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

The English language version of this exam will be updated on July 13, 2022. If a localized version of this exam is available, it will be updated approximately 8 weeks after the English version is updated.

Following the current exam guide, we have included a table that summarizes the changes that will be made to the exam on that date. We have also included the upcoming exam guide for reference purposes.

NOTE: Passing score: 700. Learn more about exam scores.

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 follow each of the skills measured are intended to illustrate how we are assessing that skill. Related topics may be covered in the exam.

NOTE: Most questions cover features that are general availability (GA). The exam may contain questions on Preview features if those features are commonly used.

Develop an instrumentation strategy (5—10%)

Design and implement logging

- assess and configure a logging framework
- design a log aggregation and storage strategy (e.g., Azure storage)
• design a log aggregation and query strategy (e.g., Azure Monitor, Splunk, Exabeam, LogRhythm)
• interrogate Log Analytics logs using basic Kusto (KQL) queries
• 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, Nagios, Zabbix)
• create feedback loop from platform monitoring tools (e.g., Azure Diagnostics extension, Log Analytics agent, 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
• leverage Application Insights Smart Detection and Dynamic thresholds in Azure Monitor
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
- design approach for piecemeal recovery (e.g., to improve recovery time objective strategies)
- integrate health check with compute environment
- implement different types of health checks (container 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, Azure AD B2B, etc.)
- implement Service Principals and Managed Identity
- design an application access solution using Azure AD B2C
- configure service connections

Design a sensitive information management strategy

- evaluate and configure vault solution (Azure Key Vault, Hashicorp Vault)
- manage security certificates
- design a secrets storage and retrieval strategy (KeyVault secrets, GitHub secrets, Azure Pipelines secrets)
- 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 Code scanning, GitHub Secret scanning, pipeline-based scans, Git hooks, SonarQube, Dependabot, etc.)

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
- 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
- design approach for efficient code reviews (e.g., GitHub code review assignments, schedule reminders, Pull Analytics)

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)
- integrate GitHub Codespaces

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, ServiceNow)
• 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 and project management platforms (e.g., Email, SMS, Slack, Teams, ServiceNow, etc.)
• integrate build and release with communication platforms (e.g., build fails, release fails)
• integrate GitHub pull request approvals via mobile apps

**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

Design a process for standardizing builds across organization

- Manage self-hosted build agents (VM templates, containerization, etc.)
- Create reusable 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, Azure App Configuration)
- 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 following table summarizes the changes that will be implemented on July 13, 2022 to the English language version of this exam. Following the table, the upcoming exam guide is included.

<table>
<thead>
<tr>
<th>Objective</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audience Profile</td>
<td>Major Revision</td>
</tr>
<tr>
<td>Develop an instrumentation strategy</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Design and implement logging</td>
<td>Removed</td>
</tr>
<tr>
<td>Design and implement telemetry</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Integrate logging and monitoring solutions</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Develop a Site Reliability Engineering (SRE) Strategy</td>
<td>Removed</td>
</tr>
<tr>
<td>Develop an actionable alerting strategy</td>
<td>Major revision</td>
</tr>
<tr>
<td>Design a failure prediction strategy</td>
<td>Removed</td>
</tr>
<tr>
<td>Design and implement a health check</td>
<td>Removed</td>
</tr>
<tr>
<td>Develop a security and compliance plan</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Design an authentication and authorization strategy</td>
<td>Removed</td>
</tr>
<tr>
<td>Design a sensitive information management strategy</td>
<td>Major Revision</td>
</tr>
<tr>
<td>Develop security and compliance</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Design governance enforcement mechanisms</td>
<td>Removed</td>
</tr>
<tr>
<td>Manage source control</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Develop a modern source control strategy</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Plan and implement branching strategies for the source code</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Configure repositories</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Integrate source control with tools</td>
<td>Removed</td>
</tr>
<tr>
<td>Integrate on-premises network with an Azure virtual network</td>
<td>Removed</td>
</tr>
<tr>
<td>Facilitate communication and collaboration</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Communicate deployment and release information with business stakeholders</td>
<td>Major Revision</td>
</tr>
<tr>
<td>Task</td>
<td>Revision</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Generate DevOps process documentation</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Automate communication with team members</td>
<td>Major Revision</td>
</tr>
<tr>
<td>Define and implement continuous integration</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Design build automation</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Design a package management strategy</td>
<td>Major Revision</td>
</tr>
<tr>
<td>Design an application infrastructure management strategy</td>
<td>Major Revision</td>
</tr>
<tr>
<td>Implement a build strategy</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Maintain build strategy</td>
<td>Minor Revision</td>
</tr>
<tr>
<td>Design a process for standardizing builds across organization</td>
<td>Major Revision</td>
</tr>
<tr>
<td>Define and implement a continuous delivery and release management strategy</td>
<td>Major Revision</td>
</tr>
<tr>
<td>Develop deployment scripts and templates</td>
<td>Major Revision</td>
</tr>
<tr>
<td>Implement an orchestration automation solution</td>
<td>Removed</td>
</tr>
<tr>
<td>Plan the deployment environment strategy</td>
<td>Major Revision</td>
</tr>
</tbody>
</table>

**Audience Profile**

DevOps engineers are developers or infrastructure administrators who also have subject matter expertise in working with people, processes, and products to enable continuous delivery of value in organizations.

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

DevOps engineers work on cross-functional teams that include developers, site reliability engineers, and Azure administrators.

DevOps engineers must have experience with administering and developing in Azure, with strong skills in at least one of these areas. They should be familiar with both Azure DevOps and GitHub.

**Skills Measured**

NOTE: The bullets that follow each of the skills measured are intended to illustrate how we are assessing that skill. Related topics may be covered in the exam.

NOTE: Most questions cover features that are general availability (GA). The exam may contain questions on Preview features if those features are commonly used.
Configure processes and communications (10—15%)

Configure activity traceability and flow of work

- plan and implement a structure for the flow of work and feedback cycles
- identify appropriate metrics related to flow of work, such as cycle times, time to recovery, and lead time
- integrate pipelines with work item tracking tools, such as Azure DevOps and GitHub
- implement traceability policies decided by development
- integrate a repository with Azure Boards

Configure collaboration and communication

- communicate actionable information by using custom dashboards in Azure DevOps
- document a project by using tools, such as wikis and process diagrams
- configure release documentation, including release notes and API documentation
- automate creation of documentation from Git history
- configure notifications by using webhooks

Design and implement source control (15—20%)

Design and implement a source control strategy

- design and implement an authentication strategy
- design a strategy for managing large files, including Git LFS and git-fat
- design a strategy for scaling and optimizing a Git repository, including Scalar and cross-repository sharing
- implement workflow hooks

Plan and implement branching strategies for the source code

- design a branch strategy, including trunk-based, feature branch, and release branch
- design and implement a pull request workflow by using branch policies and branch protections
- implement branch merging restrictions by using branch policies and branch protections

Configure and manage repositories

- integrate GitHub repositories with Azure Pipelines, one of the services in Azure DevOps
- configure permissions in the source control repository
- configure tags to organize the source control repository
- recover data by using Git commands
- purge data from source control
Design and implement build and release pipelines (40—45%)

**Design and implement pipeline automation**
- integrate pipelines with external tools, including dependency scanning, security scanning, and code coverage
- design and implement quality and release gates, including security and governance
- design integration of automated tests into a pipeline
- design and implement a comprehensive testing strategy
- implement orchestration of tools, such as GitHub Actions and Azure Pipelines

**Design and implement a package management strategy**
- design a package management implementation that uses Azure Artifacts, GitHub Packages, NuGet, and npm
- design and implement package feeds, including upstream sources
- design and implement a dependency versioning strategy for code assets and packages, including semantic versioning and date-based
- design and implement a versioning strategy for pipeline artifacts

**Design and implement pipelines**
- select a deployment automation solution, including GitHub Actions and Azure Pipelines
- design and implement an agent infrastructure, including cost, tool selection, licenses, connectivity, and maintainability
- develop and implement pipeline trigger rules
- develop pipelines, including classic and YAML
- design and implement a strategy for job execution order, including parallelism and multi-stage
- develop complex pipeline scenarios, such as containerized agents and hybrid
- configure and manage self-hosted agents, including virtual machine (VM) templates and containerization
- create reusable pipeline elements, including YAML templates, task groups, variables, and variable groups
- design and implement checks and approvals by using YAML environments

**Design and implement deployments**
- design a deployment strategy, including blue/green, canary, ring, progressive exposure, feature flags, and A/B testing
- design a pipeline to ensure reliable order of dependency deployments
- plan for minimizing downtime during deployments by using VIP swap, load balancer, and rolling deployments
• design a hotfix path plan for responding to high-priority code fixes
• implement load balancing for deployment, including Azure Traffic Manager and the Web Apps feature of Azure App Service
• implement feature flags by using Azure App Configuration Feature Manager
• implement application deployment by using containers, binary, and scripts

Design and implement infrastructure as code (IaC)

• recommend a configuration management technology for application infrastructure
• implement a configuration management strategy for application infrastructure, including IaC
• define an IaC strategy, including source control and automation of testing and deployment
• design and implement desired state configuration for environments, including Azure Automation State Configuration, Azure Resource Manager, Bicep, and Azure Policy guest configuration

Maintain pipelines

• monitor pipeline health, including failure rate, duration, and flaky tests
• optimize pipelines for cost, time, performance, and reliability
• analyze pipeline load to determine agent configuration and capacity
• design and implement a retention strategy for pipeline artifacts and dependencies

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

Design and implement a strategy for managing sensitive information in automation

• implement and manage service connections
• implement and manage personal access tokens
• implement and manage secrets, keys, and certificates by using Azure Key Vault, GitHub secrets, and Azure Pipelines secrets
• design and implement a strategy for managing sensitive files during deployment
• design pipelines to prevent leakage of sensitive information

Automate security and compliance scanning

• automate analysis of source code by using GitHub code scanning, GitHub secrets scanning, pipeline-based scans, and SonarQube
• automate security scanning, including container scanning and OWASP ZAP
• automate analysis of licensing, vulnerabilities, and versioning of open-source components by using WhiteSource and GitHub Dependency Scanning
Implement an instrumentation strategy (10—15%)

Configure monitoring for a DevOps environment

- configure and integrate monitoring by using Azure Monitor
- configure and integrate with monitoring tools, such as Azure Monitor and Application Insights
- manage access control to the monitoring platform
- configure alerts for pipeline events

Analyze metrics

- inspect distributed tracing by using Application Insights
- inspect application performance indicators
- inspect infrastructure performance indicators, including CPU, memory, disk, and network
- identify and monitor metrics for business value
- analyze usage metrics by using Application Insight
- interrogate logs using basic Kusto Query Language (KQL) queries