

CCNA 200-301 v1.1

MODULE 5 — Automation and Programmability

Complete Hands-on Training Labs

Topology • Sequential Device Configuration • Verification Output • Troubleshooting

NETWORK GUIDE ACADEMY

Instructor / Student Lab Guide

Scope: This workbook follows a single forward sequence. Each topology is built before it is verified. Base management configuration is introduced once in Lab 6.1 and reused by later labs; it is not duplicated. Later commands are verification, controlled state changes, API/automation actions, or troubleshooting.

Module 6 lab map

Lesson	Lab focus	Hands-on outcome
6.1	Automation impact on network management	Compare repetitive manual checks with an automated fleet verification workflow.
6.2	Traditional vs controller-based networking	Trace management, policy and telemetry paths in both models.
6.3	Software-defined architecture	Identify underlay, overlay, fabric, control/data planes and northbound/southbound interfaces.
6.4	AI and machine learning in network operations	Interpret predictive and generative AI outputs from network telemetry.
6.5	REST-based APIs	Use authentication, CRUD operations, HTTP verbs, JSON encoding and status codes.
6.6	Ansible and Terraform	Run an Ansible verification workflow and interpret a Terraform desired-state plan.
6.7	JSON-encoded data	Identify JSON components, validate syntax and extract nested values.

How to use this workbook

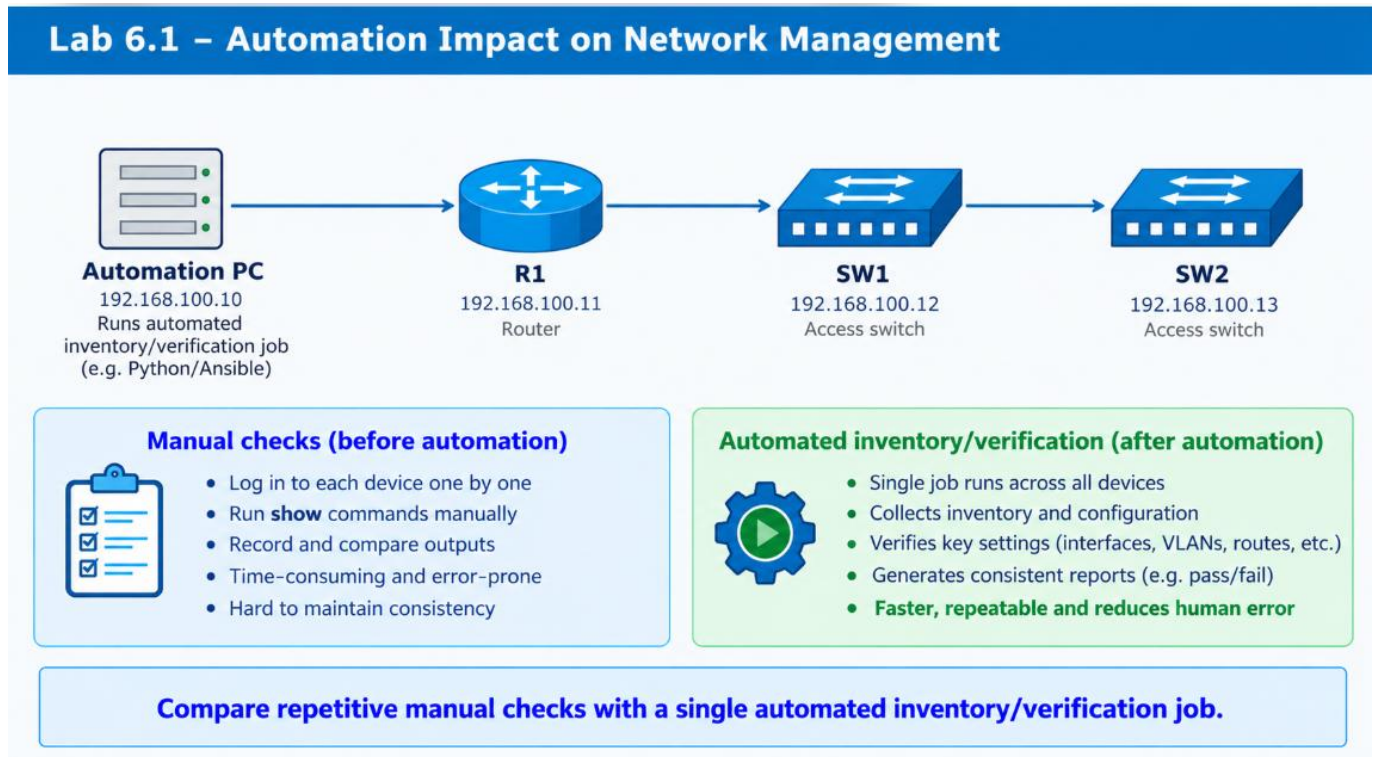
1. Build or review the topology exactly as shown for the lab.
2. Follow the steps in order. Do not reapply an earlier configuration block unless the lab explicitly tells you to change state.
3. Run each verification command immediately after the related action.
4. Compare your result with the sample output, then read the What to verify bullets.
5. For API and automation labs, use a training environment or sandbox; never test write/delete operations against an unapproved production system.

Exam alignment: Cisco CCNA 200-301 v1.1 Automation and Programmability covers automation impact, controller-based networking, software-defined architecture, AI/ML in network operations, REST APIs, Ansible/Terraform capabilities, and JSON data.

Lab 6.1 - Automation Impact on Network Management

Compare a manual device-by-device operational check with one automated verification job. The goal is to understand why automation improves consistency, speed, scale and repeatability without assuming that automation removes the need for engineering validation.

Lab topology



Addressing / device plan

Node	Role	Management address
Automation-PC	Operator / automation workstation	192.168.100.10
R1	Branch router	192.168.100.11
SW1	Access switch 1	192.168.100.12
SW2	Access switch 2	192.168.100.13

Lab objectives

- Measure the repetitive work required by a manual health check.
- Run one automation job that collects the same facts from all managed devices.
- Compare consistency, error handling and repeatability.
- Identify where human review is still required.

Topology build - full management configuration

Build this baseline once. Labs 6.2, 6.4 and 6.6 reuse it; do not reapply the same base configuration later.

Physical / logical port plan

```
SW1 Gi1/0/1 -> R1 Gi0/0
SW1 Gi1/0/2 -> SW2 Gi1/0/1 (802.1Q trunk)
SW1 Gi1/0/10 -> Automation-PC eth0
Management VLAN: 100
Management subnet: 192.168.100.0/24
Default gateway for hosts/switches: 192.168.100.11 (R1)
```

Automation-PC - full network configuration

```
Automation-PC$ sudo ip addr flush dev eth0
Automation-PC$ sudo ip addr add 192.168.100.10/24 dev eth0
Automation-PC$ sudo ip link set eth0 up
Automation-PC$ sudo ip route replace default via 192.168.100.11
Automation-PC$ ip -br addr show eth0
Automation-PC$ ip route
```

R1 - full management and SSH configuration

```
R1# configure terminal
R1(config)# hostname R1
R1(config)# no ip domain-lookup
R1(config)# ip domain-name ccna.lab
R1(config)# username lab privilege 15 secret CcnaLab123!
R1(config)# enable secret CcnaEnable123!
R1(config)# interface GigabitEthernet0/0
R1(config-if)# description MANAGEMENT-VLAN100
R1(config-if)# ip address 192.168.100.11 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)# exit
R1(config)# crypto key generate rsa modulus 2048
R1(config)# ip ssh version 2
R1(config)# line vty 0 4
R1(config-line)# login local
R1(config-line)# transport input ssh
R1(config-line)# exec-timeout 15 0
R1(config-line)# end
R1# copy running-config startup-config
```

SW1 - full management, VLAN and SSH configuration

```
SW1# configure terminal
SW1(config)# hostname SW1
SW1(config)# no ip domain-lookup
SW1(config)# vlan 100
SW1(config-vlan)# name MANAGEMENT
SW1(config-vlan)# exit
SW1(config)# interface GigabitEthernet1/0/1
SW1(config-if)# description T0-R1-Gi0/0
SW1(config-if)# switchport mode access
SW1(config-if)# switchport access vlan 100
SW1(config-if)# no shutdown
SW1(config-if)# exit
SW1(config)# interface GigabitEthernet1/0/2
SW1(config-if)# description T0-SW2-Gi1/0/1
SW1(config-if)# switchport mode trunk
SW1(config-if)# switchport trunk allowed vlan 100
SW1(config-if)# no shutdown
SW1(config-if)# exit
SW1(config)# interface GigabitEthernet1/0/10
SW1(config-if)# description AUTOMATION-PC
SW1(config-if)# switchport mode access
SW1(config-if)# switchport access vlan 100
SW1(config-if)# spanning-tree portfast
SW1(config-if)# no shutdown
SW1(config-if)# exit
SW1(config)# interface Vlan100
SW1(config-if)# description MANAGEMENT-SVI
SW1(config-if)# ip address 192.168.100.12 255.255.255.0
SW1(config-if)# no shutdown
SW1(config-if)# exit
SW1(config)# ip default-gateway 192.168.100.11
SW1(config)# ip domain-name ccna.lab
SW1(config)# username lab privilege 15 secret CcnaLab123!
SW1(config)# crypto key generate rsa modulus 2048
SW1(config)# ip ssh version 2
SW1(config)# line vty 0 4
SW1(config-line)# login local
SW1(config-line)# transport input ssh
SW1(config-line)# exec-timeout 15 0
SW1(config-line)# end
SW1# copy running-config startup-config
```

SW2 - full management, trunk and SSH configuration

```
SW2# configure terminal
SW2(config)# hostname SW2
SW2(config)# no ip domain-lookup
SW2(config)# vlan 100
SW2(config-vlan)# name MANAGEMENT
SW2(config-vlan)# exit
SW2(config)# interface GigabitEthernet1/0/1
SW2(config-if)# description T0-SW1-Gi1/0/2
SW2(config-if)# switchport mode trunk
SW2(config-if)# switchport trunk allowed vlan 100
SW2(config-if)# no shutdown
SW2(config-if)# exit
SW2(config)# interface Vlan100
SW2(config-if)# description MANAGEMENT-SVI
SW2(config-if)# ip address 192.168.100.13 255.255.255.0
SW2(config-if)# no shutdown
SW2(config-if)# exit
SW2(config)# ip default-gateway 192.168.100.11
SW2(config)# ip domain-name ccna.lab
SW2(config)# username lab privilege 15 secret CcnaLab123!
SW2(config)# crypto key generate rsa modulus 2048
SW2(config)# ip ssh version 2
SW2(config)# line vty 0 4
SW2(config-line)# login local
SW2(config-line)# transport input ssh
SW2(config-line)# exec-timeout 15 0
SW2(config-line)# end
SW2# copy running-config startup-config
```

Baseline verification

```
Automation-PC$ ping -c 3 192.168.100.11
Automation-PC$ ping -c 3 192.168.100.12
Automation-PC$ ping -c 3 192.168.100.13
Automation-PC$ ssh lab@192.168.100.11
Automation-PC$ ssh lab@192.168.100.12
Automation-PC$ ssh lab@192.168.100.13

SW1# show vlan brief
SW1# show interfaces trunk
SW1# show ip interface brief | include Vlan100
SW2# show ip interface brief | include Vlan100
```

What to verify

- Automation-PC can ping and SSH to all three managed devices.
- VLAN 100 exists on both switches; SW1-SW2 carries VLAN 100 across the trunk.
- R1 Gi0/0 and both switch management SVIs are up/up with the planned addresses.

Step 1 - Perform the manual check on R1

```
R1# show clock
R1# show ip interface brief
R1# show version | include uptime|Cisco IOS

R1# show clock
*10:15:12.442 UTC Mon Oct 21 2026

R1# show ip interface brief
Interface                IP-Address      OK? Method Status          Protocol
GigabitEthernet0/0       192.168.100.11 YES manual up              up
GigabitEthernet0/1       10.10.10.1     YES manual up              up

R1# show version | include uptime|Cisco IOS
Cisco IOS XE Software, Version 17.x
R1 uptime is 12 days, 4 hours, 18 minutes
```

What to verify

- The management interface is up/up.
- The timestamp, software version and uptime are captured in a consistent format.
- The same checks would have to be repeated separately on SW1 and SW2 if performed manually.

Step 2 - Run the prebuilt automated verification job

The training workstation contains a prepared read-only script. The script is not the learning target in this lesson; the operational result is. It connects to the same three devices and collects the same facts.

```
Automation-PC$ python3 verify_fleet.py
[OK] R1 192.168.100.11 reachable mgmt=up/up uptime=12d version=17.x
[OK] SW1 192.168.100.12 reachable mgmt=up/up uptime=31d version=17.x
[OK] SW2 192.168.100.13 reachable mgmt=up/up uptime=31d version=17.x

Summary: 3 checked, 3 successful, 0 failed
```

What to verify

- All three devices are checked with one repeatable workflow.
- Output fields are normalized, making differences easier to spot.
- A failed device would be reported explicitly instead of being silently skipped.
- The automation job performs collection; an engineer still validates anomalies and decides what action is safe.

Step 3 - Controlled failure test

Temporarily remove SW2 management reachability in the lab or point the training inventory to an unused management IP, then rerun the same job.

```
Automation-PC$ python3 verify_fleet.py
[OK]   R1  192.168.100.11  reachable
[OK]   SW1 192.168.100.12  reachable
[FAIL] SW2 192.168.100.13  timeout

Summary: 3 checked, 2 successful, 1 failed
```

What to verify

- Automation reports success and failure using the same standard for every device.
- The failure is visible in the summary and can be logged, retried or escalated.
- Automation improves operational consistency, but it does not prove that an automatically proposed remediation is correct.

Troubleshooting sequence

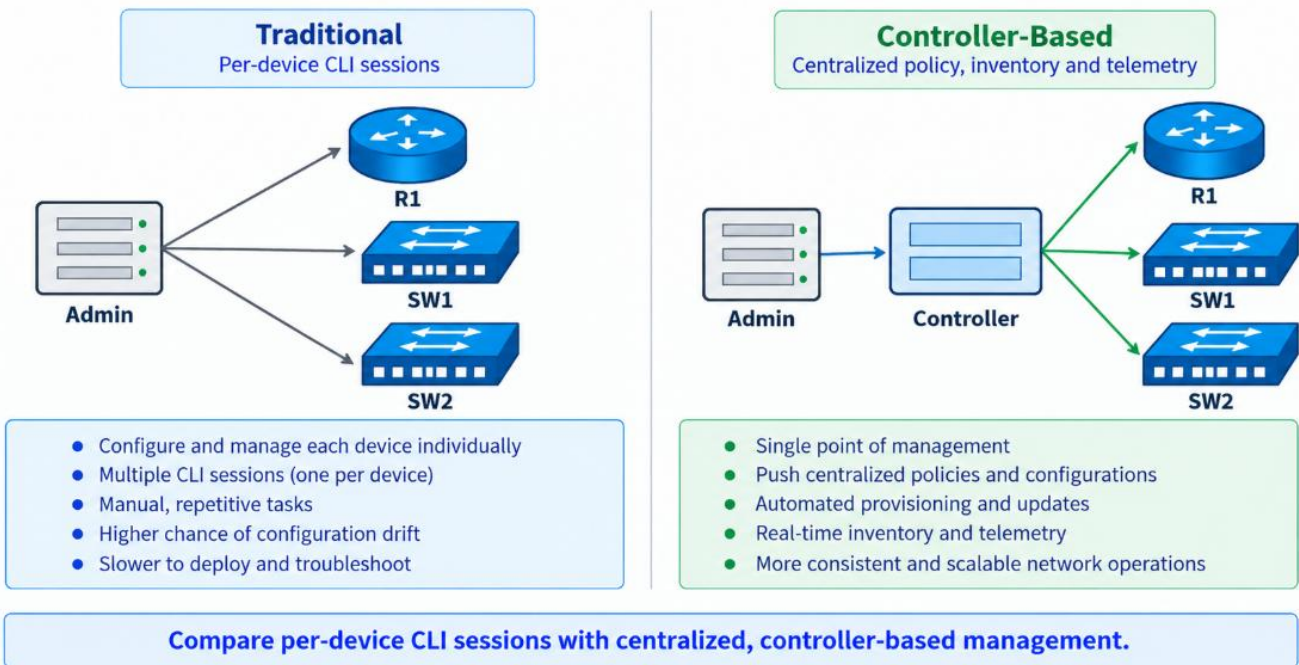
1. Confirm the automation workstation can route to the device management subnet.
2. Verify SSH/API credentials and authorization.
3. Check whether a timeout is connectivity-related or an authentication failure.
4. Test one failed target manually before changing the script or the device.

Lab 6.2 - Traditional Networks vs Controller-Based Networking

Trace how an operator manages the same network in a traditional per-device model and in a controller-based model. This is an interpretation lab; no duplicate device configuration is required.

Architecture view

Lab 6.2 - Traditional vs Controller-Based Networking



Topology build - add the controller to the Lab 6.1 baseline

Do not repeat the Lab 6.1 management configuration. Add only the controller connection and device-facing management services below.

SW1 - controller access port

```
SW1# configure terminal
SW1(config)# interface GigabitEthernet1/0/20
SW1(config-if)# description CONTROLLER
SW1(config-if)# switchport mode access
SW1(config-if)# switchport access vlan 100
SW1(config-if)# spanning-tree portfast
SW1(config-if)# no shutdown
SW1(config-if)# end
```

Controller - management addressing

```
Controller$ sudo ip addr flush dev eth0
Controller$ sudo ip addr add 192.168.100.20/24 dev eth0
Controller$ sudo ip link set eth0 up
Controller$ sudo ip route replace default via 192.168.100.11
Controller$ ping -c 3 192.168.100.11
Controller$ ping -c 3 192.168.100.12
Controller$ ping -c 3 192.168.100.13
```

R1 / SW1 / SW2 - enable controller-facing management services

```
! Apply once on each IOS/IOS XE device
(config)# username controller privilege 15 secret ControllerLab123!
(config)# ip http secure-server
(config)# restconf
(config)# netconf-yang
(config)# end
```

Controller inventory onboarding

```
Controller > Credentials > Add
Name: CCNA-LAB-CREDENTIALS
Username: controller
Password: ControllerLab123!
Protocol: SSH / HTTPS

Controller > Inventory > Add Devices
R1 192.168.100.11
SW1 192.168.100.12
SW2 192.168.100.13

Controller > Discover / Synchronize
```

What to verify

- The controller reaches all three device management addresses.
- The device credentials authenticate successfully and inventory status becomes reachable/synchronized.
- RESTCONF/NETCONF are enabled only as lab southbound examples; the exact protocol used by a real controller is platform-specific.

Step 1 - Traditional management path

```
Admin-PC> ssh lab@192.168.100.11
R1# show ip interface brief

Admin-PC> ssh lab@192.168.100.12
SW1# show vlan brief

Admin-PC> ssh lab@192.168.100.13
SW2# show vlan brief
```