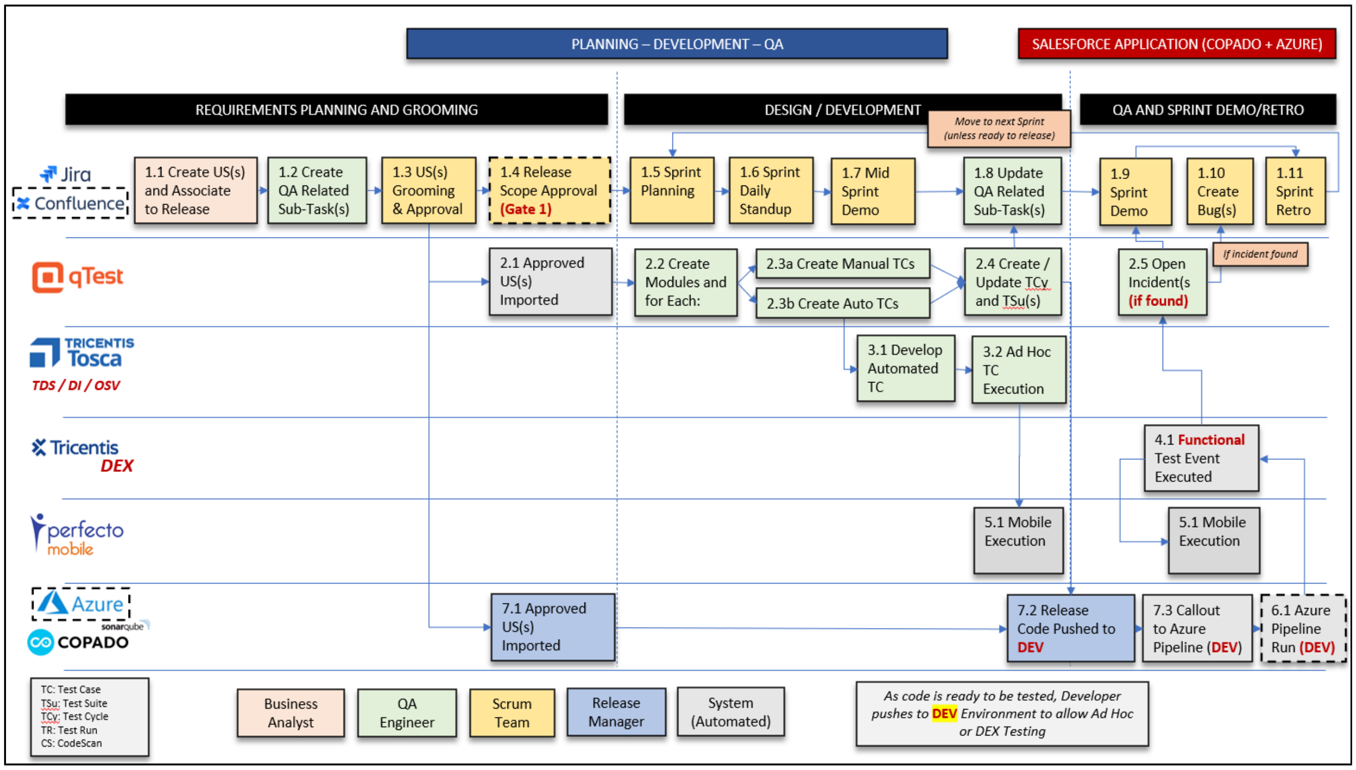
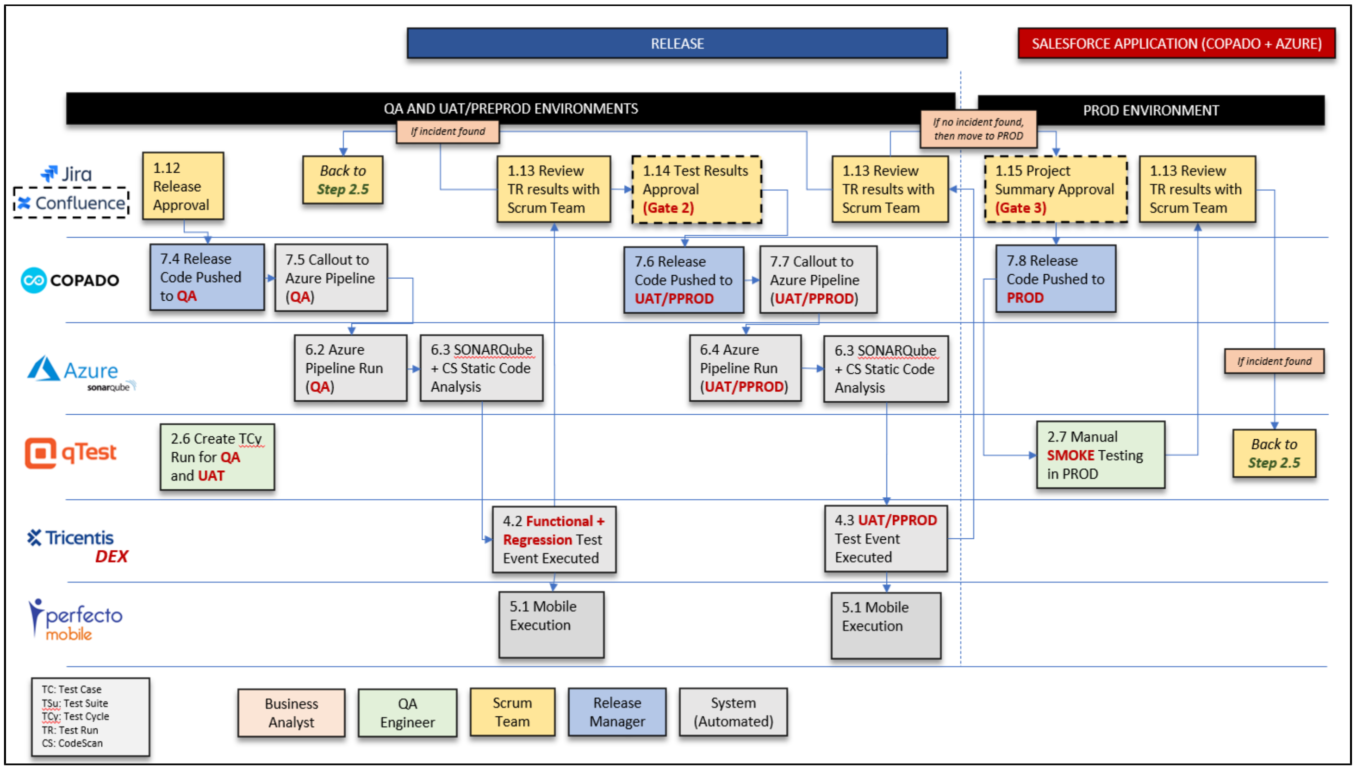
End to End DevOps - SalesForce


PLANNING – DEVELOPMENT – QA
SALESFORCE APPLICATION (COPADO + AZURE)
REQUIREMENTS PLANNING AND GROOMING
DESIGN / DEVELOPMENT
QA AND SPRINT DEMO/RETRO
Move to next Sprint (unless ready to release)
Jira
Confluence
qTest
TRICENTIS Tosca
TDS / DI / OSV
Tricentis DEX
perfecto mobile
Azure
sonarqube
COPADO
1.1 Create US(s) and Associate to Release
1.2 Create QA Related Sub-Task(s)
1.3 US(s) Grooming & Approval
1.4 Release Scope Approval (Gate 1)
1.5 Sprint Planning
1.6 Sprint Daily Standup
1.7 Mid Sprint Demo
1.8 Update QA Related Sub-Task(s)
1.9 Sprint Demo
1.10 Create Bug(s)
1.11 Sprint Retro
2.1 Approved US(s) Imported
2.2 Create Modules and for Each:
2.3a Create Manual TCs
2.3b Create Auto TCs
2.4 Create / Update TCy and TSu(s)
2.5 Open Incident(s (if found)
If incident found
3.1 Develop Automated TC
3.2 Ad Hoc TC Execution
4.1 Functional Test Event Executed
5.1 Mobile Execution
5.1 Mobile Execution
7.1 Approved US(s) Imported
7.2 Release Code Pushed to DEV
7.3 Callout to Azure Pipeline (DEV)
6.1 Azure Pipeline Run (DEV)
TC: Test Case
TSu: Test Suite
TCy: Test Cycle
TR: Test Run
CS: CodeScan
Business Analyst
QA Engineer
Scrum Team
Release Manager
System (Automated)
As code is ready to be tested, Developer pushes to DEV Environment to allow Ad Hoc or DEX Testing


RELEASE
SALESFORCE APPLICATION (COPADO + AZURE)
QA AND UAT/PREPROD ENVIRONMENTS
PROD ENVIRONMENT
If incident found
If no incident found, then move to PROD
Jira
Confluence
COPADO
Azure
sonarqube
qTest
Tricentis DEX
perfecto mobile
1.12 Release Approval
Back to Step 2.5
1.13 Review TR results with Scrum Team
1.14 Test Results Approval (Gate 2)
1.13 Review TR results with Scrum Team
1.15 Project Summary Approval (Gate 3)
1.13 Review TR results with Scrum Team
7.4 Release Code Pushed to QA
7.5 Callout to Azure Pipeline (QA)
7.6 Release Code Pushed to UAT/PPROD
7.7 Callout to Azure Pipeline (UAT/PPROD)
7.8 Release Code Pushed to PROD
6.2 Azure Pipeline Run (QA)
6.3 SONARQube + CS Static Code Analysis
6.4 Azure Pipeline Run (UAT/PPROD)
6.3 SONARQube + CS Static Code Analysis
If incident found
2.6 Create TCy Run for QA and UAT
2.7 Manual SMOKE Testing in PROD
Back to Step 2.5
4.2 Functional + Regression Test Event Executed
4.3 UAT/PPROD Test Event Executed
5.1 Mobile Execution
5.1 Mobile Execution
TC: Test Case
TSu: Test Suite
TCy: Test Cycle
TR: Test Run
CS: CodeScan
Business Analyst
QA Engineer
Scrum Team
Release Manager
System (Automated)

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. Details of the steps are available at 1. How to import User Stories from Jira to qTest	Automated
2.2	Create Modules and for Each:	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. More details are available at 2. Create Modules and for Each:	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 Manual Test Cases creation can be done by following the details at 3a. Create Manual TCs	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. Automation Test Cases creation can be done by following the details at 3b. Create Auto TCs	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. Detailed steps can be found at 4. Create / Update Test Cycle(s) and Test Suite(s)	QA Engineer
2.5	Open Incident(s **(if found)**	During the Test Execution, if any defect is found, click on the BUG icon in TestSteps section under "Test Execution" section to create a new BUG or link an existing BUG. BUGs can ideally be raised during the Test Runs. Under some circumstances, we can raise the BUG from Defects module. Steps to create a BUG during Test Run can be found at 5. Open Incident(s) (if found)	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 Details of steps are available at 6. Create Test Run for QA and UAT	QA Engineer
2.7	Manual **S MOKE** Te sting 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. Details are available at 7. Manual SMOKE Testing in PROD	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 which are then added to the TestEvents to be executed in DEX environment.	QA Engineer
3.2	Ad Hoc TC Execution	Adhoc Test Execution is performed to find out all the bugs and inconsistencies corresponding to the application under test. Details of steps are available at : How to perform Adhoc Test Execution	QA Engineer
3.3	Generate Pre Approval Report	After a Test Case has been developed and a dry run has been tested in the INT environment the QA Engineer will generate a Test Case Overview or "Pre-Approval Report" for the SQA team. The reports are uploaded to their respective places in sharepoint. https://abbvie.sharepoint.com/teams/Ext_OUSDEVOPS/Automated%20Testing/ Details of current procedures are available at How to Generate Pre Approval Report	QA Engineer
3.4	Generate Execution Report	After the VTP Test Case Pre-Approval Report has been approved in SNOW, the QA Engineer runs a formal execution of the Test Case in the QA environment. The results are held temporarily in the Execution Lists in Tosca. The QA Engineer then needs to print a Execution Report with both failed/passed test cases and include screenshots. The report to print is named "ExecutionEntries with detailed Logs_Used Value". Details of the current procedures are available at How To Generate Execution Reports **Note:** Both pre approval & post test execution reports will be generated for STP tests too, but these reports are **mandatory for VTP** Test Cases. Hence the brackets: (VTP Only)	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. To learn about how to run automated tests in Perfecto click here To learn about how to run manual tests in Perfecto click here	Automated

6. Azure Pipeline

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

ID	Step Name	Description	Owner
6.1	Azure Pipeline Run **(DEV)**	Azure Pipeline triggers automatically through the Copado callout with Dev Org as a parameter. The pipeline has the SonarQube Codescan along with the Tosca Tests tasks configured w.r.t Org ID. Once the tests are completed, results will be published to the Output Console.	Automated
6.2	Azure Pipeline Run (**QA**)	Azure Pipeline triggers automatically through the Copado callout with QA Org as a parameter. The pipeline has the SonarQube Codescan along with the Tosca Tests tasks configured w.r.t Org ID. Once the tests are completed, results will be published to the Output Console.	Automated

6.3	SONARQube + CS 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
6.4	Azure Pipeline Run (**UAT /PPROD**)	Azure Pipeline triggers automatically through the Copado callout with PREPROD Org as a parameter. The pipeline has the SonarQube Codescan along with the Tosca Tests tasks configured w.r.t Org ID. Once the tests are completed, results will be published to the Output Console.	Automated

7. Copado

ID	Step Name	Description	Owner
7.1	Approved US(s) Imported	The User Stories that are Approved as part of the Release in JIRA will be exported from JIRA and imported to Copado. **Opportunity:** The initial implementation will have this process executed manually, targeting to be automated in a subsequent Release. Details of the current implementation is found in this Document: **Manual Import of User Stories to Copado**	Release Manager
7.2	Release Code Pushed to **DEV**	The User Stories are deployed to DEV using Copado. All User Stories related to the Release are bundled together and all checks completes. This bundle is deployed to DEV by checking the '**Ready to Promote'** checkbox. Details of the implementation is found in this Document: **Salesforce Deployment - OneCRM Design**	Release Manager
7.3	Callout to Azure Pipeline **(DEV)**	URL Callout Step is added to Deployments Automatically. This will execute SCA and Automated Testing. This step will execute after Deployment Step to trigger Azure Pipeline by passing DEV as a parameter **Opportunity:** Targeting to move the implementation from URL Callout to using 'Salesforce Flow' or 'Copado Functions'. Details of the current implementation is found in this Document: **Execution of Automated Testing via Azure Pipeline**	Automated
7.4	Release Code Pushed to **QA**	The User Stories are deployed to QA using Copado. All User Stories related to the Release are bundled together and all checks completes. This bundle is deployed to DEV by checking the '**Ready to Promote'** checkbox. Details of the implementation is found in this Document: **Salesforce Deployment - OneCRM Design**	Release Manager
7.5	Callout to Azure Pipeline **(QA)**	URL Callout Step is added to Deployments Automatically. This will execute SCA and Automated Testing. This step will execute after Deployment Step to trigger Azure Pipeline by passing QA as a parameter **Opportunity:** Targeting to move the implementation from URL Callout to using 'Salesforce Flow' or 'Copado Functions'. Details of the current implementation is found in this Document:: **Execution of Automated Testing via Azure Pipeline**	Automated
7.6	Release Code Pushed to **UAT /PPROD**	The User Stories are deployed to PREPROD using Copado. All User Stories related to the Release are bundled together and all checks completes. This bundle is deployed to PREPROD by checking the '**Ready to Promote'** checkbox. Details of the implementation is found in this Document: **Salesforce Deployment - OneCRM Design**	Release Manager
7.7	Callout to Azure Pipeline (**UAT/PPROD**)	URL Callout Step is added to Deployments Automatically. This will execute SCA and Automated Testing. This step will execute after Deployment Step to trigger Azure Pipeline by passing PREPROD as a parameter **Opportunity:** Targeting to move the implementation from URL Callout to using 'Salesforce Flow' or 'Copado Functions'. Details of the current implementation is found in this Document: **Execution of Automated Testing via Azure Pipeline**	Automated
7.8	Release Code Pushed to **PROD**	The User Stories are deployed to PROD using Copado. All User Stories related to the Release are bundled together and all checks completes. This bundle is deployed to PROD by checking the '**Ready to Promote'** checkbox. Details of the implementation is found in this Document: **Salesforce Deployment - OneCRM Design**	Release Manager