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

This exam was updated on September 29, 2021. Following the current exam guide, we have included a version of the exam guide with Track Changes set to "On," showing the changes that were made to the exam on that date.

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

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. This list is NOT definitive or exhaustive.

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)
- 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 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 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, 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 exam guide below shows the changes that were implemented on September 29, 2021.

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. This list is NOT definitive or exhaustive.

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)
- 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 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 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, 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