**Introduction to Azure Pipelines**

# What is Azure Pipelines?

* 6 minutes

Microsoft Azure Pipelines is a cloud service that you can use to automatically build, test, and deploy your code project. You can also make it available to other users. And it works with just about any language or project type.

Mara is excited about replicating the team's build process on Azure Pipelines. Amita, the tester, finally has some free time and wants to catch up. Mara decides that now is a great time to tell her about her plan: setting up an automated build pipeline for the Space Game web site by using Azure Pipelines.

When she hears Mara's plan, Amita is a bit hesitant. But since Mara's plan is to replicate the build process but not replace it, she's also curious. She knows the build process could use some improvements.

**Amita:** It sounds like an interesting exercise, but you must want to prove a DevOps point!

**Mara:** You already know me so well!

**Amita:** What improvements do you expect to see, especially because you're going to do what we already do?

**Mara:** I think that just moving to Azure Pipelines will bring many benefits. Remember, Azure Pipelines is a cloud service. We can use it to automatically build and test code. And it will be available to others as well. It works with just about any language or project type.

Our build server has problems. Even keeping it up-to-date is hard. Because Azure Pipelines provides build servers that Microsoft hosts and maintains, it always has the latest patches and security updates. We won't have to worry about maintaining build servers.

Also, we have all kinds of scripts written by different people. We don't even understand how some of them work. Azure Pipelines comes with a catalog of tasks. A task is a packaged script or procedure that's been abstracted with a set of inputs. I'm going to try to map what our build scripts do to those tasks. At least we can standardize how things get done and increase the level of automation.

And Azure Pipelines works with many different languages and application types. If we want to expand in those directions, we won't have to retool.

**Amita:** I know it's selfish, but why do I care? One of my big problems is that I never know when a build is ready to test. Sometimes someone remembers to update the spreadsheet, but many times they forget. It seems like I'm the last person to know.

**Mara:** Right, that's something we can easily fix. We can set up the pipeline to notify you automatically, either through email or some other notification, when a build is ready. You'll never have to wait for someone to remind you again.

**Amita:** Okay, so your goal right now is to build the app and let me know when it's ready?

**Mara:** Right! Of course, I've got bigger plans. I know you're all going to love this first step, so I want to build on it to give us true continuous integration.

## What is continuous integration?

Continuous integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control.

CI encourages developers to share their code and unit tests by merging their changes into a shared version control repository after every small task completion. Committing code triggers an automated build system to grab the latest code from the shared repository and to build, test, and validate the full master branch (also known as the trunk or main).

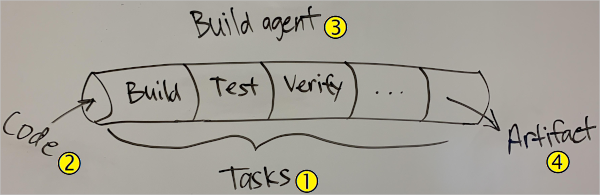
In this short video, Abel Wang, Cloud Advocate at Microsoft, explains the concept of continuous integration.

**Ask Abel**

**Amita:** Give me the 5-minute rundown on continuous integration.

**Mara:** Let me draw you a picture.

Mara moves to the whiteboard and draws the pipeline.



This is my continuous integration (CI) pipeline. CI is the process of automating the build and testing of code every time a team member commits changes to version control. I know we don't do automated testing yet, but give it time.

A pipeline defines the continuous integration process for the app. It's made up of steps called tasks Callout 1. It can be thought of as a script that defines how your build, test, and deployment steps are run. I'm going to try to map our scripts to tasks.

The pipeline runs when you submit code changes Callout 2. You can configure the pipeline to run automatically, or you can run it manually. You connect your pipeline to a source repository like GitHub, Bitbucket, or Subversion. We'll use GitHub. One of our tasks for this sprint is to start using GitHub.

A build agent Callout 3 builds or deploys the code. When your build or deployment runs, the system begins one or more jobs. An agent is installable software that runs one build or deployment job at a time. Because we're using Azure Pipelines, we can use a Microsoft-hosted agent. With Microsoft-hosted agents, maintenance and upgrades are taken care of for us. Each time we run a pipeline, we'll get a fresh virtual machine. There are six virtual machine images to choose from, including Ubuntu 18.04, which is what we use.

The final product of the pipeline is a build artifact Callout 4. Think of an artifact as the smallest compiled unit that we need to test or deploy the app. For example, an artifact can be:

* A Java or .NET application packaged into a .jar or .zip file.
* A C++ or JavaScript library.
* A virtual machine, cloud, or Docker image.

And that's it. I know we can do this.

**Amita:** It sounds great. Let's see what you have to do to get it to work and how long it takes you. You can give us all a demo.

**Mara:** Will do!

## Implement and manage build infrastructure

Now that you and the team are familiar with Azure Pipelines, let's discuss a few of the configuration details. Specifically, we'll talk a bit more about build agents, integrating with third-party systems, managing your pipeline configuration, and using concurrent pipelines.

### Build agents

As you know, a build agent is a piece of installable software that runs one build or deployment job at a time. To build your code or deploy your software you need at least one agent. As you add more code and people, you'll eventually need more than one. Let's examine build agents in a bit more depth.

### The differences between implementing hosted and private agents

You can use either a Microsoft-hosted or a private agent. What are the differences?

If your pipelines are in Azure Pipelines, then you've got a convenient option to build and deploy using a Microsoft-hosted agent. With Microsoft-hosted agents, maintenance and upgrades are taken care of for you. Each time you run a pipeline, you get a fresh virtual machine. The virtual machine is discarded after one use.

For many teams this is the simplest way to build and deploy. You can try it first and see if it works for your build or deployment. If not, you can use a self-hosted agent.

An agent that you set up and manage on your own to run build and deployment jobs is a self-hosted agent. You can use self-hosted agents in Azure Pipelines. Self-hosted agents give you more control and let you install any software you need for your builds and deployments.

You can install the agent on Linux, macOS, or Windows machines. You can also install an agent on a Linux Docker container. After you've installed the agent on a machine, you can install any other software on that machine as required by your build or deployment jobs.

### Agent pools

Instead of managing each agent individually, you can organize agents into agent pools. An agent pool defines the sharing boundary for all agents in that pool. In Azure Pipelines, agent pools are scoped to the Azure DevOps organization so you can share an agent pool across projects.

A project agent pool provides access to an organization agent pool. When you create a build or release pipeline, you specify which pool it uses. Pools are scoped to your project so you can only use them across build and release pipelines within a project.

To share an agent pool with multiple projects, in each of those projects, you create a project agent pool pointing to an organization agent pool. While multiple pools across projects can use the same organization agent pool, multiple pools within a project cannot use the same organization agent pool. Also, each project agent pool can use only one organization agent pool.

### Agent queues

If you are a project team member, you create and manage agent build queues from the agent pools tab in project settings.

### Service endpoints for integration with third-party systems

Service endpoints are a way for Azure DevOps to connect to external systems or services. They are a bundle of securely stored properties that includes but is not limited to:

* Service name
* Description
* Server URL
* Certificates or tokens
* User names and passwords

Extensions are then able to access the service endpoint to get the stored details to perform the necessary operations on that service.

### Concurrent pipelines

You can run concurrent pipelines (also called parallel jobs) in Azure Pipelines. One parallel job in Azure Pipeline lets you run a single build or release job at any given time. This rule is true whether you run the job on Microsoft-hosted or self-hosted agents. Parallel jobs are purchased at the organization level, and they are shared by all projects in an organization.

#### Microsoft-hosted CI/CD

If you want to run your builds and releases on machines that Microsoft manages, use Microsoft-hosted parallel jobs. Your jobs run on the pool of hosted agents. Microsoft provides a free tier of service by default for every organization. [Consult the Azure DevOps documentation](https://docs.microsoft.com/en-us/azure/devops/pipelines/licensing/concurrent-jobs?view=azure-devops) to see the criteria.

If you want Azure Pipelines to orchestrate your builds and releases, but use your own machines to run them, use self-hosted parallel jobs. You start by deploying agents on your machines. You can register any number of these self-hosted agents in your organization. Microsoft charges based on the number of jobs you want to run at a time, not the number of agents registered.

#### Plan a strategy for concurrent pipelines

Here are some steps to take to plan for concurrent pipelines.

#### Determine how many parallel jobs you need

Begin by seeing if the free tier offered in your organization is enough for your teams. When you've reached the per-month limit for the free tier of Microsoft-hosted parallel jobs, you can start by buying one parallel job. As the number of queued builds and releases exceeds the number of parallel jobs you have, your build and release queues will grow longer. When you find the queue delays are too long, you can purchase additional parallel jobs as needed. A simple rule of thumb is to estimate that you'll need one parallel job for every four to five users in your organization.