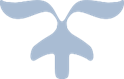


VSTS

Guide for Visual Studio Team Services



|  |  |  |  |
| --- | --- | --- | --- |
| **Document Revision History** | | | |
| **Version No.** | **Author** | **Revision Comments** | **Date** |
| 1.0 | Navpreet | Draft Version |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1.2 Define user stories 5](#_Toc506454596)

[1.3 Track progress 8](#_Toc506454597)

[1.4 Agile workflow states 9](#_Toc506454598)

[1.5 Update status with Kanban or task boards 10](#_Toc506454599)

[1.6 Map user stories to features 10](#_Toc506454600)

[1.7 Define tasks 11](#_Toc506454601)

[1.8 Track test progress 13](#_Toc506454602)

[1.9 General Information 17](#_Toc506454604)

[1.10 Track business value 17](#_Toc506454606)

[1.11 Backlog list order 17](#_Toc506454607)

[1.12 List work items 18](#_Toc506454608)

[2.1 Consider using Release Management if? 20](#_Toc506454609)

[2.2 How does Release Management work? 20](#_Toc506454609)

[2.3 Environments in Release Management 21](#_Toc506454610)

[2.4 Artifact sources 24](#_Toc506454612)

[2.5 Team Build 25](#_Toc506454613)

[2.6 Approvals and gates overview 26](#_Toc506454614)

[2.7 Build Automation 26](#_Toc506454616)

[3.1 To create a branch 28](#_Toc506454617)

[4.1 Manage documents and document libraries…………………………………………………… 29](#_Toc506454618)

5.1 How to Request for a code review  [30](#_Toc506454619)

#### **Introduction: -**

#### Visual studio team services or visual studio online or team foundation server are same thing with different names not just a version controlling system like GIT-HUB, SVN tools but it is helping in whole software development life cycle (SDLC) for a project in various stages like in requirement phase, design phase, development phase, testing phase, release and deployment phase and finally maintenance phase as well. In general, Vestas follows standard process and Microsoft guidelines. These are following features you need to be familiar to utilize VSO in my work page.

#### **Detail designing Phase**

* 1. **Agile process work item types and workflow**

#### To gain insight into a portfolio of features, scenarios, or user experiences, product owners and program managers can map user stories to features. When teams work in sprints, they define tasks which automatically link to user stories.

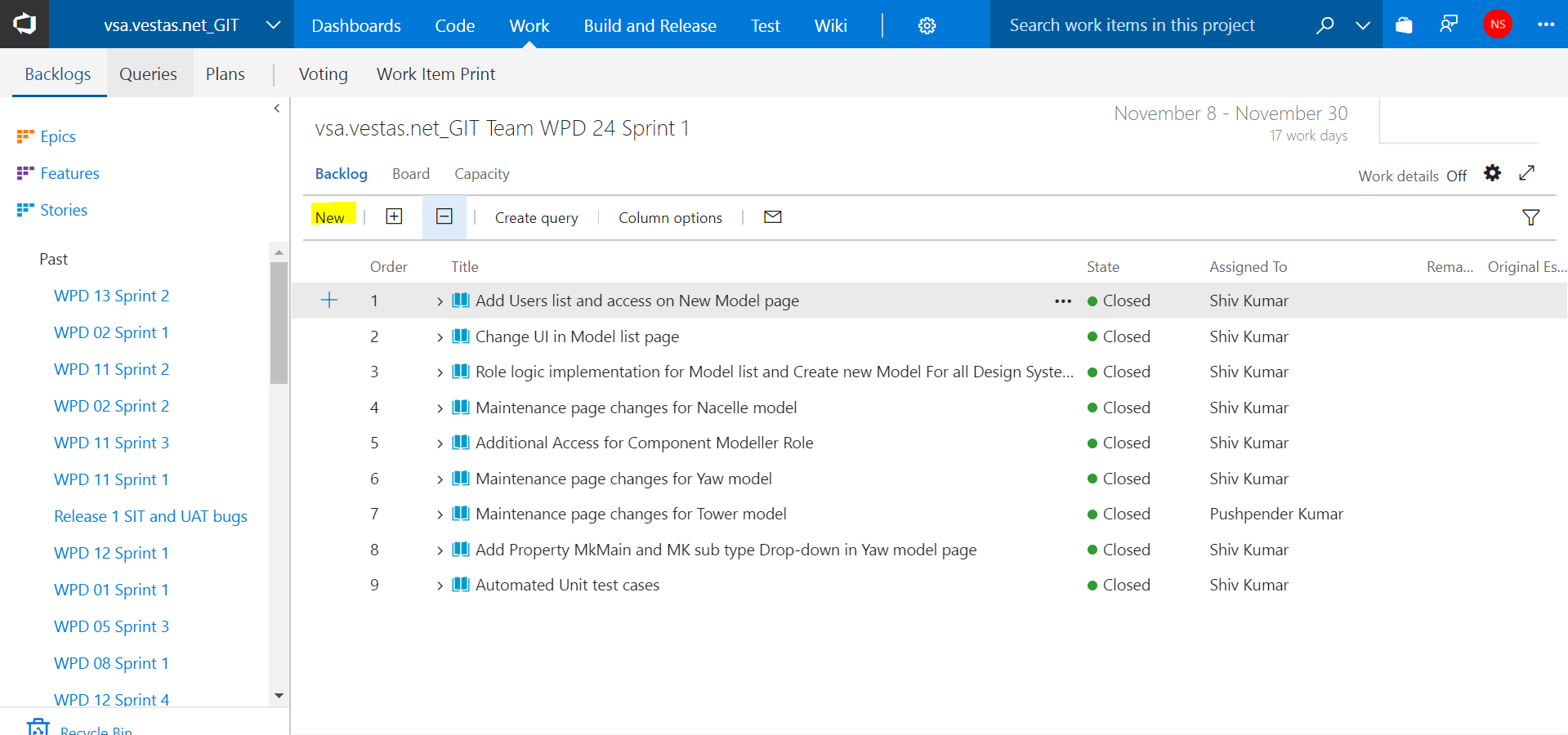


Using the web portal or Microsoft Test Manager, testers can create and run test cases. Bugs and issues are used to track code defects and blocking issues.

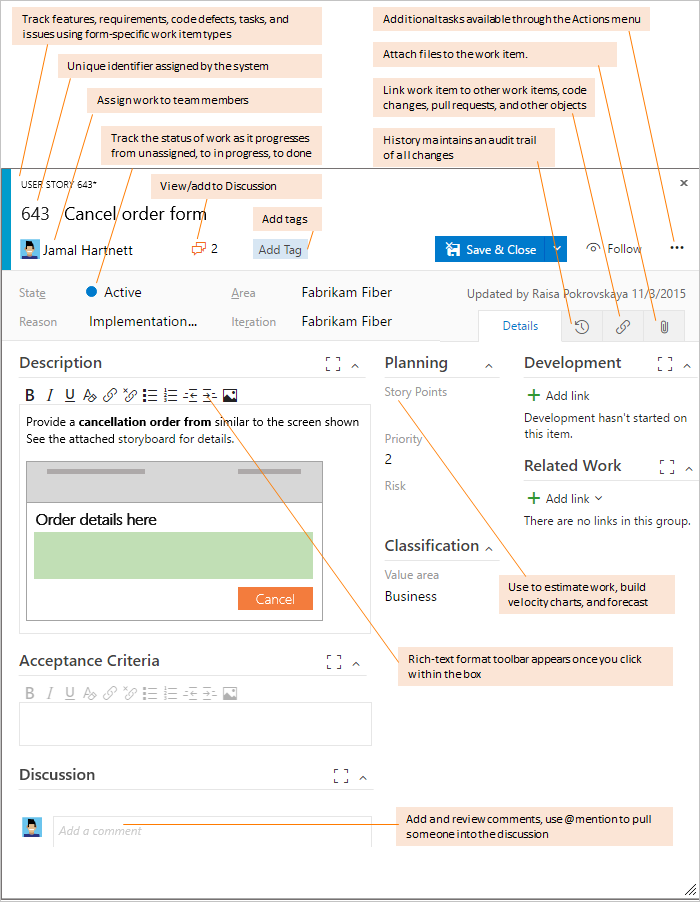
## **1.2 Define user stories**

User stories define the applications, requirements, and elements that teams need to create. Product owners typically define and stack rank user stories. The team then estimates the effort and work to deliver the highest priority items.

Create user stories from the quick add panel on the [product backlog page](https://docs.microsoft.com/en-us/vsts/work/backlogs/create-your-backlog). From that page, you can also drag-and-drop items to reorder them or [map them to features](https://docs.microsoft.com/en-us/vsts/work/backlogs/organize-backlog).



Later, you can open each user story to provide more details and estimate the story points.



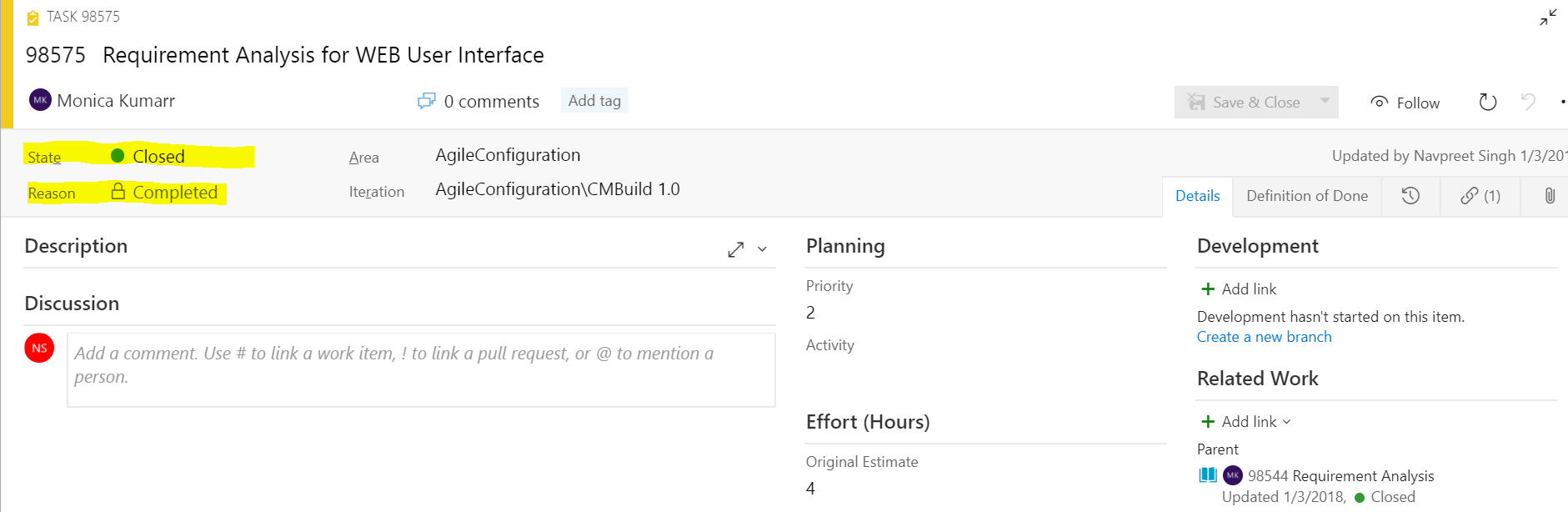
By defining the **Story Points**, teams can use the forecast feature and velocity charts to estimate future sprints or work efforts. By prioritizing the user stories on the backlog page (which is captured in the Stack Rank field), product owners can indicate which items should be given higher priority.

Use the following guidance and that provided for [fields used in common across work item types](https://docs.microsoft.com/en-us/vsts/work/work-items/guidance/agile-process-workflow#definitions-in-common) when filling out the form.

| **Field/tab** | **Usage** |
| --- | --- |
| Description | For user stories, provide enough detail for estimating how much work will be required to implement the story. Focus on who the feature is for, what users want to accomplish, and why. Don't describe how the feature should be developed. Do provide sufficient details so that your team can write tasks and test cases to implement the item. |
| Acceptance Criteria | Provide the criteria to be met before the bug or user story can be closed. Before work begins, describe the customer acceptance criteria as clearly as possible. Conversations between the team and customers to define the acceptance criteria will help ensure that your team understands your customers' expectations. The acceptance criteria can be used as the basis for acceptance tests so that you can more effectively evaluate whether an item has been satisfactorily completed. |
| Value Area | The area of customer value addressed by the epic, feature, requirement, or backlog item. Values include:   * **Architectural:** Technical services to implement business features that deliver solution * **Business**: Services that fulfill customers or stakeholder needs that directly deliver customer value to support the business (Default) |
| Story Points | Estimate the amount of work required to complete a user story using any numeric unit of measurement your team prefers.  Agile velocity charts and forecast tools reference the values in this field. For additional guidance, see the Estimating white paper. |
| Priority | A subjective rating of the user story, feature, or requirement as it relates to the business. Allowed values are:   * **1**: Product cannot ship without the feature. * **2**: Product cannot ship without the feature, but it doesn't have to be addressed immediately. * **3**: Implementation of the feature is optional based on resources, time, and risk. |
| Risk | A subjective rating of the relative uncertainty around the successful completion of a user story. Allowed values are:   * **1 - High** * **2 - Medium** * **3 - Low** |

## **1.3 Track progress**

As work progresses, you change the State field to update the status. Optionally, you can specify a reason. The state and reason fields appear on the work item form in the header area.



### **1.4 Agile workflow states**

By updating the workflow, teams know which items are new, in progress, or completed. These diagrams show the main progression and regression states of the user story, bug, and task.

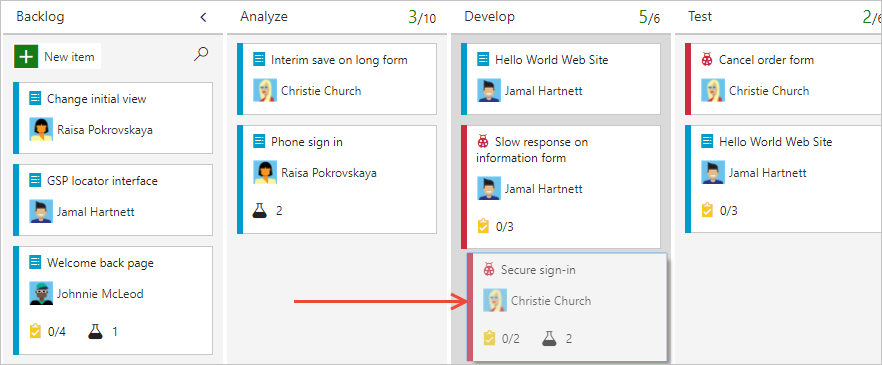


A typical workflow progression for a user story follows:

* The product owner creates a user story in the New state with the default reason, New user story
* The team updates the status to Active when they decide to complete the work during the sprint
* A user story is moved to Resolved when the team has completed all its associated tasks and unit tests for the story pass
* A user story is moved to the Closed state when the product owner agrees that the story has been implemented according to the Acceptance Criteria and acceptance tests pass.

### **1.5 Update status with Kanban or task boards**

Teams can use the [Kanban board](https://docs.microsoft.com/en-us/vsts/work/kanban/kanban-basics) to update the status of requirements, and the [sprint task board](https://docs.microsoft.com/en-us/vsts/work/scrum/task-board) to update the status of tasks. Dragging items to a new state column updates both the State and Reason fields.



You can customize the Kanban board to support additional [swim lanes](https://docs.microsoft.com/en-us/vsts/work/kanban/expedite-work) or [columns](https://docs.microsoft.com/en-us/vsts/work/kanban/add-columns). For additional customization options.

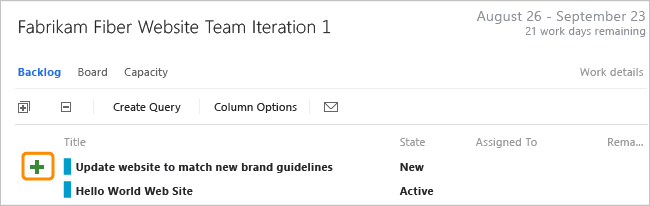
## **1.6 Map user stories to features**

When you manage a suite of products or user experiences, you might want to view the scope and progress of work across the product portfolio. You can do this by [defining features](https://docs.microsoft.com/en-us/vsts/work/backlogs/define-features-epics) and [mapping user stories to features](https://docs.microsoft.com/en-us/vsts/work/backlogs/organize-backlog).

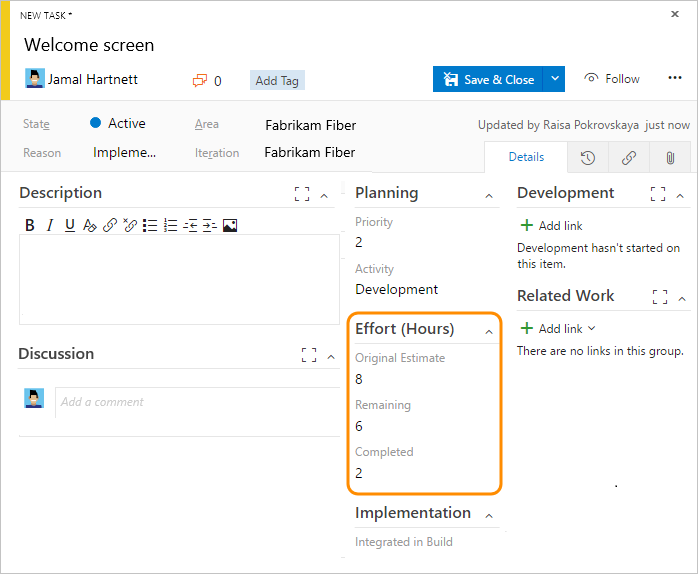
Using portfolio backlogs, you can [drill down from one backlog to another](https://docs.microsoft.com/en-us/vsts/work/scale/portfolio-management) to view the level of detail you want. Also, you can use portfolio backlogs to view a rollup of work in progress across several teams when you [setup a hierarchy of teams](https://docs.microsoft.com/en-us/vsts/work/scale/multiple-teams).

## **1.7 Define tasks**

When your team manages their work in sprints, they can use the [sprint backlog page](https://docs.microsoft.com/en-us/vsts/work/scrum/sprint-planning) to break down the work to be accomplished into distinct tasks.



Name the task and estimate the work it will take.



Using Agile processes, teams forecast work and define tasks at the start of each sprint, and each team member performs a subset of those tasks. Tasks can include development, testing, and other kinds of work. For example, a developer can define tasks to implement user stories, and a tester can define tasks to write and run test cases.

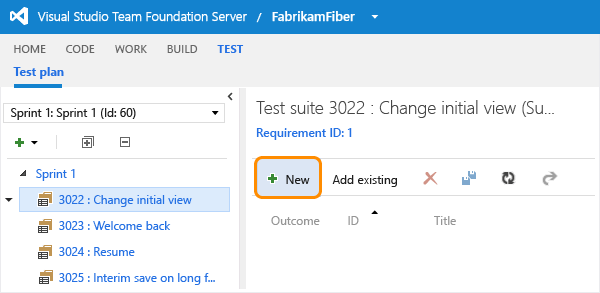
When teams estimate work using hours or days, they define tasks and the **Remaining Work** and **Activity** (optional) fields.

| Field/tab | Usage |
| --- | --- |
| [Original Estimate](https://docs.microsoft.com/en-us/vsts/work/track/query-numeric) | The amount of estimated work required to complete a task. Typically, this field doesn't change after it is assigned.  You can specify work in hours or in days. There are no inherent time units associated with this field. |
| [Remaining Work](https://docs.microsoft.com/en-us/vsts/work/track/query-numeric) | The amount of work remaining to complete a task. As work progresses, update this field. It's used to calculate [capacity charts](https://docs.microsoft.com/en-us/vsts/work/scale/capacity-planning), the [sprint burndown chart](https://docs.microsoft.com/en-us/vsts/work/scrum/sprint-burndown), and the following (TFS only) reports: [Burndown and Burn Rate](https://msdn.microsoft.com/library/dd380678.aspx), [Remaining Work](https://msdn.microsoft.com/library/dd380673.aspx), and [Status on All Iterations](https://msdn.microsoft.com/library/dd380706.aspx).  If you divide a task into subtasks, specify hours for the subtasks only. You can specify work in any unit of measurement your team chooses. |
| [Completed Work](https://docs.microsoft.com/en-us/vsts/work/track/query-numeric) | The amount of work spent implementing a task.1 |
| [Activity](https://docs.microsoft.com/en-us/vsts/work/track/query-numeric) | Select the type of activity this task represents when your team estimates sprint capacity by activity. |
| [Integrated in Build](https://docs.microsoft.com/en-us/vsts/work/track/build-test-integration) | Product build number that incorporates the code or fixes a bug. |

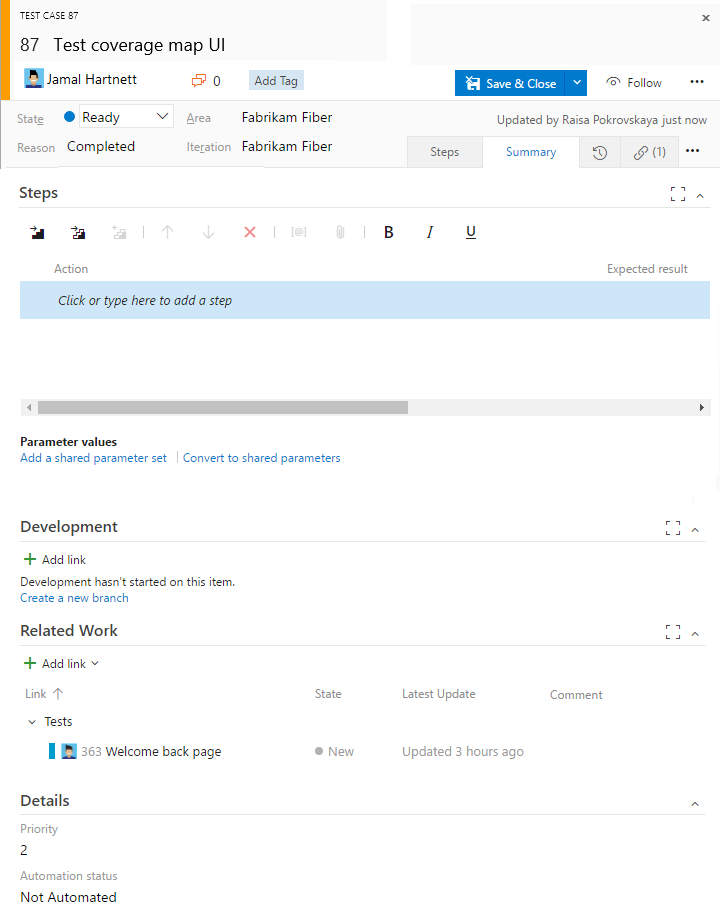
If you use [Microsoft Project](https://docs.microsoft.com/en-us/vsts/work/backlogs/office/create-your-backlog-tasks-using-project) to assign resources and track a schedule, you can update these fields using Project.

## **1.8 Track test progress**

From the web portal or Test Manager, you can [create test cases that automatically link to a user story or bug](https://docs.microsoft.com/en-us/vsts/manual-test/getting-started/create-test-cases). Or, you can link a user story to a test case from the (links tab).



The test case contains a number of fields, many of which are automated and integrated with Test Manager and the build process. For a description of each field, see [Query based on build and test integration fields](https://docs.microsoft.com/en-us/vsts/work/track/build-test-integration).



The (links tab) captures the links to user stories and bugs in a test case. By linking user stories and bugs to test cases, the team can track the progress made in testing each item. By defining these links, you support information that appears in the [Stories Overview Report](https://docs.microsoft.com/en-us/vsts/report/sql-reports/stories-overview-report-agile) report.

### Track code defects

You can [create bugs from the web portal web portal, Visual Studio, or when testing with Test Manager](https://docs.microsoft.com/en-us/vsts/work/backlogs/manage-bugs).

The following fields and tabs appear in most work items. Each tab is used to track specific information, such as history, links, or attachments. These three tabs provide a history of changes, view of linked work items, and ability to view and attach files.

The only required field for all WITs is **Title**. When the work item is saved, the system assigns it a unique **ID**. The form highlights required field in yellow. For information about other fields, see [Work item field index](https://docs.microsoft.com/en-us/vsts/work/work-items/guidance/work-item-field).

| **Field/tab** | **Usage** |
| --- | --- |
| [Title](https://docs.microsoft.com/en-us/vsts/work/track/titles-ids-descriptions) | Enter a description of 255 characters or less. You can always modify the title later. |
| [Assigned To](https://docs.microsoft.com/en-us/vsts/work/track/query-by-workflow-changes) | Assign the work item to the team member responsible for performing the work. Depending on the context you are working in, the drop-down menu will list only team members or contributors to the team project. |
| [State](https://docs.microsoft.com/en-us/vsts/work/track/query-by-workflow-changes) | When the work item is created, the State defaults to the first state in the workflow. As work progresses, update it to reflect the current state. |
| [Reason](https://docs.microsoft.com/en-us/vsts/work/track/query-by-workflow-changes) | Use the default first. Update it when you change state. Each State is associated with a default reason. |
| [Area](https://docs.microsoft.com/en-us/vsts/work/customize/set-area-paths) | Choose the area path associated with the product or team, or leave blank until assigned during a planning meeting.  To change the dropdown list of areas, see [Add and modify area and iteration paths](https://docs.microsoft.com/en-us/vsts/work/customize/set-area-paths). |
| [Iteration](https://docs.microsoft.com/en-us/vsts/work/customize/set-area-paths) | Choose the sprint or iteration in which the work is to be completed, or leave it blank and assign it later, during a planning meeting.  To change the drop-down list of iterations, see [Add and modify area and iteration paths](https://docs.microsoft.com/en-us/vsts/work/customize/set-area-paths). |
| [(History)](https://docs.microsoft.com/en-us/vsts/work/track/history-and-auditing) | Review the audit trail that the system captures and capture additional information.  Every time that the work item is updated, information is appended to the history. History includes the date of the change, who made the change, and which fields were changed. You can also add formatted text to the history field. |
| [(Links)](https://docs.microsoft.com/en-us/vsts/work/track/link-work-items-support-traceability) | Add all types of links, such as hyperlinks, change sets, source files, and so on.  This tab also lists all links defined for the work item. |
| [(Attachments)](https://docs.microsoft.com/en-us/vsts/work/track/linking-attachments) | Share more detailed information by adding files to the work item, such as email threads, documents, images, log files, or other file types. |

## **1.9 General Information**

Before you start tracking work, you must have a team project. To create one hosted in the cloud, see [Sign up for VSTS](https://docs.microsoft.com/en-us/vsts/user-guide/sign-up-invite-teammates). To create one hosted on an on-premises TFS, see [Create a team project](https://docs.microsoft.com/en-us/vsts/accounts/create-team-project).

If you have a team project, start tracking work:

* [Add work items to manage a project](https://docs.microsoft.com/en-us/vsts/work/backlogs/add-work-items) - to gain more familiarity with the work item form features
* [Create a backlog](https://docs.microsoft.com/en-us/vsts/work/backlogs/create-your-backlog) - to develop your product backlog
* [Plan a sprint](https://docs.microsoft.com/en-us/vsts/work/scrum/sprint-planning) - to start working in Scrum
* [Kanban](https://docs.microsoft.com/en-us/vsts/work/kanban/kanban-basics) - to start working in Kanban
* [Excel](https://docs.microsoft.com/en-us/vsts/work/backlogs/office/bulk-add-modify-work-items-excel) or [Project](https://docs.microsoft.com/en-us/vsts/work/backlogs/office/create-your-backlog-tasks-using-project) - to create a work breakdown structure

### 

Issues are used to track events that may block progress or shipping a user story. Bugs, on the other hand, are used to track code defects.

Work items you add from the widget are automatically scoped to your [team's default area and iteration paths](https://docs.microsoft.com/en-us/vsts/work/scale/set-team-defaults). To change the team context, see [Switch team context](https://docs.microsoft.com/en-us/vsts/teams/switch-team-context?toc=/vsts/work/scale/toc.json&bc=/vsts/work/scale/breadcrumb/toc.json).

### 1.10 Track business value

You can use the Priority field to differentiate the value of various stories. Or, you can add a custom field to the User Story WIT that tracks the relative value of stories. To learn how, see [Customize a field for a process](https://docs.microsoft.com/en-us/vsts/work/customize/process/customize-process-field).

### **1.11 Backlog list order**

The [Stack Rank](https://docs.microsoft.com/en-us/vsts/work/track/planning-ranking-priorities) field is used to track the relative ranking of user stories, however by default it doesn't appear on the work item form. The sequence of items on the backlog page is determined according to where you have [added the items or moved the items on the page](https://docs.microsoft.com/en-us/vsts/work/backlogs/create-your-backlog#move-items-priority-order). As you drag items, a background process updates this field.

## **1.12 List work items**

You can use shared work item queries to list work items based on their type, such as change requests, bugs, tasks, and requirements.

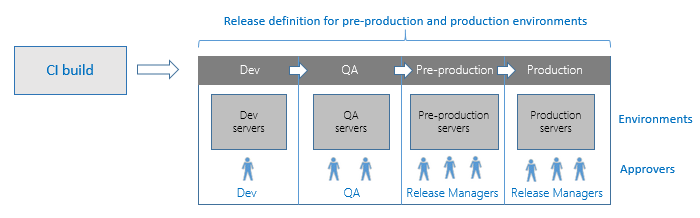


**For Bulk Edit**

<https://docs.microsoft.com/en-us/vsts/work/backlogs/office/bulk-add-modify-work-items-excel?toc=/vsts/work/work-items/toc.json>

1. **Release Management**

**Release Management** is a service in Visual Studio Team Services (VSTS) and Team Foundation Server (TFS 2015.2 and later) and an essential element of DevOps that helps your team **continuously deliver** software to your customers at a faster pace and with lower risk. You can **fully automate** the testing and delivery of your software in multiple environments all the way to production, or set up semi-automated processes with **approvals** and **on-demand deployments**.

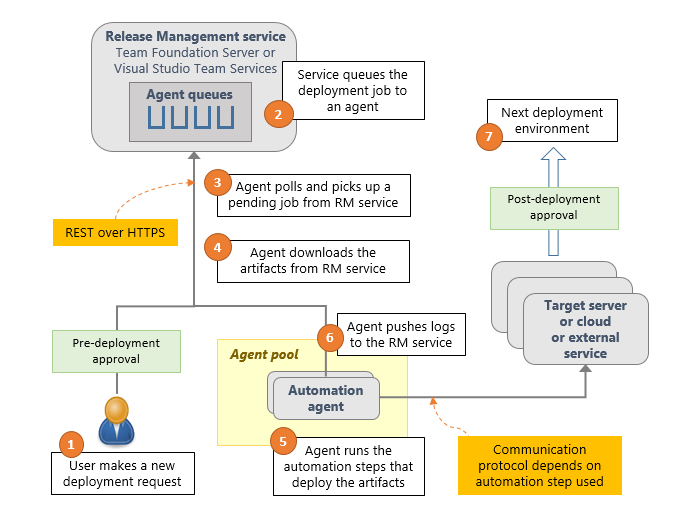


**2.1 Consider using Release Management if:**

* **You develop applications and need to deploy them regularly to any platform,** public or private cloud services, or App stores. Release Management has many out-of-the-box tasks to deploy a variety of applications. If you cannot find an out-of-the-box task to deploy your application using Release Management, consider this: if you can script the deployment of your application using Shell scripts or PowerShell scripts, utilities such as Ant or Maven, batch files or EXE utilities, then you can deploy it using Release Management. Release Management also integrates with third party deployment systems such as Chef and Docker.
* **You use a continuous integration (CI) system** and are looking for a fully-fledged continuous delivery or release management system. Whether you use Team Build from VSTS or TFS, or Jenkins as your CI system, you can set up Release Management to automatically deploy new builds to multiple environments. Even if we do not yet support integration with your favorite CI system or artifact repository, you can still write custom tasks to download and deploy artifacts from it.
* **You need to track the progress of releases.** If you use several environments for your tests, Release Management helps you monitor whether a release has been deployed and tested on each of these environments. Release Management also tracks whether an issue fixed by a developer, or a product backlog item completed by your team, has been deployed to a specific environment.
* **You need control of the deployments.** Release Management lets you specify which users can change the configuration of an environment, or approve the release to be deployed into a particular environment. If there is a problem with your deployment, Release Management helps you roll back to a previous deployment, and provide all the logs in one place to help you debug the problem.
* **You need audit history for all releases and their deployments.** Release Management provides a history of all changes to the definitions, configurations, and deployments. It also provides a history of all the activity performed during each deployment. Each release is accompanied by a listing of new features and developer commits that went into that release.

## **2.2 How does Release Management work?**

The Release Management service stores the data about your release definitions, environments, tasks, releases, and deployments in VSTS or TFS.



Release Management runs the following steps as part of every deployment:

1. **Pre-deployment approval:** When a new deployment request is triggered, Release Management checks whether a pre-deployment approval is required before deploying a release to an environment. If it is required, it sends out email notifications to the appropriate approvers.
2. **Queue deployment job:** Release Management schedules the deployment job on an available [automation agent](https://docs.microsoft.com/en-us/vsts/build-release/concepts/agents/agents). An agent is a piece of software that can run tasks in the deployment.
3. **Agent selection**: An automation agent picks up the job. The agents for Release Management are the same as those that run your Builds in VSTS and TFS. A release definition can contain settings to select an appropriate agent at runtime.
4. **Download artifacts**: The agent downloads all the artifacts specified in that release (provided you have not opted to skip the download). The agent currently understands two types of artifacts: Team Build artifacts and Jenkins artifacts.
5. **Run the deployment tasks**: The agent then runs all the tasks in the deployment job to deploy the app to the target servers for an environment.
6. **Generate progress logs**: The agent creates detailed logs for each step while running the deployment, and pushes these logs back to VSTS or TFS.
7. **Post-deployment approval:** When deployment to an environment is complete, Release Management checks if there is a post-deployment approval required for that environment. If no approval is required, or upon completion of a required approval, Release Management proceeds to trigger deployment to the next environment.

# 2.3 Environments in Release Management

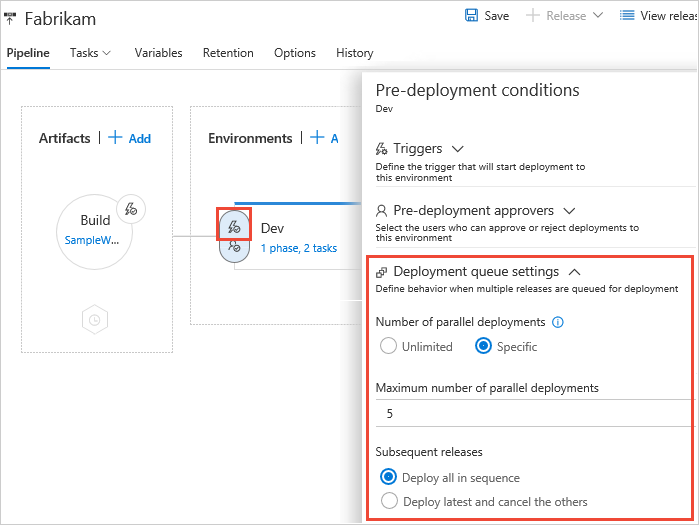
* An **environment** is a logical and independent entity that represents where you want to deploy a release generated from a release definition. We'll examine these two characteristics in more detail to help you understand how to divide your release process into environments.
* First, an environment in a release definition is a **logical** entity. It can represent any physical or real environment that you need. For example, the deployment in an environment may be to a collection of servers, a cloud, or multiple clouds. In fact, you can even use an environment to represent shipping the software to an app store, or the manufacturing process of a boxed product.
* Second, you must be able to deploy to an environment **independently** of other environments in the definition. For example, your definition might consist of two environments A and B, and Release Management could deploy Release 2 to A and Release 1 to B. If you make any assumptions in B about the existence of a certain release in A, the two environments are not independent.

Here are some suggestions and examples for environments:

* **Dev, QA, Prod** - As new builds are produced, they can be deployed to Dev. They can then be promoted to QA, and finally to Prod. At any time, each of these environments may have a different release (set of build artifacts) deployed to them. This is a good example of the use of environments in a release definition.
* **Customer adoption rings** (for example, early adopter ring, frequent adopter ring, late adopter ring) - You typically want to deploy new or beta releases to your early adopters more often than to other users. Therefore, you are likely to have different releases in each of these rings. This is a good example of the use of environments in a definition.
* **Database and web tiers of an application** - These should be modeled as a single environment because you want the two to be in sync. If you model these as separate environments, you risk deploying one build to the database environment and a different build to the web tier environment.
* **Staging and production slots of a web site** - There is clearly an interdependence between these two slots. You do not want the production slot to be deployed independently of the build version currently deployed to the staging slot. Therefore, you must model the deployment to both the staging and production slots as a single environment.
* **Multiple geographic sites with the same application** - In this example, you want to deploy your website to many geographically distributed sites around the globe and you want all of them to be the same version. You want to deploy the new version of your application to a staging slot in all the sites, test it, and - if all of them pass - swap all the staging slots to production slots. In this case, given the interdependence between the sites, you cannot model each site as a different environment. Instead, you must model this as a single environment with parallel deployment to multiple sites (typically by using phases).

## **Multiple test environments to test the same application** - Having one or more release definitions, each with multiple environments intended to run test automation for a build, is a common practice. This is fine if each of the environments deploys the build independently, and then runs tests. However, if you set up the first environment to deploy the build, and subsequent environments to test the same shared deployment, you risk overriding the shared e **Queuing policies**

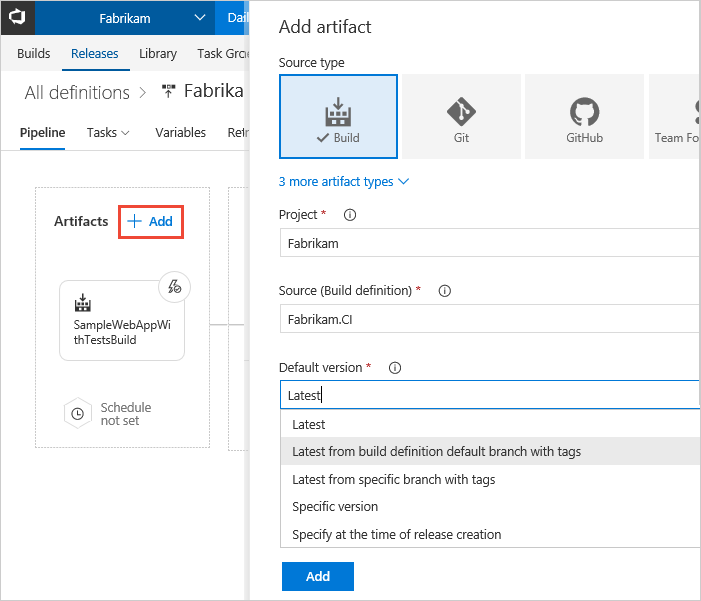
In some cases, you may be generating builds more quickly than they can be deployed. Alternatively, you may configure multiple agents and, for example, be creating releases from the same release definition for deployment of different artifacts. In such cases, it's useful to be able to control how multiple releases are queued into an environment. **Queuing policies** give you that control.



## **2.4 Artifact sources**

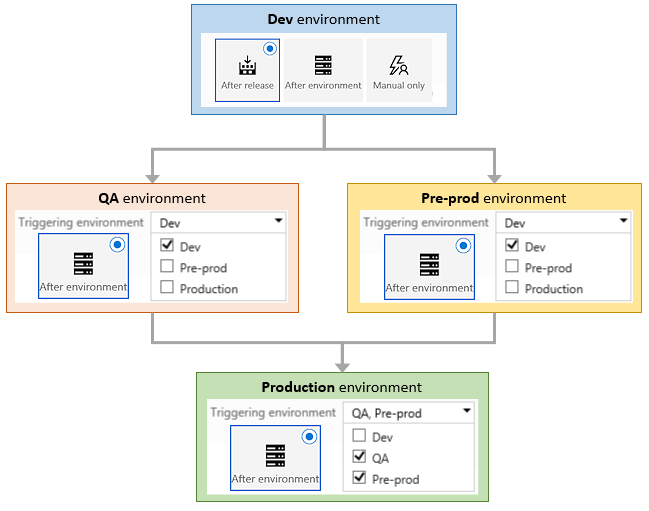
There are several types of tools you might use in your application lifecycle process to produce or store artifacts. For example, you might use continuous integration systems such as Team Build to produce artifacts. You might also use version control systems such as Git or TFVC to store your artifacts. You can configure Release Management to deploy artifacts from all these sources.

By default, a release created from the release definition will use the latest version of the artifacts. At the time of linking an artifact source to a release definition, you can change this behavior by selecting one of the options to use the latest build from a specific branch by specifying the tags, a specific version, or allow the user to specify the version when the release is created from the definition.



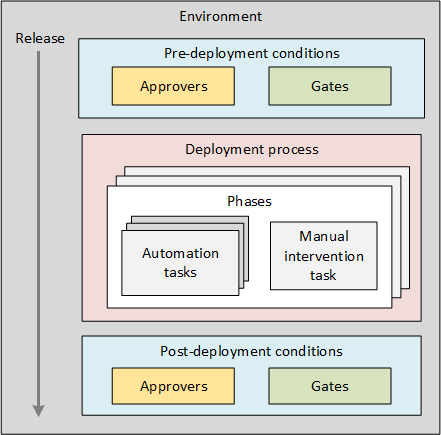
### **2.5 Team Build**

You can link a release definition to any of the build definitions in your Visual Studio Team Services (VSTS) account or Team Foundation Server project collection



# 2.6 Approvals and gates overview

# ****Approvals**** and ****gates**** give you additional control over the start and completion of the deployment process. Each environment in a release definition can be configured with pre-deployment and post-deployment conditions that can include waiting for users to manually approve or reject deployments, and checking with other automated systems until specific conditions are verified. In addition, you can configure a manual intervention to pause the deployment process and prompt users to carry out manual tasks, then resume or reject the deployment.



## **2.7 Build Automation**

Many teams have multiple requirements for build, but the following practices can be applied to most teams. You may adopt the whole approach or just implement the components that work out best for you.

**Daily Builds:** Have a build definition for scheduled builds. Aim for a daily schedule with builds released to the internal SWE environment by the end of each day.

**One-click builds for non-internal environments:** For Integration/UAT environments, you automate the builds. Instead of scheduling them on a per day basis, you can trigger them by queuing them in VSTF. (The reason for not scheduling them is that a build is not required on Integration/UAT environments on a daily basis. Rather, they tend to happen on an as-needed basis. This will depend on your team's needs and you can adopt the rhythm that works best for your team.)

**Gated Check-ins:** Set up gated check-ins to ensure that only code that complies and passed unit testing gets checked in. It ensures that code quality remains high and that there are no broken builds. Integrate Fortify and Code Analysis to get further insight into code quality.

**Code Analysis Integrations:** To get insight into whether the code is of good quality or if any changes need to be made, integrate Code Analysis into the build definitions and set the threshold to low. The changes can be identified and fixed early, which is required in the Agile world.+

**Fortify Integrations:** Use Fortify for security-based checks of the build definitions associated with your check-ins and daily builds. This ensures that any security vulnerabilities are identified as soon as possible and can be fixed quickly.

1. **Branching Strategy**

**Branching:**Branching is an important and powerful technique for creating a parallel set of versions of your files.



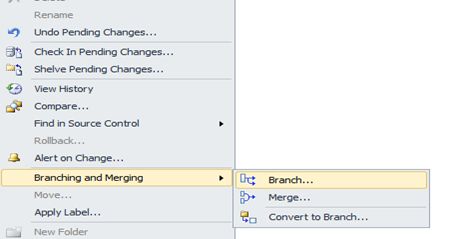
### **3.1 To create a branch**

1. In **Source Control Explorer**, right-click the branch that you want to branch, point to **Branching and Merging**, and then click **Branch**.
2. The **Branch from** dialog box appears.
3. In the **Target Branch Name** box, specify the path of the new branch.
4. (Optional) In the **Branch from version** section, click one of the following options in the **By** list:
   * If you click **Latest Version**, the branch is created for the most recent version in version control.
   * If you click **Change set**, you can specify the number of the change set in the **Changeset**box. As an alternative, click the ellipses (**…**) to open the **Find Change sets** dialog box.

For more information, see Find and view change sets.

* + If you click **Date**, you can specify a date in the **Date** box.
  + Click **Branch**.

The branch is created and appears in **Source Control Explorer**



1. **Documentation in VSTS**

# Manage documents and document libraries

You can share documents and files that you want to make available to all team members by uploading them to the project portal for your team project. You can create document libraries and organize the files that you upload to your project portal within those libraries, in addition to folders and subfolders. The folders and subfolders always appear in alphabetical order.

Your team project's **Documents** page displays all the project portal document libraries as its child nodes. These nodes are the same names that appear when you click **Documents** in the project portal. The **Documents** node is another view of the document libraries on the project portal.

You can view documents by double-clicking them. You can also upload, delete, move, and perform other tasks on the documents, libraries, and folders.

When you create a team project, the process template that you use determines the default names and content of the folders that are located under the **Documents** node of Team Explorer. You can add other folders to the Documents node. The folders and subfolders always appear in alphabetical order.

You can use Team Explorer or your project portal to manage documents and document libraries.

**Requirements**

* Your team project has a project portal enabled and is associated with a SharePoint site. The **Documents** page appears in Team Explorer only when these conditions are met.
* You must be a member of the **Team Foundation Valid Users** security group. If the necessary security permissions are set explicitly, your **View project-level information** permission on the team project must be set to **Allow**.
* You must also have **Contribute** permissions set for the project portal in SharePoint Products.

## Upload a document

#### To upload a document in Team Explorer

1. In Team Explorer, under the **Documents** node, right-click the folder where you want to upload the document, and then click **Upload Document**.
2. In the **Open** dialog box, click the document that you want to upload.

If necessary, use the navigation buttons to locate the document.

1. Click **Open**.

The document is uploaded to the folder that you right-clicked.

#### To upload a document in Windows Explorer

1. In Team Explorer, expand the **Documents** node or child node where you want to upload the document.
2. In Windows Explorer, click the document that you want to upload, and then drag it to the folder or document library under the **Documents** node. Note that in Windows 8, Windows Explorer is File Explorer.

## Create a document library

#### To create a document library in Team Explorer

1. In Team Explorer, right-click the Documents node for the team project where you want to create the document library, and then click **New Document Library**.
2. Type a name for the library, and then press ENTER.

## Add a folder or subfolder

#### To add a folder or subfolder in Team Explorer

1. In Team Explorer, right-click the folder that is above where you want to add the folder, and then click **New Folder**.

2. Type a name for the new folder, and then press ENTER.

## Move, copy, or delete a document or folder

#### To move or copy a document or folder to a new folder

1. In Team Explorer, right-click the document or folder, and then click **Cut** or **Copy**.
2. Right-click the target folder, and then click **Paste**.

To delete a document or folder

1. In Team Explorer, right-click the document or folder, and then click **Delete**.
2. Click **OK**.

## Rename a document or folder

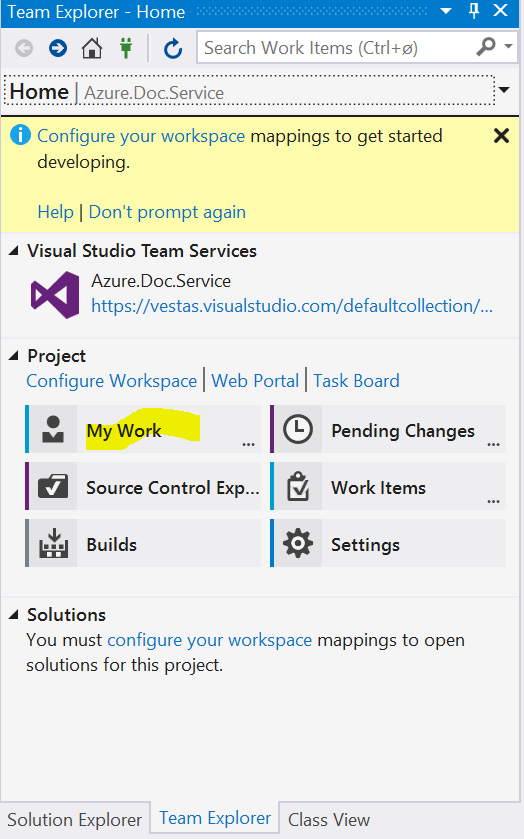
#### To rename a document or folder

1. Right-click the document or folder, and then click **Rename**.
2. Type a new name, and then press ENTER.

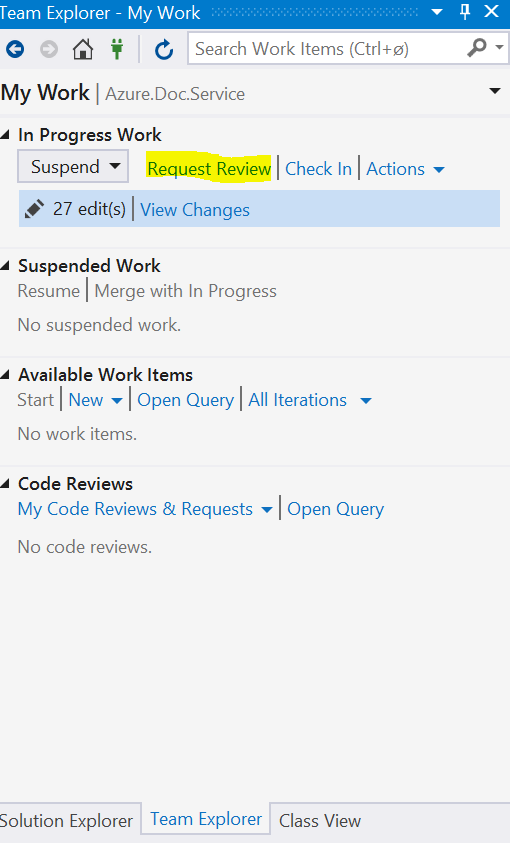
**5 Code Review**

#### **5.1 How to Request for a code review**

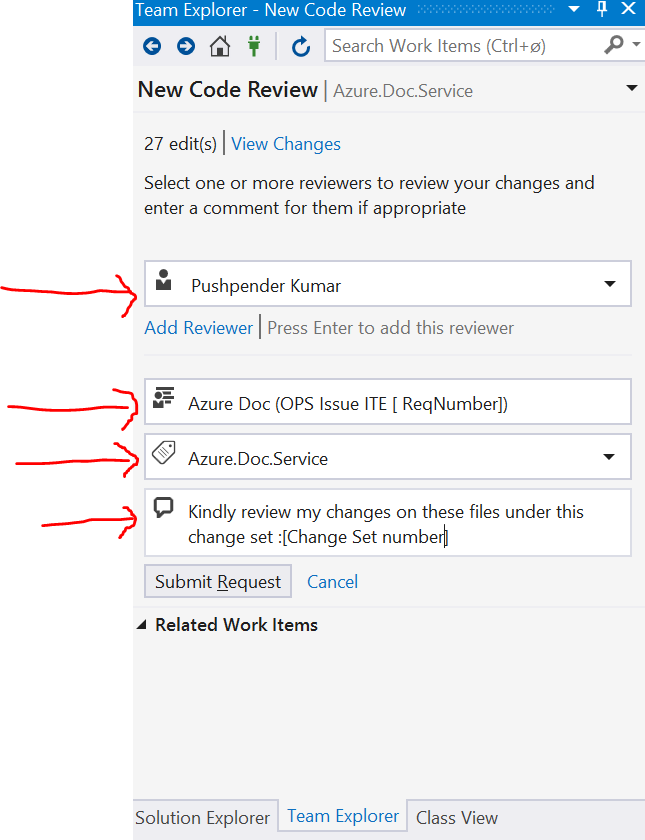
* Before Developer checks in the change, Requester goes to my work page.



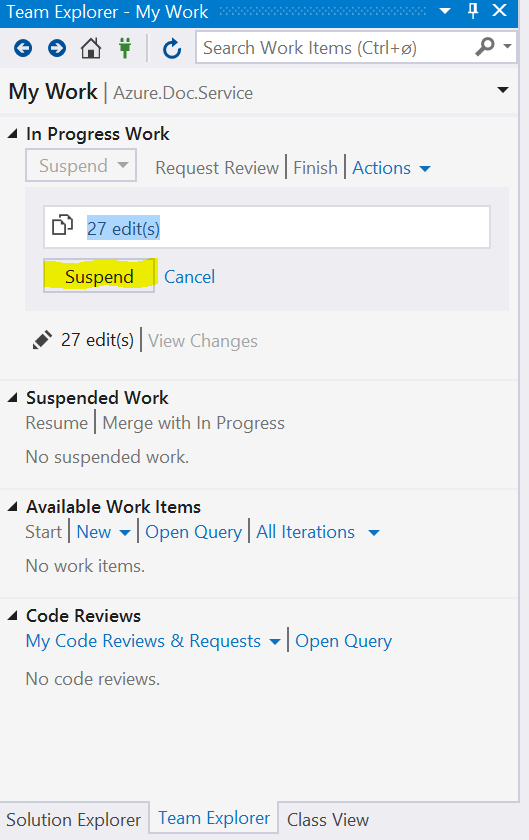
* Developer Request a Review.



* Developers submits a request to Reviewer (Colleague) with following details added in request.

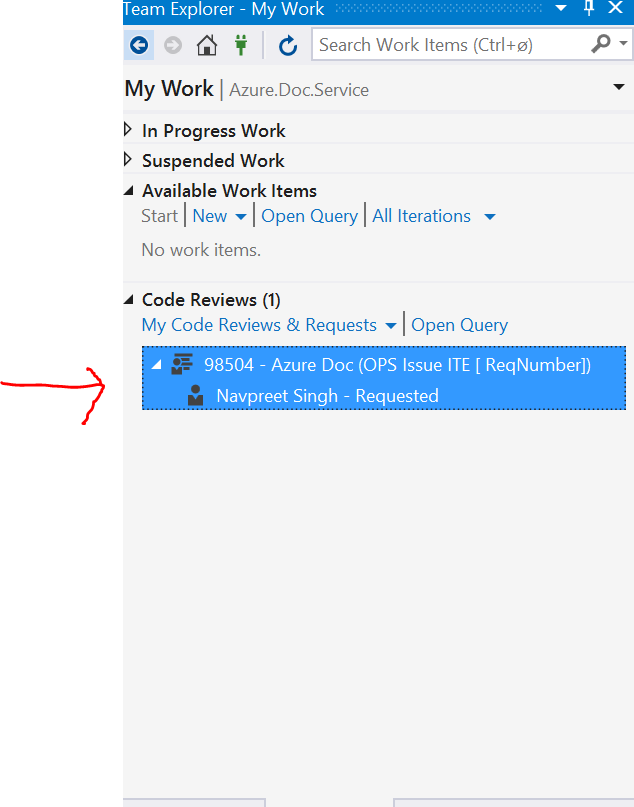


* Developers suspends his work on his code so that he can work on something else while he waits to hear back from Reviewer.

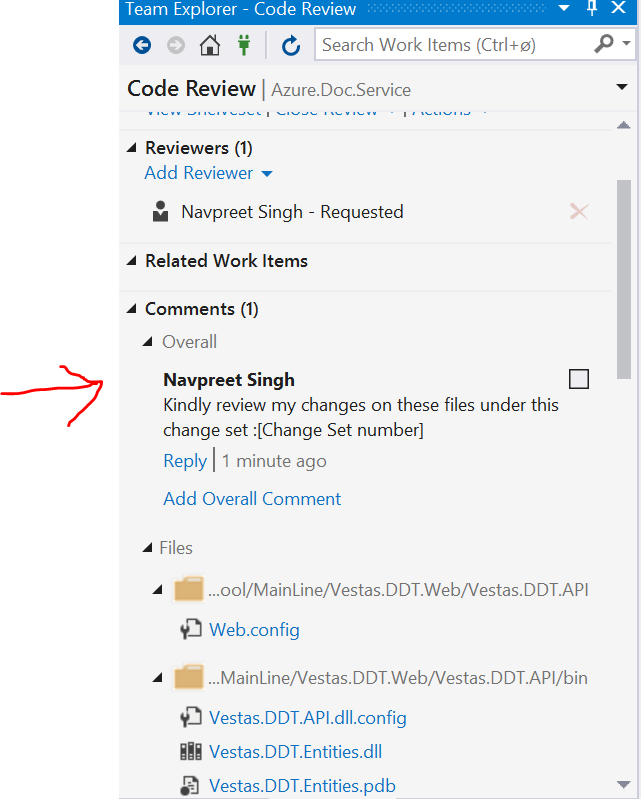


#### **5.2 How to Respond to the code review request**

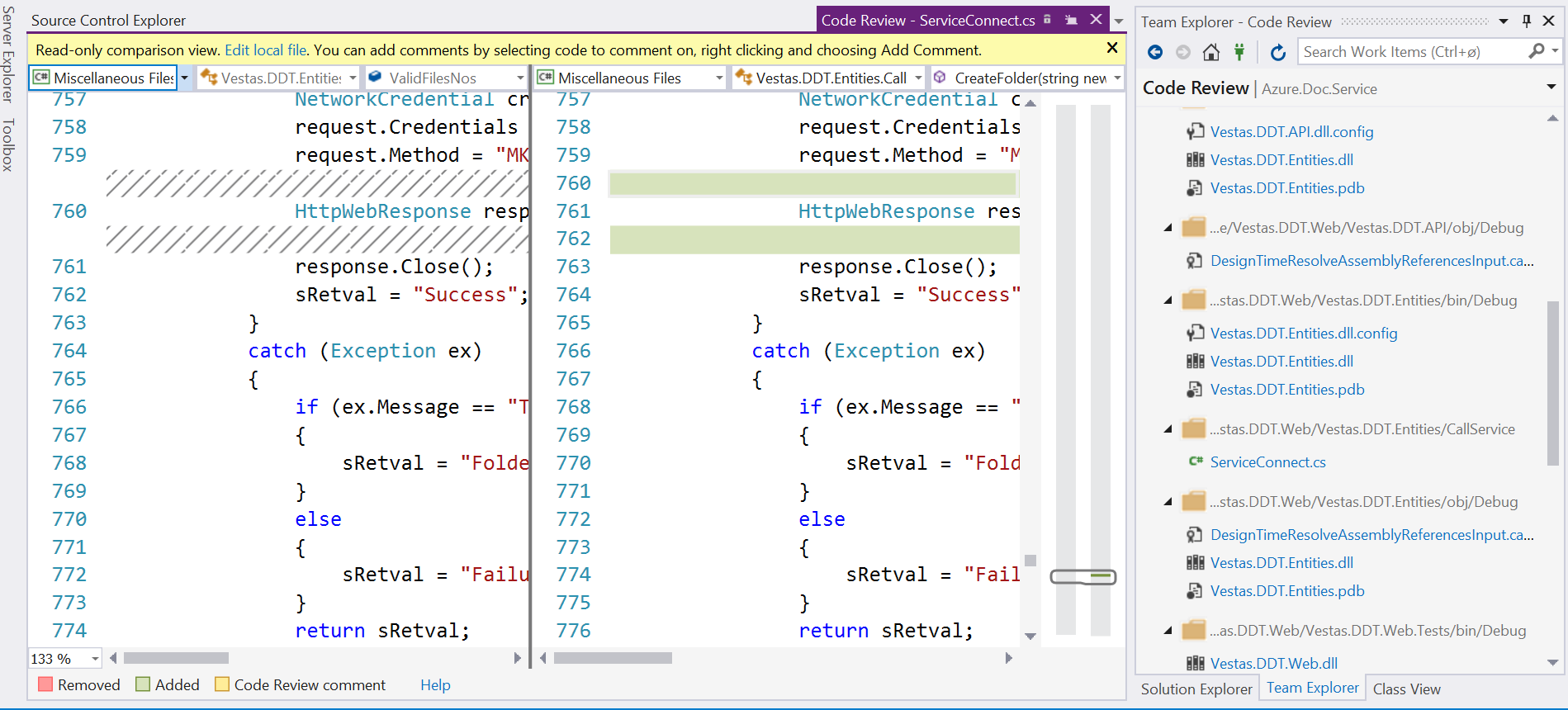
* Reviewer will see the code review request in the team explorer, look at the changes, and give Developer his feedback.

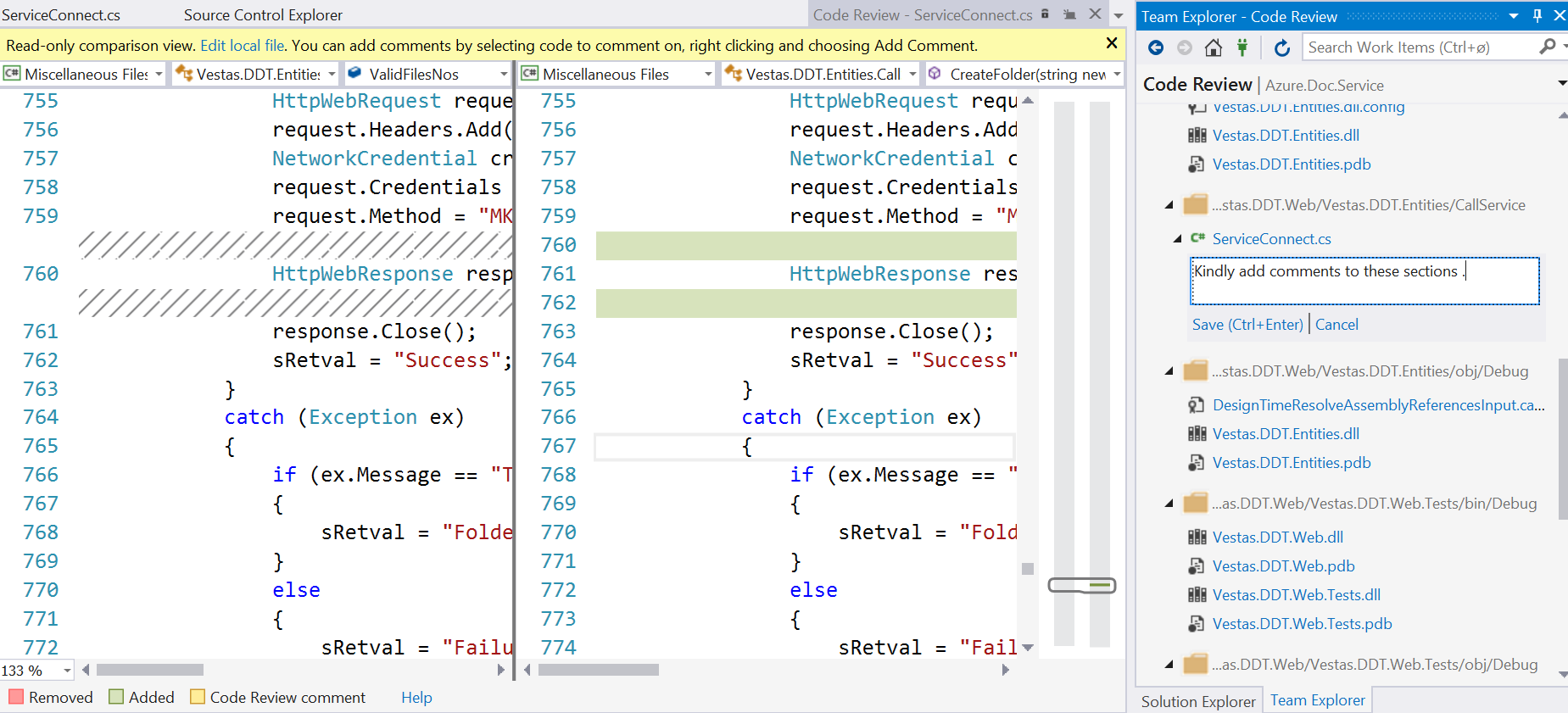


* Reviewer sees Developers’ code review request.

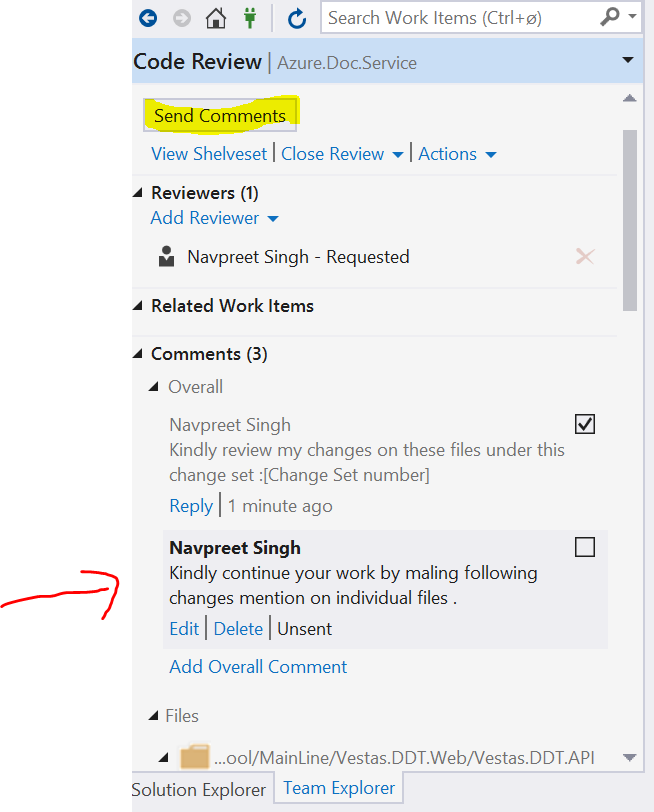


* Reviewer suggests an option by comparing the code and can change or add comments on lines and files.





* After reviewing all files , reviewer will add overall comments on the files and send the feedback back to devlopers .



#### **5.3 Update the code based on the review feedback**

* Developer will accept the review comments and resume his work on that changes.

