in stage before released to production
4. Stage release - UAT team takes a stage release when:
   1. At Least one user story in the sprint is QA complete and status updated to “Done”.
   2. No major blocking issues in QA (Example: dataload not working, outputs failing)
   3. On-demand: When Dev / Product team requests UAT to validate a user story on demand
   4. UAT finds a critical functional issue during testing: The issue will be logged in AzureDevOps as a bug and communicated to the team. After the issue is fixed, UAT will take a stage release; validate the fix and close the defect
   5. UAT finds a critical regression issue during testing: The issue will be logged in AzureDevOps as a bug and communicated to the team. After the issue is fixed, UAT will take a stage release; validate the fix and close the defect
5. User stories to be tested in the release:
   1. UAT will pick up the stories that are marked as “Done” by QA and update the status of those stories to “Stage” before beginning testing



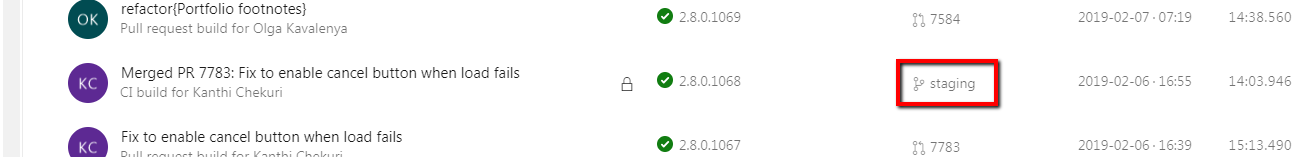


* 1. After the functional and regression testing for the story is completed, the story will be marked as “Closed” and a tag “Release” will be added to let product management team know that the story is good to be released and added to the release notes documentation
  2. UAT team will pick up stories that are marked as “Testing” only if there is a need to test the user story in parallel with QA

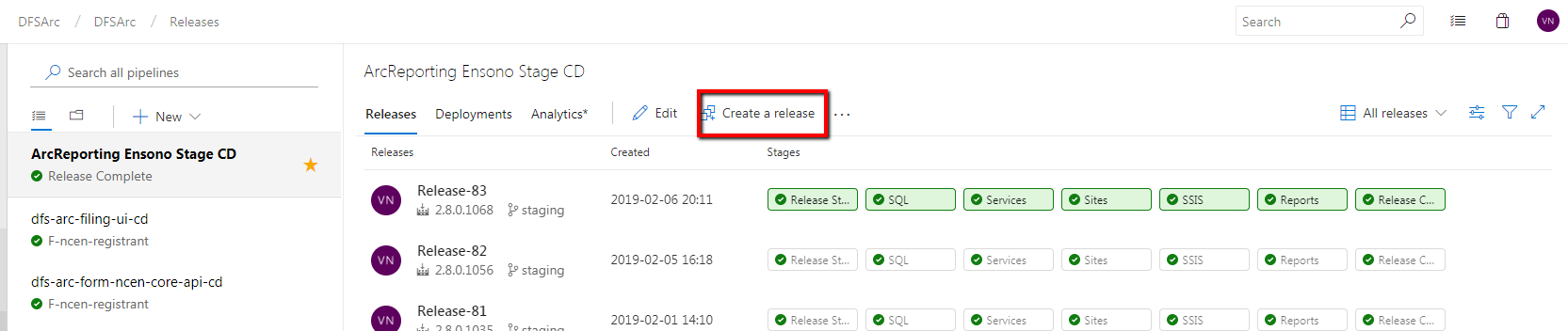


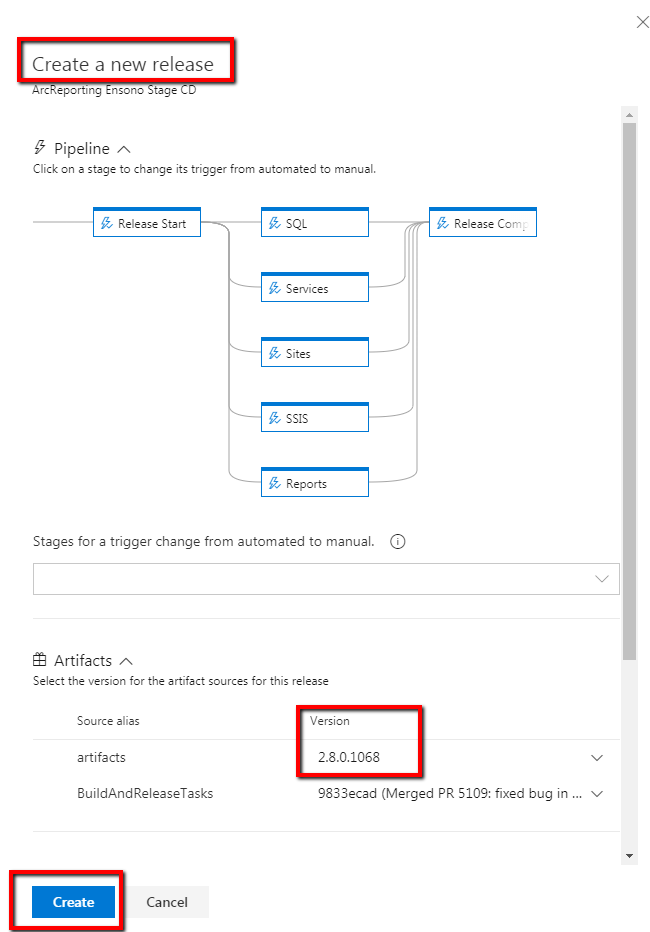
5. Trigger a stage release from AzureDev Ops

1. After the team decides to provide a stage build for the sprint, a Pull Request is created by development team to merge the code from development branch to stage branch (see above in Staging section)

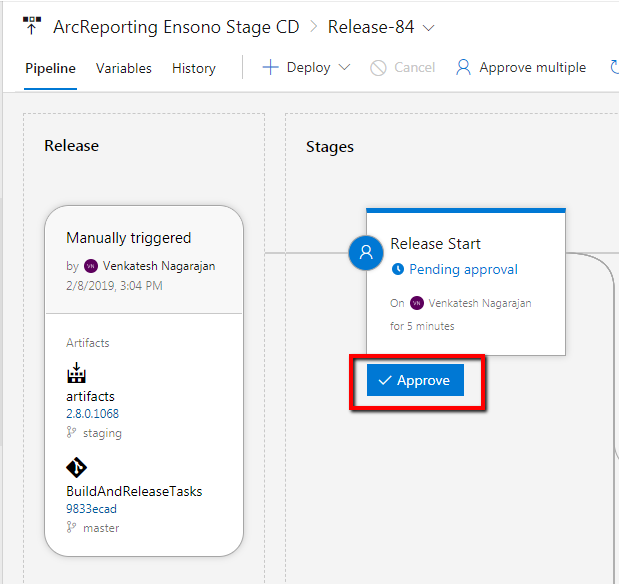


1. After the code is merged into the stage branch, development team sends out a communication to UAT that a stage build is ready
2. UAT team will create the release request with the correct stage build version in the correct pipeline in AzureDevOps





6. After the release is created successfully, UAT team will approve the release to proceed with the stage deployment. Any deployment to stage has to be approved by the UAT team.



7. UAT team will track the progress of the release through automated emails. If there are packages that fail to deploy during the release, UAT team will communicate the same to the development and infrastructure teams and re-deploy the failed packages.