

# Azure Cloud Commanders

Microsoft Learning Room



# Recording

This session is being recorded on behalf of the  
Azure Cloud Commanders Learning Room.

If you do not consent to be a part of a recorded session, please  
disconnect your line.



# Inclusive session guidelines

## Code of Conduct

- Be welcoming and respectful
- Lead with empathy
- Be friendly, open, and welcoming
- Be understanding of differences
- Be kind and considerate to others

## Audience Guidance

- Use raise hand to ask a question
- Questions in chat to be prefaced with Q:
- @mention to respond to an individual
- Return to mute when not talking
- Use alt-text for images and gifs
- Immersive reader can be found by clicking on ... in the meeting chat

“Engage in cutting-edge AI technologies, build in-demand skills,  
and earn a digital badge by completing one of six unique challenges.

Each topic features interactive community events, and expert-led sessions to deepen your understanding.

The challenge ends on November 1”

Build Intelligent Apps

Deploy cloud-native apps using Azure Container Apps



# Deploy cloud-native apps using Azure Container Apps





# Hello!

Thank you for joining us today

## Speaker: Luke Murray

Microsoft Learn Expert and Microsoft Azure MVP, and Senior Consultant from Hamilton, New Zealand.



<https://x.com/lukemurraynz>

<https://www.linkedin.com/in/ljmurray/>

<https://luke.geek.nz/>

<https://www.youtube.com/@lukemurraynz>

**DATACOM**



APPLIED SKILLS

# Deploy cloud-native apps using Azure Container Apps

Validate your technical skills and open doors to new possibilities of advancement with Microsoft Applied Skills.

## At a glance



**Level**

Intermediate



**Role**

Developer, DevOps Engineer



**Product**

Azure, Azure Container Apps, Azure Container Registry, Azure Pipelines



**Subject**

Application development

## Overview

To earn this Microsoft Applied Skills credential, learners demonstrate the ability to deploy cloud-native applications using Azure Container Apps.

Candidates should be familiar with Azure platform services, as well as container-related technologies. Candidates for this credential should be familiar with:

- Continuous integration and continuous deployment (CI/CD)
- Azure DevOps
- Azure CLI
- PowerShell
- Bash

## Tasks performed

- Configure a secure connection between an Azure Container Registry and an ACA
- Create and configure a container app in Azure Container Apps
- Configure continuous integration by using Azure Pipelines
- Scale a deployed app in Azure Container Apps
- Manage revisions in Azure Container Apps



# What we are going to cover today....

## Deploy cloud-native apps using Azure Container Apps credential

### Introduction to Azure Container Apps

- What are Azure Container Apps?
- Container Apps vs. Other Azure Container Services
- Container Apps Architecture

### Getting Started with Azure Container Apps

- Prerequisites
- Setting Up the Environment
- Demo – Environment setup

### Building and Deploying Your First Container App

- Creating a Simple Application
- Building and Pushing the Container Image
- Deploying the Container App
- Demo - Build, Push, and Deploy

### Scaling and Managing Container Apps

- Scaling Concepts
- Configuring Scaling Rules
- Demo – Implementing Scaling rules
- Revision management
- Demo – Create a new revision and spilt traffic

### CI/CD and DevOps integration

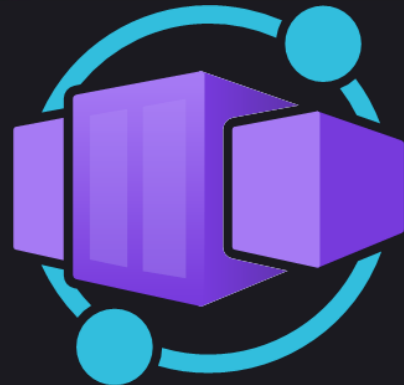
- CI/CD Pipeline Overview
- Implementing a CI/CD Pipeline
- Demo – Run a Azure DevOps Pipeline



# Introduction to Azure Container Apps

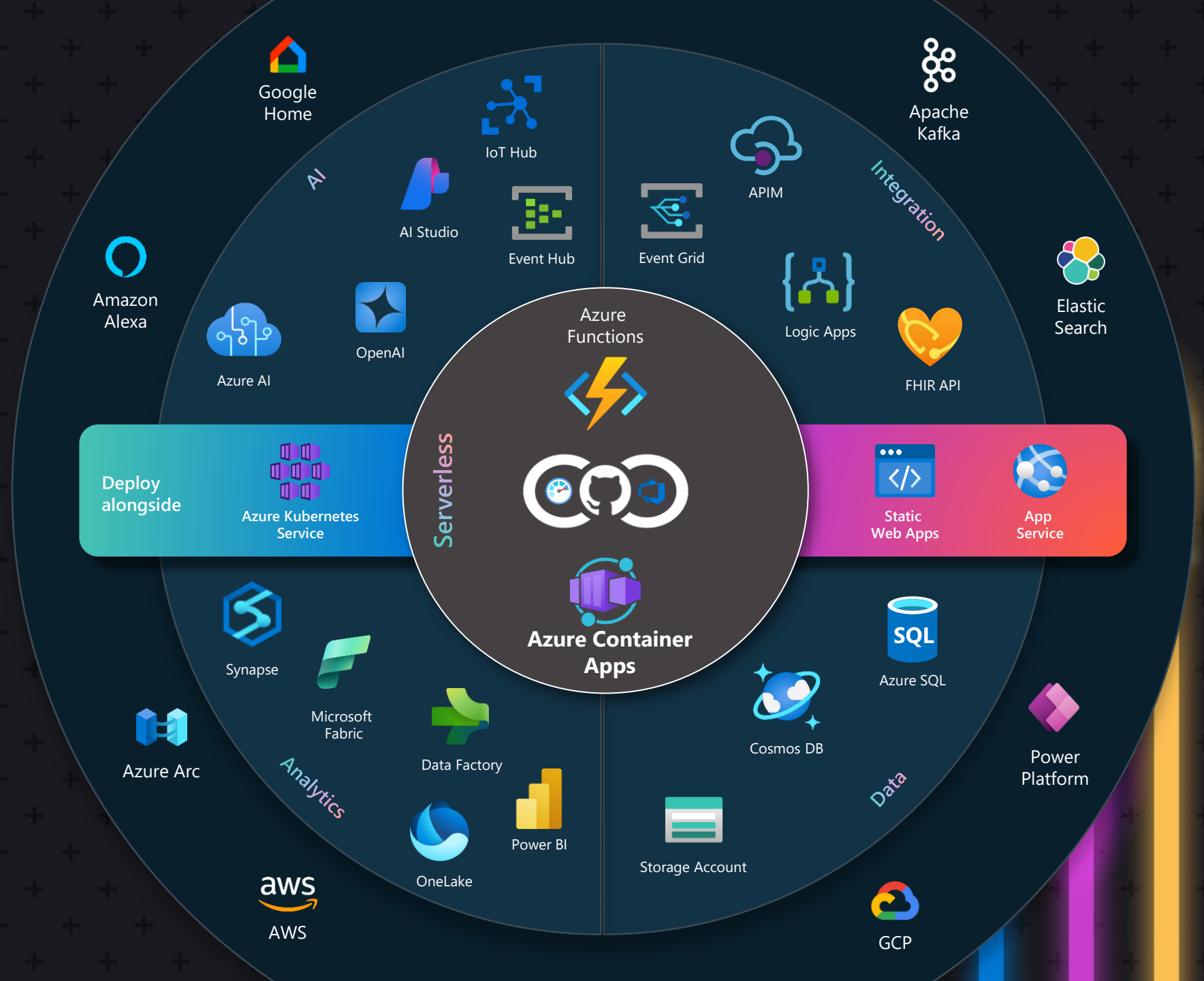


**“Azure Container Apps enables  
executing application code packaged in  
any container without managing any  
infrastructure.”**



# Examine cloud- native apps

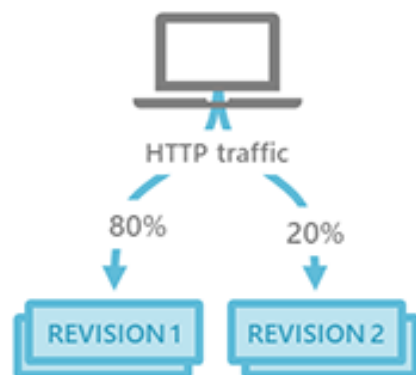
Designed to thrive in a dynamic, virtualized cloud environment, cloud-native apps make extensive use of Platform as a Service (PaaS) compute infrastructure and managed services



Feature	Azure Container Apps	Azure Kubernetes Service (AKS)	AKS Automatic	Azure Container Instances
Management Complexity	Low	High	Medium	Very Low
Orchestration	Managed Kubernetes-based	Full Kubernetes	Full Kubernetes	None
Scaling	Automatic, including scale to zero	Manual or auto-scaling	Automatic	Manual
Minimum Instances	Can scale to zero	At least one node	At least one node	Single instance
Load Balancing	Built-in	Configurable	Configurable	Not included
Kubernetes API Access	No direct access	Full access	Full access	No access
Deployment Complexity	Low	High	Medium	Very Low
Revision Management	Built-in	Manual	Manual	Not applicable
Traffic Splitting	Supported	Requires additional setup	Requires additional setup	Not supported
Event-driven Scaling	Native support (KEDA)	Requires additional setup	Requires additional setup	Not supported
Dapr Integration	Native	Supported	Supported	Not supported
Networking	VNet integration	Advanced networking	Advanced networking	Basic
Service Discovery	Built-in	Configurable	Configurable	Not included
Persistent Storage	Limited options	Multiple options	Multiple options	Limited
Security	Managed security	Customizable	Customizable with defaults	Basic
Monitoring	Built-in Azure Monitor	Azure Monitor for containers	Azure Monitor for containers	Basic Azure Monitor
Cost	Pay-per-use	Pay for nodes	Pay for nodes with optimization	Pay-per-second
Use Case	Microservices, event-driven apps	Complex containerized apps	Simplified Kubernetes	Simple, isolated containers
Learning Curve	Low	Steep	Moderate	Very Low
Customization	Limited	Extensive	Moderate	Limited
Multi-region Support	Limited	Supported	Supported	Supported
Hybrid/Edge Scenarios	Limited	Supported	Supported	Not ideal

# Example use case

## PUBLIC API ENDPOINTS



HTTP requests are split between two versions of the container app where the first revision gets 80% of the traffic, while a new revision receives the remaining 20%.

### AUTO-SCALE CRITERIA

Scaling is determined by the number of concurrent HTTP requests.

## BACKGROUND PROCESSING



A continuously-running background process that transforms data in a database.

### AUTO-SCALE CRITERIA

Scaling is determined by the level of CPU or memory load.

## EVENT-DRIVEN PROCESSING

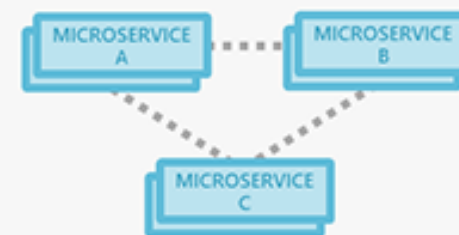


A queue reader application that processes messages as they arrive in a queue.

### AUTO-SCALE CRITERIA

Scaling is determined by the number of messages in the queue.

## MICROSERVICES



Deploy and manage a microservices architecture with the option to integrate with Dapr.

### AUTO-SCALE CRITERIA

Individual microservices can scale according to any KEDA scale triggers.



# Container Apps architecture



**Environments** are an isolation boundary around a collection of container apps.

**ENVIRONMENT:** OPTIONAL CUSTOM VIRTUAL NETWORK

CONTAINER APP 1

REVISION 1

REPLICA

CONTAINER(S)

REVISION 2

REPLICA

CONTAINER(S)

CONTAINER APP 2

REVISION 1

REPLICA

CONTAINER(S)

REVISION 2

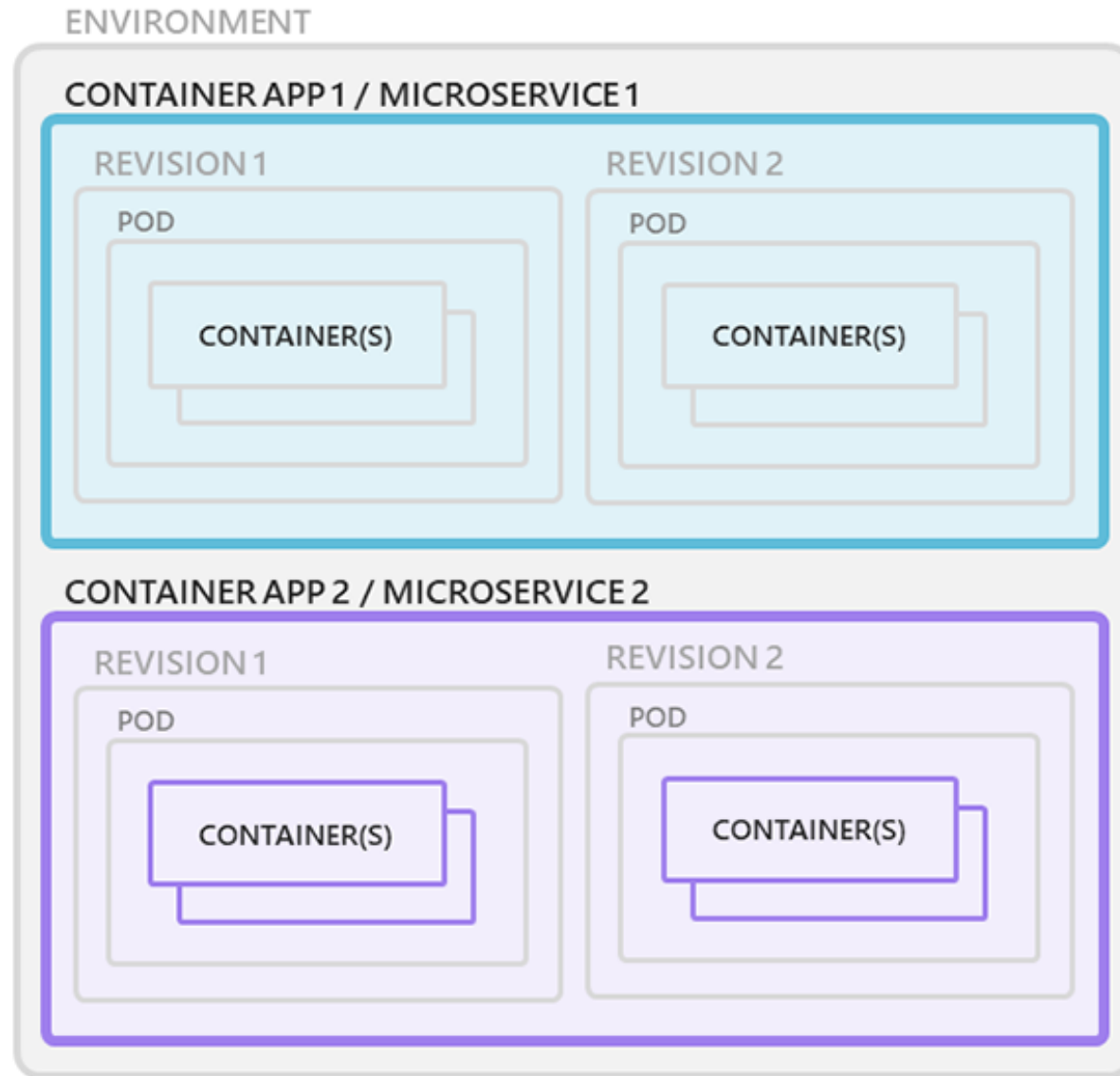
REPLICA

CONTAINER(S)

# Container Apps architecture



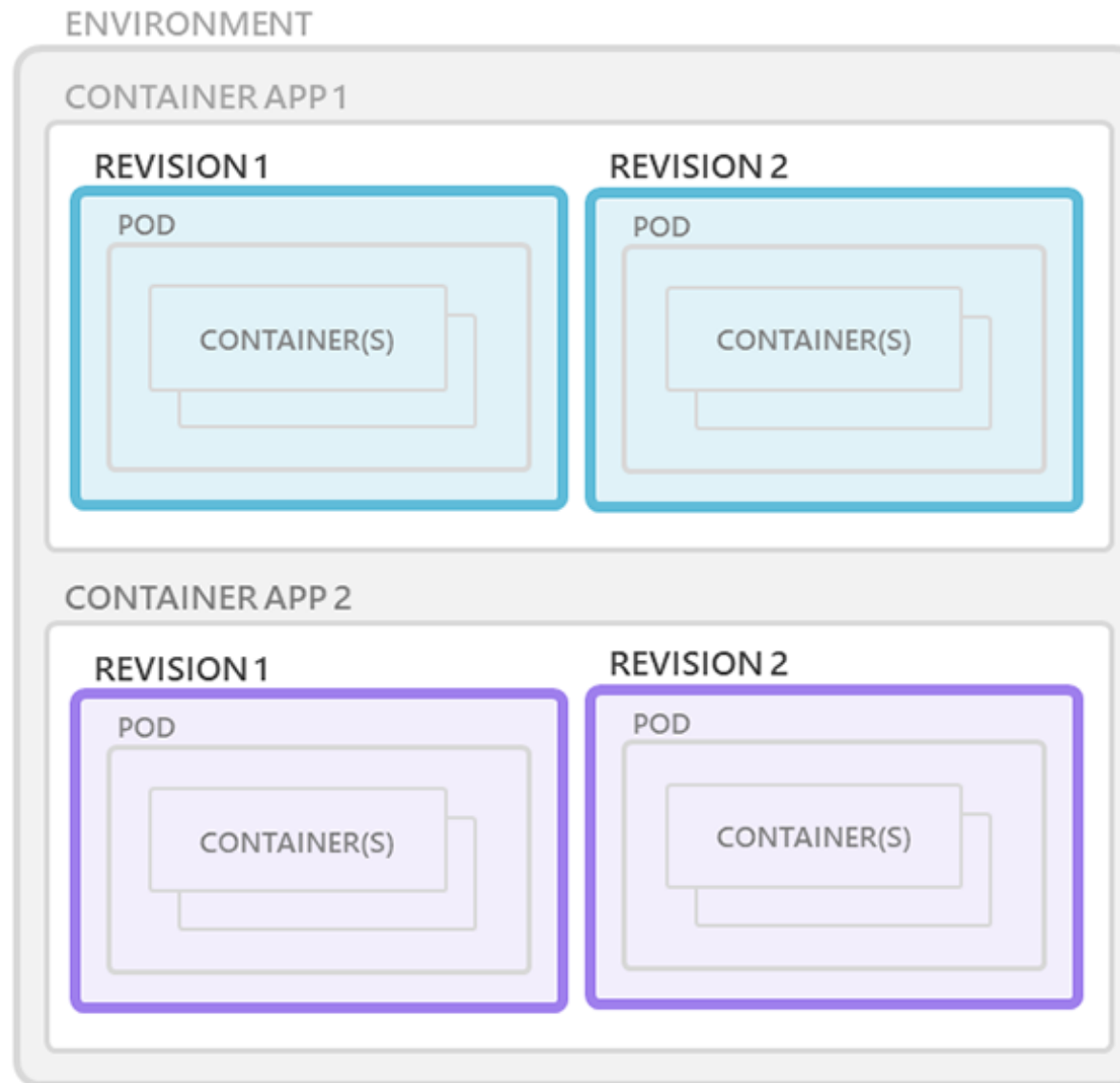
Container apps are  
deployed as  
**microservices.**



# Container Apps architecture



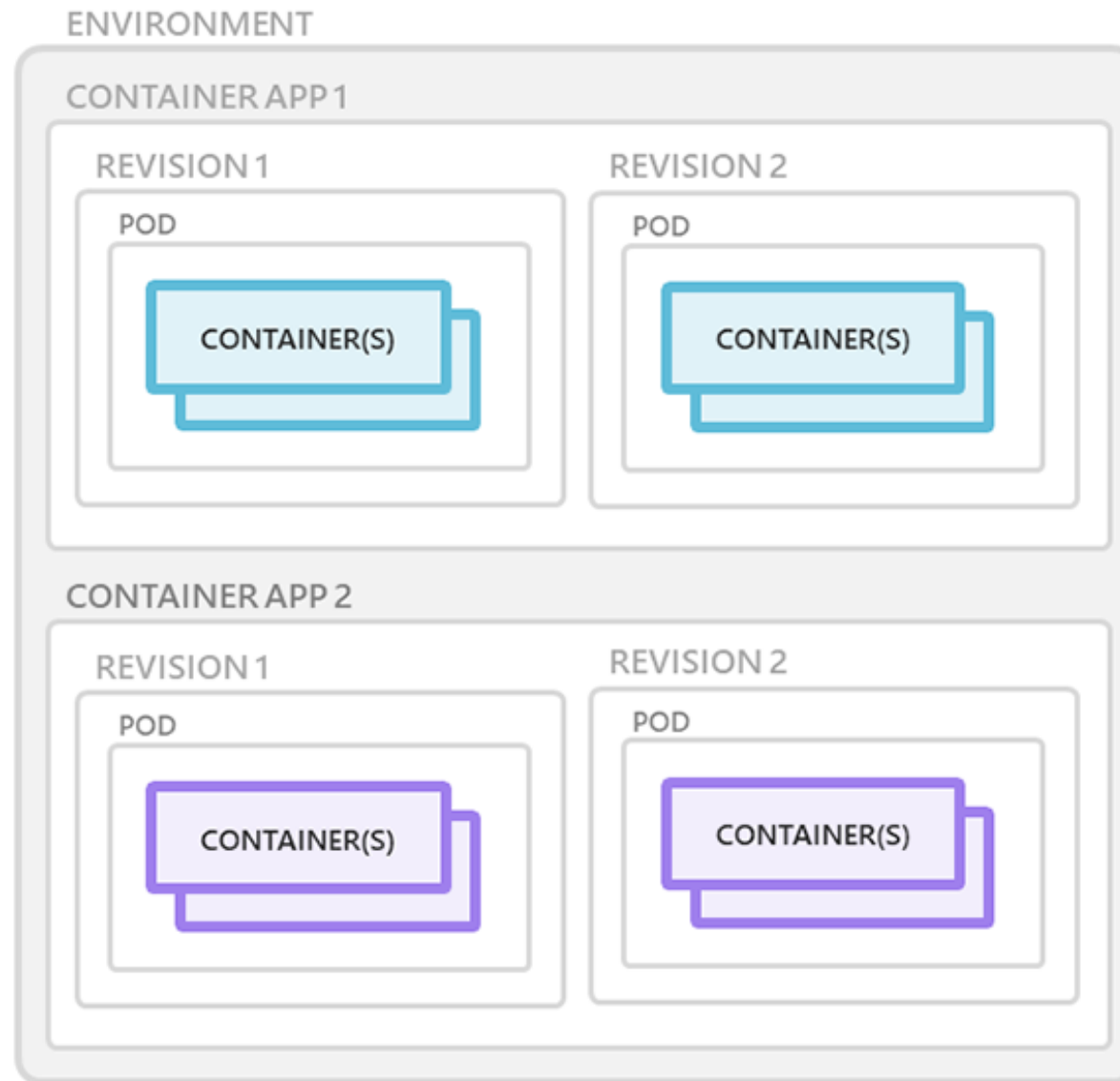
**Revisions** are immutable snapshots of a container app.



# Container Apps architecture



**Containers** for an Azure Container App are grouped together in pods inside revision snapshots.



# Container Apps architecture

## Scaling Container App or Job using KEDA

```
az containerapp job create `
  --name "aca-job-demo" `
  --resource-group "rg-aca-job-event-trigger" `
  --environment "aca-environment" `
  --replica-timeout 600 `
  --replica-retry-limit 1 `
  --replica-completion-count 1 `
  --parallelism 1 `
  --image "quickstart-jobs:latest" `
  --cpu "0.25" `
  --memory "0.5Gi" `
  --min-executions 0 `
  --max-executions 1 `
  --trigger-type "Event" `
  --secrets service-bus-connection-string='Endpoint=sb://servicebus...' `
  --scale-rule-name "azure-servicebus-queue-rule" `
  --scale-rule-type "azure-servicebus" `
  --scale-rule-auth "connection=service-bus-connection-string" `
  --scale-rule-metadata "namespace=servicebus-ns-job" `
  "queueName=queue-messages" `
  "messageCount=1"
```

The screenshot displays the Azure portal interface for a Service Bus Namespace named 'service-bus-namespace-13579'. The left sidebar shows the navigation menu with options like Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. The main content area is divided into two panels. The left panel shows the 'Shared access policies' section, which includes a table of policies. The right panel shows the 'SAS Policy: RootManageSharedAccess' configuration page. This page includes fields for the Primary Key, Secondary Key, Primary Connection String, Secondary Connection String, and the SAS Policy ARM ID. The Primary Key is 'C5mZSy1B5zvaZluRjyhVu0cUzdT9ckdQr+ASbDvEnOM=' and the Secondary Key is '0FdNGLc6RLOy8oB2nZHb3VM6Q0k9amHbV+ASbA7uXD4='.

Home > rg-aca-job-event-trigger > service-bus-namespace-13579 |

service-bus-namespace-13579 |  
Service Bus Namespace

Search

+ Add

Search to filter i

Policy

RootManageSi

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Shared access policies

Geo-Recovery

Migrate to premium

Encryption

Configuration

Properties

SAS Policy: RootManageSharedAccess

Save Discard Delete Regenerate Primary Key

☒ Manage

☒ Send

☒ Listen

Primary Key

C5mZSy1B5zvaZluRjyhVu0cUzdT9ckdQr+ASbDvEnOM=

Secondary Key

0FdNGLc6RLOy8oB2nZHb3VM6Q0k9amHbV+ASbA7uXD4=

Primary Connection String

Endpoint=sb://service-bus-namespace-13579.servicebus.windows.net/

Secondary Connection String

Endpoint=sb://service-bus-namespace-13579.servicebus.windows.net/

SAS Policy ARM ID

/subscriptions/82f6d75e-85f4-434a-ab74-5dddd9fa8910/resourcegro



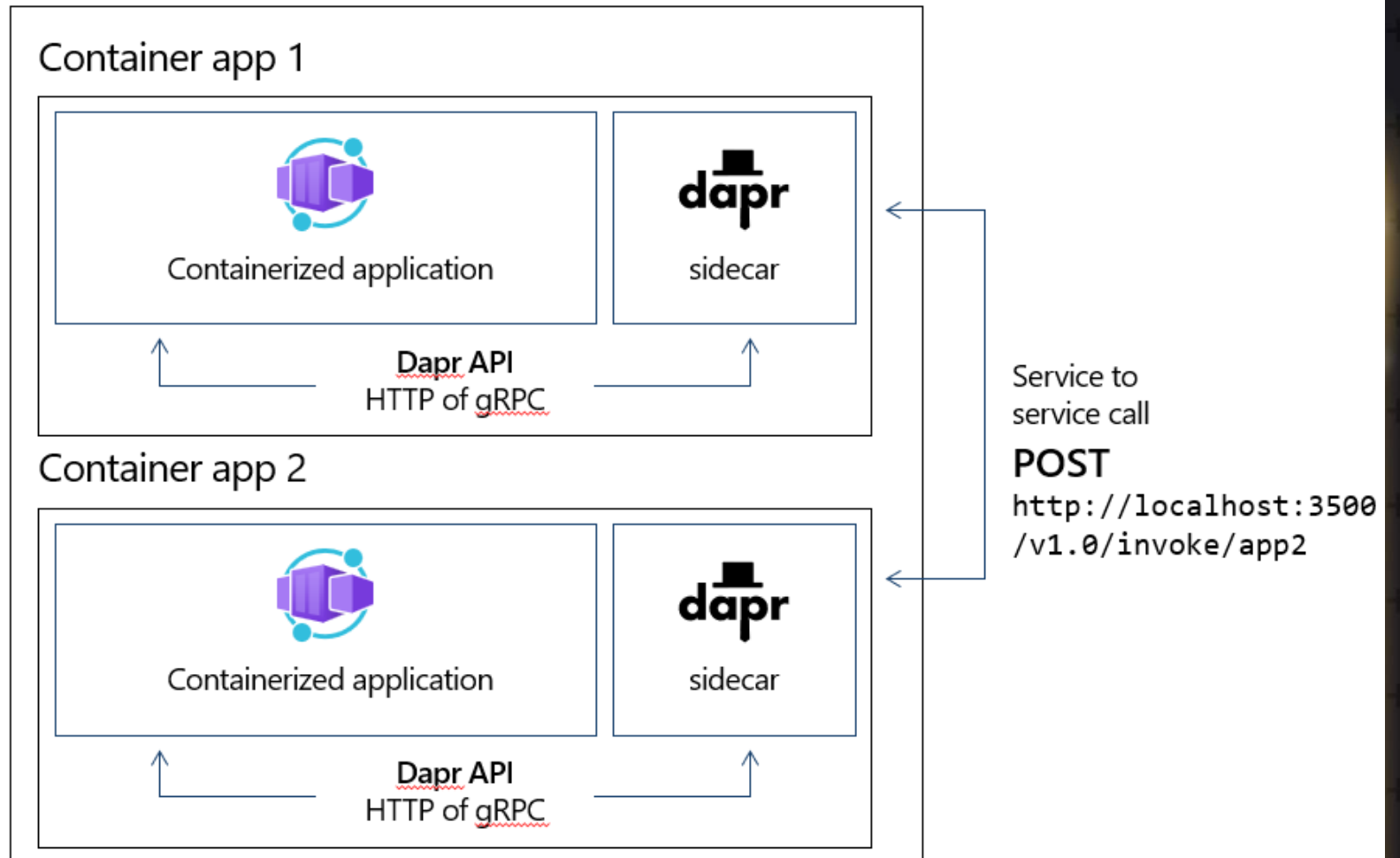
# Container Apps architecture

## Dapr sidecar

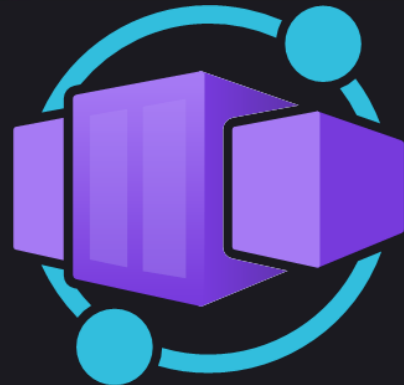
### Service to service invocation

Fully managed Dapr APIs provide a rich set of capabilities and productivity gains

#### Environment



# Getting Started with Azure Container Apps



# Prerequisites for Azure Container Apps

Microsoft Azure subscription

VNET (Virtual Network)

Azure DevOps

Container Registry

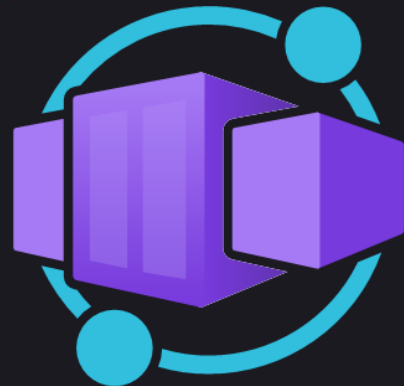
Container App Environment

# Prerequisites for Azure Container Apps

## VNET Virtual Network

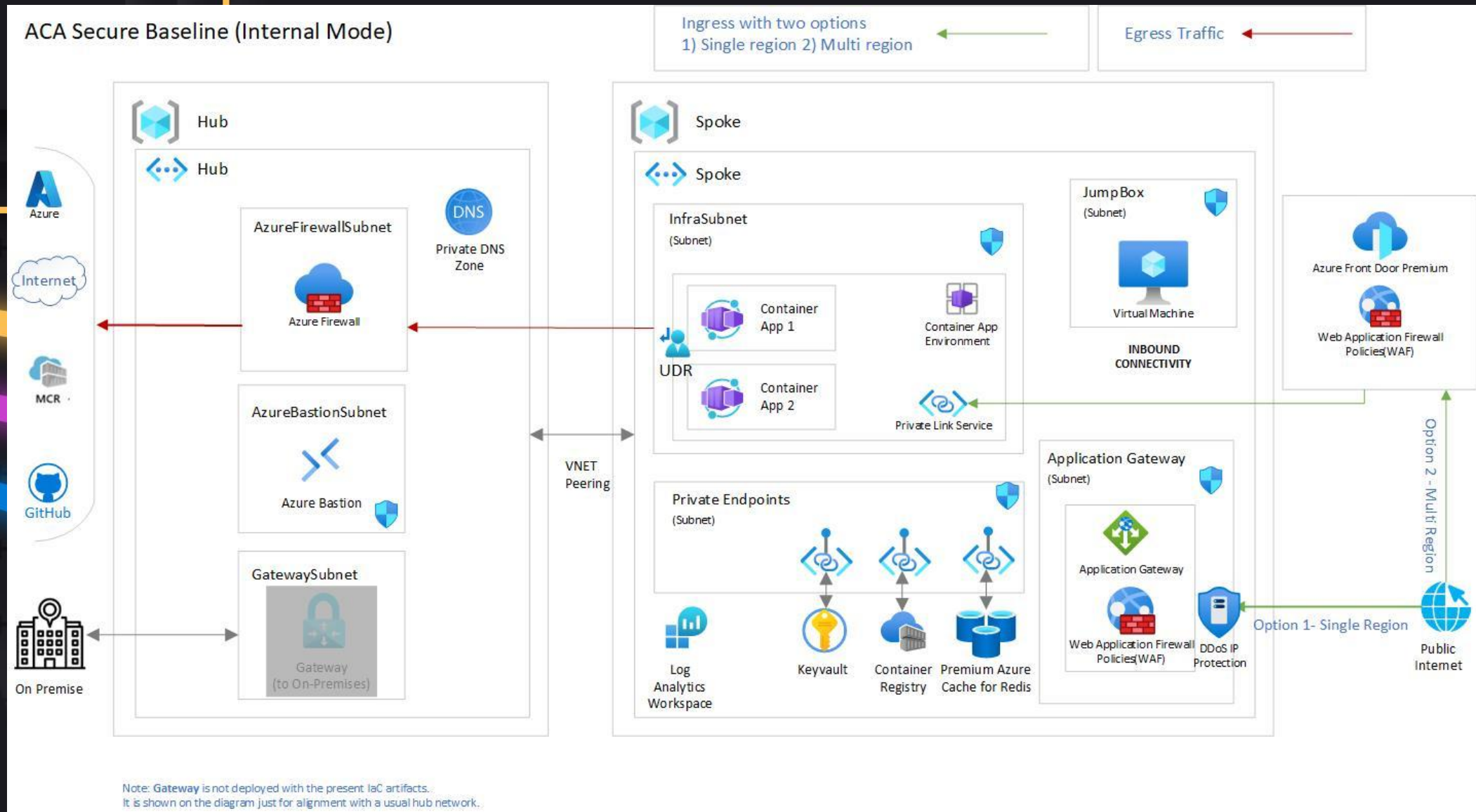
- Workload profiles Supports user defined routes (UDR) and egress through NAT Gateway. The minimum required subnet size is /27.
- Consumption only Doesn't support user defined routes (UDR), egress through NAT Gateway, peering through a remote gateway, or other custom egress. The minimum required subnet size is /23.

# DEMO - ENVIRONMENT

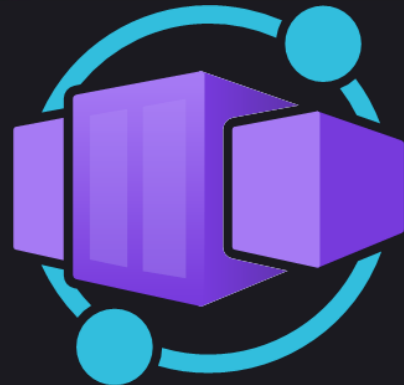




# Azure Container Apps Landing Zone accelerator



# Building and Deploying your first Container App



# Container Apps and Container Registry

## Examine Azure Container Apps containers and containers registries

Azure Container Apps manages the details of Kubernetes and container orchestration for you. Containers in Azure Container Apps can use the runtime, programming language, or development stack of your choice.

Azure Container Apps supports:

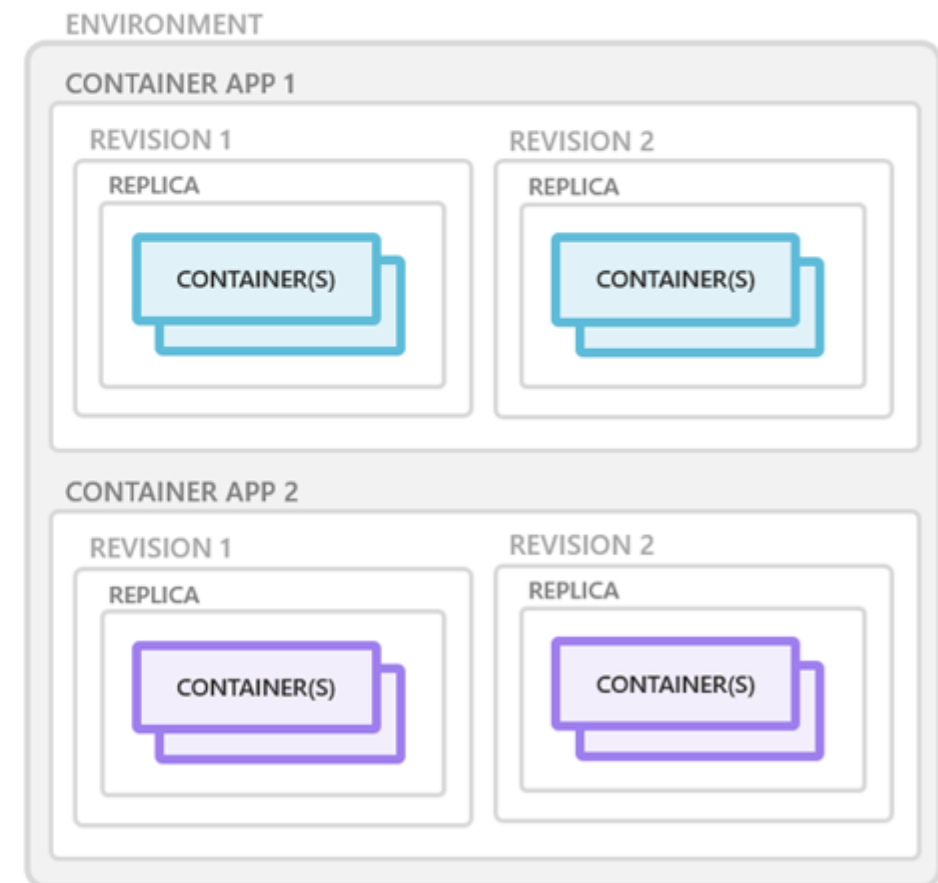
- Any Linux-based x86-64 (linux/amd64) container image with no required base image
- Containers from any public or private container registry
- Sidecar and init containers



**Containers** for an Azure Container App are grouped together in pods inside revision snapshots.

Features include:

- Changes to the template configuration section trigger a new container app revision.
- If a container crashes, it automatically restarts.



# Create Container Apps

## Create a container app and container app environment (1/3)

Container Apps can be found in the Azure Marketplace under **Containers**, or by entering **container app** in the search text box.

To specify a container image when creating a container app instance, you need an existing registry resource.

Containers can be configured after creating the container app instance. Containers can be sourced from registries such as Azure Container Registry or Docker Hub.

Every container app must be part of a container app environment. An environment provides an isolated network for one or more container apps, making it possible for them to easily invoke each other.

### Create Container App ...

Basics Container Bindings Tags Review + create

Azure Container Apps are containerized apps that scale on demand without requiring you to manage cloud infrastructure. You'll need a container and an environment for your first app. Select existing resources, or create them now. [Learn more](#)

#### Project details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \*

Visual Studio Enterprise

Resource group \*

RG1

[Create new](#)

Container app name \*

#### Container Apps Environment

The environment is a secure boundary around one or more container apps that can communicate with each other and share a virtual network, logging, and Dapr configuration. [Container Apps Pricing](#)

Show environments in all regions ⓘ

☐

Region \*

Central US

Container Apps Environment \*

managedEnvironment-RG1-8920 (RG1)

[Create new](#)

# Create Container Apps

## Create a container app and container app environment (2/3)

The **Basics** tab on the Create Container Apps Environment page is used to name the environment and provides the option to enable zone redundancy.

The **Workload profiles** tab on the Create Container Apps Environment page can be used to add dedicated workload profiles.

The **Monitoring** tab on the Create Container Apps Environment page provides option for specifying monitoring and logging preferences.

The **Networking** tab on the Create Container Apps Environment page provides the option of selecting your own virtual network.

### Create Container Apps Environment ...

#### Basics

#### Workload profiles

#### Monitoring

#### Networking

The environment is a secure boundary around one or more container apps that can communicate with each other and share a virtual network, logging, and Dapr configuration. [Learn more](#)

#### Environment details

Environment name \*

Environment name

Environment type \*

- ☒ **Workload Profiles:** Supports the Consumption and Dedicated plans. Run serverless apps with support for scale-to-zero and pay only for resources your apps use. Optionally, run apps with customized hardware and increased cost predictability using Dedicated workload profiles.
- ☐ **Consumption only:** Supports the Consumption plan. Run serverless apps with support for scale-to-zero and pay only for resources your apps use.

#### Zone redundancy

A Container App Environment can be deployed as a zone redundant service in the regions that support it. This is a deployment time only decision. You can't make Container App Environment zone redundant after it has been deployed. [Learn more](#)

Zone redundancy \*

- ☒ **Disabled:** Your Container App Environment and the apps in it will not be zone redundant.
- ☐ **Enabled:** Your Container App Environment and the apps in it will be zone redundant. This requires vNet integration.



# Create Container Apps

## Create a container app and container app environment (3/3)

The **Networking** tab provides the following options:

- Use your own virtual network: Select Yes to use your own virtual network.
- Virtual network: Select your virtual network from the dropdown. Only networks located in the same region as the container apps environment are listed.
- Infrastructure subnet: Select the subnet that you've prepared for your container app.

When using the Consumption only Architecture for Azure Container Apps, you must use a dedicated subnet with a CIDR range of /23 or larger.

### Create Container Apps Environment ...

Basics Workload profiles Monitoring Networking

Selecting your own virtual network allows you to connect your application to other Azure resources or on-premises systems through the same network. [Learn more](#)

#### Virtual network

Use your own virtual network \*

☐ No ☒ Yes

Virtual network \*


[Create new](#)

Infrastructure subnet \* ⓘ

Virtual IP

☐ Internal: The endpoint is an internal load balancer  
☒ External: Exposes the hosted apps on an internet-accessible IP address

Infrastructure resource group

 Azure recommends using the default name unless it does not conform with your company's Azure Policies. [Learn more](#)

# Managed identities in Azure Container Apps

## Examine managed identities in Azure Container Apps

A managed identity from Microsoft Entra allows your container app to access other Azure AD-protected resources.

### Managed identities provide these benefits:

- Your app connects to resources with the managed identity. You don't need to manage credentials in your container app.
- You can use role-based access control to grant specific permissions to a managed identity.
- System-assigned identities are automatically created and managed. They're deleted when your container app is deleted.
- You can add and delete user-assigned identities and assign them to multiple resources. They're independent of your container app's life cycle.
- You can use managed identity to authenticate with a private Azure Container Registry without a username and password to pull containers for your Container App.
- You can use managed identity to create connections for Dapr-enabled applications via Dapr components.

### Common use cases:

System-assigned identities are best for workloads that:

- are contained within a single resource
- need independent identities

User-assigned identities are ideal for workloads that:

- run on multiple resources and can share a single identity
- need pre-authorization to a secure resource

# Ingress in Azure Container Apps

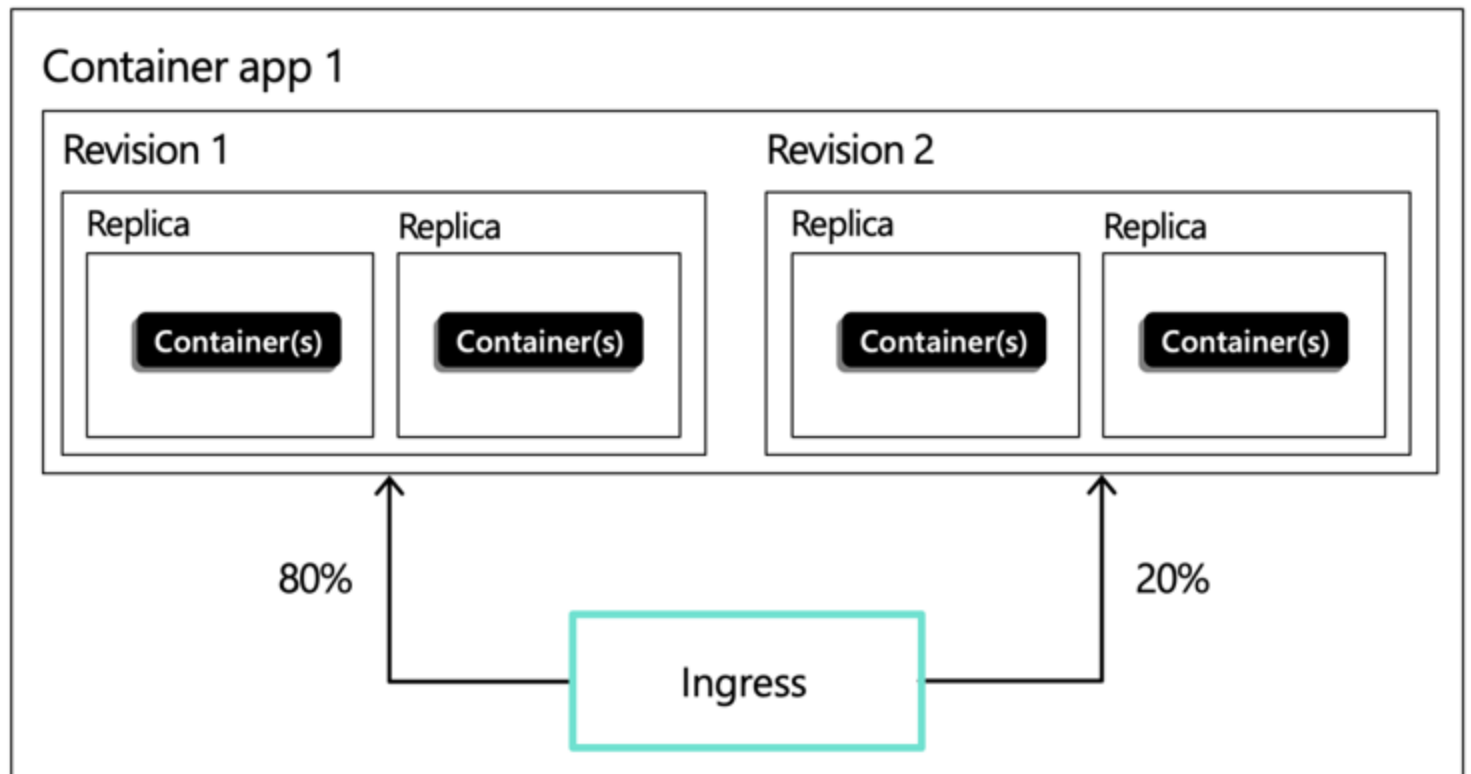
## Examine ingress in Azure Container Apps

Azure Container Apps allows you to expose your container app to the public web, your virtual network (VNET), and other container apps within your environment by enabling ingress.

Ingress supports:

- External and internal ingress
- HTTP and TCP ingress types
- Domain names
- IP restrictions
- Authentication
- Traffic splitting between revisions
- Session affinity

Environment (virtual network boundary)



# Secrets in Azure Container Apps

## Examine the management of secrets in Azure Container Apps

Azure Container Apps allows your application to securely store sensitive configuration values. Once secrets are defined at the application level, secured values are available to revisions in your container apps. Additionally, you can reference secured values inside scale rules.

### Defining secrets:

Secrets are defined as a set of name/value pairs. The value of each secret is specified directly or as a reference to a secret stored in Azure Key Vault.

To define secrets directly using the portal:

1. Open your container app in the Azure portal.
2. In the left-side menu under Settings, select **Secrets**.
3. Select **Add**.
4. On the *Add secret context* page, enter the following:  
*Name*: Enter the name of the secret.  
*Type*: Select **Container Apps Secret**.  
*Value*: Enter the value of the secret.
5. Select **Add**.

### Reference secret from Key Vault:

When you define a secret, you create a reference to a secret stored in Azure Key Vault. Container Apps automatically retrieves the secret value from Key Vault and makes it available as a secret in your container app.

To reference a secret from Key Vault, you must first enable a managed identity in your container app and grant the identity access to the Key Vault secrets.

To grant access to Key Vault secrets, create an access policy in Key Vault for the managed identity you created. Enable the "Get" secret permission on this policy.

# Storage mounts in Azure Container Apps

## Examine the storage mounts in Azure Container Apps

A container app has access to three different types of storage: container file system, ephemeral volume, and Azure Files volume. A single app can take advantage of more than one type of storage if necessary.

### Container file system

Container file system storage has the following characteristics:

- The storage is temporary and disappears when the container is shut down or restarted.
- Files written to this storage are only visible to processes running in the current container.
- There are no capacity guarantees. The available storage depends on the amount of disk space available in the container.

### Ephemeral volume

Ephemeral storage has the following characteristics:

- Ephemeral storage is scoped to a single replica.
- Files are persisted for the lifetime of the replica. If a container in a replica restarts, the files in the volume remain.
- Any containers in the replica can mount the same volume.
- A container can mount multiple ephemeral volumes.
- The available storage depends on the total number of vCPUs allocated to the replica.

### Azure Files volume

Azure Files storage has the following characteristics:

- Files written under the mount location are persisted to the file share and are available via the mount location.
- Multiple containers can mount the same file share, including ones that are in another replica, revision, or container app.
- All containers that mount the share can access files written by any other container or method.
- More than one Azure Files volume can be mounted in a single container.



# Cloud service connections in Azure Container Apps

## Examine cloud service connections in Azure Container Apps

Azure Container Apps provides a Service Connector feature that manages the configuration of the network settings and connection information between different services.

Service Connector features:

- Connects Azure services using a single Azure CLI command or in a few steps in the Azure portal.
- Supports an increasing number of databases, storage, real-time services, state, and secret stores.
- Configures network settings, authentication, and manages connection environment variables or properties.
- Validates connections and provides suggestions to fix faulty connections.

Dashboard > aca-apl2003

aca-apl2003 | Service Connector (preview) ☆ ...

Container App

Search

Application

Revisions

Containers

Scale and replicas

Settings

Authentication

Secrets

Ingress

Continuous deployment

Custom domains

Dapr

Identity

Service Connector (preview)

CORS

Locks

Monitoring

Alerts

Metrics

Using Service Connector will register resource provider 'Microsoft.ServiceLinker' to your Subscription.

+ Create Edit Refresh Validate Sample code Delete

**Safely and Quickly Connect your**

Don't like spending time figuring out what the best database service is? Use Service Connector to make it quick.

**Connect your Services**

Don't get lost trying to figure out what authentication method to use or where to store your secrets safely. Don't worry about determining the right firewall settings. Service Connector makes it a breeze to safely connect your app with your services.

**Access from inside your app**

No need to store connection configuration in your code. Service Connector makes them available for you via environment variables, ready to be used by your application.

**Create connection**

Basics Authentication Networking Review + Create

Select the service instance and client type.

Container \* ⓘ

simple-hello-world-container

Service type \* ⓘ

Service Bus

Connection name \* ⓘ

servicebus\_2e6c4

Visual Studio Enterprise

Namespace \* ⓘ

sb-apl2003

Create new

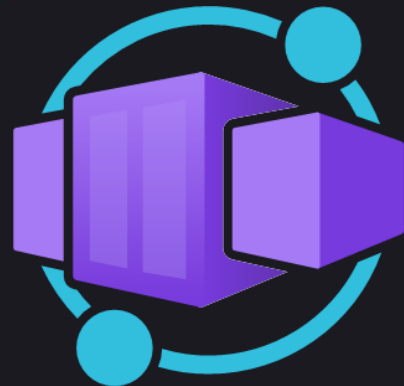
Client type \* ⓘ

.NET

Connect to

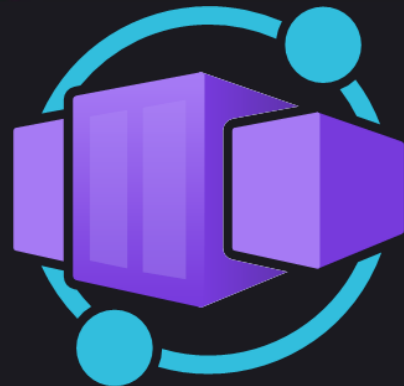
# DEMO

## Build, Push, Deploy





# CI/CD Integration



# Continuous Deployment options for Container Apps

## Review continuous deployment options for container apps

Continuous deployment to a container apps instance is implemented by creating revisions (an app versioning process).

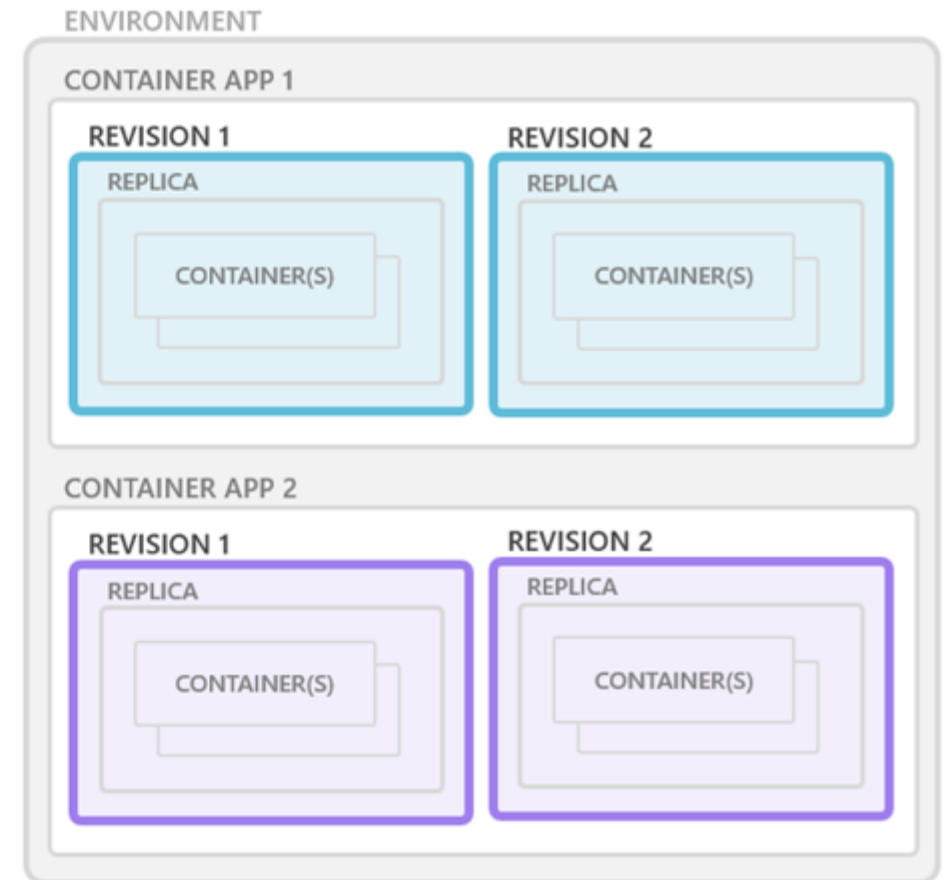
Azure Container Apps provides support for continuous deployment using either GitHub Actions or Azure Pipelines.

As commits are pushed to your repository, a workflow (GitHub) or pipeline (Azure Pipelines) is triggered which updates the container image in the container registry.

Azure Container Apps creates a new revision based on the updated container image.



**Revisions** are immutable snapshots of a container app.



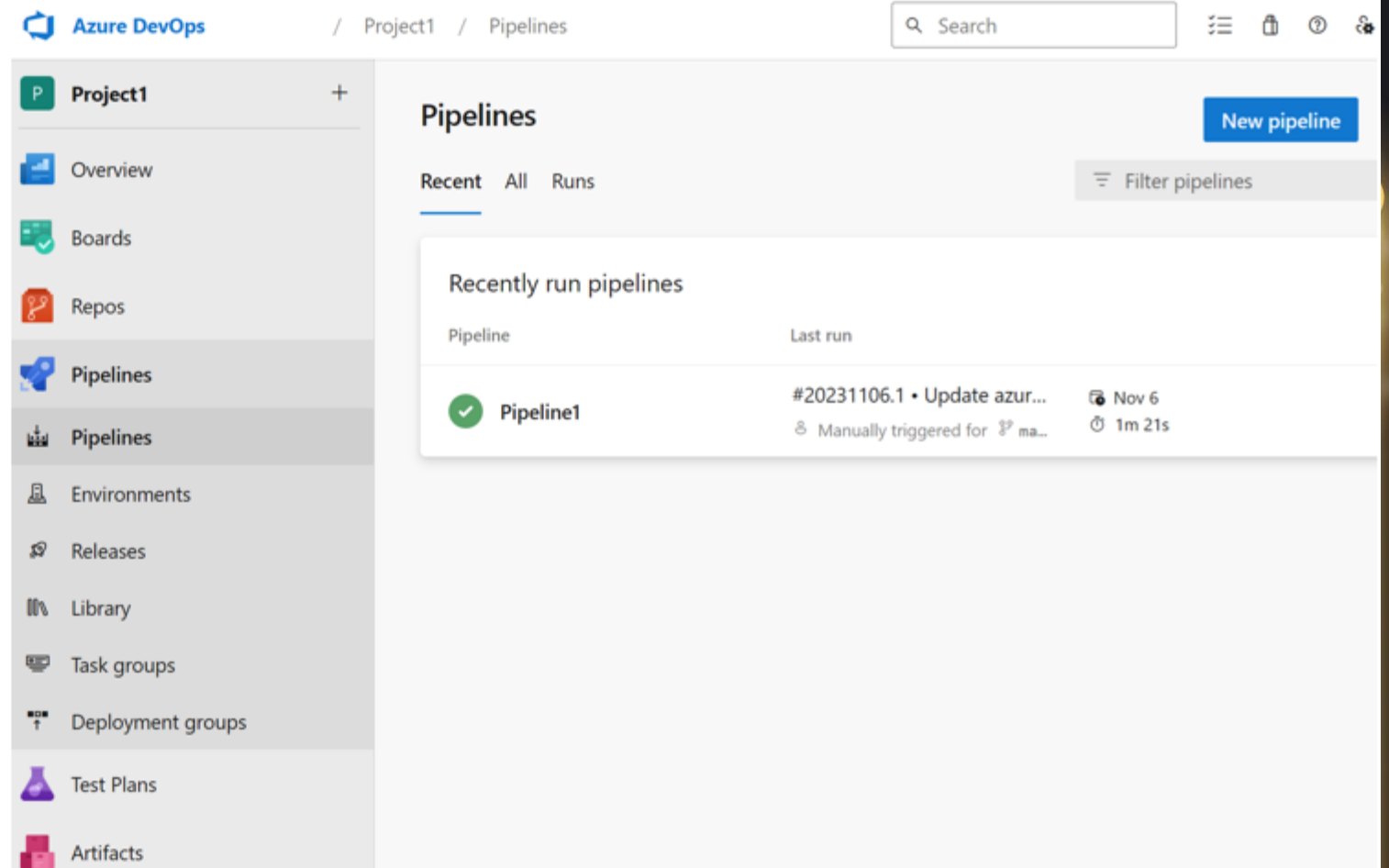
# Azure DevOps and Azure Pipelines

Azure DevOps supports collaboration between developers, project managers, and other contributors as they work together to develop software.


## Azure Pipelines:

Azure Pipelines automatically builds and tests code projects. It supports all major languages, and project types and combines continuous integration, continuous delivery, and continuous testing to build, test, and deliver your code to any destination.

Azure Pipelines requires your source code to be in a version control system. Azure DevOps supports two forms of version control - Git and Azure Repos.



The screenshot displays the Azure DevOps web interface for a project named 'Project1'. The left sidebar contains a navigation menu with options: Overview, Boards, Repos, Pipelines (selected), Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main content area is titled 'Pipelines' and includes a 'New pipeline' button. Below this, there are tabs for 'Recent', 'All', and 'Runs', with 'Recent' being the active tab. A 'Filter pipelines' button is also present. The 'Recently run pipelines' section shows a table with one entry:

Pipeline	Last run
 Pipeline1	#20231106.1 • Update azur... Manually triggered for ma... Nov 6 1m 21s

# Azure pipeline configuration and deployment tasks

Azure Pipelines can be used to automate build, test, and deployment processes.

## Azure Pipeline tasks:

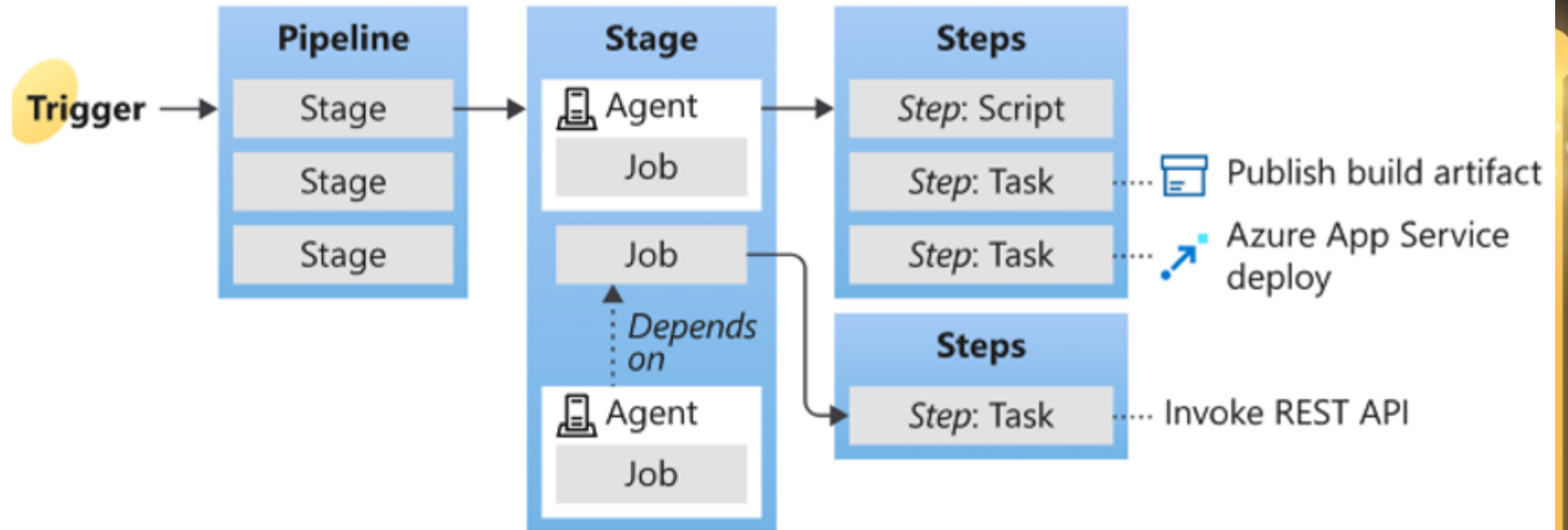
Tasks are used to perform an action in a pipeline. A task that build or deploys your code requires an agent.

## Built-in tasks and templates:

The Azure Pipelines service provides some built-in tasks to enable fundamental build and deployment scenarios.

## Built-in task for Azure Container Apps:

The Azure Container Apps Deploy task is an Azure DevOps Task that can be used to build and deploy Azure Container Apps.



# Examine agents and agent pools for pipelines

To build your code or deploy your software using Azure Pipelines, you need at least one agent. There are three types of agents to choose from: Microsoft-hosted agents, Self-hosted agents, and Azure Virtual Machine Scale Set agents.

## Microsoft-hosted agents

If your pipelines are in Azure Pipelines, then you've got a convenient option to run your jobs using a Microsoft-hosted agent.

With Microsoft-hosted agents, maintenance and upgrades are taken care of for you.

Each time you run a pipeline; you get a fresh virtual machine for each job in the pipeline.

## Self-hosted agents

An agent that you set up and manage on your own to run jobs is a self-hosted agent.

Self-hosted agents give you more control to install dependent software needed for your builds and deployments.

With self-hosted agents, machine-level caches and configuration persist from run to run, which can boost speed.

## Azure Virtual Machine Scale Set agents

A form of self-hosted agents, using Azure Virtual Machine Scale Sets, that can be auto-scaled to meet demands.

# Examine agents and agent pools for pipelines

Self-hosted agents can be used to build and deploy Windows, Azure, and other Visual Studio solutions.

## Step 1: Obtain a personal access token from DevOps

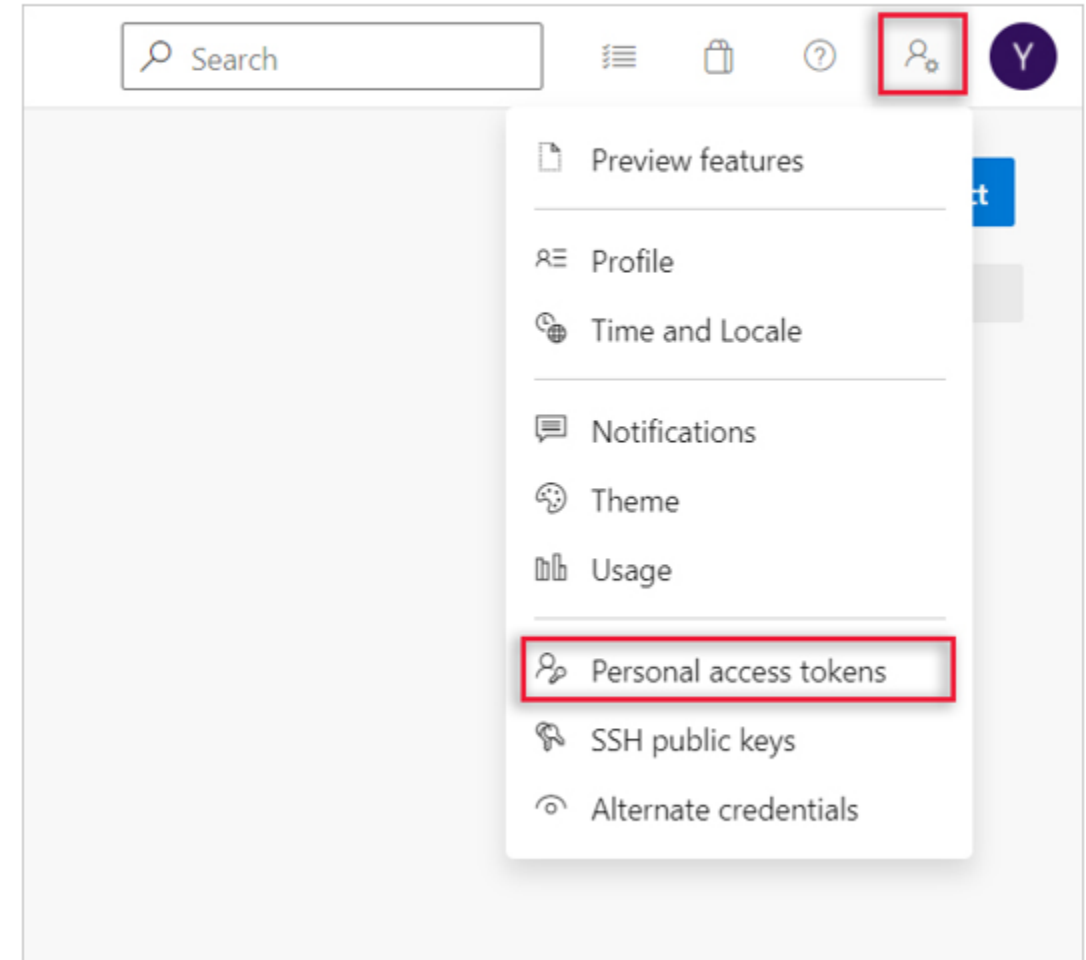
- From the home page of your DevOps organization, open your user settings, and then select **Personal access tokens**.
- To create a personal access token, select + **New Token**.
- Select the **Agent Pools (read, manage)** and **Deployment group (read, manage)** scopes, and then copy the token.

## Step 2: Download the agent

- Sign into DevOps as the Azure DevOps organization owner.
- Select **Organization settings**, and then select **Agent pools**.
- On the **Get the agent** dialog box, select **Windows**, select the processor architecture, and then select **Download**.

## Step 3: Configure the agent

- Create a local folder for the agent, for example: C:\agents
- Unpack the agent zip file into the directory you created.
- Open PowerShell as an Administrator, then run the following PowerShell command in the agent's folder: **.\config**
- Follow the prompts to complete the configuration.





# Examine environment and secret variables for pipelines

## Examine environment and secret variables for pipelines

Variables give you a convenient way to get key bits of data into various parts of the pipeline. The most common use of variables is to define a value that you can then use in your pipeline. All variables are strings and are mutable.

### Variable types

**User-defined variables:** When you define a variable, you can use different syntaxes (macro, template expression, or runtime) and the syntax you select determines where in the pipeline your variable renders.

**System variables:** Azure Pipelines provides system variables with predefined, read-only values. System variables get set with their current value when you run the pipeline. As a pipeline author or end user, you can change the value of a system variable before the pipeline runs.

**Environment variables:** Environment variables are specific to the operating system you're using. They're injected into a pipeline in platform-specific ways.

### Set secret variables:

Secret variables are encrypted variables that you can use in pipelines without exposing their value. Secret variables can be used for private information like passwords, IDs, and other identifying data that you wouldn't want to have exposed in a pipeline.

The recommended ways to set secret variables are in the UI, in a variable group, and in a variable group from Azure Key Vault.

Secret variables set in the pipeline settings UI for a pipeline are scoped to the pipeline where they're set. You can use variable groups to share secret variables across pipelines.



# DEMO - ADO



[illegible]