Contents

[1 Azure Pipelines 2](#_Toc48229274)

[2 Self-hosted agents in Azure Pipelines 2](#_Toc48229275)

[3 Provision infrastructure in Azure Pipelines 3](#_Toc48229276)

[3.1 Infrastructure as Code 3](#_Toc48229277)

[3.2 Environment drift and Snowflake 3](#_Toc48229278)

[3.3 Idempotence 4](#_Toc48229279)

[3.4 Infrastructure as Code tools 4](#_Toc48229280)

[3.5 Terraform with Azure 4](#_Toc48229281)

[4 Configure infrastructure in Azure Pipelines 5](#_Toc48229282)

[4.1 Configuration as code 5](#_Toc48229283)

[4.2 Configuration management 5](#_Toc48229284)

[4.3 Configuration drift and Snowflake 5](#_Toc48229285)

[4.4 Configuration as code and Infrastructure as Code 5](#_Toc48229286)

[4.5 Configuration as code tools 5](#_Toc48229287)

[4.6 Ansible with Azure 5](#_Toc48229288)

[4.7 Ansible playbooks 6](#_Toc48229289)

[5 Azure DevOps vs. Jenkins 6](#_Toc48229290)

[5.1 Pros 6](#_Toc48229291)

[5.2 Cons 6](#_Toc48229292)

# Azure Pipelines [[1]](#footnote-1)

Azure Pipelines is a cloud service to automatically build and test code project and make it available to other users. Azure Pipelines supports continuous integration (CI) and continuous delivery (CD) to constantly and consistently test and build code and ship it to any target, accomplished by defining a pipeline using the YAML syntax or through the user interface (Classic). To summarize, Azure Pipelines:

• Works with any language, application type, or platform.

• Deploys to different types of targets viz., container registries, virtual machines, Azure services etc.

• Builds on Windows, Linux, or Mac machines.

• Integrates with GitHub, GitHub Enterprise, Azure Repos Git & TFVC, Bitbucket Cloud, and Subversion.

• Works with package formats viz., NuGet, npm, Maven etc.

# Self-hosted agents in Azure Pipelines [[2]](#footnote-2)

If you use a self-hosted agent, you can run incremental builds. For example, if you define a pipeline that does not clean the repo and does not perform a clean build, your builds will typically run faster. When you use a Microsoft-hosted agent, you don't get these benefits because the agent is destroyed after the build or release pipeline is completed. Microsoft-hosted agents have a few limitations:

• Build duration: A build job can run for up to six hours.

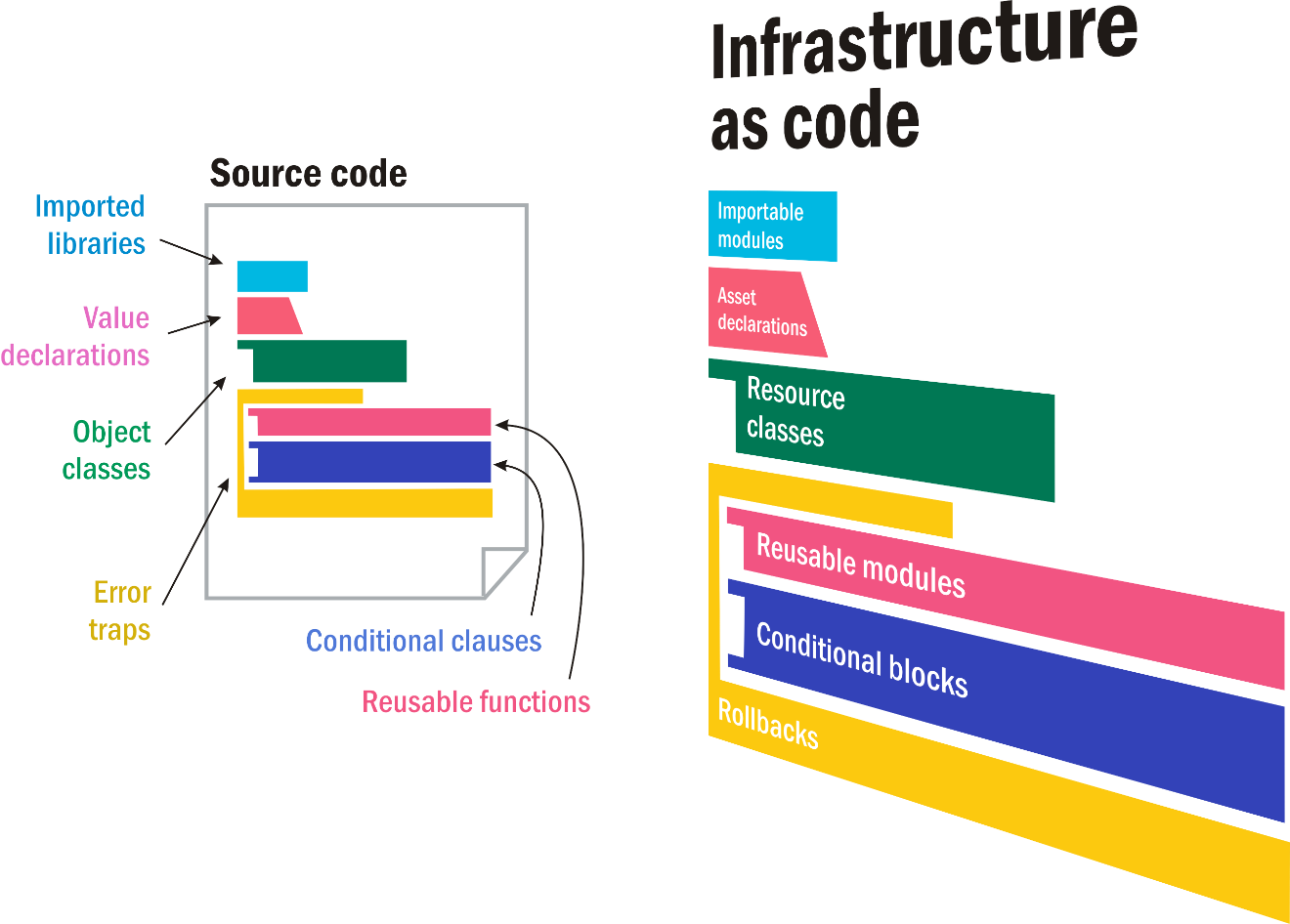
• Disk space: Hosted agents provide a fixed amount of storage for your sources and your build outputs. This may not be enough storage.

• 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.

# Provision infrastructure in Azure Pipelines [[3]](#footnote-3)



## Infrastructure as Code

Infrastructure as Code (IaC) is the management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model, using the same versioning as DevOps team uses for source code. Like the principle that the same source code generates the same binary, an IaC model generates the same environment every time it is applied. IaC is a key DevOps practice and is used in conjunction with Continuous Delivery (CD).

## Environment drift and Snowflake

Infrastructure as Code evolved to solve the problem of environment drift in the release pipeline. Over time, each environment becomes a snowflake, that is, a unique configuration that cannot be reproduced automatically. With snowflakes, administration and maintenance of infrastructure involves manual processes which were hard to track and contributed to errors.

## Idempotence

An idempotent operation is one that provides the same result each time you apply it. Idempotence is a principle of Infrastructure as Code. Idempotence is the property that a deployment command always sets the target environment into the same configuration, regardless of the environment’s starting state. Idempotency is achieved by either automatically configuring an existing target or by discarding the existing target and recreating a fresh environment.

## Infrastructure as Code tools

**• Azure CLI** is used to connect to Azure and run administrative commands on Azure resources.

**• Azure PowerShell** is a module that you add to Windows PowerShell or PowerShell Core.

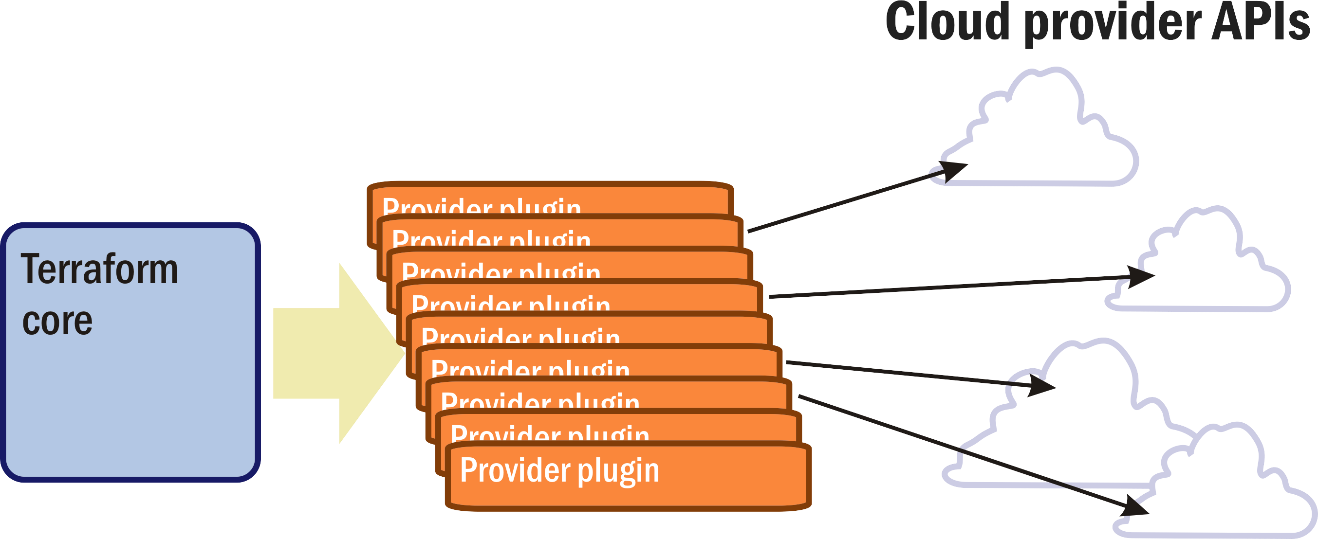
**• Azure Resource Manager** is the interface for managing and organizing cloud resources.

**• Ansible** by Red Hat is an open-source product that automates cloud provisioning, configuration management, and application deployments.

**• Terraform** by HashiCorp is an open-source tool that provisions and manages cloud infrastructure.

**• Azure SDK** enables you to access and manage Azure resources from application code by using any programming language.

## Terraform with Azure



Hashicorp Terraform is an open-source tool for provisioning and managing cloud infrastructure. It codifies infrastructure in configuration files that describe the topology of cloud resources. These resources include virtual machines, storage accounts, and networking interfaces. The Terraform CLI provides a simple mechanism to deploy and version the configuration files to Azure. Terraform's template-based configuration files enable you to define, provision, and configure Azure resources in a repeatable and predictable manner.

Terraform can help bridge the gap between first-generation virtualization and containerization. A containerized infrastructure is different from one built around virtual machines since the application is virtualized, not the operating system and services that support it. Typically, a Configuration management installation deals with one type of environment or the other. Terraform can manage an infrastructure in which both components play active roles.

# Configure infrastructure in Azure Pipelines [[4]](#footnote-4)

## Configuration as code

Configuration as code enables you to describe the configuration you need to run your application or service. An easier way to remove the possibility of human error when initializing machines is to configure them automatically through code. By treating configuration as code, you can configure many systems identically from a single consistent definition. Also, you can add automatic configuration tasks to their CI/CD pipeline.

## Configuration management

Configuration management is a term that's associated with configuration as code. Configuration management refers to the automated management of configuration, typically in the form of automated scripts or programs that you run regularly on your deployments.

## Configuration drift and Snowflake

Configuration drift happens when resources change over time from their original deployment state manually or through automated processes, eventually becoming a snowflake. With snowflakes, infrastructure administration and maintenance typically become manual processes, which can be hard to track and are prone to human error. The more an environment drifts from its original state, the more likely it is for an application to encounter issues.

## Configuration as code and Infrastructure as Code

The concept of configuration as code is similar to the concept of infrastructure as code. In fact, you can combine the two models to automatically provision and then configure your systems all in one step. For example, your CI/CD pipeline might use an Azure Resource Manager template, Terraform etc. to bring up the infrastructure. Then, you might run Chef, Puppet, Ansible etc. to set up the infrastructure with everything the application or service needs to run.

## Configuration as code tools

**• Ansible** by Red Hat is an open-source product that automates cloud provisioning, configuration management, and application deployments by using Ansible commands directly, or by writing Ansible playbooks to deploy more complex configurations

**• Azure Automation** is a service in Azure that helps you automate manual tasks.

**• Azure Custom Script Extension** is a way to download and run scripts on your Azure VMs.

**• Chef** is an infrastructure automation tool that enables you to configure and manage your systems.

**• Cloud-init** by Canonical is a way to customize a Linux VM as it boots for the first time.

**• PowerShell Desired State Configuration (DSC)** is a management platform that defines the configuration of target machines.

**• Puppet** is an automation platform that handles the application delivery and deployment process.

## Ansible with Azure

Ansible by Red Hat is an open-source product that automates cloud provisioning, configuration management, and application deployments. Using Ansible you can provision virtual machines, containers, and network and complete cloud infrastructures.

## Ansible playbooks

Ansible playbooks allow you to direct Ansible to configure your environment. Playbooks are coded using YAML so as to be human-readable. At a basic level, playbooks can be used to manage configurations of and deployments to remote machines. At a more advanced level, they can sequence multi-tier rollouts involving rolling updates, and can delegate actions to other hosts, interacting with monitoring servers and load balancers along the way.

# Azure DevOps vs. Jenkins

## Pros

|  |  |
| --- | --- |
| Azure DevOps [[5]](#footnote-5) | Jenkins [[6]](#footnote-6) |
| • Source control integration.  • Azure Boards integration.  • Templates for multiple Agile types.  • Agile Management.  • Agile Metrics.  • Customized Reporting.  • The backlogs and Kanban boards for planning and tracking work are second to none.  • Support for Scrum artifacts and processes.  • Unified environment for all DevOps tasks and procedures.  • YAML backend is a plus.  • CI/CD built-in, customizable.  • Navigation User Interface.  • Flexible Requirements Hierarchy Management.  • Excel Integration. | • Helm chart makes it trivial to deploy Jenkins in a "GitOps" friendly way.  • Overall simple architecture to deploy (when compared to "modern" CI/CD systems).  • Easily extract metrics from build system into a platform like Prometheus.  • Jenkins "Configuration as Code" plugin supports nearly everything needed to run Jenkins itself in a "GitOps" compliant manner.  • Jenkins can be used much like the modern CI/CD forms that use container steps rather than complicated groovy scripts.  • Jenkins supports unlimited number of credential backends.  • Jenkins Shared Libraries are one of the most polarizing features. |

## Cons

|  |  |
| --- | --- |
| Azure DevOps [5](#FN5) | Jenkins [6](#FN6) |
| • Streamlined permissions structure.  • Azure DevOps is not as customizable as JIRA.  • Security and permissions are managed in different places instead of one central location.  • Central Dashboard of Development Metrics.  • Azure DevOps is great for simple queries, but complex queries and the display of results doesn't always produce intuitive results.  • Azure DevOps allows for the tracking of Epics and Initiatives, but there doesn't seem to be a structured interface for tracking product investment themes. | • There's no (maintained) YAML interface for Jenkins Pipelines.  • The Kubernetes Plugin works well but complicates Docker Builds.  • The PodTemplate approach to define before the Jenkins slave starts the containers is problematic in a modern CI/CD environment.  • It's not practical today to run "multi-master" Jenkins for High Availability without using Jenkins Enterprise.  • Docker layer caching is non-trivial if running with Ephemeral Jenkins Slaves under kubernetes. |

1. <https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops> [↑](#footnote-ref-1)
2. <https://docs.microsoft.com/en-us/learn/modules/host-build-agent/2-choose-a-build-agent> [↑](#footnote-ref-2)
3. <https://docs.microsoft.com/en-us/learn/modules/provision-infrastructure-azure-pipelines/> [↑](#footnote-ref-3)
4. <https://docs.microsoft.com/en-us/learn/modules/configure-infrastructure-azure-pipelines/> [↑](#footnote-ref-4)
5. <https://www.trustradius.com/products/azure-devops/reviews> [↑](#footnote-ref-5)
6. <https://cloudposse.com/devops/jenkins-pros-cons-2020> [↑](#footnote-ref-6)