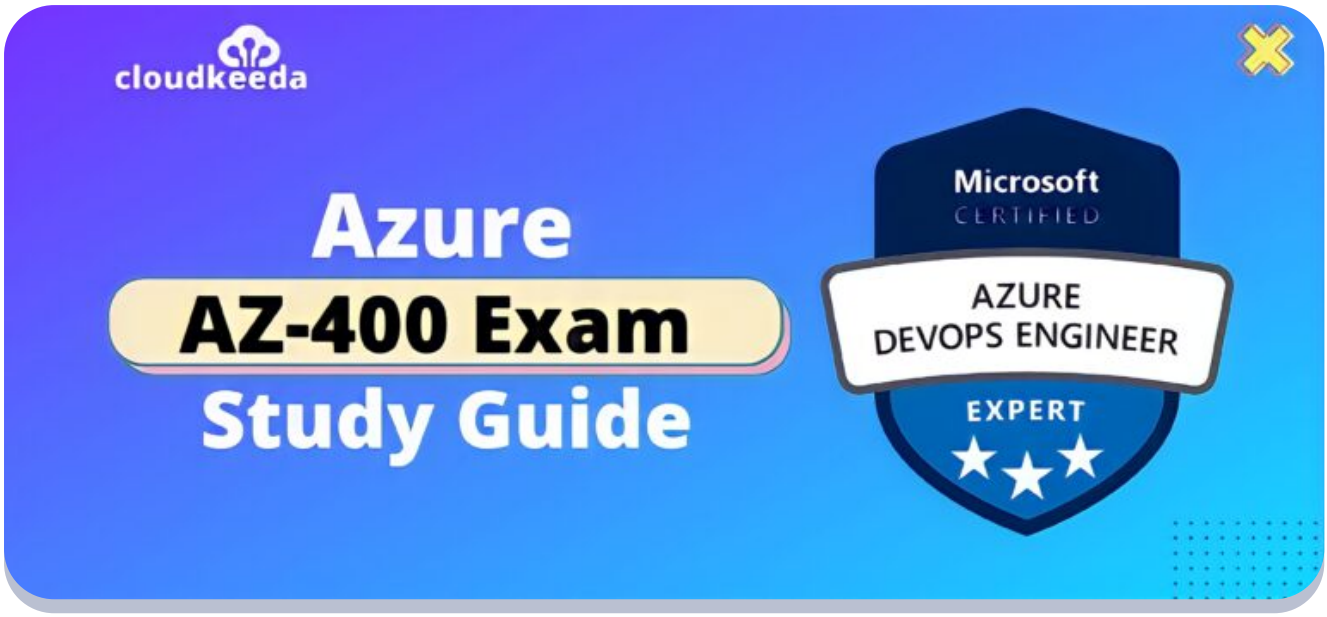


AZ-400: Microsoft Azure DevOps Engineer Expert Exam Study Guide

May 2, 2022 by manish

5/5 - (9 votes)

DevOps practitioners are recently in high demand in the IT business since organizations that use these practices are overwhelmingly high functioning. Azure DevOps engineer capable of merging processes, people, and technology to produce services and solutions that satisfy corporate objectives was required for the Microsoft AZ-400 exam.

The topics covered in this blog are:

AZ-400 Certification Overview

The Azure DevOps engineer certification (Microsoft AZ-400) is an expert-level exam that certifies subject matter experts' ability to work with processes and technology while also incorporating people skills to bring commercial value to clients.

Are you new to Azure Cloud? Do check out our blog post on the Microsoft Azure Certification Path 2022 and choose the best certification for you.

Who Is Azure DevOps Engineer?

Professionals who work as Azure DevOps Engineers are in charge of creating systems that control code releases and development cycle management. A

well-known IT position ensures an organization's continual supply of business value.

These IT experts know how to use Microsoft's Azure DevOps SaaS and have experience with the Software Development Lifecycle and application development.

Who Is This Certification For?

- If you want to learn how Azure solutions are firmly integrated with the DevOps platform.
- If your firm uses Azure for development or deployment, you can give azure DevOps engineer certification.
- If you want to study and grasp DevOps and Agile principles and establish yourself in the market for success, this is the place to be.
- If you're hoping to move into the DevOps area and work in administration, software development, lead engineering, or project management, the Microsoft AZ 400 exam is for you.
- If you want to improve your development abilities and learn more

about version control, development, and deployments, Microsoft AZ-400 is the place.

Benefits of AZ-400 Certification

- The Azure DevOps engineer certification is critical for individuals wishing to establish their reputation and market worth as experienced DevOps practitioners, agile practitioners, and cloud computing professionals.
- Microsoft AZ 400 also assists system administrators, software developers, project managers, and technical leads who want to be DevOps leaders.
- The IT industry's DevOps sector will explode, and AZ-400 certification professionals will immediately qualify for high-paying roles as Azure DevOps engineers at most MNCs.

Also Check: ADF Interview Questions

AZ-400 Exam Details

Exam Name Microsoft Certified: Azure DevOps Engineer Expert	**Exam Duration** 210 Minutes
Exam Type Multiple Choice Examination	**Number of Questions** 40 - 60
Exam Fee $165	**Eligibility/Pre-Requisite** Azure Administrator Associate or Azure Developer Associate certification
Exam validity 2 Years	**Exam Languages** English, Japanese, Korean, and Simplified Chinese

AZ-400 Exam Skills Measured

Define and implement continuous integration	**20-25%**
Develop a security and compliance plan	**10-15%**
Manage source control	**10-15%**
Facilitate communication and collaboration	**10-15%**
Implement release management strategy and continuous delivery	**10-15%**
Develop an instrumentation strategy	**5-10%**
Develop Site Reliability Engineering (SRE) strategy	**5-10%**

How to Register for Azure AZ 400 Exam

You can register for the Azure DevOps Engineer Certification (AZ-400 Exam) by going to the Official Microsoft Page.

Pre-requisites for AZ-400 Certification

- Candidates must understand either Microsoft Azure Administration AZ-104 Certification or Microsoft Azure Developer AZ-204 Certification

and be an expert in at least one of these areas before taking this test.

- For AZ-400 certification exam candidates, proficiency in designing and deploying Azure DevOps practices for version control, configuration management, build compliance, release, and testing using Azure technologies are highly encouraged.
- Agile knowledge is a plus.
- Expertise in expediting delivery through optimizing techniques, automating processes, and deploying application code and infrastructure solutions.

AZ-400 Study Guide

Develop an instrumentation strategy (5-10%)

Design and implement logging

- Assess and configure a logging framework
  - Overview of Azure platform logs
- Design a log aggregation and storage strategy (e.g., Azure storage)
  - Design and Implement an Azure Storage Strategy
- Design a log aggregation and query strategy (e.g., Azure Monitor, Splunk, Exabeam, LogRhythm)
  - Samples for Kusto Queries
  - Integrate logs with Splunk using Azure Monitor
- Manage access control to logs (workspace-centric/resource-centric)
  - Manage access to log data and workspaces in Azure Monitor
- Integrate crash analytics (App Center Crashes, Crashlytics)
  - What is App Center?
  - How App Center Diagnostics works
  - App Center Crashes (Windows)
  - Monitor Your App's Health with App Center Crash & Analytics

Design and implement telemetry

- Design and implement distributed tracing
  - What is Distributed Tracing?
  - Distributed tracing
- Inspect application performance indicators
  - Common APM Performance Metrics
- Inspect infrastructure performance indicators
  - How to chart performance with VM insights
  - Monitor your Kubernetes cluster performance with Container insights
- Define and measure key metrics (CPU, memory, disk, network)
  - View VM metrics
- Implement alerts on key metrics (email, SMS, webhooks, Teams/Slack)
  - Create, view, and manage log alerts using Azure Monitor
  - How to trigger complex actions with Azure Monitor alerts
- Integrate user analytics (e.g., Application Insights funnels, Visual Studio App Center, TestFlight, Google Analytics)
  - Discover how customers are using your application with Application Insights Funnels
  - Get Started with WPF/WinForms
  - Testing the iOS SDK Integration with TestFlight

Integrate logging and monitoring solutions

- Configure and integrate container monitoring (Azure Monitor, Prometheus, etc.)
  - Enable Container insights
  - Configure scraping of Prometheus metrics with Container insights
- Configure and integrate with monitoring tools (Azure Monitor Application Insights, Dynatrace, New Relic, Naggios, Zabbix)
  - Set up integration with Azure Monitor
  - Activate Azure integrations
  - Integrate Nagios with Microsoft Azure Stack Hub
  - Integrate Zabbix with Azure

- Create feedback loop from platform monitoring tools (e.g., Azure Diagnostics extension, Log Analytics agent, Azure Platform Logs, Event Grid)
  - Azure Performance Diagnostics VM Extension for Windows
  - Create diagnostic settings to send platform logs to different destinations
- Manage Access control to the monitoring platform
  - Manage access to log data and workspaces in Azure Monitor

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

Develop an actionable alerting strategy

- Identify and recommend metrics on which to base alerts
  - Supported resources for metric alerts in Azure Monitor
- Implement alerts using appropriate metrics
  - Create, view, and manage metric alerts using Azure Monitor
- Implement alerts based on appropriate log messages
  - Create, view, and manage log alerts using Azure Monitor
- Implement alerts based on application health checks
  - Configure resource health alerts using Azure portal
- Analyze combinations of metrics
  - Advanced features of the Azure metrics explorer
- Develop communication mechanism to notify users of degraded systems
  - Get notified when Azure service incidents impact your resources
- Implement alerts for self-healing activities (e.g., scaling, failovers)
  - Design for self-healing
  - New features for Azure Alerts and Auto scale

Design a failure prediction strategy

- Analyze behavior of system with regards to load and failure conditions

    - Failure mode analysis for Azure applications
- Calculate when a system will fail under various conditions
    - How Azure uses machine learning to predict VM failures
- Measure baseline metrics for system
    - Establish baseline metrics
- Leverage Application Insights Smart Detection and Dynamic thresholds in Azure Monitor
    - Foretell and prevent downtime with predictive maintenance

Design and implement a health check

- Analyze system dependencies to determine which dependency should be included in health check
    - Visualize dependencies in an Azure Virtual Machine
- Calculate healthy response timeouts based on SLO for the service
    - Service Level Objectives
    - Azure DB Timeout exception
    - Timeout errors which happen sometimes on Azure SQL Databases called in Azure Web jobs
- Design approach for partial health situations
    - Requirements for health monitoring
- Integrate health check with compute environment
    - Understanding health criteria in Azure Monitor for VMs
- Implement different types of health checks (container liveness, startup, shutdown)
    - Configure liveness probes
    - Configure Liveness, Readiness and Startup Probes

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.)
    - What is Azure AD Privileged Identity Management?
    - What is Conditional Access?
    - How it works: Azure AD Multi-Factor Authentication
- Implement Service Principals and Managed Identity
    - Authentication with service principals in Azure AD
    - Use managed identities with Azure virtual machines
- Design an application access solution using Azure AD B2C
    - Using Azure B2C with An App
- Configure service connections
    - Manage service connections

Design a sensitive information management strategy

- Evaluate and configure vault solution (Azure Key Vault, Hashicorp Vault)
    - Getting started with Azure Key Vault using the Azure portal
- Manage security certificates
    - Add a TLS/SSL certificate in Azure App Service
- Design a secrets storage and retrieval strategy (KeyVault secrets, GitHub secrets, Azure Pipelines secrets)
    - Storing and using secrets in Azure
- Formulate a plan for deploying secret files as part of a release
    - Use Azure Key Vault secrets in Azure Pipelines
    - A CI/CD journey with Azure DevOps and Terraform

Develop security and compliance

- Automate dependencies scanning for security (container scanning, OWASP)
    - Dependency Scanning
    - Container Scanning
    - OWASP Dependency-Check

- Automate dependencies scanning for compliance (licenses: MIT, GPL)
  - Compliance and vulnerability scanning
- Assess and report risks
  - Manage risks in Azure DevOps
- Design a source code compliance solution (e.g., GitHub Code scanning, GitHub Secret scanning, pipeline-based scans, Git hooks, SonarQube, Dependabot, etc.)
  - Get Started with Git Hooks
  - Control source code quality using the SonarQube platform
  - Securing Azure Pipelines

Design governance enforcement mechanisms

- Implement Azure policies to enforce organizational requirements
  - Create and manage policies to enforce compliance
- Implement container scanning (e.g., static scanning, malware, crypto mining)
  - Static Security Analysis of Container Images with CoreOS Clair
  - 5 open-source tools for container security
  - Hacker's mass-scan for Docker vulnerability to mine Monero cryptocurrency
- Design and implement Azure Container Registry Tasks
  - Automate container image builds and maintenance with ACR Tasks
- Design break-the-glass strategy for responding to security incidents
  - "Break glass": what to do in an emergency

Manage source control (10-15%)

Develop a modern source control strategy

- Integrate/migrate disparate source control systems (e.g., GitHub, Azure Repos)
  - Exercise – Migrating your repository to GitHub

    - Migrating Code to Azure DevOps Repos (4 Different Scenarios)
- Design authentication strategies
    - Authenticating to GitHub
    - Other authentication methods
- Design approach for managing large binary files (e.g., Git LFS)
    - Handling Large Binary Files with LFS
- Design approach for cross repository sharing (e.g., Git sub-modules, packages)
    - Two Git repositories share common code
    - Cross-repository Component Sharing using Mono-repo Multi-packages Architecture
- Implement workflow hooks
    - How To Ease Your Team's Development Workflow with Git Hooks

Plan and implement branching strategies for the source code

- Define Pull Requests (PR) guidelines to enforce work item correlation
    - Link work items to pull request
    - Add link from a work item to a GitHub commit, pull request, or issue
- Implement branch merging restrictions (e.g., branch policies, branch protections, manual, etc.)
    - Branch policies and settings
    - Configuring protected branches and required status checks
    - Managing a branch protection rule
- Define branch strategy (e.g., trunk based, feature branch, release branch, GitHub flow)
    - DevOps tech: Trunk-based development
    - Best Branching Strategies for High-Velocity Development
    - GitHub flow
- Design and implement a PR workflow (code reviews, approvals)
    - How to code review in a Pull Request
    - Approving a pull request with required reviews

- Enforce static code analysis for code-quality consistency on PR
  - Automate Code Review with Static Code Analysis

Configure repositories

- Configure permissions in the source control repository
  - Access permissions on GitHub
  - Setting permissions for deleting or transferring repositories
- Organize the repository with git-tags
  - Create a tag in a GitHub repository
- Plan for handling oversized repositories
  - How to handle big repositories with Git
  - Working with large files and repositories
- Plan for content recovery in all repository states
  - Restoring a deleted repository
  - Repairing and recovering broken git repositories
- Purge data from source control
  - About large files on GitHub
  - Removing sensitive data from a repository

Integrate source control with tools

- Integrate GitHub with DevOps pipelines
  - GitHub integration with Azure Pipelines
- Integrate GitHub with identity management solutions (Azure AD)
  - Azure AD SSO integration with a GitHub Enterprise Cloud Organization
- Design for GitOps
  - GitOps for Azure Rendering
  - Azure DevOps pipelines repo that supports the GitOps flow
- Design for ChatOps
  - ChatOps for Azure Active Directory
  - ChatOps with Azure DevOps and Microsoft Teams via YellowAnt!

- 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)
  - About dashboards, charts, reports, & widgets
  - Creating Dashboard in Azure DevOps
- Design a cost management communication strategy
  - Define Azure Cost Management
  - Cost Management discipline template
- Integrate release pipeline with work item tracking (e.g., AZ DevOps, Jira, ServiceNow)
  - Link work items to builds and deployments in Azure Boards
  - Integrate with ServiceNow change management
  - Azure Pipelines integration with Jira Software
- Integrate GitHub as repository with Azure Boards
  - Connect Azure Boards to GitHub (Cloud)
- Communicate user analytics
  - What is the Analytics service?

Generate DevOps process documentation

- Design onboarding process for new employees
  - What are your recommendations on how to onboard DevOps engineers?
- Assess and document external dependencies (e.g., integrations, packages)
  - Overview of Azure DevOps dependency tracker

    - Creating a Dependency
- Assess and document artifacts (version, release notes)
    - Azure Artifacts overview
    - Artifact Details – Get Package Versions
    - Release artifacts and artifact sources

Automate communication with team members

- Integrate monitoring tools with communication platforms (e.g., Teams, Slack, dashboards)
    - Use Azure Pipelines with Microsoft Teams
    - Azure Pipelines with Slack
    - Add continuous monitoring to your release pipeline
- Notify stakeholders about key metrics, alerts, severity using communication and project management platforms (e.g., Email, SMS, Slack, Teams, ServiceNow, etc.)
    - Create a service hook for Azure DevOps with Slack
    - Create a service hook for Azure DevOps with Microsoft Teams
- Integrate build and release with communication platforms (e.g., build fails, release fails)
    - Get notifications only for failed builds
    - Send Email After Release Deployment in Azure DevOps

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)
    - Code Coverage Report (with JaCoCo) in Azure DevOps Pipeline
    - Building security into your Azure DevOps Pipeline
- Implement quality gates (e.g., code coverage, internationalization, peer review)

    - Release deployment control using gates
    - Code coverage for pull requests
    - Create pull requests
- Design a testing strategy (e.g., integration, load, fuzz, API, chaos)
    - Create test plans and test suites
    - Load test overview – Azure Test Plans
    - Integrate automated test in Azure DevOps using the Postman API
    - Fuzz Testing at Microsoft and the Triage Process
- Integrate multiple tools (e.g., GitHub Actions, Azure Pipeline, Jenkins)
    - Use GitHub Actions to trigger a run in Azure Pipelines
    - Configuring a CD pipeline for your Jenkins CI

Design a package management strategy

- Recommend package management tools (e.g., GitHub Packages, Azure Artifacts, Azure Automation Runbooks Gallery, Nuget, Jfrog, Artifactory)
    - Introduction to GitHub Packages
    - Azure Artifacts overview
    - Introducing the Azure Automation Runbook Gallery
    - An introduction to NuGet
    - Package Management
- Design an Azure Artifacts implementation including linked feeds
    - What are feeds?
- Design versioning strategy for code assets (e.g., SemVer, date based)
    - Build Versioning in Azure DevOps Pipelines
    - Automated date-based versioning for ASP.NET Core assemblies using Azure DevOps
- Plan for assessing and updating and reporting package dependencies (GitHub Automated Security Updates, NuKeeper, GreenKeeper)
    - Configuring Dependabot security updates
    - What Greenkeeper does, and why?
- Design a versioning strategy for packages (e.g., SemVer, date based)

    - Nuget Package Versioning in Azure DevOps with GitVersion
- Design a versioning strategy for deployment artifacts
    - Best way to handle versioning w/ yaml pipelines

Design an application infrastructure management strategy

- Assess a configuration management mechanism for application infrastructure
    - Azure Automation State Configuration overview
- Define and enforce desired state configuration for environments
    - The what, why and how of Azure Automation Desired State Configuration (DSC)
    - Configuring Azure DSC Automation with PowerShell in 5 steps

Implement a build strategy

- Design and implement build agent infrastructure (include cost, tool selection, licenses, maintainability)
    - Create a build agent that runs on Azure
- Develop and implement build trigger rules
    - Trigger one pipeline after another
- Develop build pipelines
    - Create your first pipeline
- Design build orchestration (products that are composed of multiple builds)
    - Create a multi-platform pipeline
- Integrate configuration into build process
    - CI and CD with Azure DevOps – Quickstart
- Develop complex build scenarios (e.g., containerized agents, hybrid, GPU)
    - Run a self-hosted agent in Docker
    - Self-hosted GPU agents for Azure Pipelines
    - Best Practices for creating a VM with GPU inside Azure DevOps

Pipelines
- Enabling DevOps in A Hybrid Cloud Environment at DoD

Maintain build strategy

- Monitor pipeline health (failure rate, duration, flaky tests)
  - Add continuous monitoring to your release pipeline
  - Test failures report
  - Manage flaky tests
- Optimize build (cost, time, performance, reliability)
  - Reducing Cost & Complexity for Azure Pipelines CI/CD
  - Identifying Cost-Saving Opportunities in Azure DevOps
  - Pipeline caching
  - Improving Angular CI Build Time Using Azure DevOps "Cache Task"
  - Performance tuning an Azure DevOps build configuration
- Analyze CI load to determine build agent configuration and capacity
  - AZURE DEVOPS BUILD AGENT ANALYSIS
- Manage pipeline health
  - Pipeline reports
- Identify the number of agents and jobs to run in parallel
  - Configure and pay for parallel jobs
- Investigate test failures
  - Test Failures

Design a process for standardizing builds across organization

- Manage self-hosted build agents (VM templates, containerization, etc.)
  - Self-hosted agents
  - Run a self-hosted agent in Docker
  - Use an ARM-template generated host as build agent
- Create reuseable build subsystems (YAML templates, Task Groups, Variable Groups, etc.)

- How to reuse yaml templates in different Azure DevOps team project?
- Task groups for builds and releases (classic)
- Add & use variable groups

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.)
  - CI/CD Tools Comparison
  - Jenkins vs Github Actions
  - whats the best CI/CD service
  - CircleCI vs Travis CI vs Jenkins vs Alternatives
- Design and implement Infrastructure as code (ARM, Terraform, PowerShell, CLI)
  - Azure Resource Manager templates
  - Terraform
  - Azure CLI Scripts and Tasks
- Develop application deployment process (container, binary, scripts)
  - Use deployment scripts in ARM templates
- Develop database deployment process (migrations, data movement, ETL)
  - SQL Server database migration to Azure SQL Database
  - Azure SQL hybrid data movement
- Integrate configuration management as part of the release process
  - The Link Between Change Management and Release Management
- Develop complex deployments (IoT, Azure IoT Edge, mobile, App Center, DR, multiregion, CDN, sovereign cloud, Azure Stack, etc.)
  - Deploy IoT Edge modules at scale using the Azure portal
  - Setup Azure DevOps and App Center to deploy your application

    - Multi regional deployment with Azure DevOps and Azure App Services
    - End to End Azure CDN Deployment with Self-Managed CA Signed Certificates
    - Deploy to Azure Stack Hub App Service using Azure Pipelines

Implement an orchestration automation solution

- Combine release targets depending on release deliverable (e.g., Infrastructure, code, assets, etc.)
    - Integrating Infrastructure as Code into a Continuous Delivery Pipeline
- Design the release pipeline to ensure reliable order of dependency deployments
    - Add stages, dependencies, & conditions
- Organize shared release configurations and process (YAML templates, variable groups, Azure App Configuration)
    - Grouping Shared Variables in Azure DevOps Pipeline
- Design and implement release gates and approval processes
    - Controlling Deployments using Release Gates
    - Release deployment control using approvals

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.)
    - Considerations on using Deployment Slots in your DevOps Pipeline
    - Blue-Green deployments using Azure Traffic Manager
    - Feature Toggles
- Select the appropriate desired state solution for a deployment environment (PowerShell DSC, Chef, Puppet, etc.)
    - Use infrastructure automation tools with virtual machines in Azure

- Plan for minimizing downtime during deployments (VIP Swap, Load balancer, rolling deployments, etc.)
  - Deployments with VIP swap
  - Blue-Green Deployment on Azure with Zero Downtime
  - Rolling deployment strategy
- Design a hotfix path plan for responding to high priority code fixes
  - Doing a hotfix with Azure DevOps

AZ-400 Exam Day Tips

Below are some of the Microsoft AZ-400 exam strategies and some recommendations that will help you.

1. You can take the AZ-104 and AZ-900 examinations to accustom yourself to the Azure offerings' services.
2. Schedule the azure DevOps engineer certification exam at least 60-90 days in advance. If you have coupons from a learning partner, try to use them or keep an eye out for Microsoft's open Cloud Skill Challenges.
3. If you are giving a Virtual Exam for the first time, read the exam information provided by PearsonVUE to verify that your desk and workspace are clean before beginning the exam.
4. The best time to schedule the AZ 400 certification exam is always a question. You can suit it according to you. If you're an early bird, and your mind is free of the stresses of the day, give it in the morning. If you're having trouble with wait times, you can make scheduling in the evenings or afternoons PST time zones.
5. You will have access to a whiteboard where you can jot down exam-related ideas.
6. Before the AZ 400 exam starts, you can adjust the brightness of your screen or go to Dark Mode according to your choice. Staring at a white screen can make it very challenging to focus. You can consider switching to dark mode from near the bottom left when you start.

7. Use the AZ 400 certification Exam Outline to write down your target dates for each module and part to stay on track. For instance, you can choose a deadline and work backward to determine how much time you need to devote to each module and section.

AZ 400 Exam Retake Policy

The AZ-400 Exam Retake Policy is as follows:

1. If a candidate fails on the first attempt, they must wait for 24 hours before retaking the exam.
2. If a candidate again fails on the second attempt, then the candidate will have to wait for 14 days.
3. A candidate will be given a maximum of five attempts to retake an exam in a year.

Conclusion

One of the new role-based Azure certifications, Azure DevOps Solutions, validates the competencies of Azure DevOps Professionals. AZ-400 exam can help you pave the way in your career and learn a new set of skills to hike your salary as well.

Related/References

- AZ-600: Azure Stack Hub Operator Associate Exam Study Guide
- AZ-700: Azure Network Engineer Associate Exam Study Guide
- DP-203: Microsoft Azure Data Engineer Associate Exam Study Guide
- DP-100: Microsoft Azure Data Scientist Associate Exam Study Guide
- PL-300: Microsoft Power BI Data Analyst Associate Exam Study Guide

Leave a Comment

Name *

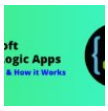
Email *

Save my name, email, and website in this browser for the next time I comment.

Post Comment

Recent Posts

Azure Storage Explorer: Download, Install, and Setup Overview

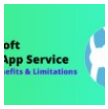


What are Azure Logic Apps: Components, Advantages and How it Works

Microsoft Azure Application Insights: A Complete Beginners

Guide

Microsoft Azure Service Bus: A Complete Beginners Guide



Azure App Service: Types, Benefits and Limitations