Microsoft

Microsoft Azure Virtual Training Day: Cloud Native Apps

Microsoft

Build optimized Cloud Applications

Module objectives:

- Describe the fundamental structure of a cloud-native app
- Identify situations where you should build a cloud-native app

Introduction

Introduction

- Cloud-native apps represent a modern approach to app development, where software systems are designed with cloud technologies in mind.
- Focus on **architectural modularity**, rather than monolithic, all-in-one applications

Scenario: Smart fridges, smarter service, at scale

- Suppose you work Adatum, a manufacturer of home appliances, where you lead a small development team and you've been tasked with building an app for smart fridges.
- Start by creating a small inventory management app for the fridges
- Later, add functionality to the app
- It’s the nature of cloud-native apps to have loosely coupled functionality, so we can be more agile in our design and don't need to predict future requirements.

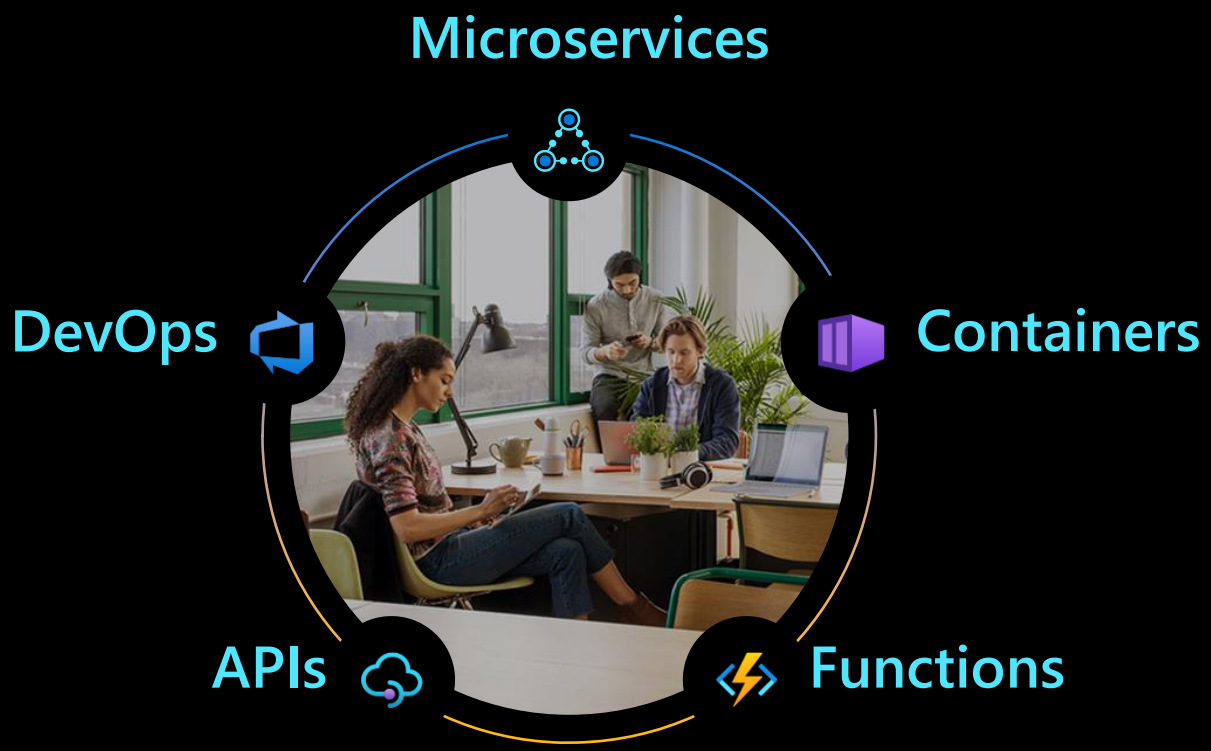
What are cloud-native apps?


We're living in
a future defined by
accelerated innovation
A new class of applications is being built

The app of the future is cloud native

What is cloud native?

Package application code and dependencies in containers, deploy as microservices and manage them using DevOps processes and tools



Using containers with cloud-native apps

Using containers with cloud-native apps

- **Containers,** loosely isolated environments that can run software packages, are usually key components of cloud-native apps.
- Containers can scale out more cost effectively and nimbly than virtual machines.

Manage containers easily with a Kubernetes service

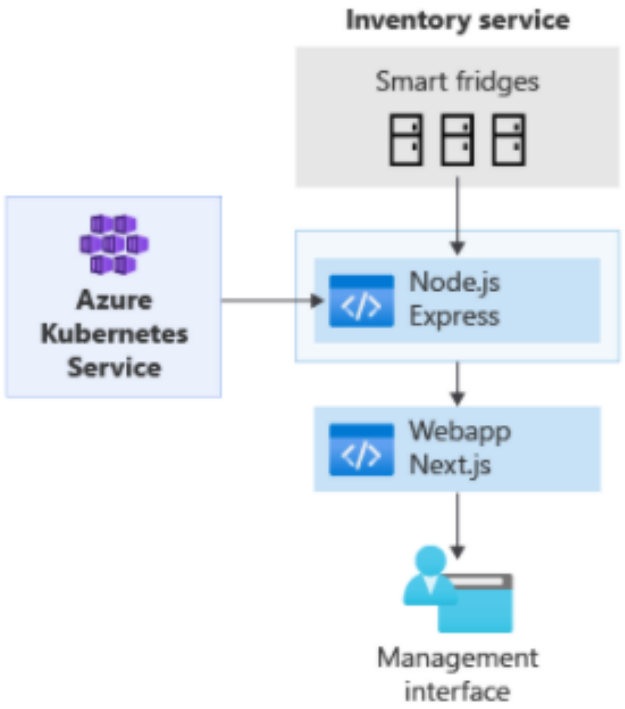
- Kubernetes is a technology that manages multiple containers for you.
- Key benefits of Kubernetes: **self-healing**, **load balancing**, and **simplified security configuration management**.

Designing a cloud-native app

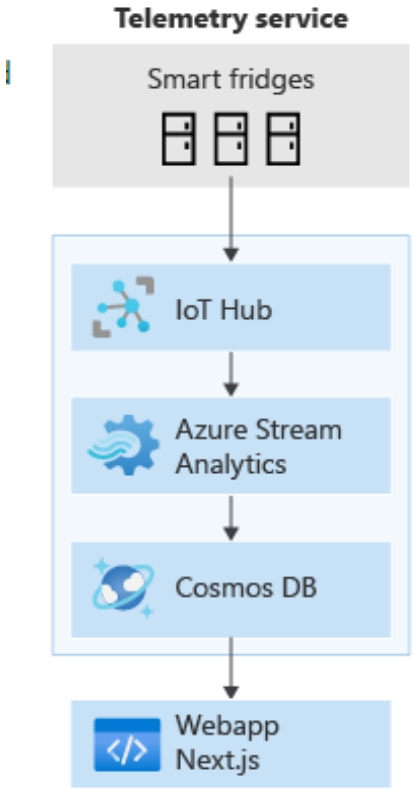
Designing a cloud-native app

Because cloud-native apps are made up of the components of your choice, you can easily architect a solution that uses technologies you're comfortable with.

Scenario: Architecting a cloud-native solution for Adatum Starting small



Scenario: Architecting a cloud-native solution for Adatum Growing our application

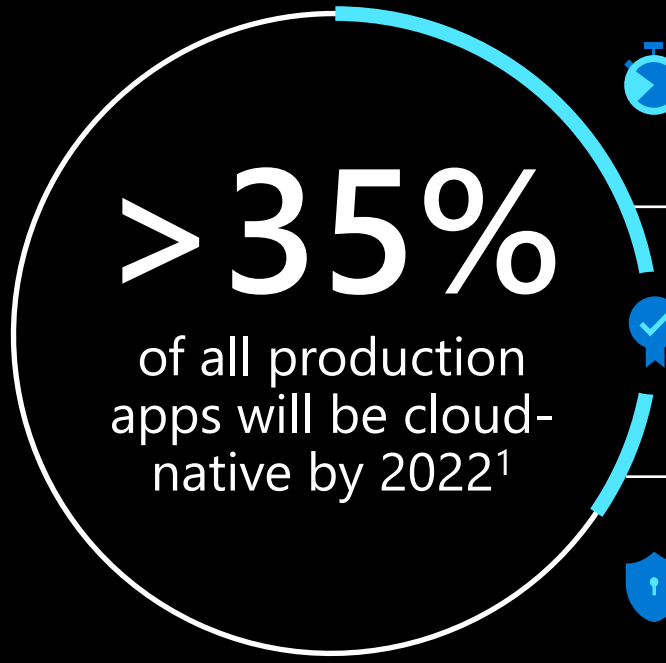


When to use cloud-native apps

Challenges for modern developers

[1]IDC. "IDC FutureScape: Worldwide IT Industry 2019 Predictions," October 2018. [2] Forrester. "The Forrester New Wave™: Public Cloud Enterprise Container Platforms," July 29, 2019.

Cloud native applications



Agility

Strongest developer experience[2]

Most complete tool chain from source to production

Opensource Functions runtime

Reliability

Fully managed database with >99.999-percent HA

Single-digit millisecond latencies on reads and writes

Available in more regions than any other cloud providers

Security

$1B investment every year in security

>90 compliance certifications

Out of the box integration with Azure Policy, Active Directory, and Security Center

Summary

Summary

In this module, you learned about cloud-native apps.

- Describe the fundamental structure of a Cloud Native App
- Identify situations where you should build a Cloud Native App

Microsoft

Manage your Applications with Azure Kubernetes Service

Module objectives:

- Create a Kubernetes AKS cluster
- Run a container in Kubernetes
- Connect the container to a webapp

Introduction

Introduction

- When you're creating cloud-native applications, you can enjoy the many benefits of using containers, which are a great way to bundle and run applications.

- Many cloud-native architectures turn to Kubernetes to deploy and manage containers.

Scenario: Connecting fridges, at scale

Suppose you work for Adatum Corporation: a manufacturer of home appliances, such as refrigerators. You've been tasked with building an app for smart fridges.

Running containers with Kubernetes

Running containers with Kubernetes

- Containers are a virtualization technology.
- Containers don’t have their own internal operating system.

Using containers in the cloud

Azure Container Registry provides storage for container images in the cloud. Container Registry provides security benefits, such as:

- Building container images
- Authentication for who can see and use your images
- You can sign images to increase trust and reduce the chances of an image becoming accidentally-or intentionally-corrupted or otherwise infected
- All images stored in a container registry are encrypted at rest

Azure Kubernetes Service (AKS) does the heavy lifting

AKS handles Kubernetes for you, by deploying, managing, and scaling Kubernetes clusters.

Demo: Create an AKS cluster

Developing with containers and AKS

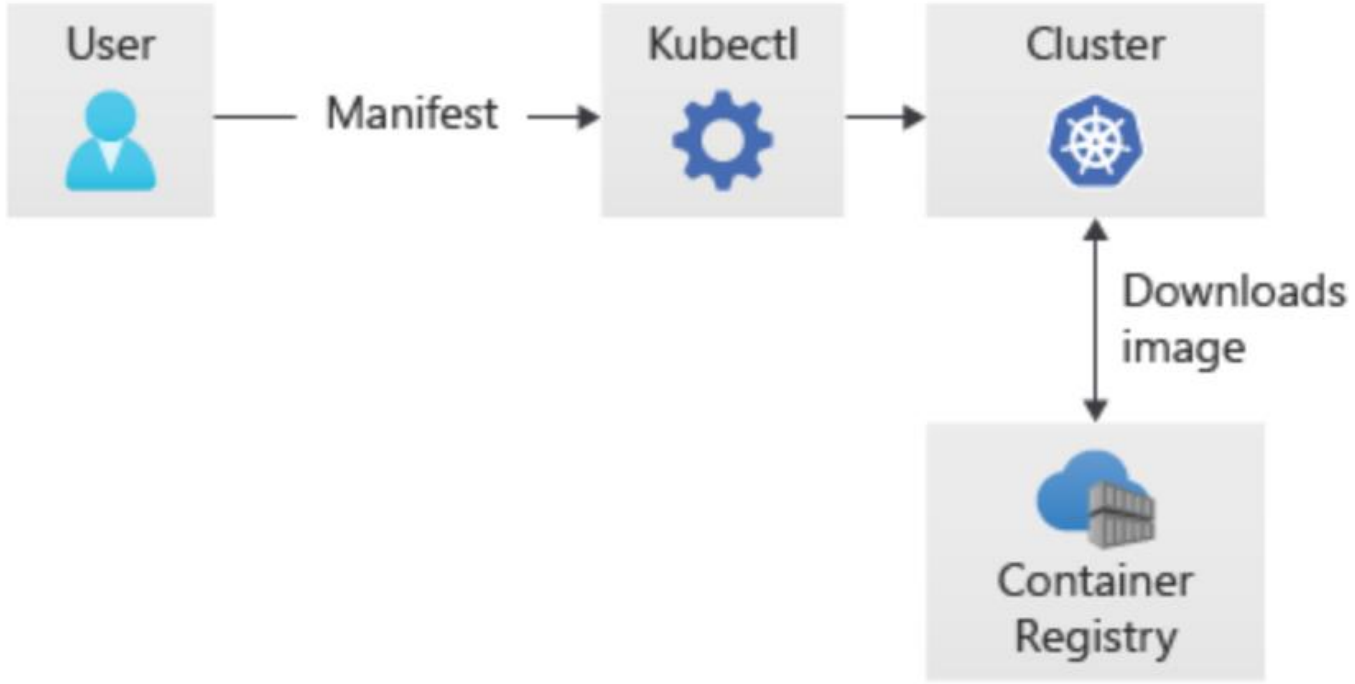
Open-source benefits

AKS can use either **kubectl**, the standard Kubernetes command-line tool. With kubectl and AKS, you can also leverage other open-source tools e.g., Argo CD.

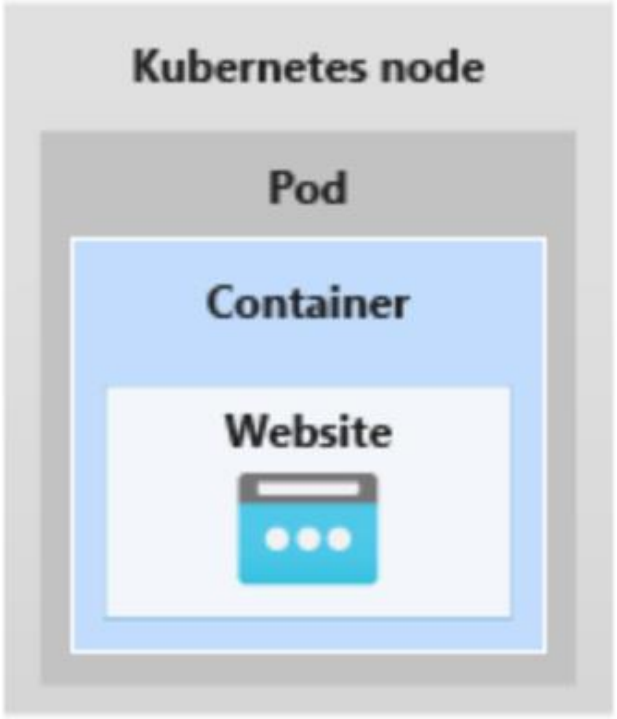
Deploying to a cluster

We can use kubectl to deploy a container from our container registry to the Kubernetes cluster.

Creating a deployment manifest



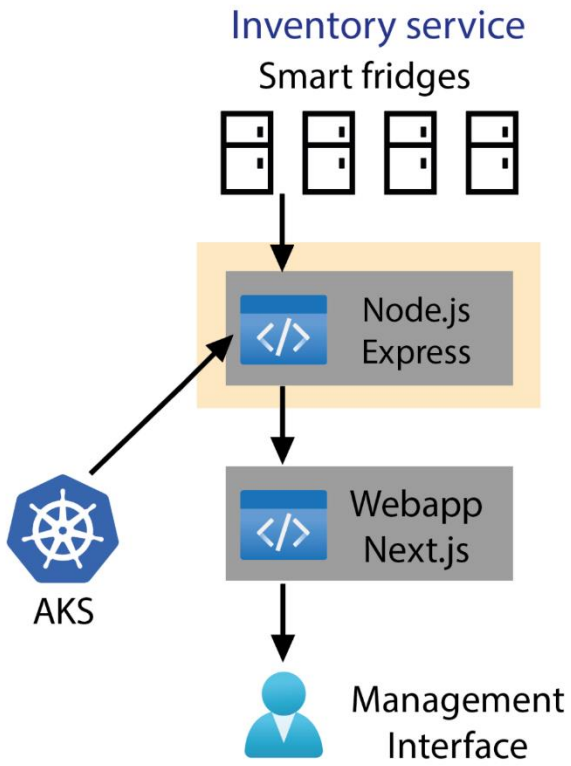
Running a container image in Kubernetes



Kubernetes health checks

One of the key benefits of Kubernetes is the ability to restore applications to the exact instance that had been tested and saved, otherwise known as **self-healing**.

What our container will do



Demo: Set up a development environment with AKS

Connecting cloud-native components

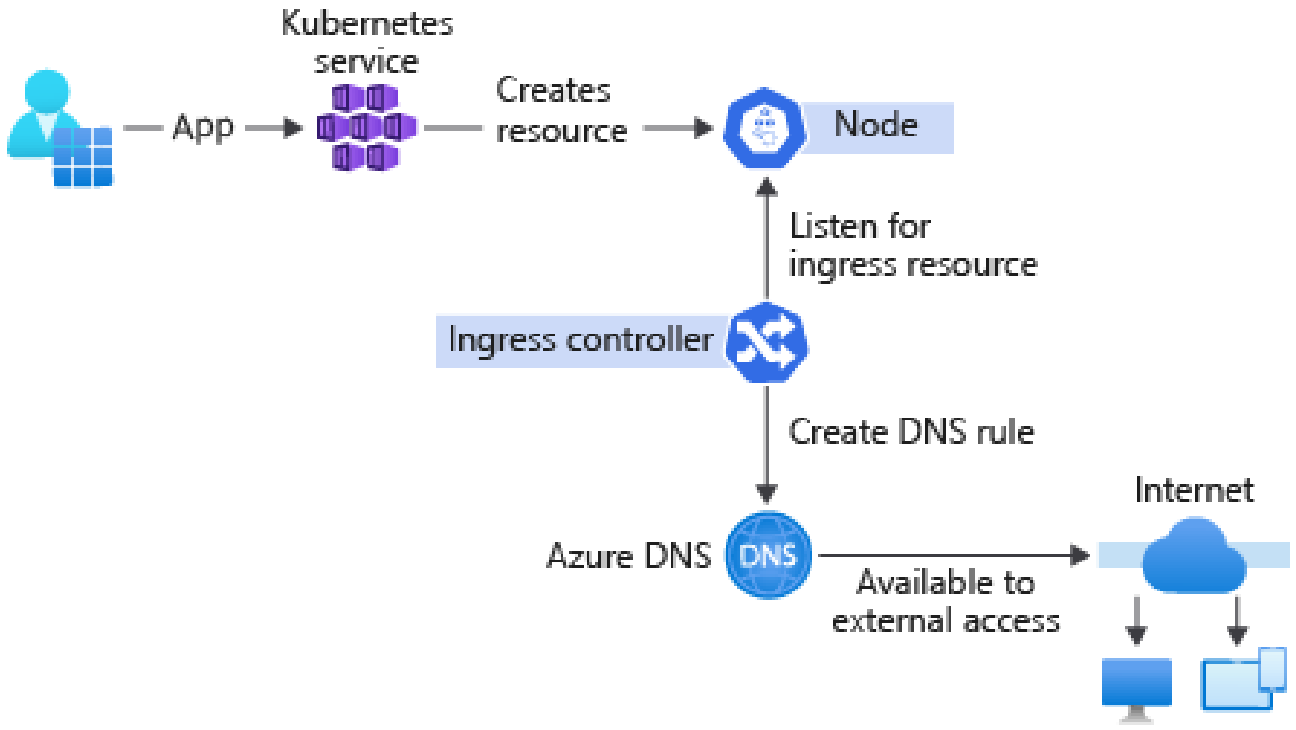
Connecting cloud-native components

In the exercise, we deployed an express.js container image to AKS. To connect the image to the management webapp, we can expose the API by an AKS add-on: **HTTP application routing**. This add-on makes it easy to access applications on the cluster.

Connecting the pieces together

- An AKS cluster blocks all inbound traffic to the cluster to assure network security.
- To expose applications hosted on AKS to the world, you need to create a Kubernetes Service or an Ingress Controller.

Ingress controllers



Reliable rolling deployments

Once you have set up your manifest files, Kubernetes provides a rich feature set for deployment options, such as:

- Canary deployments
- Deploying services in parallel
- Taking only a specific amount of system capacity offline at a time
- Circuit Breakers if a deployment malfunctions

Connecting the smart fridge solution

In our scenario, we've hosted a Node container in AKS to process inventory messages from smart fridges. In order for a management webapp to receive information from the Node container, we have to enable HTTP application routing and create an ingress manifest file.

Demo: Connect cloud-native components

Summary

Summary

After completing this module, you know how to:

- Create a Kubernetes AKS cluster
- Run a container in Kubernetes
- Connect the container to a webapp

Microsoft

Leverage managed databases for cloud-native applications

Module objectives:

- Describe the characteristics and functionality of Azure Cosmos DB.
- Learn the concept of service in the context of cloud-native applications
- Set up a basic service
- Implement Azure Database for PostgreSQL

Define the concept of services

Define the concept of services

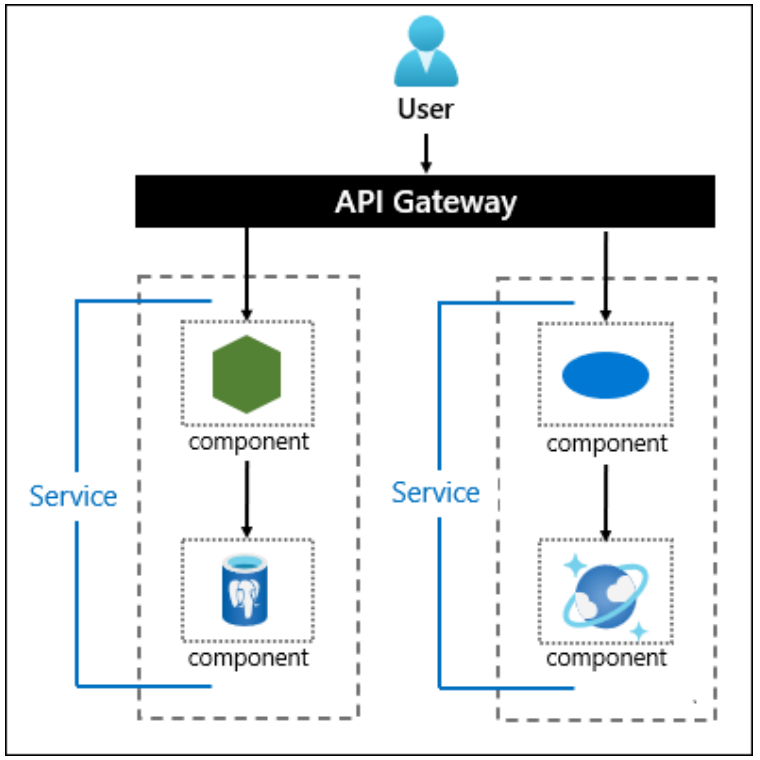
Typically, you transition from the traditional software programming model to cloud-native applications when you need to improve the agility, availability, and resiliency of your workloads.

What is a service?

The term *service* represents a collection of components that collectively deliver specific, workload-oriented functionality to your cloud-native application.

How can services use Azure capabilities?

In the context of cloud-native applications, you can optimize the use of services by using Azure capabilities.



Describe Azure Cosmos DB

What is Cosmos DB?

- Azure Cosmos DB is a fully managed, cloud-native NoSQL database.

What are the advantages of Cosmos DB over relational databases?

- Support for different consistency models
- Built-in replication
- Multiple-region writes
- Configurable conflict resolution mechanism

What is the Cosmos DB resource model?

- To implement Azure Cosmos DB, you need to first create an Azure Cosmos DB account in your Azure subscription.
- The number of resources available to process data within a database or its individual collections depends on the number of available Request Units (RU). Cosmos DB offers 3 modes of RU allocation:
  - Provisioned throughput mode
  - Autoscale mode
  - Serverless mode

What are the benefits and use cases of Cosmos DB in Azure IoT scenarios?

Azure Cosmos DB offers many capabilities that make it particularly suitable for IoT scenarios, including:

- Partitioning
- Time to Live (TTL)
- Change feed
- Performance and resiliency SLAs
- Schema-less databases
- Automatic indexing

Note: A logical partition can't exceed 20 GB in size.

Demonstration: Set Up Azure Cosmos DB

Describe Azure Database for PostgreSQL

What is Azure Database for PostgreSQL?

- A Microsoft-managed implementation of the PostgreSQL Community Edition database engine.
- Available in 3 deployment modes:
  - Single Server
  - Flexible Server
  - Hyperscale

Demo: Set up Azure Database for PostgreSQL

Summary

Summary

After completing this module, you know how to:

- Describe the characteristics and functionality of Azure Cosmos DB.
- Learn the concept of service in the context of cloud-native applications
- Set up a basic service
- Implement Azure Database for PostgreSQL
- Identify cloud-native applications scenarios in which Azure Database for PostgreSQL can provide meaningful benefits.

Microsoft

Build an IoT service for your cloud-native apps by using IoT Central

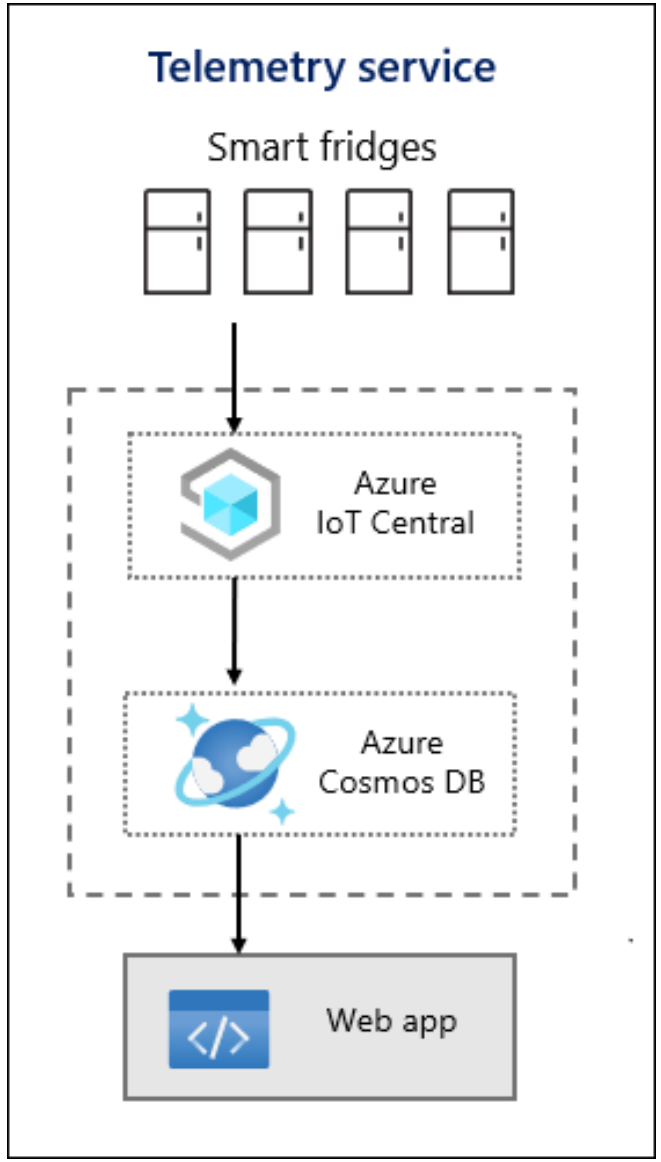
Module objectives:

- Describe the architecture and components of IoT services.
- Integrate Azure data stores with IoT pipelines.
- Implement Azure Cosmos DB for processing telemetry data.

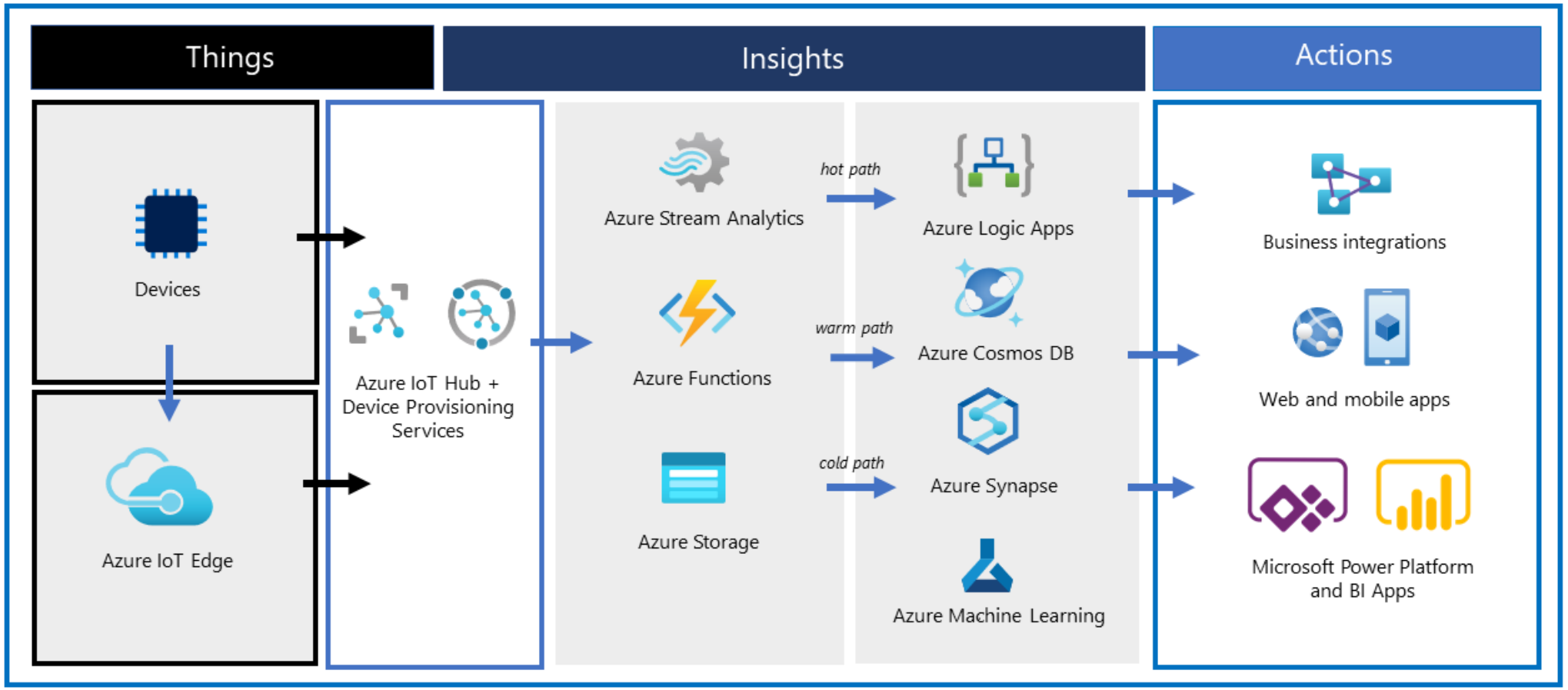
Define IoT service architecture

What is IoT services architecture?

- **Device-side:** This group includes devices that serve primarily as sources of telemetry but might also perform initial telemetry processing and analytics.
- **Cloud-side:** This group includes cloud-based services that are optimized for data collection, persistence, and analytics.



Insights and Actions



IoT services data pipeline

The flow of device-generated data typically consists of several stages, including:

- **Storage.** This stage includes preserving data for a short term or a longer term, relying on technologies such as in-memory caches, temporary queues, databases, and data lakes.
- **Routing.** This stage involves delivering data to one or more storage endpoints, analysis processes, and actions.
- **Analysis.** This stage consists of evaluating and processing data records based on customizable criteria.
- **Action.** This stage involves responding to customizable rules to address a condition indicated by the state or value of collected data.

Azure IoT services and technologies

Microsoft offers a comprehensive portfolio of services that deliver various types of IoT functionality, including:

- **Azure IoT Central.** This service implements a wide range of IoT capabilities, including telemetry collection, processing, analytics, and secure device management.
- **Azure IoT Hub.** This service is optimized for reliable and secure bidirectional communications between IoT devices and cloud services.
- **Azure Time Series Insights.** This highly performing analytics, storage, and visualization service for time series data provides capabilities such as filtering and aggregation.

Integrate data stores with IoT pipelines

What are Azure Cosmos DB-specific design considerations?

When designing Azure Cosmos DB database and container hierarchy, the proper choice of the partition key is essential for ensuring optimal performance and efficiency.

What are data pipelines in IoT scenarios?

A common occurrence in IoT scenarios is the implementation of multiple concurrent data paths, either by partitioning the ingested data stream, or by forwarding data records to multiple pipelines.

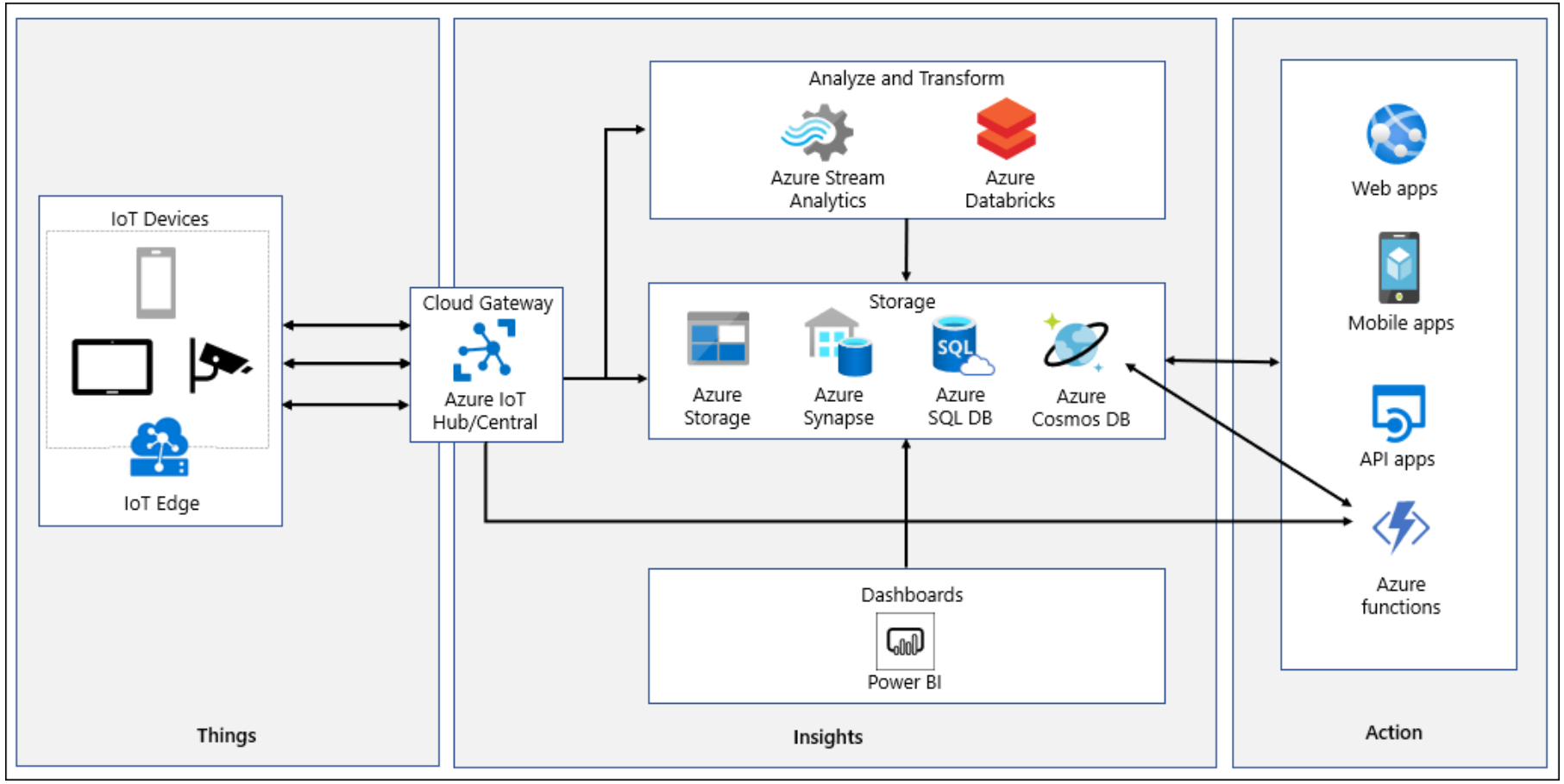
A fast (hot) processing pipeline:

- Performs real-time processing.
- Analyzes data.
- Displays data content.
- Generates short term, time-sensitive information.
- Triggers corresponding actions, such as alerts.
- Subsequently archives the data.

A slow (cold) processing pipeline:

- Performs more complex analysis, potentially combining data from multiple sources and over a longer timeframe.
- Generates artifacts such as reports or machine learning models.

What is the role of Azure services in implementing IoT pipelines?



Demo: Integrate Azure Cosmos DB with the IoT data pipeline

Analyze telemetry data

What are the primary IoT analytics options?

The primary IoT analytics options reflect the data processing principles of the Lambda architecture.

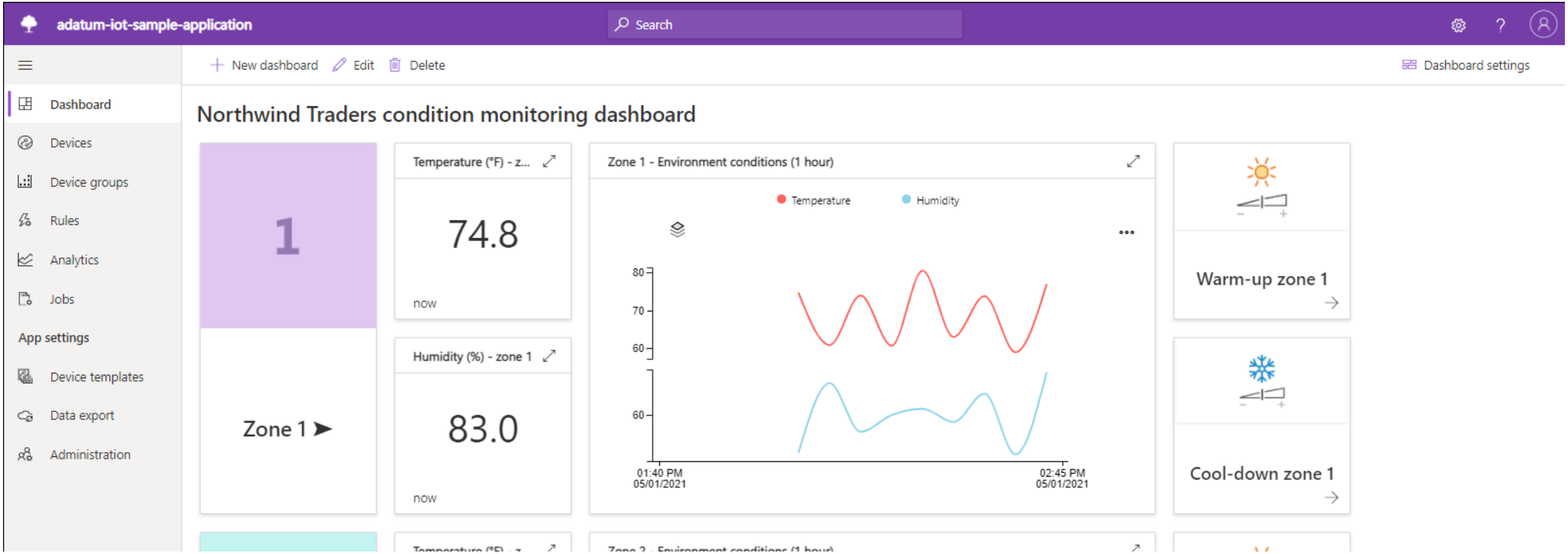
A fast (hot) processing pipeline:

- Performs real-time processing.
- Analyzes data.
- Displays data content.
- Generates short term, time-sensitive information.
- Triggers corresponding actions, such as alerts.
- Subsequently archives the data.

A slow (cold) processing pipeline:

- Performs more complex analysis, potentially combining data from multiple sources and over a longer timeframe.
- Generates artifacts such as reports or machine learning models.

What are the analytics capabilities of Azure IoT Central?



What are the analytics capabilities of Azure Time Series Insights?

Though Azure Time Series Insights are built into Azure IoT Central, it's also available as a separate service, which closely integrates with cloud gateways such as Azure IoT Hub and Azure Event Hub.

What are the analytics capabilities of Azure Stream Analytics?

Azure Stream Analytics is part of the hot data path. It provides real-time analytics and complex event-processing that's optimized for high volumes of streaming data that originates from IoT devices, social media feeds, and applications.

Demo: Integrate Next.js web app with the IoT data pipeline

Microsoft

Deploy and maintain cloud-native apps with GitHub actions and Azure Pipelines

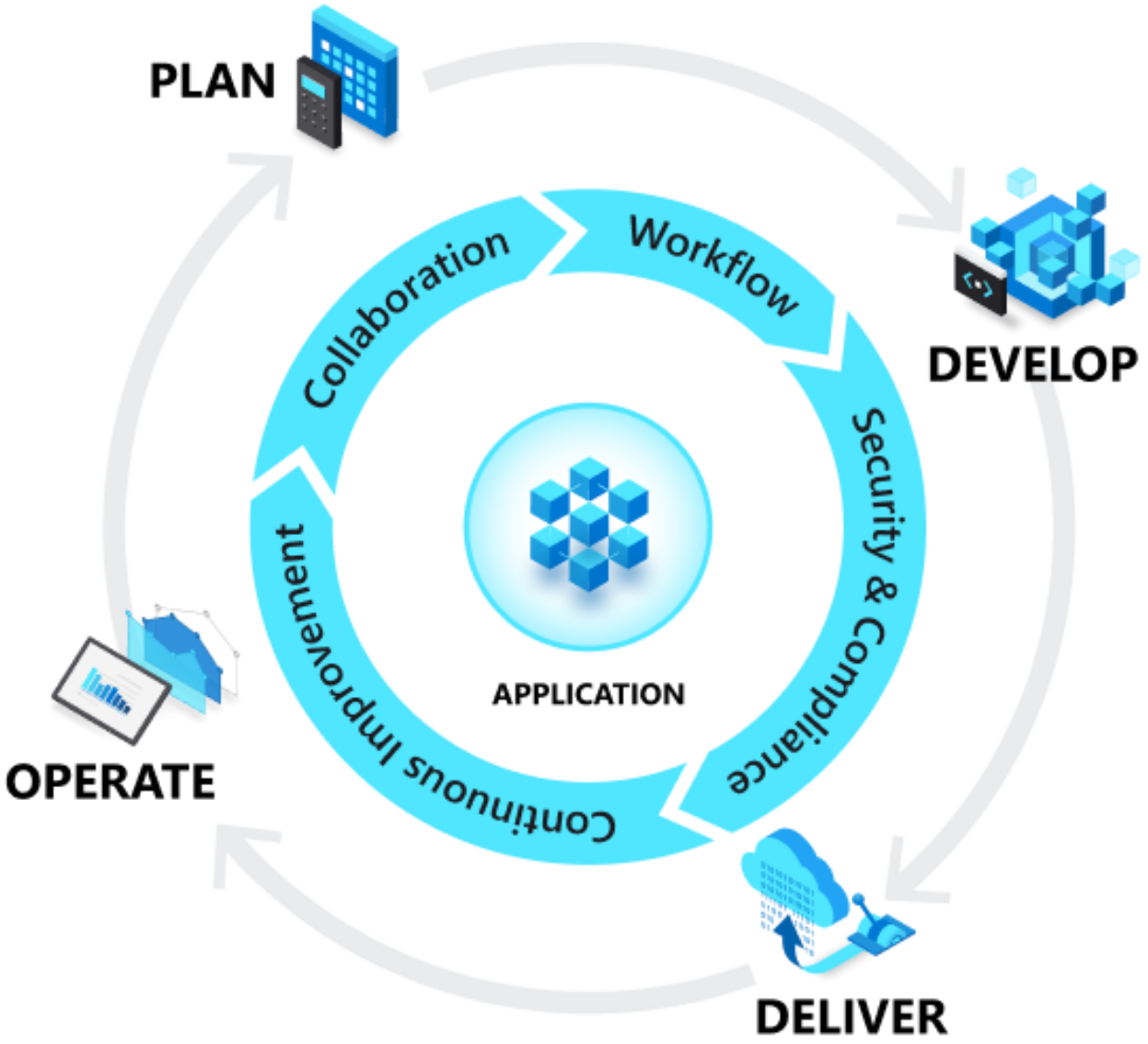
Module objectives:

- Describe the principles of DevOps and their implementation in cloud-native application scenarios.
- Implement DevOps principles by using GitHub repositories, actions, and workflows, as well as Azure Pipelines and Azure Repos.
- Build and deploy infrastructure and applications by using GitHub workflows and Azure Pipelines.

Define the principles and benefits of DevOps

What is DevOps?

A compound of development (Dev) and operations (Ops), DevOps is the union of people, process, and technology to continually provide value to customers.



What are the main DevOps components?

The main DevOps components that you want to focus on include:

- Source control
- Continuous integration (CI)
- Continuous delivery (CD)
- CI/CD pipelines
- Infrastructure as Code (IaC)

What are CI/CD pipelines?

CI is the process of automating the build and testing of code every time an update is committed to the target repository in a version control system.

What is IaC?

Infrastructure as Code (IaC) applies DevOps principles to managing and maintaining services that traditionally are the responsibility of infrastructure and platform teams within an IT organization.

- **Declarative code**: defines what the code should accomplish, not how to achieve the result.
- **Imperative code**: defines both what the program should accomplish and how to achieve the result.

What are the benefits of GitHub Actions?

GitHub Actions provide task automation and workflow functionality.

GitHub Actions consist of the following components:

- **Workflow**. An automated process that implements the pipeline.
- **Runner**. A server that provides compute resources for running a workflow.
- **Event**. An activity that triggers a workflow.
- **Job**. A group of steps that execute on a runner.
- **Step**. A task that can execute one or more actions.
- **Action**. A standalone command that delivers a desired outcome.

Demo: Implement Infrastructure as Code by using GitHub Actions

Deploy and maintain cloud-native applications by using GitHub Actions

How do cloud-native applications benefit from DevOps?

DevOps practices facilitate implementing cloud-native applications. They closely align with Twelve-Factor App guidelines that serve as the foundation for building cloud-native apps. In particular, they help address the following guidelines:

- **Code Base**: A single code base for each microservice that's stored in its own repository and tracked with version control.
- **Build, Release, Run**: Each release must enforce a strict separation across the build, release, and run stages.

How to implement CI/CD by using GitHub Actions

Cloud-native applications are particularly suitable for containerization and container orchestration, which further enhances their agility.

GitHub Actions also include support for capabilities such as:

- Provisioning resources by using Azure Resource Manager templates.
- Running Azure CLI commands.
- Applying Azure Policy.

Commonly-used actions applicable to Azure-based cloud-native applications

Scenarios include:

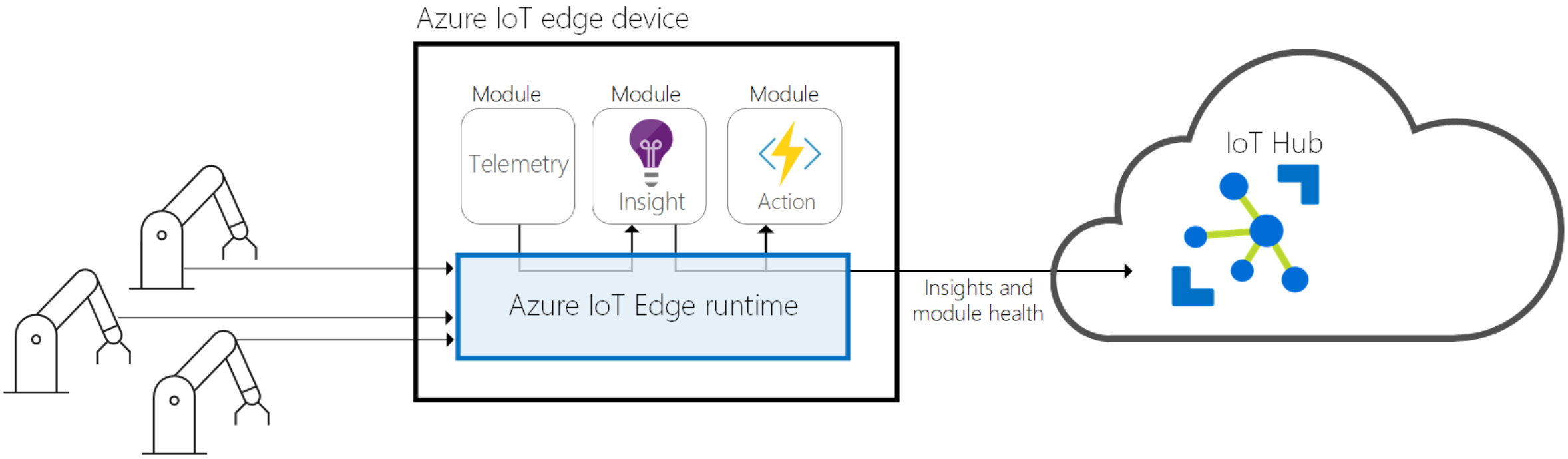
- **azure/login.** This action performs a non-interactive sign-in by using Azure AD service principal credentials.
- **azure/arm-deploy.** This action deploys an Azure Resource Manager template to a target Azure resource group.
- **azure/k8s-create-secret.** This action creates a generic secret or docker-registry secret in a Kubernetes cluster.
- **azure/k8s-bake.** This action prepares a Kubernetes manifest file for deployment into a target Kubernetes cluster.
- **azure/k8s-deploy.** This action deploys a Kubernetes manifest file for deployment into a target Kubernetes cluster.

Demo: Provision cloud-native applications by using GitHub Actions

Explore the role of GitHub in IoT scenarios

What is IoT Edge?

Azure IoT Edge is a managed service that provides the ability to deploy and manage containerized workloads on cross-platform IoT devices.



How to apply DevOps principles in IoT Edge scenarios

The same rationale that favors the use of DevOps in regard to cloud-native applications applies to IoT apps.

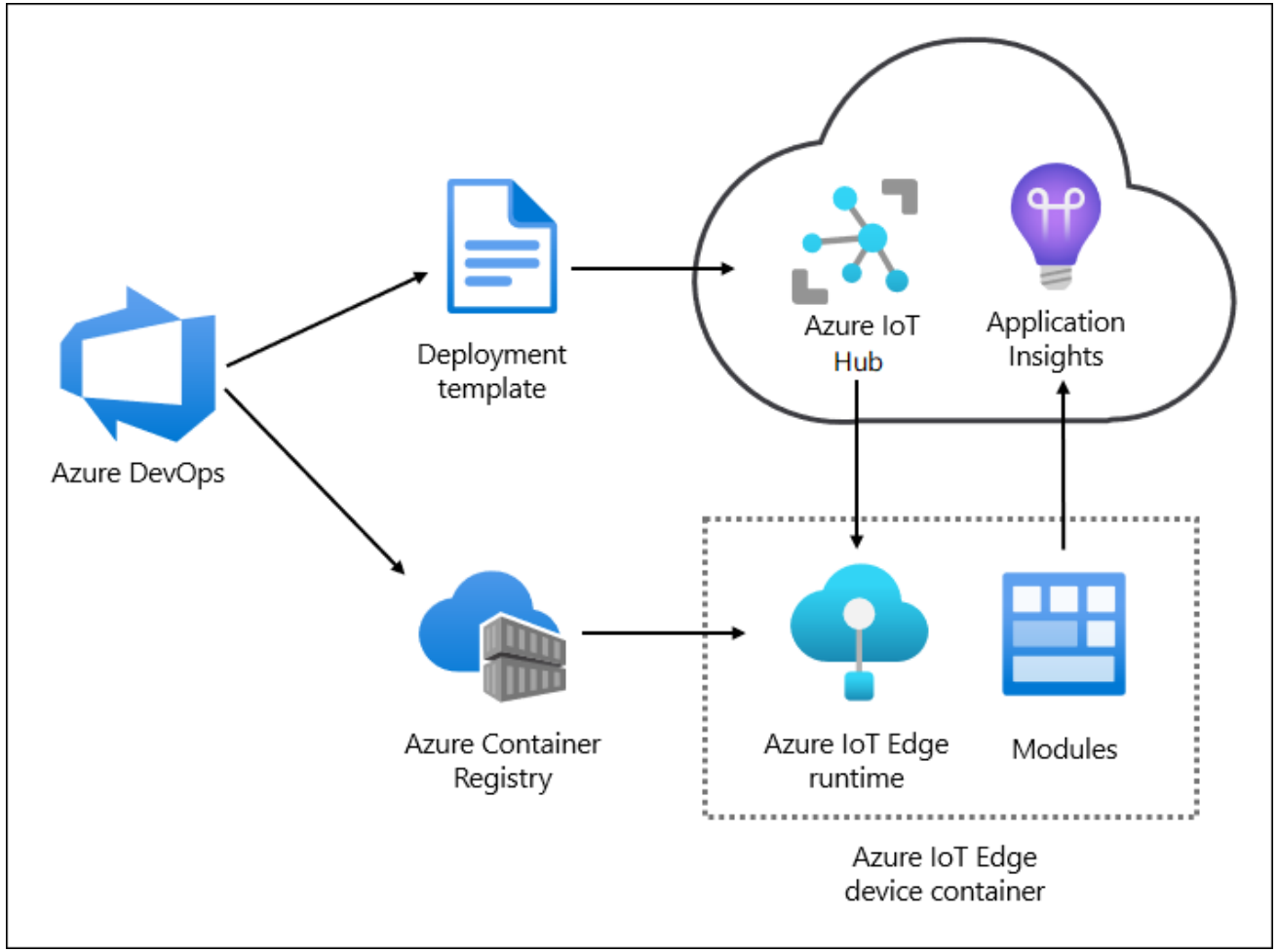
- **Classic approach:** You rely on the visual designer included in the Azure DevOps web-based portal to define a build pipeline that builds your code, tests it, and publishes resulting artifacts.
- **YAML-based approach:** Your pipeline takes the form of a YAML-formatted file, which, by default, resides in the same repository as the code used to build artifacts.

How to design CI/CD for Azure IoT Edge applications

You're considering a design that will include the following infrastructure components:

- **Azure Repos**. One of Azure DevOps components that serves as code repository.
- **Azure Pipelines**. One of Azure DevOps components that automates builds and deployments.
- **Azure Container Registry**. An Azure-hosted private Docker registry that serves as a store for containerized IoT Edge modules.
- **Azure IoT Hub**. An Azure service that provides the ability to connect to, monitor, and manage IoT devices.
- **Azure IoT Hub Device Provisioning Service**. A component of Azure IoT hub that facilitates the automatic provisioning of IoT devices.

How to implement CI/CD for Azure IoT Edge applications



Demo: Configure CI/CD for IoT Edge applications