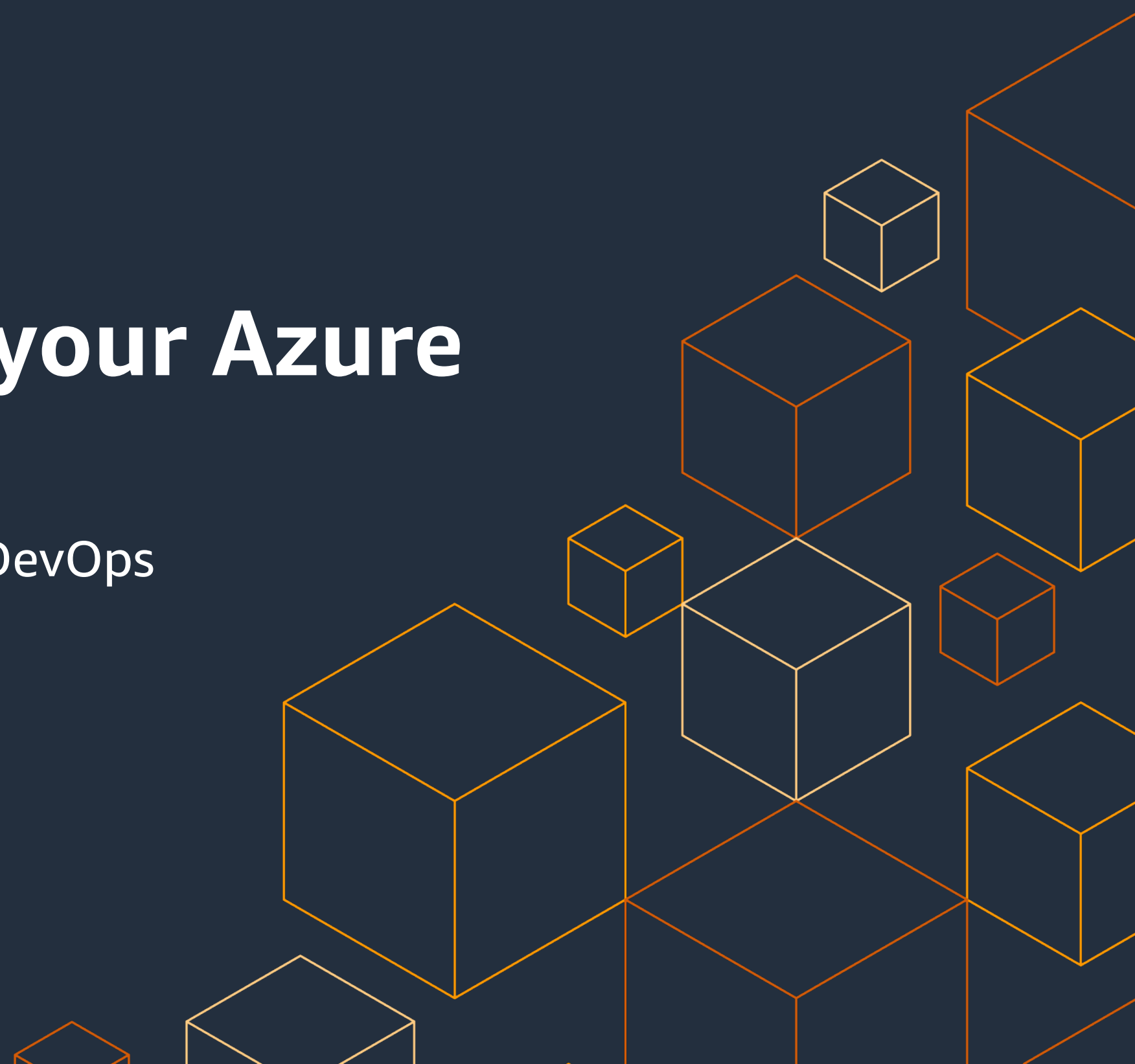




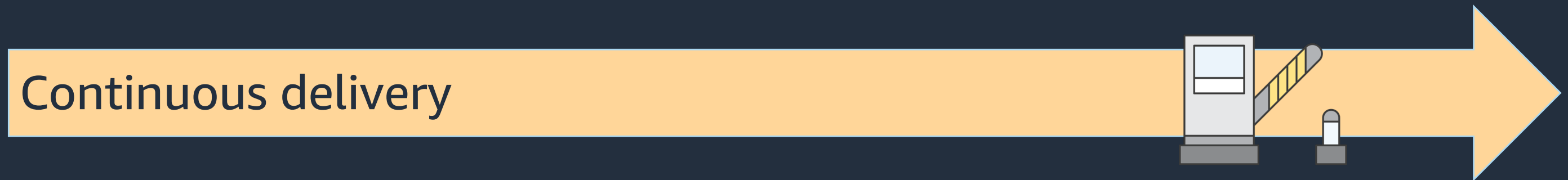
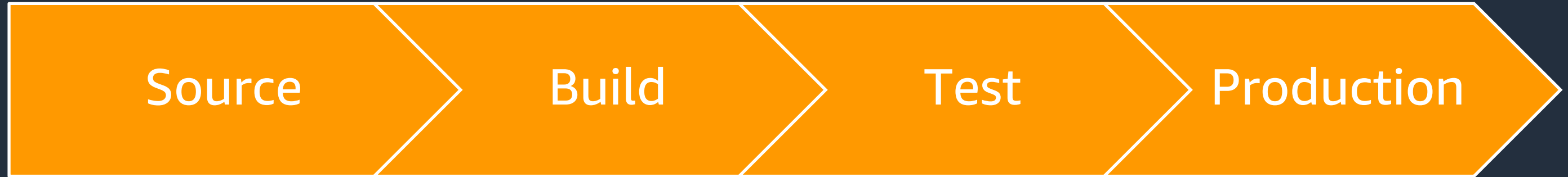
# Reach AWS from your Azure DevOps pipeline

Using the AWS Toolkit for Azure DevOps

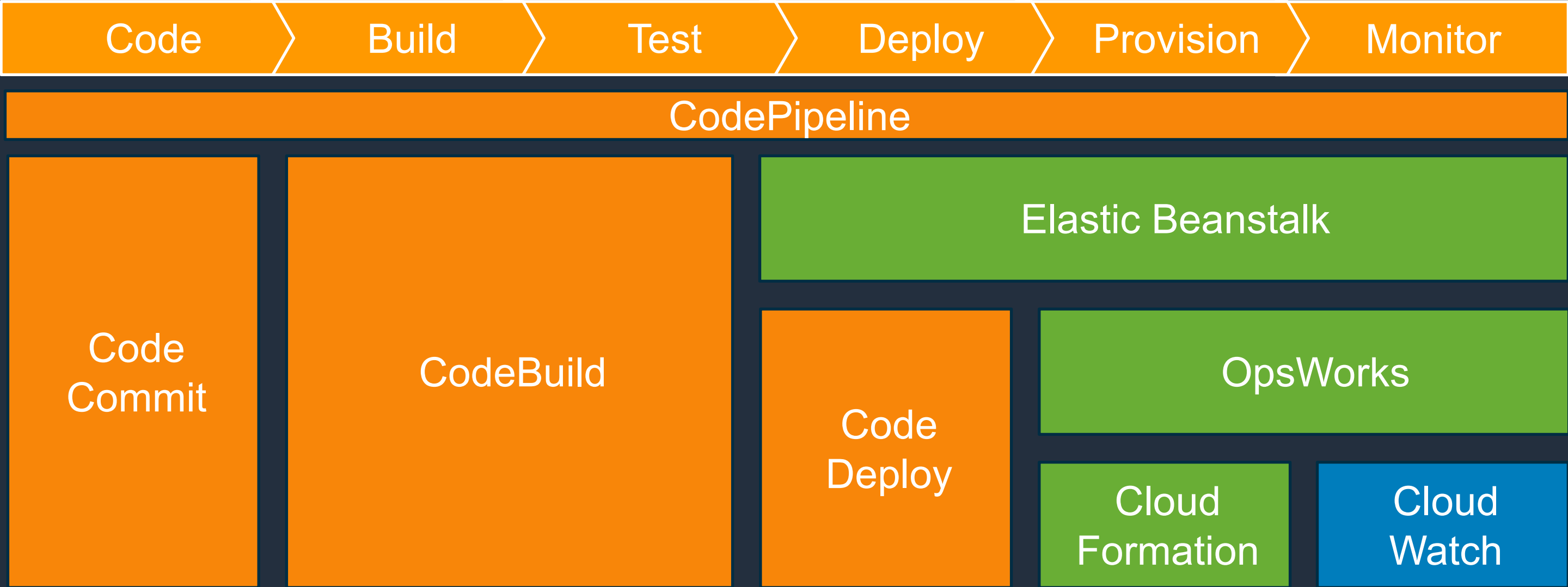
Roman Martynenko  
Sr. Solutions Architect, AWS



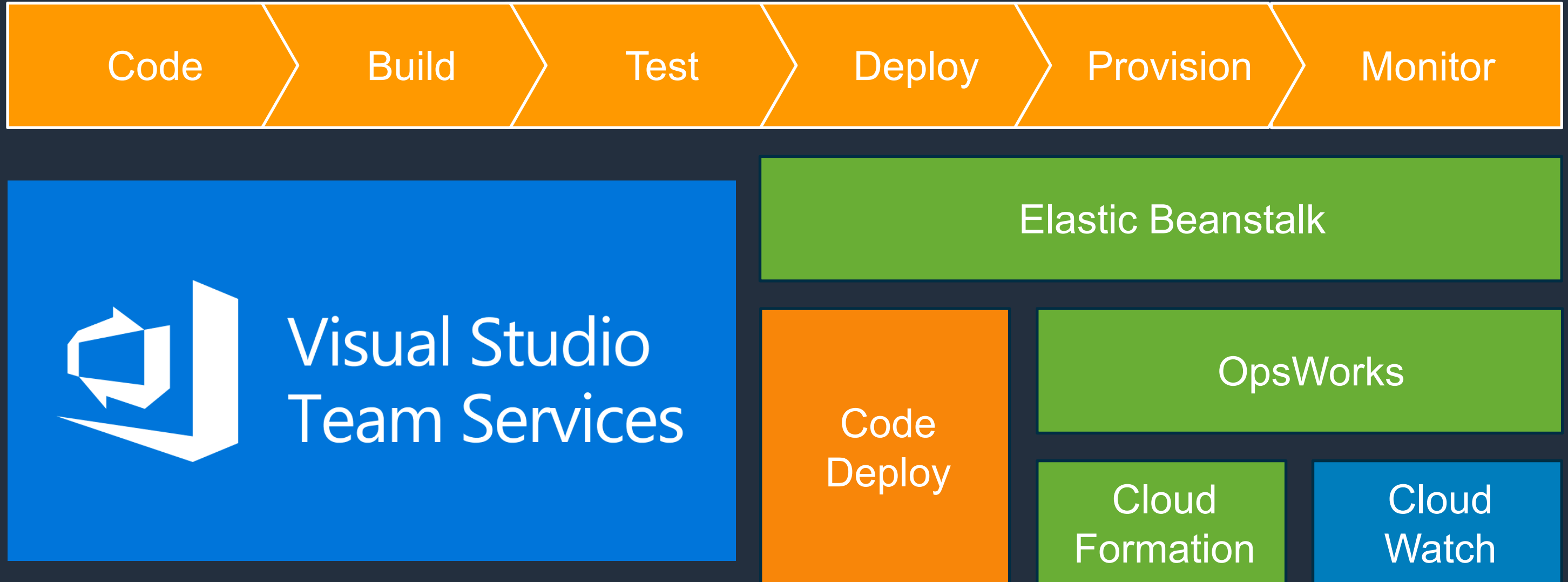
# Release processes levels



# AWS DevOps Services



# AWS DevOps Services



# What is the Toolkit for Azure DevOps?



Extension for hosted and on-premises Azure DevOps (aka: TFS, VSTS)

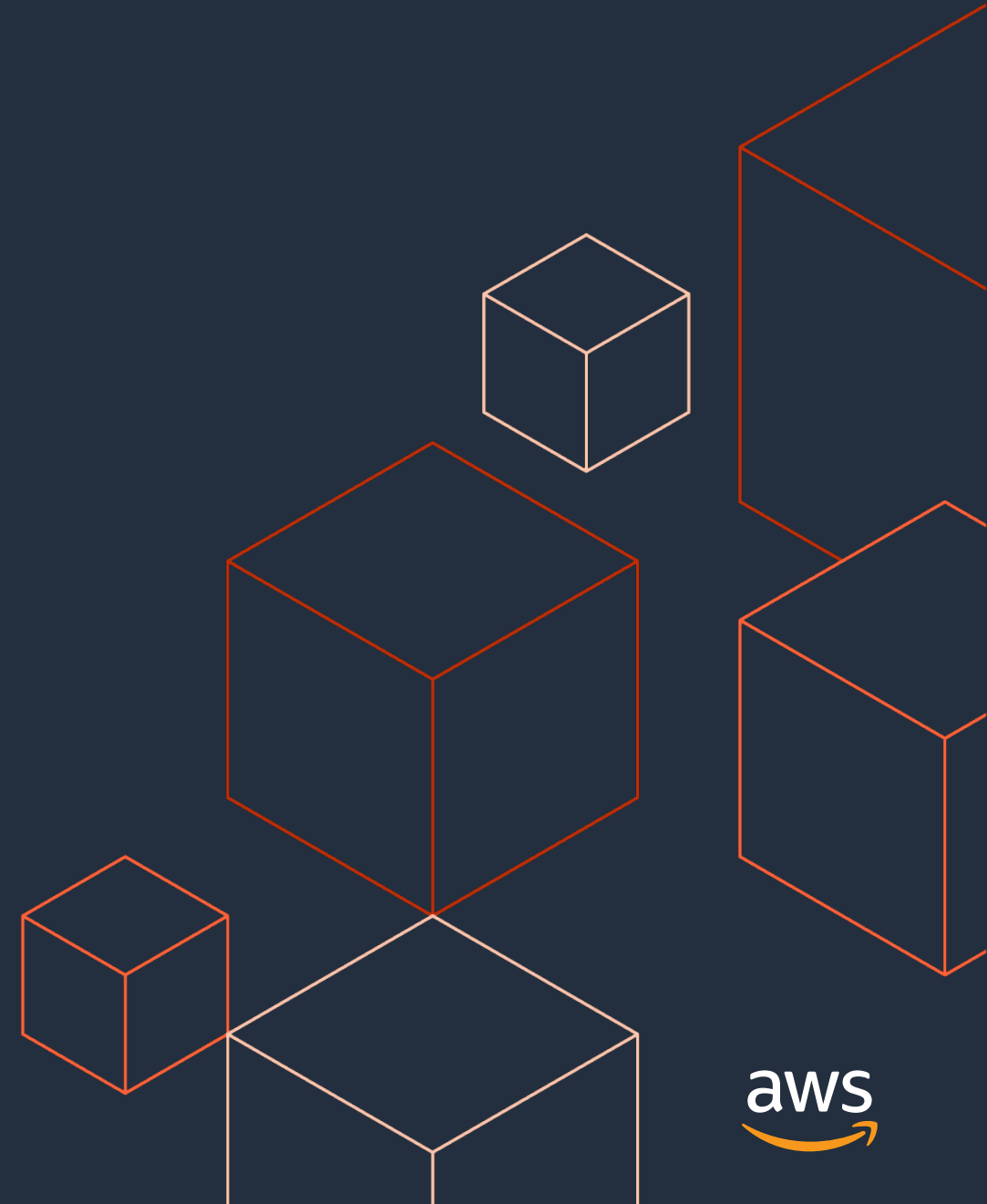
Continue to use your existing tools!

Adds AWS-specific tasks that can be added to your build and release pipelines

Interacts with AWS Services

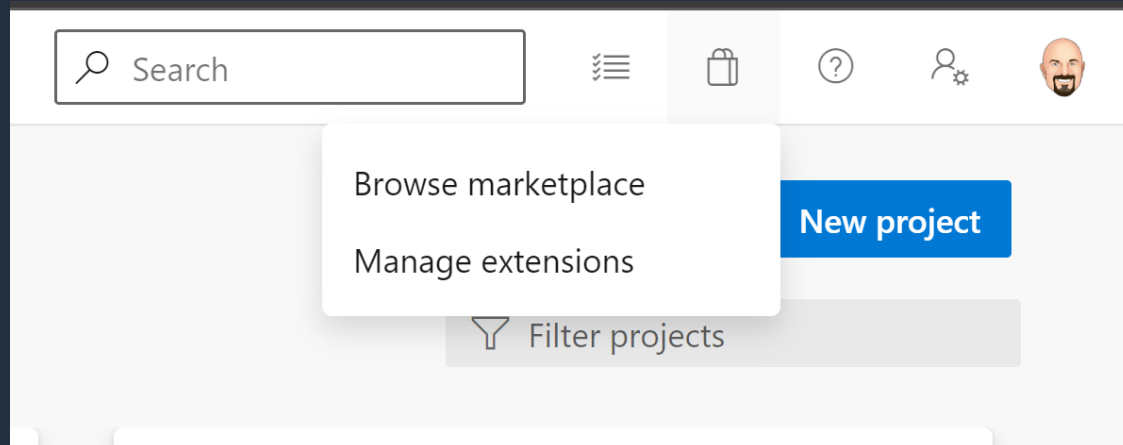
- ECR/ECS
- Elastic Beanstalk
- CodeDeploy
- Lambda
- (and many others!)

# Installing the Toolkit




# Available in the Extensions Marketplace

Search the [Azure DevOps Marketplace](#) for “AWS Toolkit”



Visual Studio | Marketplace

Azure DevOps > Azure Pipelines > AWS Toolkit for Azure DevOps



## AWS Toolkit for Azure DevOps


Amazon Web Services | 📦 28,028 installs | ★★★★★ (18) | Free

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.

Get it free

[Overview](#) [Q & A](#) [Rating & Review](#)

### Overview

Coverage:  codecov unknown

The AWS Toolkit for Azure DevOps adds tasks to easily enable build and release pipelines in Azure DevOps (formerly VSTS) and Azure DevOps Server (previously known as Team Foundation Server (TFS)) to work with AWS services including Amazon S3, AWS Elastic Beanstalk, AWS CodeDeploy, AWS Lambda, AWS CloudFormation, Amazon Simple Queue Service and Amazon Simple Notification Service, and run commands using the AWS Tools for Windows PowerShell module and the AWS CLI.

The AWS Toolkit for Azure DevOps is available from the [Visual Studio Marketplace](#).

This is an open source project because we want you to be involved. We love issues, feature requests, code reviews, pull requests or any positive contribution. Please see the [CONTRIBUTING](#) guide for how to help, including how to build your own extension.

# Providing AWS Credentials



# Adding an AWS Service Connection

The screenshot displays the AWS CodePipeline console interface. On the left, the 'Project Settings' sidebar for 'Test Project' is visible, with the 'Service connections' option highlighted under the 'Boards' section. The main content area shows a 'Create your first service connection' prompt with a 'Create service connection' button. A modal dialog titled 'New service connection' is open on the right, listing various service types. The 'AWS' option is selected and highlighted with a red box.

**Project Settings**  
Test Project

**General**

- Overview
- Teams
- Permissions
- Notifications
- Service hooks
- Dashboards

**Boards**

- Project configuration
- Team configuration
- GitHub connections
- Pipelines
- Agent pools
- Parallel jobs
- Settings
- Test management
- Release retention
- Service connections**
- XAML build services

**Create your first service connection**  
Service connections help you manage your pipeline to external services.

**Create service connection**

**New service connection**

Choose a service or connection type

Search connection types

- ☒ **aws AWS**
- ☐ Azure Classic
- ☐ Azure Repos/Team Foundation Server
- ☐ Azure Resource Manager
- ☐ Azure Service Bus
- ☐ Bitbucket Cloud
- ☐ Chef
- ☐ Docker Host
- ☐ Docker Registry
- ☐ Generic
- ☐ GitHub
- ☐ GitHub Enterprise Server
- ☐ Incoming WebHook
- ☐ Jenkins

# Setting AWS Credentials in Service Endpoint

One-time setup for all users of the pipeline

Obtain credentials from IAM

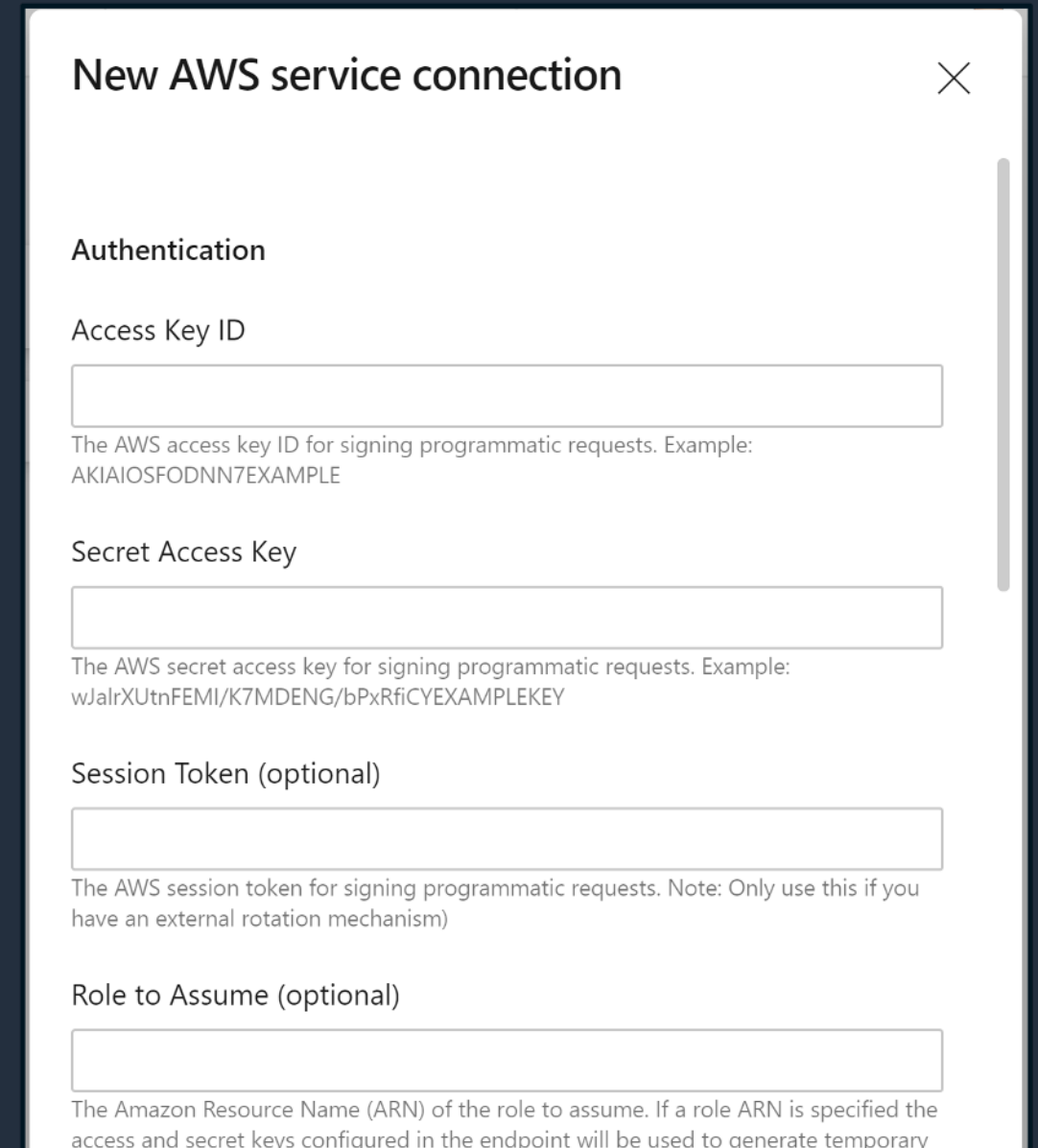
- Access Key ID
- Secret Access Key

Can also use STS assume-role for temporary credentials generated at build time

- Role to assume
- External ID (optional)

Can use STS-assigned credentials

- Session Token
- **NOTE:** STS credentials expire and will need to be refreshed with an external process



The screenshot shows a 'New AWS service connection' dialog box with a close button (X) in the top right corner. The dialog is divided into sections for 'Authentication', 'Session Token (optional)', and 'Role to Assume (optional)'. Each section has a text input field and a descriptive note below it.

**New AWS service connection** [X]

**Authentication**

Access Key ID

[Text Input Field]

The AWS access key ID for signing programmatic requests. Example: AKIAIOSFODNN7EXAMPLE

Secret Access Key

[Text Input Field]

The AWS secret access key for signing programmatic requests. Example: wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

Session Token (optional)

[Text Input Field]

The AWS session token for signing programmatic requests. Note: Only use this if you have an external rotation mechanism)

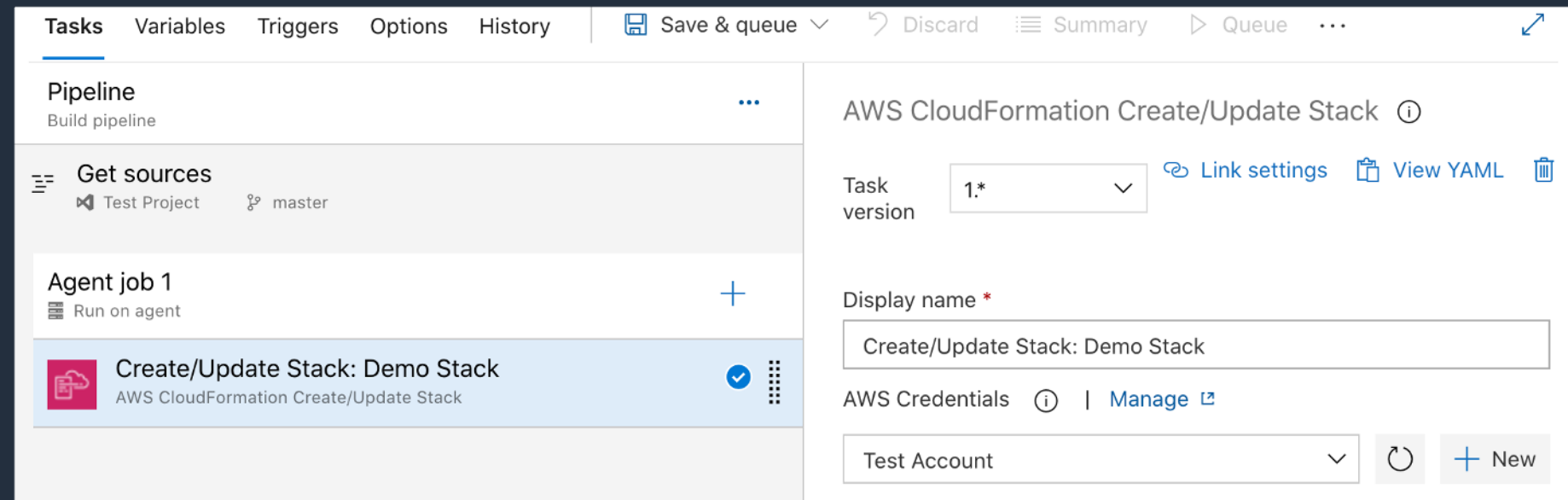
Role to Assume (optional)

[Text Input Field]

The Amazon Resource Name (ARN) of the role to assume. If a role ARN is specified the access and secret keys configured in the endpoint will be used to generate temporary

# Supplying AWS Credentials to Tasks

- Configure a service endpoint of type AWS for use in tasks
- Create specific named variables in your build
- Use standard AWS environment variables in the build agent process
- Create specific named variables in your build
- Use EC2 Instance Profiles



# Setting AWS Credentials as Pipeline Variables

Set well-known variables in your pipeline:

- `AWS.AccessKeyID` – IAM Access Key
- `AWS.SecretAccessKey` – IAM Secret Key

If using STS-issued credentials, also supply the token using

- `AWS.SessionToken`

Optionally, can also specify the region using

- `AWS.Region`

# Setting AWS Credentials as Environment Variables

Set well-known environment variables on your build server/user profile:

- `AWS_ACCESS_KEY_ID` – IAM Access Key
- `AWS_SECRET_ACCESS_KEY` – IAM Secret Key

If using STS-issued credentials, also supply the token using

- `AWS_SESSION_TOKEN`

Optionally, can also specify the region using

- `AWS_REGION`

# Setting AWS Credentials as EC2 Instance Profiles

When running a build agent on EC2:

- Associate EC2 instance profile with permissions to call AWS services
- Tasks will automatically obtain credentials from EC2 metadata
- Credentials will be automatically refreshed by EC2

**NOTE:** These credentials will be for all users' pipelines executed on the server

# Pipeline tasks



# Amazon S3



Upload/download file and folder content to/from an S3 Bucket file or folder

Supports Globbing (`/**/*`) to select files

Useful to do things like:

- Upload published assets for Web front-ends
- Download latest branding images
- etc.

Task names:

- S3Upload
- S3Download



# AWS Elastic Beanstalk



Creates an application version or deploys an application to Amazon EC2 instance(s) using AWS Elastic Beanstalk

Can package/deploy ASP.NET/ASP.NET Core directly from the workspace, or can copy to/deploy from S3 URL.

Task names:

- BeanstalkCreateApplicationVersion
- BeanstalkDeployApplication

# AWS Lambda



Deployment of AWS Lambda functions for all supported language runtimes.

Builds, packages and deploys a .NET Core AWS Lambda function or serverless application.

Can also create a deployment package for later deployment in another build or release pipeline.

Can also invoke an existing AWS Lambda function, supplying a payload as a JSON string.

Task names:

- LambdaDeployFunction
- LambdaInvokeFunction
- LambdaNETCoreDeploy

# Amazon ECR

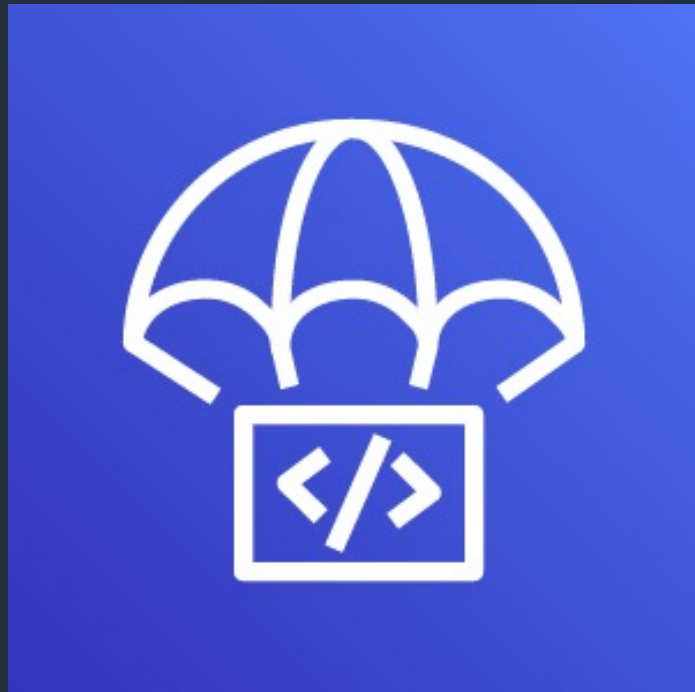


Pull/push Docker images from/to ECR  
Supports tagging (for both pulling and pushing)

Task names:

- ECRPullImage
- ECRPushImage

# AWS CodeDeploy Application Deployment



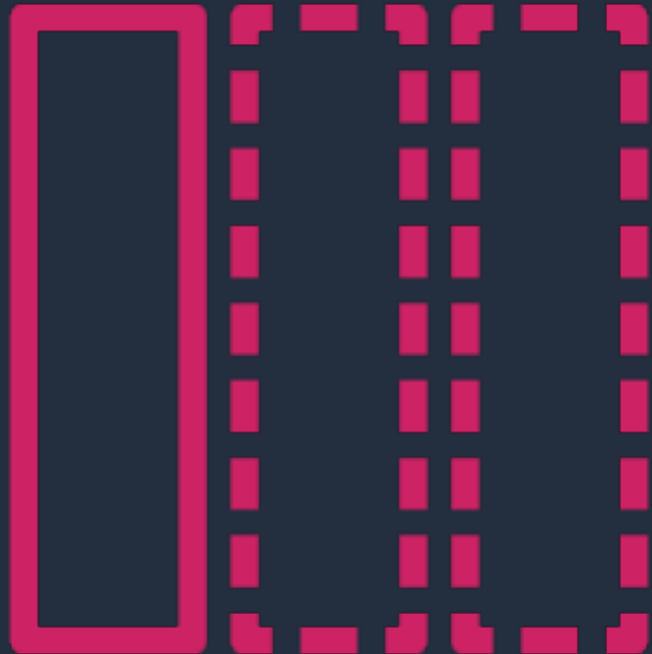
Deploys an application to Amazon EC2 instances using AWS CodeDeploy

Can supply either a folder, a pre-generated zip file, or an S3 URL for the application revision

Task name:

- CodeDeployDeployApplication

# Amazon SNS & SQS



Sends a message to an Amazon Simple

Notification Service (SNS) topic or Amazon Simple Queue Service (SQS) queue

Can also specify delay prior to sending

Task name:

- SendMessage

# AWS CloudFormation



Deploys/updates/deletes stacks

Creates/executes changesets

Task names:

- CloudFormationCreateOrUpdateStack
- CloudFormationExecuteChangeSet
- CloudFormationDeleteStack

# AWS Systems Manager



Creates, updates, or reads a parameter in Systems Manager (SSM) Parameter Store

Runs a SSM command remotely on a fleet of Amazon EC2 instances and/or on-premise machines.

Task names:

- `SystemsManagerSetParameter`
- `SystemsManagerGetParameter`
- `SystemsManagerRunCommand`

# AWS Secrets Manager



Updates a secret, optionally creating a secret if it does not exist, or reads a secret  
Supports pipeline variable for output

Can supply secret value inline or from a file (must be a file for binary secrets)

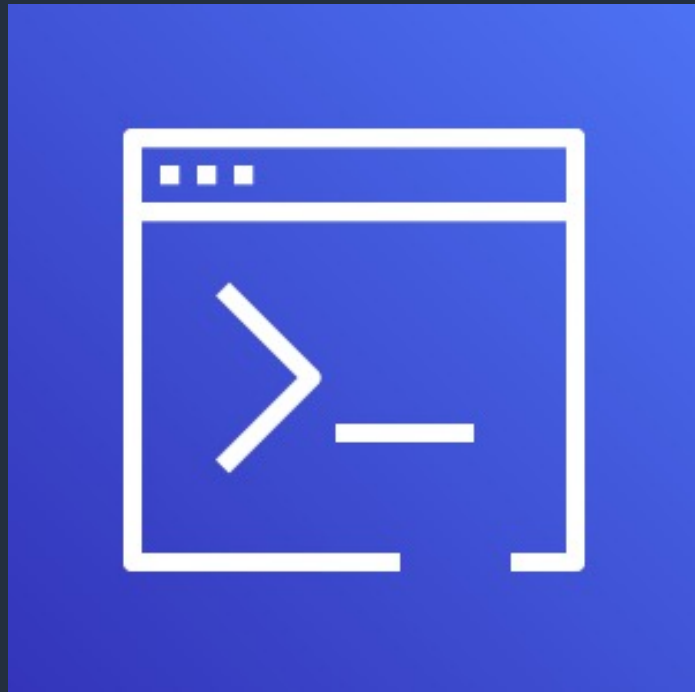
Can optionally specify KMS key (will use default CMK if none is specified)

Task names:

- SecretsManagerCreateOrUpdateSecret
- SecretsManagerGetSecret



# AWS CLI/PowerShell



Run a single CLI Command, an AWS CLI script, or an AWS Tools for Windows PowerShell script

Supports the ability to capture output in pipeline variables using *task.setvariable*

CLI environment is either CMD or Bash depending upon selected agent platform

Task names:

- AWSCLI
- AWSShellScript
- AWSPowerShellModuleScript

# Using AWS Tasks in Your Pipeline



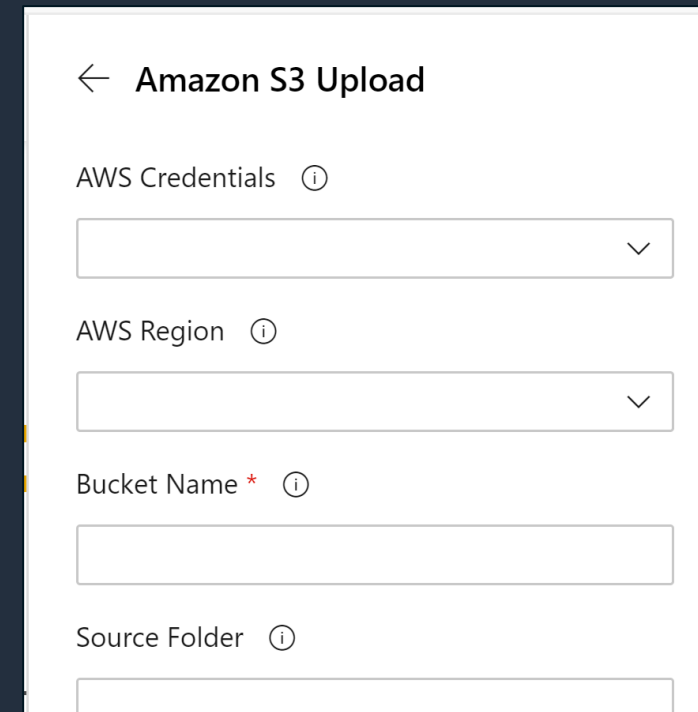
# YAML Pipeline Syntax

## Adding task via tasks sidebar

- Provides a property editor
- Generates YAML after clicking "Add"

## Also supports Intellisense in Pipeline Editor

- CTRL/Option-Space for type-ahead
- Supports tasks and parameters



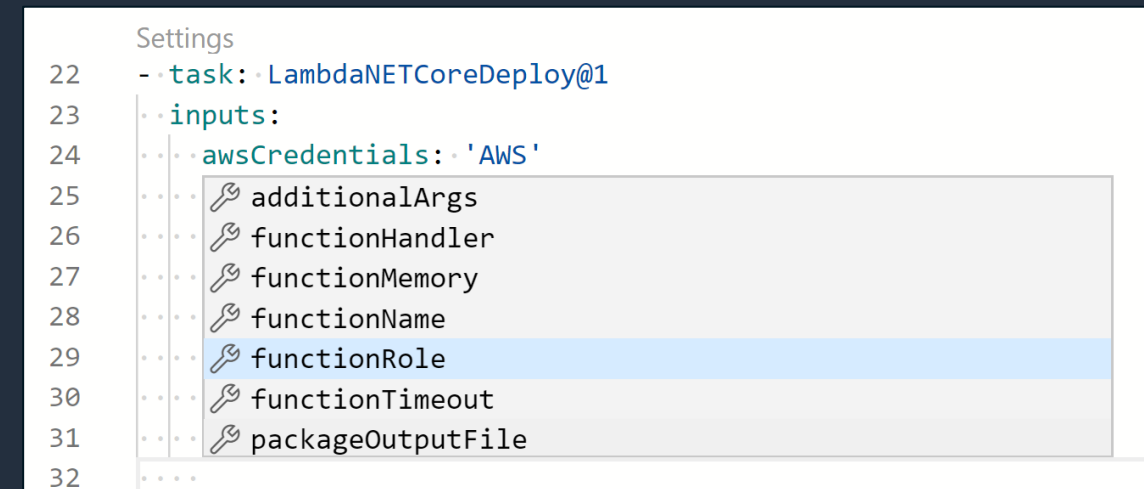
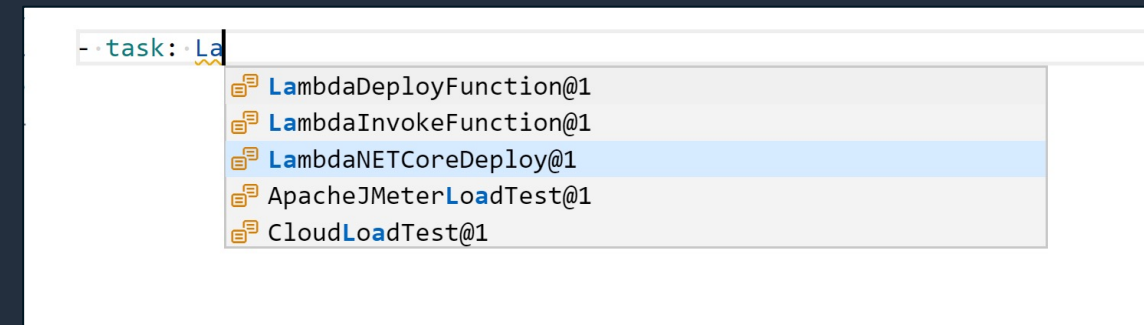
← Amazon S3 Upload

AWS Credentials ⓘ

AWS Region ⓘ

Bucket Name \* ⓘ

Source Folder ⓘ



# Example Pipeline: Serverless Application Deployment

trigger:

- master

pool:

vmImage: 'ubuntu-latest'

steps:

- task: LambdaNETCoreDeploy@1

inputs:

awsCredentials: 'MyAWSCredentials'

regionName: 'us-east-1'

command: 'deployServerless'

packageOnly: false

lambdaProjectPath: './'

stackName: 'AzureServerlessDemo'

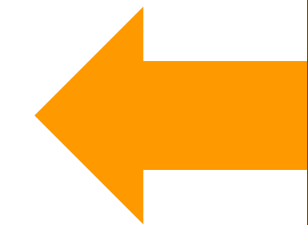
s3Bucket: 'my-unique-bucket-name'

s3Prefix: 'azuredevops'

Service Connection Name



If no value is specified for lambdaProjectPath, builds & deploys the project at the root of the source tree.



# Setting Task Parameters: GUI vs. YAML

AWS Credentials ⓘ  
MyAWSCredentials ▾

AWS Region ⓘ  
US East (N. Virginia) [us-east-1] ▾

Deployment Type \* ⓘ  
☐ Function  
☒ Serverless Application  
☐ Create deployment package only \* ⓘ

Path to Lambda Project ⓘ

Serverless Application Properties ^

Stack Name ⓘ  
AzureServerlessDemo

S3 Bucket ⓘ  
my-unique-bucket-name

S3 Prefix ⓘ  
azuredevops

```
- task: LambdaNETCoreDeploy@1
  inputs:
    awsCredentials: 'MyAWSCredentials'
    regionName: 'us-east-1'
    command: 'deployServerless'
    packageOnly: false
    lambdaProjectPath: './'
    stackName: 'AzureServerlessDemo'
    s3Bucket: 'my-unique-bucket-name'
    s3Prefix: 'azuredevops'
```

# Debugging your Pipeline

Can obtain diagnostic logging information using the *System.Debug* variable

- Enabled by setting to `true`
- Logs written to pipeline output with `##[debug]` prefix

```
11  
12     variables:  
13     - name: System.Debug  
14       value: true  
15
```

## ✓ LambdaNETCoreDeploy

```
1  ##[debug]Evaluating condition for step: 'LambdaNETCoreDeploy'  
2  ##[debug]Evaluating: SucceededNode()  
3  ##[debug]Evaluating SucceededNode:  
4  ##[debug]=> True  
5  ##[debug]Result: True  
6  Starting: LambdaNETCoreDeploy  
7  =====  
8  Task           : AWS Lambda .NET Core  
9  Description    : Builds, packages and deploys a .NET Core AWS Lambda function or serv  
10 Version       : 1.7.0  
11 Author        : Amazon Web Services  
12 Help          : Please refer to [AWS Lambda Developer Guide](https://docs.aws.amazon  
13  
14 More information on this task can be found in the [task reference](https://docs.aws  
15  
16 #####Task Permissions  
17 This task requires permissions to call the following AWS service APIs (depending on  
18 * lambda:CreateFunction  
19 * lambda:UpdateFunctionCode
```



# DEMO



# Resources

- AWS Toolkit for Azure DevOps Home Page:  
<https://aws.amazon.com/vsts/>
- User Guide (includes task reference):  
<https://docs.aws.amazon.com/vsts/latest/userguide>
- GitHub Repo:  
<https://github.com/aws/aws-toolkit-azure-devops>





# Thank you!

