Study guide for Exam AZ-400: Designing and Implementing Microsoft DevOps Solutions

Purpose of this document

This study guide should help you understand what to expect on the exam and includes a summary of the topics the exam might cover and links to additional resources. The information and materials in this document should help you focus your studies as you prepare for the exam.

Useful links	Description
Review the skills measured as of May 2, 2023	This list represents the skills measured AFTER the date provided. Study this list if you plan to take the exam AFTER that date.
Review the skills measured prior to May 2, 2023	Study this list of skills if you take your exam PRIOR to the date provided.
Change log	You can go directly to the change log if you want to see the changes that will be made on the date provided.
How to earn the certification	Some certifications only require passing one exam, while others require passing multiple exams.
Certification renewal	Microsoft associate, expert, and specialty certifications expire annually. You can renew by passing a **free** online assessment on Microsoft Learn.
Your Microsoft Learn profile	Connecting your certification profile to Microsoft Learn allows you to schedule and renew exams and share and print certificates.
Exam scoring and score reports	A score of 700 or greater is required to pass.
Exam sandbox	You can explore the exam environment by visiting our exam sandbox.

Useful links	Description
Request accommodations	If you use assistive devices, require extra time, or need modification to any part of the exam experience, you can request an accommodation.
Take a practice test	Are you ready to take the exam or do you need to study a bit more?

Updates to the exam

Our exams are updated periodically to reflect skills that are required to perform a role. We have included two versions of the Skills Measured objectives depending on when you are taking the exam.

We always update the English language version of the exam first. Some exams are localized into other languages, and those are updated approximately eight weeks after the English version is updated. While Microsoft makes every effort to update localized versions of exams as noted, there may be times when localized versions of an exam are not updated on this schedule. Other available languages are listed in the **Schedule Exam** section of the **Exam Details** webpage. If the exam isn't available in your preferred language, you can request an additional 30 minutes to complete the exam.

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.

Skills measured as of May 2, 2023

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.

- Configure processes and communications (10–15%)
- Design and implement source control (15–20%)
- Design and implement build and release pipelines (40–45%)
- Develop a security and compliance plan (10–15%)
- Implement an instrumentation strategy (10–15%)

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 Azure Pipelines and GitHub Actions with work item tracking tools
- 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 Boards
- 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
- 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 pipelines
- Design and implement a comprehensive testing strategy (including local tests, unit tests, integration tests, and load tests)
- Design and implement UI testing
- 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 Automanage Machine 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 Mend Bolt 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 Insights
- Interrogate logs using basic Kusto Query Language (KQL) queries

Study resources

We recommend that you train and get hands-on experience before you take the exam. We offer self-study options and classroom training as well as links to documentation, community sites, and videos.

Study resources	Links to learning and documentation
Get trained	Choose from self-paced learning paths and modules or take an instructor led course
Find documentation	DevOps resource center Azure DevOps documentation Azure Boards Azure Key Vault Keys, Secrets, and Certificates Overview Azure Monitor Azure Pipelines Azure Repos Work with Azure DevOps and GitHub
Ask a question	Microsoft Q&A | Microsoft Docs
Get community support	Azure DevOps - Microsoft Tech Community

Study resources	Links to learning and documentation
Follow Microsoft Learn	Microsoft Learn - Microsoft Tech Community
Find a video	Exam Readiness Zone Microsoft Learn Shows

Change log

Key to understanding the table: The topic groups (also known as functional groups) are in bold typeface followed by the objectives within each group. The table is a comparison between the two versions of the exam skills measured and the third column describes the extent of the changes.

Skill area prior to May 2, 2023	Skill area as of May 2, 2023	Change
Audience profile		No change
Configure processes and communications	**Configure processes and communications**	No change
Configure activity traceability and flow of work	Configure activity traceability and flow of work	Minor
Configure collaboration and communication	Configure collaboration and communication	Minor
Design and implement source control	**Design and implement source control**	No change
Design and implement a source control strategy	Design and implement a source control strategy	No change
Plan and implement branching strategies for the source code	Plan and implement branching strategies for the source code	No change
Configure and manage repositories	Configure and manage repositories	Minor
Design and implement build and release pipelines	**Design and implement build and release pipelines**	No change
Design and implement pipeline automation	Design and implement pipeline automation	Minor
Design and implement a package management strategy	Design and implement a package management strategy	No change
Design and implement pipelines	Design and implement pipelines	No change

Skill area prior to May 2, 2023	Skill area as of May 2, 2023	Change
Design and implement deployments	Design and implement deployments	No change
Design and implement infrastructure as code (IaC)	Design and implement infrastructure as code (IaC)	No change
Maintain pipelines	Maintain pipelines	No change
Develop a security and compliance plan	**Develop a security and compliance plan**	No change
Design and implement a strategy for managing sensitive information in automation	Design and implement a strategy for managing sensitive information in automation	No change
Automate security and compliance scanning	Automate security and compliance scanning	No change
Implement an instrumentation strategy	**Implement an instrumentation strategy**	No change
Configure monitoring for a DevOps environment	Configure monitoring for a DevOps environment	No change
Analyze metrics	Analyze metrics	No change

Skills measured prior to May 2, 2023

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.

- Configure processes and communications (10–15%)
- Design and implement source control (15–20%)
- Design and implement build and release pipelines (40–45%)
- Develop a security and compliance plan (10–15%)

- Implement an instrumentation strategy (10–15%)

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 (including local tests, unit tests, integration tests, and load tests)
- Design and implement UI testing
- 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 Automanage Machine 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 Mend Bolt 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 Insights
- Interrogate logs using basic Kusto Query Language (KQL) queries