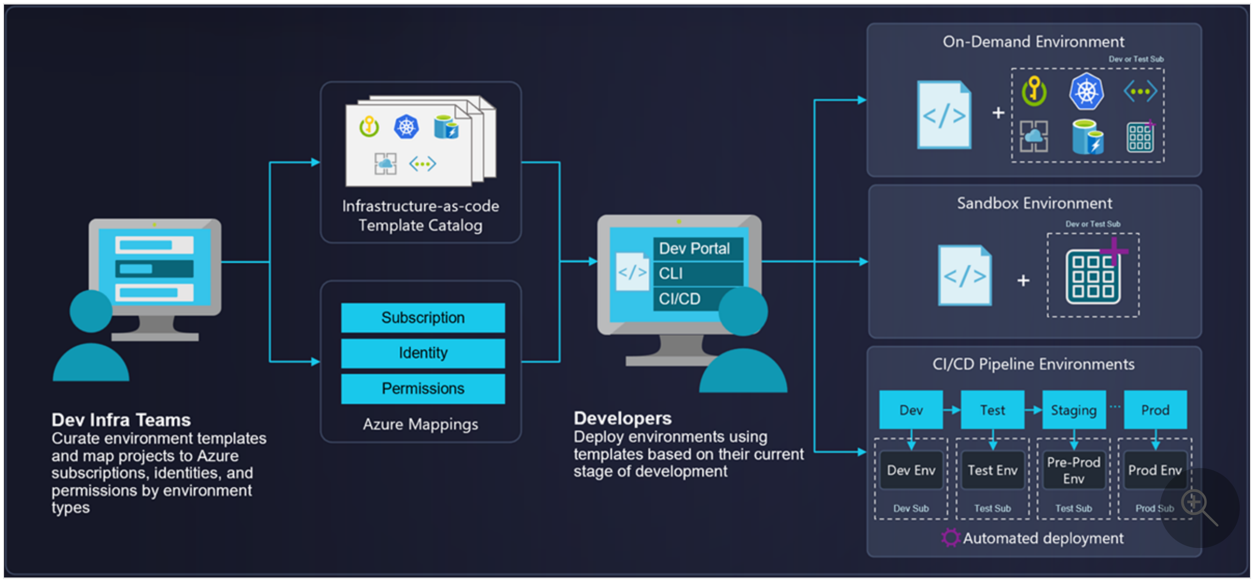
What is Azure Deployment Environments?

Article • 10/01/2023

Azure Deployment Environments empowers development teams to quickly and easily spin up app infrastructure with project-based templates that establish consistency and best practices while maximizing security. This on-demand access to secure environments accelerates the stages of the software development lifecycle in a compliant and cost-efficient way.

A deployment environment is a preconfigured collection of Azure resources deployed in predefined subscriptions. Azure governance is applied to those subscriptions based on the type of environment, such as sandbox, testing, staging, or production.


Dev Infra Teams
Curate environment templates and map projects to Azure subscriptions, identities, and permissions by environment types
Infrastructure-as-code Template Catalog
Subscription
Identity
Permissions
Azure Mappings
Dev Portal
CLI
CI/CD
Developers
Deploy environments using templates based on their current stage of development
On-Demand Environment
Dev or Test Sub
Sandbox Environment
Dev or Test Sub
CI/CD Pipeline Environments
Dev
Test
Staging
Prod
Dev Env
Test Env
Pre-Prod Env
Prod Env
Dev Sub
Test Sub
Test Sub
Prod Sub
Automated deployment

With Azure Deployment Environments, your platform engineer can enforce enterprise security policies and provide a curated set of predefined infrastructure as code (IaC) templates.

ⓘ **Note**

Azure Deployment Environments currently supports only Azure Resource Manager (ARM) templates.

You can learn more about the key concepts for Azure Deployment Environments.

Usage scenarios

Azure Deployment Environments enables usage scenarios for both DevOps teams and developers. Common scenarios include:

- Quickly create on-demand Azure environments by using reusable IaC templates.
- Create sandbox environments to test your code.
- Preconfigure various types of environments and seamlessly integrate with your continuous integration and continuous delivery (CI/CD) pipeline.
- Create preconfigured environments for trainings and demos.

Developer scenarios

Developers have the following self-service experience when working with environments.

> **Note**
>
> Developers have a CLI-based experience to create and manage environments for Azure Deployment Environments.

- Deploy a preconfigured environment for any stage of the development cycle.
- Spin up a sandbox environment to explore Azure.
- Create platform as a service (PaaS) and infrastructure as a service (IaaS) environments quickly and easily by following a few simple steps.
- Deploy environments right from where they work.

Platform engineering scenarios

Azure Deployment Environments helps your platform engineer apply the right set of policies and settings on various types of environments, control the resource configuration that developers can create, and centrally track environments across projects by doing the following tasks:

- Provide a project-based, curated set of reusable IaC templates.
- Define specific Azure deployment configurations per project and per environment type.
- Provide a self-service experience without giving control over subscriptions.

- Track costs and ensure compliance with enterprise governance policies.

Azure Deployment Environments supports two built-in roles:

- **Dev Center Project Admin**: Creates environments and manages the environment types for a project.
- **Deployment Environments User**: Creates environments based on appropriate access.

Benefits

Azure Deployment Environments provides the following benefits to creating, configuring, and managing environments in the cloud:

- **Standardization and collaboration**: Capture and share IaC templates in source control within your team or organization, to easily create on-demand environments. Promote collaboration through inner-sourcing of templates from source control repositories.

- **Compliance and governance**: Platform engineering teams can curate environment templates to enforce enterprise security policies and map projects to Azure subscriptions, identities, and permissions by environment types.

- **Project-based configurations**: Create and organize environment templates by the types of applications that development teams are working on, rather than using an unorganized list of templates or a traditional IaC setup.

- **Worry-free self-service**: Enable your development teams to quickly and easily create app infrastructure (PaaS, serverless, and more) resources by using a set of preconfigured templates. You can also track costs on these resources to stay within your budget.

- **Integration with your existing toolchain**: Use APIs to provision environments directly from your preferred CI tool, integrated development environment (IDE), or automated release pipeline. You can also use the comprehensive command-line tool.

Components shared with Microsoft Dev Box

Microsoft Dev Box and Azure Deployment Environments are complementary services that share certain architectural components. Dev Box provides developers with a cloud-based development workstation, called a dev box, which is configured with the tools they need for their work. Dev centers and projects are common to both services, and they help organize resources in an enterprise.

When configuring Deployment Environments, you may see Dev Box resources and components. You may even see informational messages regarding Dev Box features. If you're not configuring any Dev Box features, you can safely ignore these messages.

Next steps

Start using Azure Deployment Environments:

- Key concepts for Azure Deployment Environments
- Azure Deployment Environments scenarios
- Quickstart: Create and configure a dev center
- Quickstart: Create and configure a project
- Quickstart: Create and access environments