**Describe pipelines and concurrency**

**Advanced**

**Administrator**

**Developer**

**DevOps Engineer**

**Security Engineer**

**Security Operations Analyst**

**Service Adoption Specialist**

**Solution Architect**

**Technology Manager**

**Azure**

**Azure Artifacts**

**Azure Boards**

**Azure Cloud Services**

**Azure DevOps**

**Azure Pipelines**

**Azure Repos**

**Azure Test Plans**

**GitHub**

**This module describes parallel jobs and how to estimate their usage. Also, it presents Azure Pipelines for open-source projects, explores Visual Designer and YAML pipelines.**

**Learning objectives**

**By the end of this module, you're able to:**

* **Use and estimate parallel jobs.**
* **Use Azure Pipelines for open-source or private projects.**
* **Use Visual Designer.**
* **Work with Azure Pipelines and YAML.**

[**Start**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/1-introduction/)**Add**

**Prerequisites**

**None**

**This module is part of these learning paths**

* [**AZ-400: Implement CI with Azure Pipelines and GitHub Actions**](https://learn.microsoft.com/training/paths/az-400-implement-ci-azure-pipelines-github-actions/)

**Module assessment**

**Assess your understanding of this module. Sign in and answer all questions correctly to earn a pass designation on your profile.**

[**Take the module assessment**](https://learn.microsoft.com/training/modules/describe-pipelines-concurrency/8-knowledge-check/)

* [**Introduction**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/1-introduction)**1 min**
* [**Understand parallel jobs**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/2-understand-parallel-jobs)**4 min**
* [**Estimate parallel jobs**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/3-estimate-parallel-jobs)**6 min**
* [**Describe Azure Pipelines and open-source projects**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/4-describe-azure-pipelines-open-source-projects)**6 min**
* [**Explore Azure Pipelines and Visual Designer**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/5-explore-azure-pipelines-visual-designer)**4 min**
* [**Describe Azure Pipelines and YAML**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/6-describe-azure-pipelines-yaml)**4 min**
* [**Enable Continuous Integration with Azure Pipelines**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/7-enable-continuous-integration-azure-pipelines)**45 min**
* [**Knowledge check**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/8-knowledge-check)**5 min**
* [**Summary**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/9-summary)**1 min**

**Introduction**

Completed100 XP

* 1 minute

This module describes parallel jobs and how to estimate their usage. Also, it presents Azure DevOps for open-source projects, explores Visual Designer and YAML pipelines.

**Learning objectives**

After completing this module, students and professionals can:

* Use and estimate parallel jobs.
* Use Azure Pipelines for open-source or private projects.
* Use Visual Designer.
* Work with Azure Pipelines and YAML.

**Prerequisites**

* Understanding of what DevOps is and its concepts.
* Familiarity with version control principles is helpful but is not necessary.
* Beneficial to have experience in an organization that delivers software.

**Understand parallel jobs**

Completed100 XP

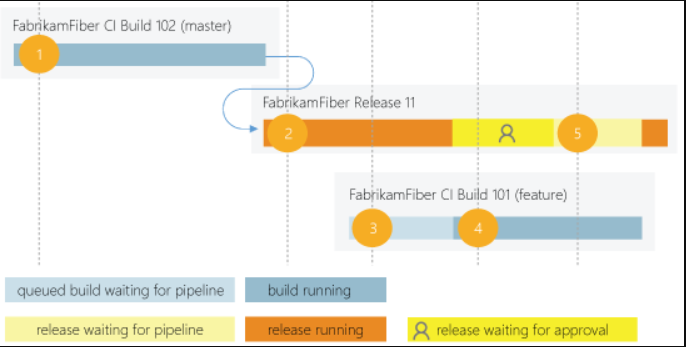
* 4 minutes

**How a parallel job is consumed by a build or release**

Consider an organization that has only one Microsoft-hosted parallel job.

This job allows users in that organization to collectively run only one build or release job at a time.

When more jobs are triggered, they're queued and will wait for the previous job to finish.



A release consumes a parallel job only when it's being actively deployed to a stage.

While the release is waiting for approval or manual intervention, it doesn't consume a parallel job.

**A simple example of parallel jobs**

* FabrikamFiber CI Build 102 (main branch) starts first.
* Deployment of FabrikamFiber Release 11 is triggered by the completion of FabrikamFiber CI Build 102.
* FabrikamFiber CI Build 101 (feature branch) is triggered. The build can't start yet because Release 11's deployment is active. So, the build stays queued.
* Release 11 waits for approvals. Fabrikam CI Build 101 starts because a release waiting for approvals doesn't consume a parallel job.
* Release 11 is approved. It resumes only after Fabrikam CI Build 101 is completed.

**Relationship between jobs and parallel jobs**

The term job can refer to multiple concepts, and its meaning depends on the context:

* When you define a build or release, you can define it as a collection of jobs. When a build or release runs, you can run multiple jobs as part of that build or release.
* Each job consumes a parallel job that runs on an agent. When there aren't enough parallel jobs available for your organization, then the jobs are queued up and run one after the other.

You don't consume any parallel jobs when you run a server job or deploy to a deployment group.

**Next unit: Estimate parallel jobs**

**Estimate parallel jobs**

Completed100 XP

* 6 minutes

**Determine how many parallel jobs you need**

You could begin by seeing if the free tier offered in your organization is enough for your teams.

When you've reached the 1,800 minutes per month limit for the free tier of Microsoft-hosted parallel jobs, you can start by buying one parallel job to remove this monthly time limit before deciding to purchase more.

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 extra parallel jobs as needed.

**Simple estimate**

A simple rule of thumb: Estimate that you'll need one parallel job for every four to five users in your organization.

**Detailed estimate**

In the following scenarios, you might need multiple parallel jobs:

* If you have multiple teams, and if each of them requires a CI build, you'll likely need a parallel job for each team.
* If your CI build trigger applies to multiple branches, you'll likely need a parallel job for each active branch.
* If you develop multiple applications by using one organization or server, you'll likely need more parallel jobs: one to deploy each application simultaneously.

**View available parallel jobs**

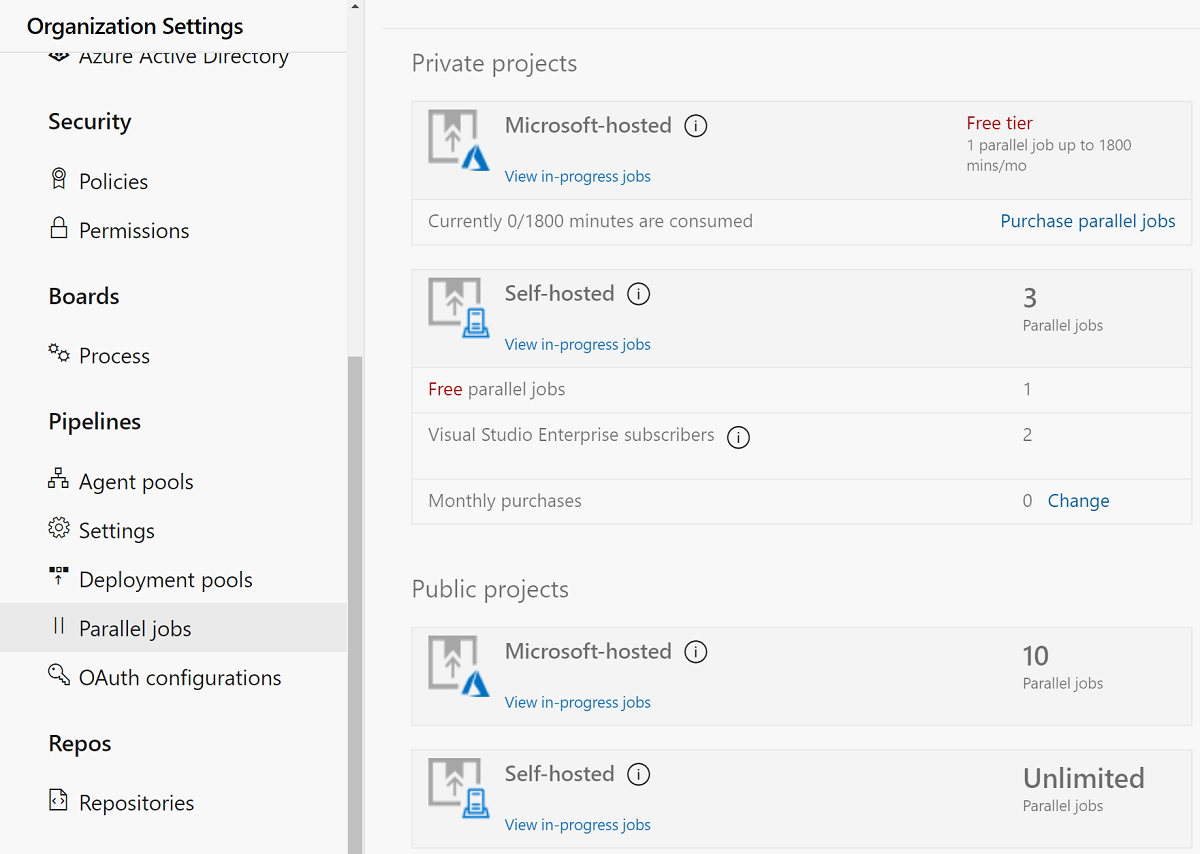
Browse to Organization settings > Pipelines > Parallel jobs.

**Location of parallel jobs in organization settings**

URL example: https://{your\_organization}/\_settings/buildqueue?\_a=concurrentJobs

View the maximum number of parallel jobs that are available in your organization.

Select View in-progress jobs to display all the builds and releases that are actively consuming an available parallel job or queued waiting for a parallel job to be available.



**Sharing of parallel jobs across projects in a collection**

Parallel jobs are purchased at the organization level, and they're shared by all projects in an organization.

Currently, there isn't a way to partition or dedicate parallel job capacity to a specific project or agent pool. For example:

* You purchase two parallel jobs in your organization.
* You queue two builds in the first project, and both the parallel jobs are consumed.
* You queue a build in the second project. That build won't start until one of the builds in your first project is completed.

**Next unit: Describe Azure Pipelines and open-source projects**

**Describe Azure Pipelines and open-source projects**

Completed100 XP

* 6 minutes

Azure DevOps offers developers a suite of DevOps capabilities, including Source control, Agile planning, Build, Release, Test, and more.

But to use Azure DevOps features requires the user to first sign in using a Microsoft or GitHub Account.

However, this blocks many engaging scenarios where you want to publicly share your code and artifacts or provide a wiki library or build status page for unauthenticated users.

With public projects, users can mark an Azure DevOps Team Project as public.

This will enable anonymous users to view the contents of that project in a read-only state enabling collaboration with anonymous (unauthenticated) users that wasn't possible before.

Anonymous users will essentially see the same views as authenticated users, with non-public functionality such as settings or actions (such as queue build) hidden or disabled.

**Public versus private projects**

Projects in Azure DevOps provide a repository for source code and a place for a group of developers and teams to plan, track progress, and collaborate on building software solutions.

One or more projects can be defined within an organization in Azure DevOps.

Users that aren't signed into the service have read-only access to public projects on Azure DevOps.

Private projects require users to be granted access to the project and signed in to access the services.

**Supported services**

Non-members of a public project will have read-only access to a limited set of services, precisely:

* Browse the code base, download code, view commits, branches, and pull requests.
* View and filter work items.
* View a project page or dashboard.
* View the project Wiki.
* Do a semantic search of the code or work items.

For more information, see [Differences and limitations for non-members of a public project](https://learn.microsoft.com/en-us/azure/devops/organizations/public/feature-differences).

**A practical example: .NET Core CLI**

Supporting open-source development is one of the most compelling scenarios for public projects. A good example is the .NET Core CLI.

Their source is hosted on GitHub, and they use Azure DevOps for their CI builds.

However, if you click on the build badges in their readme, you'll not see the build results unless you were one of the project's maintainers.

Since this is an open-source project, everybody should view the full results to see why a build failed and maybe even send a pull request to help fix it.

Thanks to public projects capabilities, the team will enable just that experience. Everyone in the community will have access to the same build results, whether they are a maintainer on the project.

**How do I qualify for the free tier of Azure Pipelines for public projects?**

Microsoft will automatically apply the free tier limits for public projects if you meet both conditions:

* Your pipeline is part of an Azure Pipelines public project.
* Your pipeline builds a public repository from GitHub or the same public project in your Azure DevOps organization.

**Are there limits on who can use Azure Pipelines?**

You can have as many users as you want when you're using Azure Pipelines. There's no per-user charge for using Azure Pipelines.

Users with both basic and stakeholder access can author as many builds and releases as they want.

**Are there any limits on the number of builds and release pipelines that I can create?**

No. You can create hundreds or even thousands of definitions for no charge. You can register any number of self-hosted agents for no cost.

**As a Visual Studio Enterprise subscriber, do I get more parallel jobs for Azure Pipelines?**

Yes. Visual Studio Enterprise subscribers get one self-hosted parallel job in each Azure DevOps Services organization where they're a member.

**When you're using the per-minute plan, you can run only one job at a time.**

If you run builds for more than 14 paid hours in a month, the per-minute plan might be less cost-effective than the parallel jobs model.

See [Azure DevOps Services Pricing | Microsoft Azure](https://azure.microsoft.com/pricing/details/devops/azure-devops-services/) for current pricing.

**Next unit: Explore Azure Pipelines and Visual Designer**

**Explore Azure Pipelines and Visual Designer**

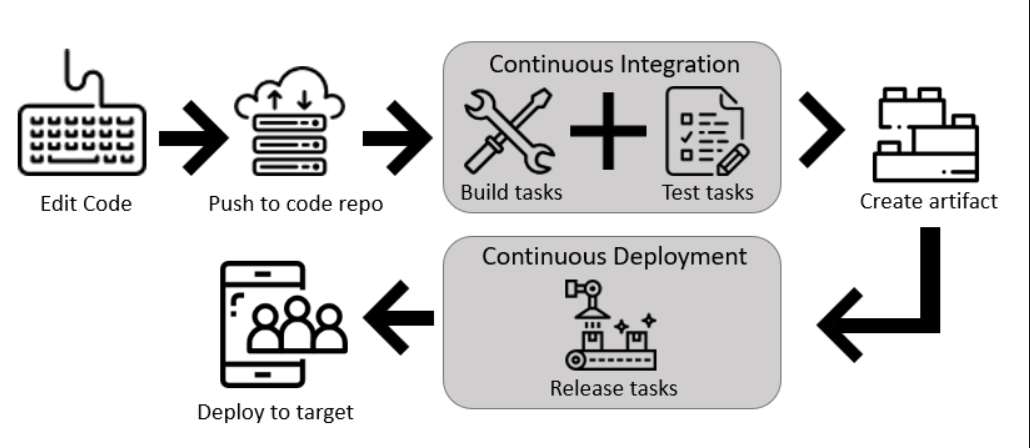
Completed100 XP

* 4 minutes

You can create and configure your build and release pipelines in the Azure DevOps web portal with the visual designer. (Often referred to as "Classic Pipelines").

Configure Azure Pipelines to use your Git repo.

1. Use the Azure Pipelines visual designer to create and configure your build and release pipelines.
2. Push your code to your version control repository. This action triggers your pipeline and runs tasks such as building or testing code.
3. The build creates an artifact used by the rest of your pipeline to run tasks such as deploying to staging or production.
4. Your code is now updated, built, tested, and packaged. It can be deployed to any target.



**Benefits of using the Visual Designer**

The visual designer is great for new users in continuous integration (CI) and continuous delivery (CD).

* The visual representation of the pipelines makes it easier to get started.
* The visual designer is in the same hub as the build results. This location makes it easier to switch back and forth and make changes.

If you think the designer workflow is best for you, create your first pipeline using the [visual designer](https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started-designer).

**Next unit: Describe Azure Pipelines and YAML**

[**Previous**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/4-describe-azure-pipelines-open-source-projects/)

**Describe Azure Pipelines and YAML**

Completed100 XP

* 4 minutes

Mirroring the rise of interest in infrastructure as code, there has been considerable interest in defining pipelines as code. However, pipeline as code doesn't mean executing a script that's stored in source control.

Codified pipelines use their programming model to simplify the setup and maximize reuse.

A typical microservice architecture will require many deployment pipelines that are identical. It's tedious to craft these pipelines via a user interface or SDK.

The ability to define the pipeline and the code helps apply all principles of code sharing, reuse, templatization, and code reviews. Azure DevOps offers you both experiences. You can either use YAML to define your pipelines or use the visual designer to do the same. You will, however, find that more product-level investments are being made to enhance the YAML pipeline experience.

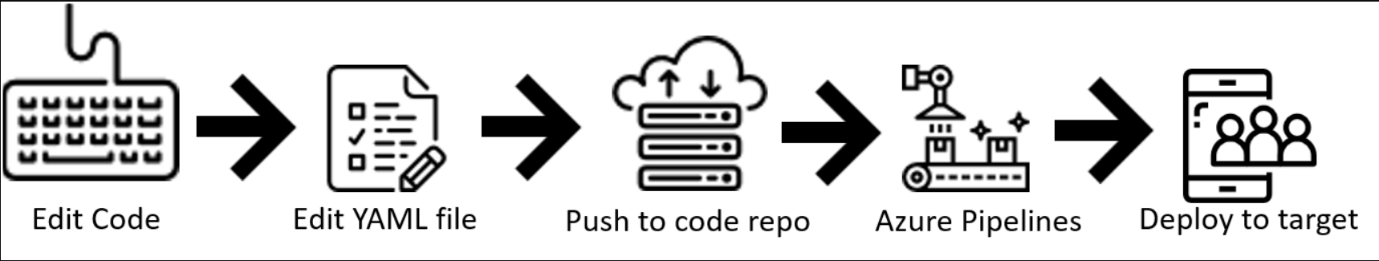
When you use YAML, you define your pipeline mostly in code (a YAML file) alongside the rest of the code for your app. When using the visual designer, you define a build pipeline to build and test your code and publish artifacts.

You also specify a release pipeline to consume and deploy those artifacts to deployment targets.

**Use Azure Pipelines with YAML**

You can configure your pipelines in a YAML file that exists alongside your code.

1. Configure Azure Pipelines to use your Git repo.
2. Edit your azure-pipelines.yml file to define your build.
3. Push your code to your version control repository. This action kicks off the default trigger to build and deploy and then monitor the results.
4. Your code is now updated, built, tested, and packaged. It can be deployed to any target.



**Benefits of using YAML**

* The pipeline is versioned with your code and follows the same branching structure. You get validation of your changes through code reviews in pull requests and branch build policies.
* Every branch you use can modify the build policy by adjusting the azure-pipelines.yml file.
* A change to the build process might cause a break or result in an unexpected outcome. Because the change is in version control with the rest of your codebase, you can more easily identify the issue.

If you think the YAML workflow is best for you, create your first pipeline by using [YAML](https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started-yaml).

While there's a slightly higher learning curve and a higher degree of code orientation when defining pipelines with YAML, it's now the preferred method.

**Next unit: Enable Continuous Integration with Azure Pipelines**

**Enable Continuous Integration with Azure Pipelines**

Completed100 XP

* 45 minutes

**Estimated time:** 45 minutes.

**Scenario**

In this lab, you will learn how to define build pipelines in Azure DevOps using YAML. The pipelines will be used in two scenarios:

* As part of Pull Request validation process.
* As part of the Continuous Integration implementation.

**Objectives**

After completing this lab, you'll be able to:

* Include build validation as part of a Pull Request.
* Configure CI pipeline as code with YAML.

**Requirements**

* This lab requires **Microsoft Edge** or an [Azure DevOps-supported browser](https://learn.microsoft.com/en-us/azure/devops/server/compatibility).
* **Set up an Azure DevOps organization:** If you don't already have an Azure DevOps organization that you can use for this lab, create one by following the instructions available at [Create an organization or project collection](https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/create-organization).

**Exercises**

During this lab, you'll complete the following exercises:

* Exercise 0: Configure the lab prerequisites.
* Exercise 1: Include build validation as part of a Pull Request.
* Exercise 2: Configure CI Pipeline as Code with YAML.

[Screenshot of a launch button which will take you to the lab.](https://go.microsoft.com/fwlink/?linkid=2270037)

**Next unit: Knowledge check**

**Knowledge check**

Completed200 XP

* **Module assessment**
* 5 minutes

 Answer 100% of questions correctly in order to pass. [**Retake**](https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/8-knowledge-check)

Dismiss alert

Choose the best response for each question.

**Check your knowledge**

Top of Form

**1.**

**Which of the following choices isn't used to create and configure an Azure Pipeline?**

XML.

**Correct. Azure Pipelines support YAML and Visual Designer.**

YAML.

**Incorrect. Azure Pipelines support YAML and Visual Designer.**

Visual Designer.

**2.**

**Which of the following choices is a benefit of using the Visual Designer?**

Every branch you use can modify the build policy by modifying the azure-pipelines.yml file.

The visual designer is in the same hub as the build results.

**Correct. The visual designer is in the same hub as the build results. This location makes it easier to switch back and forth and make changes.**

A change to the build process might cause a break or result in an unexpected outcome. Because the change is in version control with the rest of your codebase, you can more easily identify the issue.

**Incorrect. The visual designer is in the same hub as the build results. This location makes it easier to switch back and forth and make changes.**

**3.**

**Which of the following choices is a benefit of using YAML?**

The pipeline is versioned with your code and follows the same branching structure.

**Correct. The pipeline is versioned with your code and follows the same branching structure. You get validation of your changes through code reviews in pull requests and branch build policies.**

The YAML representation of the pipelines makes it easier to get started.

**Incorrect. The pipeline is versioned with your code and follows the same branching structure. You get validation of your changes through code reviews in pull requests and branch build policies.**

The YAML is in the same hub as the build results.

Bottom of Form

**Next unit: Summary**

**Summary**

Completed100 XP

* 1 minute

This module described parallel jobs and how to estimate their usage. Also, it presented Azure DevOps for open-source projects, explored Visual Designer and YAML pipelines.

You learned how to describe the benefits and usage of:

* Use and estimate parallel jobs.
* Use Azure Pipelines for open-source or private projects.
* Use Visual Designer.
* Work with Azure Pipelines and YAML.

“” https://learn.microsoft.com/en-us/training/modules/describe-pipelines-concurrency/9-summary#completion”