# Module 5

## Implementing Azure App Service

# Module Overview

- Introduction to App Service
- Planning app deployment in App Service
- Implementing and maintaining web apps
- Configuring web apps
- Monitoring web apps and WebJobs
- Implementing Traffic Manager

# Lesson 1: Introduction to App Service

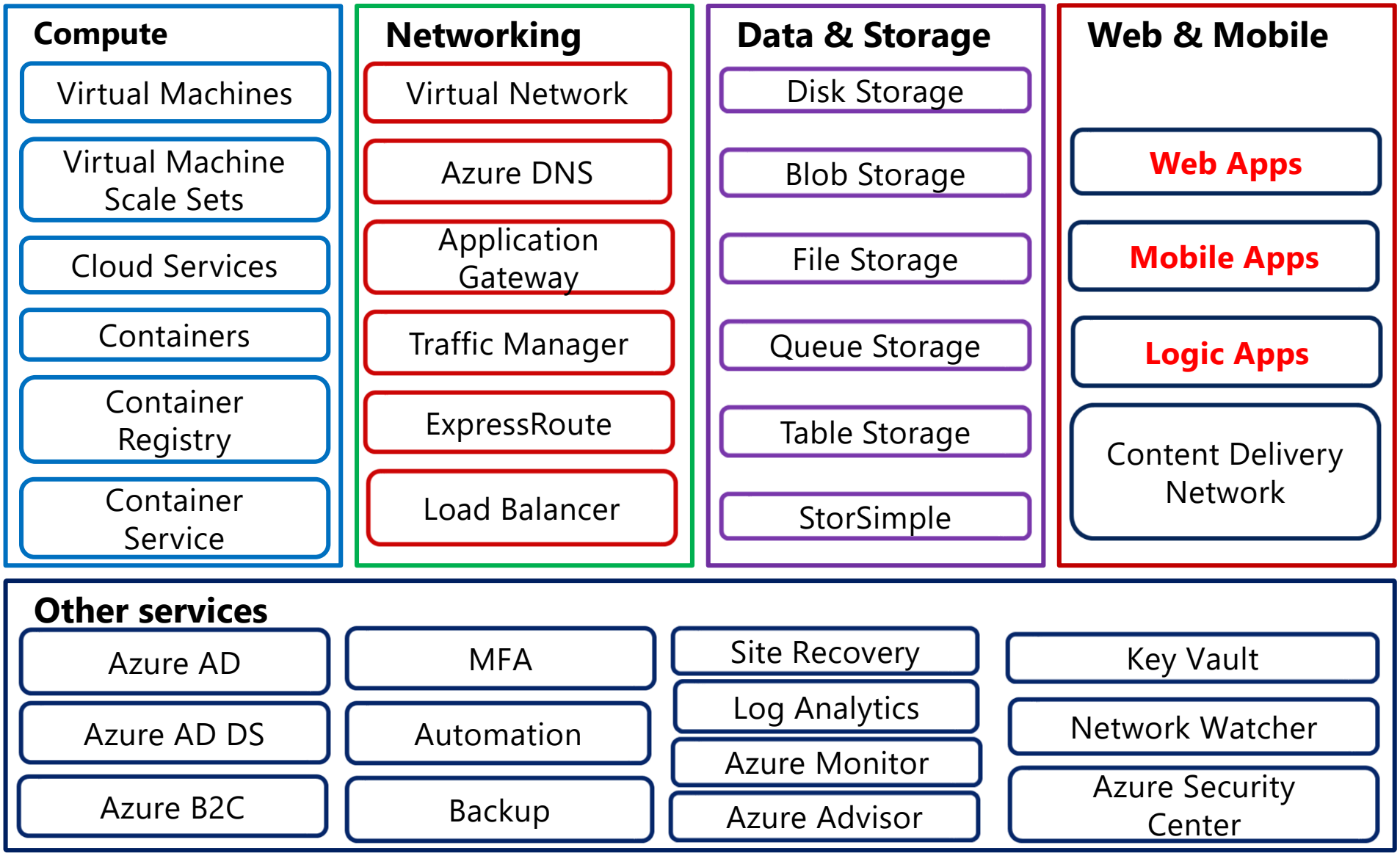
- Demonstration: Preparing the lab environment
- Overview of App Service
- Overview of Web Apps
- Overview of Mobile Apps
- Overview of Logic Apps
- Overview of API Apps
- Overview of the App Service Environment

# Demonstration: Preparing the lab environment

In this demonstration, you will learn how to prepare the lab environment

**Note**: To prepare the lab environment for this module, you must complete this task

# Overview of App Service



# Overview of Web Apps

Web Apps concepts:

- Gallery applications
- Autoscaling
- Continuous integration and deployment
- Deployment slots
- Testing in production
- Azure WebJobs
- Hybrid connections
- Azure virtual network integration
- Authentication and authorization
- Logs and Alerts

# Overview of Mobile Apps

## The Mobile Apps Feature:

- Fulfills Mobile App requirements:
  - Store and access data
  - Send notifications
  - Authenticate and authorize
  - Implement business logic
  - Cross-platform support (Windows, iOS, Android)
- Includes support for the same features as Web Apps:
  - Autoscaling
  - Continuous integration, delivery, and deployment
  - WebJobs
  - Hybrid connections and Azure virtual network integration
  - Authentication and authorization
  - Alerts and logs

# Overview of Logic Apps

Logic Apps integrate apps by using:

- Connectors:
  - Built-in actions
  - Managed connectors
    - Standard connectors
    - On-premises connectors
    - Integration account connectors
    - Enterprise connectors
- Triggers:
  - Poll
  - Push
  - Recurrence

# Overview of API Apps

- Features of API Apps:
  - Visual Studio integration
  - Consumption model
  - Bring your existing API
  - Support for Swagger metadata
  - Support for cross-origin resource sharing
- To implement an API app:
  - Create an API app
  - Create a starter backend API
  - Implement client code
  - Optionally, configure Swagger endpoint and enable CORS

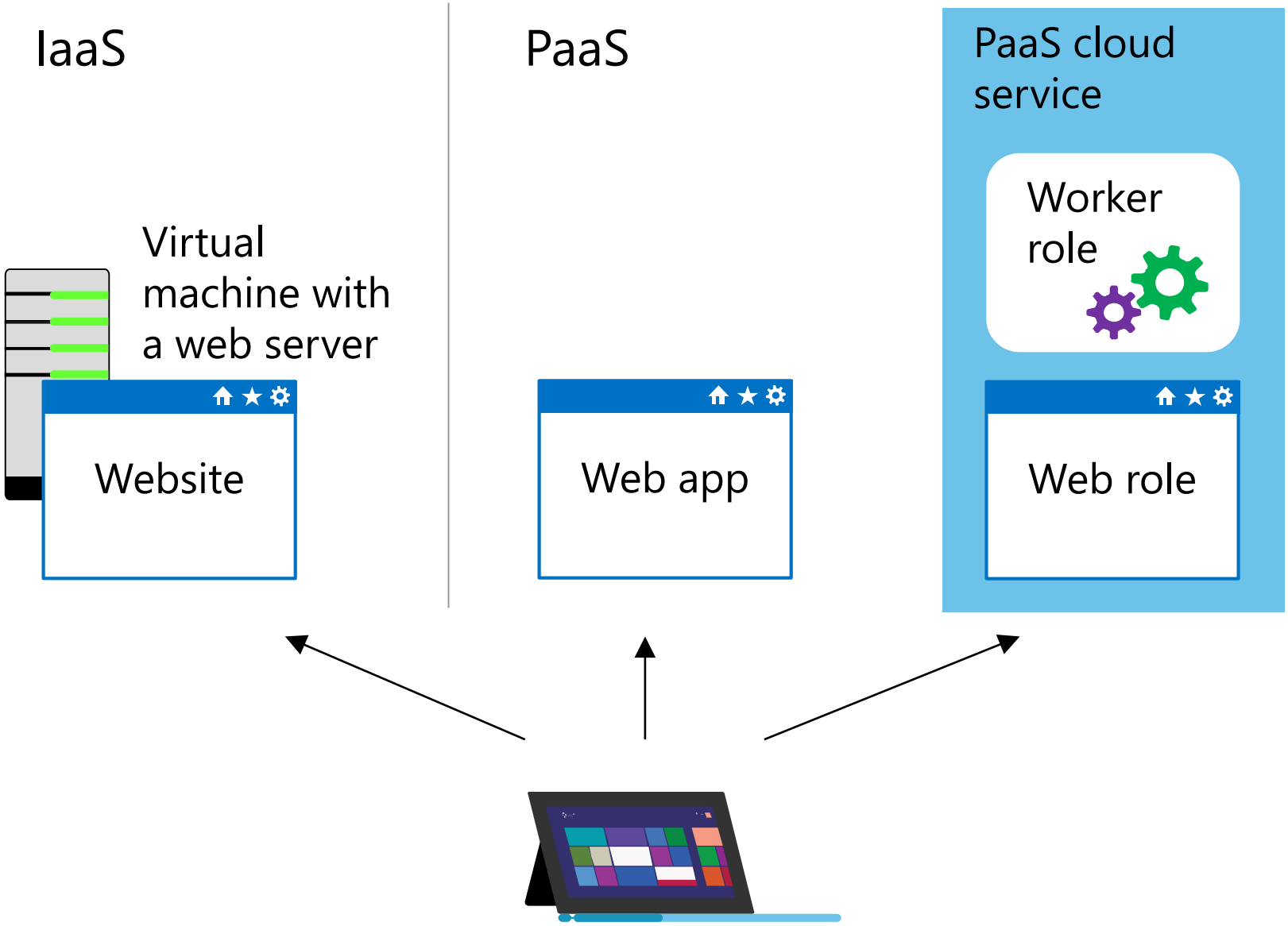
# Overview of the App Service Environment

- Front-end and worker instances
- Direct virtual network connectivity (a single subnet)
- External or internal VIP
- Two versions:
  - ASEv1:
    - Classic and Azure Resource Manager deployment models
    - Up to 55 instances
    - Requires instance–level and Service Plan–level scaling
    - Core-based pricing model
  - ASEv2:
    - Azure Resource Manager deployment model
    - Up to 100 instances
    - Service Plan–level scaling
    - Core-based plus flat monthly fee pricing model

# Lesson 2: Planning app deployment in App Service

- Comparing Web Apps, Azure Cloud Services, and Azure VMs
- Managing App Service plans
- Comparing app deployment methods in App Service

# Comparing Web Apps, Azure Cloud Services, and Azure VMs



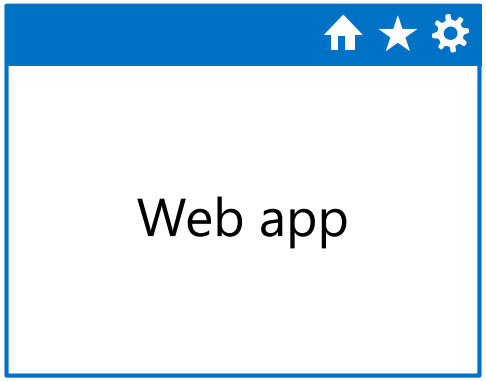
# Managing App Service plans

Pricing tiers available for all Windows Server-based Service Apps include:

- Free
- Shared
- Basic
- Standard
- Premium
- PremiumV2
- Isolated
- Consumption

# Comparing app deployment methods in App Service

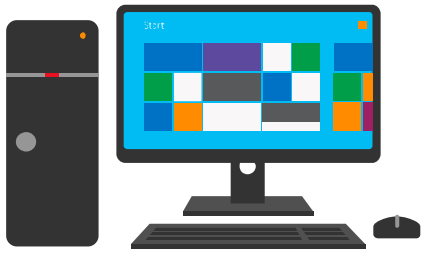
**Cloud**




VSTS

GitHub

**On-premises**



FTP, Web Deploy,
Visual Studio,
Web Matrix,
MSBuild

TFS
Git

# Lesson 3: Implementing and maintaining web apps

- Creating web apps
- Deploying web apps
- Updating web apps
- Demonstration: Deploying web apps

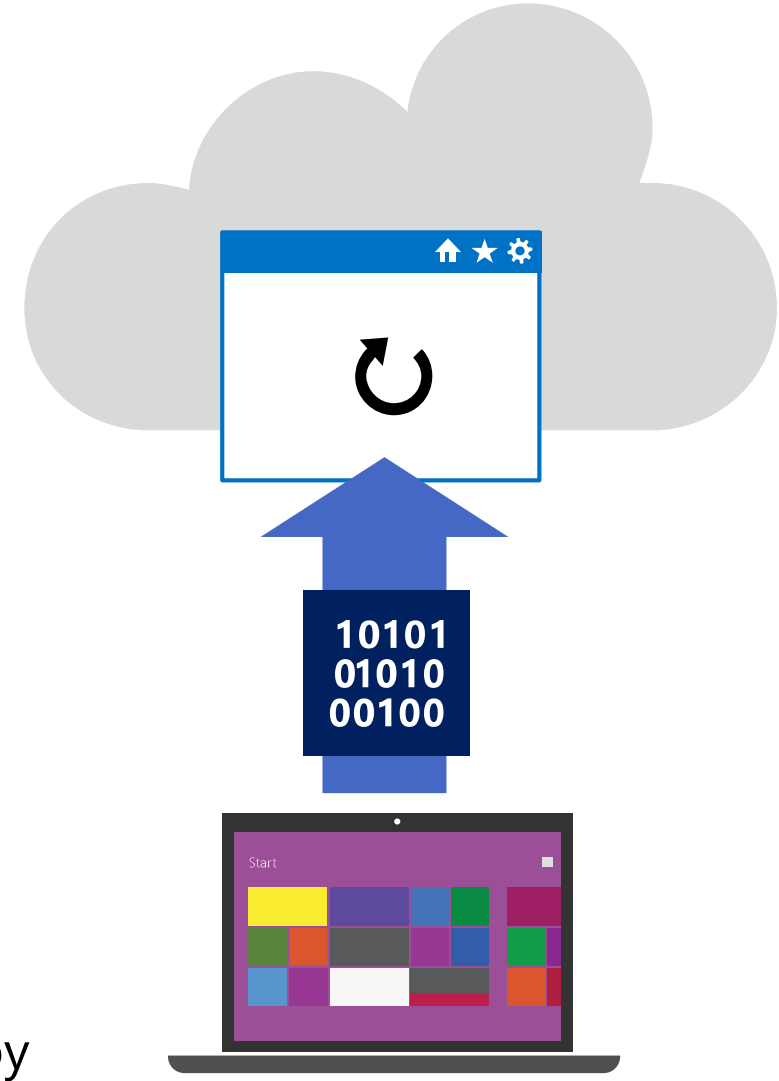
# Creating web apps

Create an Azure web app by using:

- The Azure portal
- **New-AzureRmWebApp**
  - **New-AzureRmWebApp –ResourceGroupName *rgname* –Name *webappname* –AppServicePlan *appserviceplanname***
- **az webapp create**
  - **az webapp create –-resource-group *rgname* –-name *webappname* –-plan *appserviceplanname***
  - **az webapp create –-resource-group *rgname* –-name *webappname* –-plan *appserviceplanname* --deployment-container-image-name *publisher/image:tag***
- An Azure Resource Manager template
- Visual Studio

# Deploying web apps

- Advantages of Web Deploy:
  - Uploads only modified files
  - Supports HTTPS
  - Allows setting NTFS permissions
  - Supports database publishing
  - Allows setting connection strings
- Scripted deployment with:
  - MSDeploy.exe
  - Azure PowerShell
  - Azure CLI
  - Git
- FTP:
  - Passive vs active
  - Limitations in comparison with Web Deploy



# Updating web apps

- Deploy updates by using:
  - FTP
  - Web Deploy
- Continuous deployment and delivery:
  - Connect a project to a web app in Azure
  - Check in the changes
  - Build and deploy automatically
- Staging and production slots:
  - Some settings change on slot swap
  - Some settings do not change on slot swap
  - App settings and connection strings support “Slot setting”

# Demonstration: Deploying web apps

In this demonstration, you will see how to:

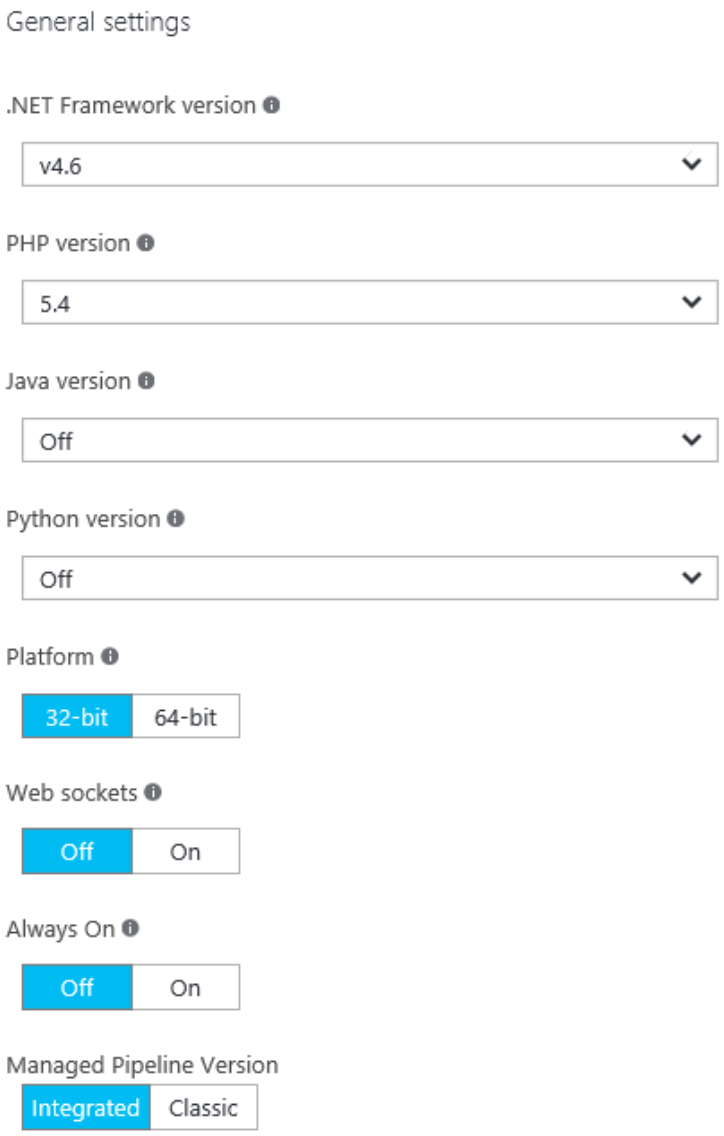
- Create a new .NET Core web app by using Visual Studio
- Create a new Azure web app by using Azure CLI
- Publish the web app from Visual Studio

# Lesson 4: Configuring web apps

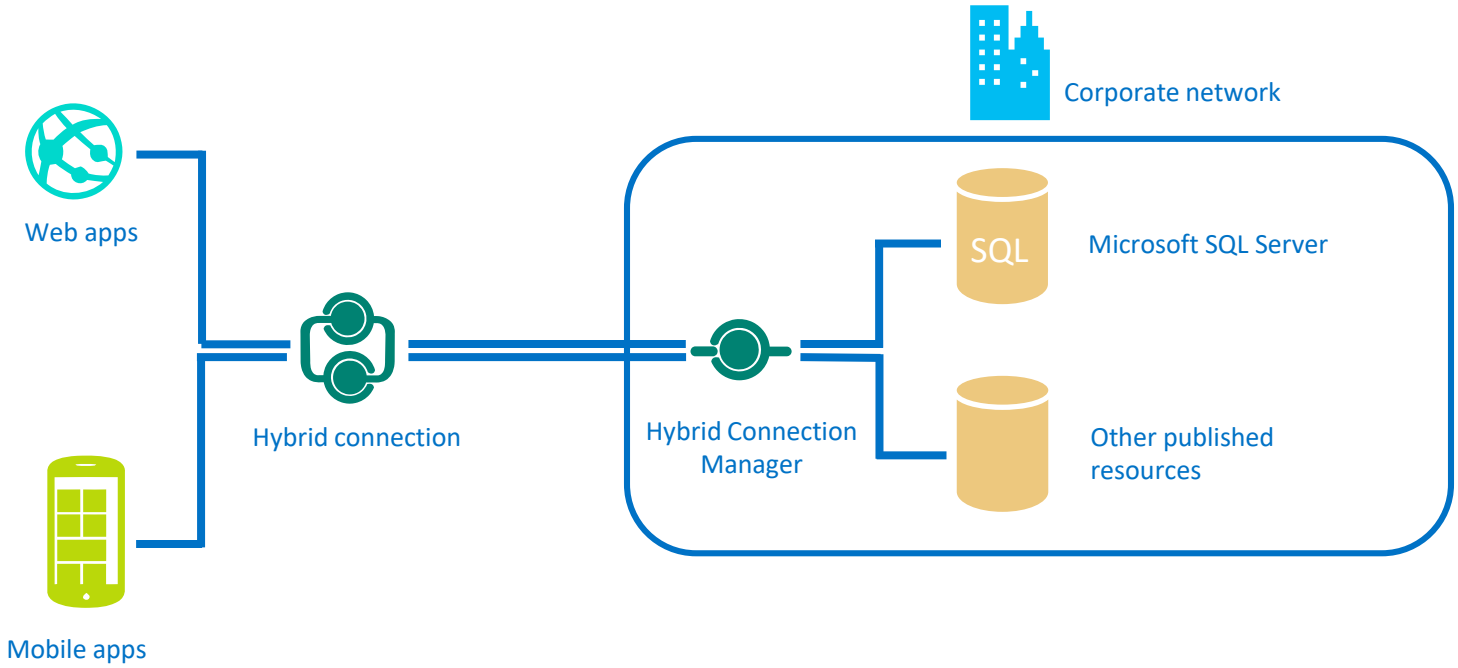
- Configuring a web app's application and authentication settings
- Configuring virtual network connectivity and hybrid connectivity
- Configuring availability and scalability
- Implementing WebJobs
- Demonstration: Configuring web app settings and autoscaling and creating a WebJob

# Configuring a web app's application and authentication settings

- Framework versions
- Platform and Web sockets
- Always On
- Managed Pipeline Version
- ARR Affinity
- Auto Swap
- Debugging
- Certificates, Domain Names, and SSL Bindings
- App Settings
- Connection Strings
- Default Documents
- Diagnostic logs
- Authentication and Authorization



# Configuring virtual network connectivity and hybrid connectivity



# Configuring availability and scalability

- Free and shared pricing tiers – no scaling support
- Basic pricing tier:
  - Instance size
  - Instance count
- Standard and premium pricing tiers:
  - Instance size
  - Instance count
  - Scale based on a metric:
    - One or more rules
    - Instance limits
    - Schedule
  - Scale to a specific instance count:
    - Instance count
    - Schedule

# Implementing WebJobs

- WebJobs are scripts that run:
  - Continuously
  - Triggered
    - Scheduled
    - Manual
- WebJobs can be:
  - Batch files (.cmd, .bat)
  - PowerShell scripts (.ps1)
  - Bash shell scripts (.sh)
  - PHP scripts (.php)
  - Python scripts (.py)
  - Node.js JavaScripts (.js)
  - JavaScript (.jar)

# Demonstration: Configuring web app settings and autoscaling and creating a WebJob

In this demonstration, you will see how to:

- Configure web app settings
- Configure autoscaling

# Lesson 5: Monitoring web apps and WebJobs

- Configuring application and site diagnostics
- Monitoring web apps
- Using Kudu
- Demonstration: Using Kudu to monitor a WebJob

# Configuring application and site diagnostics

- Configure the following application logging settings:
  - Log storage location
  - Logging level
  - Retention period
- Configure the following site diagnostics settings:
  - Web server logging
  - Detailed error messages
  - Failed request tracing

# Monitoring web apps

- Access diagnostic logs by using:
  - FTP
  - Azure PowerShell
  - Azure CLI
- View logs in Visual Studio by using Application Insights
- Monitor web apps in the Azure portal by:
  - Adding metrics
  - Configuring alerts
    - Email notifications
    - Webhooks
    - Logic apps

# Using Kudu

- Kudu:
  - Provides Git support for web apps
  - Runs WebJobs
  - Implements a diagnostic and troubleshooting user interface
- To access the Kudu user interface for a web app, enter https://mysite.scm.azurewebsites.net
- In the Kudu user interface, you can:
  - Run commands
  - View processes
  - Access log files
  - Add extensions

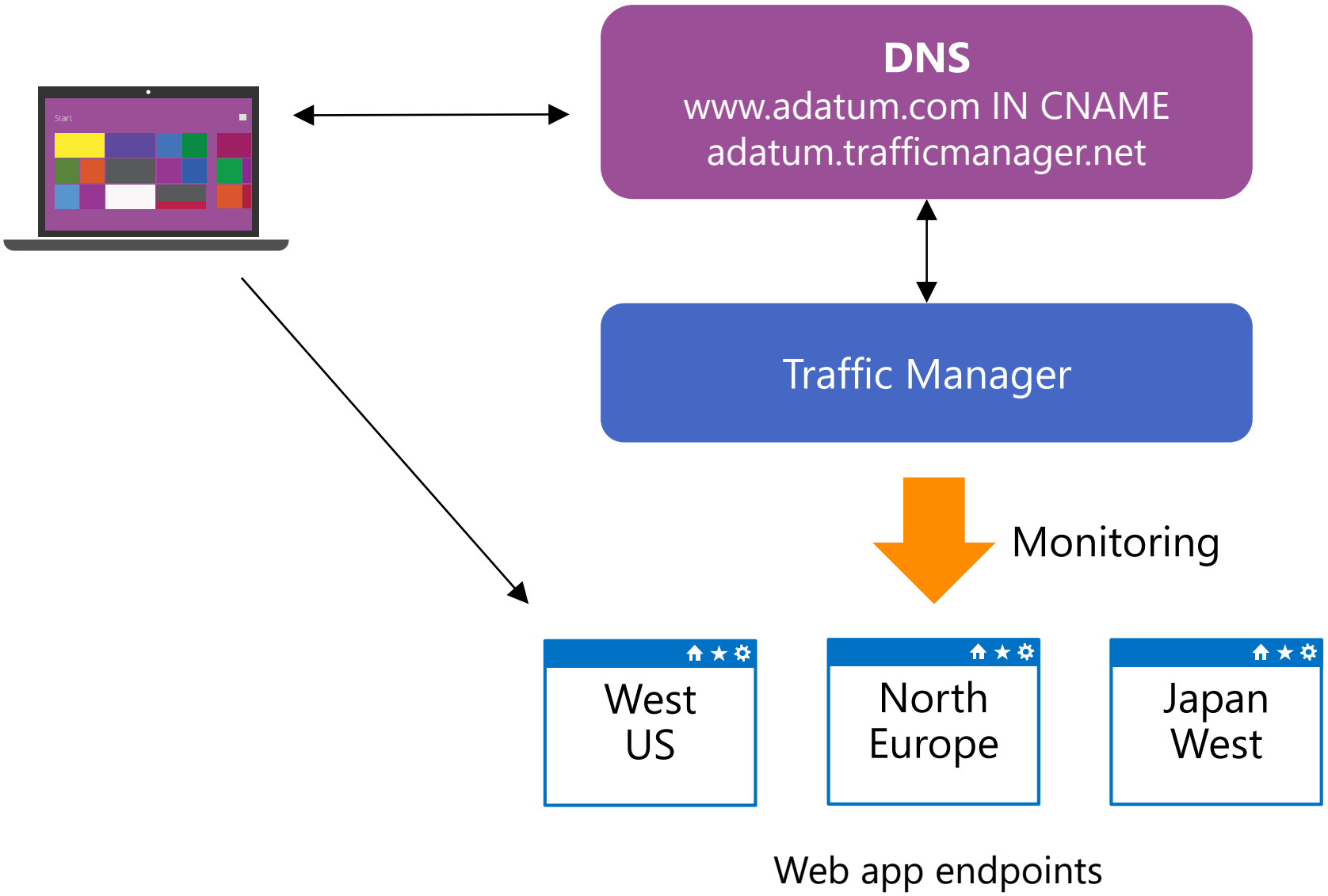
# Demonstration: Using Kudu to monitor a WebJob

In this demonstration, you will see how to use Kudu to monitor the status of a WebJob

# Lesson 6: Implementing Traffic Manager

- Overview of Traffic Manager
- Configuring Traffic Manager
- Traffic Manager best practices
- Demonstration: Configuring Traffic Manager

# Overview of Traffic Manager



# Configuring Traffic Manager

1. Add a DNS CNAME record
2. Create a Traffic Manager profile
3. Configure a DNS prefix
4. Choose a load-balancing method:
   - Performance
   - Weighted
   - Priority
   - Geographic
5. Add endpoints to the Traffic Manager profile
6. Configure endpoint monitoring

# Traffic Manager best practices

- Consider implications of changing DNS TTL value
- Use staging endpoints for testing in production
- Make endpoints consistent:
  - Same web app and port number
  - Same monitoring settings
- Disable endpoints for web app maintenance

# Demonstration: Configuring Traffic Manager

In this demonstration, you will see how to:

- Create a new Traffic Manager profile
- Add an endpoint to a Traffic Manager profile by using the Azure portal
- Test Traffic Manager

# Lab: Implementing web apps

- Exercise 1: Creating web apps
- Exercise 2: Deploying a web app
- Exercise 3: Managing web apps
- Exercise 4: Implementing Traffic Manager

Logon Information

Virtual machine: **20533E-MIA-CL1**
User name: **Student**
Password: **Pa55w.rd**

Estimated Time: 60 minutes

# Lab Scenario

A. Datum Corporation's public-facing web app currently runs on an IIS web server at the company's chosen ISP. A. Datum wants to migrate this web app into Azure. You must test the Web Apps functionality by setting up a test A. Datum web app. The A. Datum development team has provided you with web app content to deploy. You must ensure that the team will be able to stage changes to the test web app before you deploy these changes to the public-facing web app. A. Datum is a global company, so you also want to test Azure Traffic Manager, and demonstrate how it distributes traffic across multiple instances of the web app.

# Lab Review

- In the lab, you deployed the A. Datum production website to the production slot of an Azure web app. You also deployed a new version of the site to a staging slot. Within a web browser, how can you tell which is the production site and which is the staging site?
- At the end of the lab, you used an FQDN within the trafficmanager.net domain to access your web app. How can you use your own registered domain name to access this web app?

# Module Review and Takeaways

- Review Question