


Run a Jenkins server on Azure

04/30/2018 • 12 minutes to read • Contributors 

In this article

[Architecture](#)

[Recommendations](#)

[Scalability considerations](#)

[Availability considerations](#)

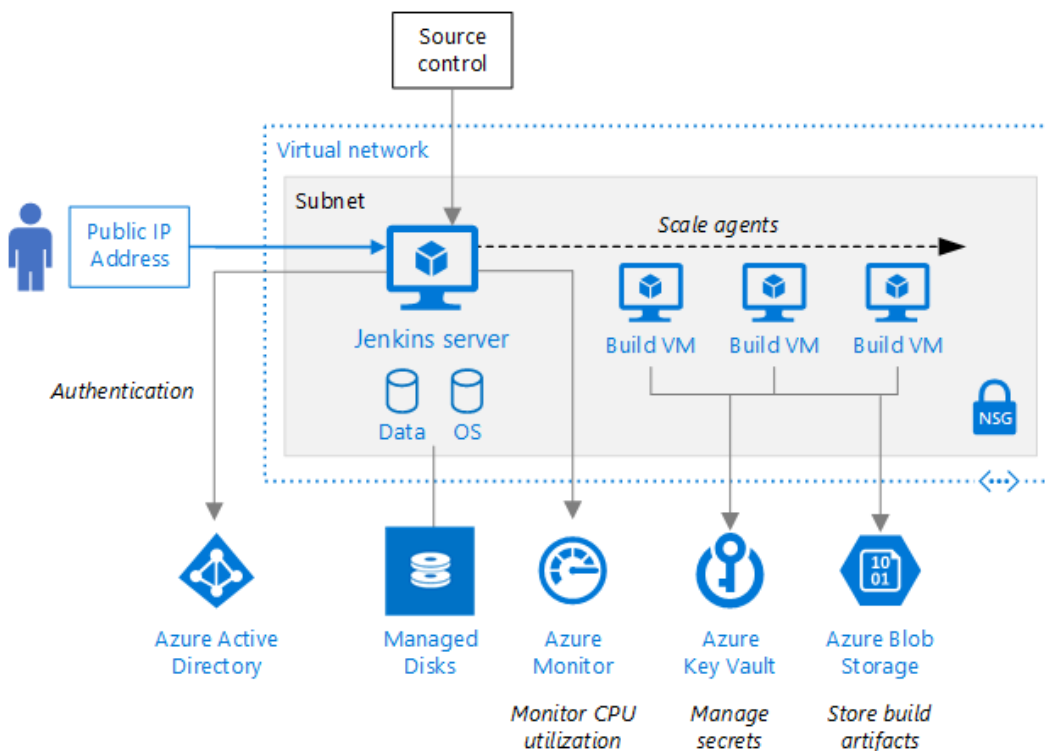
[Security considerations](#)

[Manageability considerations](#)

[Communities](#)

[Deploy the solution](#)

This scenario shows how to 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. [Deploy this solution.](#)



Download a [Visio file](#) that contains this architecture diagram.

This architecture supports disaster recovery with Azure services but does not cover more advanced scale-out scenarios involving multiple masters or high availability (HA) with no downtime. For general insights about the various Azure components, including a step-by-step tutorial about building out a CI/CD pipeline on Azure, see [Jenkins on Azure](#).

The focus of this document 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 Master. To deploy this VM, use the [solution template for Jenkins on Azure](#). Other Jenkins offerings are available in the Azure Marketplace.

❗ **Note**

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. For high performance, [premium storage](#) is recommended.
- **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 [role-based access control](#) (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](#). For added help storing secrets associated with the application in the pipeline, see also the [Azure Credentials](#) plugin for Jenkins.
- **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

The following recommendations apply for most scenarios. Follow these recommendations unless you have a specific requirement that overrides them.

Azure AD

The [Azure AD](#) tenant for your Azure subscription is used to enable SSO for Jenkins users and set up [service principals](#) that enable Jenkins jobs to access Azure resources.

SSO authentication and authorization are implemented by the Azure AD plugin installed on the Jenkins server. SSO allows you to authenticate using your organization credentials from Azure AD when logging on to the Jenkins server. When configuring the Azure AD plugin, you can specify the level of a user's authorized access to the Jenkins server.

To provide Jenkins jobs with access to Azure resources, an Azure AD administrator creates service principals. These grant applications — in this case, the Jenkins jobs — [authenticated, authorized access](#) to Azure resources.

[RBAC](#) further defines and controls access to Azure resources for users or service principals through their assigned role. Both built-in and custom roles are supported. Roles also help secure the pipeline and ensure that a user's or agent's responsibilities are assigned and authorized correctly. In addition, RBAC can be set up to limit access to Azure assets. For example, a user can be limited to working with only the assets in a particular resource group.

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. An Azure Storage account must be configured before this plugin can be used by the Jenkins jobs.

Jenkins Azure plugins

The solution template for Jenkins on Azure installs several Azure plugins. The Azure DevOps Team builds and maintains the solution template and the following plugins, which work with other Jenkins offerings in Azure Marketplace as well as any Jenkins master set up on premises:

- [Azure AD plugin](#) allows the Jenkins server to support SSO for users based on Azure AD.
- [Azure VM Agents](#) plugin uses an Azure Resource Manager template to create Jenkins agents in Azure virtual machines.
- [Azure Credentials](#) plugin allows you to store Azure service principal credentials in Jenkins.
- [Windows Azure Storage plugin](#) uploads build artifacts to, or downloads build dependencies from, [Azure Blob storage](#).

We also recommend reviewing the growing list of all available Azure plugins that work with Azure resources. To see all the latest list, visit [Jenkins Plugin Index](#) and search for Azure. For example, the following plugins are available for deployment:

- [Azure Container Agents](#) helps you to run a container as an agent in Jenkins.
- [Kubernetes Continuous Deploy](#) deploys resource configurations to a Kubernetes cluster.
- [Azure Container Service](#) deploys configurations to Azure Container Service with Kubernetes, DC/OS with Marathon, or Docker Swarm.
- [Azure Functions](#) deploys your project to Azure Function.
- [Azure App Service](#) deploys to Azure App Service.

Scalability considerations

Jenkins can scale to support very large workloads. For elastic builds, do not run builds on the Jenkins master server. Instead, offload build tasks to Jenkins agents, which can be elastically scaled in and out as need. Consider two options for scaling agents:

- Use the [Azure VM Agents](#) plugin to create Jenkins agents that run in Azure VMs. This plugin enables elastic scale-out for agents and can use distinct types of virtual machines. You can select a different base image from Azure Marketplace or use a custom image. For details about how the Jenkins agents scale, see [Architecting for Scale](#) in the Jenkins documentation.
- Use the [Azure Container Agents](#) plugin to run a container as an agent in either [Azure Container Service with Kubernetes](#), or [Azure Container Instances](#).

Virtual machines generally cost more to scale than containers. To use containers for scaling, however, your build process must run with containers.

Also, use Azure Storage to share build artifacts that may be used in the next stage of the pipeline by other build agents.

Scaling the Jenkins server

You can scale the Jenkins server VM up or down by changing the VM size. The [solution template for Jenkins on Azure](#) specifies the DS2 v2 size (with two CPUs, 7 GB) by default. This size handles a small to medium team workload. Change the VM size by choosing a different option when building out the server.

Selecting the correct server size depends on the size of the expected workload. The Jenkins community maintains a [selection guide](#) to help identify the configuration that best meets your requirements. Azure offers many [sizes for Linux VMs](#) to meet any requirements. For more information about scaling the Jenkins master, refer to the Jenkins community of [best practices](#), which also includes details about scaling Jenkins master.

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. To assess your availability requirements, consider two common metrics:

- Recovery Time Objective (RTO) specifies how long you can go without Jenkins.
- Recovery Point Objective (RPO) indicates how much data you can afford to lose if a disruption in service affects Jenkins.

In practice, RTO and RPO imply redundancy and backup. Availability is not a question of hardware recovery — that is part of Azure — but rather ensuring you maintain the state of your Jenkins server. Microsoft offers a [service level agreement](#) (SLA) for single VM instances. If this SLA doesn't meet your uptime requirements, make sure you have a plan for disaster recovery, or consider using a [multi-master Jenkins server](#) deployment (not covered in this document).

Consider using the disaster recovery [scripts](#) in step 7 of the deployment to create an Azure Storage account with managed disks to store the Jenkins server state. If Jenkins goes down, it can be restored to the state stored in this separate storage account.

Security considerations

Use the following approaches to help lock down security on a basic Jenkins server, since in its basic state, it is not secure.

- Set up a secure way to log into the Jenkins server. This architecture uses HTTP and has a public IP, but HTTP is not secure by default. Consider setting up [HTTPS on the Nginx server](#) being used for a secure logon.

📌 Note

When adding SSL to your server, create a network security group rule for the Jenkins subnet to open port 443. For more information, see [How to open ports to a virtual machine with the Azure portal](#).

- Ensure that the Jenkins configuration prevents cross site request forgery (Manage Jenkins > Configure Global Security). 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 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.

Jenkins jobs often require secrets to access Azure services that require authorization, such as Azure Container Service. Use [Key Vault](#) along with the [Azure Credential plugin](#) to manage these secrets securely. Use Key Vault to store service principal credentials, passwords, tokens, and other secrets.

To get a central view of the security state of your Azure resources, use [Azure Security Center](#). Security Center monitors potential security issues and provides a comprehensive picture of the security health of your deployment. Security Center is configured per Azure subscription. Enable security data collection as described in the [Azure Security Center quick start guide](#). When data collection is enabled, Security Center automatically scans any virtual machines created under that subscription.

The Jenkins server has its own user management system, and the Jenkins community provides best practices for [securing a Jenkins instance on Azure](#). The solution template for Jenkins on Azure implements these best practices.

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 and roll up billing costs by resource group. You can also delete resources as a set, which is very useful for test deployments.

Azure provides several features for [monitoring and diagnostics](#) of the overall infrastructure. To monitor CPU usage, this architecture deploys Azure Monitor. For example, you can use Azure Monitor to monitor CPU utilization, and send a notification if CPU usage exceeds 80 percent. (High CPU usage indicates that you might want to scale up the Jenkins server VM.) You can also notify a designated user if the VM fails or becomes unavailable.

Communities

Communities can answer questions and help you set up a successful deployment. Consider the following:

- [Jenkins Community Blog](#)
- [Azure Forum](#)
- [Stack Overflow Jenkins](#)

For more best practices from the Jenkins community, visit [Jenkins best practices](#).

Deploy the solution

To deploy this architecture, follow the steps below to install the [solution template for Jenkins on Azure](#), then install the scripts that set up monitoring and disaster recovery in the steps below.

Prerequisites

- This reference architecture requires an Azure subscription.
- To create an Azure service principal, you must have admin rights to the Azure AD tenant that is associated with the deployed Jenkins server.
- These instructions assume that the Jenkins administrator is also an Azure user with at least Contributor privileges.

Step 1: Deploy the Jenkins server

1. Open the [Azure Marketplace image for Jenkins](#) in your web browser and select **GET IT NOW** from the left side of the page.
2. Review the pricing details and select **Continue**, then select **Create** to configure the Jenkins server in the Azure portal.

For detailed instructions, see [Create a Jenkins server on an Azure Linux VM from the Azure portal](#). For this reference architecture, it is sufficient to get the server up and running with the admin login. Then you can provision it to use various other services.

Step 2: Set up SSO

The step is run by the Jenkins administrator, who must also have a user account in the subscription's Azure AD directory and must be assigned the Contributor role.

Use the [Azure AD Plugin](#) from the Jenkins Update Center in the Jenkins server and follow the instructions to set up SSO.

Step 3: Provision Jenkins server with Azure VM Agent plugin

The step is run by the Jenkins administrator to set up the Azure VM Agent plugin, which is already installed.

[Follow these steps to configure the plugin](#). For a tutorial about setting up service principals for the plugin, see [Scale your Jenkins deployments to meet demand with Azure VM agents](#).

Step 4: Provision Jenkins server with Azure Storage

The step is run by the Jenkins administrator, who sets up the Windows Azure Storage Plugin, which is already installed.

[Follow these steps to configure the plugin](#).

Step 5: Provision Jenkins server with Azure Credential plugin

The step is run by the Jenkins administrator to set up the Azure Credential plugin, which is already installed.

[Follow these steps to configure the plugin](#).

Step 6: Provision Jenkins server for monitoring by the Azure Monitor Service

To set up monitoring for your Jenkins server, follow the instructions in [Create metric alerts in Azure Monitor for Azure services](#).

Step 7: Provision Jenkins server with managed disks for disaster recovery

The Microsoft Jenkins product group has created disaster recovery scripts that build a managed disk used to save the Jenkins state. If the server goes down, it can be restored to its latest state.

Download and run the disaster recovery scripts from [GitHub](#).

You may wish to review the following [Azure example scenario](#) that demonstrates specific solutions using some of the same technologies:

- [CI/CD pipeline for container-based workloads](#)