E-book series

Microsoft Azure

Code to cloud with Azure Kubernetes Service

01 /

Introduction

02 /

Inner loop

03 /

Outer loop

04 /

Next steps

INTRODUCTION

Go from code to cloud at warp speed

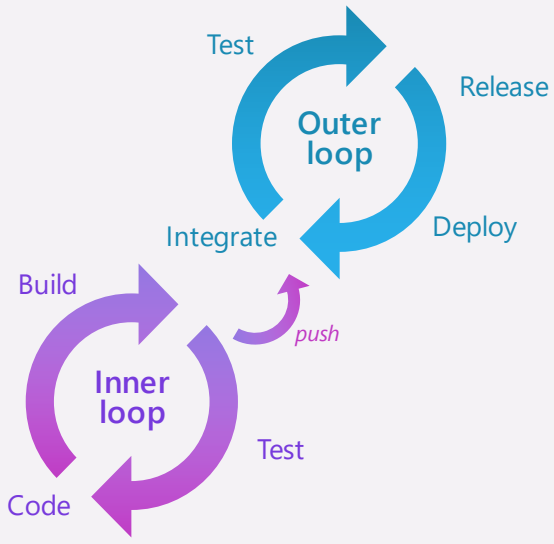
If you're just getting started with Kubernetes, you may be learning a new set of infrastructure concepts, like containerization, Kubernetes manifests, and ingresses. Even experienced developers have questions about containerizing and deploying their apps on Kubernetes clusters for the first time. However, you don't have to be an expert to get started quickly with Azure Kubernetes Service (AKS). As a fully managed version of the open-source Kubernetes platform, AKS automates the complexity of scaling distributed apps and handles many of the infrastructure details, making it the time-saving choice for teams building large-scale projects.

Using the recommendations in this e-book, you can get a noncontainerized application deployed on a Kubernetes cluster—in minutes. The code-to-cloud tool chain includes familiar environments and automated workflows that streamline the developer experience and support best practices in the software development lifecycle.

Get a faster, more productive developer experience

With Kubernetes, "code to cloud" really means "code to container to cloud." The workflow takes a suite of tools, starting with an integrated development environment (IDE) that supports Kubernetes. You also need a way to build and push a container image of your application and store it securely in a registry. Then you pull an image from the registry and deploy it to a Kubernetes cluster, for which you need secure access, a way of handling Kubernetes Secrets, and some form of Domain Name System (DNS) resolution, and other details.

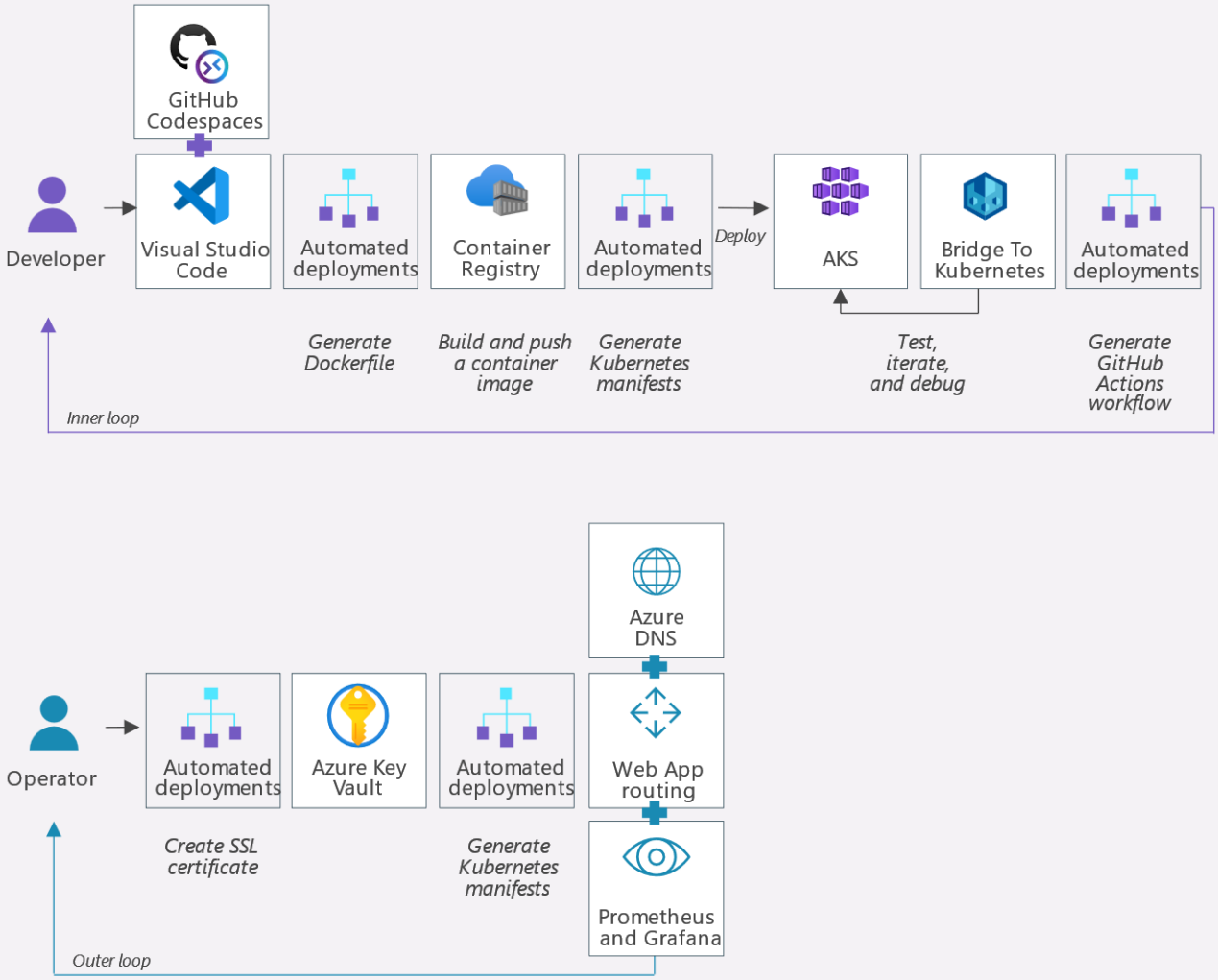
But what happens when you need to make changes to your application? You don't want to be slowed down by building and pushing a new container image every time. And what about security, keys and secrets, and other infrastructure considerations that development teams need to be aware of but often don't control?



The inner loop workflow is used to develop and test locally. The outer loop pipeline deploys apps to development and production environments.

The AKS code-to-cloud tool chain uses automation and AI to make the inner loop developer experience fast and fun while streamlining the handoff to the platform operators in the outer loop. For example, you can use Visual Studio Code with a smart copilot to automate repetitive steps and streamline the iterative process of writing, building, and debugging code. When it's time to build and deploy your code, the automated deployments feature in AKS handles critical steps for you.

In the outer loop, a continuous integration (CI) and continuous deployment (CD) pipeline automates the build and deployment process across the clusters you use for testing, development, staging, and production. Here, too, automated deployments are the key to a quick setup.



Our recommended code-to-cloud tool chain helps automate and integrate the steps needed to deploy container apps and microservices to Azure Kubernetes Service (AKS) clusters.

"We will actually rebuild the full house to run on Azure and AKS to create flexibility and scalability in our digital tech stack."

Søren Bering Andersen
Head of Digital & Technology,
LEGO House

INNER LOOP

Use Codespaces to configure your dev environment

Let's say you want to create microservices to run on a Kubernetes cluster, and you've chosen Docker to containerize your application. You're ready to set up your IDE. You need to install Docker and kubectl (the Kubernetes command-line interface), plus any tools specific to the programming language you're using. And you need to go through the same process for the next project you want to start.

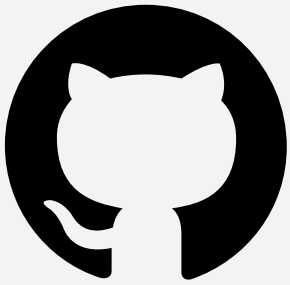
There's a better way. You can spin up a dev environment configuration as easily as Kubernetes spins up nodes. With GitHub Codespaces, you and your project's collaborators get the same environment, saving you the time that you would have spent in setup tasks. Codespaces are tailored development environments, like your own personal IDE in the cloud.

A codespace is hosted in a Docker container running on a virtual machine (VM) with up to 32 cores and 64 GB RAM. Access is flexible—you and your team can connect to your codespace from a web browser or from Visual Studio Code, or you can use Git CLI.

The environment on your local machine isn't changed. When you launch your codespace, you're good to go. You don't need to install dependencies on your developer machine to build and run the code because the codespace standardizes your project's configuration.

Your codespace can specify your runtime requirements, hardware specifications, extensions, and editor settings, not to mention all the Kubernetes tools you need, such as Docker, kubectl, the Azure CLI (az), Visual Studio Code extensions, and project-specific build tools, like npm, Gradle, Maven, or dotnet. You then check the configuration into your source code's repository, where it's ready every time you need it.

TIP ▸ Organizations that want more control over the development environment can explore Microsoft Dev Box, an Azure service that provides preconfigured, project-specific developer workstations.



INNER LOOP

Add extensions to Visual Studio Code

When you launch your codespace, it can include all the AKS extensions you need to work productively from within a familiar Visual Studio Code environment. With these extensions, you can generate smarter code faster in Visual Studio Code.

Github Copilot offers suggestions as you code and works with popular programming languages. An AI-pair programmer, it uses the language models developed by OpenAI to speed up development. Copilot can suggest code completions, generate snippets, autofill repetitive code patterns, and show you alternative solutions. It even converts comments to code.

We recommend adding the GitHub Copilot extension to your dev container for the codespace so that any future codespaces you create will have it.

AKS extension for Visual Studio Code gives you one-click access to AKS commands and features. This extension makes Visual Studio Code aware of your Azure subscription and resources, in addition to the namespaces and services running on your Kubernetes cluster. It enables you to:

- Draft a Dockerfile for your application code.
- Build a container image using Azure Container Registry.
- Draft Kubernetes deployment and service manifests.
- Draft a Kubernetes ingress that uses the Web Application Routing add-on with Azure DNS and Azure Key Vault integration.
- Draft a CI/CD workflow based on GitHub Actions.

Azure Account extension adds Azure Cloud Shell to the Visual Studio Code terminal so you can use Bash or Azure PowerShell to interactively manage Azure resources. It also provides a single Azure sign-in and subscription filtering for all other Azure extensions.

Developer tools for AKS extension streamlines noncluster tasks in Visual Studio Code, such as creating deployment files and configuring GitHub Actions workflows to deploy applications to AKS.

INNER LOOP

Use automated deployments

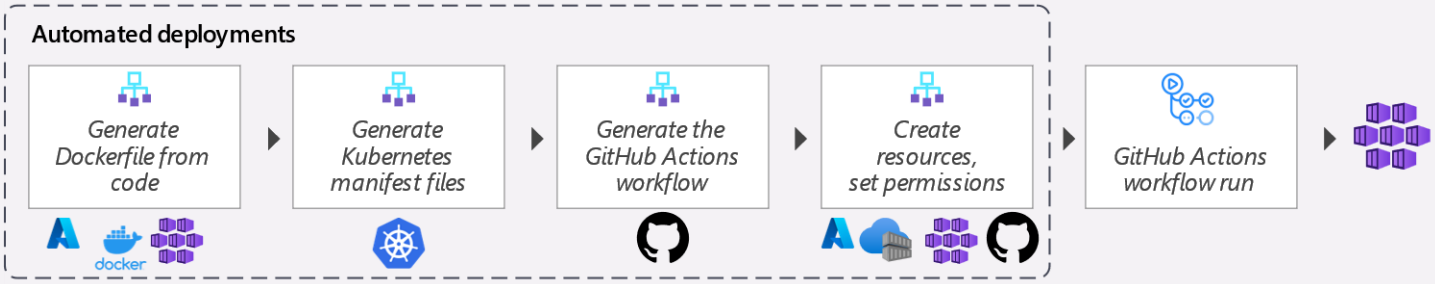
You don't need to understand all the moving parts to move to the next step quickly. Using automated deployments, you can build what is essentially a first draft of a containerized application running in AKS.

This feature is based on Draft, an open-source tool created by Microsoft and extended by the community. You can select **Automated deployments** in the Azure portal, work with the command-line version, or get the extension for Visual Studio Code.

Automated deployments generate the artifacts associated with a containerized application for you, including:

- Creating the Dockerfile, generating the Kubernetes manifests (deployment.yaml and service.yaml), and deploying your application to the cluster you specify.
- Setting up an automated workflow for the software development lifecycle and keeping it in sync by generating a GitHub Actions workflow file.

Azure resources are created for you, including Azure Key Vault and Azure DNS. Just like that, you have a CI/CD pipeline that automatically deploys changes to your Kubernetes cluster.



The automated deployments feature generates the artifacts associated with a containerized application for you, including the actions used for a CI/CD workflow.

INNER LOOP

Connect and iterate using Bridge to Kubernetes

If you had to build and deploy your code to your Kubernetes cluster every time you made a change, you'd spend a lot of time waiting. Unless you're building a small app, it's probably impractical to clone your entire application and its dependencies on your personal computer so you can iterate locally.

The Bridge to Kubernetes tool solves this problem by enabling you to run and debug your code locally—regardless of the number of microservices or interdependent services and databases.

Bridge to Kubernetes is an open-source tool that works from the command line and with Visual Studio, Visual Studio Code, and GitHub Codespaces. The *bridge* refers to a connection between your dev computer and your cluster. As you test and iterate locally, Bridge to Kubernetes takes care of routing, isolating your development traffic, and redirecting requests to your development environment.

This local port forwarding means that multiple team members can develop in isolation while avoiding disruption to other traffic in the cluster. You can run your code natively in your development environment while connected to a Kubernetes cluster and test your code changes in the context of the larger application—without having to deploy all the application dependencies locally. You also don't need to build, push, and deploy a new container image for each code change.

"A tool like GitHub Copilot is so impactful at large companies because suddenly engineers can make impactful changes to other developers' code with little previous exposure."

Severin Hacker

Chief Technology Officer,
Duolingo

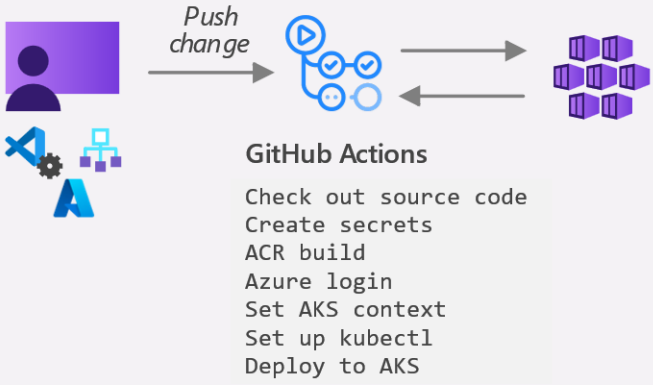
OUTER LOOP

Automate CI/CD with GitHub Actions

When you use AKS automated deployments to generate a deployment workflow, behind the scenes, GitHub Actions are doing the work. An action specifies the step to take, such as creating the manifest, creating secrets, deploying the manifest, and substituting artifacts. The result is an automated software development lifecycle workflow.

To get a closer look and customize these actions, see the GitHub Actions for Kubernetes. For example, the action to deploy Kubernetes manifests to Kubernetes clusters (`azure/k8s-deploy`) supports strategies for basic, canary, and blue-green deployments.

You can also use GitHub Actions for Azure to create repeatable workflows in your repository so you can build, test, package, release, and deploy to Azure using Azure services.



When you use automated deployments, Visual Studio Code extensions, or the Azure portal to generate a deployment workflow, you are using GitHub Actions.

OUTER LOOP

Streamline DevOps and the outer loop

Just as GitHub Actions automate the CI/CD workflow, infrastructure automation helps smooth the transition from your inner loop development environment to the outer loop. Infrastructure automation applies DevOps best practices to your Kubernetes clusters, automating version control, collaboration, and compliance.

When you combine CI/CD with infrastructure automation, you create a powerful DevOps pipeline that simplifies configuration management and creates a reliable audit trail of changes. Here are some best practices.

Automate deployment safeguards to enforce best practices. Based on Azure Policy, deployment safeguards assign the Warning level to tell you when an image is out of compliance. The Enforcement level denies deployments that don't follow best practices.

Track operations with GitOps. Infrastructure as code (IaC) automates system and infrastructure management. GitOps is a popular choice that enables you to declaratively track the desired state of your Kubernetes clusters. A Git repository contains versioned files and manifests used by the Kubernetes controllers running in your clusters. The controllers continually reconcile the cluster state with the desired state declared in your repository. Changes between versions are easily tracked, and you can recreate clusters with the same desired state when you need to.

OUTER LOOP

Monitor and observe

The code-to-cloud experience needs visibility. Managed versions of the popular Grafana dashboards and Prometheus metrics make it easy to collect and analyze AKS data. Azure Monitor managed service for Prometheus collects data about your AKS clusters, and Azure Managed Grafana brings it together.

For Azure-native monitoring with one click, you can use autoinstrumentation for Azure Monitor Application Insights. This shortcut enables Application Insights to make metrics, requests, dependencies, and other telemetry available in your Application Insights resource and provides easy access to an overview dashboard and application map.

Another way to keep tabs on your distributed apps is to store all your application settings and feature flags in one place. Azure App Configuration gives you a central location for managing these settings. You can dynamically change application settings without redeploying or restarting your application, and you can control feature availability in real time.

To go a step further, you can add a dedicated infrastructure layer called a *service mesh* to your applications. It enables you to transparently add observability, traffic management, security, and other capabilities without adding them to your own code. The Istio-based service mesh add-on provides an officially supported and tested integration for AKS.

TIP ▸ For more shortcuts and quick insights, try using Microsoft Copilot for Azure in the Azure portal. For example, you can ask Copilot, "Tell me what my average CPU utilization is for this environment" and get a detailed operational rundown.

Next steps

When you're ready to go deeper, here are some ideas.

Simplify autoscaling. As your deployment grows, you need to consider scalability. For burst or high-volume scenarios, the Kubernetes Event-driven Autoscaling (KEDA) add-on applies event-driven autoscaling to your application to meet demand cost-effectively. You can set scale rules declaratively based on several metrics, including Azure Storage Queue length, Azure Service Bus messages, and dozens of KEDA scalers.

Start secure and stay secure. To *start secure* means to detect and remediate issues before they start moving down the pipeline to the registry, cluster, nodes, and application. To *stay secure*, you need threat detection and remediation tools, plus ongoing workload protection. For a helpful end-to-end framework, see the Microsoft Containers Secure Supply Chain. For security management, we recommend Microsoft Defender for Cloud.

Learn more at azure.com

©2024 Microsoft Corporation. All rights reserved. This document is provided "as is." Information and views expressed in this document, including URL and other internet website references, may change without notice. You bear the risk of using it. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.