CCIE Automation Practical v1.2

Exam Description: The CCIE Automation Practical v1.2 is an 8-hour, hands-on exam that tests your expertise of planning, designing, developing, testing, deploying, and maintaining infrastructure automation solutions within complex network environments. Passing this exam fulfills the practical exam requirement for the CCIE Automation certification.

The following topics serve as a general guide for the exam. Please note that additional related topics may be included in any given exam version. These guidelines are subject to change at any time without notice to ensure they remain accurate and relevant.

20% 1.0 Software Design, Development, and Deployment

- 1.1 Compare, Evaluate, and Recommend a network automation solution based on business and technical factors:
  - 1.1.a Cisco solutions: Crosswork, Network as Code (netascode.cisco.com), RADKit, pyATS
  - 1.1.b Infrastructure as Code Tools: Ansible Automation Platform, HashiCorp Terraform, OpenTofu
  - 1.1.c Python Frameworks: Nornir, Netmiko, Scrapli
- 1.2 Modify an existing network automation solution based on business and technical requirements (includes gap analysis, source of truth)
- 1.3 Use Git in a CI/CD development workflow
- 1.4 Troubleshoot issues with a CI/CD pipeline (e.g., code-based failures, pipeline issues, and tool incompatibility)
- 1.5 Diagnose network automation issues - such as long running tasks, authentication/authorization problems, connection limits, control plane impact, data plane impact
- 1.6 Design a network automation solution that leverages AI to provide agentic capabilities, conversational interfaces, and/or data processing

30% 2.0 Infrastructure as Code

- 2.1 Build, manage, and operate a Python-based REST API with a web application framework (endpoints, HTTP request and response, OpenAPI specification)
- 2.2 Build, manage, and operate a Python-based CLI application to use a REST API
- 2.3 Consume and use a new API, given the documentation
  - 2.3.a REST
  - 2.3.b GraphQL
- 2.4 Create Python based Model Context Protocol (MCP) servers that include the following capabilities to enable Large Language Models (LLMs) to integrate with network automation systems
  - 2.4.a Authentication and Authorization
  - 2.4.b Elicitations
  - 2.4.c Logging/Auditing of activity
  - 2.4.d Custom Metadata between client/server
  - 2.4.e Structured data inputs/outputs

2026 Cisco Systems, Inc. This document is Cisco Public.

2.5 Create a NETCONF filter by using XPath
2.6 Configure network devices on an existing infrastructure by using NETCONF or RESTCONF, given YANG analysis tools (and driven by a source of truth)
2.7 Create and use a role by utilizing Ansible to manage infrastructure, given support documentation
- 2.7.a Loop control
- 2.7.b Conditionals
- 2.7.c Use of variables and templating
- 2.7.d Use of connection plug-ins such as network CLI, HTTPAPI, and NETCONF

2.8 Use Terraform to manage infrastructure, given support documentation
- 2.8.a Loop control
- 2.8.b Resource graphs
- 2.8.c Use of variables
- 2.8.d Resource retrieval
- 2.8.e Resource provision
- 2.8.f Management of the state of provisioned resources

2.9 Create, modify and troubleshoot Cisco Network as Code in an ACI and SD-WAN infrastructure, given support documentation
- 2.9.a Data model instances and default files
- 2.9.b Syntactic and semantic validation rules using nac-validate

25% 3.0 Network Programmability and Automation

3.1 Modify and troubleshoot Python scripts that automate against Cisco APIs using SDKs and libraries (ACI, Catalyst Center, Cisco Secure Firewall Management Center, Splunk, Catalyst SD-WAN)
3.2 Automate the configuration of a Cisco IOS XE network device (based on a provided architecture and configuration), including these components:
- 3.2.a Interfaces
- 3.2.b Static routes
- 3.2.c VLANs
- 3.2.d Access control lists
- 3.2.e BGP peering
- 3.2.f BGP and OSPF routing tables
- 3.2.g BGP and OSPF neighbors

3.3 Modify and troubleshoot an automated test by using pyATS to meet requirements
- 3.3.a Create a testbed file for connecting to Cisco IOS, IOS XE, or NX-OS devices
- 3.3.b Gather current configuration and operational state from devices using the Genie parser and models included with pyATS
- 3.3.c Develop and execute test jobs and scripts using AEtest to verify network health

3.4 Design a model-driven telemetry solution based on given business and technical requirements by using gNMI dial-in, gRPC dial-out, and NETCONF dial-in
3.5 Create YANG model-driven telemetry subscriptions for a provided visualization tool (e.g., Telegraf + Grafana)
- 3.5.a Identify model elements and cadence
- 3.5.b On-change or event drive
- 3.5.c Optimize frequency
- 3.5.d Dial-out subscription
- 3.5.e Secure telemetry streams

10% 4.0 Containers

- 4.1 Create a Docker image (including Multi-stage Dockerfile)
  - 4.1.a Run commands during image builds
  - 4.1.b Manipulate entry point and initial commands
  - 4.1.c Establish working directories
  - 4.1.d Environment variables as part of a definition to control an application
  - 4.1.e Docker ignore file
  - 4.1.f Volumes
- 4.2 Package and deploy a solution by using Docker Compose
  - 4.2.a Deploy and manage containers
  - 4.2.b Define services, networks, volumes, and links
- 4.3 Create, consume, and troubleshoot a Docker host and bridge-based networks and integrate them with external networks

15% 5.0 Security

- 5.1 Leverage OWASP secure coding practices into all solutions to meet given requirements
  - 5.1.a Input validation
  - 5.1.b Authentication and password management
  - 5.1.c Access control
  - 5.1.d Cryptographic practices
  - 5.1.e Error handling and logging
  - 5.1.f Communication security
- 5.2 Create a Certificate Signing Request (CSR) by using OpenSSL; send CSR to a provided Certificate Authority; and use the certificate to secure a web application
- 5.3 Use OAuth2+ to obtain an authentication token
- 5.4 Use a secret management system to secure an application
- 5.5 Use tokens, headers, and secrets to secure a REST API

2026 Cisco Systems, Inc. This document is Cisco Public.