Microsoft Certified: Azure DevOps Engineer Expert – Skills Measured

Design a DevOps strategy

Recommend a migration and consolidation strategy for DevOps tools

- analyze existing artifact (e.g. deployment packages, NuGet) and container repositories
- analyze existing test management tools
- analyze existing work management tools
- recommend migration and integration strategies for artifact repositories, source control, test management, and work management

Design and implement an Agile work management approach

- identify and recommend project metrics, KPIs, and DevOps measurements (e.g. cycle time, lead time, Azure Kubernetes Service, WIP limit)
- implement tools and processes to support Agile work management
- mentor team members on Agile techniques and practices
- recommend an organization structure that supports scaling Agile practices
- recommend in-team and cross-team collaboration mechanisms

Design a quality strategy

- analyze existing quality environment
- identify and recommend quality metrics
- recommend a strategy for feature flag lifecycle
- recommend a strategy for measuring and managing technical debt
- recommend changes to team structure to optimize quality
- recommend performance testing strategy

Design a secure development process

- inspect and validate code base for compliance
- inspect and validate infrastructure for compliance
- recommend a secure development strategy
- recommend tools and practices to integrate code security validation (e.g. static code analysis)
- recommend tools and practices to integrate infrastructure security validation

Design a tool integration strategy
• design a license management strategy (e.g. VSTS users, concurrent pipelines, test environments, open source software licensing, DevOps tools and services, package management licensing)
• design a strategy for end-to-end traceability from work items to working software
• design a strategy for integrating monitoring and feedback to development teams
• design an authentication and access strategy
• design a strategy for integrating on-premises and cloud resources

Implement DevOps development processes

Design a version control strategy

• recommend branching models
• recommend version control systems
• recommend code flow strategy

Implement and integrate source control

• integrate external source control
• integrate source control into third-party continuous integration and continuous deployment (CI/CD) systems

Implement and manage build infrastructure

• implement private and hosted agents
• integrate third party build systems
• recommend strategy for concurrent pipelines
• manage Azure pipeline configuration (e.g. agent queues, service endpoints, pools, webhooks)

Implement code flow

• implement pull request strategies
• implement branch and fork strategies
• configure branch policies

Implement a mobile DevOps strategy

• manage mobile target device sets and distribution groups
• manage target UI test device sets
• provision tester devices for deployment
• create public and private distribution groups

Managing application configuration and secrets
implement a secure and compliant development process
implement general (non-secret) configuration data
manage secrets, tokens, and certificates
implement applications configurations (e.g. Web App, Azure Kubernetes Service, containers)
implement secrets management (e.g. Web App, Azure Kubernetes Service, containers, Azure Key Vault)
implement tools for managing security and compliance in the pipeline

Implement continuous integration

Manage code quality and security policies

- monitor code quality
- configure build to report on code coverage
- manage automated test quality
- manage test suites and categories
- monitor quality of tests
- integrate security analysis tools (e.g. SonarQube, WhiteSource Bolt, Open Web Application Security Project)

Implement a container build strategy

- create deployable images (e.g. Docker, Azure Container Registry)
- analyze and integrate Docker multi-stage builds

Implement a build strategy

- design build triggers, tools, integrations, and workflow
- implement a hybrid build process
- implement multi-agent builds
- recommend build tools and configuration (e.g. Azure Pipelines, Jenkins)
- set up an automated build workflow

Implement continuous delivery

Design a release strategy

- recommend release tools
- identify and recommend release approvals and gates
- recommend strategy for measuring quality of release and release process
- recommend strategy for release notes and documentation
- select appropriate deployment pattern
Set up a release management workflow

- automate inspection of health signals for release approvals by using release gates
- configure automated integration and functional test execution
- create a release pipeline (e.g. Azure Kubernetes Service, Service Fabric, WebApp)
- create multi-phase release pipelines
- integrate secrets with release pipeline
- provision and configure environments
- manage and modularize tasks and templates (e.g. task and variable groups)

Implement an appropriate deployment pattern

- implement blue-green deployments
- implement canary deployments
- implement progressive exposure deployments
- scale a release pipeline to deploy to multiple endpoints (e.g. deployment groups, Azure Kubernetes Service, Service Fabric)

Implement dependency management

Design a dependency management strategy

- recommend artifact management tools and practices (Azure Artifacts, npm, maven, Nuget)
- abstract common packages to enable sharing and reuse
- inspect codebase to identify code dependencies that can be converted to packages
- identify and recommend standardized package types and versions across the solution
- refactor existing build pipelines to implement version strategy that publishes packages

Manage security and compliance

- inspect open source software packages for security and license compliance to align with corporate standards (e.g. GPLv3)
- configure build pipeline to access package security and license rating (e.g. Black Duck, White Source)
- configure secure access to package feeds

Implement application infrastructure

Design an infrastructure and configuration management strategy

- analyze existing and future hosting infrastructure
- analyze existing Infrastructure as Code (IaC) technologies
- design a strategy for managing technical debt on templates
• design a strategy for using transient infrastructure for parts of a delivery lifecycle
• design a strategy to mitigate infrastructure state drift

**Implement infrastructure as code (IaC)**

• create nested resource templates
• manage secrets in resource templates
• provision Azure resources
• recommend an Infrastructure as Code (IaC) strategy
• recommend appropriate technologies for configuration management (ARM Templates, Terraform, Chef, Puppet, Ansible)

**Manage Azure Kubernetes Service infrastructure**

• provision Azure Kubernetes Service (e.g. using ARM templates, CLI)
• create deployment file for publishing to Azure Kubernetes Service (e.g. kubectl, Helm)
• develop a scaling plan

**Implement infrastructure compliance and security**

• implement compliance and security scanning
• prevent drift by using configuration management tools
• automate configuration management by using PowerShell Desired State Configuration (DSC)
• automate configuration management by using a VM Agent with custom script extensions
• set up an automated pipeline to inspect security and compliance

**Implement continuous feedback**

**Recommend and design system feedback mechanisms**

• design practices to measure end-user satisfaction (e.g. Send a Smile, app analytics)
• design processes to capture and analyze user feedback from external sources (e.g. Twitter, Reddit, Help Desk)
• design routing for client application crash report data
• recommend monitoring tools and technologies
• recommend system and feature usage tracking tools

**Implement process for routing system feedback to development teams**

• configure crash report integration for client applications
• develop monitoring and status dashboards
• implement routing for client application crash report data
• implement tools to track system usage, feature usage, and flow
• integrate and configure ticketing systems with development team’s work management system (e.g. IT Service Management connector, ServiceNow Cloud Management, App Insights work items)

**Optimize feedback mechanisms**

• analyze alerts to establish a baseline
• analyze telemetry to establish a baseline
• perform live site reviews and capture feedback for system outages
• perform ongoing tuning to reduce meaningless or non-actionable alerts