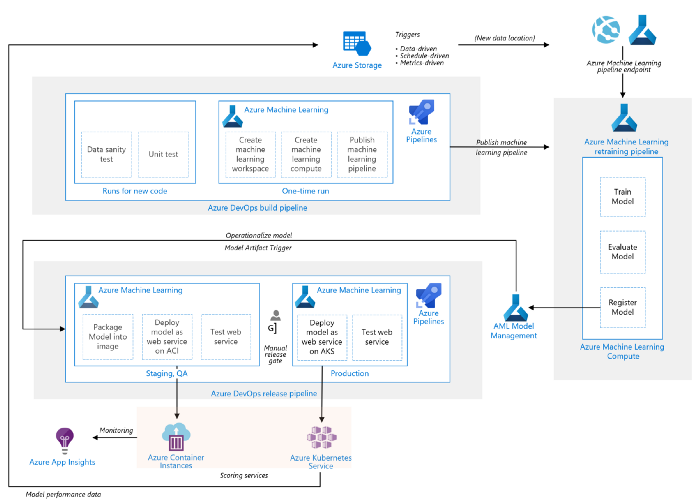
**What is a CICD pipeline?**

* In software engineering, CI/CD or CICD generally refers to the combined practices of continuous integration and either continuous delivery or continuous deployment.

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.

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.

* CI/CD bridges the gaps between development and operation activities and teams by enforcing automation in building, testing, and deployment of applications.
* A CI/CD pipeline for Machine Learning model lifecycle can be like below:

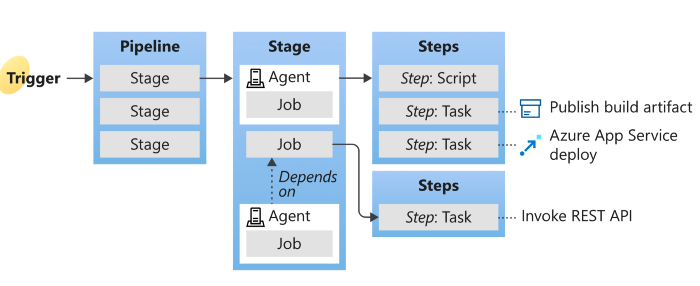


[Click here to view the source](https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/ai/mlops-python)

**Feature set**

A CICD pipeline can consist of multiple components. To implement our CICD pipeline, we used Azure Pipelines as part of [Azure DevOps Services](https://azure.microsoft.com/en-us/services/devops/).

* A pipeline defines the continuous integration and deployment process for your app. It can be thought of as a workflow that defines how your test, build, and deployment steps are run.
* [Stages](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/stages?view=azure-devops&tabs=yaml), [jobs,](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/phases?view=azure-devops&tabs=yaml)and [steps](https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema%2Cparameter-schema#steps)are the main building blocks of a Pipeline. A stage can consist of one or more jobs, and a job can consist of one or more steps. A step is the smallest building block of a pipeline to perform an action, that can be a [task](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/tasks?view=azure-devops&tabs=yaml)or script.
* When a pipeline gets executed is controlled by the [Trigger](https://docs.microsoft.com/en-us/azure/devops/pipelines/build/triggers?view=azure-devops). A pipeline can be configured to run upon a push to a repository, at scheduled times, or upon the completion of another build. All of these actions are known as triggers.
* To start the CICD pipeline, at least one agent is required. An [agent](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser)is a computing infrastructure with installed agent software that runs one job at a time. Azure DevOps provides Microsoft-hosted agents that waives the setup and maintenance effort for users, self-hosted agent is also an option when Microsoft-hosted agents do not meet requirements.



[Click here to view the source](https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/key-pipelines-concepts?view=azure-devops#agent)

Now we know the key concepts of a CICD pipeline and how it works on Azure DevOps, we can start building the first CICD pipeline on Azure DevOps.

**Azure CICD Pipeline Tutorial**

There are multiple ways of setting up the CICD pipeline on Azure DevOps. In this introduction, we will start from a simple one via YAML definition.

Azure Pipelines doesn’t support all YAML features. Unsupported features include anchors, complex keys, and sets. Also, unlike standard YAML, [Azure Pipelines](https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema%2Cparameter-schema) depends on seeing stage, job, task, or a task shortcut like script as the first key in a mapping.

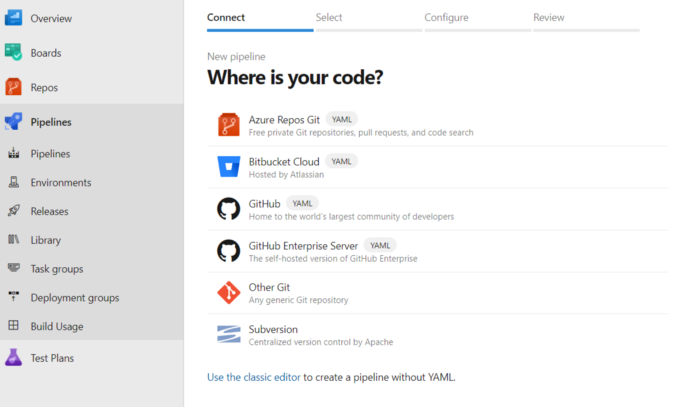
**Basic Requirements**

* An Azure DevOps account
* Git
* YAML

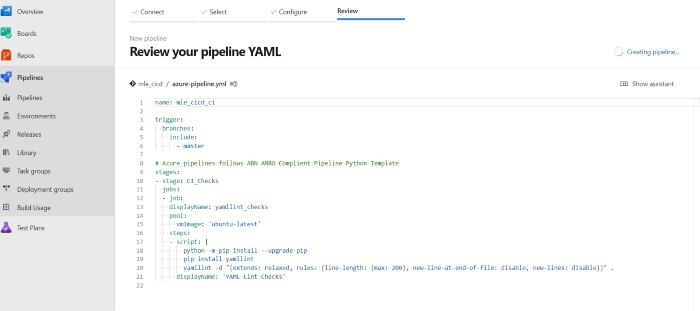
**Initial Setup**

* Create an empty repo named TemplateRepo
* Inside the repo, create an empty YAML file named *azure-pipeline.yml*and copy-paste the following lines into that file.
* Commit this file to the remote repo on master branch.
* **name**: the pipeline name in a string as cicd\_ci
* **trigger**: branch name master will trigger a build
* **stages**: there is one stage named as CI\_Checks , within this stage, there is one job with a friendly given name yamllint\_checks . The job runs by the Microsoft-hosted agent uses VM image ubuntu-latest
* **script**: a shortcut for [command-line task](https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/utility/command-line?view=azure-devops&tabs=yaml), the script consists of 3 commands as listed, and the command will be running in order. The command can be replaced with other working commands.

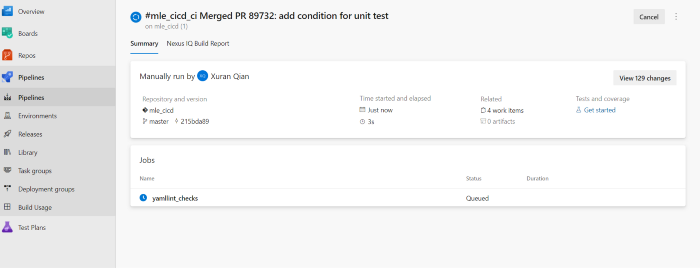
Now, we can create a new pipeline using this YAML definition. Go to your DevOps, on the left panel, Pipelines -> Pipelines -> New pipeline (a button on top right corner), then you will enter the page below. Depends on where your YAML file is located, you can choose where to import it:



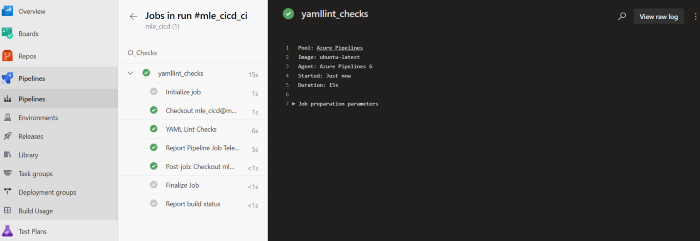
In our case, we pick Azure Repos Git and then select the repository name and YAML definition.



Once the import is done, you can run your first pipeline. The pipeline run page provides a summary that contains information relevant to this pipeline run. Because we set the pipeline trigger as branch as `master`, which means the changes to the master branch will automatically start the pipeline run.



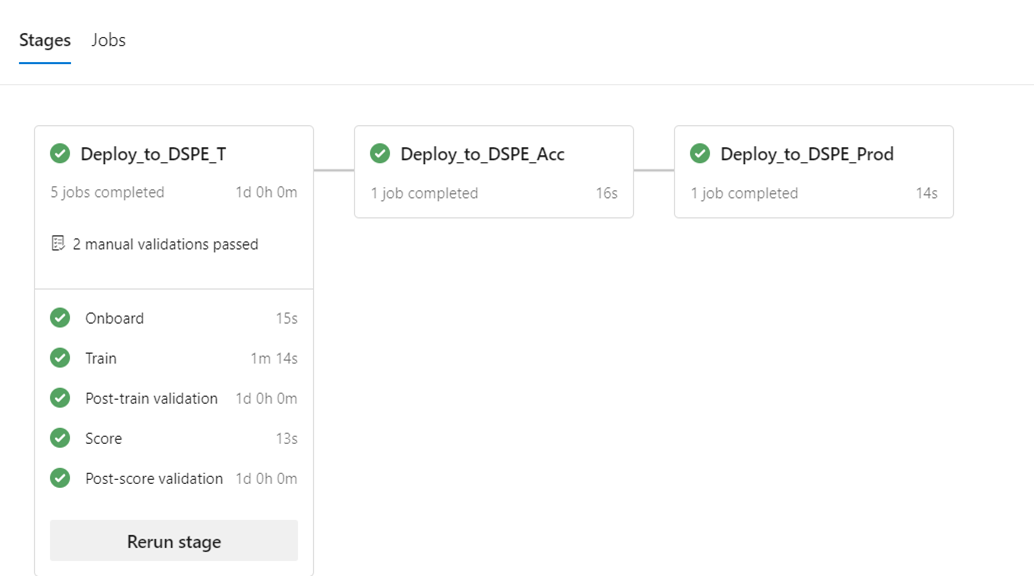
The running status of our defined stage CI\_Checksand job yamllint\_checks are showed here. You may notice that except our defined step YAML Lint Checks , there are multiple steps we did not specify, for example, Checkout xxx these are initialized and tear-down steps defined by default.



**Complex pipeline**

We will briefly cover the complex pipeline and execution orders as well.

So far we defined a simple pipeline, with one stage, one job, and one step. A pipeline, in reality, can consist of multiple stages, jobs, and steps. For example, deployment to different environments can be defined as multiple stages, within each stage multiple jobs are defined, demo as the following structure:



Such a pipeline definition will have a linear series of stages, jobs, and steps, in YAML like:

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

To let such a complex pipeline run to our desire, the understanding of the [**order of stages/jobs/steps**](https://eur03.safelinks.protection.outlook.com/?url=https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Fazure%2Fdevops%2Fpipelines%2Fprocess%2Fruns%3Fview%3Dazure-devops&data=04%7C01%7C%7C3217968323a348e75fe608d99d3acd3b%7C3a15904d3fd94256a753beb05cdf0c6d%7C0%7C0%7C637713698135392510%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=utGs0uXja3VQJD9a4TNSTrUde8JAkRZOOnPOH5CgI4s%3D&reserved=0)**is critical.**This link includes the order of a detailed step that how Azure Pipelines go through the steps, we won’t copy-paste the same text here. To summarize:

* Azure Pipelines parse the pipeline definition and hand off jobs to agents and collect the results.
* Agents are workers that prepare a run environment for the job, execute each step in the job, and report results to Azure pipelines.
* **Steps are run sequentially, one after another.** Before a step can start, all the previous steps must be finished (or skipped). Each step runs in its own process, isolating it from the environment left by previous steps.
* **When there are multiple jobs in a single stage, jobs can run parallelly, by using multiple agents.**Whenever Azure Pipelines needs to run a job, it will ask the agent pool for an agent, and each agent can only run one job at a time. To run multiple jobs in parallel, the configuration of multiple agents for the agent pool is required. Thoughtful parallelization can be beneficial to reduce deployment time, as shown in this [example](https://eur03.safelinks.protection.outlook.com/?url=https%3A%2F%2Fsamlearnsazure.blog%2F2020%2F02%2F11%2Fparallel-azure-pipelines-jobs%2F&data=04%7C01%7C%7C3217968323a348e75fe608d99d3acd3b%7C3a15904d3fd94256a753beb05cdf0c6d%7C0%7C0%7C637713698135392510%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=bu%2BhCIjN%2F2bALkMIz%2FNP3SLvp5gkrosjFL7oXzJXgaY%3D&reserved=0).
* Stages are logical boundaries in the pipeline where the pipeline can be paused and various checks can be performed. Every pipeline has at least one stage even if you do not explicitly define it. **Stages by default run sequentially in the order in which they are defined in the YAML file when no dependencies are specified.**
* **With dependencies, stages and jobs can run in the order of the dependsOn requirements.** By default, a job or stage runs if it does not depend on any other job or stage, or if all of the jobs or stages that it depends on have completed and succeeded.

**Wrap Up**

* We did a quick tutorial about Azure DevOps CICD Pipeline, how it is used in the Machine Learning lifecycle and how it can help us automate workflow.
* Through this tutorial, we were able to build, set up, and run a simple CI pipeline using YAML definition and a Microsoft-hosted agent on Azure DevOps.
* We set up a branch trigger, to allow the linting script to run every time when a branch is updated automatically.