Exam AZ-400: Designing and Implementing Microsoft DevOps Solutions – Skills Measured

This exam was updated on June 15, 2020. Following the current exam guide, we have included the former exam guide as well as a table in the second part of this OD that shows a side by side comparison of the former objectives and the new ones.

Audience Profile

Candidates for this exam should have subject matter expertise working with people, processes, and technologies to continuously deliver business value.

Responsibilities for this role include designing and implementing strategies for collaboration, code, infrastructure, source control, security, compliance, continuous integration, testing, delivery, monitoring, and feedback.

A candidate for this exam must be familiar with both Azure administration and development and must be expert in at least one of these areas.

Skills Measured

NOTE: The bullets that appear below each of the skills measured are intended to illustrate how we are assessing that skill. This list is not definitive or exhaustive.

NOTE: In most cases, exams do NOT cover preview features, and some features will only be added to an exam when they are GA (General Availability).

Develop an Instrumentation Strategy (5-10%)

Design and implement logging

- assess and Configure a log framework
- design a log aggregation and storage strategy (e.g. Azure storage)
- design a log aggregation using Azure Monitor
- manage access control to logs (workspace-centric/resource-centric)
- integrate crash analytics (App Center Crashes, Crashlytics)

Design and implement telemetry

- design and implement distributed tracing
- inspect application performance indicators
- inspect infrastructure performance indicators
- define and measure key metrics (CPU, memory, disk, network)
- implement alerts on key metrics (email, SMS, webhooks, Teams/Slack)

- integrate user analytics (e.g. Application Insights funnels, Visual Studio App Center, TestFlight, Google Analytics)

Integrate logging and monitoring solutions

- configure and integrate container monitoring (Azure Monitor, Prometheus, etc.)
- configure and integrate with monitoring tools (Azure Monitor Application Insights, Dynatrace, New Relic, Naggios, Zabbix)
- create feedback loop from platform monitoring tools (e.g. Azure Diagnostics VM extensions, Azure Platform Logs, Event Grid)
- manage Access control to the monitoring platform

Develop a Site Reliability Engineering (SRE) strategy (5-10%)

Develop an actionable alerting strategy

- identify and recommend metrics on which to base alerts
- implement alerts using appropriate metrics
- implement alerts based on appropriate log messages
- implement alerts based on application health checks
- analyze combinations of metrics
- develop communication mechanism to notify users of degraded systems
- implement alerts for self-healing activities (e.g. scaling, failovers)

Design a failure prediction strategy

- analyze behavior of system with regards to load and failure conditions
- calculate when a system will fail under various conditions
- measure baseline metrics for system
- recommend the appropriate tools for a failure prediction strategy

Design and implement a health check

- analyze system dependencies to determine which dependency should be included in health check
- calculate healthy response timeouts based on SLO for the service
- design approach for partial health situations
- integrate health check with compute environment
- implement different types of health checks (liveness, startup, shutdown)

Develop a security and compliance plan (10-15%)

Design an authentication and authorization strategy

- design an access solution (Azure AD Privileged Identity Management (PIM), Azure AD Conditional Access, MFA)
- organize the team using Azure AD groups
- implement Service Principals and Managed Identity
- configure service connections

Design a sensitive information management strategy

- evaluate and configure vault solution (Azure Key Vault, Hashicorp Vault)
- generate security certificates
- design a secrets storage and retrieval strategy
- formulate a plan for deploying secret files as part of a release

Develop security and compliance

- automate dependencies scanning for security (container scanning, OWASP)
- automate dependencies scanning for compliance (licenses: MIT, GPL)
- assess and report risks
- design a source code compliance solution (e.g. GitHub security, pipeline-based scans, Git hooks, SonarQube)

Design governance enforcement mechanisms

- implement Azure policies to enforce organizational requirements
- implement container scanning (e.g. static scanning, malware, crypto mining)
- design and implement Azure Container Registry Tasks (eg. Azure Policy)
- design break-the-glass strategy for responding to security incidents

Manage source control (10-15%)

Develop a modern source control strategy

- integrate/migrate disparate source control systems (e.g. GitHub, Azure Repos)
- design authentication strategies
- design approach for managing large binary files (e.g. Git LFS)
- design approach for cross repository sharing (e.g. Git sub-modules, packages)
- implement workflow hooks

Plan and implement branching strategies for the source code

- define Pull Requests (PR) guidelines to enforce work item correlation
- implement branch merging restrictions (e.g. branch policies, branch protections, manual, etc.)

- define branch strategy (e.g. trunk based, feature branch, release branch, GitHub flow)
- design and implement a PR workflow (code reviews, approvals)
- enforce static code analysis for code-quality consistency on PR

Configure repositories

- configure permissions in the source control repository
- organize the repository with git-tags
- plan for handling oversized repositories
- plan for content recovery in all repository states
- purge data from source control

Integrate source control with tools

- integrate GitHub with DevOps pipelines
- integrate GitHub with identity management solutions (Azure AD)
- design for GitOps
- design for ChatOps
- integrate source control artifacts for human consumption (e.g. Git changelog)

Facilitate communication and collaboration (10-15%)

Communicate deployment and release information with business stakeholders

- create dashboards combining boards, pipelines (custom dashboards on Azure DevOps)
- design a cost management communication strategy
- integrate release pipeline with work item tracking (e.g. AZ DevOps, Jira)
- integrate GitHub as repository with Azure Boards
- communicate user analytics

Generate DevOps process documentation

- design onboarding process for new employees
- assess and document external dependencies (e.g. integrations, packages)
- assess and document artifacts (version, release notes)

Automate communication with team members

- integrate monitoring tools with communication platforms (e.g. Teams, Slack, dashboards)
- notify stakeholders about key metrics, alerts, severity using communication platforms (e.g. Email, SMS, Slack, Teams)
- integrate build and release with communication platforms (e.g. build fails, release fails)

Define and implement continuous integration (20-25%)

Design build automation

- integrate the build pipeline with external tools (e.g., Dependency and security scanning, Code coverage)
- implement quality gates (e.g. code coverage, internationalization, peer review)
- design a testing strategy (e.g. integration, load, fuzz, API, chaos)
- integrate multiple tools (e.g. GitHub Actions, Azure Pipeline, Jenkins)

Design a package management strategy

- recommend package management tools (e.g. GitHub Packages, Azure Artifacts, Azure Automation Runbooks Gallery, Nuget, Jfrog, Artifactory)
- design an Azure Artifacts implementation including linked feeds
- design versioning strategy for code assets (e.g. SemVer, date based)
- plan for assessing and updating and reporting package dependencies (GitHub Automated Security Updates, NuKeeper, GreenKeeper)
- design a versioning strategy for packages (e.g. SemVer, date based)
- design a versioning strategy for deployment artifacts

Design an application infrastructure management strategy

- assess a configuration management mechanism for application infrastructure
- define and enforce desired state configuration for environments

Implement a build strategy

- design and implement build agent infrastructure (include cost, tool selection, licenses, maintainability)
- develop and implement build trigger rules
- develop build pipelines
- design build orchestration (products that are composed of multiple builds)
- integrate configuration into build process
- develop complex build scenarios (e.g. containerized agents, hybrid, GPU)

Maintain build strategy

- monitor pipeline health (failure rate, duration, flaky tests)
- optimize build (cost, time, performance, reliability)
- analyze CI load to determine build agent configuration and capacity
- manage pipeline health
- identify the number of agents and jobs to run in parallel

- investigate test failures

Design a process for standardizing builds across organization

- manage self-hosted build agents (VM templates, containerization, etc.)
- create reuseable build subsystems (YAML templates, Task Groups, Variable Groups, etc.)

Define and implement a continuous delivery and release management strategy (10-15%)

Develop deployment scripts and templates

- recommend a deployment solution (e.g. GitHub Actions, Azure Pipelines, Jenkins, CircleCI, etc.)
- design and implement Infrastructure as code (ARM, Terraform, PowerShell, CLI)
- develop application deployment process (container, binary, scripts)
- develop database deployment process (migrations, data movement, ETL)
- integrate configuration management as part of the release process
- develop complex deployments (IoT, Azure IoT Edge, mobile, App Center, DR, multi-region, CDN, sovereign cloud, Azure Stack, etc.)

Implement an orchestration automation solution

- combine release targets depending on release deliverable (e.g., Infrastructure, code, assets, etc.)
- design the release pipeline to ensure reliable order of dependency deployments
- organize shared release configurations and process (YAML templates, variable groups)
- design and implement release gates and approval processes

Plan the deployment environment strategy

- design a release strategy (blue/green, canary, ring)
- implement the release strategy (using deployment slots, load balancer configurations, Azure Traffic Manager, feature toggle, etc.)
- select the appropriate desired state solution for a deployment environment (PowerShell DSC, Chef, Puppet, etc.)
- plan for minimizing downtime during deployments (VIP Swap, Load balancer, rolling deployments, etc.)
- design a hotfix path plan for responding to high priority code fixes

The exam guide below shows the former guide for this exam. There is also a comparison table below the guide.

Audience Profile

Candidates for this exam should have subject matter expertise working with people, processes, and technologies to continuously deliver business value.

Responsibilities for this role include designing and implementing strategies for collaboration, code, infrastructure, source control, security, compliance, continuous integration, testing, delivery, monitoring, and feedback.

A candidate for this exam must be familiar with both Azure administration and development and must be expert in at least one of these areas.

Skills Measured

NOTE: The bullets that appear below each of the skills measured are intended to illustrate how we are assessing that skill. This list is not definitive or exhaustive.

NOTE: In most cases, exams do NOT cover preview features, and some features will only be added to an exam when they are GA (General Availability).

Design a DevOps strategy (20-25%)

Recommend a migration and consolidation strategy for DevOps tools

- analyze existing artifact (e.g., deployment packages, NuGet, Maven, npm) 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, 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, third-party 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 (20-25%)

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 (10-15%)

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, White Source Bolt, Open Web Application Security Project)

Implement a container build strategy

- create deployable images (e.g., Docker, Hub, 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 (10-15%)

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 (5-10%)

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 (15-20%)

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 (e.g., 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 (10-15%)

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

Comparison between original study guide and new study guide

Objective – AZ-400	Objective mapping	Objective – AZ-400 (NEW)
Design a DevOps Strategy (20-25%)		
Recommend a migration and consolidation strategy for DevOps tools	Maps closely	**Design a package management strategy** **Implement a build strategy** **Develop a modern source control strategy**
Design and implement an Agile work management approach	Maps loosely	**Communicate deployment and release information with business stakeholders**

Design a quality strategy	n/a	
Design a secure development process	Maps closely	**Develop security and compliance**
Design a tool integration strategy	Maps closely	**Design build automation** **Automate communication with team members** **Integrate source control with tools**
Implement DevOps Development Processes (20-25%)		
Design a version control strategy	Maps closely	**Develop a modern source control strategy** **Plan and implement branching strategies for the source code**
Implement and integrate source control	Maps closely	**Plan and implement branching strategies for the source code** **Integrate source control with tools**
Implement and manage build infrastructure	Maps closely	**Implement a build strategy**
Implement code flow	Maps closely	**Plan and implement branching strategies for the source code**
Implement a mobile DevOps strategy		
Managing application configuration and secrets	Maps closely	**Design a sensitive information management strategy**
Implement Continuous Integration (10-15%)		
Manage code quality and security policies	Maps closely	**Design build automation**
Implement a container build strategy	Maps closely	**Design a package management strategy** **Develop deployment scripts and templates**
Implement a build strategy	Maps closely	**Implement a build strategy** **Design build automation**
Implement Continuous Delivery (10-15%)		
Design a release strategy	Maps loosely	**Develop deployment scripts and templates**

		Implement an orchestration automation solution
Set up a release management workflow	Maps closely	**Plan the deployment environment strategy**
Implement an appropriate deployment pattern	Maps closely	**Plan the deployment environment strategy**
Implement Dependency Management (5-10%)		
Design a dependency management strategy	Maps closely	**Design a package management strategy**
Manage security and compliance	Maps closely	**Develop security and compliance**
Implement Application Infrastructure (15-20%)		
Design an infrastructure and configuration management strategy	Maps loosely	**Plan the deployment environment strategy**
Implement Infrastructure as Code (IaC)	Maps closely	**Develop deployment scripts and templates**
Manage Azure Kubernetes Service infrastructure	n/a	
Implement infrastructure compliance and security	Maps loosely	**Design build automation**
Implement Continuous Feedback (10-15%)		
Recommend and design system feedback mechanisms	Maps closely	**Design and implement logging** **Integrate logging and monitoring solutions**
Implement process for routing system feedback to development teams	Maps closely	**Automate communication with team members**
Optimize feedback mechanisms	Maps closely	**Develop an actionable alerting strategy** **Design a failure prediction strategy** **Design and implement a health check**