CICD pipeline for your extensions with Visual Studio Team Services
Abstract

A step by step tutorial to show you how to set up continuous integration and deployment pipeline for an extension using Visual Studio Team Services (VSTS). This lab is focused on the Ops in DevOps, not the development of extensions.

Important advisory

As the Visual Studio Team Services are updated regularly, based on a 3-week cadence, some of the screenshots are likely going to be out-of-date by the time you’re working through the lab.

Please use every screenshot for reference only and adapt your walk-through as needed.

Objectives

As a result of working through this hands-on lab, you will be better able to:

- Create a continuous integration and delivery pipeline for my extensions
- Understand how to automate my CI/CD pipeline
- Deliver continuous value to the marketplace
Prerequisites

You should:

- Be interested in DevOps practices.
- Be familiar with Visual Studio Team Services (VSTS).
- Have a non-production VSTS account, with administrative privileges.
- Have a non-production and unverified Visual Studio Marketplace publisher account

Don’t worry if you don’t have a VSTS account or a Visual Studio marketplace account we will help you create them.
Exercise 1 (optional): Review pre-requisites

1.1 DevOps and Pipelines
Complete this optional exercise if you're unfamiliar with DevOps, Continuous Integration (CI), Continuous Delivery (CD), CICD Pipelines, or VSTS Extensions.

1. Open your favorite browser
2. DevOps
   - Go to aka.ms/whatisdevops
   - Explore What Is DevOps
   - Explore What is Continuous Integration
   - Explore What is Continuous Delivery
3. Pipelines
   - Go to aka.ms/cicdpipelines
   - Explore Set up a CI/CD pipeline with unit testing and code coverage for your Team Services extension, which forms the basis of this hands-on lab
4. Extensions
   - Go to docs.microsoft.com/en-us/vsts/extend
   - Explore the How-to-guides section, including the Packaging and publishing section

1.2 VSTS Account
Complete this exercise if you do not have a VSTS account to use for this hands-on lab.

Do NOT use a production account for this hands-on lab.

1. Getting started for “free” with Visual Studio
   - Go to www.visualstudio.com and sign in
   - Click on Get started for free under Visual Studio Team Services
   - Click on Create new account

Visual Studio Team Services Accounts
Create new account
CICD Pipeline for your extensions with Visual Studio Team Services

- Pick a memorable and unique name, for example TECH-DEV-HOL201- youralias, select Git, and optionally change your Project Name, process, and hosting region

- Click on Continue and wait for your new account to be created

Congratulations, you’re done with the pre-requisites. Let’s get some code to explore, build, and release.
Exercise 2: Initialize your Git repository

Use the new VSTS account and team project you created during exercise 1, for example *MyFirstProject*, or a new team project in your own (non-production) account.

1. **Import a sample repository**
   - Click on **import a repository** on the *Get started with your new project* home page

   ![Importing a sample repository](image)

   - Click on **Import**
CICD Pipeline for your extensions with Visual Studio Team Services

- Set the Clone UR to https://github.com/ALM-Rangers/Countdown-Widget-Extension and click on Import

![Import a Git repository](image)

- Wait for import to complete

![On its way!](image)

- Optionally explore the extension code

![Git repository](image)

*Congratulations, you have a copy of the Countdown Widget Extension in your Git repository. Let’s create a marketplace publisher.*
Exercise 3: Create a publisher

Every item on the Visual Studio Marketplace, including extensions, belong to a publisher. You need a publisher to publish the extension we imported in the precious exercise to the marketplace.

If you already have a publisher, proceed to exercise 4.

Do NOT use a production or verified publisher for this hands-on lab.

1. Create a publisher
   - Open a new browser tab
   - Go to https://docs.microsoft.com/en-us/vsts/extend/publish/overview
   - Follow the instructions and create a new publisher

Congratulations, you now have a marketplace publisher. Let's create a service endpoint to connect the marketplace.
CICD Pipeline for your extensions with Visual Studio Team Services

Exercise 4: Create a service endpoint to the Marketplace

We need to create a service endpoint that connects your Team Project with your Marketplace publisher.

1. Create personal access token
   - Open a new browser tab
   - Go to https://docs.microsoft.com/en-us/vsts/extend/publish/command-line
   - Follow the instructions and acquire a personal access token

   ![Acquire a personal access token](image)

   Copy your generated personal access token. We’ll need it for the next exercise.
2. **Install VSTS Developer Tools**

- Click on icon ❶, then click on **Account settings** ❷

- Click on **Extensions**

- Click on **Browse Marketplace**

- Search for **VSTS Developer Tools**

- Click on **VSTS Developer Tools**
CICD Pipeline for your extensions with Visual Studio Team Services

- **Click Install**

- **Select your VSTS account** and click **Continue**

- **Wait for validation to succeed and click Confirm**
CICD Pipeline for your extensions with Visual Studio Team Services

- Click on Proceed to the account

3. Create service endpoint

- Click on your team project, for example MyFirstProject

- Click on icon ❶, then click on Services ❷
CICD Pipeline for your extensions with Visual Studio Team Services

- Click on **New Service Endpoint** ❶ and select **VSTS Marketplace** ❷

- Specify a **Connection name** ❶, paste your **personal access token** ❷ you created at the beginning of this exercise, and click **OK** ❸
• Click on your team project, for example MyFirstProject, and click on MyFirstProject Home

Congratulations, you now have completed the “mind numbing” setup of the infrastructure. It’s a once-off setup and you’ll be able to re-use the infrastructure for all your CI/CD pipelines in your Team Project of your VSTS account.

Looking for ways to automate the setup of this infrastructure? Explore https://github.com/DarqueWarrior/generator-team
Exercise 5: Create a continuous integration (CI) build

We need to create a build that is triggered with every check-in, starting with a build and package of our extension.

1. Create empty process build
   - Click on Set up Build
   - Choose an empty process

2. Configure continuous integration (CI)
   - Click on Triggers and Enable the trigger.
CICD Pipeline for your extensions with Visual Studio Team Services

- Click on **Tasks** ❶ and select the **Hosted** ❷ agent queue. Note that the default queue location is now inside the Tasks tab itself.

![Tasks and Hosted Agent Queue](image)

- Click on **Options** ❶ and configure the **Build format number** ❷ as 1.0$(rev:.r).

![Build Format Number](image)

**NOTE** – A missing dot can result in an invalid version number and break your build. You can specify the format in two ways: 
1.0$(rev:.r)  
1.0$.$(rev:r)

3. **Create build variables**

- Click on **Variables** ❶  
- Change **system.debug** ❷ variable to **true** to enable verbose logging  
- Add new variable **BuildConfiguration** ❸ and set it to **release**  
- Add new variable **BuildPlatform** ❹ and set it to **any cpu**
4. Add build tasks

- Click on Tasks

![Tasks](image1)

- Click on Add Task

![Add Task](image2)

**NOTE** - This build is based on the **Countdown Widget** sample extension and serves to showcase a few build task examples. It’s important you plan your build to meet the requirements for your extension, which in most cases will be simpler than this example scenario. Bookmark [aka.ms/cicdpipeline](aka.ms/cicdpipeline) for a few walk-throughs of various extension pipelines we’re using in production.

5. Install NPM packages

- Click on Add Task
CICD Pipeline for your extensions with Visual Studio Team Services

- Search for npm ❶ and click on Add ❷ to add the npm tasks.

- Click on the npm install task ❶. Select Version 1.* ❷ and configure the working folder ❸, containing the package.json configuration file, as shown.

6. Build the extension

- Click on Add Task

- Search for npm ❶ and click on Add ❷ to add the npm tasks.
CICD Pipeline for your extensions with Visual Studio Team Services

- Click on the second npm task.

  ![CICD Pipeline for your extensions with Visual Studio Team Services](image)

  - Select Version 1.*.
  - Set the display name to npm run build.
  - Select the custom command.
  - Configure the working folder, containing the package.json configuration file, as shown.
  - Set the Command and arguments to build.

7. **Package the extension**

   - Click on Add Task
   - Search for Package Extension and click on Add to add the task.

**NOTE** - This is where the magic of this pipeline starts! The Package Extension task allows us to override the extension manifest, allowing is to reconfigure the extension during build, and later for each release environment.
Click on **Package Extension** 1 to configure the build task

- **Root manifest folder** 2 set to the `CountdownWidget/CountdownWidget`, which contains the `vss-extension.json` manifest file.
- **Manifest file** 3 set to `vss-extension.json`
- **Package output file** 4 set to `drop\output.vsix`
- **Output Variable** 5 defaults to `Extension.OutputPath`
CICD Pipeline for your extensions with Visual Studio Team Services

- Configure **Overrides manifest** settings to override values in the manifest file

  ![Image of Visual Studio](image)

  - **Publisher ID** ☞ set to the marketplace publisher you created in Exercise 3.
  - **Extension ID** ☞ set to unique ID, for example `CountDownSample`
  - **Extension Tag** ☞ set to `Alpha`
  - **Extension name** ☞ set to `Count Down Sample`
  - **Extension version** ☞ set to `${Build.BuildNumber}`
  - **Override tasks version** ☞ is checked
  - **Extension visibility** ☞ set to `Private`
  - **Extension pricing** ☞ set to `Free`

8. **Publish artifact**

   - Click on **Add Task**
CICD Pipeline for your extensions with Visual Studio Team Services

- Search for **Publish Build Artifacts** and click on **Add** to add the task

![Image of Visual Studio Team Services with Publish Build Artifacts task]

- Click on **Publish Artifact** to configure the task

![Image of Visual Studio Team Services with Publish Artifact configuration]

  - **Path to Publish** set to `${Extension.OutputPath}` variable, which was configured during the Package Extension task
  - **Artifact Name** set to `drop` folder
  - **Artifact Type** set to `Server`

9. It's time to validate the build

- Click on **Save & Queue** drop-down and click on **Save & queue**

![Image of Visual Studio Team Services with Save & Queue options]
CICD Pipeline for your extensions with Visual Studio Team Services

- **Click in Queue**

10. **Observe the console output of the build**

11. **Verify that we have no errors**

Congratulations, you've created and tested your continuous integration build. You've completed the most complex part of our pipeline, which now looks as follows:

![Continuous Integration (CI)](image)
Exercise 6: Create a continuous delivery (CD) release

To complete the CI CD pipeline we need to create a release that is triggered by the build artifact. We’ll use the Publish Extension Task of the VSTS Developer Tools Build Tasks we used to package the extension during the build. This updates the VSIX package file by unzipping the content, updating the configuration values and zipping all the files. The task then deploys it to the configured Marketplace publisher account and deploy the extension to distinct DEV → BETA → PROD environments.

This exercise is based on the new release editor. You can enable the new editor by selecting in your profile.

1. Create empty release
   - You can get started with your release definition in two ways:
     
     Click on Release in your build summary and click the Yes button when promoted to create a new release definition.

     1. Click on Releases ❶ and then on New Definition ❷
CICD Pipeline for your extensions with Visual Studio Team Services

- Select the **Empty process**.

- Click on the Artifact trigger ① and verify that the **Continuous deployment trigger** ② is enabled.

- Click on the **Build artifacts** ① and verify that your **Artifact** ② defaults as shown.
• Click on the environment ❶ and change the name to DEV ❷.

• Change the release name to Countdown Sample.

• Click on pre-deployment conditions ❶ and review the Approvals which are set to Automatic by default ❷.
2. Configure DEV environment

- Click on 1 phase(s) 0 task(s).

- Click on + (add a task to the phase).

- Search for Publish Extension ① and Add ② to the agent phase.

- Click on Publish Extension task to configure the settings that need attention.
• Set the **VSTS Marketplace connection** to the service endpoint ❶ you created in exercise 4, and select VSIX file ❷.

• Configure the remainder of the Publish Extension task.

• **VSIX file** ❶ set to the
  
  `$(System.DefaultWorkingDirectory)/MyFirstProject-CI/drop/output.vsix`
  
  file, which was created by the build.

• **Publisher ID** ❷ set to the marketplace publisher, which you created in Exercise 3.

• **Extension ID** ❸ set to unique ID, for example CountdownSample.
CICD Pipeline for your extensions with Visual Studio Team Services

- **Extension Tag** set to DEV to match the DEV environment.
- **Extension name** set to Count Down Sample DEV.

**NOTE** – If you deploy your extension to the same publisher and/or the same VSTS accounts, we recommend that you change the extension name to include the extension tag, for example **CountdownSampleBETA**.

It makes it much easier to distinguish which extension is which by just looking at the name.

- **Override tasks version** is checked.
- **Extension visibility** set to Private.
- **Extension pricing** set to Free.
- **Share with** set to our VSTS account, whereby you can configure other VSTS accounts as well to share your DEV extension.

- **Save** the release configuration.
3. Configure BETA environment

- Select the DEV environment ❶, click on Add ❷ and select **Clone selected environment** ❸.

- Change the environment name to BETA. ❹
CICD Pipeline for your extensions with Visual Studio Team Services

- Click on **pre-deployment conditions** ❶ and review the **trigger** ❷. Select **Specific users** ❸ for approval type and add your account to the list of approvers.

- Update the configuration of the cloned Publish Extension task.

- **Extension Tag** ❶ set to BETA to match the DEV environment
CICD Pipeline for your extensions with Visual Studio Team Services

- **Extension name** set to Count Down Sample BETA
- **Share with** set to our VSTS account, whereby you’d deploy to a different environment (ring) in a production environment

4. **Save the release definition**

5. **It’s time to validate the release**

- **Click on + Release** and select **Create Release**

- Review the **Artifacts**, which refer to our latest build, the **Automated deployments**, and click **Queue**.
CICD Pipeline for your extensions with Visual Studio Team Services

- Click on **new release** that’s been created to observe the deployment.

- Click on **Logs** and verify that the DEV release is successful and notice that the BETA release waits for manual approval, as configured.

- Click on the **Approvers** icon and click on **Approve** to approve the release to the BETA environment.
• Verify that the BETA release completes successfully as well

• Open a new browser tab and go to

• Verify your publisher is selected and that the Countdown Sample DEV and BETA extension have been publisher successfully.

**IMPORTANT - We’re intentionally NOT implementing the PROD environment, which publishes a public version of the extension. It’s important we do not duplicate features on the marketplace, and review the extension product documentation before we flip the public switch. There are scenarios in which you cannot undo, for example uninstall, a public extension publication.**
CICD Pipeline for your extensions with Visual Studio Team Services

Congratulations, you’ve created and tested your continuous delivery for your extension. At this stage your pipeline looks as follows:

But, does it work? Let’s move to exercise 7 and find out.
Exercise 7: Trigger a change

Let's validate that we have a fully automated CI/CD pipeline by making a minor change to the code base.

1. **Make code change**

   - Click on **Code** and click on **Files**
   - Click on the **README.md** file

   **NOTE** – We’re editing the root README.md file, which is not the same file as the Overview.md file that defines what the user sees on the marketplace.

   - Click on **Edit** to modify the file
CICD Pipeline for your extensions with Visual Studio Team Services

- Make a simple change, for example add an exclamation mark after the heading, and click Commit icon

- Optionally add a commit Comment and click Commit

- The is committed to the repo

2. Observe the pipeline

- Click on Build & Release, select Build, and notice that a new build has been triggered by the README.md update
Click on **Build & Release**, select **Release**, and notice that a new release has been triggered when the build completed.

<table>
<thead>
<tr>
<th>All release definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
</tr>
<tr>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>

3. **Validate the extension was published**

- Click on **icon** and select **Account settings**

- Click on **Extensions**

- You'll notice that we have the two extensions shared with our account, as expected.
CICD Pipeline for your extensions with Visual Studio Team Services

- When you click on one of the shared extensions, you can review the extension on the publisher and find the DEV or BETA suffix, as highlighted

Thank you for exploring this hands-on lab with us. We hope you have enjoyed the tour and look forward to your feedback. You can reach us on aka.ms/vsar and @almrangers.
CICD Pipeline for your extensions with Visual Studio Team Services

Exercise 8: (optional) Explore unit testing and code coverage

We have covered the core steps of creating a CI/CD pipeline for your extension. To fully embrace DevOps practices, we need to implement testing, metrics (such as code coverage), and application insights telemetry.

We'll leave you with a list of useful references to explore as an optional exercise:

Extension pipelines

- Setup a CI/CD pipeline with package management for your Visual Studio Team Services extension
- Set up a CI/CD pipeline with unit testing and code coverage for your Visual Studio Team Services extension
- Set up a CI/CD pipeline for your Visual Studio Team Services extension
- Setup a CI/CD pipeline with build & release tasks for your Visual Studio Team Services extension – coming soon

Extension development

- Yeoman – generator-vsts-extension
- VSTS Extension Project Template

Other pipelines

- Set up a CI/CD pipeline to run automated tests efficiently
- Set up a CI/CD pipeline for your Yeoman generator package

Thank you for exploring this hands-on lab with us. We hope you have enjoyed the tour and look forward to your feedback. You can reach us on aka.ms/vsar and @almrangers.
DEPRECATED Exercise 6: Create a continuous delivery (CD) release using old editor

To complete the CI CD pipeline we need to create a release that is triggered by the build artifact. We’ll use the Publish Extension Task of the VSTS Developer Tools Build Tasks we used to package the extension during the build. This updates the VSIX package file by unzipping the content, updating the configuration values and zipping all the files. The task then deploys it to the configured Marketplace publisher account and deploy the extension to distinct DEV → BETA → PROD environments.

This exercise is based on the classic release editor. See Exercise 6b: Create a continuous delivery (CD) release with new editor for the new editor experience.

1. Create empty release
   - You can get started with your release definition in two ways:
     2. Click on Release in your build summary and click the Yes button when promoted to create a new release definition.

-or-

3. Click on Releases 1 and then on New Definition 2
CICD Pipeline for your extensions with Visual Studio Team Services

- Select the **Empty** release template and click **Next**
CICD Pipeline for your extensions with Visual Studio Team Services

- Verify that the project is set to your project ①, the build to your CD build ② created in exercise 5, check **Continuous Deployment** ③, and click Create

- Change the environment name to **DEV**
CICD Pipeline for your extensions with Visual Studio Team Services

• Change the release name to **Countdown Sample**

![Release name format](image)

• Click on **General** and set the **Release name format** to **Countdown Sample-$\{rev:r\}**

2. **Configure DEV environment**

• Click on **Environments** to return to the environments tab

• Click on , select **Assign approvers**, review the **Approvals** which are set to **Automatic** by default, and click **Cancel** to return to the release

![Assign approvers](image)

• Click on **Add Tasks**
CICD Pipeline for your extensions with Visual Studio Team Services

- Look for **Publish Extension** in the list of **Deploy** tasks, click on **Add** to add the task, and **Close** to return to the configuration.

- Set the **VSTS Marketplace connection** to the service endpoint you created in exercise 4, and select VSIX file.
CICD Pipeline for your extensions with Visual Studio Team Services

- Click on the **Publish Extension** task to configure

![Publish Extension Task Diagram]

- **VSIX file** ① set to the
  
  `$(System.DefaultWorkingDirectory)/MyFirstProject-CI/drop/output.vsix`

  file, which was created by the build.

- **Publisher ID** ② set to the marketplace publisher, which you created in Exercise 3.

- **Extension ID** ③ set to unique ID, for example CountDownSample

- **Extension Tag** ④ set to DEV to match the DEV environment

- **Extension name** ⑤ set to Count Down Sample DEV
NOTE – If you deploy your extension to the same publisher and/or the same VSTS accounts, we recommend that you change the extension name to include the extension tag, for example **CountdownSampleBETA**.

It makes it much easier to distinguish which extension is which by just looking at the name.

- **Override tasks version** is checked
- **Extension visibility** set to Private
- **Extension pricing** set to Free
- **Share with** set to our VSTS account, whereby you can configure other accounts as well to share your DEV extension

3. **Configure BETA environment**

- Click on of the DEV environment and select **Clone environment**
• Select **Specific users** 1 for the **Pre-deployment approval**, add your account, ensure the **Trigger** 2 is checked, and click **Create**

![Add new environment dialog](image)

• Rename the new environment to **BETA**

![Environment list](image)
CICD Pipeline for your extensions with Visual Studio Team Services

- Click on the **Publish Extension** task to configure

![Publish Extension Task](image)

- **Extension Tag** ❶ set to BETA to match the DEV environment
- **Extension name** ❷ set to Count Down Sample BETA
- **Share with** ❸ set to our VSTS account, whereby you'd deploy to a different environment (ring) in a production environment
- **Save** the release definition
4. It’s time to validate the release

- **Click on + Release and select Create Release**

- Review the **Artifacts**, which refer to our latest build, the **Automated deployments**, and click **Create**.

- **Click on new release** that’s been created to observe the deployment
CICD Pipeline for your extensions with Visual Studio Team Services

- Click on Logs and verify that the DEV release is successful and notice that the BETA release waits for manual approval, as configured

- Click on the Approvers icon and click on Approve to approve the release to the BETA environment
CICD Pipeline for your extensions with Visual Studio Team Services

- Verify that the BETA release completes successfully as well

- Open a new browser tab and go to https://marketplace.visualstudio.com/manage/publishers

- Verify your publisher is selected and that the Countdown Sample DEV and BETA extension have been publisher successfully.

**IMPORTANT** - We’re intentionally **NOT** implementing the PROD environment, which publishes a public version of the extension. It’s important we do not duplicate features on the marketplace, and review the extension product documentation before we flip the public switch. There are scenarios in which you cannot undo, for example uninstall, a public extension publication.
Congratulations, you’ve created and tested your continuous delivery for your extension. At this stage your pipeline looks as follows:

Using the new release editor, you’ll be able to visualize and configure your pipeline with a similar visual view:

But, does it work? Let’s move to exercise 7 and find out.
CICD Pipeline for your extensions with Visual Studio Team Services