Microsoft

AZ-203.2
Module 01: Create Azure App Service Web Apps

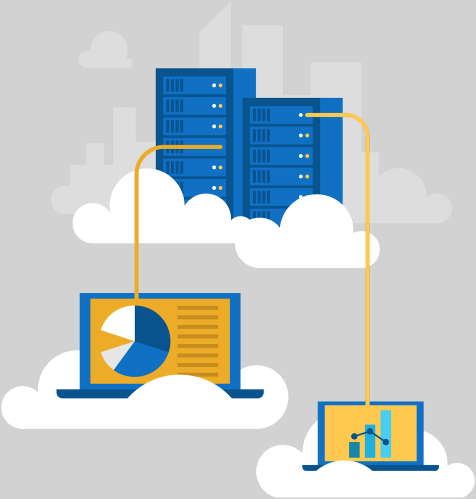
Kishore Chowdary

Microsoft

Topics

- Azure App Service core concepts
- Creating an Azure App Service web app
- Creating background tasks by using WebJobs

Lesson 01: Azure App Service core concepts



App Service

- Service for hosting web applications, REST APIs, and mobile backends can be developed in many of the following languages:

.NET

.NET Core

Java

Ruby

Node.JS

PHP

Python

- Applications can execute and scale in a fully managed, sandbox environment

Web Apps

- Scalable hosting for web applications
  - Provides a quick way to host your web application in the cloud
  - Allows you to scale your web app without being required to redesign for scalability
  - Integrates with Visual Studio
  - Provides an open platform for many different programming languages
- Advantages
  - Near instant deployment
  - SSL and Custom Domain Names available in some tiers
  - WebJobs provide background processing for independent scaling
  - Can scale to larger machines without redeploying applications

Key features of App Service Web Apps

- Multiple languages and frameworks
  - First-class support for ASP.NET , ASP.NET Core, Java, Ruby, Node.js, PHP, or Python
- DevOps optimization
  - Continuous integration and deployment with Visual Studio Team Services, GitHub, Bitbucket, Docker Hub, or Azure Container Registry
- Global scale with high availability
  - Scale up or out manually or automatically. Host anywhere in the Microsoft global datacenter infrastructure
- Connections to SaaS platforms and on-premises data
  - More than 50 connectors for enterprise systems (such as SAP), SaaS services (such as Salesforce), and internet services (such as Facebook)

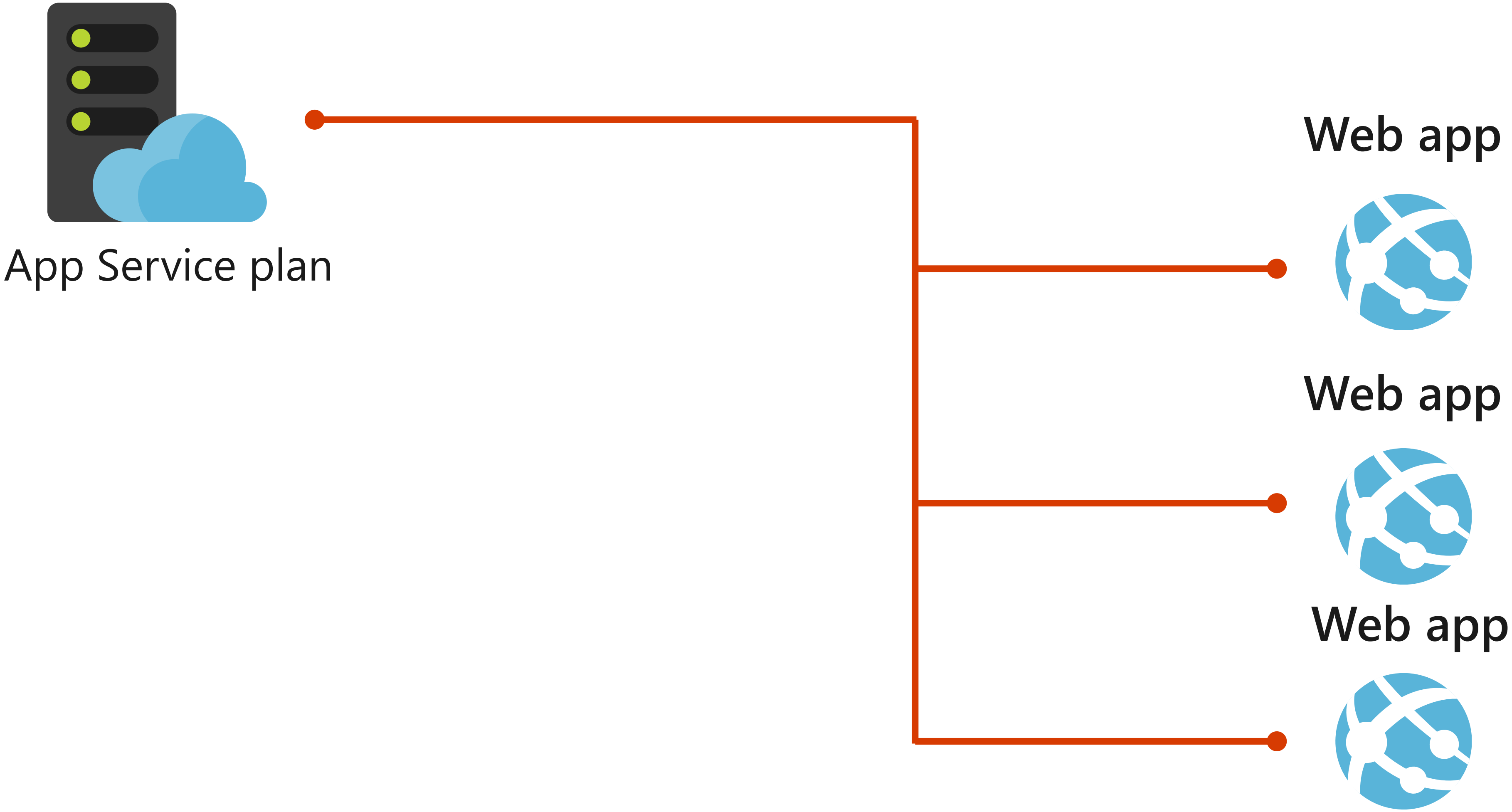
Key features of App Service Web Apps (cont.)

- Security and compliance
  - App Service is ISO, SOC, and PCI compliant
- Application templates
  - Templates in the Azure Marketplace, such as WordPress, Joomla, and Drupal
- Visual Studio integration
  - Streamline the work of creating, deploying, and debugging
- API and mobile features
  - Turn-key Cross-Origin Resource Sharing (CORS) support for RESTful API scenarios, and enables authentication, offline data sync, push notifications, and more
- Serverless code
  - Run code on-demand without having to explicitly provision or manage infrastructure

App Service plans

- App Service plans can logically group apps within a subscription:
  - Characteristics such as features, capacity, and tiers are shared among the website instance in the group
  - The App Service plan is the unit of billing in most cases
- Multiple App Service plans can exist in a single Resource Group and multiple apps can exist in a single App Service plan

App Service plans (continued)



App Service plan

Web app

Web app

Web app

Authentication and authorization

- Built-in authentication and authorization support
    - No extra code required to make use of these features
- User claims are made available to code
    - If you wish to enhance the authentication support, you can use your existing code with popular identity frameworks:
        - ASP.NET Identity
        - PHP server variables
- Built-in token store
- Logging and tracing enabled for authentication events
- Support for popular identity providers
    - Azure Active Directory (Azure AD), Microsoft accounts, Facebook, Google, Twitter, more…

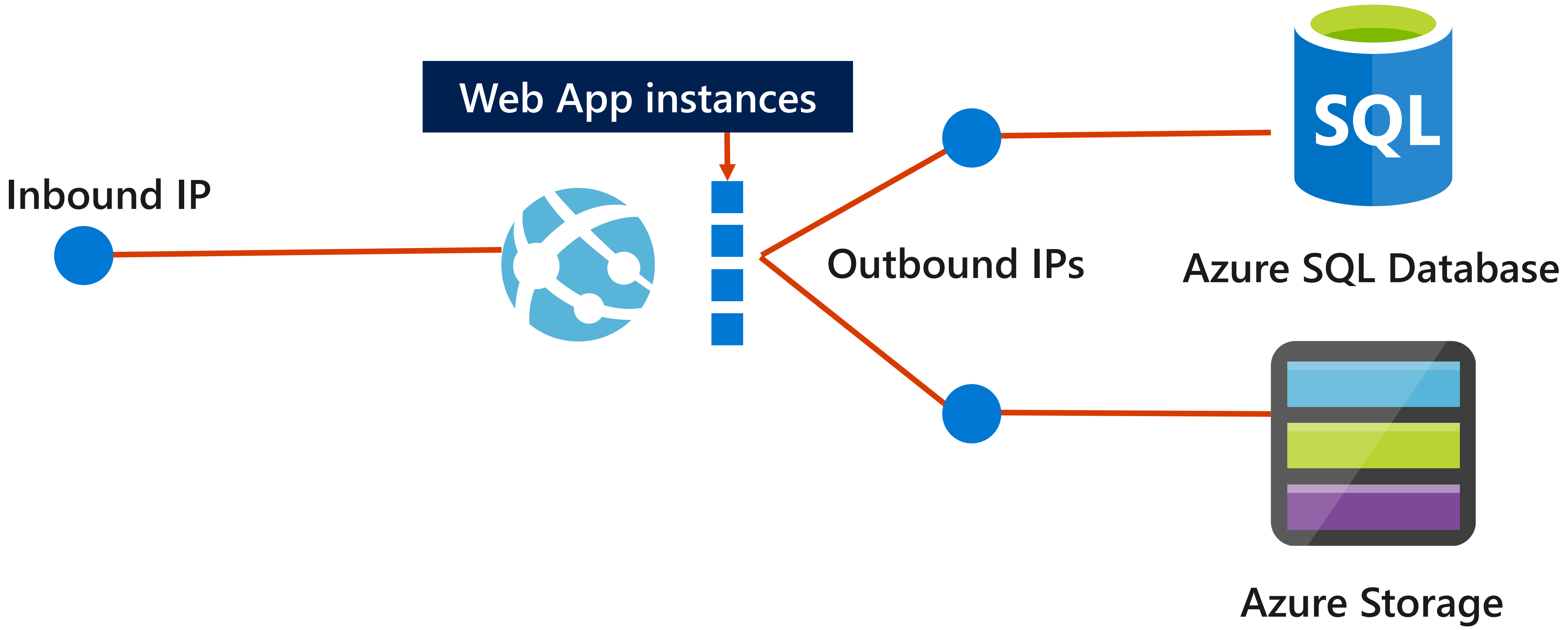
OS and runtime patching

- OS and application stack are managed by Azure on your behalf
- Monthly OS patching
  - Physical servers
  - Guest virtual machines
- Stable versions of application runtimes are periodically added to App Services
  - Some are installed side by side, while others replace existing versions
  - You can manually migrate from one application runtime to another

Inbound and outbound IP addresses

- Each app has a single inbound IP address
    - Regardless of scale-out quantity
- Inbound IP address can change
    - Delete an app and re-create it in a new resource group
    - Delete the last app in a resource group + region combination and re-create it
    - Delete an existing SSL binding
- You can opt to use a state inbound IP
- Each app has a set number of outbound IP addresses
    - The set and quantity changes as you scale your app between tiers

Outbound IP addresses



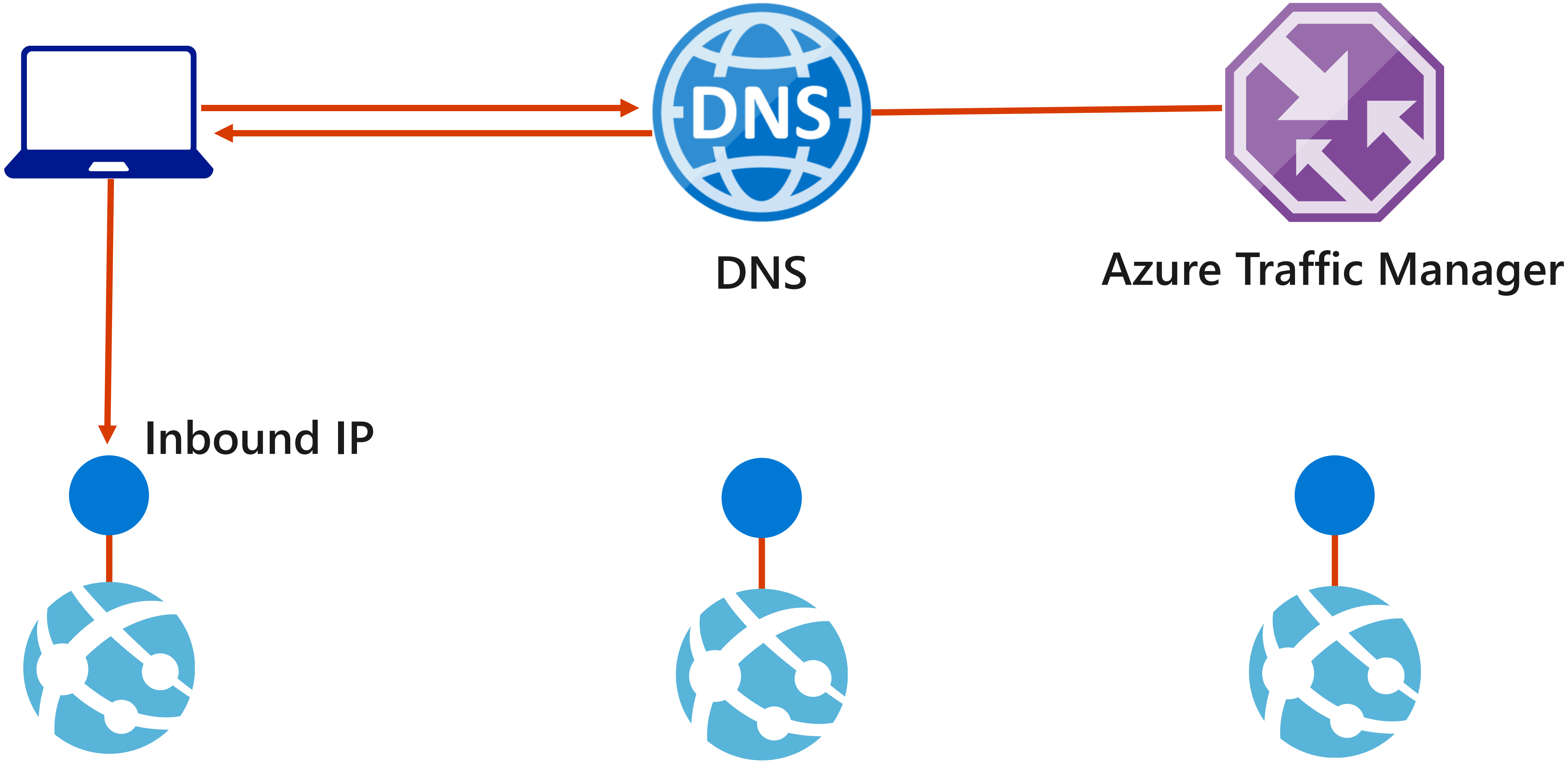
When IP addresses change

- Inbound
  - When you delete an app and recreate it
  - When you delete the last app in a resource group
  - When you delete an existing SSL binding
- Outbound
  - When you scale from a lower tier (Basic, Standard, Premium) to the Premium V2 tier

Controlling traffic by using Azure Traffic Manager

- Routes requests from clients to apps in Azure
- Keeps track of app status (running, stopped, deleted)
  - Will automatically route traffic away from an unavailable app
- Configured by using profiles
  - Stores the routing method for requests
  - Stores a list of endpoints (apps) to route requests to
  - Stores information about endpoint status

Azure Traffic Manager and Web Apps



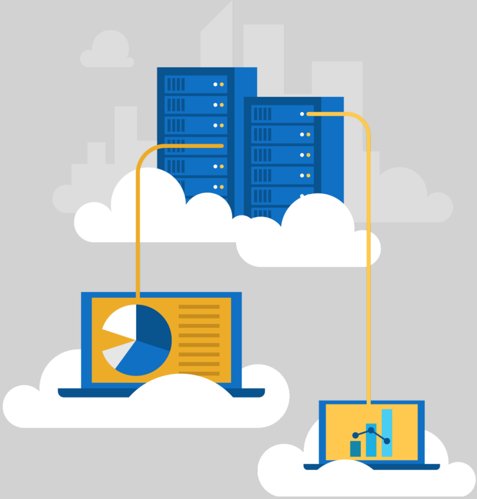
Azure Traffic Manager routing methods

- Priority
  - Distribute users to a specific app
  - In case of failure, route users to backup apps based on a priority scheme
- Weighted
  - Distribute traffic across apps according to weights that you define
  - Your weight definition could potentially distribute users evenly
- Performance
  - Route users to the “closest” app location based on latency
- Geographic
  - Route users to specific app locations based on their current location

App Service environments (ASEs)

- App Service variant that provides a fully isolated and dedicated environment for securely running App Service apps at high scale
- Ideal for application workloads that require:
  - Very high scale, higher than typical App Service capacity
  - Network isolation and secure network access
  - High memory utilization
- Single or Multi-region
- Deployed to a virtual network
- An ASE is dedicated exclusively to a single subscription
  - Max 100 instances

Lesson 02: Creating an Azure App Service web app



Creating a web app with Azure CLI (continued)

```
# generate a unique name and store as a shell variable
webappname=mywebapp$RANDOM

# create a resource group
az group create --location westeurope --name myResourceGroup

# create an App Service plan
az appservice plan create --name $webappname --resource-group myResourceGroup --sku
FREE

# create a Web App
az webapp create --name $webappname `
    --resource-group myResourceGroup `
    --plan $webappname
```

Deploying a web app with Azure CLI

```
# store a repository url as a shell variable
gitrepo=https://github.com/Azure-Samples/php-docs-hello-world

# deploy code from a Git repository
az webapp deployment source config --name $webappname --resource-group myResourceGroup
--repo-url $gitrepo --branch master --manual-integration

# print out the FQDN for the Web App
echo http://$webappname.azurewebsites.net
```

Creating a Web App with Azure PowerShell

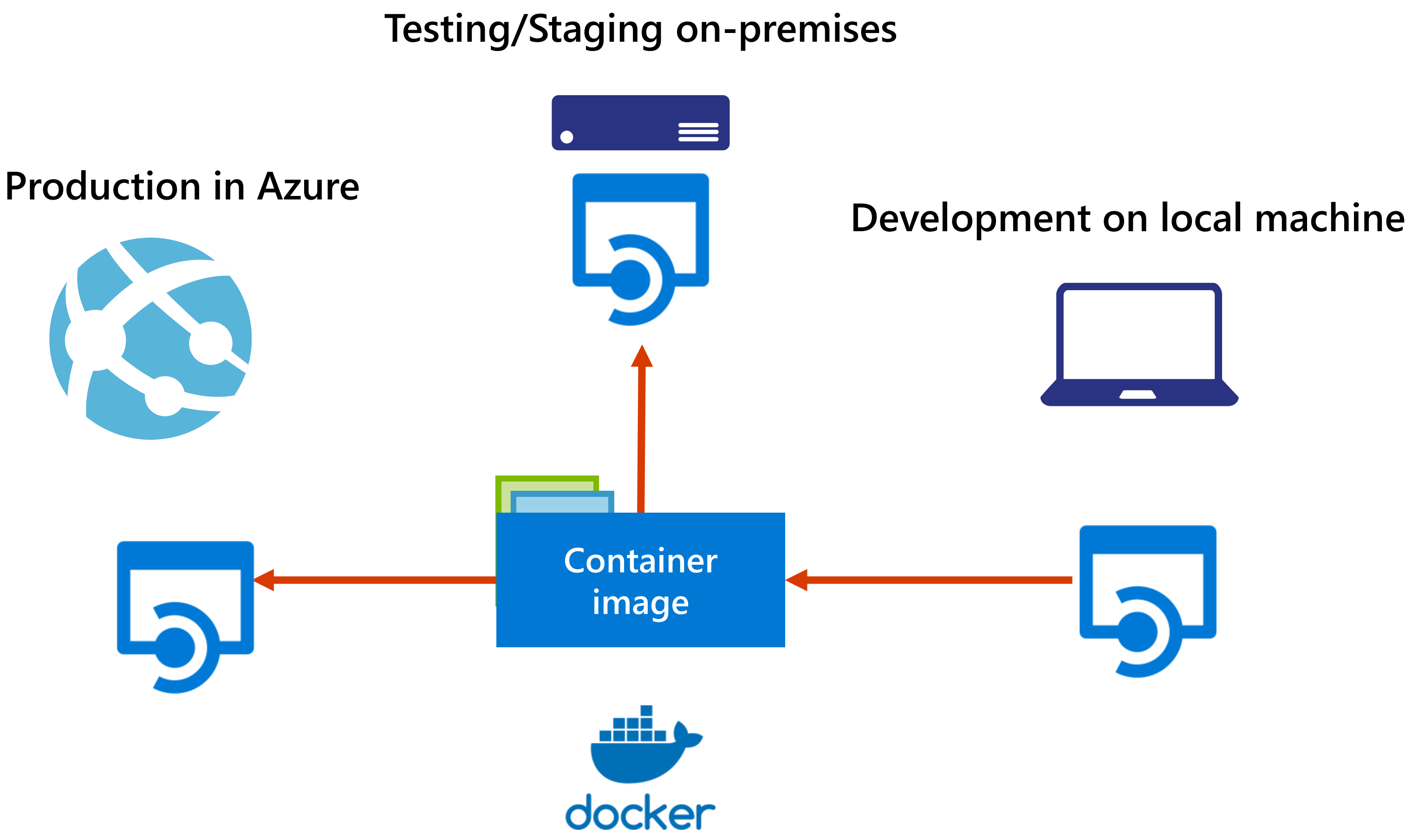
Command	Notes
`New-AzureRmResourceGroup`	Creates a resource group in which all resources are stored.
`New-AzureRmAppServicePlan`	Creates an App Service plan.
`New-AzureRmWebApp`	Creates an Azure Web App.
`Set-AzureRmResource`	Modifies a resource in a resource group.

App Service on Linux

Why Linux?

- Many application stacks are optimized for Linux:
  - Ruby/Rails, PHP, Node, and others
  - Often, better tools are available on Linux for these stacks
- New and upcoming frameworks are built for Linux first and then Windows
- Portability of Docker containers
- Linux is at the forefront of innovations in nano and microservice architecture

Docker in App Service on Linux


Testing/Staging on-premises

Production in Azure

Development on local machine

Container image

Web apps for Linux containers

Deploy applications and solutions that are containerized directly to App Service Web Apps

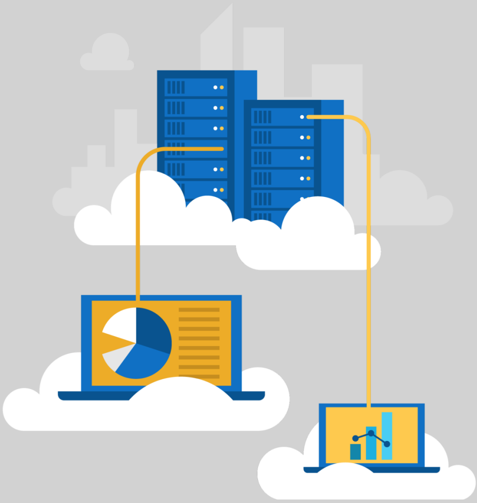
- Simplifies deployment
- Matches the already popular container workflow using:
  - CI/CD with Docker Hub, Azure Container Registry, or GitHub
- Compatible with existing App Service features:
  - Auto-scale, Deployment Slots, and others

Web apps for Linux containers (continued)

Containers can be sourced from your existing registries:

- Docker Hub:
  - Deploy images already shared on Docker Hub
  - Deploy the most popular official images
  - Private images are available on Docker Hub
- Azure Container Registry:
  - Managed service for hosting Docker images
  - Can deploy to Docker Swarm, Kubernetes, or Web App for Containers

Lesson 03: Creating background tasks by using WebJobs



WebJobs

- Built-in feature of Azure App Service
- Doesn’t incur additional costs and doesn’t require new resources
- Runs background tasks and scripts within the same application context as your apps
- Supports the following programs or scripts:
  - .cmd, .bat, .exe (using Windows cmd)
  - .ps1 (using PowerShell)
  - .sh (using Bash)
  - .php (using PHP)
  - .py (using Python)
  - .js (using Node.js)
  - .jar (using Java)

WebJob types

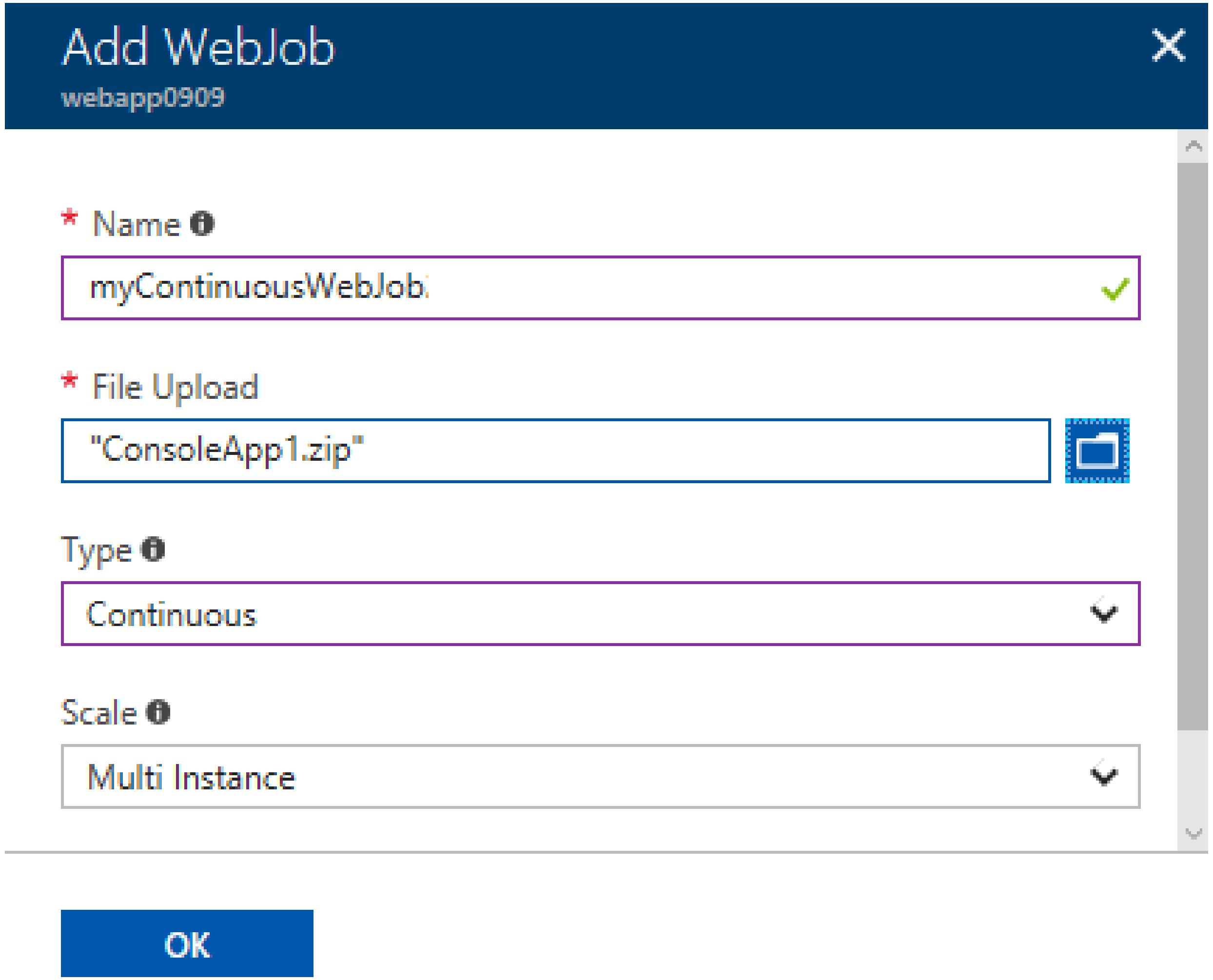
Continuous

- Starts immediately when the WebJob is created
- Runs on all instances that the web app runs on
- Supports remote debugging

Triggered

- Starts only when triggered manually or on a schedule
- Runs on a single instance

Creating a continuous WebJob



Add WebJob

webapp0909

* Name

myContinuousWebJob

* File Upload

"ConsoleApp1.zip"

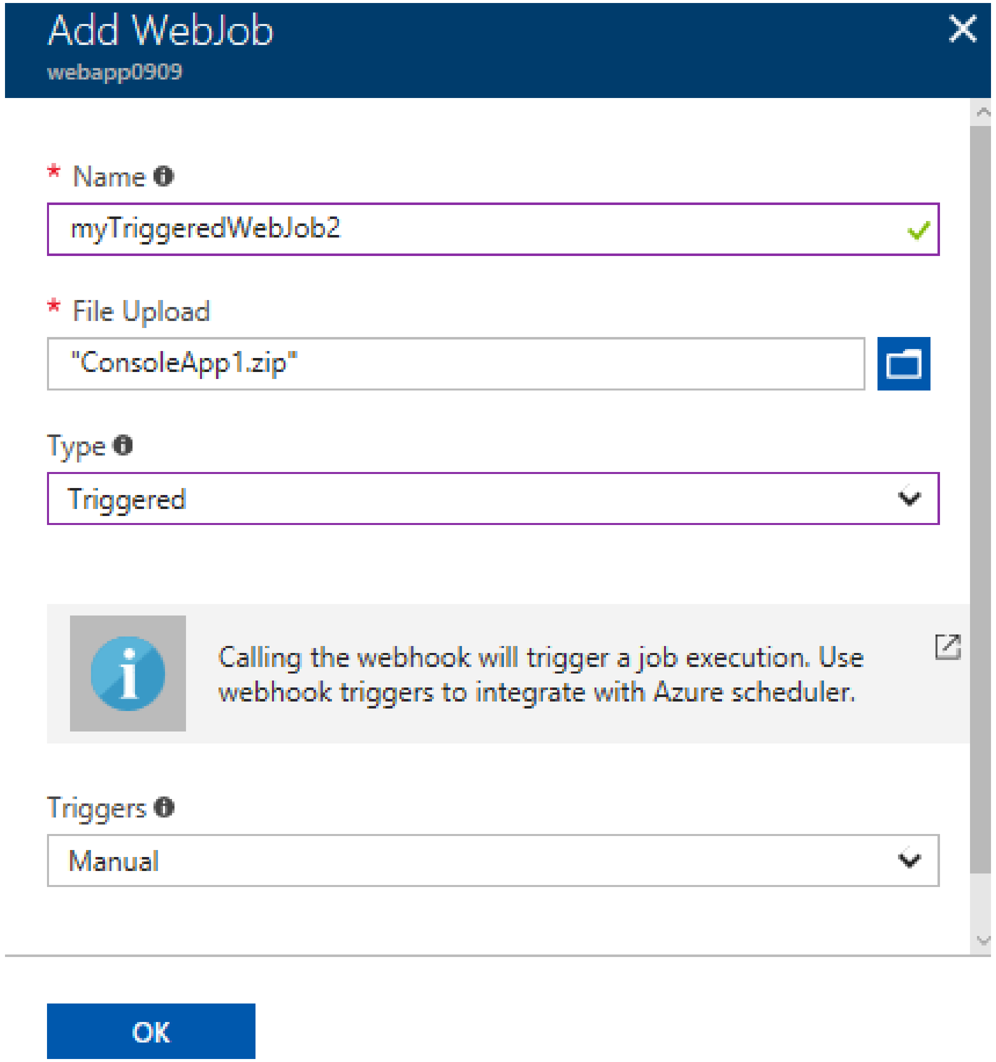
Type

Continuous

Scale

Multi Instance

Creating a triggered WebJob



Add WebJob

webapp0909

* Name

myTriggeredWebJob2

* File Upload

"ConsoleApp1.zip"

Type

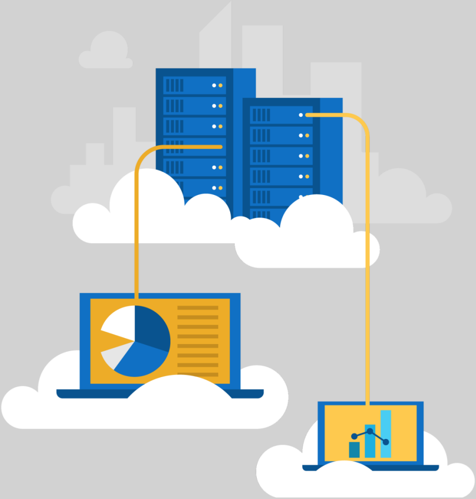
Triggered

Calling the webhook will trigger a job execution. Use webhook triggers to integrate with Azure scheduler.

Triggers

Manual

Demo – Azure Web Apps



Microsoft

Review

- Azure App Service core concepts
- Creating an Azure App Service web app
- Creating background tasks by using WebJobs

Microsoft