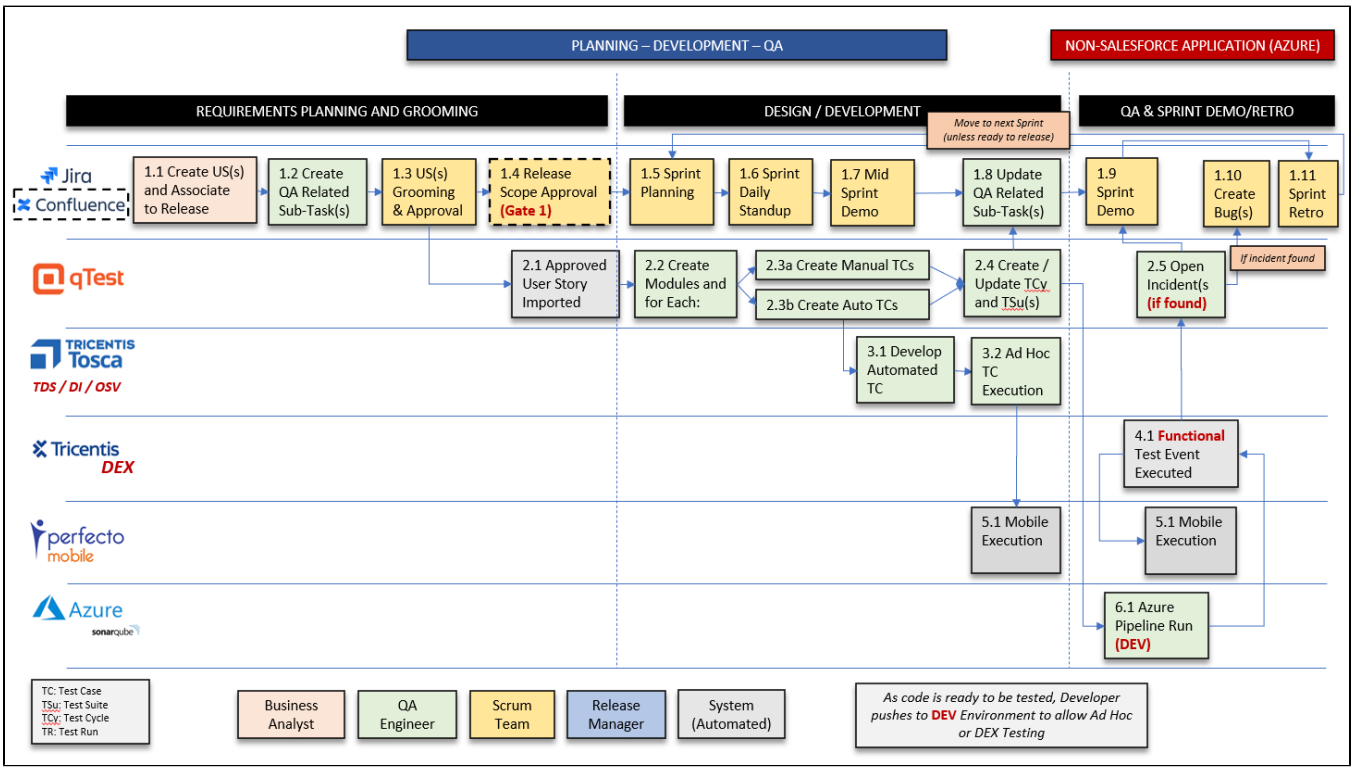
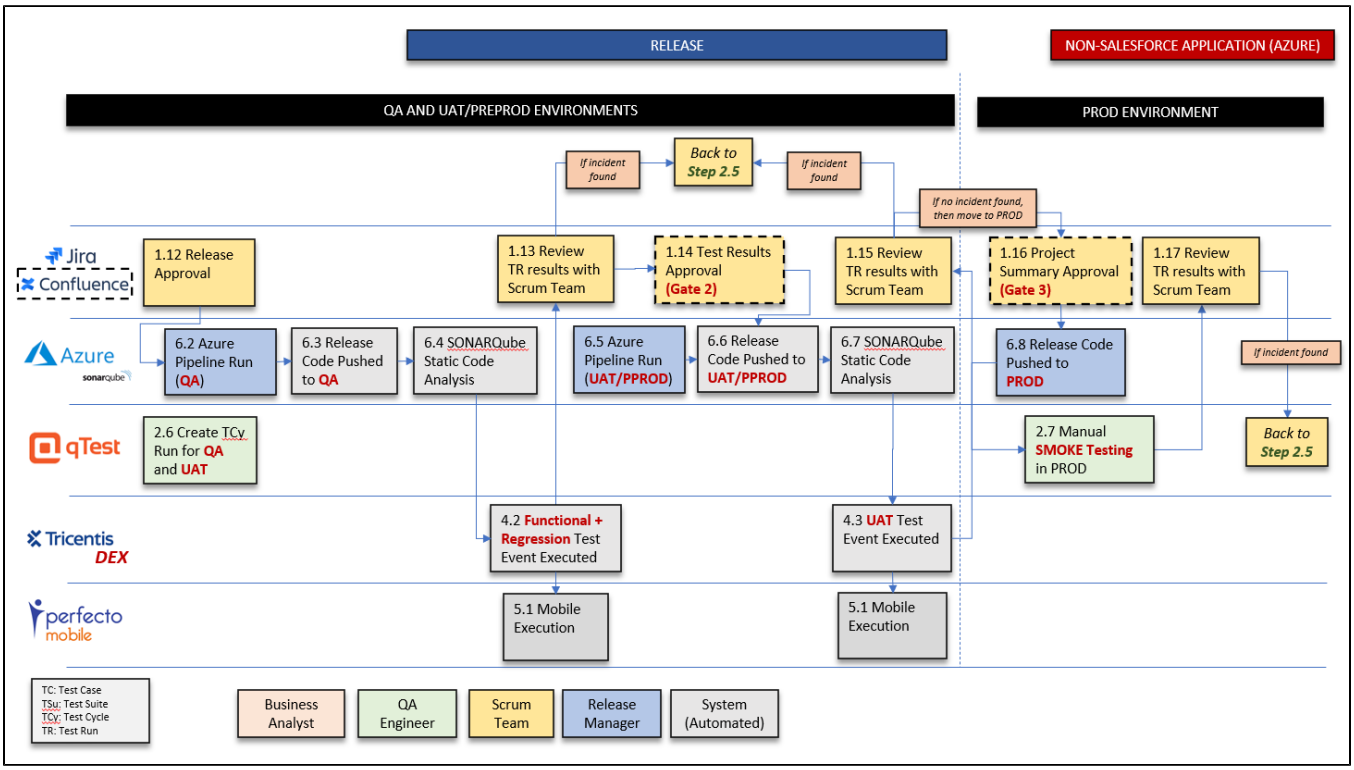
End to End DevOps - Standard





1. Jira/Confluence

ID	Step Name	Description	Owner

1.1	Create US(s) and Associate to Release	As per a previous process of demand, initial grooming/estimation and prioritization of work against the intended evolution of the product and identifying the most urgent elements to address, User Stories from the backlog are associated with a target release or new ones are created toward that release. The Business Analyst mostly takes charge of this activity. It is important that each User Story is small in size so that it can be developed, tested and marked as done in the same sprint.	Business Analyst
1.2	Create QA Related Sub-Task (s)	If a User Story qualifies for it, the QA Engineer ensures that sub-tasks related to QA activities are created under it and that the story points allocation for the QA and Manual/Automated Testing efforts are properly accounted for against the story. This sub-task will be used to track progress and ultimately completion of the QA work on that story.	QA Engineer
1.3	US(s) Grooming & Approval	Following BA and QA initial documentation of the User Story, this step allows for ensuring crystal clear understanding by all of all details around it, tweaking the documentation. This steps ends with approval against the story by the Product Owner toward development.	Scrum Team
1.4	Release Scope Approval **(Gate 1)**	As part of the End to End DevOps Agile SLC policy, a Gate 1 is required to set a clear scope on a Release and this is what happens in that step with Confluence based approval of the User Stories in scope. This is achieved by the Product Owner and/or SQA.	Scrum Team
1.5	Sprint Planning	This ceremony helps to set up the entire team for the coming sprint, creating a smooth pathway for a successful sprint. Sprint planning re quires the participation of all the scrum roles: the development team, scrum master and the product owner. The planning, of course, is prior to the sprint. It typically lasts for an hour or two. This is also the opportunity for the QA Engineer to review the User Stories selected for the sprint and get agreement on which stories will be automated.	Scrum Team
1.6	Sprint Daily Standup	This short scrum ceremony (15 minutes) makes sure that everyone knows what's happening. It's a way to ensure transparency across the team. This is not the time to dive into the weeds but rather answer three key questions: • *What did you do yesterday?* • *What will you do today?* • *Are there any blockers or impediments preventing you from doing your work?*	Scrum Team
1.7	Mid Sprint Demo	At mid-sprint (typically after a week), it is expected that these smaller user stories development is completed and a demo session is organized for the Scrum team to review the work and get full understanding of the developed functionality. The most important part is that the QA team gets this complete understanding so that manual and/or automated test scripts work can start right after.	Scrum Team
1.8	Update QA Related Sub-Task (s)	As the sprint progresses, the QA Engineer updates the QA related sub-tasks under stories to allow proper tracking of the work.	QA Engineer
1.9	Sprint Demo	After the sprint has been completed, it's time to get the team together to demo or showcase their work. Each team member reviews the newly developed features or whatever it was that they worked on during the sprint. There is no time limit on that ceremony. The outcome of the end-of-sprint manual and automated testing should also be reviewed (in addition to SONARQube reporting).	Scrum Team
1.10	Create Bug(s)	After an incident is created in qTest, a bug will be required to be added under the corresponding User Story in Jira with the below details being required: **Summary** – brief context explaining what the defect is about **Severity** – severity of the defect as it relates to the application under test **Priority** – this will enable the team to determine the sequence in which to fix defects **Type** – the type of defect (functional, technical, etc.) **Target Release** – the release in which the defect is expected to be fixed in **Environment** – the environment in which the defect was detected **Description** – this is the are where you'll enter a more detailed description of the defect such as steps to recreate, the expected result, the actual result, etc. **Attachments** – any artifacts relevant to the defect should be added. This is typically where you'll be adding screenshots.	Scrum Team
1.11	Sprint Retro	The last scrum ceremony is called the Sprint Retrospective. It occurs at the end of a sprint, after the review, and is usually an hour in duration. The retrospective includes the development team, scrum master and product owner. Because scrum is part of an agile process, it is all about change, which includes getting feedback and quickly acting on it. Scrum seeks continuous improvement and the retrospective is a method to make sure that the product and development culture is constantly improving. Feedback collected in that ceremony should be clearly documented and taken seriously with tracked follow up action items.	Scrum Team
1.12	Release Approval	Upon completion of all the Development and QA activities across one or many sprints (number of sprints to run prior to releasing to be planned as part of PI process), a green light is given to initiate the release process which will include moving and testing the code across several environments from Dev to QA to UAT to Prod (with possibly Preprod prior).	Scrum Team
1.13	Review TR results with Scrum Team	After testing against each environment (QA, UAT, Preprod and Prod), it is the opportunity for the Scrum Team to review the Test Results (Tosca, SonarQube) and any incident that might have been identified. The Dev team will have to work at addressing the identified bugs going back to step 2.5. This will definitely have an impact on the Release timeline unless some bug fixing time was pre-allocated in the process.	Scrum Team

1.14	Test Results Approval **(Gate 2)**	Following successful testing against the QA environment and as part of the End to End DevOps Agile SLC policy, a Gate 2 is required to provide formal approval on QA Testing against a Release and this is what happens in that step with Confluence based approval. This is achieved by the Product Owner and/or SQA and will be followed by promoting the code to the UAT environment.	Scrum Team
1.15	Project Summary Approval **(Gate 3)**	Following successful roll out and testing of the code to the UAT environment and as part of the End to End DevOps Agile SLC policy, a Gate 3 is required to authorize the release to Production and this is what happens in that step with Confluence based approval. This is achieved by the Product Owner and/or SQA.	Scrum Team

2. qTest

ID	Step Name	Description	Owner
2.1	Approved US(s) Imported	Jira approved user stories are automatically imported into qTest corresponding projects after a few minutes. qTest project requirements tab will be updated with these stories shown under the "Imported from Jira" folder.	Automated
2.2	Create Modules and for Each:	Modules are folder structures inside Requirements and Test Design menu. These modules contain Requirements and Test Cases in respective sections. You can define modules as a group of functionalities having the same nature of a product. Creates a plan with Releases (what we refer to as “Projects” at AbbVie) and Builds to define the testing objectives for each Release or Build and adds specific requirements to the Release and/or Build scope (project release scope). "Release" and "Build" are the names of the objects, but they can be used for other time boxes such as Sprints (for Agile projects) or different phases of testing (for waterfall projects). • **Note:** • 1. qTest releases and builds will be administered by a qTest admin. Project users will not need to create these instances. • 2. An option to add Modules for Automation Test Cases. AbbVie has decided to go with a centralized approach for managing projects within qTest. All projects will appear under a single tree node similar to the structure displayed below on the **Test Plan** module.	QA Engineer
2.3a	Create Manual TCs	Test cases are there because there is a requirement which lead to the creation of it. You may want to have the ability of writing the test cases from the requirement page itself. This will allow you to read the description of the requirement while compiling a list of tests to provide coverage all within on screen. From the test steps tab being by entering any precondition which may need to be satisfied before proceeding to execute the steps of the test case	QA Engineer
2.3b	Create Auto TCs	Test Cases can be created for Automation candidates and linked with Test Cases in Tosca Project. After automating the Test Cases in Tosca, an option is available to link the completed test cases with qTest Test Cases.	QA Engineer
2.4	Create / Update TCy and TSu(s)	For all planned test cycles, we may need to set up within the release. For example, a project may have cycles of ITC Testing, Regression Testing, UAT Testing, etc. Further, within the test cycles Test Suites need to be created. Further, the test runs need to be added within the test suites. This is nothing but adding the test cases which will be executed during each phase or round of testing.	QA Engineer
2.5	Open Incident(s **(if found)**	Click on the BUG icon in TestSteps section to create a new BUG or link an existing BUG.	QA Engineer
2.6	Create TCy Run for **QA** and **UAT**	Test Runs are where you will be running the test cases against the application under test. You can log defects and link existing defects from within the test run feature of qTest. A Test Run may be created in two ways. 1. Scheduled Run: 2. Quick Run	PQA Engineer
2.7	Manual **SMOKE** Testing in PROD	Once the build deployment is completed in Production environment, a Smoke Test is performed manually to conclude that the Application is in working state. Smoke Test is performed manually as per AbbVie guidelines.	QA Engineer

3. Tosca (+TDS/DI/OSV)

ID	Step Name	Description	Owner
3.1	Develop Automated TC	Automation Test Cases are developed in Tosca for the corresponding requirements based on the User Stories. Once the Automation Test Cases are completed and review successfully, same can be included in the ExecutionLists. ExecutionLists are then added to the TestEvents to be executed in DEX environment.	QA Engineer

3.2	Ad Hoc TC Execution	Launch the desired application to be tested and follow the below steps. 1. Select the desired test using the check box and click the drop-down menu next to Run button. Select Test Pad + Desktop Explorer. This will launch the test pad as well as Integrated Explorer window. 2. Within the integrated explorer window, the project name will be preselected with a placeholder Title for the session a user is about to record. It is recommended that the title is modified to give it a more meaningful name (eg: <Test Name>_UAT Cycle 1>). You may add a description to add additional useful information pertaining to the test cases. The application(s) under test must be selected as well. This allows explorer to know which running application. Click the Start button to initiate recording the session. 3. Continue testing the application based on your test case. The tool will record all the screens, clicks, keyboard entries automatically. 4. Click the Stop button to end the session. A new browser tab will be displayed with your recorded session. Your recorded session is available for review and is automatically linked to the test run which was initiated. Here you may add/remove content to enhance your session. This includes adding labels, highlighting any items, etc. 5. Proceed to completing the test results on test pad as normal. 6. You can access your recorded session at a later time by navigating to the test run it is attached to then clicking on the sessions tab. This will present any linked sessions to that test run.	QA Engineer

4. Tosca Distributed Execution (DEX)

ID	Step Name	Description	Owner
4.1	**Functional** Test Event Executed	Functional Test Cases that are Automated has to be added to the corresponding ExecutionLists. At the end of each sprint, all the Test Cases which are Automation candidates are triggered through TestEvents to be executed in DEX environment.	Automated
4.2	**Functional + Regression** Test Event Executed	Regression Test Cases are modified and updated according to the latest functionalities. Test Events for respective environments are configured to be triggered from Azure Pipelines. Hence, the Regression ExecutionList is added to the respective Test Event.	Automated
4.3	**UAT/PREPROD** Test Event Executed	VTP and STP protocols are followed in PREPROD environments as part of the AbbVie Testing Process.	Automated

5. Perfecto Mobile

ID	Step Name	Description	Owner
5.1	Mobile Execution	Applications which has to be tested in Mobile environment would be validated on Perfecto platform. Perfecto execution is integrated with Tosca.	Automated

6. Azure

Find the **link** for the pipeline with the below mentioned tasks.

ID	Step Name	Description	Owner
6.1	Pipeline Run Triggered (Dev)	• **CI Enabled:** When the developer pushes the code to the Azure Repo, the pipeline triggers and runs the tasks in the pipeline automatically. It hits the Dev branch first. • **Manual:** If the pipeline is enabled with parameters, the developers can manually choose the branch in which the code to be deployed.	Developer
6.2	SONARQube Static Code Analysis	• **Prepare Analysis Configuration:** Configures the required settings before executing the build. • **Run Code Analysis:** Executes the analysis of source code. • **Publish Quality Gate Result:** Displays the Quality Gate status in the build summary letting you know if your code meets quality standards for production. • **Build Breaker:** Breaks the build when the Quality gates fail.	Automated
6.3	Release Code Pushed to Dev	On the successful completion of the build task, the code gets deployed automatically	Automated
6.4	Pipeline Run Triggered (QA)	**Gated Merge:** When the approver approves the Gated Approval, the code merges to QA branch	QA Approver
6.5	Release Code Pushed to QA	The code gets deployed to QA branch after a successful build in QA	Automated
6.6	Pipeline Run Triggered (PREPROD)	**Gated Merge:** When the approver approves the Gated Approval, the code merges to PREPROD branch	PREPROD Approver
6.7	Release Code Pushed to PREPROD	The code gets deployed to PREPROD, after a successful build in PREPROD	Automated
6.8	Pipeline Run Triggered (PROD)	**Gated Merge:** When the approver approves the Gated Approval, the code merges to the Master branch	PROD Approver
6.9	Release Code Pushed to PROD Environment	The code gets deployed to PROD Environment	Automated