Contents

[1 Azure and Jenkins 2](#_Toc46342892)

[1.1 There are two ways to integrate Jenkins with Azure Pipelines: 2](#_Toc46342893)

[1.2 Host Jenkins servers in Azure 2](#_Toc46342894)

[1.3 Scale build automation on demand 2](#_Toc46342895)

[1.4 Deploy code into Azure services 2](#_Toc46342896)

[2 Azure Pipelines vs. Jenkins 3](#_Toc46342897)

[2.1 Configuration 3](#_Toc46342898)

[2.2 Visual Configuration 3](#_Toc46342899)

[2.3 Container-Based Builds 3](#_Toc46342900)

[2.4 Agent Selection 3](#_Toc46342901)

[2.5 Environment Variables 3](#_Toc46342902)

[2.6 Predefined Variables 3](#_Toc46342903)

[2.7 Success and Failure Handling 3](#_Toc46342904)

[3 Jenkins plug-ins for Azure 4](#_Toc46342905)

[4 Case Study: Run a Jenkins Server on Azure 5](#_Toc46342906)

[4.1 Architecture 5](#_Toc46342907)

[4.2 Recommendations 6](#_Toc46342908)

# Azure and Jenkins

Jenkins is an open-source automation server used to set up continuous integration and delivery (CI/CD) for software projects. You can host Jenkins deployment in Azure or extend existing Jenkins configuration using Azure resources. Jenkins plugins are also available to simplify CI/CD of applications to Azure.

## There are two ways to integrate Jenkins with Azure Pipelines:

• One way is to run CI jobs in Jenkins separately. This involves configuration of a CI pipeline in Jenkins and a web hook in Azure DevOps that invokes the CI process when source code is pushed to a repository or a branch.

• The alternate way is to wrap a Jenkins CI job inside an Azure pipeline. In this approach, a build definition will be configured in Azure Pipelines to use the Jenkins tasks to invoke a CI job in Jenkins, download, and publish the artifacts produced by Jenkins.

An Azure CD pipeline can be configured to pick these build artifacts, irrespective of the approach, for deployment. While there are pros and cons with both the approaches, the latter approach has multiple benefits:

1. End-to-end traceability from work item to source code to build and release

2. Triggering of a Continuous Deployment (CD) when the build is completed successfully

3. Execution of the build as part of the branching strategy

## Host Jenkins servers in Azure

Host Jenkins in Azure to centralize build automation and scale deployment as the needs of software projects grow. You can deploy Jenkins in Azure using:

• Jenkins solution template in Azure Marketplace

• Azure virtual machines

• On a Kubernetes cluster running in Azure Container Service

## Scale build automation on demand

Add build agents to existing Jenkins deployment to scale Jenkins build capacity as the number of builds and complexity of jobs and pipelines increase. You can run these build agents on Azure virtual machines by using the Azure VM Agents plug-in. Once configured with an Azure service principal, Jenkins jobs and pipelines can use this credential to:

• Securely store and archive build artifacts in Azure Storage using the Azure Storage plug-in

• Manage and configure Azure resources with the Azure CLI

## Deploy code into Azure services

Use Jenkins plugins to deploy applications to Azure as part of Jenkins CI/CD pipelines. Deploying into Azure App Service and Azure Container Service lets you stage, test, and release updates to applications without managing the underlying infrastructure. Plug-ins are available to deploy to the following services and environments:

• Azure App Service on Linux

• Azure App Service

# Azure Pipelines vs. Jenkins

Azure Pipelines is a cloud native continuous integration pipeline, providing the management of build and release pipelines and build agent virtual machines hosted in the cloud. Jenkins has traditionally been installed by enterprises in their own data centers and managed in an on-premises fashion, though a number of providers offer managed Jenkins hosting. However, Azure Pipelines offers a fully on-premises option as well with Azure DevOps Server, for those customers who have compliance or security concerns that require them to keep their code and build within the enterprise data center. In addition, Azure Pipelines supports a hybrid cloud and on-premises model, where Azure Pipelines manages the build and release orchestration and enabling build agents both in the cloud and installed on-premises, for customers with custom needs and dependencies for some build agents but who are looking to move most workloads to the cloud.

## Configuration

• Azure Pipelines uses the industry-standard YAML to configure the build pipeline.

• Jenkinsfiles use a domain specific language based on the Groovy programming language.

## Visual Configuration

• Azure Pipelines uses the classic editor.

• Jenkins uses the graphical interface.

## Container-Based Builds

• Both Azure Pipelines and Jenkins support container-based builds.

## Agent Selection

• Azure Pipelines offers a number of options to configure where your build environment runs.

• Jenkins offers build agent selection using the agent option to ensure that your build pipeline - or a particular stage of the pipeline - runs on a particular build agent machine.

## Environment Variables

• In Azure Pipelines, you can configure variables that are used both within the YAML configuration and are set as environment variables during job execution.

• In Jenkins, you typically define environment variables for the entire pipeline.

## Predefined Variables

• Both Jenkins and Azure Pipelines set a number of environment variables to allow you to inspect and interact with the execution environment of the continuous integration system.

## Success and Failure Handling

• Azure Pipelines has a rich conditional execution framework that allows you to run a job, or steps of a job, based on a number of conditions including pipeline success or failure.

• Jenkins allows you to run commands when the build finishes (using the post section), when the build succeeds (using the success section), when the build fails (using the failure section), or always (using the always section) of the pipeline.

# Jenkins plug-ins for Azure

Many Azure services and features are accessible via Jenkins plug-ins. These services support an array of possibilities regarding continuous integration and continuous deployment (CI/CD) for your DevOps environment.

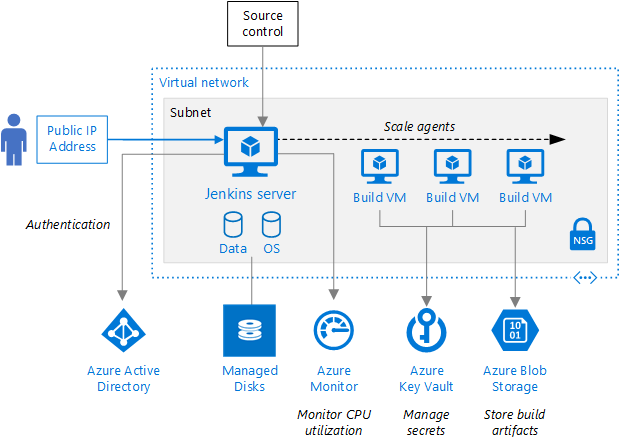
|  |  |  |
| --- | --- | --- |
| Serial Number | Jenkins plug-in | Description |
| 1 | [**Azure AD plug-in**](https://plugins.jenkins.io/azure-ad) | Jenkins plugin that supports authentication & authorization via Azure Active Directory. |
| 2 | [**Azure VM Agents plug-in**](https://plugins.jenkins.io/azure-vm-agents) | Jenkins plugin to create Jenkins agents in Azure virtual machines (via Azure Resource Manager template). |
| 3 | [**Azure Credentials plug-in**](https://plugins.jenkins.io/azure-credentials) | Jenkins plugin to manage Azure credentials. |
| 4 | **Windows** [**Azure Storage plug-in**](https://plugins.jenkins.io/windows-azure-storage) | Jenkins plugin for uploading build artifacts to, or downloading build dependencies from, Microsoft Azure Blob storage. |
| 5 | [**Azure Container Agents plug-in**](https://plugins.jenkins.io/azure-container-agents) | Azure Container Agents plugin can help you to run a container as an agent in Jenkins |
| 6 | [**Kubernetes Continuous Deploy plug-in**](https://plugins.jenkins.io/azure-vmss) | Jenkins plugin to deploy resource configurations to a Kubernetes cluster. |
| 7 | [**Azure Container Service plug-in**](https://plugins.jenkins.io/azure-acs) | Jenkins plugin to deploy configurations to Azure Container Service (AKS). |
| 8 | [**Azure Function plug-in**](https://plugins.jenkins.io/azure-function) | Jenkins plugin to deploy an Azure Function. |
| 9 | [**Azure App Service plug-in**](https://plugins.jenkins.io/azure-app-service) | Jenkins plugin to deploy an Azure App Service (currently supports only Web App). |
| 10 | [Azure Artifact Manager plug-in](https://plugins.jenkins.io/azure-artifact-manager) | Azure Artifact Manager plugin is an Artifact Manager that allows you store your artifacts into Azure Blob Storage. Azure Artifact Manager plugin works transparently to Jenkins and your jobs, it is like the default Artifact Manager. |
| 11 | [Azure Container Registry Task plug-in](https://plugins.jenkins.io/azure-container-registry-tasks) | Jenkins plugin to send a docker-build request to [Azure Container Registry](https://docs.microsoft.com/en-us/azure/container-registry/container-registry-tasks-overview). |
| 12 | [Azure IoT Edge plug-in](https://plugins.jenkins.io/azure-iot-edge) | Azure IoT Edge plugin makes it easy to set up a CI/CD pipeline for developing an IoT Edge solution on Jenkins. |
| 13 | [Azure Service Fabric plug-in](https://plugins.jenkins.io/service-fabric) | Jenkins plugin for Linux Azure Service Fabric projects. |
| 14 | [Azure virtual machine scale set plug-in](https://plugins.jenkins.io/azure-vmss) | Jenkins plugin to deploy VM images to Azure virtual machine scale sets. |
| 15 | Azure Commons | Jenkins plugin which provides common APIs for Azure related Jenkins plugins. |
| 16 | Azure Dev Spaces | Azure Dev Spaces Plugin helps to manage dev spaces in Azure Dev Spaces. |

**Note:**

Plug-ins in bold are recommended by Azure. To see all the latest list, visit Jenkins Plugin Index [[1]](#footnote-1)

# Case Study: Run a Jenkins Server on Azure

Deploy and operate a scalable, enterprise-grade Jenkins server on Azure secured with single sign-on (SSO). The architecture also uses Azure Monitor to monitor the state of the Jenkins server.



This architecture supports disaster recovery with Azure services but does not cover more advanced scale-out scenarios involving multiple primaries or high availability (HA) with no downtime. The focus is on the core Azure operations needed to support Jenkins, including the use of Azure Storage to maintain build artifacts, the security items needed for SSO, other services that can be integrated, and scalability for the pipeline. The architecture is designed to work with an existing source control repository. For example, a common scenario is to start Jenkins jobs based on GitHub commits.

## Architecture

The architecture consists of the following components:

• Resource group: A resource group is used to group Azure assets so they can be managed by lifetime, owner, and other criteria. Use resource groups to deploy and monitor Azure assets as a group and track billing costs by resource group. You can also delete resources as a set, which is very useful for test deployments.

• Jenkins server: A virtual machine is deployed to run Jenkins as an automation server and serves as Jenkins Primary. To deploy this VM, use the solution template for Jenkins on Azure. Nginx is installed on the VM to act as a reverse proxy to Jenkins. You can configure Nginx to enable SSL for the Jenkins server.

• Virtual network: A virtual network connects Azure resources to each other and provides logical isolation. In this architecture, the Jenkins server runs in a virtual network.

• Subnets: The Jenkins server is isolated in a subnet to make it easier to manage and segregate network traffic without affecting performance.

• Network security groups: Use network security groups to restrict network traffic from the Internet to the subnet of a virtual network.

• Managed disks: A managed disk is a persistent virtual hard disk (VHD) used for application storage and also to maintain the state of the Jenkins server and provide disaster recovery. Data disks are stored in Azure Storage.

• Azure Blob storage: The Windows Azure Storage plugin uses Azure Blob storage to store the build artifacts that are created and shared with other Jenkins builds.

• Azure Active Directory (Azure AD): Azure AD supports user authentication, allowing you to set up SSO. Azure AD service principals define the policy and permissions for each role authorization in the workflow, using Azure role-based access control (Azure RBAC). Each service principal is associated with a Jenkins job.

• Azure Key Vault: To manage secrets and cryptographic keys used to provision Azure resources when secrets are required, this architecture uses Azure Key Vault.

• Azure monitoring services: This service monitors the Azure virtual machine hosting Jenkins. This deployment monitors the virtual machine status and CPU utilization and sends alerts.

## Recommendations

Follow these recommendations unless you have a specific requirement that overrides them:

• Azure AD: Use the Azure AD tenant to enable SSO for Jenkins users and set up service principals that enable Jenkins jobs authenticated and authorized access to Azure resources.

• Storage: Use the Jenkins Windows Azure Storage plugin, which is installed from the Azure Marketplace, to store build artifacts that can be shared with other builds and tests

• Scalability considerations: Jenkins can scale to support very large workloads. For elastic builds, do not run builds on the Jenkins primary server. Instead, offload build tasks to Jenkins agents, which can be elastically scaled in and out as need.

• Scaling the Jenkins server: You can scale the Jenkins server VM up or down by changing the VM size.

• Availability considerations: Availability in the context of a Jenkins server means being able to recover any state information associated with your workflow, such as test results, libraries you have created, or other artifacts. Critical workflow state or artifacts must be maintained to recover the workflow if the Jenkins server goes down.

• Security considerations: Consider setting up HTTPS on the Nginx server being used for a secure logon. Ensure that the Jenkins configuration prevents Cross-Site Request Forgery (CSRF). This is the default for Microsoft Jenkins Server. Configure read-only access to the Jenkins dashboard by using the Matrix Authorization Strategy Plugin. Install the Azure Credentials plugin to use Key Vault to handle secrets for the Azure assets, the agents in the pipeline, and third-party components. Use Azure role-based access control (Azure RBAC) to restrict the access of the service principal to the minimum required to run the jobs. This helps limit the scope of damage from a rogue job.

• Manageability considerations: Use resource groups to organize the Azure resources that are deployed. Deploy production environments and development/test environments in separate resource groups, so that you can monitor each environment's resources. You can also delete resources as a set, which is very useful for test deployments.

1. <https://plugins.jenkins.io/ui/search?query=Azure> [↑](#footnote-ref-1)