# Hosting Your Own Build Agent in Azure Pipeline

# Introduction

* 1 minute

In this module, you set up your own build agent running on a Microsoft Azure virtual machine.

Up until now, you used a Microsoft-hosted agent that runs Ubuntu to build the Space Game web application. Most of the time, a Microsoft-hosted agent can do everything you need.

However, you occasionally need additional processing power, disk space, or time to build your applications. In this module, you learn how to set up your own build agent, which can run either in the cloud or on-premises. You follow along with the Tailspin web team as they set up a build agent that runs on an Azure virtual machine (VM).

## Learning objectives

In this module, you will:

* Choose when to use Microsoft-hosted build agents and when to host your own.
* Describe your options for managing your own build agents.
* Bring up and configure your own agent to work with Azure Pipelines.
* Connect your agent to a pipeline and build your application.

## Prerequisites

The modules in this learning path form a progression.

To follow the progression from the beginning, be sure to first complete the Evolve your DevOps practices  learning path.

We also recommend you start at the beginning of this learning path, Build applications with Azure DevOps .

If you want to go through just this module, you need to set up a development environment on your Windows, macOS, or Linux system. You need:

* An Azure DevOps organization
* A GitHub  account
* Visual Studio Code
* .NET Core 3.1 SDK
* Git

You can get started with Azure DevOps for free.

This environment lets you complete the exercises in this and future modules. You can also use it to apply your new skills to your own projects.

**Note**

Keep in mind that you can use Azure DevOps to build and deploy almost any kind of application written in any language. In this module, you'll be working with a .NET Core application written in C#.

You don't need to be an expert in .NET or C# to complete this module. You can apply the patterns you learn here to your own projects that use your favorite programming languages and frameworks.

## Meet the team

You met the Space Game web team at Tailspin Toys in previous modules. As a refresher, here are the team members you'll work with in this module.



Andy is the development lead.



Amita is in QA.



Tim is in operations.



Mara just joined as a developer and reports to Andy.

Mara has prior experience with DevOps and is helping the team adopt a more streamlined process by using Azure DevOps.

# Choose a Microsoft-hosted or self-hosted build agent

* 6 minutes

In this unit, you learn about some of the factors to consider when you're choosing a build agent. You learn about some of the benefits and limitations of using a Microsoft-hosted agent, as well as what's involved when you set up your own private build agent.

Let's see what's happening with the Tailspin web team. Tim, from operations, wants to learn more about how build agents work in Azure Pipelines. He starts a conversation with our developers, Andy and Mara.

**Tim:** Hi, Andy and Mara. I've been following how you're using Microsoft Azure Pipelines to build the Space Game web application. But I'm curious to learn more about how this works. Does it connect to one of our build machines?

**Andy:** It's possible to connect it to one of our build machines, but right now we use an agent that's hosted by Microsoft.

**Tim:** But we build on Linux. Microsoft provides Linux build agents?

**Mara:** Yes! In fact, you can choose Windows, Linux, or macOS for your build agent. If your application runs on multiple platforms, you can configure the pipeline to build on all of them.

**Tim:** Interesting. One of the other teams mentioned some of the challenges they're having with their build infrastructure. Perhaps Microsoft Azure Pipelines and either a Microsoft-hosted agent or their own build agent can help?

**Andy:** I'd be interested to know as well. Let's chat a bit more about build agents. Perhaps you can share what you learn with the other team.

## What are build agents and agent pools?

A build agent is a system that performs build tasks. Think of it as a dedicated server that runs your build process.

Imagine that you have an Azure Pipelines project that receives build requests many times per day. Or perhaps you have multiple projects that can each use the same type of build agent. You can organize build agents into agent pools to help ensure that there's a server ready to process each build request.

When a build is triggered, Azure Pipelines selects an available build agent from the pool. If all agents are busy, the process waits for one to become available.

When you use a Microsoft-hosted agent, you specify the VM image to use from the pool. Here's an example from your existing build configuration that uses an Ubuntu 18.04 build agent:

ymlCopy

pool:

vmImage: 'ubuntu-18.04'

demands:

- npm

When you use a Microsoft-hosted agent, you use vmImage to specify which type of system you need. Microsoft provides many types of VM images, including ones that run Windows, macOS, and various flavors of Linux.

The demands section specifies which software or capabilities you require the build machine to have.

When you use a build agent from your own pool, also known as a private pool, you specify the name of your pool. Here's an example.

ymlCopy

pool:

name: 'MyAgentPool'

demands:

- npm

When you don't require a demands section, you can shorten the syntax like this:

ymlCopy

pool: 'MyAgentPool'

You'll create a build agent and add it to a pool later in this module.

## What kind of agents can I use?

When you're choosing a build agent, there are two factors to consider:

* The operating system you want to build on
* Whether you can use a Microsoft-hosted agent or you need to provide your own agent

Azure Pipelines supports these operating systems:

* Windows
* macOS
* Linux (Ubuntu, Red Hat Enterprise Linux, and CentOS)

The build agent you choose depends mainly on what tools you use to build your code. For example, if you use Xcode to build your applications, you might choose a macOS agent. If you need Visual Studio, you'd likely choose a Windows agent.

Your existing build configuration uses a Microsoft-hosted agent. Hosted agents run on infrastructure that Microsoft provides for you.

A private agent uses infrastructure that you provide. Your agent can be a system that runs in the cloud or in your datacenter. Either system works, as long as the agent meets your requirements and can connect to Azure Pipelines. In this module, you use a VM that runs on Azure, which we provide.

## When should I use my own build agent?

For many build tasks, a Microsoft-hosted agent does everything you need. It's the easiest way to get started.

Microsoft takes care of all the security and other operating system updates for you. All you need to do is define the build configuration that you want to run.

Hosted agents also contain software for building many common types of applications. You can add any other software you need during the build process.

Microsoft-hosted agents have a few limitations, which include:

* **Build duration**: A build job can run for up to six hours.
* **Disk space**: Hosted agents provide a fixed amount of storage for your sources and your build outputs. This may not be enough storage.
* **CPU, memory, and network**: Hosted agents run on Microsoft Azure general purpose VMs. Standard\_DS2\_v2 describes the CPU, memory, and network characteristics you can expect.
* **Interactivity**: You can't sign in to a hosted agent.
* **File shares**: You can't drop build artifacts to Universal Naming Convention (UNC) file shares.

Although hosted agents are relatively easy to set up, there are some benefits to using your own build agents, aside from the limitations we just described.

For example, when you use hosted agents, you're sharing infrastructure with other Azure DevOps users. Although it ordinarily takes just seconds to start your build, it can take longer depending on the load on the Microsoft system.

Also, when you use hosted agents, you get a clean system with each build. When you bring your own build agent, you can decide whether to perform a clean build each time or perform an incremental build. With an incremental build, you build upon existing build tools and compiled code. An incremental build can take less time to complete, because the system already has many of the build tools and dependent components installed.

As a tradeoff, because the build infrastructure is yours, it's your responsibility to ensure that your build agents contain the latest software and security patches.

## How do you set up a private build agent?

A private build agent contains the software that's required to build your applications. It also contains agent software, which enables the system to connect to Azure Pipelines and receive build jobs.

When you set up a private agent, you provide the infrastructure that the builds use to run on. This gives you flexibility in how you bring up and maintain your agents.

For example, you can:

* **Set up the build agent manually**: You bring up the system, sign in, and interactively install your build tools and the agent software.
* **Automate the process**: You bring up the system and run a script or tool to install your build tools and the agent software. You can configure the agent after the system comes online or during the provisioning process.

For example, when you run build agents on Azure, you can use what's called an Azure Resource Manager template to bring up the system and configure it to act as a build agent, all in one step. Terraform  by HashiCorp is another way to automate the process. Terraform works with many types of infrastructure, including Azure.

* **Create an image**: You create an image, or snapshot, of a configured environment. You then use the image to create as many identical systems as you need in your pool.

Manual configuration is a good way to get started, because it enables you to understand the process. It's also the fastest way to get set up when you need just one build agent.

Automation is useful when you need many build agents or you need to bring up and tear down build infrastructure on a regular basis. You can move from a manual process to an automated process when you need multiple agents.

Images are a form of automation. They can help save time because all the software is pre-configured. As a tradeoff, you might need to periodically rebuild your images to incorporate the latest OS patches and build tools. Packer  by HashiCorp is a popular tool for creating images.

## What does the team decide?

Let's check back with the team.

**Tim:** I think I understand some of the various approaches. I'd be interested in creating a private build agent that can build the Space Game website so that I can demo it to the other teams. Would that be hard to set up?

**Mara:** I think we can do it. We could use a VM from our lab or, better yet, we can use a VM that runs on Azure as our build agent. I'd bet we can set up some scripts to run to set up the VM. When we're done experimenting, we can tear down the VM so that we're no longer paying for it.

**Tim:** That sounds good. I can learn a little bit about running VMs on Azure in the process.

**Andy:** Have fun! I can't wait to see what you discover.