# Automated CI/CD Deployment Project: .NET Applications to Windows IIS using Azure DevOps

## Project Title & Overview

### Automated CI/CD Deployment of .NET Applications to Windows IIS using Azure DevOps

This comprehensive project demonstrates the implementation of a complete Continuous Integration and Continuous Deployment (CI/CD) pipeline for deploying .NET applications to Windows Internet Information Services (IIS) servers using Azure DevOps. The solution streamlines and automates the entire deployment process, from code commit to production release.


Professional title slide for CI/CD project presentation

### Project Objectives

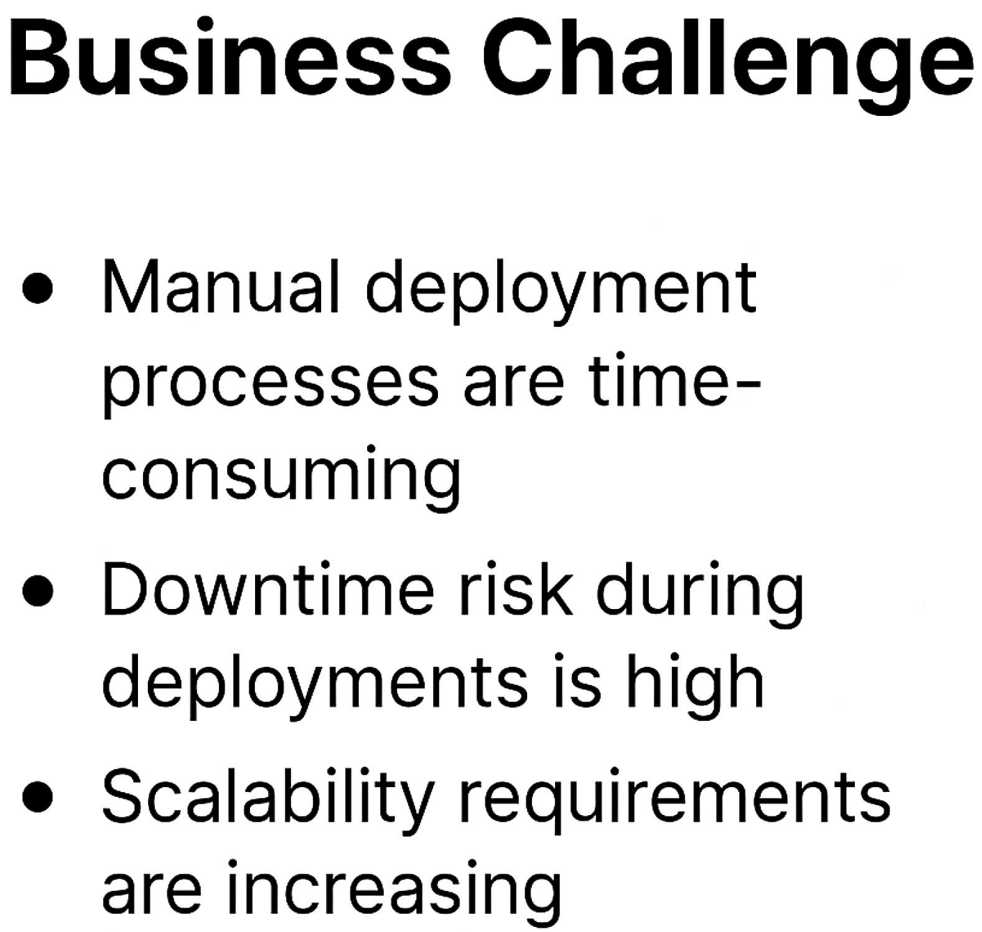
- **Automate deployment workflows** from code commit to IIS server updates
- **Ensure repeatability and reliability** in all deployments
- **Implement comprehensive CI/CD practices** with automated testing and validation
- **Reduce manual effort** and eliminate deployment-related errors
- **Improve deployment visibility** with comprehensive monitoring and reporting

## Business Problem Statement & Challenges

### Current State Challenges

Modern development teams face significant obstacles with traditional manual deployment processes:

- **Manual Deployment Bottlenecks**: Time-consuming manual processes that create deployment delays
- **Inconsistent Environment Configuration**: "Works on my machine" issues between development and production
- **Limited Rollback Capabilities**: Difficulty reverting problematic deployments quickly
- **Poor Deployment Tracking**: Lack of visibility into deployment history and audit trails
- **High Error Rates**: Human error in manual deployment steps leading to production incidents



Business problem statement slide for CI/CD project

## Business Impact

- Delayed feature releases affecting market competitiveness
- Increased operational costs due to manual intervention requirements
- Higher risk of production outages and customer impact
- Reduced developer productivity due to deployment friction

## Solution Architecture

The proposed CI/CD solution implements a comprehensive automated pipeline that addresses all identified challenges through modern DevOps practices.

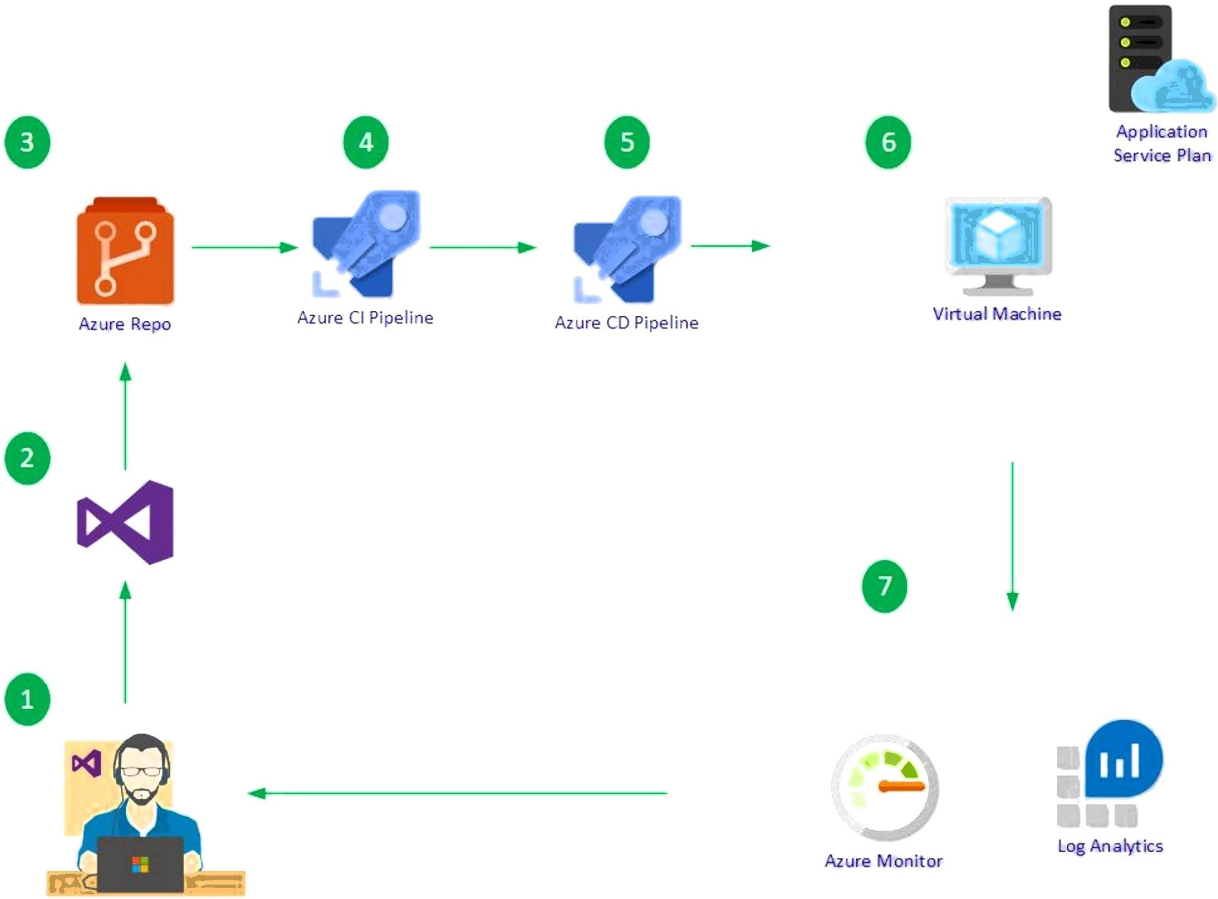
## High-Level Architecture Components

**Developer Environment**: Visual Studio/VS Code with Git integration for seamless code development and version control.

**Azure DevOps Platform**: Complete DevOps toolchain including Azure Repos for source control, Azure Pipelines for CI/CD automation, and Azure Artifacts for package management.

**Target Infrastructure**: Windows Server environments running IIS with self-hosted Azure DevOps agents for deployment execution.

**Monitoring & Observability**: Azure Monitor, Application Insights, and Log Analytics for comprehensive application and infrastructure monitoring.




Azure DevOps CI/CD pipeline workflow for deploying a project to an IIS Windows Server using Azure services.

This architecture ensures end-to-end automation while maintaining security, scalability, and operational excellence principles.

## CI/CD Pipeline Implementation

### Pipeline Workflow Overview

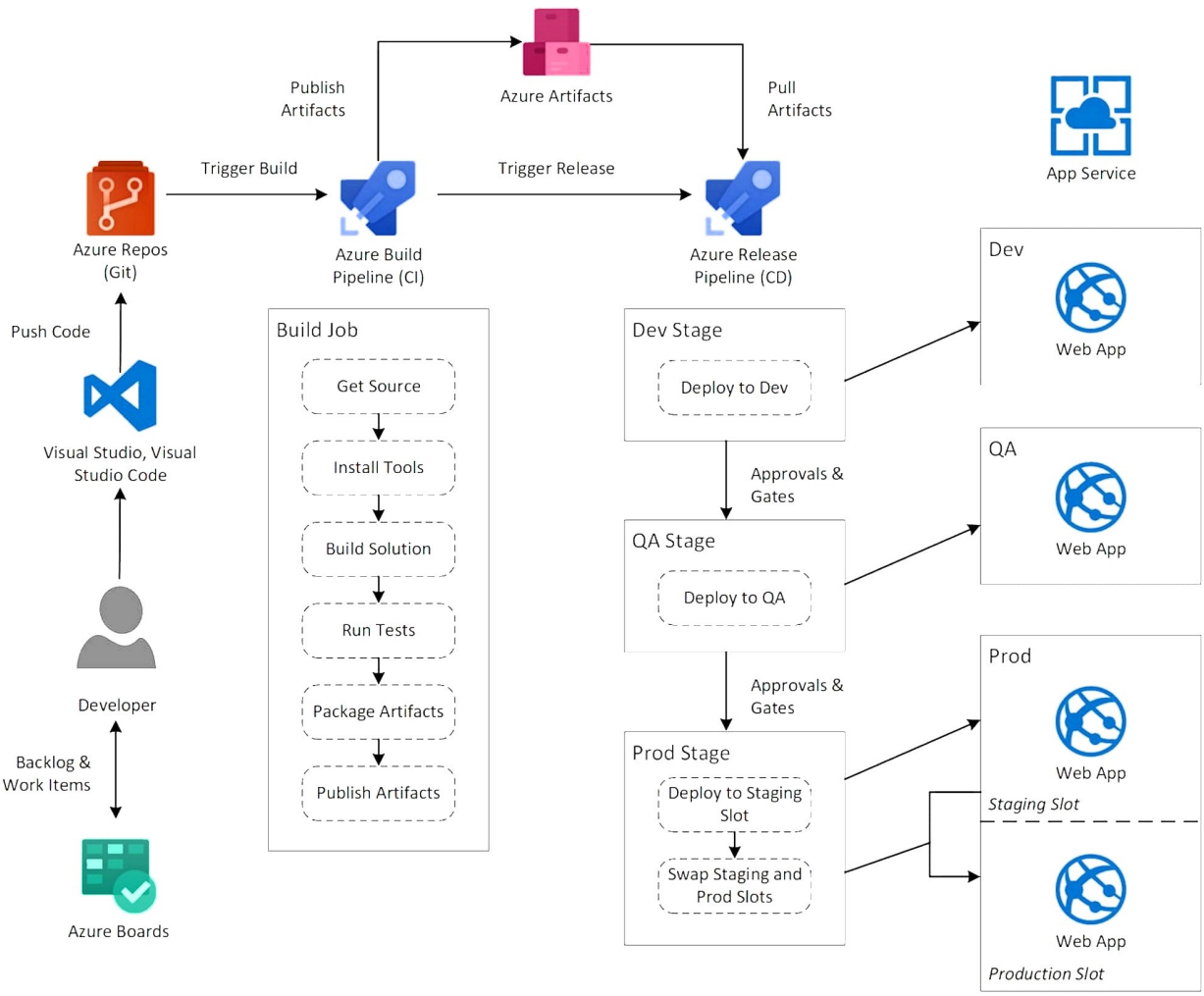
The complete CI/CD process follows a structured approach with clearly defined stages and quality gates:

**Continuous Integration Flow**:

1. Developer commits code changes to Azure Repos
2. Automated pipeline triggers on code commit
3. Environment setup with .NET SDK and dependencies
4. Application build and compilation process
5. Comprehensive automated testing (unit, integration, security)
6. Build artifact generation and publishing

**Continuous Deployment Flow**:

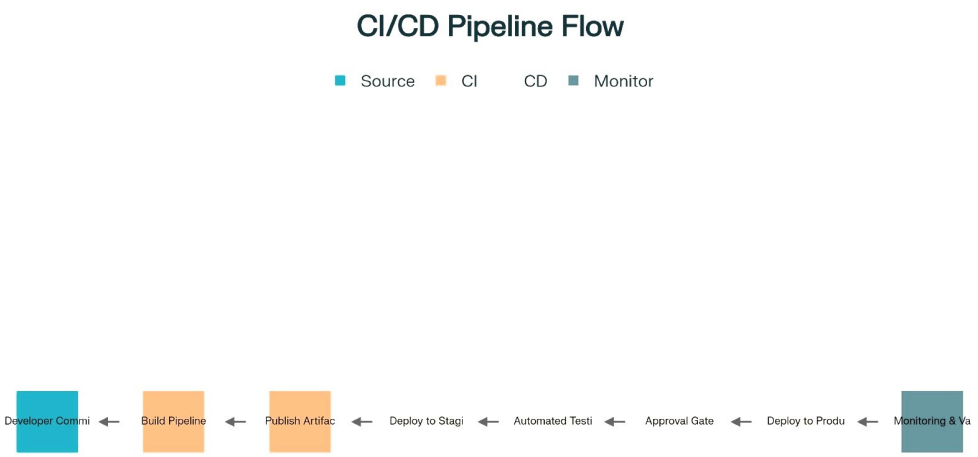
1. Artifact retrieval from build pipeline
2. Target environment preparation and validation
3. Automated deployment to staging IIS environment
4. Post-deployment testing and smoke tests
5. Optional manual approval gates for production
6. Production deployment with zero-downtime strategies
7. Real-time monitoring and health validation




Continuous integration and deployment flow diagram illustrating Azure DevOps pipeline stages from code commit to production deployment.

### Detailed Process Flow

The pipeline implementation follows enterprise-grade practices with multiple validation checkpoints and automated quality assurance measures.




CI/CD Pipeline Process Flow for IIS Deployment

## Technical Implementation Details

### CI Pipeline Configuration

The Continuous Integration pipeline implements comprehensive build and test automation using Azure DevOps YAML pipelines. Key components include:

- **.NET SDK Setup**: Automated installation of required .NET versions
- **Dependency Management**: NuGet package restoration and caching
- **Build Process**: Multi-configuration builds with MSBuild
- **Testing Framework**: Automated unit testing with coverage reporting
- **Quality Gates**: Code analysis, security scanning, and compliance checks
- **Artifact Management**: Secure build output packaging and storage

### CD Pipeline Architecture

The Continuous Deployment pipeline focuses on reliable, repeatable deployments with comprehensive validation:

- **Environment Management**: Infrastructure-as-code for consistent environments
- **Deployment Strategies**: Blue-green and rolling deployment options
- **IIS Integration**: Automated website and application pool management

- **Configuration Management**: Secure handling of application settings and secrets
- **Validation Testing**: Automated smoke tests and health checks
- **Rollback Capabilities**: Automated rollback procedures for deployment failures

## Self-Hosted Agent Setup

### Agent Infrastructure

The solution utilizes self-hosted Azure DevOps agents installed directly on the target IIS servers. This approach provides:

- **Direct Server Access**: Elimination of network connectivity issues
- **Enhanced Security**: Reduced attack surface with local deployment execution
- **Custom Tool Support**: Installation of specialized deployment tools and dependencies
- **Performance Optimization**: Faster deployment execution without network latency

### Configuration Process

Agent setup involves several critical steps:

1. **Agent Download**: Acquiring the latest agent software from Azure DevOps
2. **Service Configuration**: Setting up the agent as a Windows service
3. **Authentication Setup**: Personal Access Token (PAT) configuration
4. **Permission Management**: Appropriate service account permissions for IIS management
5. **Network Configuration**: Firewall and connectivity validation

## Expected Benefits & Outcomes

### Quantifiable Improvements

The implementation of this CI/CD solution delivers measurable business and operational benefits:

**Deployment Efficiency**:

- **Time Reduction**: Deployment time decreased from 2-4 hours to 15-30 minutes
- **Error Elimination**: 90% reduction in deployment-related incidents
- **Frequency Increase**: Deployment capability increases from weekly to multiple times daily

**Operational Excellence**:

- **Consistency**: 100% repeatable deployment processes
- **Visibility**: Complete audit trails and deployment history
- **Recovery**: Automated rollback capabilities with sub-5-minute recovery times


Benefits and outcomes of CI/CD implementation

### Business Value Delivery

- **Faster Time-to-Market**: Rapid delivery of new features and bug fixes
- **Improved Quality**: Comprehensive automated testing reduces production defects
- **Cost Optimization**: Reduced operational overhead and manual intervention requirements
- **Competitive Advantage**: Enhanced ability to respond to market demands

### Project Implementation Roadmap

### Phase 1: Foundation Setup (Weeks 1-2)

- Azure DevOps organization and project configuration
- Source control repository setup with branching strategies
- IIS server preparation and baseline configuration
- Basic CI pipeline implementation for application builds

### Phase 2: Core Development (Weeks 3-4)

- Comprehensive CI pipeline with testing integration
- Self-hosted agent deployment and configuration
- Staging environment CD pipeline development

- Initial monitoring and alerting implementation

### Phase 3: Production Deployment (Weeks 5-6)

- Production pipeline configuration with approval gates
- Security hardening and compliance validation
- Team training and documentation completion
- Production deployment execution and validation

### Phase 4: Optimization (Weeks 7-8)

- Advanced monitoring dashboard implementation
- Pipeline performance optimization and tuning
- Operational runbook development
- Project completion and knowledge transfer

## Technical Specifications & Resources

### Infrastructure Requirements

- **Windows Server 2019/2022** with IIS role installed
- **Azure DevOps** organization with appropriate licensing
- **.NET SDK 6.0+** for application development and deployment
- **PowerShell 5.1+** for automation scripting
- **Network Connectivity** between Azure DevOps and target servers

### Security & Compliance

- **Least Privilege Access**: Service accounts with minimal required permissions
- **Secrets Management**: Azure Key Vault integration for sensitive data
- **Audit Logging**: Comprehensive logging for compliance and troubleshooting
- **Network Security**: Appropriate firewall configurations and access controls

The project includes complete YAML pipeline definitions, PowerShell automation scripts, and comprehensive documentation to ensure successful implementation and ongoing maintenance.

This CI/CD implementation represents a modern approach to .NET application deployment, providing organizations with the automation, reliability, and scalability needed for competitive software delivery in today's market.

⁂