AWS Tools for Microsoft Visual Studio Team Services

User Guide

AWS Tools for Microsoft Visual Studio Team Services: User Guide

Table of Contents

AWS Tools for Microsoft Visual Studio Team Services

AWS Tools for Microsoft Visual Studio Team Services is an extension for Visual Studio Team Services that contains tasks you can use in build and release definitions in VSTS to interact with AWS services. AWS Tools for VSTS is available through the Visual Studio Marketplace.

You can use these tasks in a VSTS project or in an on-premises Team Foundation Server environment. The available AWS tasks include:

- **Deployment tasks**
  - AWS CodeDeploy Deployment Application Task
  - AWS CloudFormation Create-Update Stack Task
  - AWS CloudFormation Delete Stack Task
  - AWS CloudFormation Execute Change Set Task
  - AWS Elastic Beanstalk Deployment Task
  - AWS Lambda .NET Core Deployment Task
  - AWS Lambda Invoke Function Task
- **General purpose tasks**
  - AWS CLI
  - AWS Tools for Windows PowerShell Script Task
  - AWS S3 Download Task
  - AWS S3 Upload Task
  - AWS Send Message Task

What's in This Guide

The AWS Tools for VSTS User Guide describes how to install and use the AWS Tools for VSTS.

Getting Started (p. 2)

How to set up a VSTS account, install the AWS Tools for VSTS and how to set up AWS credentials to use the tasks.

Using the AWS Tools for VSTS (p. 5)

Walk-through topics demonstrating how to use tasks in the AWS Tools for VSTS in your build and release definitions.

Task Reference (p. 16)

Describes the tasks included in the AWS Tools for VSTS.

Getting Started

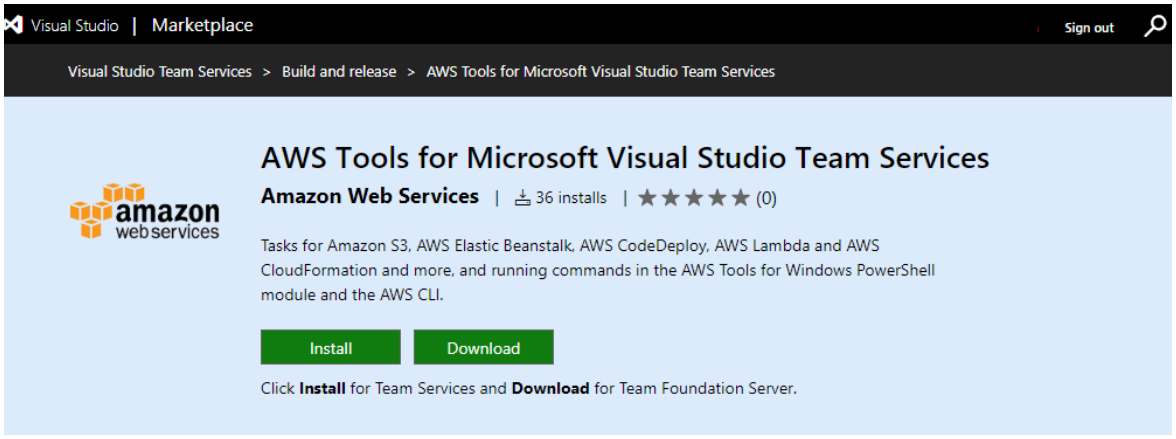
This section provides information about how to install, set up, and use the AWS Tools for Microsoft Visual Studio Team Services.

Set up a VSTS Account

To use Visual Studio Team Services (VSTS), you need to sign up for a Visual Studio Team Services Account.

Install the AWS Tools for VSTS Extension

The AWS Tools for VSTS is installed from the Visual Studio Marketplace. Sign in to your VSTS account, then search for *AWS Tools for Microsoft Visual Studio Team Services*. Choose **Install** to download to VSTS, or choose **Download** to install into an on-premises Team Foundation Server.


Visual Studio | Marketplace
Sign out
Visual Studio Team Services > Build and release > AWS Tools for Microsoft Visual Studio Team Services
amazon web services
Install
Download

AWS Tools for Microsoft Visual Studio Team Services

Amazon Web Services | 36 installs | ★★★★★ (0)

Tasks for Amazon S3, AWS Elastic Beanstalk, AWS CodeDeploy, AWS Lambda and AWS CloudFormation and more, and running commands in the AWS Tools for Windows PowerShell module and the AWS CLI.

Click **Install** for Team Services and **Download** for Team Foundation Server.

Set up AWS Credentials for the AWS Tools for VSTS

To use the AWS Tools for VSTS to access AWS, you need an AWS account and AWS credentials. To increase the security of your AWS account, we recommend that you use an *IAM user* to provide access credentials instead of using your root account credentials.

Note
For an overview of IAM users and why they are important for the security of your account, see Overview of Identity Management: Users in the *IAM User Guide*.

To sign up for an AWS account

1. Open http://aws.amazon.com/, and then choose **Sign Up**.

2. Follow the onscreen instructions. Part of the signup procedure involves receiving a phone call and entering a PIN using your phone keypad.

Next, create an IAM user and download (or copy) its secret access key. To use the AWS Tools for VSTS, you must have a set of valid AWS credentials, which consist of an access key and a secret key. These keys are used to sign programmatic web service requests and enable AWS to verify that the request comes from an authorized source. You can obtain a set of account credentials when you create your account. However, we recommend that you do not use these credentials with AWS Tools for VSTS. Instead, create one or more IAM users, and use those credentials.

To create an IAM user

1. Open the IAM console (you may need to sign in to AWS first).
2. Choose **Users** in the sidebar to view your IAM users.
3. If you don't have any IAM users set up, choose **Create New Users** to create one.
4. Select the IAM user in the list that you want to use to access AWS.
5. Open the **Security Credentials** tab, and then choose **Create Access Key**.

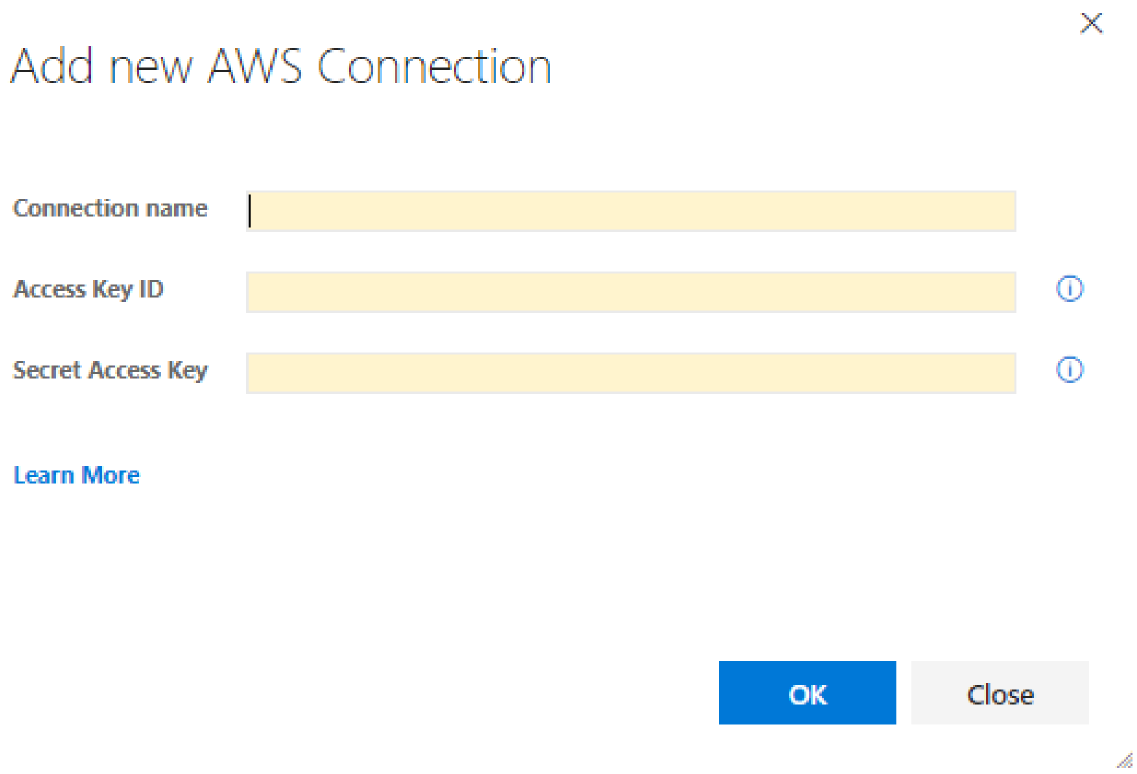
   Note
   You can have a maximum of two active access keys for any given IAM user. If your IAM user has two access keys already, you need to delete one of them before creating a new key.
6. In the dialog box that opens, choose **Download Credentials** to download the credential file to your computer. Or choose **Show User Security Credentials** to view the IAM user's access key ID and secret access key (which you can copy and paste).

   Important
   There is no way to obtain the secret access key once you close the dialog box. You can, however, delete its associated access key ID and create a new one.

Create an AWS Connection

To use the tasks contained in the tools, you must link an AWS subscription to VSTS or Team Foundation Server. **Each VSTS/TFS project is associated with its own set of credentials. The credentials are used by the VSTS/TFS build agents when running builds and/or releases for a project containing tasks from the AWS tools.**

You can link your subscription from the **Services** tab in the Account Administration section. Add the AWS subscription to use in the Build or Release Management definition by opening the Account Administration page (choose the gear icon on the top right of the page), and then choose **Services**. Choose **+ New Service Endpoint**. Select the **AWS** endpoint type. This opens the **Add new AWS Connection** form.



Provide the following parameters, and then click **OK**:

- Connection name
- Access key ID
- Secret access key

The connection name is used to refer to these credentials when you are configuring tasks that access AWS in your build and release definitions.

The credentials associated with the project are used by VSTS or TFS build agents that execute the AWS tasks you configure in your build and/or release pipelines. You can associate a single set of credentials to be used in all AWS tasks in a project or you can associate multiple sets of credentials. Project team members reference the associated credentials when configuring tasks for a project's build and/or release definitions.

Note
We strongly suggest you use access and secret keys generated for an Identity and Access Management (IAM) user account. You can configure an IAM user account with permissions granting access to only the services and resources required to support the tasks you intend to use in your build and release definitions. For more information, see Best Practices for Managing AWS Access Keys.

Tasks can also use assumed role credentials by adding the Amazon Resource name (ARN) of the role to be assumed and an optional identifier when configuring the endpoint. The access and secret keys specified will then be used to generate temporary credentials for the tasks when they are executed by the build agents. Temporary credentials are valid for up to 15 minutes by default. To enable a longer validity period you can set the 'aws.rolecredential.maxduration' variable on your build or release definition, specifying a validity period in seconds between 15 minutes (900 seconds) and one hour (3600 seconds).

For more information, see About Access Keys.

Using the AWS Tools for VSTS

The following tutorials demonstrate how to use tasks from the AWS Tools for Microsoft Visual Studio Team Services in your VSTS projects.

Prerequisites

- Either a VSTS account or on-premises Team Foundation Server.
- An AWS account and preferably an associated IAM user account.
- Task specific permissions.
- An existing VSTS project with the build definition template specified in the tutorial.

See Getting Started (p. 2) for instructions to install the AWS Tools for VSTS and set up your credentials.

Topics

Archiving Build Artifacts to AWS

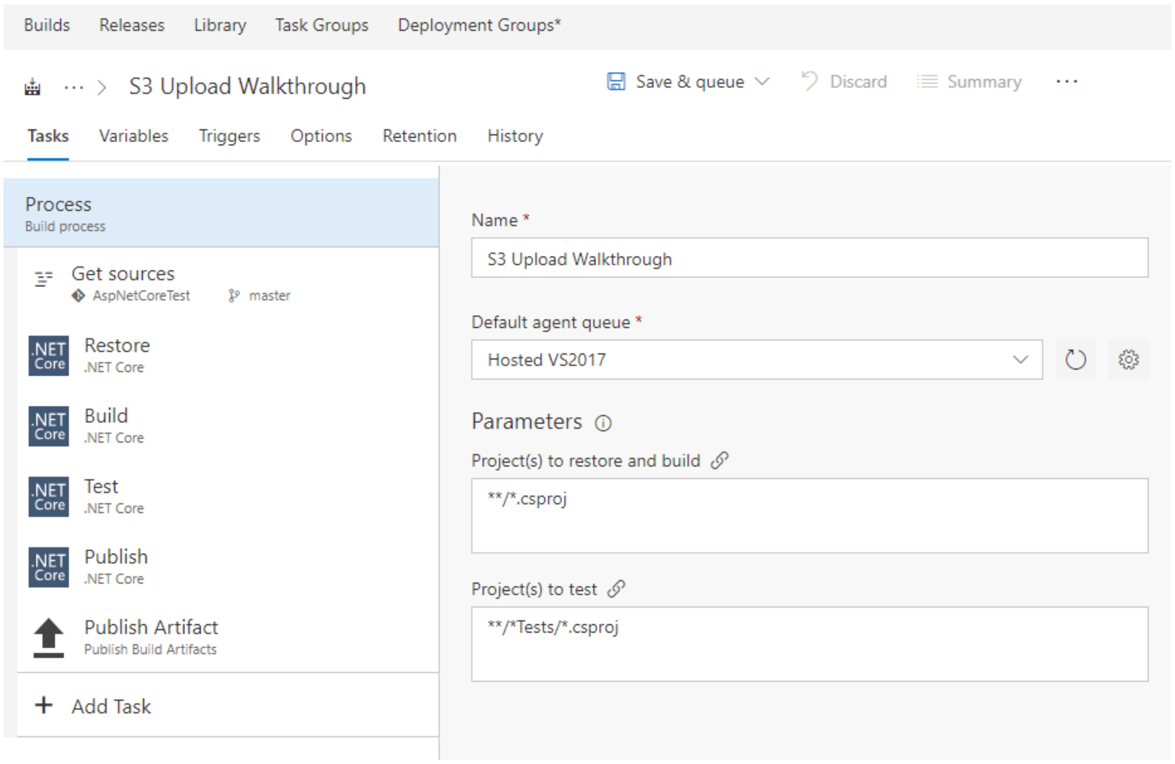
The following tutorial demonstrates how to use the *AWS S3 Upload* task to upload archival data to an Amazon Simple Storage Service (Amazon S3) bucket from a Visual Studio Team Services (VSTS) build definition.

Prerequisites

- The AWS Tools for VSTS installed in VSTS or an on-premises Team Foundation Server.
- An AWS account and preferably an associated IAM user account.
- An S3 bucket.

Archiving Build Artifacts with the AWS S3 Upload Task

This walkthrough assumes you are using a build based on the ASP.NET Core template which contains the following default tasks.



Name *
S3 Upload Walkthrough

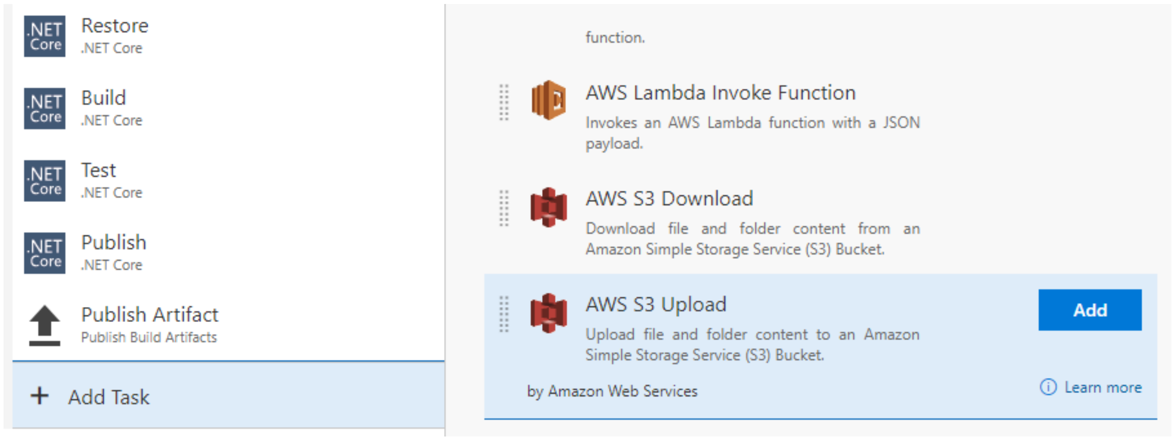
Default agent queue *
Hosted VS2017

Project(s) to restore and build
**/*.csproj

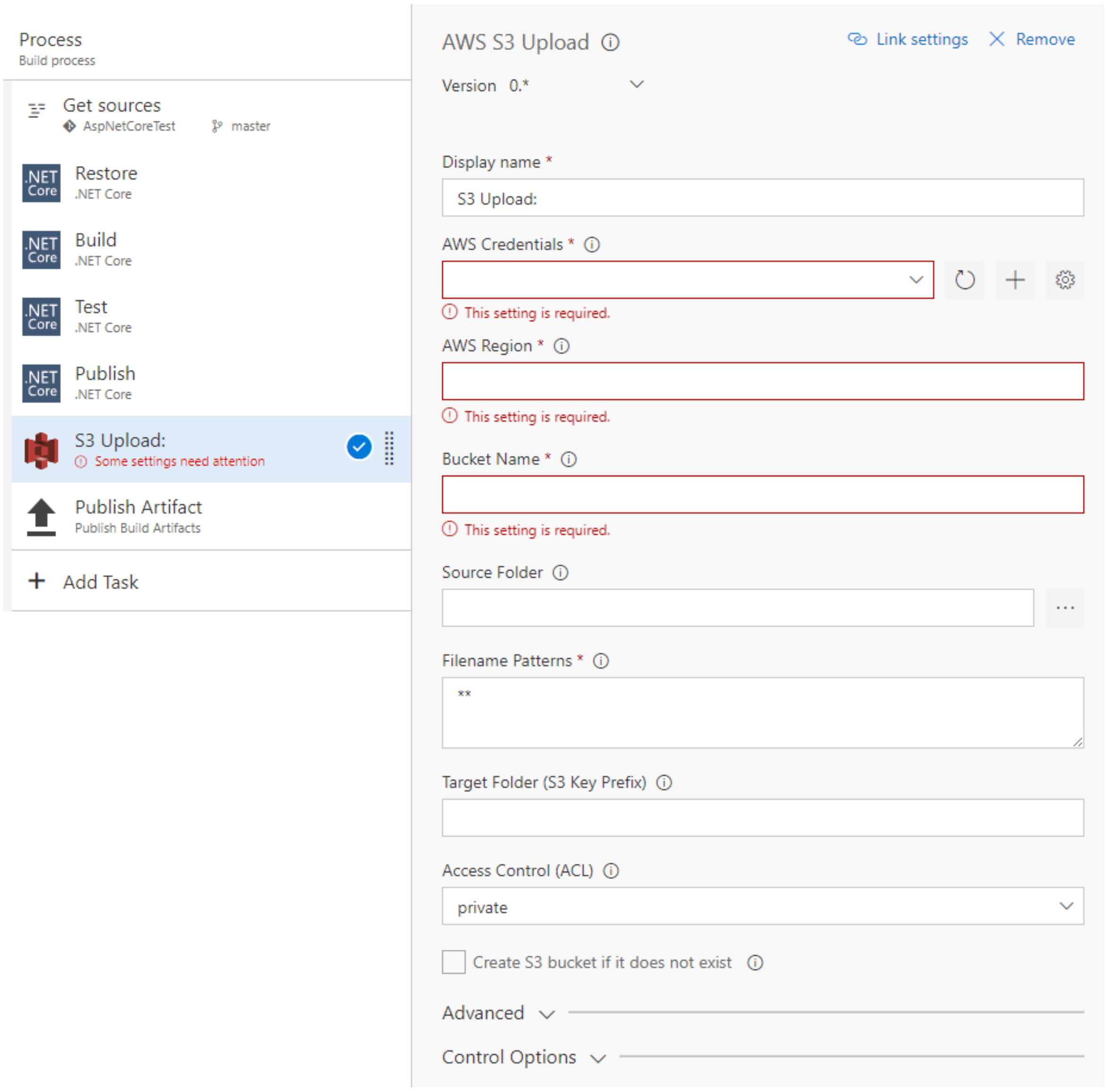
Project(s) to test
**/*Tests/*.csproj

Add the S3 Upload Task to the Build Definition

To capture the build output produced by the *Publish* task and upload it to Amazon S3 you need to add the *AWS S3 Upload* task between the existing *Publish* and *Publish Artifacts* tasks. Click the **Add Task** link. In the right hand panel, scroll through the available tasks until you see the AWS S3 Upload task. Click the **Add** button to add it to the build definition.


Restore
.NET Core
Build
.NET Core
Test
.NET Core
Publish
.NET Core
Publish Artifact
Publish Build Artifacts
Add Task
function.
AWS Lambda Invoke Function
Invokes an AWS Lambda function with a JSON payload.
AWS S3 Download
Download file and folder content from an Amazon Simple Storage Service (S3) Bucket.
AWS S3 Upload
Upload file and folder content to an Amazon Simple Storage Service (S3) Bucket.
Add
by Amazon Web Services
Learn more

If the new task is not added immediately after the *Publish* task, drag and drop it into position.



Display name * S3 Upload:

AWS Credentials *

This setting is required.

AWS Region *

This setting is required.

Bucket Name *

This setting is required.

Source Folder

Filename Patterns * **

Target Folder (S3 Key Prefix)

Access Control (ACL) private

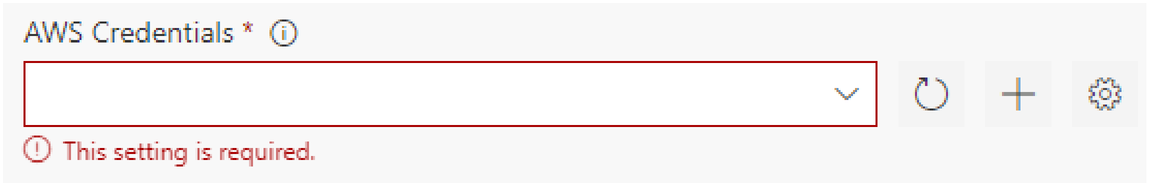
Create S3 bucket if it does not exist

Click on the new task and you will see the properties for it in the right pane.

Configure the Task Properties

For the new task you need to make the following configurations changes.

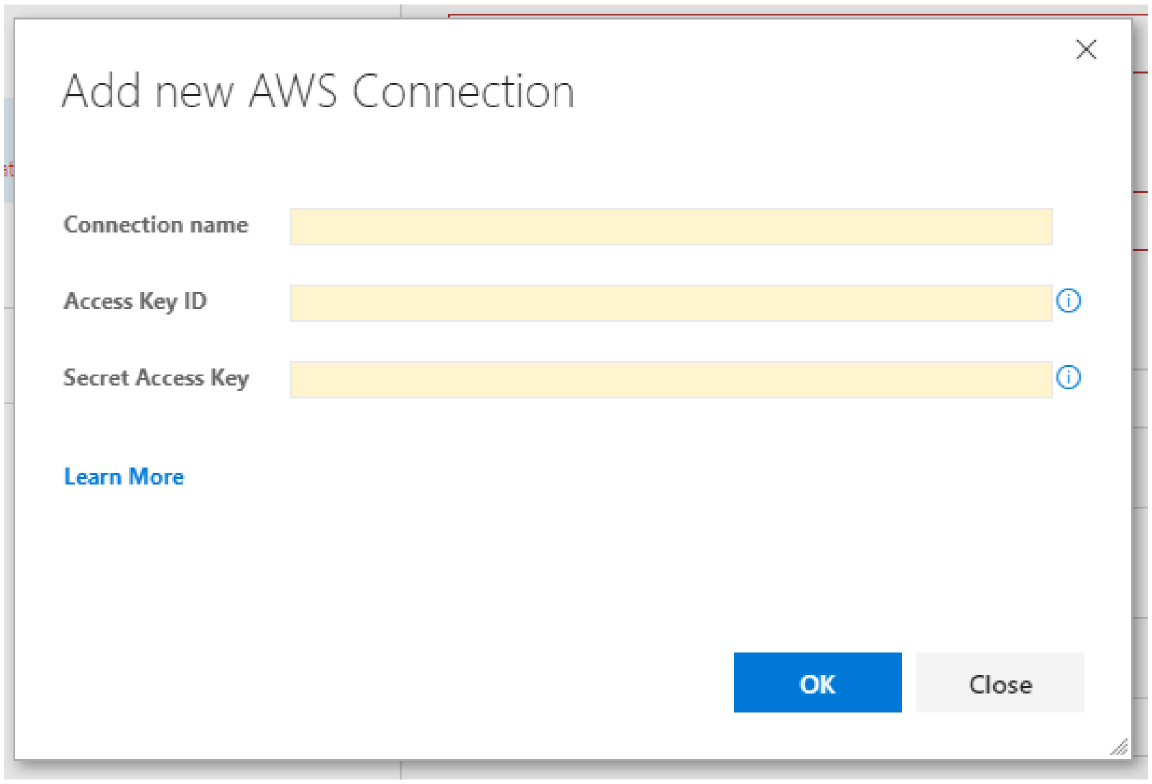
- AWS Credentials: If you have existing AWS credentials configured for your tasks you can select them using the dropdown link in the field. If not, to quickly add credentials for this task, click the **+** link.



AWS Credentials *

This setting is required.

This opens the **Add new AWS Connection** form.


Add new AWS Connection

Connection name

Access Key ID

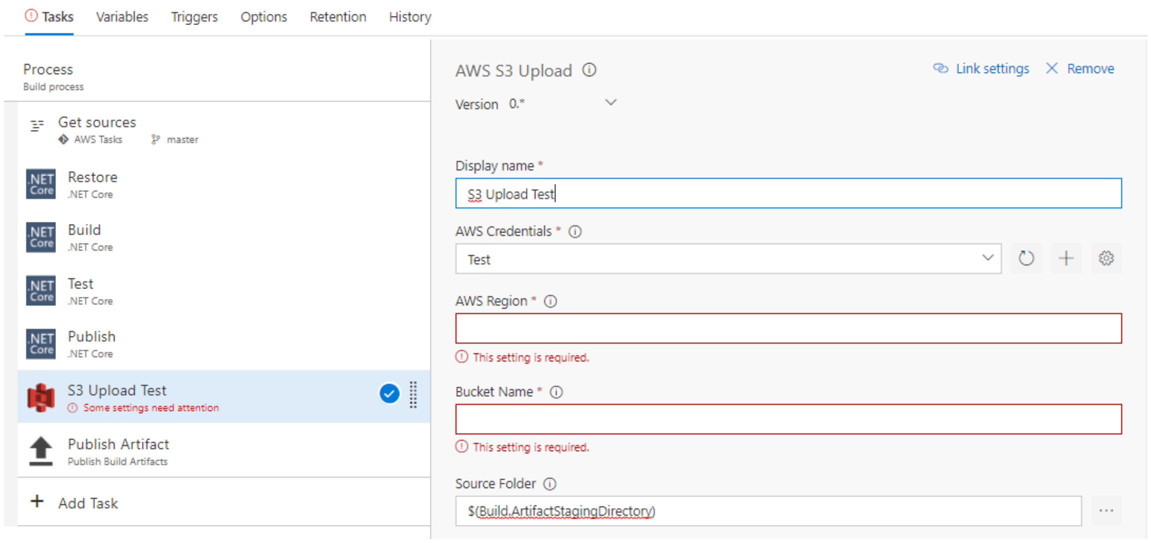
Secret Access Key

This task requires credentials for a user with a policy enabling the user to put objects to S3. If the create S3 bucket option is enabled you also need permission to create a bucket. Enter the access key and secret keys for the credentials you want to use and assign a name that you will remember.

Note

We recommend that you do not use your account's root credentials. Instead, create one or more IAM users, and then use those credentials. For more information, see Best Practices for Managing AWS Access Keys.

Click **OK** to save them. The dialog will close and return to the AWS S3 Upload Task configuration with the new credentials selected.



AWS S3 Upload

Version 0.*

Display name * S3 Upload Test

AWS Credentials * Test

AWS Region *

This setting is required.

Bucket Name *

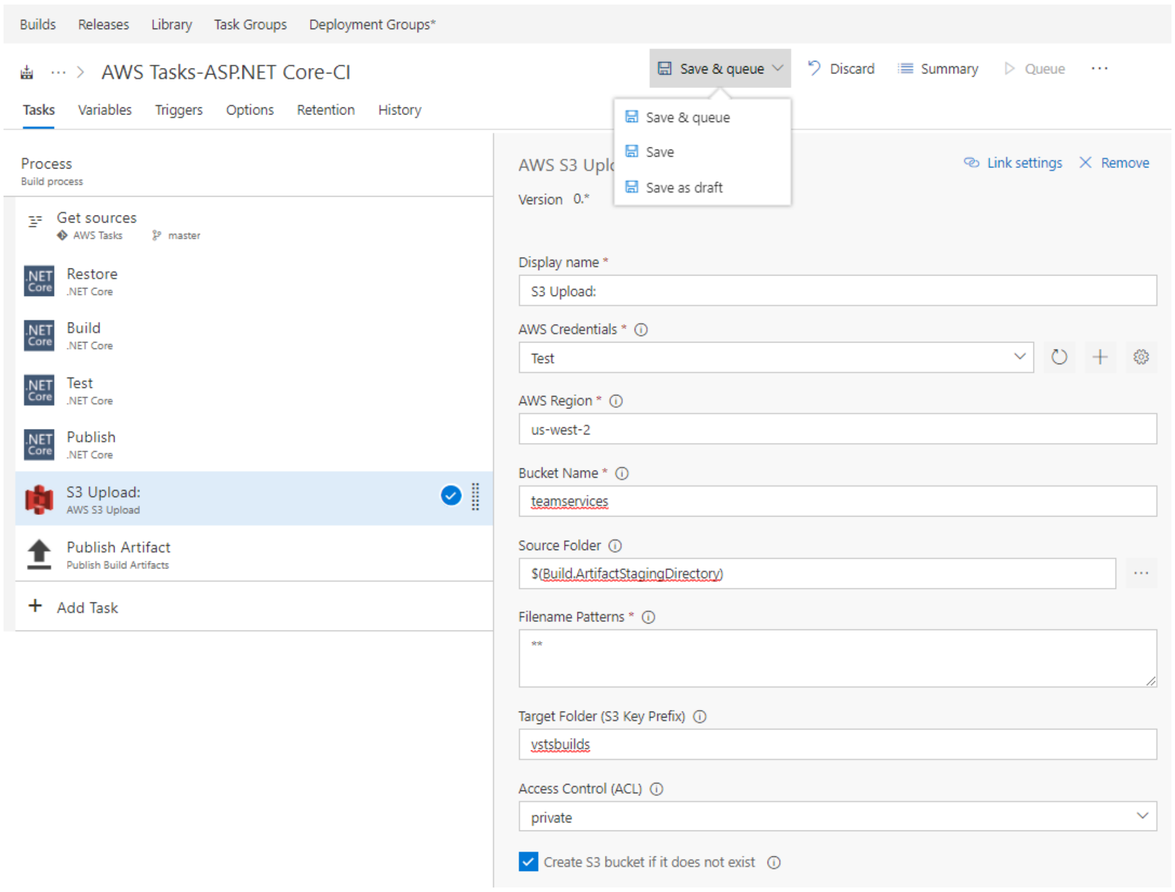
This setting is required.

Source Folder $(Build.ArtifactStagingDirectory)

- Set the region in which the bucket exists or will be created in, for example 'us-east-1', 'us-west-2' etc.
- Enter the name of the bucket (bucket names must be globally unique).
- The **Source Folder** points to a folder in your build area that contains the content to be uploaded. Team Services provides a number of variables, detailed here, that you can use to avoid hard-coded paths. For this walk-through use the variable **Build.ArtifactStagingDirectory**, which is defined as *the local path on the agent where artifacts are copied to before being pushed to their destination*.
- **Filename Patterns** can contain one or more globbing patterns used to select files under the **Source Folder** for upload. The default value shown here selects all files recursively. Multiple patterns can be specified, one per line. For this walk-through, the preceeding task (*Publish*) emits a zip file containing the build which is the file that will be uploaded.
- **Target Folder** is the *key prefix* in the bucket that will be applied to all of the uploaded files. You can think of this as a folder path. If no value is given the files are uploaded to the root of the bucket. Note that by default the relative folder hierarchy is preserved.
- **There are 3 additional options that can be set:**
  - Create S3 bucket if it does not exist. The task will fail if the bucket cannot be created.
  - Overwrite (in the Advanced section) - this is selected by default.
  - Flatten folders (also in Advanced section). Flattens the folder structure and copies all files into the specified target folder in the bucket, removing their relative paths to the source folder.

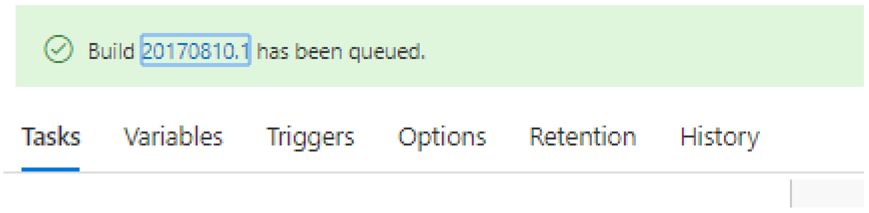
Run the Build

With the new task configured you are ready to run the build. Click the Save and queue option.

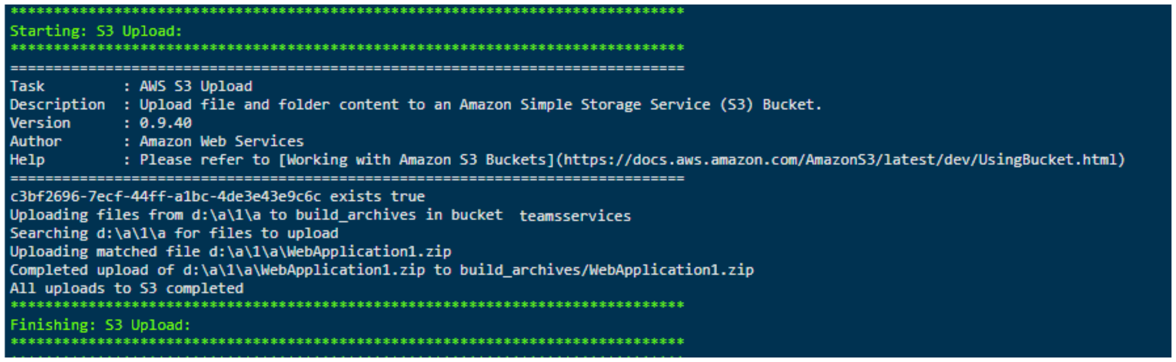


AWS S3 Upl
Version 0.*
Display name * S3 Upload:
AWS Credentials * Test
AWS Region * us-west-2
Bucket Name * teamservices
Source Folder $(Build.ArtifactStagingDirectory)
Filename Patterns * **
Target Folder (S3 Key Prefix) vstsbuilds
Access Control (ACL) private
✓ Create S3 bucket if it does not exist

During the build you can view the log by clicking on the build number in the queue message.



When the build has completed you will be able to see the S3 upload logs.



```
******************************************************************************
Starting: S3 Upload:
******************************************************************************
==============================================================================
Task         : AWS S3 Upload
Description  : Upload file and folder content to an Amazon Simple Storage Service (S3) Bucket.
Version      : 0.9.40
Author       : Amazon Web Services
Help         : Please refer to [Working with Amazon S3 Buckets](https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingBucket.html)
==============================================================================
c3bf2696-7ecf-44ff-a1bc-4de3e43e9c6c exists true
Uploading files from d:\a\1\a to build_archives in bucket  teamsservices
Searching d:\a\1\a for files to upload
Uploading matched file d:\a\1\a\WebApplication1.zip
Completed upload of d:\a\1\a\WebApplication1.zip to build_archives/WebApplication1.zip
All uploads to S3 completed
******************************************************************************
Finishing: S3 Upload:
******************************************************************************
```

That completes the walk-through. As you have seen using the new AWS tasks is easy to do. Consider expanding the project and adding other AWS tasks.

Deploying an ASP.NET Web App to AWS

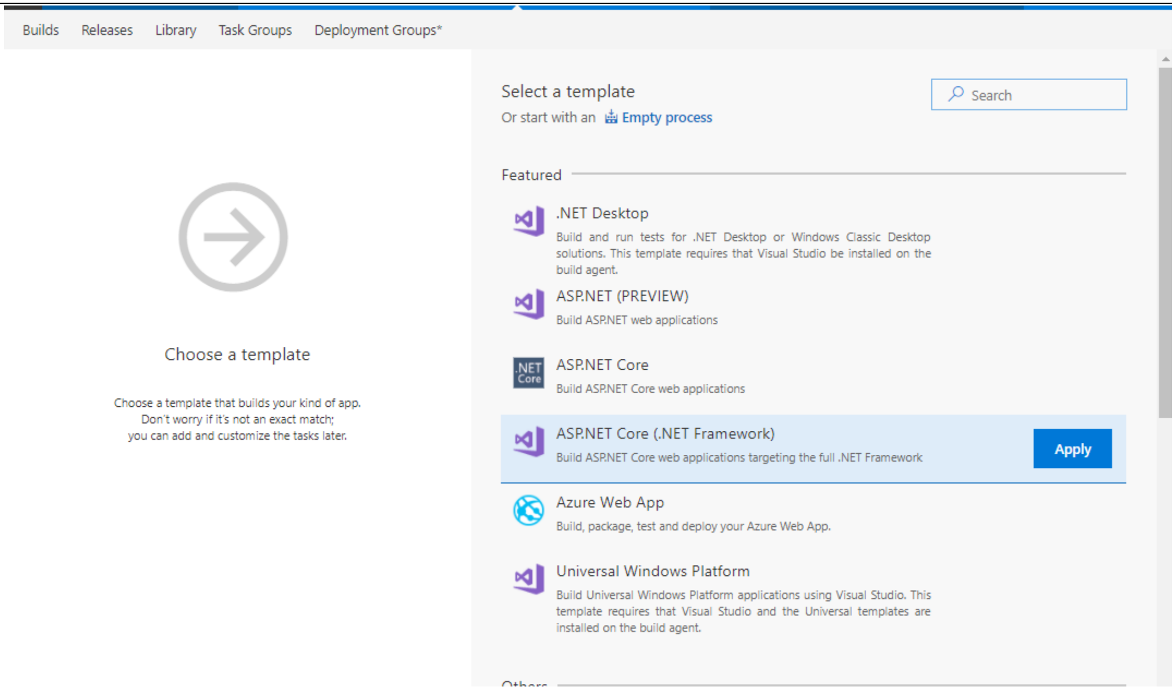
The following tutorial demonstrates how to use the *AWS Elastic Beanstalk Deployment* task to deploy a web application to the AWS Cloud from a Visual Studio Team Services (VSTS) build definition.

Prerequisites

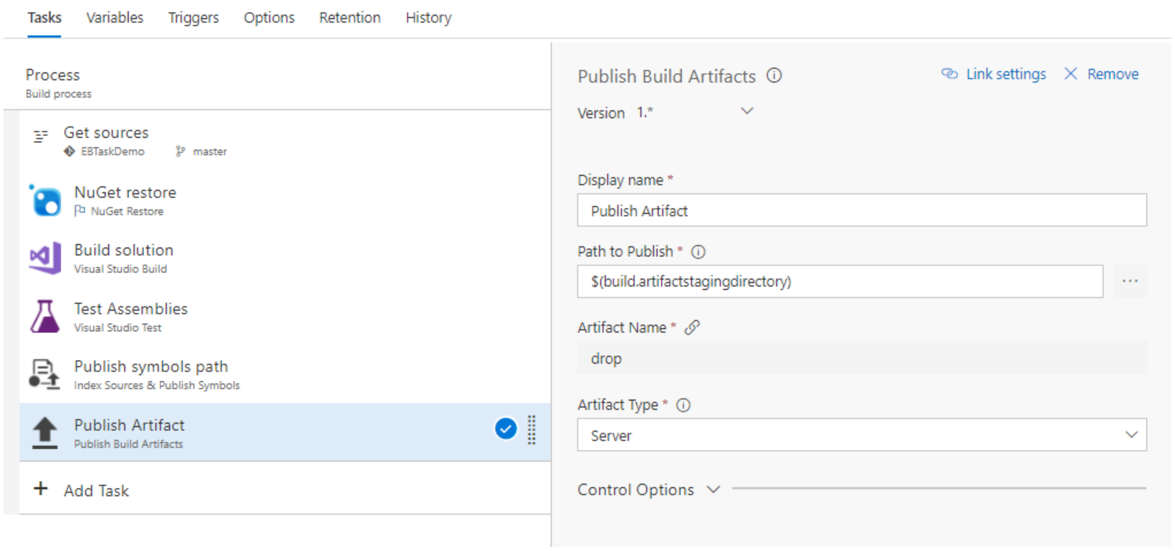
- The AWS Tools for VSTS installed in VSTS or an on-premises Team Foundation Server.
- An AWS account and preferably an associated IAM user account.
- An Elastic Beanstalk application and environment.

Deploying an ASP.NET Application Using the AWS Elastic Beanstalk Deployment Task

This walkthrough assumes you are using a build based on the ASP.NET Core (.NET Framework) template that will produce a Web Deploy archive for deployment.


Builds
Releases
Library
Task Groups
Deployment Groups*
Select a template
Search
Or start with an Empty process
Featured
.NET Desktop
Build and run tests for .NET Desktop or Windows Classic Desktop solutions. This template requires that Visual Studio be installed on the build agent.
ASP.NET (PREVIEW)
Build ASP.NET web applications
ASP.NET Core
Build ASP.NET Core web applications
ASP.NET Core (.NET Framework)
Build ASP.NET Core web applications targeting the full .NET Framework
Apply
Azure Web App
Build, package, test and deploy your Azure Web App.
Universal Windows Platform
Build Universal Windows Platform applications using Visual Studio. This template requires that Visual Studio and the Universal templates are installed on the build agent.
Choose a template
Choose a template that builds your kind of app.
Don't worry if it's not an exact match;
you can add and customize the tasks later.

The build process page containing the default tasks for the template is displayed.



Version 1.*

Display name * Publish Artifact

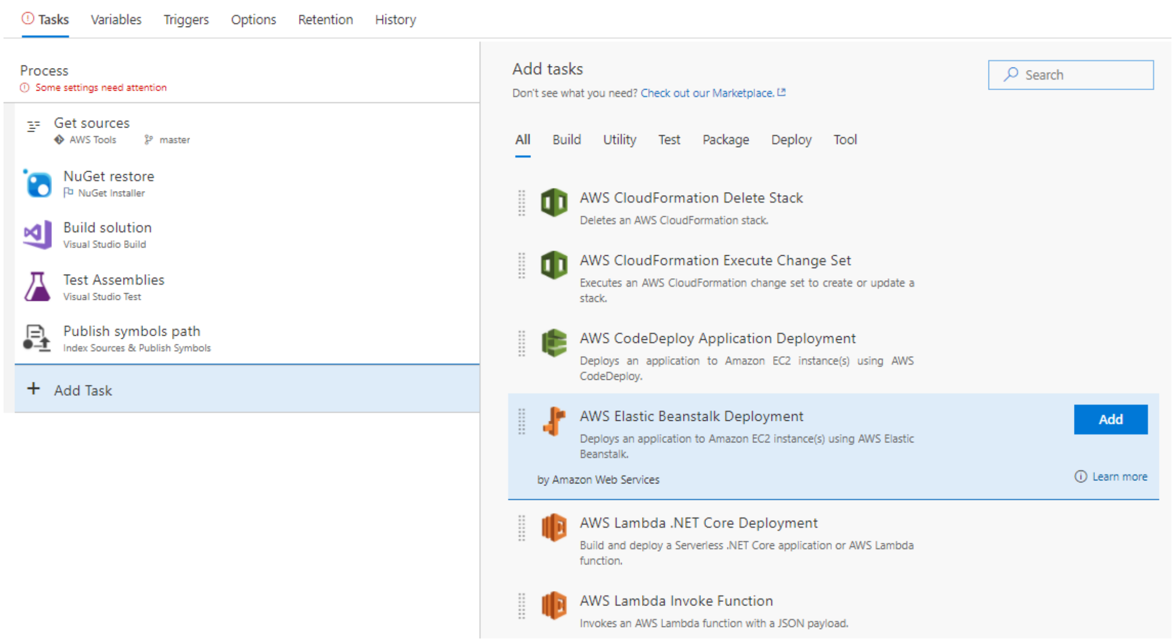
Path to Publish * $(build.artifactstagingdirectory)

Artifact Name * drop

Artifact Type * Server

Add the AWS Elastic Beanstalk Deployment Task to the Build Definition

Click the **Add Task** link. In the right hand panel, scroll through the available tasks until you see the *AWS Elastic Beanstalk Deployment* task. Click the **Add** button to add it to bottom of the build definition.



Tasks Variables Triggers Options Retention History

Process
Some settings need attention

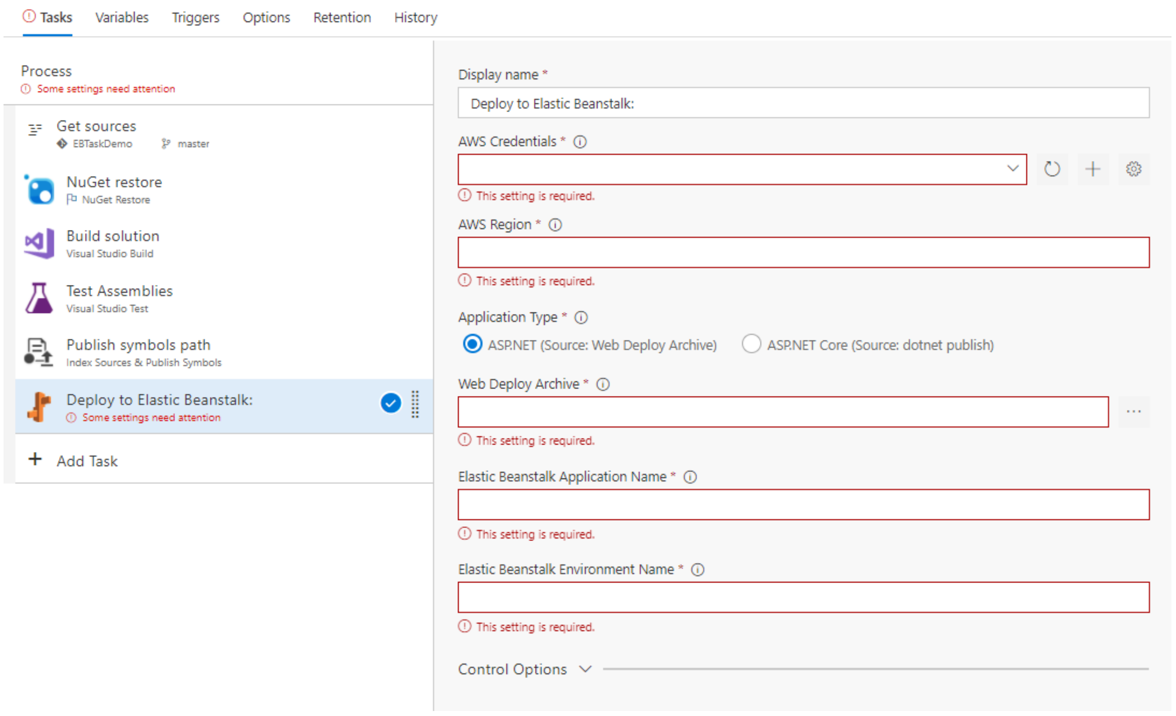
- Get sources — AWS Tools, master
- NuGet restore — NuGet Installer
- Build solution — Visual Studio Build
- Test Assemblies — Visual Studio Test
- Publish symbols path — Index Sources & Publish Symbols
- Add Task

Add tasks
Don't see what you need? Check out our Marketplace.

All Build Utility Test Package Deploy Tool

- AWS CloudFormation Delete Stack — Deletes an AWS CloudFormation stack.
- AWS CloudFormation Execute Change Set — Executes an AWS CloudFormation change set to create or update a stack.
- AWS CodeDeploy Application Deployment — Deploys an application to Amazon EC2 instance(s) using AWS CodeDeploy.
- AWS Elastic Beanstalk Deployment — Deploys an application to Amazon EC2 instance(s) using AWS Elastic Beanstalk. by Amazon Web Services — Add — Learn more
- AWS Lambda .NET Core Deployment — Build and deploy a Serverless .NET Core application or AWS Lambda function.
- AWS Lambda Invoke Function — Invokes an AWS Lambda function with a JSON payload.

Click on the new task and you will see the properties for it in the right pane.



Tasks Variables Triggers Options Retention History

Process
Some settings need attention

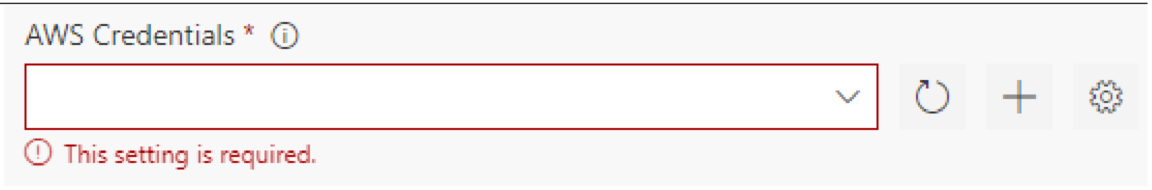
- Get sources — EBTaskDemo, master
- NuGet restore — NuGet Restore
- Build solution — Visual Studio Build
- Test Assemblies — Visual Studio Test
- Publish symbols path — Index Sources & Publish Symbols
- Deploy to Elastic Beanstalk: — Some settings need attention
- Add Task

Display name *: Deploy to Elastic Beanstalk:
AWS Credentials *: This setting is required.
AWS Region *: This setting is required.
Application Type *: [✓ ASP.NET (Source: Web Deploy Archive)] ASP.NET Core (Source: dotnet publish)
Web Deploy Archive *: This setting is required.
Elastic Beanstalk Application Name *: This setting is required.
Elastic Beanstalk Environment Name *: This setting is required.
Control Options

Configure the Task Properties

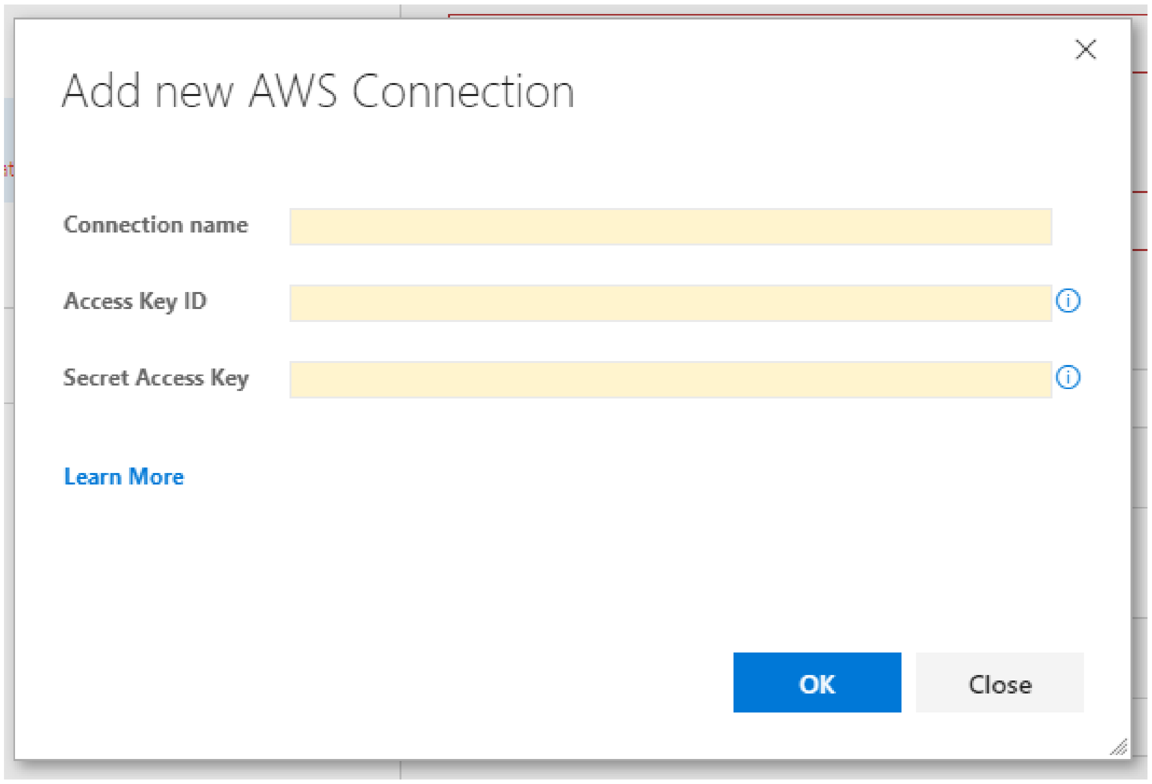
For the new task you need to make the following configuration changes.

- AWS Credentials: If you have existing AWS credentials configured for your tasks you can select them using the dropdown link in the field. If not, to quickly add credentials for this task, click the **+** link.


AWS Credentials *

This setting is required.

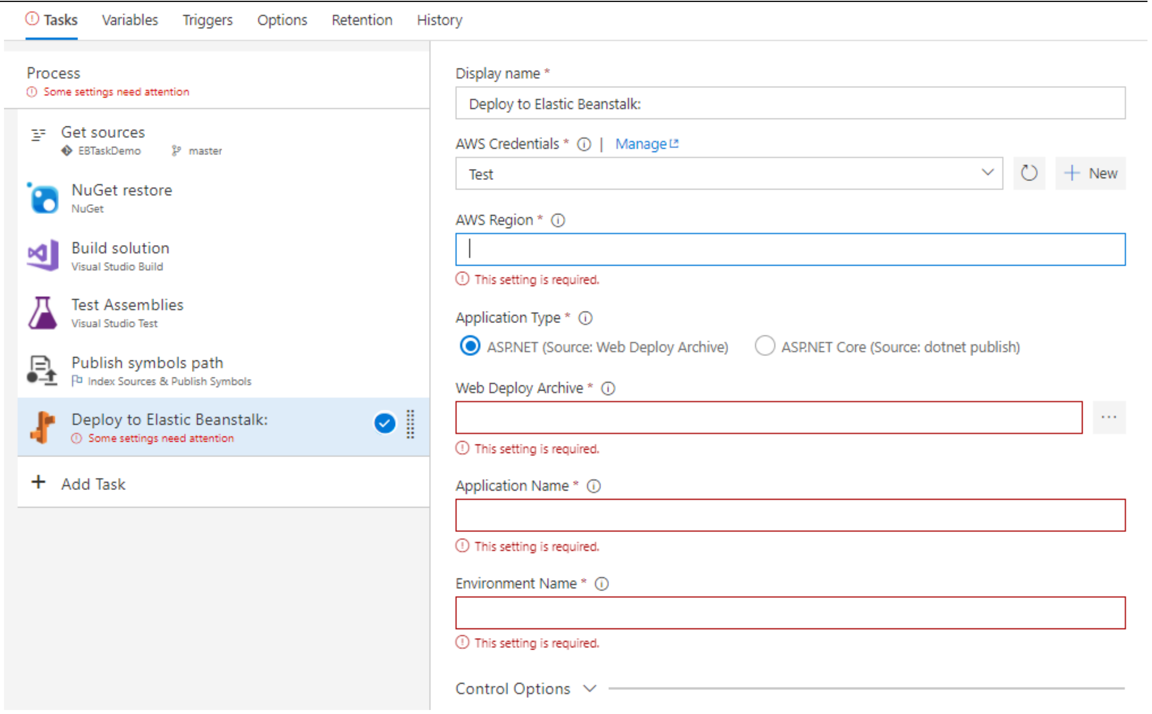
This opens the **Add new AWS Connection** form.


Add new AWS Connection

This task requires credentials for a user with a policy enabling the user to update a Beanstalk environment and describe an environment status and events. Enter the access key and secret keys for the credentials you want to use and assign a name that you will remember.

> **Note**
> We recommend that you do not use your account's root credentials. Instead, create one or more IAM users, and then use those credentials. For more information, see Best Practices for Managing AWS Access Keys.

Click **OK** to save them. The dialog will close and return to the Elastic Beanstalk Deployment Task configuration with the new credentials selected.



Display name *: Deploy to Elastic Beanstalk:

AWS Credentials *: Test

AWS Region *:
This setting is required.

Application Type *: [✓ ASP.NET (Source: Web Deploy Archive)] ASP.NET Core (Source: dotnet publish)

Web Deploy Archive *:
This setting is required.

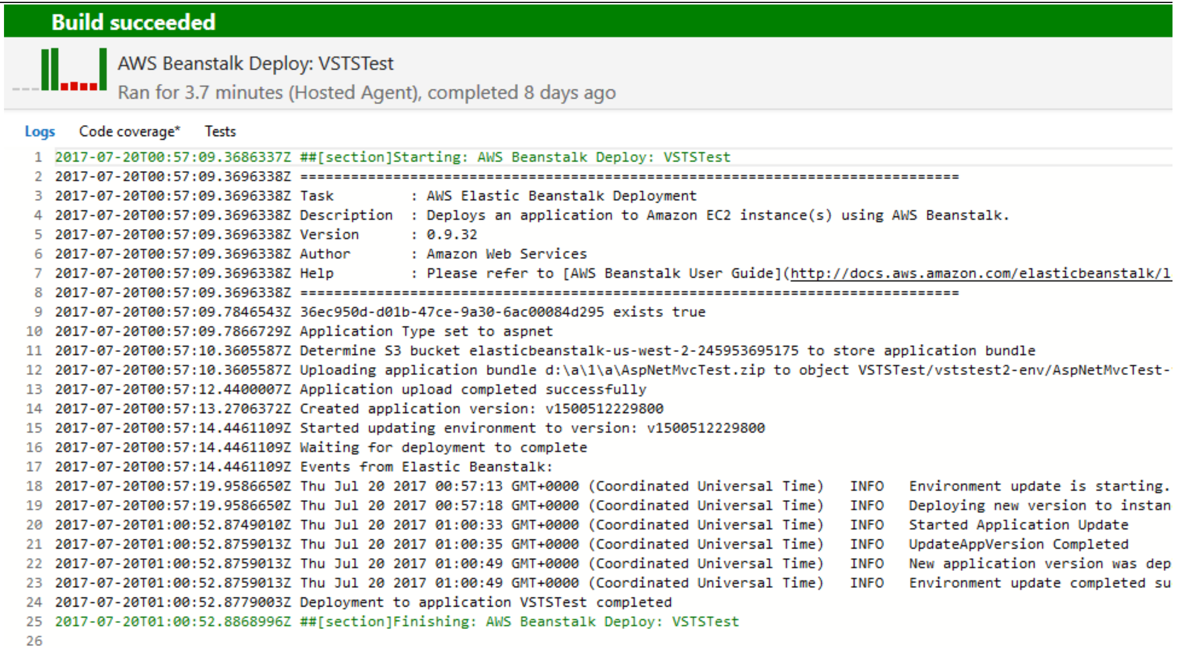
Application Name *:
This setting is required.

Environment Name *:
This setting is required.

- AWS Region: The AWS region that that the Beanstalk environment is running in.
- Application Type: Set to ASP.NET
- Web Deploy Archive: The path to the Web Deploy archive. If the archive was created using the arguments above, the file will have the same name as the directory containing the web application and will have a ".zip" extension. It can be found in the build artifacts staging directory which can be referenced as `$(build.artifactstagingdirectory)`.
- Application Name: The name you used to create the Beanstalk application. A Beanstalk application is the container for the environment for the .NET web application.
- Environment Name: The name you used to create the Beanstalk environment. A Beanstalk environment contains the actual provisioned resources that are running the .NET web application.

Run the Build

With the new task configured you are ready to run the build. Click the Save and queue option. When the build has completed running you should see a log similar to this.



Build succeeded

AWS Beanstalk Deploy: VSTSTest

Ran for 3.7 minutes (Hosted Agent), completed 8 days ago

Logs   Code coverage*   Tests

```
2017-07-20T00:57:09.3686337Z ##[section]Starting: AWS Beanstalk Deploy: VSTSTest
2017-07-20T00:57:09.3696338Z ==============================================================================
2017-07-20T00:57:09.3696338Z Task         : AWS Elastic Beanstalk Deployment
2017-07-20T00:57:09.3696338Z Description  : Deploys an application to Amazon EC2 instance(s) using AWS Beanstalk.
2017-07-20T00:57:09.3696338Z Version      : 0.9.32
2017-07-20T00:57:09.3696338Z Author       : Amazon Web Services
2017-07-20T00:57:09.3696338Z Help         : Please refer to [AWS Beanstalk User Guide](http://docs.aws.amazon.com/elasticbeanstalk/l
2017-07-20T00:57:09.3696338Z ==============================================================================
2017-07-20T00:57:09.7846543Z 36ec950d-d01b-47ce-9a30-6ac00084d295 exists true
2017-07-20T00:57:09.7866729Z Application Type set to aspnet
2017-07-20T00:57:10.3605587Z Determine S3 bucket elasticbeanstalk-us-west-2-245953695175 to store application bundle
2017-07-20T00:57:10.3605587Z Uploading application bundle d:\a\1\a\AspNetMvcTest.zip to object VSTSTest/vststest2-env/AspNetMvcTest-
2017-07-20T00:57:12.4400007Z Application upload completed successfully
2017-07-20T00:57:13.2706372Z Created application version: v1500512229800
2017-07-20T00:57:14.4461109Z Started updating environment to version: v1500512229800
2017-07-20T00:57:14.4461109Z Waiting for deployment to complete
2017-07-20T00:57:14.4461109Z Events from Elastic Beanstalk:
2017-07-20T00:57:19.9586650Z Thu Jul 20 2017 00:57:13 GMT+0000 (Coordinated Universal Time)   INFO   Environment update is starting.
2017-07-20T00:57:19.9586650Z Thu Jul 20 2017 00:57:18 GMT+0000 (Coordinated Universal Time)   INFO   Deploying new version to instan
2017-07-20T01:00:52.8749010Z Thu Jul 20 2017 01:00:33 GMT+0000 (Coordinated Universal Time)   INFO   Started Application Update
2017-07-20T01:00:52.8759013Z Thu Jul 20 2017 01:00:35 GMT+0000 (Coordinated Universal Time)   INFO   UpdateAppVersion Completed
2017-07-20T01:00:52.8759013Z Thu Jul 20 2017 01:00:49 GMT+0000 (Coordinated Universal Time)   INFO   New application version was dep
2017-07-20T01:00:52.8759013Z Thu Jul 20 2017 01:00:49 GMT+0000 (Coordinated Universal Time)   INFO   Environment update completed su
2017-07-20T01:00:52.8779003Z Deployment to application VSTSTest completed
2017-07-20T01:00:52.8868996Z ##[section]Finishing: AWS Beanstalk Deploy: VSTSTest
```

Task Reference

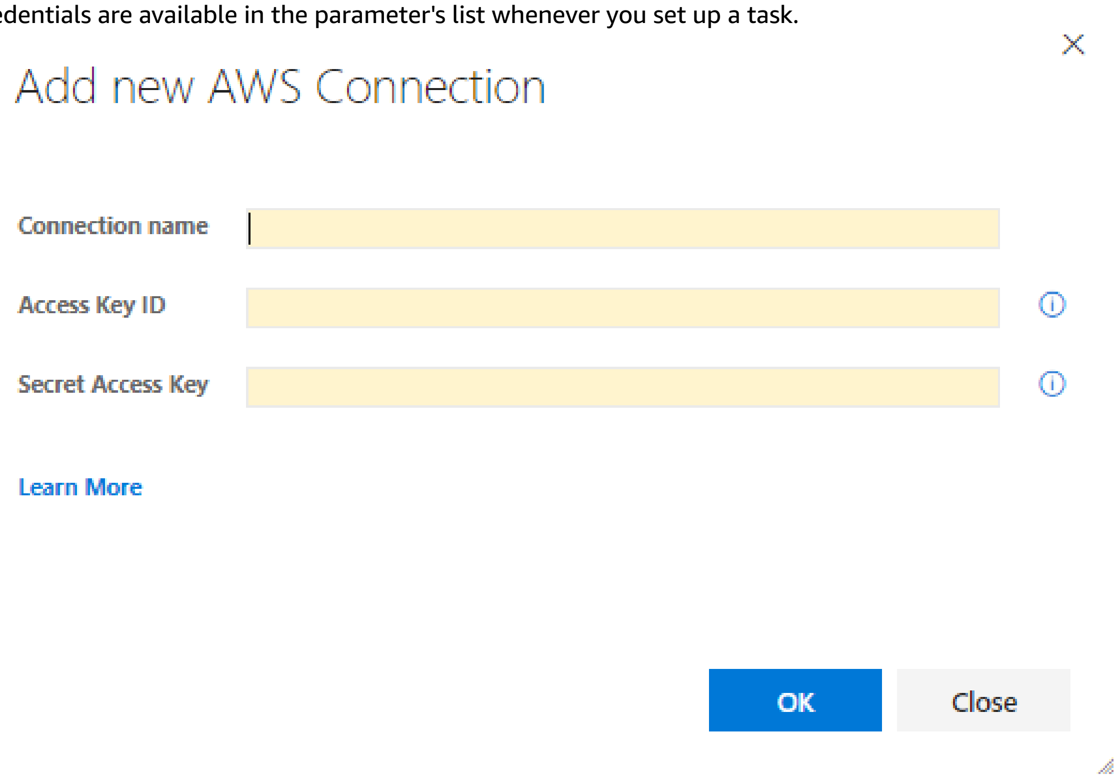
This reference describes the tasks that are included in the AWS Tools for Microsoft Visual Studio Team Services.

Prerequisites

- You must have an AWS account. For information on setting up an account, see Set up AWS Credentials for the AWS Tools for VSTS (p. 2).
- A required parameter for each task is `AWS Credentials`. If you haven't created an AWS Connection, choose the **+** to the right of the parameter. A dialog box opens asking you to enter your AWS access key and secret key credentials and to provide a name. We do not recommend using your root credentials. Instead, use the credentials associated with an IAM user account.

  The secret key text is automatically masked. Choose **OK** to save the credentials into your VSTS account and return to the configuration of the new task.

  The name for the new credentials is entered into the **AWS Credentials** box. Once created, those credentials are available in the parameter's list whenever you set up a task.



Topics

- AWS CodeDeploy Deployment Application Task (p. 24)
- Amazon Elastic Container Registry Push Image (p. 26)
- AWS Elastic Beanstalk Deployment Task (p. 27)
- AWS Lambda Deployment Task (p. 28)
- AWS Lambda .NET Core Deployment Task (p. 31)
- AWS Lambda Invoke Function Task (p. 32)
- AWS S3 Download Task (p. 33)
- AWS S3 Upload Task (p. 35)
- AWS Systems Manager Get Parameter (p. 36)
- AWS Systems Manager Run Command (p. 38)
- AWS Send Message Task (p. 40)

AWS CLI

Synopsis

Runs a command using the AWS CLI. Note that you must have the AWS CLI installed to use this task. See Installing the AWS Command Line Interface for more details.

Description

The AWS CLI uses a multipart structure on the command line. It starts with the base call to AWS. The next part specifies a top-level command, which often represents an AWS service that the AWS CLI supports. Each AWS service has additional subcommands that specify the operation to perform. You can specify the general AWS CLI options, or the specific parameters for an operation, in any order on the command line. If you specify an exclusive parameter multiple times, only the last value applies.

```
<command> <subcommand> [options and parameters]
```

Parameters can take various types of input values such as numbers, strings, lists, maps, and JSON structures.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS CLI. You can rename it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Command*

The AWS CLI command to run. Run `aws help` in the AWS Command Line Interface to get a complete list of commands, or see CommandStructure in the AWS Command Line Interface.

Subcommand

The AWS CLI subcommand to run. Run `aws help` in the AWS Command Line Interface to get a complete list of commands, or see CommandStructure in the AWS Command Line Interface.

Options and Parameters

The arguments to pass to the AWS CLI command. Run `aws <command> --help` in the AWS Command Line Interface to get the complete list of arguments supported by the command.

Advanced

Fail on Standard Error

If true, this task fails if any errors are written to the StandardError stream.

AWS Tools for Windows PowerShell Script Task

Synopsis

Runs a PowerShell script that uses cmdlets from the AWS Tools for Windows PowerShell module. The module is automatically installed if it isn't already available in the environment.

Description

This task accepts a PowerShell command or script that uses cmdlets from the Tools for Windows PowerShell module to interact with AWS services. You can specify the script to run via its file name, or you can enter it into the task configuration. Before running the supplied script, the task tests to see if the required Tools for Windows PowerShell module is already installed. If it isn't installed, the latest available version from the PowerShell Gallery is downloaded and installed.

Note
If an installation is performed, the module is installed in the `current user` scope. The location is compatible with automatic module load. As a result, you don't need to import the module in your script.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Name*

The default name of the task, AWS Tools for Windows PowerShell Script.

AWS Credentials*

The AWS credentials used by the cmdlets in the module. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The default AWS Region that the cmdlets assume in the module. AWS cmdlets invoked without a –`Region` parameter automatically use this value. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Arguments

Optional arguments to pass to the script. You can use ordinal or named parameters.

Script Source*

The type of script to run. Choose **Script File** to run a script that is contained in a file. Choose **Inline Script** to enter the script to run in the task configuration.

Script Path*

Required if the `Script Source` parameter is set to **Script File**. Specify the full path to the script you want to run.

Inline Script*

Required if the `Script Source` parameter is set to **Inline Script**. Enter the text of the script to run.

ErrorActionPreference

Prepends the line *$ErrorActionPreference = 'VALUE'* at the top of your script.

Advanced

Fail on Standard Error

If this option is selected, the task will fail if any errors are written to the error pipeline, or if any data is written to the Standard Error stream. Otherwise, the task relies on the exit code to determine failure.

Ignore $LASTEXITCODE

If this option is not selected, the line *if ((Test-Path -LiteralPath variable:\LASTEXITCODE)) { exit $LASTEXITCODE }* is appended to the end of your script. This causes the last exit code from an external command to propagate as the exit code of PowerShell. Otherwise, the line is not appended to the end of your script.

Working Directory

The working directory where the script runs.

AWS CloudFormation Create-Update Stack Task

Synopsis

Creates a new AWS CloudFormation stack or updates the stack if it exists.

Description

Creates or updates a stack based on the specified parameters. When you need to change a stack's settings or its resources, update the stack instead of deleting it and creating a new stack.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, Create/Update Stack. You can rename it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Stack Name*

The name associated with the stack. The name must be unique in the region in which you are creating the stack.

A stack name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and cannot be longer than 128 characters.

Template Source*

Specifies the location of the template to use to create or update the stack. You can specify the template using the path to a file in the local file system, a URL to the file, or an object in Amazon S3. If you select an object in Amazon S3, you can specify the bucket and object name (key).

Note that CloudFormation limits the size of template files uploaded to the service to 51,200 bytes. If your template is larger than the allowed size you should choose either the URL or Amazon S3 location options. You can also specify a bucket name for the local file option. If a bucket name is specified, the template is uploaded to the bucket by the task. The object key will be the template filename, less any path.

When the task uploads the template to a bucket or you specify an Amazon S3 bucket name and object key, the task generates a URL to the object and supplies the URL to CloudFormation.

Template File*

The path to the template file for the stack. For more information, see Template Anatomy in the *AWS CloudFormation User Guide*.

Template Parameters File

The path to the file containing the template parameters.

Create or Update the Stack Using a Change Set

If checked, a change set containing a list of changes to apply to a stack will be created and then validated. If the changes validate successfully, the change set can be executed to make the changes. You can choose to use a change set to create a new stack or update an existing stack.

Default: not checked.

Change Set Name

This parameter is required if the option to use a change set is selected. Specifies the name of the change set to create, validate, and (optionally) execute to create or update the stack.

Description

Optional description for the change set.

Automatically Execute the Change Set

If checked, the change set is automatically executed when validation succeeds. If it isn't checked the change set is validated but not executed. You can execute the change set later by using the `|CFNlong| Execute Change Set` task.

Default: checked.

Capabilities

You must specify capabilities before AWS CloudFormation can update certain stacks. Some stack templates might include resources that can affect permissions in your AWS account by, for example, creating new AWS Identity and Access Management (IAM) users. For those stacks, you must explicitly acknowledge their capabilities by specifying this parameter.

Create or Update IAM Resources ('CAPABILITY_IAM')

If your stack manipulates IAM resources, you can specify either capability. Otherwise, an `InsufficientCapabilities` error is returned.

Default: checked.

Create or Update Named IAM Resources ('CAPABILITY_NAMED_IAM')

If your stack manipulates IAM resources with custom names, you must add this capability. Otherwise, an `InsufficientCapabilities` error is returned.

Default: checked.

Advanced

Role ARN

The Amazon Resource Name (ARN) of an IAM role that AWS CloudFormation assumes when it executes the change set. AWS CloudFormation uses the role's credentials to make calls on your behalf. AWS CloudFormation uses this role for all future operations on the stack. As long as users have permission to operate on the stack, AWS CloudFormation uses this role even if the users don't have permission to pass it.

Ensure that the role grants least privilege.

If you don't specify a value, AWS CloudFormation uses the role that was previously associated with the stack. If no role is available, AWS CloudFormation uses a temporary session that is generated from your user credentials.

Resource Types

The template resource types that you have permissions to work with if you execute this change set. For example, `AWS::EC2::Instance`, `AWS::EC2::*`, or `Custom::MyCustomInstance`.

If the list of resource types doesn't include a resource type that you're updating, the stack update fails. By default, AWS CloudFormation grants permissions to all resource types. IAM uses this parameter for condition keys in IAM policies for AWS CloudFormation.

For more information, see Controlling Access with AWS Identity and Access Management in the *AWS CloudFormation User Guide*.

Notification ARNs

The ARNs of Amazon SNS topics that AWS CloudFormation associates with the stack. To remove all associated notification topics, specify an empty list.

Tags

Collection of tags to apply to the resources created by your template. Tags can be specified as *tagkey=tagvalue*, one per line.

Options

On Failure

Determines what action to take if stack creation fails.

Default: *ROLLBACK*.

Disable Rollback

If checked, disables rollback of the stack if stack creation failed. You can specify `DisableRollback` or `OnFailure`, but not both.

Default: not checked.

Output Variable

The name of the variable that will contain the stack ID on task completion. You can use `$(variableName)` to refer to the stack ID in subsequent tasks.

AWS CloudFormation Delete Stack Task

Synopsis

Deletes an AWS CloudFormation stack.

Description

Deletes the specified stack.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS CloudFormation Delete Stack. You can rename it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Stack Name*

The name or unique stack ID that is associated with the stack.

AWS CloudFormation Execute Change Set Task

Synopsis

Executes an AWS CloudFormation change set to create or update a stack.

Description

When you execute a change set, AWS CloudFormation deletes all other change sets associated with the stack because they aren't valid for the updated stack.

AWS CloudFormation updates a stack using the input information that was provided when the specified change set was created.

If a stack policy is associated with the stack, AWS CloudFormation enforces the policy during the update. You can't specify a temporary stack policy that overrides the current policy.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS CloudFormation Execute Change Set. You can rename it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Change Set Name*

The name or Amazon Resource Name (ARN) of the change set that you want to execute.

Stack Name

The stack name or ARN of the stack that is associated with the change set. This value is required if you specify the name of a change set to execute. If the change set ARN is specified, this field is optional.

The name must be unique in the region in which you are creating the stack. A stack name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and cannot be longer than 128 characters.

AWS CodeDeploy Deployment Application Task

Synopsis

Deploys an application to Amazon EC2 instances by using AWS CodeDeploy.

Description

This can be a variety of application content, such as code, web and configuration files, executable files, packages, scripts, and multimedia files.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS CodeDeploy Deployment. You can rename it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Application Name*

The name of the AWS CodeDeploy application.

Deployment Group Name*

The name of the deployment group the revision is deployed to.

Revision Bundle*

The location of the application revision artifacts to deploy. You can supply a filename or folder. If a folder is supplied the task will recursively zip the folder contents into an archive file before uploading the archive to Amazon S3. If a filename is supplied the task uploads it unmodified to Amazon S3. CodeDeploy requires the appspec.yml file describing the application to exist at the root of the specified folder or archive file.

Bucket Name*

The name of the bucket to which the revision bundle is uploaded.

Target Folder

Optional folder (key prefix) for the uploaded revision bundle in the bucket. If not specified the, bundle is uploaded to the root of the bucket.

Description

Optional description for the deployment.

Existing File Behavior

How AWS CodeDeploy should handle files that already exist in a deployment target location but weren't part of the previous successful deployment.

Advanced

Update Outdated Instances Only

If checked, deploys to only those instances that are not running the latest application revision.

Default: not checked.

Ignore Application Stop Failures

When checked, if the deployment causes the ApplicationStop deployment lifecycle event to an instance to fail, the deployment to that instance is not considered failed at that point. It continues on to the BeforeInstall deployment lifecycle event.

Default: not checked.

Output

Output Variable

The name of the variable that will contain the deployment ID on task completion. You can use the variable $(variableName) to refer to the function result in subsequent tasks.

Amazon Elastic Container Registry Push Image

Synopsis

Pushes a Docker image identified by name, with optional tag, or image ID to the Elastic Container Registry (ECR).

Description

This task pushes a Docker image to the Elastic Container Registry. The image to push can be identified using its image ID or by name, with optional tag suffix. The task handles the work of appropriately tagging the image as required by ECR and also the login process to your registry prior to executing the Docker Push command.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS Systems Manager Get Parameter. You can rename it or append the name of the associated Parameter Store parameter or parameter path to it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Image Identity*

How the image to be pushed is identified. You can select from either the image ID or the image name. If name is selected, a tag can also be specified.

Source Image Name

Required if Image Identity is set to image name. Specifies the name of the image to push.

Source Image Tag

Optional tag that can be suffixed to the image name. If a tag is not specified, 'latest' is assumed.

Source Image ID

Required if Image Identity is set to image ID. The ID of the image to push.

Target Repository Name*

The name of the repository to which the image will be pushed.

Target Repository Tag

Optional tag for the new image in the repository. If not specified, ECR will assume 'latest'.

Create repository if it does not exist

If checked, the task will check to see if the repository exists and if it does not, will attempt to create it.

Image Tag Output Variable

The name of a build variable that will be created or updated with the pushed image reference. The image tag will be of the form *aws_account_id.dkr.ecr.region.amazonaws.com/imagename*, where **imagename** is in the format *repositoryname[:tag]*

AWS Elastic Beanstalk Deployment Task

Synopsis

This task deploys an application to Amazon EC2 instances by using Elastic Beanstalk.

Description

This task deploys and scales web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.

Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, and automatic scaling to application health monitoring.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS Elastic Beanstalk Deployment. You can rename it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Application Type*

The type of application bundle to deploy. ASP.NET application deployments use Web Deploy archives. ASP.NET Core deployments are performed using the `dotnet publish` command line tool.

Web Deploy Archive

Required if `Application Type` is set to **ASP.NET**. The path to the web deploy archive containing the application to deploy to Elastic Beanstalk.

Published Application Path

Required if `Application Type` is set to **ASP.NET Core**. The path to the directory where the command `dotnet publish` outputs the published application.

Application Name*

The name of the Elastic Beanstalk application.

Environment Name*

The name of the Elastic Beanstalk environment that will run the application.

An environment represents the AWS resources (e.g., load balancer, Auto Scaling group, and Amazon EC2 instances) created specifically to run your application.

Version Label

Version label for the new application revision. If not specified the task will construct a version label based on the current date and time, expressed in milliseconds (for example *v20171120222623*).

AWS Lambda Deployment Task

Synopsis

Supports deployment of AWS Lambda functions for all supported Lambda language runtimes. Note that this task can be used to deploy .NET Core-based functions but it does not build the deployment package first. To perform a build and deployment for .NET Core-based functions, or to deploy .NET Core-based serverless applications, please refer to the AWS Lambda .NET Core Deployment task.

Description

Applications that are based on Lambda (also referred to as serverless applications) are composed of functions triggered by events. A typical serverless application consists of one or more functions triggered by events such as object uploads to Amazon S3, Amazon SNS notifications, and API actions. Those functions can stand alone or use other resources such as Amazon DynamoDB tables or Amazon S3 buckets. The most basic serverless application is simply a function.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS Lambda Deploy Function. You can rename it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Deployment Mode*

The operating mode for the task. The default setting, *Update code only*, exclusively uploads new code for an already published function. The alternative setting, *Update code and configuration (or create a new function)* can be used to publish new functions or to deploy new code, and other configuration settings, to a pre-existing function. When updating code and configuration the task performs the configuration changes first, then uploads and optionally publishes the new code.

Function Name*

The name of the function to update or create.

Description

A short, user-defined function description. Lambda does not use this value.

Function Handler*

Displayed when creating a new function, or updating code and configuration for an existing function. Specifies the name of the handler that will be called when your function is invoked. Different languages have different rules for the formatting of this field. For more information, see https://docs.aws.amazon.com/lambda/latest/dg/programming-model-v2.html.

Runtime*

Displayed when creating a new function, or updating code and configuration for an existing function. Specifies the language runtime appropriate to the code you are deploying.

Code Location*

Specifies the source location of the deployment package to be uploaded. You can choose from a file in the local file system or a file previously uploaded to Amazon S3. If the source location is Amazon S3 you can also optionally specify a specific version of the file.

Role ARN or Name*

Displayed when creating a new function, or updating code and configuration for an existing function. Specifies the role to be assumed when your function is invoked. The role supplies credentials (if needed) to your function as well as controlling AWS resource access. You can specify either the name of the role or the role's Amazon Resource Name (ARN). If the role name is specified the task will retrieve and use the role ARN for you.

Memory Size

Displayed when creating a new function, or updating code and configuration for an existing function. The amount of memory, in MB, your Lambda function is given. Lambda uses this memory size to infer the amount of CPU and memory allocated to your function. Your function use-case determines your CPU and memory requirements. For example, a database operation might need less memory compared to an image processing function. The default value is 128 MB. The value must be a multiple of 64 MB.

Timeout

Displayed when creating a new function, or updating code and configuration for an existing function. Specifies the execution time at which Lambda should terminate the function. Because the execution time has cost implications, we recommend you set this value based on your expected execution time. The default is 3 seconds.

Publish

This parameter can be used to request AWS Lambda to create or update the Lambda function and publish a version as an atomic operation.

Advanced

Advanced settings are only displayed when creating a new function, or updating code and configuration for an existing function.

Dead Letter ARN

The Amazon Resource Name (ARN) of an Amazon SQS queue or Amazon SNS topic to be used as your Dead Letter Queue (DLQ).

KMS Key ARN

The Amazon Resource Name (ARN) of the KMS key used to encrypt your function's environment variables. If not provided, AWS Lambda will use a default service key.

Environment Variables

Your function's environment configuration settings. Specify one pair per line, in *key*=*value* format.

Tags

List of tags (key-value pairs) assigned to the new function. Enter as *key*=*value*, one per line. Tags can only be specified when creating a new function and are ignored when updating functions.

Security Group IDs

List of security group IDs, one per line. If your Lambda function accesses resources in a VPC at least one security group and one subnet ID belonging to the same VPC must be specified.

Subnet IDs

List of subnet IDs, one per line. If your Lambda function accesses resources in a VPC at least one security group and one subnet ID belonging to the same VPC must be specified.

Tracing configuration

Your function's trace settings. Can be either X-Ray, PassThrough or Active. If PassThrough, Lambda will only trace the request from an upstream service if it contains a tracing header with "sampled=1". If Active, Lambda will respect any tracing header it receives from an upstream service. The default setting of X-Ray means that if no tracing header is received, Lambda will call X-Ray for a tracing decision.

AWS Lambda .NET Core Deployment Task

Synopsis

Builds and deploys a .NET Core AWS Lambda function or serverless application. For other languages supported by Lambda, please refer to the AWS Lambda Deploy Function task.

Description

Applications based on Lambda (also referred to as serverless applications) are composed of functions triggered by events. A typical serverless application consists of one or more functions triggered by events such as object uploads to Amazon S3, Amazon SNS notifications, and API actions. Those functions can stand alone or use other resources such as Amazon DynamoDB tables or Amazon S3 buckets. The most basic serverless application is simply a function.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS Lambda .NET Core Deployment. You can rename it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Path to Lambda Project*

The relative path to the location of the Lambda project.

Deployment Type*

Either *Function* or *Serverless application*.

Function performs a single Lambda function deloyment. *Serverless application* performs a deployment with AWS CloudFormation, allowing a multiple function deployment.

Function Deployment: Lambda Function Properties

Function Name

The name of the Lambda function to invoke. You can also specify the Amazon Resource Name (ARN) of the function.

Function Role

The name of the IAM role providing access to AWS services for the deployed Lambda function.

Function Handler

The function within your code that Lambda calls to begin execution. The format is `<assembly-name>::<type-name>::<function-name>`.

Function Memory (MB)

The memory allocated to the Lambda function. The value must be in multiples of 64.

Function Timout (Seconds)

The function execution time at which Lambda should terminate the function.

Serverless Application Deployment: Serverless Application Properties

Stack Name

AWS CloudFormation stack name. A stack is a collection of AWS resources that you can manage as a single unit.

S3 Bucket

The S3 bucket used to store the built project.

S3 Prefix

The S3 object key prefix used for the objects uploaded to Amazon S3.

Advanced

Additional Command Line Arguments for Lambda Tools

Additional arguments you can use when executing the `dotnet lambda` command.

AWS Lambda Invoke Function Task

Synopsis

This task invokes an AWS Lambda function with a JSON payload.

Description

This task invokes a previously deployed Lambda function.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS Lambda Invoke Function. You can rename it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Function Name*

The name of the Lambda function to invoke. You can also specify the Amazon Resource Name (ARN) of the function.

Payload

The JSON formatted payload to pass to the function.

Invocation Type

Either *Asynchronous execution* or *Synchronous execution returning the output from the function*.

Synchronous Execution Output

Output Variable

The name of the variable that will contain the function output on task completion. You can use the variable as `$(variableName)` to refer to the function result in subsequent tasks.

Log Type

For synchronous execution, returns the base64-encoded last 4 KB of log data produced by your Lambda function in the `x-amz-log-result` header.

AWS S3 Download Task

Synopsis

Downloads file and folder content from an Amazon Simple Storage Service (S3) bucket.

Description

Downloads file and folder content from an Amazon Simple Storage Service (S3) bucket to a folder location. The source location in the bucket, or key prefix, can also be specified. If a source location is not supplied,the bucket root is used. You specify the files to download using a set of one or more globbing patterns. The default pattern is `**`, causing all files in all folders at and beneath the source location to be downloaded, preserving the relative folder paths.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS S3 Download. You can rename it or append the name of the associated S3 bucket to it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Bucket Name*

The name of the bucket containing the content to download.

Source Folder

The source folder (key prefix) in the bucket that the content patterns are run against. If not set, the root of the bucket is assumed.

Filename Patterns

Glob patterns to select the file and folder content to download. Supports multiple lines of minimatch patterns. The default is `**`.

Target Folder*

The target folder to contain the downloaded content. You can browse for it or you can use variables.

Advanced

Overwrite

If checked, replaces existing files in and beneath the target folder. If not checked and the file already exists in the target location, an error is thrown.

Default: checked (overwrite).

Force path style addressing

If checked the task will always use path style addressing to work with the bucket. The default behavior, when unchecked, is to use virtual host style addressing if the bucket name is DNS compatible and path style otherwise. For more information see Virtual Hosting of Buckets.

Flatten folders

If checked the task will remove the key prefix from the downloaded objects causing them to be written to the selected download folder. If this option is unchecked, the key prefix of each object is preserved and objects are downloaded to a subfolder hierarchy matching the key prefix of the object.

AWS S3 Upload Task

Synopsis

Uploads file and folder content to an Amazon Simple Storage Service (S3) bucket.

Description

This task accepts a source location from which to upload files to an Amazon S3 bucket. The target location in the bucket, or key prefix, can also be specified. If you don't supply a target location, the files are uploaded to the bucket root. You specify the files to upload by using a set of one or more globbing patterns. The default pattern is `**`, which causes all files in all folders at and beneath the source location to be uploaded, preserving the relative folder paths.

The task can optionally create the bucket to which the content is to be uploaded.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS S3 Upload. You can rename it or append the name of the associated S3 bucket to it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Bucket Name*

The name of the bucket where the content will be uploaded. Bucket names must be globally unique. If the bucket does not exist, it will be created.

Source Folder

The source folder that the content patterns are run against. You can browse for the folder. If not set, the root of the repository is assumed. Use variables if the files are not in the repo.

Example: code:*$(agent.builddirectory)*

Filename Patterns

One or more globbing patterns, one per line, that are used to select the files in the source folder to be uploaded. Supports multiple lines of minimatch patterns.

Default: `**` to select all files and subfolders of the source location.

Target Folder*

The target folder (also known as key prefix) in the S3 bucket that all uploaded files will share, or the folder path in the bucket. You can use variables.

If not set, the root of the bucket is assumed.

Access Control

The canned access control list (ACL) to apply to the uploaded content. See Canned ACL for an explanation of the possible values. The default is *Private*.

Create S3 Bucket If It Does Not Exist

If checked and the specified bucket does not exist, the task attempts to automatically create it.

Default: checked (auto-create).

Advanced

Overwrite

If checked, existing files in the bucket at the target location are overwritten.

Default: checked (overwrite).

Flatten Folders

If checked, the relative subfolders of the files being uploaded are removed and all files are placed directly into the target location.

Default: unchecked (preserve folder hierachy).

Force path style addressing

If checked the task will always use path style addressing to work with the bucket. The default behavior, when unchecked, is to use virtual host style addressing if the bucket name is DNS compatible and path style otherwise. For more information see Virtual Hosting of Buckets.

AWS Systems Manager Get Parameter

Synopsis

Reads one or more values from Systems Manager's Parameter Store into build variables.

Description

This task reads a parameter value, or hierarchy of values identified by common path, into build variables in the build or release definition. These variables are then accessible from downstream tasks in the definition. The names used for the build variables are customizable.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS Systems Manager Get Parameter. You can rename it or append the name of the associated Parameter Store parameter or parameter path to it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Read Mode*

Sets the mode of operation. Choose from reading a single parameter value identified by name, or a hierarchy of parameter values identified by common path.

Parameter Name

Required if Read Mode is set to get a single parameter value. Identifies the name of the parameter to read.

Parameter Path

Required if Read Mode is set to parameter hierarchy. Identifies the path of the parameter(s) to be read.

Recursive

Available when reading a parameter hierarchy. If checked, values for the specified Parameter Path and all sub-paths are read. If unchecked only the values for parameters matching the supplied path are read, values in sub-paths are ignored.

Variable Name Transform

Specifies how the build variable name(s) to hold the parameter value(s) are created. You can choose from

- Use parameter names (including any paths) as variable names. The full parameter name is used to set the build variable name.
- Use leaf of parameter names as variable names. The path is removed and the resulting leaf text used as the build variable name.

- Replace text in the parameter name using a regular expression. Replace text in the parameter name to form the build variable name.
- Use custom name. Available for single parameter read mode only, enables entry of a custom name for the build variable.

Custom Variable Name

Required if Variable Name Transform is set to 'use custom name'. Specifies the desired name for the build variable.

Search Pattern

Required if Variable Name Transform is set to 'replace text'. Specifies the search pattern as a regular expression.

Replacement Text

Specifies the replacement text pattern when Variable Name Transform is set to 'replace text'.

Global Match

When replacing text, specifies if the replacement stops at the first match or replaces all occurrences of the search pattern.

Case-insensitive Match

When replacing text specifies if the search should be done in a case-insensitive manner.

AWS Systems Manager Run Command

Synopsis

Runs a Systems Manager or user-provided Command on a fleet of EC2 instances. Commands can also target on-premise machines if the required Systems Manager agent is installed.

Description

This task runs a Systems Manager Command, or a user-provided Command, on a fleet of EC2 instances. On-premise machines can also be targets if the required Systems Manager agent is installed. The command to run is identified by name. The targets on which the command will be run are identified using either instance IDs or tags. Parameters specific to the selected Command can also be specified.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, AWS Systems Manager Get Parameter. You can rename it or append the name of the associated Parameter Store parameter or parameter path to it.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Document Name*

The name of the Command document to run. This can be a Systems Manager-provided document or a custom document private to your account and to which you have access.

Parameters

The required and optional parameters for the document to be executed, specified as JSON. Refer to the specific command to be run for details.

Example format: `{ "param1" : [ "value" ], "param2" : [ "value","value2" ] }`

Comment

User-specified information about the command, such as a brief description of what the command should do. Maximum length 100 characters.

Service Role ARN

The Amazon Resource Name (ARN) or name of the IAM role Systems Manager uses to send notifications. If the name of a role is supplied the task will automatically determine the ARN.

Select Targets by*

How the instances to be targetted by the command are selected. You can choose from Instance IDs, Tags or the name of a build variable containing a list of instance IDs.

Instance IDs

Required if target selection is set to Instance IDs. You can specify up to 50 instance IDs, one per line.

Tags

Required if target selection is set to Tags. Specify tags one per line, in the format 'Key=Value'.

Variable Name

Required if target selection is set to Build Variable Name. Specify the name of the variable. Do not enclose the variable name in $() syntax. The variable should contain a comma-delimited list of instance IDs.

Execution Concurrency

The maximum number of instances that are allowed to execute the command at the same time. You can specify a number such as 10 or a percentage such as 10%. The default value is 50.

Max Errors Before Stop

The maximum number of errors allowed without the command failing. When the command fails one more time beyond the value specified, the systems stops sending the command to additional targets. You can specify a number like 10 or a percentage like 10%. The default value is 50.

Timeout (seconds)

If this time is reached and the command has not already started executing, it will not execute. Minimum value of 30, maximum value of 2592000. Default value: 600.

Notification ARN

An Amazon Resource Name (ARN) for a Simple Notification Service (SNS) topic. Run Command pushes notifications about command status changes to this topic.

Notification Events

The different events for which you can receive notifications.

Notification Type

Select **Command** to receive notification when the status of a command changes. For commands sent to multiple instances, select **Invocation**, to receive notification on a per-instance basis when the status of a command changes.

S3 Bucket Name

The name of the S3 bucket where command execution responses should be stored.

S3 Key Prefix

The directory structure within the S3 bucket where the responses should be stored.

Command ID Output Variable

The name of a variable that will contain the unique ID assigned to the command. The command ID can be used future references to the request.

AWS Send Message Task

Synopsis

Sends a message to an Amazon Simple Notification Service (SNS) topic or to an Amazon Simple Queue Service (SQS) queue.

Description

This task accepts a message to be sent to an Amazon SNS topic or to an Amazon SQS queue. If the message is to be sent to a queue, you can configure an optional delay (in seconds). If you don't specify a delay, the task assumes the default delay that is associated with the queue.

Parameters

You can set the following parameters for the task. Required parameters are noted by an asterisk (*). Other parameters are optional.

Displayname*

The default name of the task, Send Message. The delivery target, topic or queue, is appended to the name.

AWS Credentials*

The AWS credentials to use. If needed, choose **+**, and then add a new AWS connection.

AWS Region*

The AWS Region name to use. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Message Target*

The target for the message, a topic in Amazon SNS or an Amazon SQS queue.

Message

The message to send. For the allowed values, see the respective service help pages for Publish and SendMessage.

Topic ARN*

Required parameter only if `Message Target` is set to **SNS Topic**. Supply the Amazon Resource Name (ARN) of the topic.

Queue Url*

Required parameter only if `Message Target` is set to **SQS Queue**. Supply the URL of the queue.

Delay (seconds)

Available for Amazon SQS queues only. The length of time, in seconds, to delay a specific message. Valid values: 0 to 900. Maximum: 15 minutes. Messages with a positive `DelaySeconds` value become available for processing after the delay period is finished. If you don't specify a value, the default value for the queue applies.

Document History

This topic describes important changes to the AWS Tools for Microsoft Visual Studio Team Services over the course of its history.

This documentation was built on: Feb 14, 2018

Feb 14, 2018

New SDK version, 1.0 released.