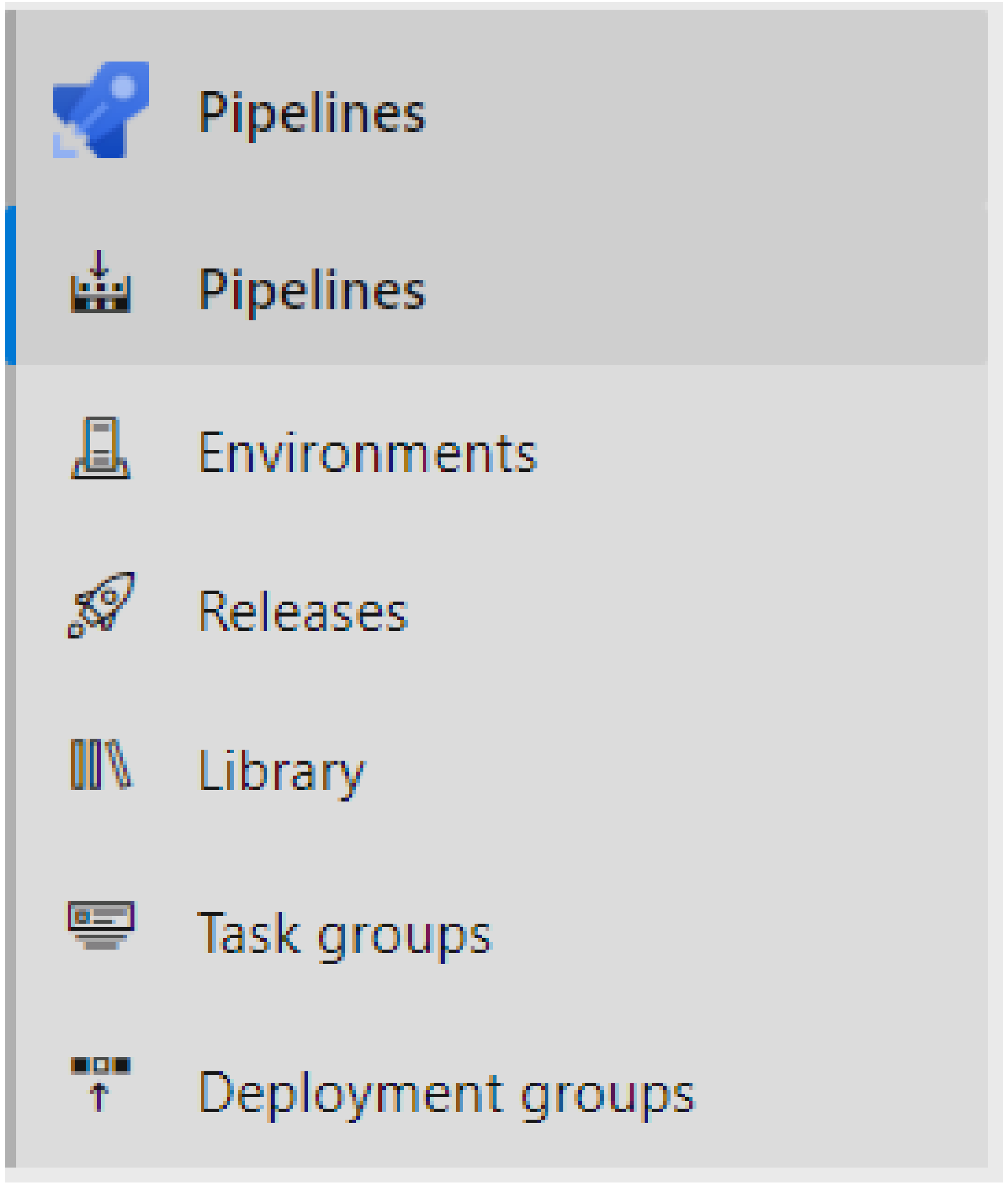
Azure DevOps: Pipeline


Pipelines
Pipelines
Environments
Releases
Library
Task groups
Deployment groups

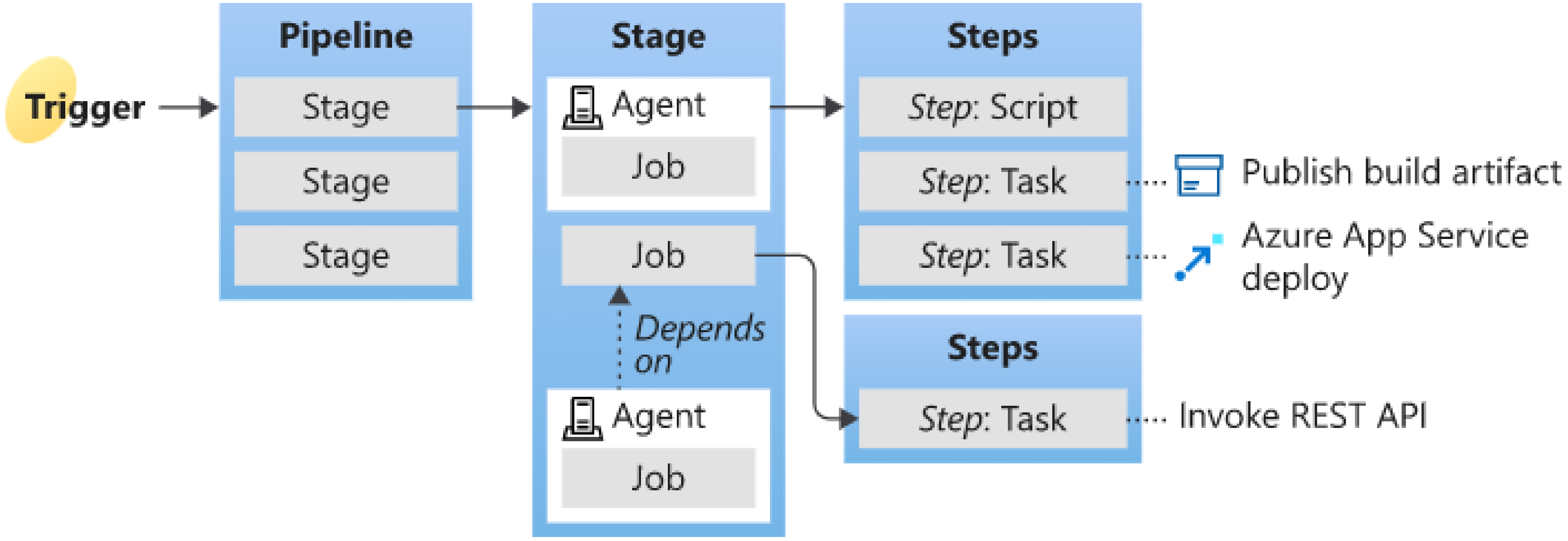
Azure DevOps: Managing Pipeline

Key concepts: Azure Pipelines

Types of Pipeline

Build Pipeline
Release Pipeline

Key concepts overview



- A trigger tells a Pipeline to run.
- A pipeline is made up of one or more stages. A pipeline can deploy to one or more environments.
- A stage is a way of organizing jobs in a pipeline and each stage can have one or more jobs.
- Each job runs on one agent. A job can also be agentless.
- Each agent runs a job that contains one or more steps.
- A step can be a task or script and is the smallest building block of a pipeline.
- A task is a pre-packaged script that performs an action, such as invoking a REST API or publishing a build artifact.
- An artifact is a collection of files or packages published by a run.

Agent

When your build or deployment runs, the system begins one or more jobs. An agent is computing infrastructure with installed agent software that runs one job at a time. For example, your job could run on a Microsoft-hosted Ubuntu agent.

Approvals

Approvals define a set of validations required before a deployment runs. Manual approval is a common check performed to control deployments to production environments. When checks are configured on an environment, pipelines will stop before starting a stage that deploys to the environment until all the checks are completed successfully.

Artifact

An artifact is a collection of files or packages published by a run. Artifacts are made available to subsequent tasks, such as distribution or deployment. For more information, see Artifacts in Azure Pipelines.

Continuous delivery

Continuous delivery (CD) is a process by which code is built, tested, and deployed to one or more test and production stages. Deploying and testing in multiple stages helps drive quality. Continuous integration systems produce deployable artifacts, which include infrastructure and apps. Automated release pipelines consume these artifacts to release new versions and fixes to existing systems. Monitoring and alerting systems run constantly to drive visibility into the entire CD process. This process ensures that errors are caught often and early.

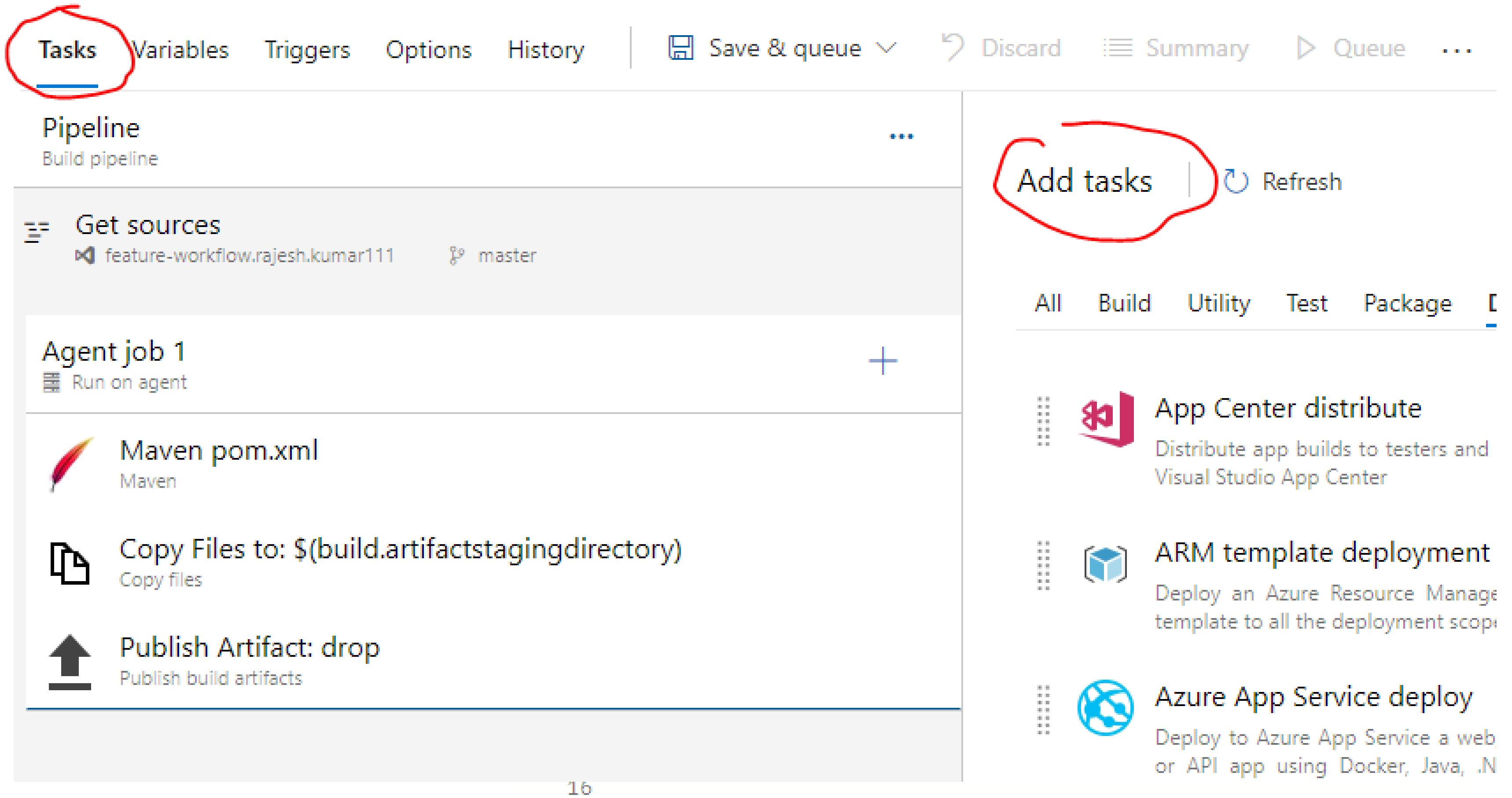
Continuous integration

Continuous integration (CI) is the practice used by development teams to simplify the testing and building of code. CI helps to catch bugs or problems early in the development cycle, which makes them easier and faster to fix. Automated tests and builds are run as part of the CI process. The process can run on a set schedule, whenever code is pushed, or both. Items known as artifacts are produced from CI systems. They're used by the continuous delivery release pipelines to drive automatic deployments.

Deployment

Classic pipelines - For Classic pipelines, a deployment is the action of running the tasks for one stage, which can include running automated tests, deploying build artifacts, and any other actions are specified for that stage.

YAML pipelines - For YAML pipelines, a deployment typically refers to a deployment job. **A deployment job is a collection of steps** that are run sequentially against an environment. You can use strategies like run once, rolling, and canary for deployment jobs.


Tasks Variables Triggers Options History | Save & queue Discard Summary Queue ...

Pipeline
Build pipeline

Get sources
feature-workflow.rajesh.kumar111 master

Agent job 1
Run on agent

Maven pom.xml
Maven

Copy Files to: $(build.artifactstagingdirectory)
Copy files

Publish Artifact: drop
Publish build artifacts

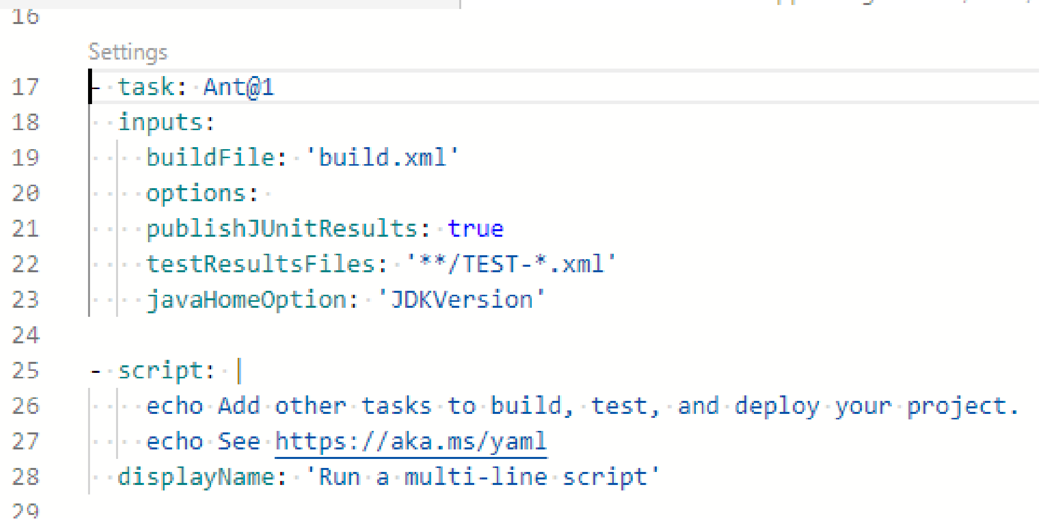
Add tasks | Refresh

All Build Utility Test Package

App Center distribute
Distribute app builds to testers and Visual Studio App Center

ARM template deployment
Deploy an Azure Resource Manage template to all the deployment scop

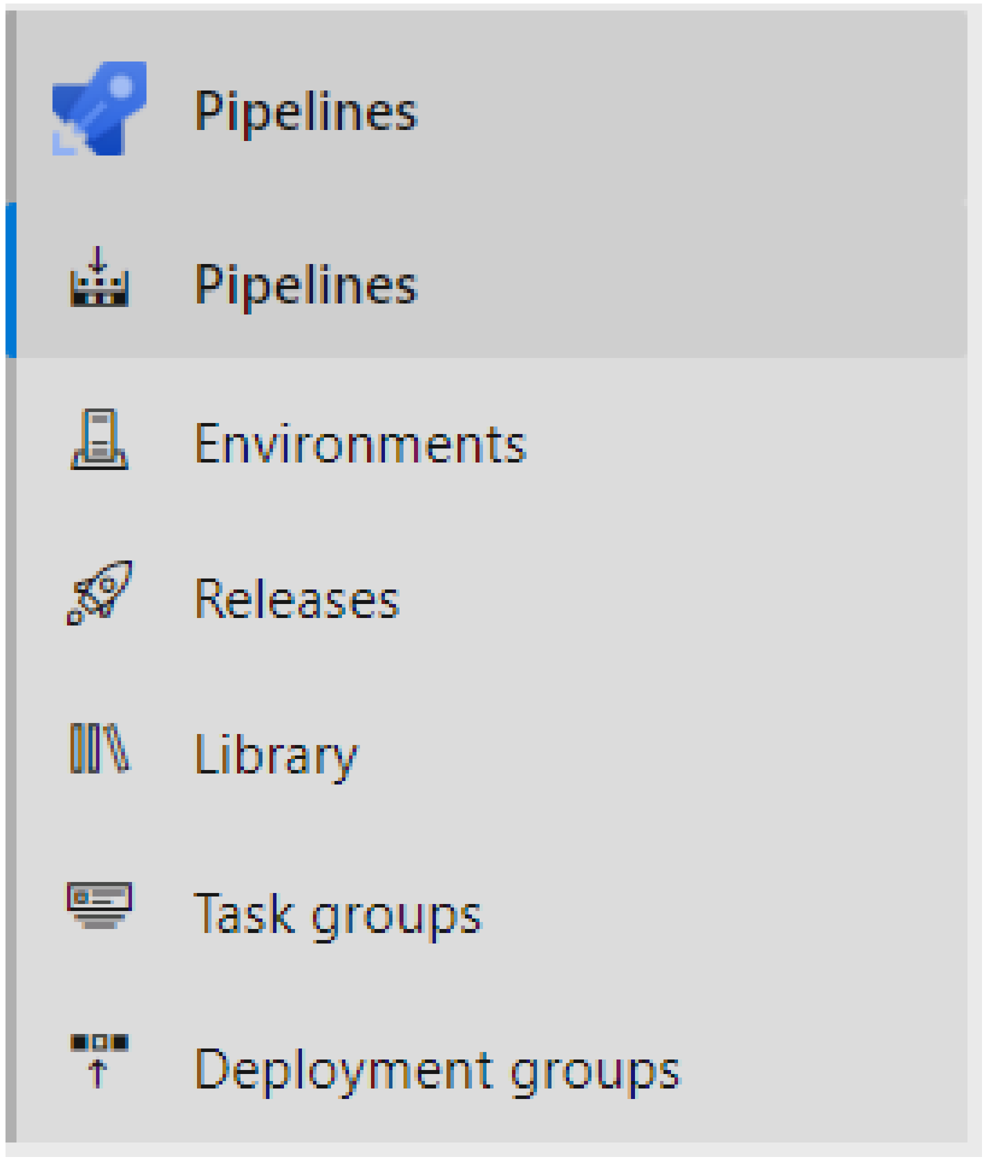
Azure App Service deploy
Deploy to Azure App Service a web or API app using Docker, Java, .N


Settings

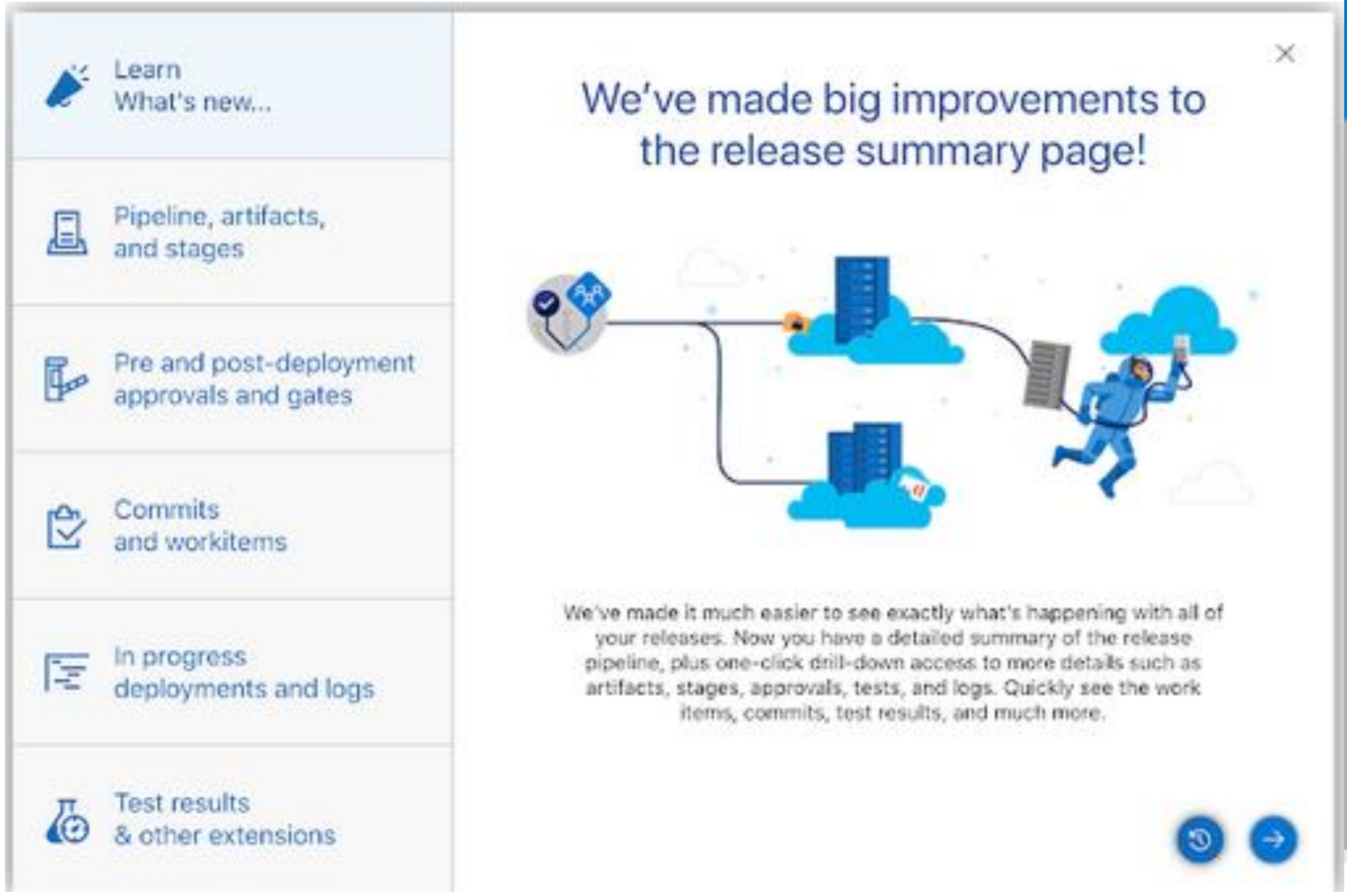
```
- task: Ant@1
  inputs:
    buildFile: 'build.xml'
    options: 
    publishJUnitResults: true
    testResultsFiles: '**/TEST-*.xml'
    javaHomeOption: 'JDKVersion'

- script: |
    echo Add other tasks to build, test, and deploy your project.
    echo See https://aka.ms/yaml
  displayName: 'Run a multi-line script'
```

Environment

An environment is a collection of resources, where you deploy your application. It can contain one or more virtual machines, containers, web apps, or any service that's used to host the application being developed. A pipeline might deploy the app to one or more environments after build is completed and tests are run.



Releases – It's a Pipeline



Learn
What's new...

Pipeline, artifacts, and stages

Pre and post-deployment approvals and gates

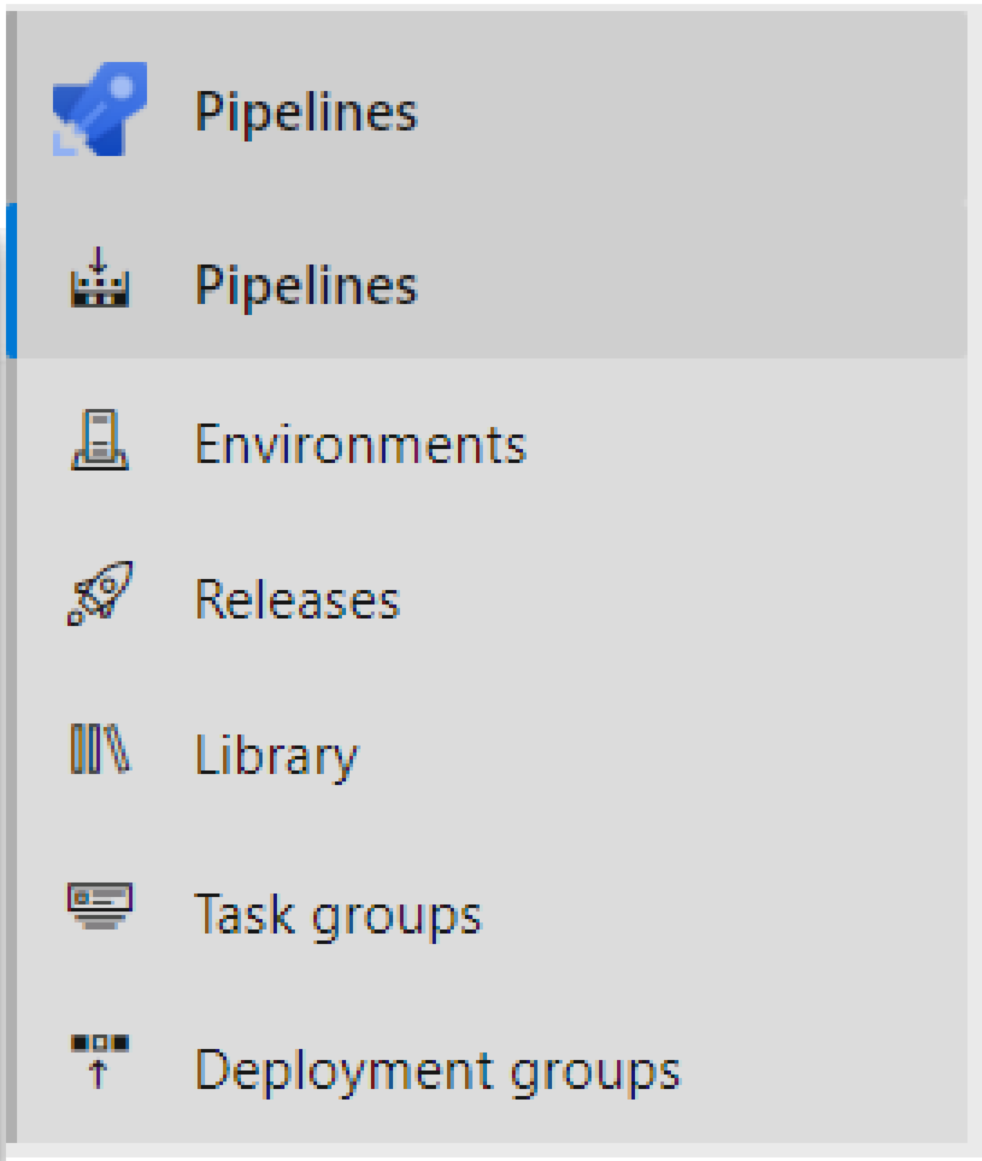
Commits and workitems

In progress deployments and logs

Test results & other extensions

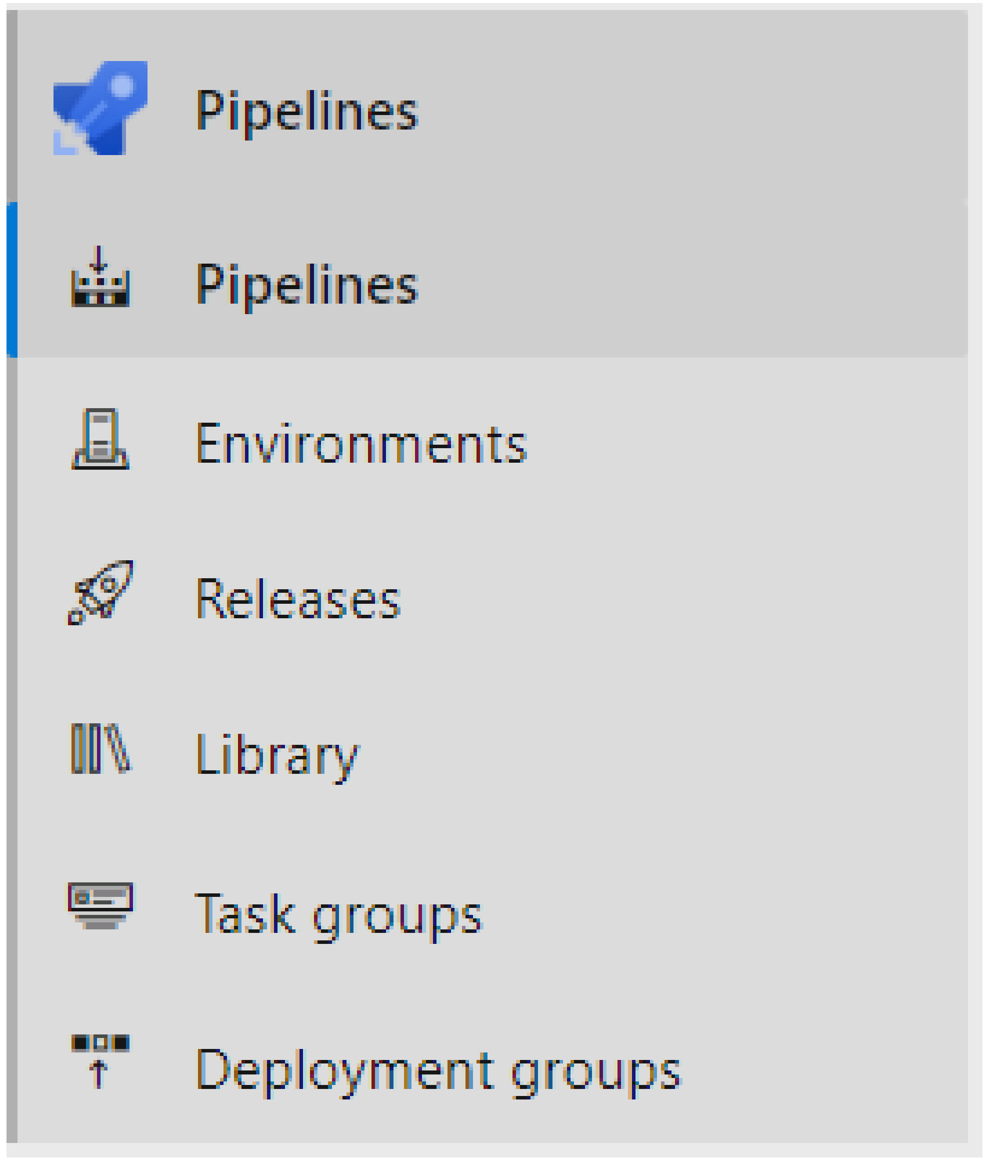
We've made big improvements to the release summary page!

We've made it much easier to see exactly what's happening with all of your releases. Now you have a detailed summary of the release pipeline, plus one-click drill-down access to more details such as artifacts, stages, approvals, tests, and logs. Quickly see the work items, commits, test results, and much more.



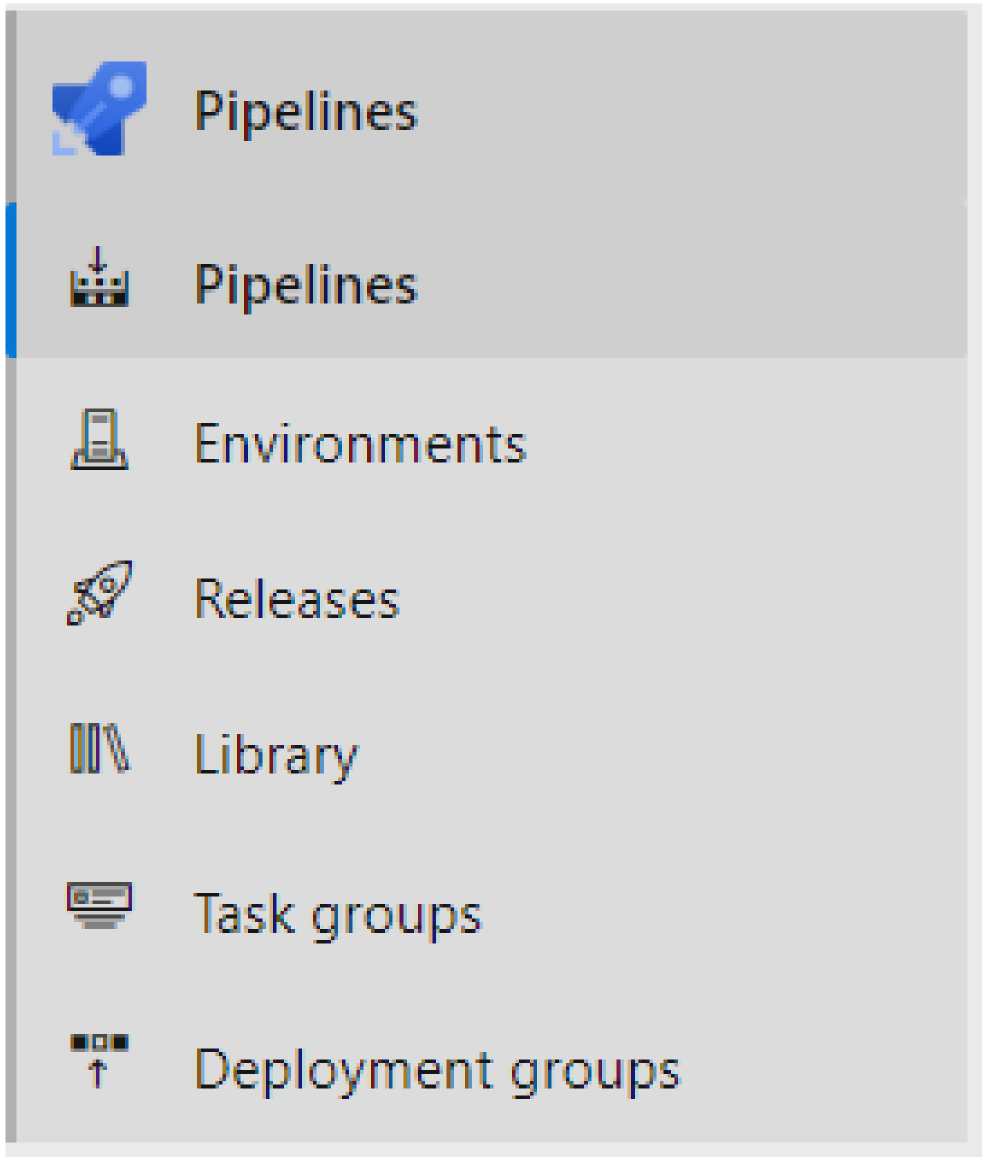
Library

A *library* is a collection of build and release assets for an Azure DevOps project. Assets defined in a library can be used in multiple build and release pipelines of the project. The **Library** tab can be accessed directly in Azure Pipelines.



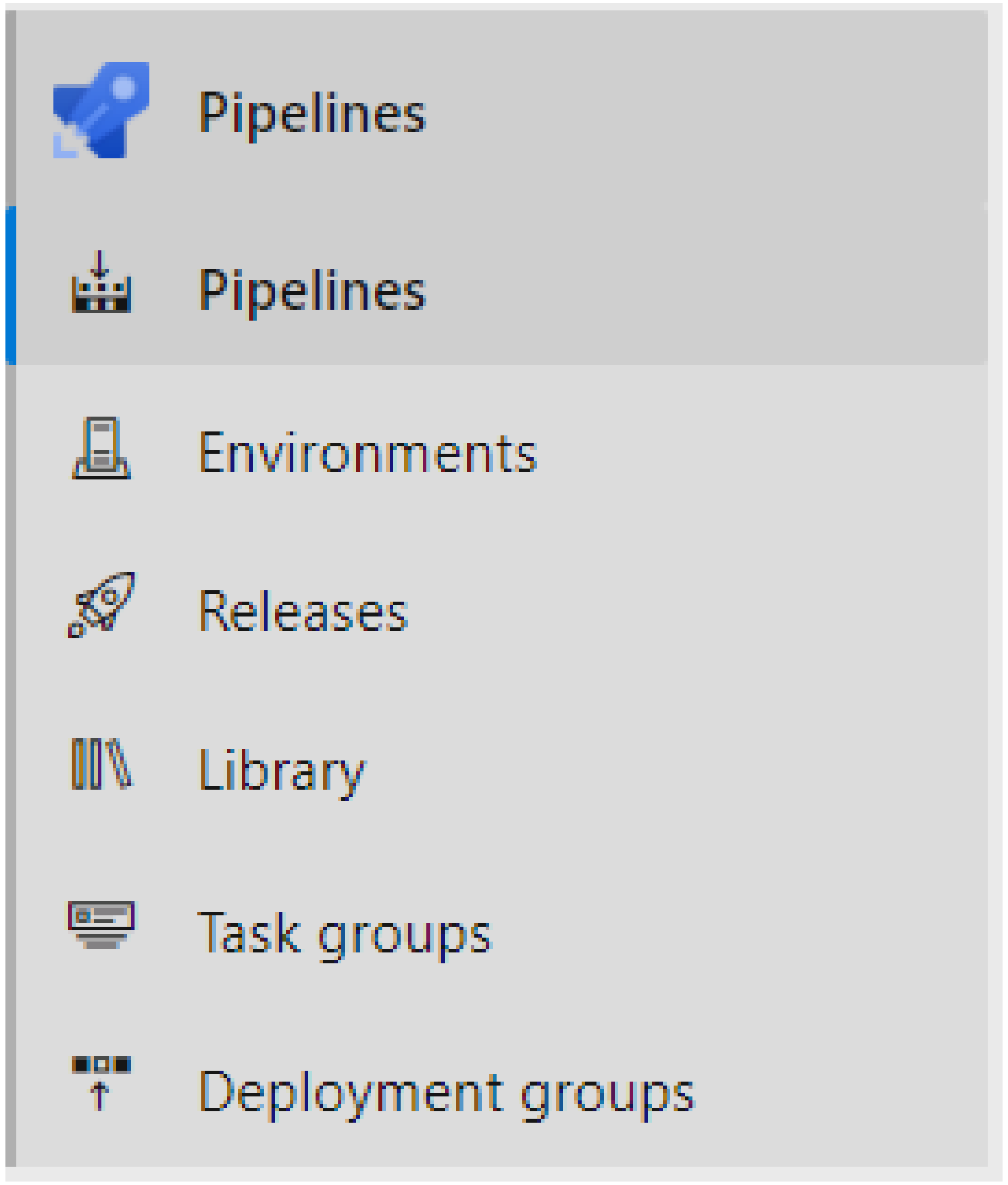
Task Group

A *task group* allows you to encapsulate a sequence of tasks, already defined in a build or a release pipeline, into a single reusable task that can be added to a build or release pipeline, just like any other task. You can choose to extract the parameters from the encapsulated tasks as configuration variables, and abstract the rest of the task information.



Deployment group

A deployment group is a set of deployment target machines that have agents installed. A deployment group is just another grouping of agents, like an agent pool. You can set the deployment targets in a pipeline for a job using a deployment group. Learn more about provisioning agents for deployment groups.



Types of Pipeline

Build Pipeline
Release Pipeline

How a Build Is Set Up

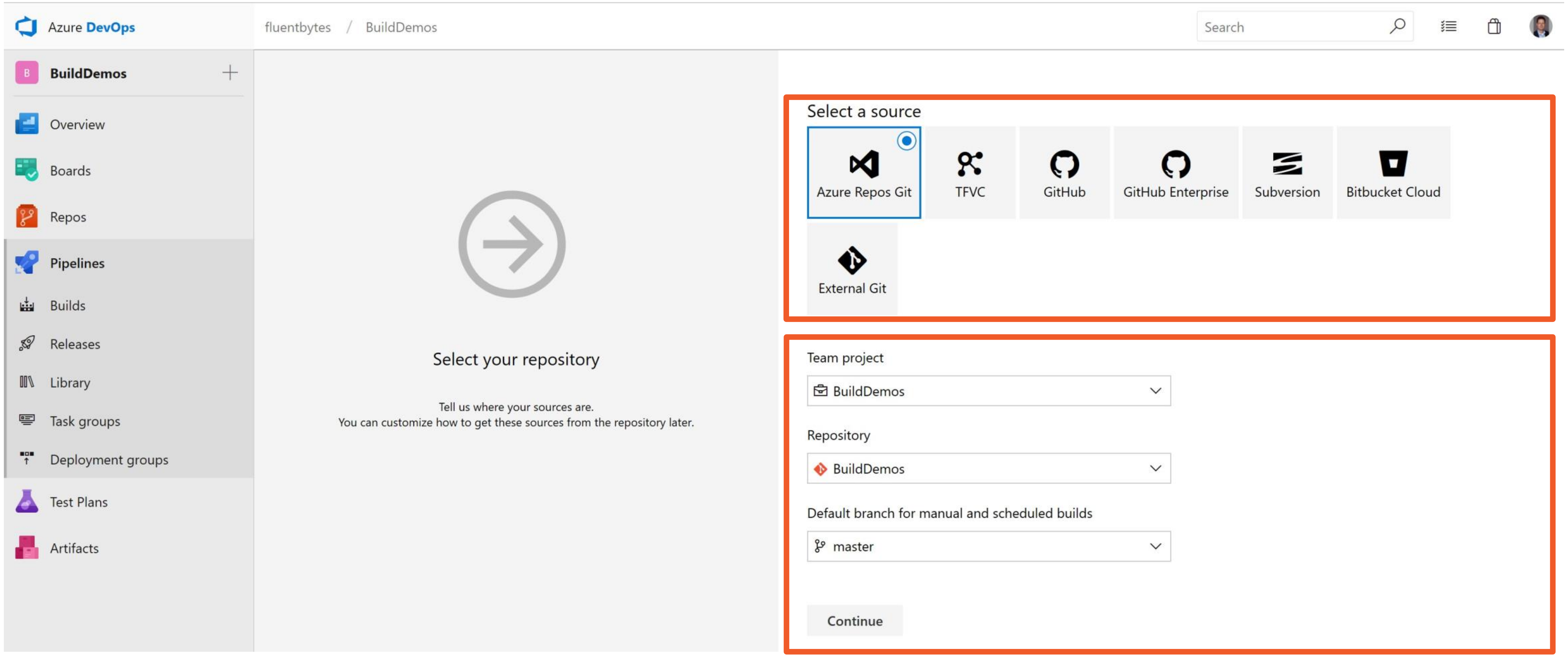
Build definition

Build Steps / Tasks

Build Agent

Pipeline
- Use the classic editor

Select the Build Source



Select your repository

Tell us where your sources are.
You can customize how to get these sources from the repository later.

Select a source

[✓ Azure Repos Git] TFVC GitHub GitHub Enterprise Subversion Bitbucket Cloud External Git

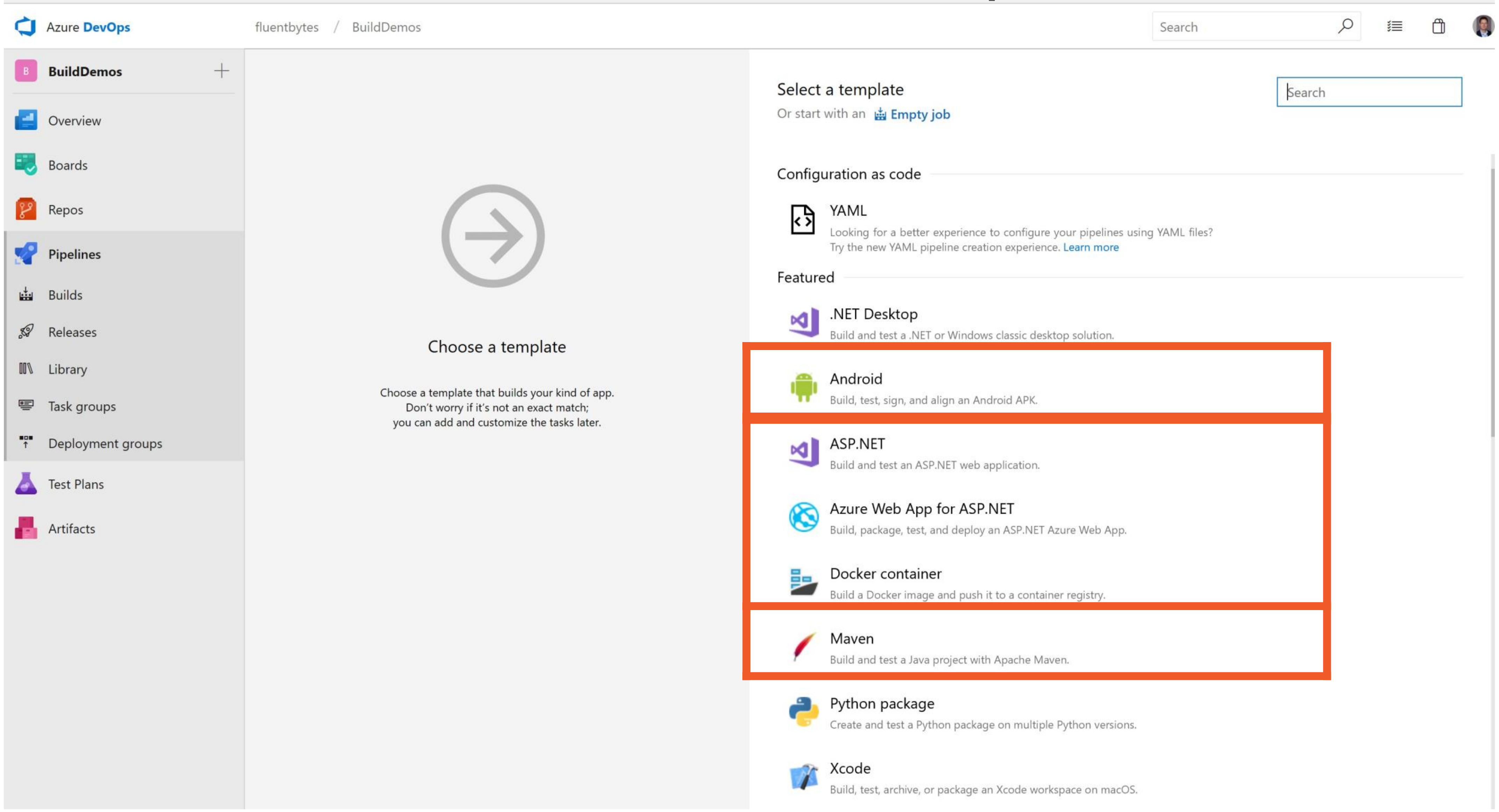
Team project: BuildDemos

Repository: BuildDemos

Default branch for manual and scheduled builds: master

Continue

Select the Build Template



Azure DevOps fluentbytes / BuildDemos Search

BuildDemos

Overview
Boards
Repos
Pipelines
Builds
Releases
Library
Task groups
Deployment groups
Test Plans
Artifacts

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.

Select a template

Or start with an Empty job

Search

Configuration as code

YAML
Looking for a better experience to configure your pipelines using YAML files?
Try the new YAML pipeline creation experience. Learn more

Featured

.NET Desktop
Build and test a .NET or Windows classic desktop solution.

Android
Build, test, sign, and align an Android APK.

ASP.NET
Build and test an ASP.NET web application.

Azure Web App for ASP.NET
Build, package, test, and deploy an ASP.NET Azure Web App.

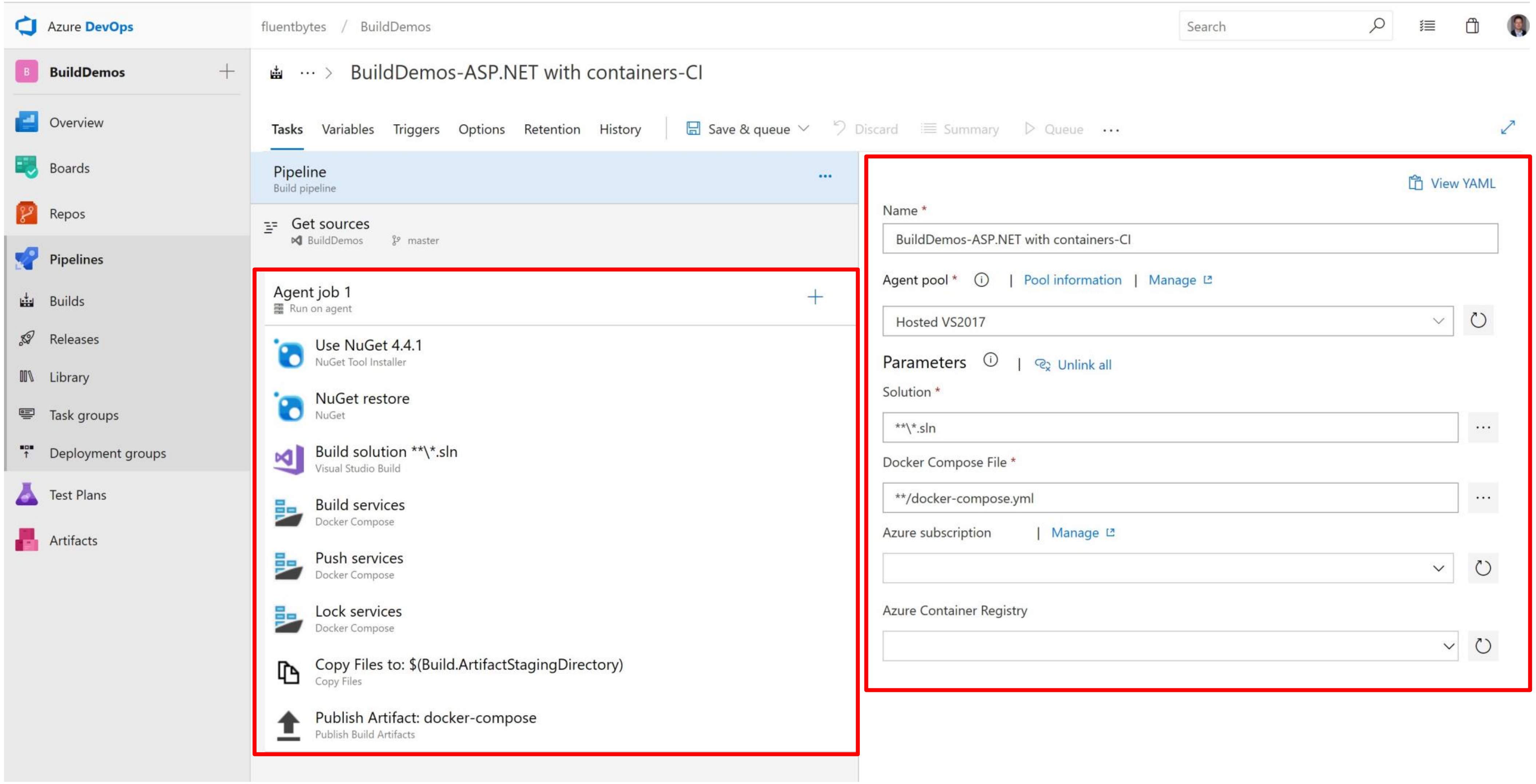
Docker container
Build a Docker image and push it to a container registry.

Maven
Build and test a Java project with Apache Maven.

Python package
Create and test a Python package on multiple Python versions.

Xcode
Build, test, archive, or package an Xcode workspace on macOS.

Configure the Build Tasks



BuildDemos-ASP.NET with containers-CI

Pipeline
Build pipeline

Get sources
BuildDemos master

Agent job 1
Run on agent

Use NuGet 4.4.1
NuGet Tool Installer

NuGet restore
NuGet

Build solution ***.sln
Visual Studio Build

Build services
Docker Compose

Push services
Docker Compose

Lock services
Docker Compose

Copy Files to: $(Build.ArtifactStagingDirectory)
Copy Files

Publish Artifact: docker-compose
Publish Build Artifacts

Name *
BuildDemos-ASP.NET with containers-CI

Agent pool * | Pool information | Manage
Hosted VS2017

Parameters | Unlink all

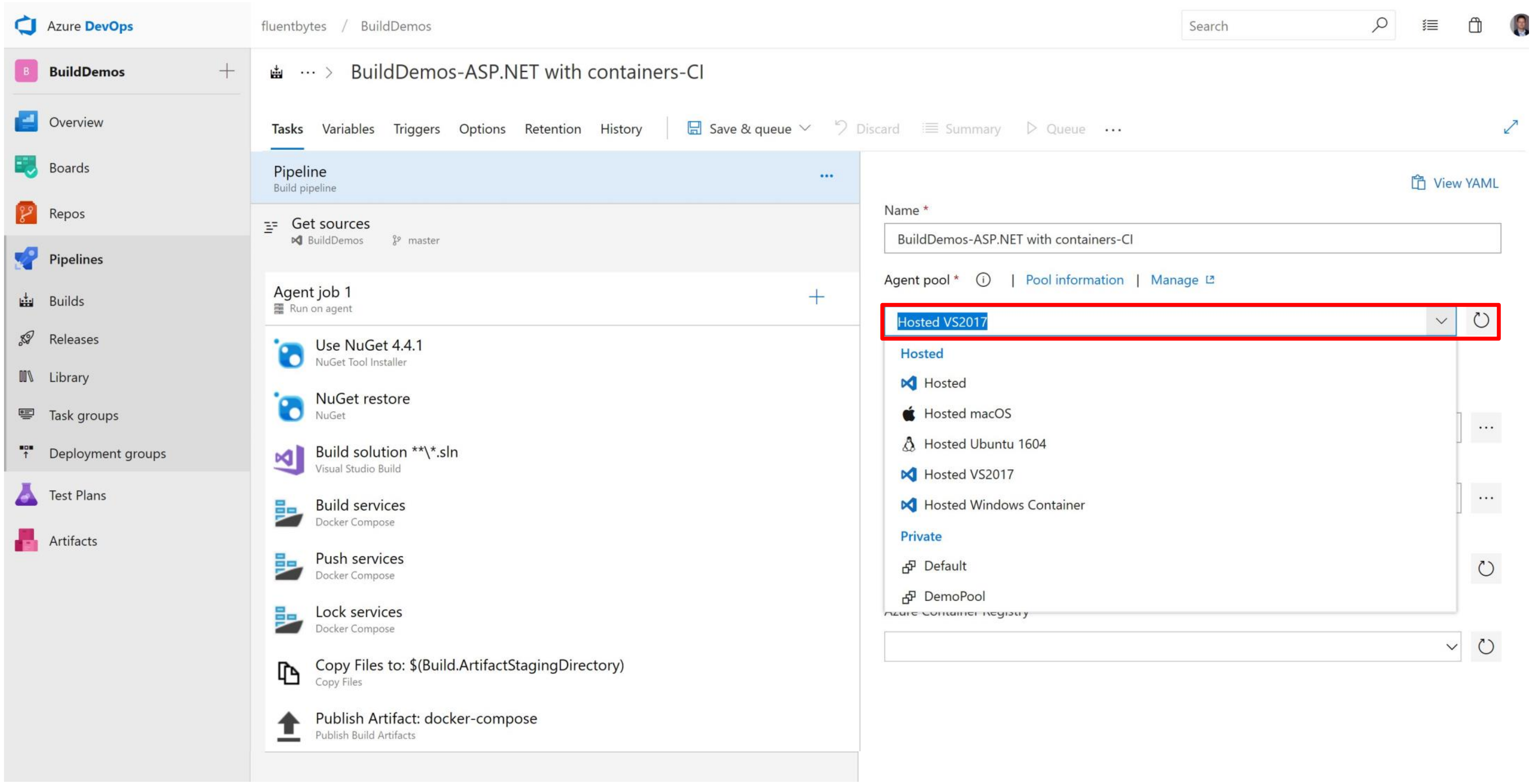
Solution *
***.sln

Docker Compose File *
**/docker-compose.yml

Azure subscription | Manage

Azure Container Registry

Select the Build Agent


Azure DevOps
fluentbytes / BuildDemos
Search

BuildDemos
Overview
Boards
Repos
Pipelines
Builds
Releases
Library
Task groups
Deployment groups
Test Plans
Artifacts

BuildDemos-ASP.NET with containers-CI

Tasks Variables Triggers Options Retention History | Save & queue | Discard | Summary | Queue | ...

Pipeline
Build pipeline

Get sources
BuildDemos master

Agent job 1
Run on agent

Use NuGet 4.4.1
NuGet Tool Installer

NuGet restore
NuGet

Build solution ***.sln
Visual Studio Build

Build services
Docker Compose

Push services
Docker Compose

Lock services
Docker Compose

Copy Files to: $(Build.ArtifactStagingDirectory)
Copy Files

Publish Artifact: docker-compose
Publish Build Artifacts

View YAML

Name *
BuildDemos-ASP.NET with containers-CI

Agent pool * | Pool information | Manage
Hosted VS2017

Hosted
- Hosted
- Hosted macOS
- Hosted Ubuntu 1604
- Hosted VS2017
- Hosted Windows Container

Private
- Default
- DemoPool

Azure Container Registry

Demo

Configure and run an ASP.NET build

Build Infrastructure

Agents and Pipelines

Hosted Build Agent

Pipelines

Custom Build Agent

Build Security

Pools & Queues

(Hosted) Agent

Agent Queues

Agent Pools

Demo

Setting up a custom agent and security

Setting up Continuous Integration Builds

Demo

Continuous integration with Azure DevOps and GitHub

Configuring More Specialized Builds

Outline

Build details

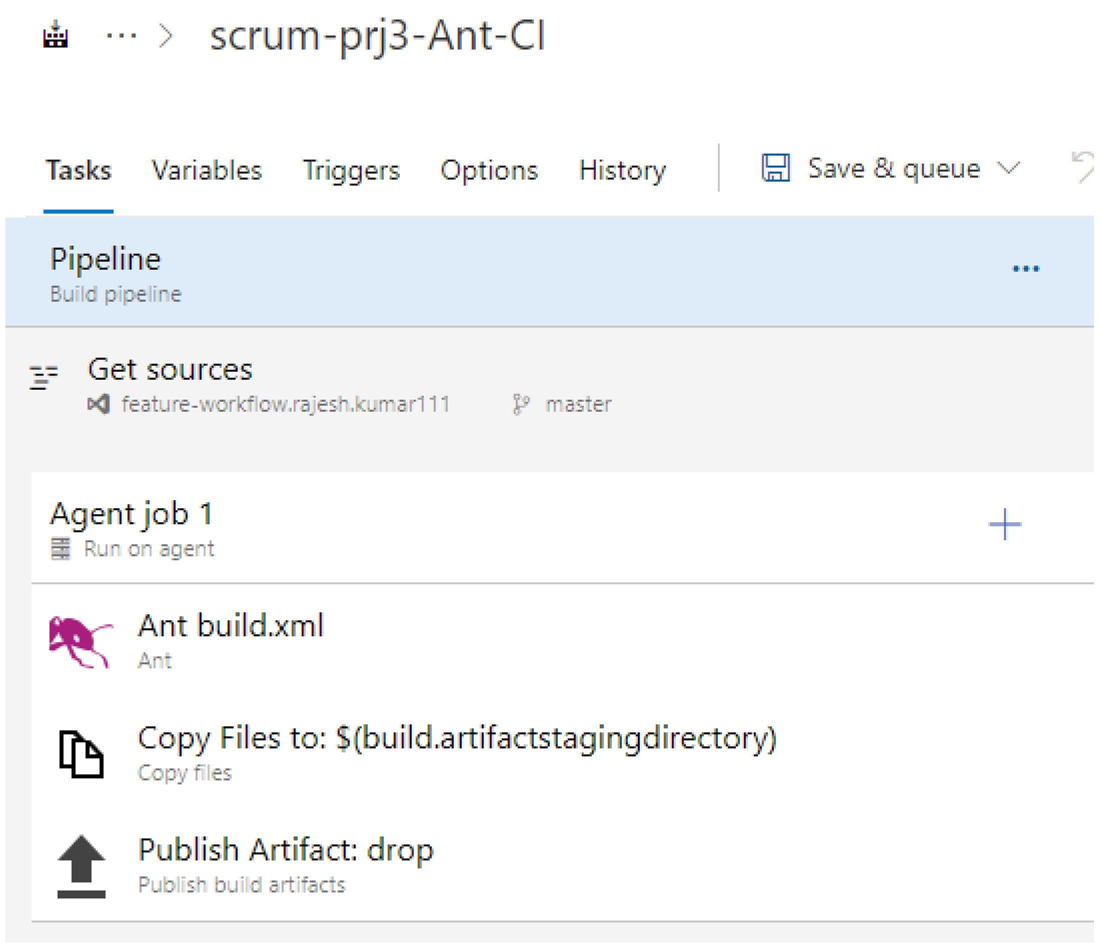
Tasks and the market place

Optimize your builds

Yaml builds

Build Details

Build Variables


scrum-prj3-Ant-CI

Pipeline
Build pipeline

Get sources
feature-workflow.rajesh.kumar111 master

Agent job 1
Run on agent

Ant build.xml
Ant

Copy Files to: $(build.artifactstagingdirectory)
Copy files

Publish Artifact: drop
Publish build artifacts

Custom Variables

$(variablename)

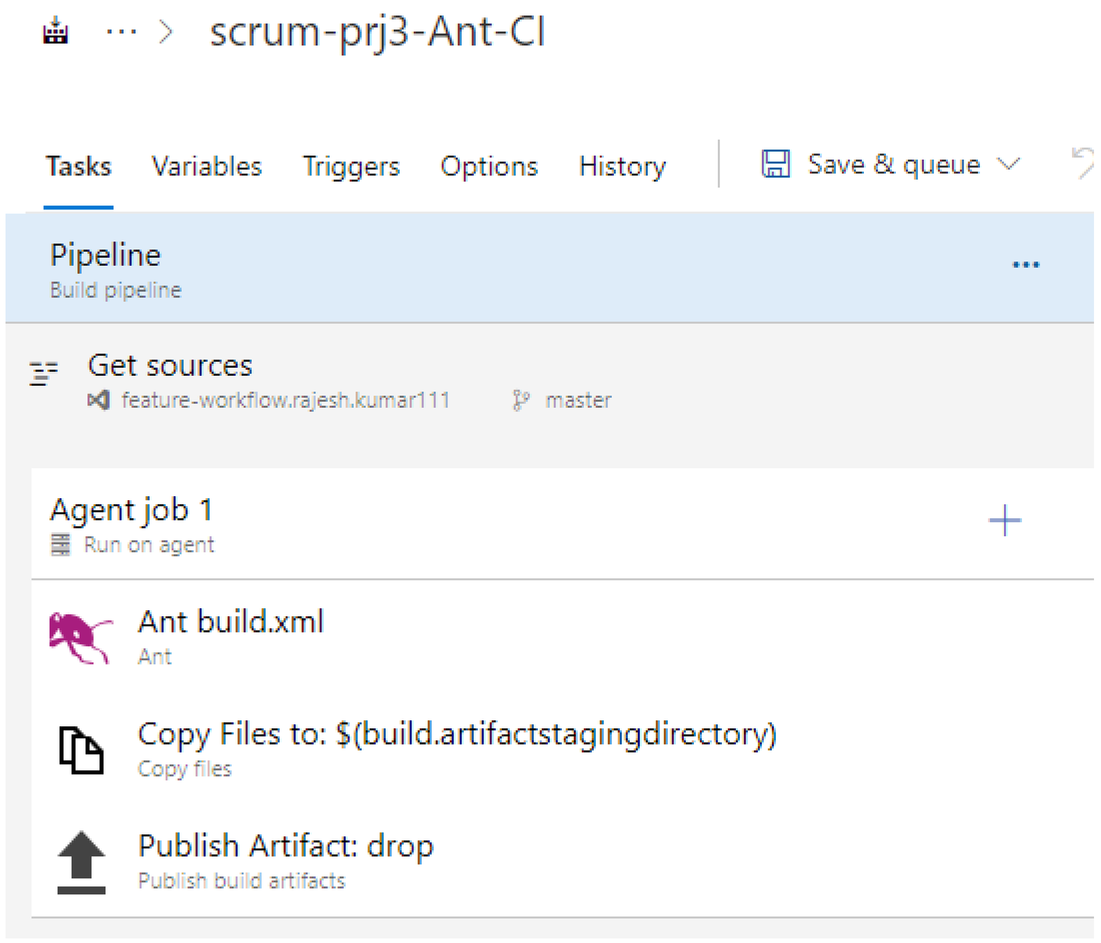
Build In Variables

Variables From PowerShell

Secrets

Environment Variable

Build Triggers


scrum-prj3-Ant-CI

Pipeline
Build pipeline

Get sources
feature-workflow.rajesh.kumar111 master

Agent job 1
Run on agent

Ant build.xml
Ant

Copy Files to: $(build.artifactstagingdirectory)
Copy files

Publish Artifact: drop
Publish build artifacts

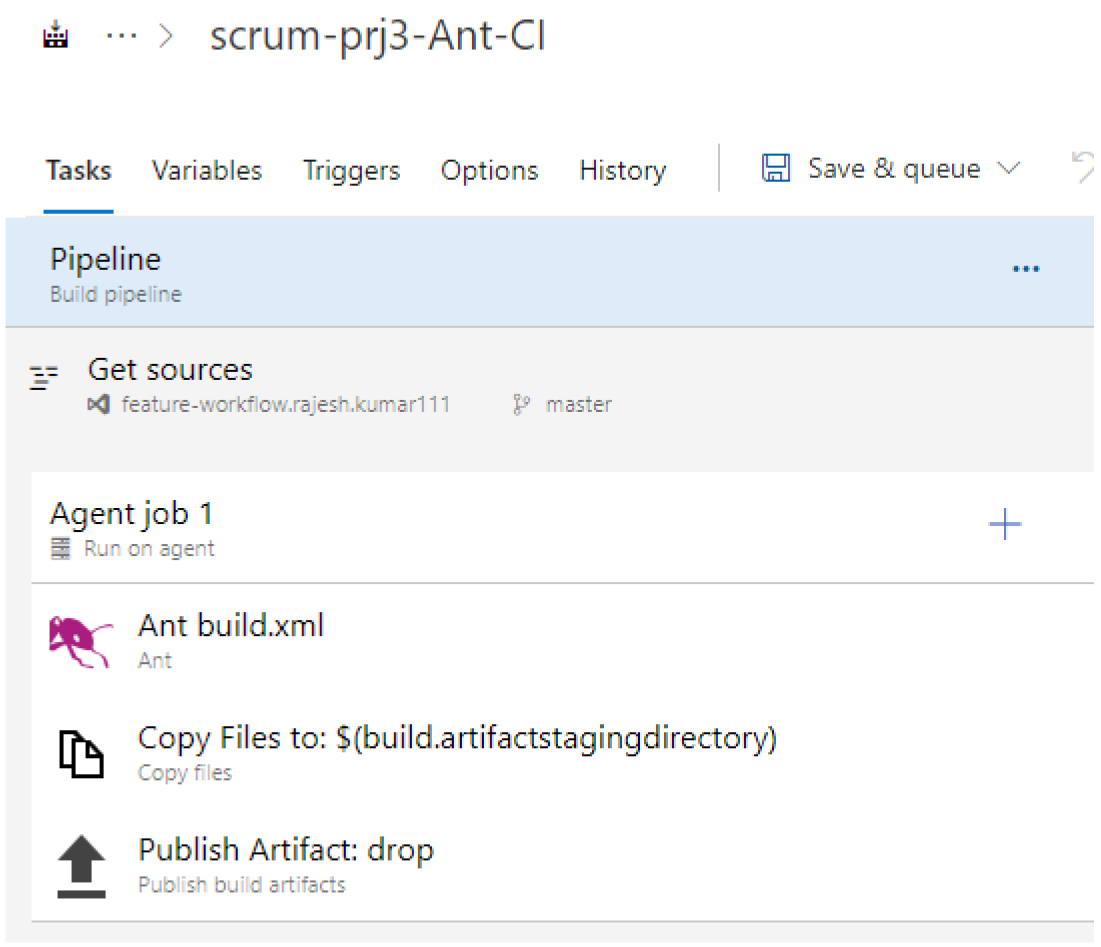
Continuous Integration

Branch Filters

Pull Request

Gated Check-in

Build Options



Pipeline
Build pipeline

Get sources
feature-workflow.rajesh.kumar111 master

Agent job 1
Run on agent

Ant build.xml
Ant

Copy Files to: $(build.artifactstagingdirectory)
Copy files

Publish Artifact: drop
Publish build artifacts

Build Job Properties

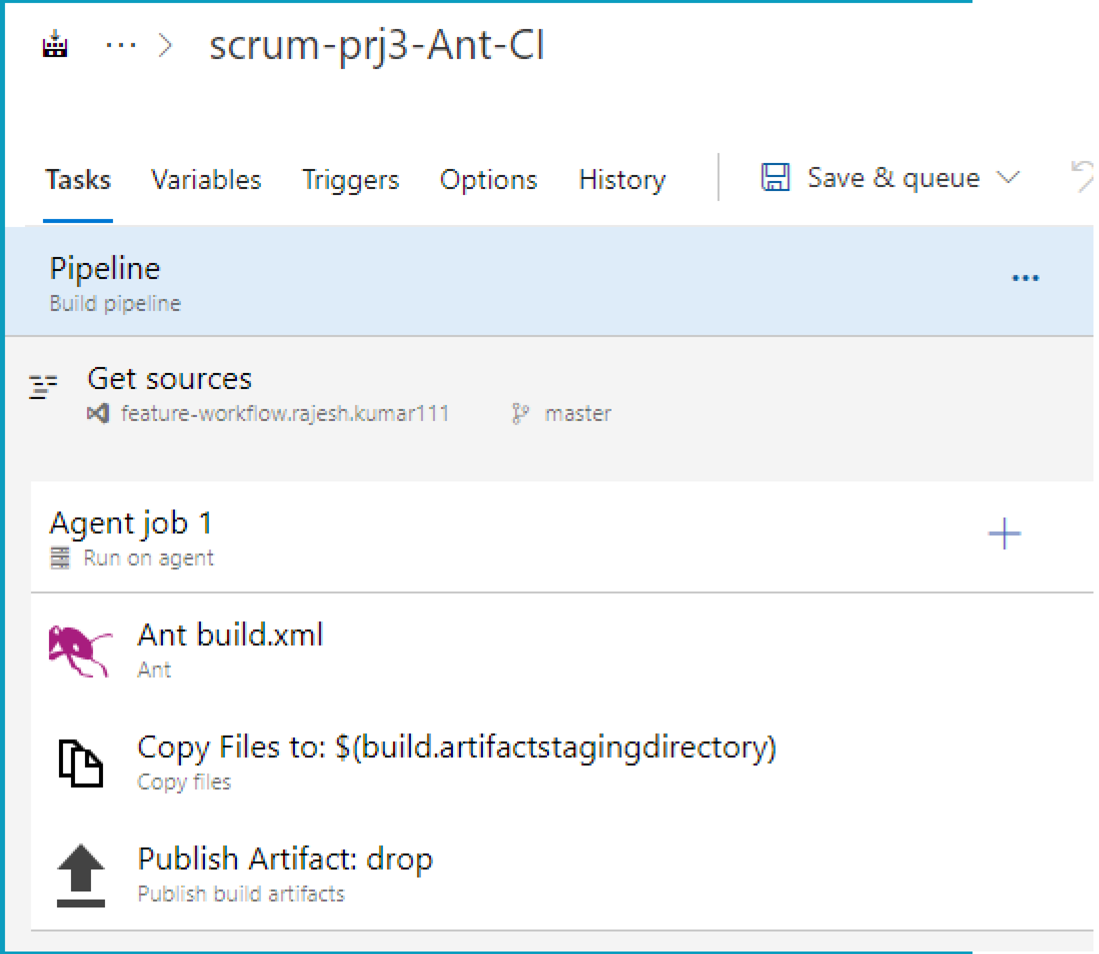
Demands

Build Number Format

Work-Items

Oauth Token

Build Retention & History


scrum-prj3-Ant-CI

Tasks Variables Triggers Options History | Save & queue

Pipeline
Build pipeline

Get sources
feature-workflow.rajesh.kumar111 master

Agent job 1
Run on agent

Ant build.xml
Ant

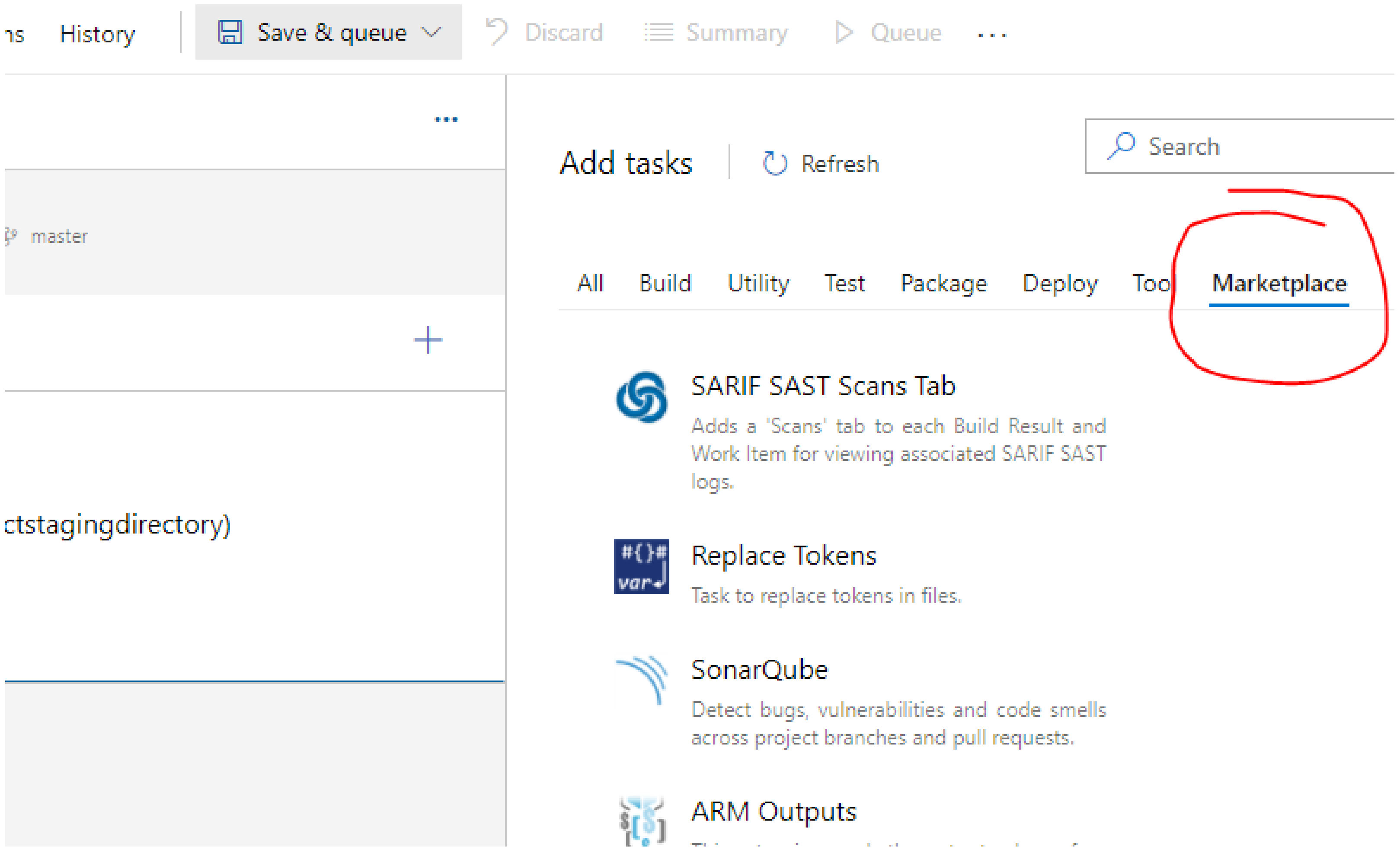
Copy Files to: $(build.artifactstagingdirectory)
Copy files

Publish Artifact: drop
Publish build artifacts

Retention Policy

Change History

Tasks and the Market Place



SARIF SAST Scans Tab

Adds a 'Scans' tab to each Build Result and Work Item for viewing associated SARIF SAST logs.

Replace Tokens

Task to replace tokens in files.

SonarQube

Detect bugs, vulnerabilities and code smells across project branches and pull requests.

ARM Outputs

Demo

More build details for various builds

Optimize Your Builds

Fast Feedback Starts with Fast Build Server(s)

Configuration optimized for the task at hand:

- Fast IO (local disks, SSD preferred)
- Fast CPU

Located 'near' the sources and the drop location

Different Builds for Different Purposes

CI Build

(Continuous Integration)

Compile/test

Nightly build

(schedule)

Compile
Regression
Provisioning
Code Analysis

Release build

(manual)

Compile
Test
Regression
Co nfig management
Archive

Optimizing the Build

Enable parallel build executionfor faster results

Enable parallel test executionfor faster results

Using multipliers to scalebuilds over multiple agents

Make integration tests part of the release process instead of the build

Publish to a NuGet feed at the end of a build

Demo

Optimizing the build to run parallel

Configuration and Infrastructureas Code

Outline

Configuration as code

Transform configuration

Infrastructure as code

Artifact location

Configuration as Code

Configuration as Code

Important Continuous Delivery concept

Keep configuration in Source Control

The build outputs the needed artifacts

Transform Configuration

Configuration and Secrets

Have admin define secrets in variables

Use the build to replace secrets

Use transform tasks

Infrastructure as Code

Important Infrastructure Artifacts

PowerShell Scripts

PowerShell DSC scripts

ARM Templates

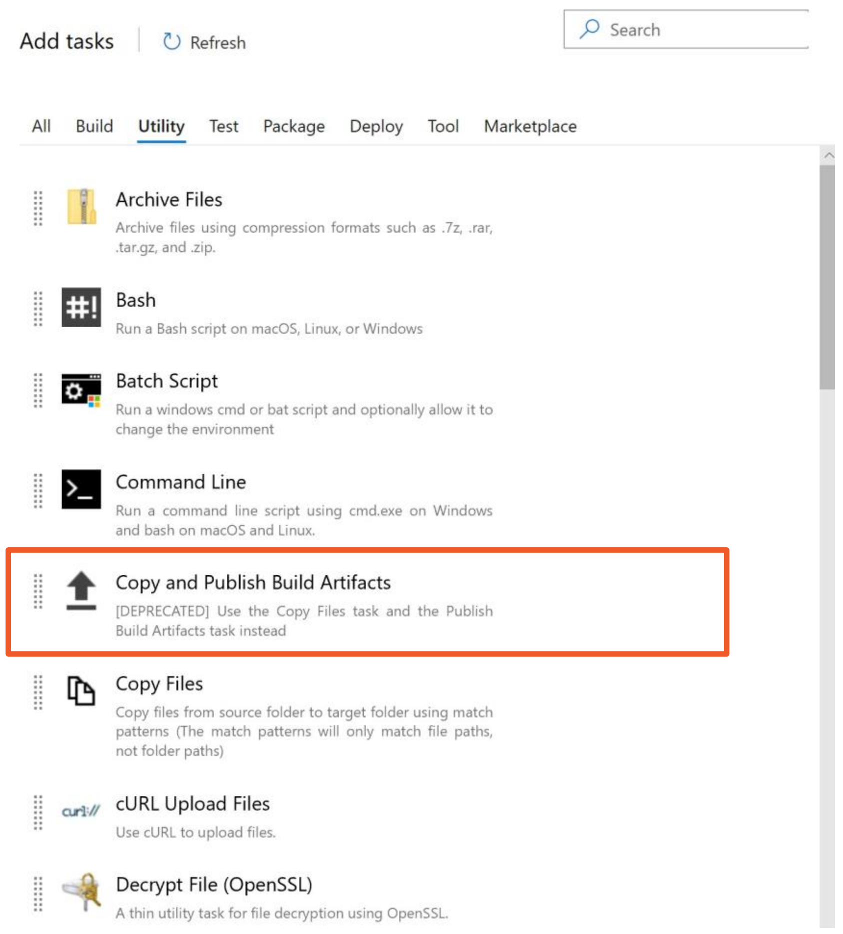
Bash scripts

Puppet modules

Chef recipes

Artifact Location

Artifacts and the Artifact Store



Add tasks | Refresh

Search

All Build Utility Test Package Deploy Tool Marketplace

Archive Files
Archive files using compression formats such as .7z, .rar, .tar.gz, and .zip.

Bash
Run a Bash script on macOS, Linux, or Windows

Batch Script
Run a windows cmd or bat script and optionally allow it to change the environment

Command Line
Run a command line script using cmd.exe on Windows and bash on macOS and Linux.

Copy and Publish Build Artifacts
[DEPRECATED] Use the Copy Files task and the Publish Build Artifacts task instead

Copy Files
Copy files from source folder to target folder using match patterns (The match patterns will only match file paths, not folder paths)

cURL Upload Files
Use cURL to upload files.

Decrypt File (OpenSSL)
A thin utility task for file decryption using OpenSSL.

Azure DevOps has an artifact location buildin

You can copy yourartifacts to the artifactrepository

This is where you will pull them from when we are using release pipelines ina later stage

Demo

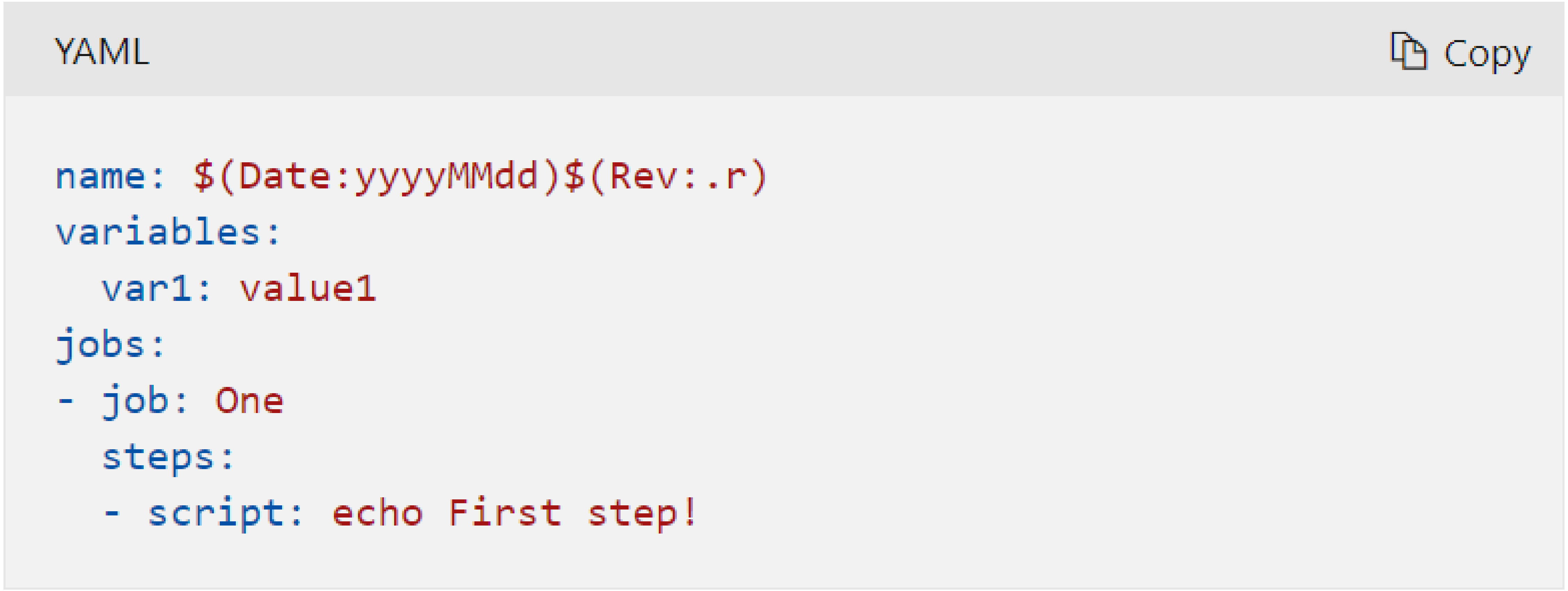
Adding configuration and infrastructure as code to your build

Pipeline

- Use the Pipeline YAML

This hierarchy is reflected in the structure of a YAML file like:

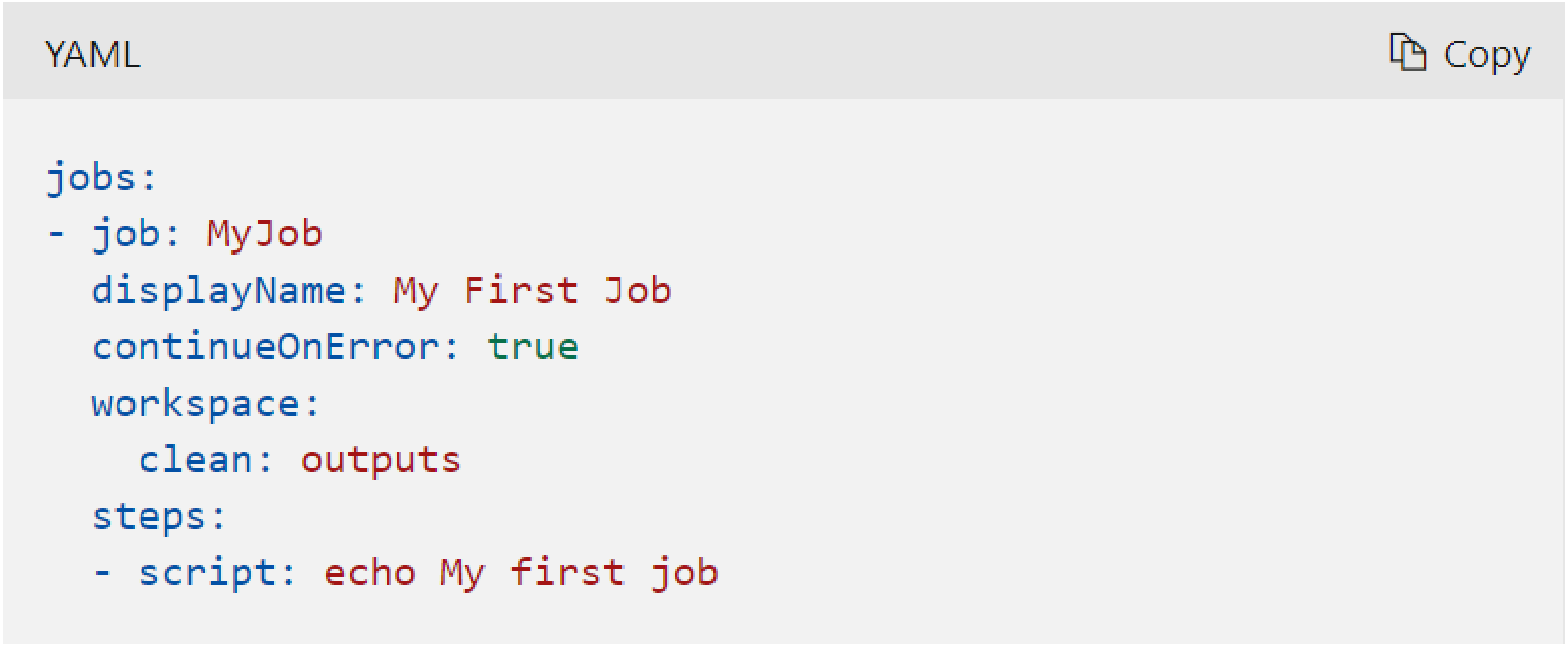
- Pipeline
  - Stage A
    - Job 1
      - Step 1.1
      - Step 1.2
      - ...
    - Job 2
      - Step 2.1
      - Step 2.2
      - ...
  - Stage B
    - ...



```yaml
name: $(Date:yyyyMMdd)$(Rev:.r)
variables:
  var1: value1
jobs:
- job: One
  steps:
  - script: echo First step!
```

YAML

```yaml
stages:
- stage: Build
  jobs:
  - job: BuildJob
    steps:
    - script: echo Building!
- stage: Test
  jobs:
  - job: TestOnWindows
    steps:
    - script: echo Testing on Windows!
  - job: TestOnLinux
    steps:
    - script: echo Testing on Linux!
- stage: Deploy
  jobs:
  - job: Deploy
    steps:
    - script: echo Deploying the code!
```



```yaml
jobs:
- job: MyJob
  displayName: My First Job
  continueOnError: true
  workspace:
    clean: outputs
  steps:
  - script: echo My first job
```

https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema%2Cparameter-schema