Networking Architecture & Firewall Automation

Enterprise-grade networking and firewall automation for Ansible Automation Platform — built for data center and cloud engineers who need reliability at scale.

Networking Fundamentals for Ansible Automation Platform

When deploying AAP, networking is not optional — it is foundational. Automation depends on reliable communication between every platform component.

Controller

Central orchestration hub

Execution Nodes

Run playbooks & jobs

Databases

State & audit storage

External APIs

AWS, Azure, and more

If networking is unstable → automation fails. Every integration point requires reachable, stable, and properly routed connectivity.

IP Addressing Fundamentals

What Is IP Addressing?

IP addressing uniquely identifies every system on a network. AAP components require predictable, stable addresses to communicate reliably.

```
10.10.1.15    # Private Data Center
172.16.5.20   # Internal VLAN
192.168.1.10  # Small Network
```

Why IP Planning Matters for AAP

Automation requires consistent, routable connectivity. Poor planning introduces silent failures.

- Static IPs or DNS-resolvable hostnames
- Reachable subnets across all VLANs
- Proper inter-VLAN routing

Failure symptoms include intermittent job errors, routing loops, and firewall rule conflicts.

DNS: The Backbone of AAP Communication

Why DNS Is Critical

AAP components rely on hostname resolution at every layer — from controller-to-node communication to certificate validation and API integrations.

```
controller.company.local
execution1.company.local
db.company.local
```

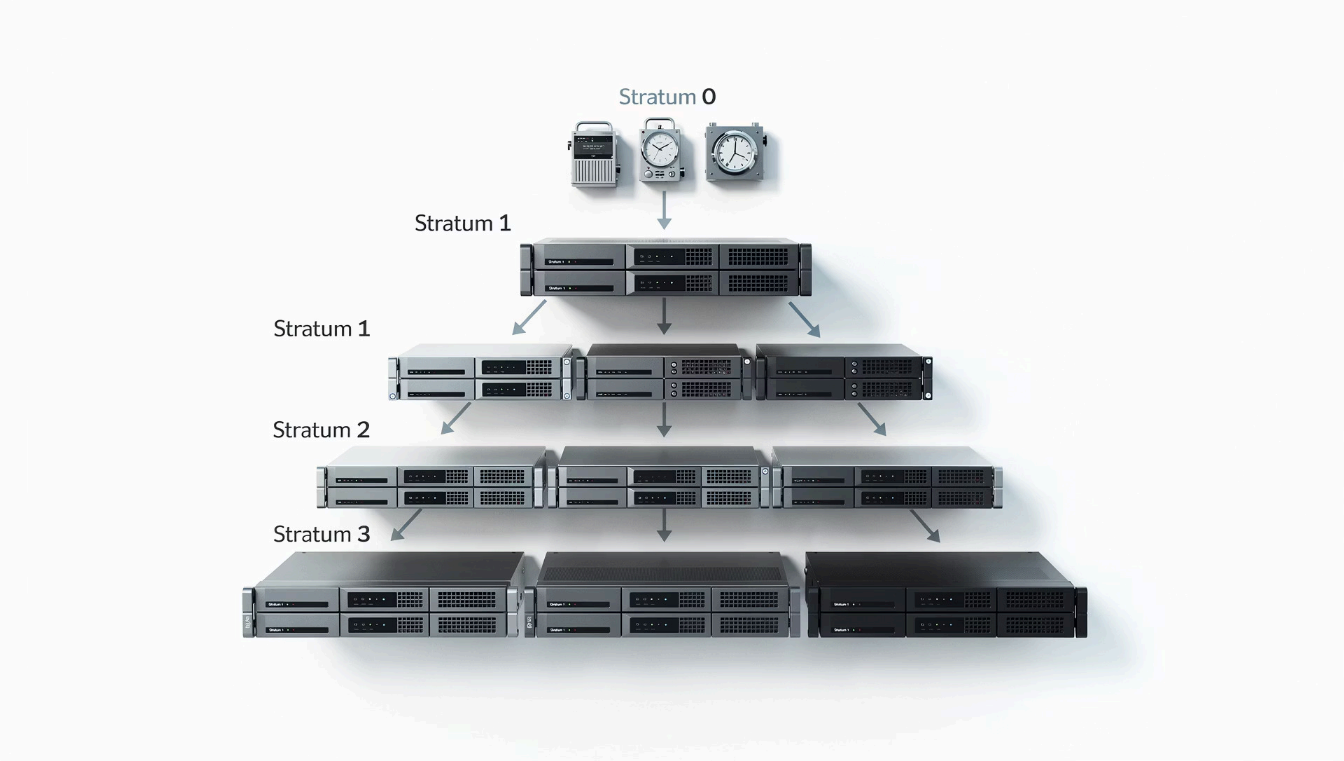
When DNS Fails

- Controller cannot connect to execution nodes
- TLS/SSL certificate validation breaks
- API integrations silently fail

NTP: Time Synchronization Across AAP

Why Time Drift Is Dangerous

Even minor clock skew can silently break critical AAP functions. Synchronized time is a non-negotiable infrastructure requirement.



Kerberos Auth
Requires clocks within 5 minutes

SSL Certificates
Validity windows are time-sensitive

Token Expiry
OAuth & API tokens depend on time

Audit Logs
Consistent timestamps for forensics

Enterprise best practice: Deploy a centralized NTP server and enforce the same time source across controller, execution nodes, database, and all managed hosts.

Network Ports and Connectivity for AAP

Every AAP deployment crosses firewall boundaries. Understanding which ports must be open — and why — is the difference between a working automation platform and hours of troubleshooting.

Common Ports Used in AAP

These are the primary ports required for Ansible Automation Platform components to communicate. Every firewall policy must account for each of these flows.

Component	Port	Purpose
Controller UI / API	443	HTTPS Web UI and REST API access
PostgreSQL Database	5432	Controller-to-database communication
SSH	22	Managed Linux node connections
WinRM (HTTP/HTTPS)	5985 / 5986	Managed Windows node connections
Automation Hub	443	Content and collection access
EDA (Event-Driven Ansible)	443	Inbound webhooks and event sources

All ports must be explicitly permitted through stateful firewall rules. Default-deny policies will block AAP communications unless these flows are configured.

Connectivity Requirements at a Glance

Controller Must Reach

→ **Execution Nodes**

Port 443 / SSH for job dispatch

→ **Database (PostgreSQL)**

Port 5432 for state management

→ **Automation Hub**

Port 443 for content sync

→ **External Cloud APIs**

AWS, Azure, GCP endpoints

Managed Nodes Must Allow

→ **SSH (Linux)**

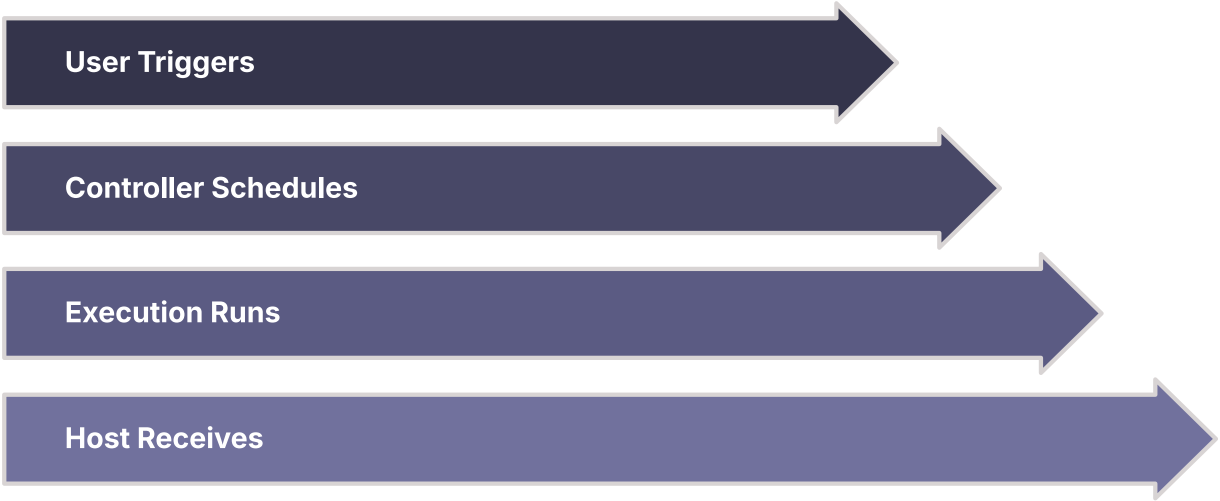
Port 22 — primary management channel

→ **WinRM (Windows)**

Ports 5985/5986 — PowerShell remoting

Control Node and Execution Node Communication

In AAP architecture, every automation job follows a strict, layered execution path — from user intent to managed infrastructure.



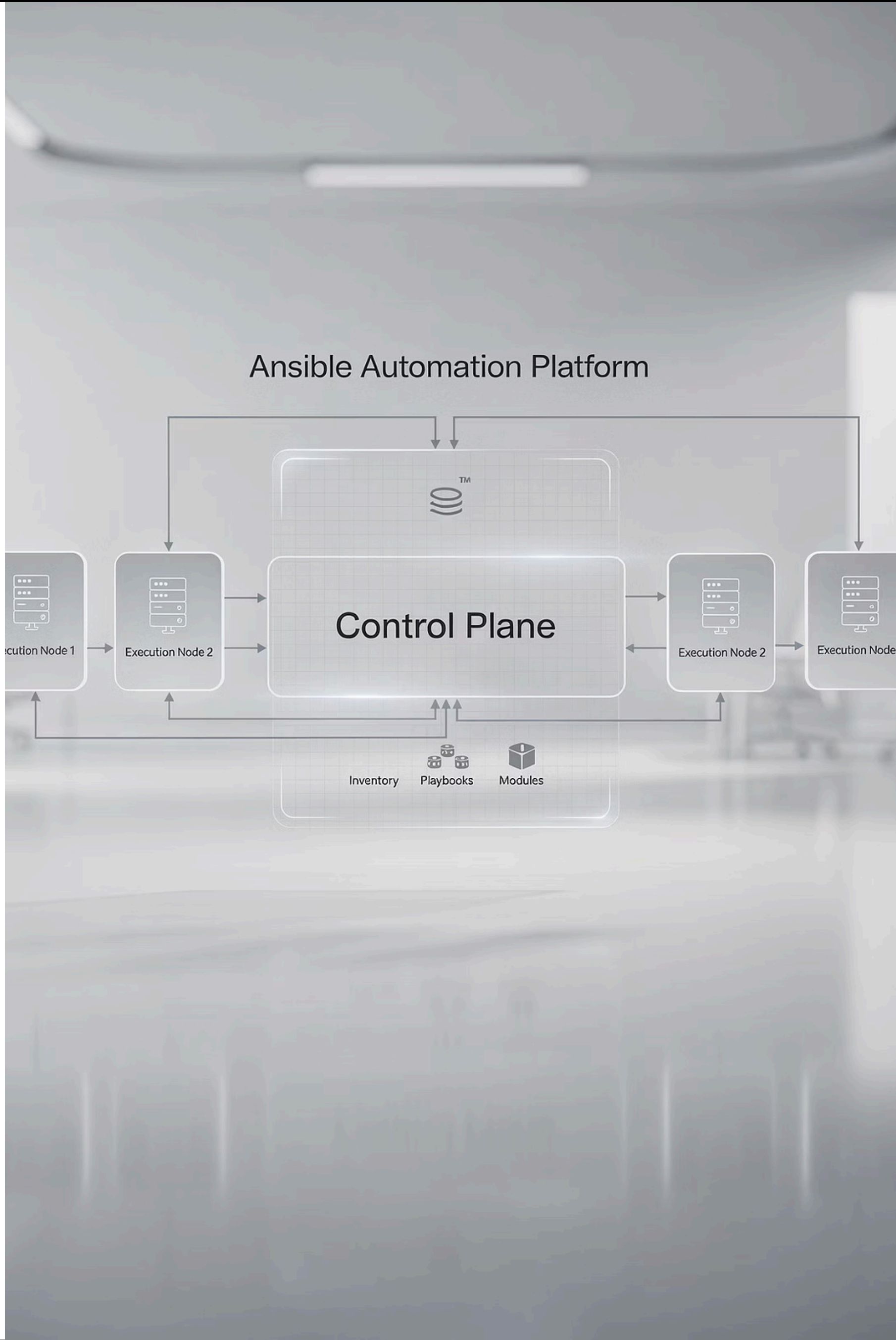
User Triggers

Controller Schedules

Execution Runs

Host Receives

This separation of control and execution planes enables scalable, distributed automation across complex multi-zone environments.



Control Plane vs. Execution Plane

Architecture Overview

Layer	Role
Control Plane	Scheduling, orchestration, API serving
Execution Plane	Runs automation tasks and playbooks

This two-plane model allows horizontal scaling of execution capacity without expanding the controller footprint.

Execution Node Responsibilities

- Pull execution environment container images from Automation Hub
- Run Ansible playbooks in isolated environments
- Communicate with managed hosts via SSH or WinRM

Inter-Plane Communication

- SSH for remote execution dispatch
- HTTPS APIs for job status and callbacks
- Secure tokens for authentication

Network Design Best Practice: VLAN Segmentation

Proper VLAN placement reduces attack surface and simplifies firewall policy management for AAP deployments.

Management VLAN

Place the **Controller** here. Restrict access to authorized admins only.

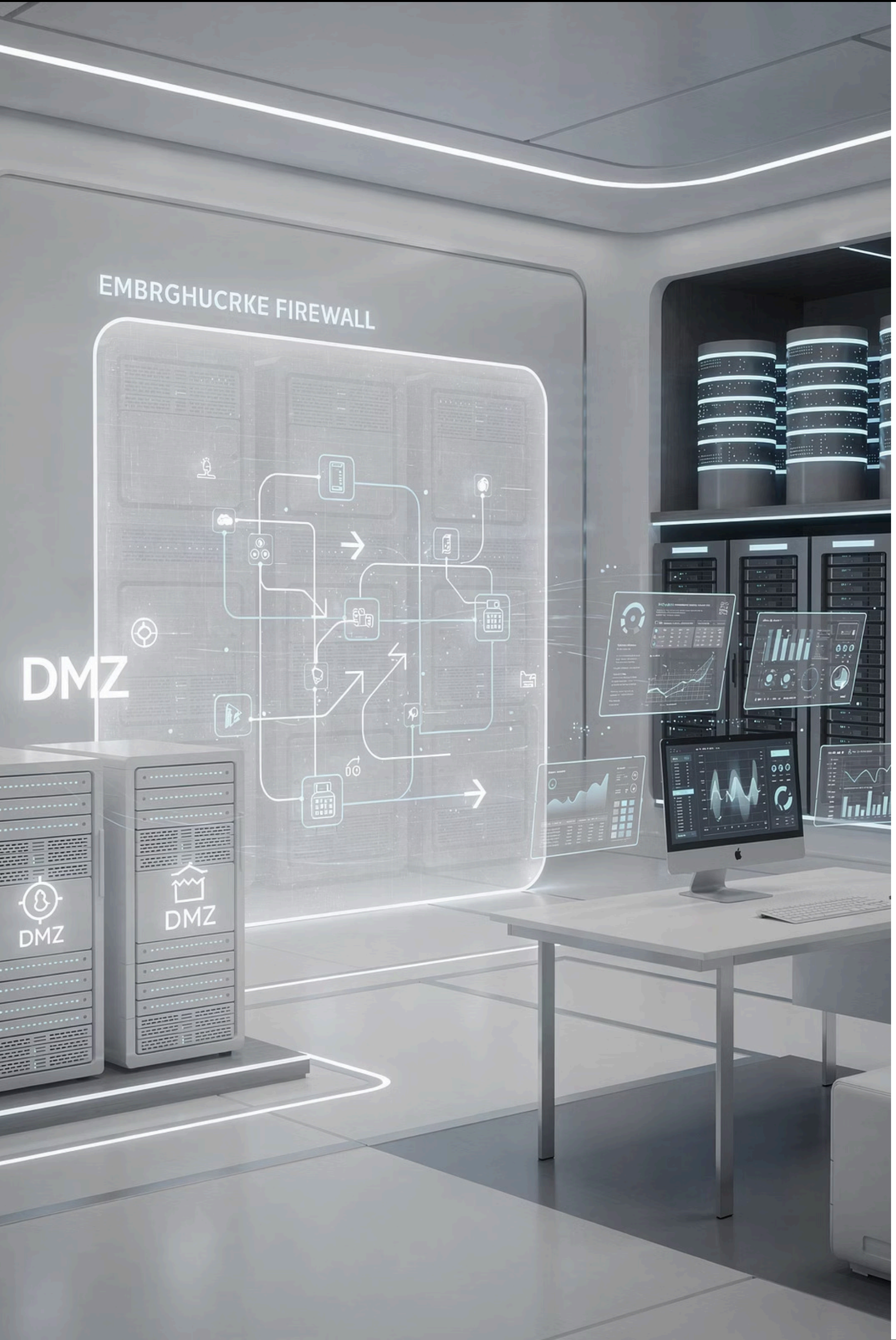
Automation VLAN

Place **Execution Nodes** here. Allow outbound to workload VLAN.

Workload VLAN

Place **Managed Hosts** here. Accept only SSH/WinRM from automation VLAN.

Allow only controlled, explicitly defined traffic between zones. Default-deny between VLANs is the correct posture.



Firewall Architecture and Security Zones

A well-designed firewall architecture is the foundation of a secure, automation-ready enterprise network. Security zones enforce segmentation, least privilege, and controlled traffic flows.

Firewall Fundamentals

What a Firewall Controls
Core Enforcement Principles

- **Least privilege** — only necessary traffic is allowed
- **Segmentation** — zones isolate blast radius
- **Threat prevention** — IPS/IDS integration at the perimeter

Inbound Traffic

External access into internal zones

Outbound Traffic

Internal systems reaching the internet

East-West

Lateral movement between internal segments

North-South

Traffic crossing the perimeter boundary

Security Zones and Traffic Flows

Enterprise environments segment infrastructure into functional zones. Each zone has a distinct trust level and a specific set of allowed traffic flows.

Zone	Purpose
DMZ	Public-facing systems — web servers, reverse proxies, API gateways
Management	Automation controllers, bastion hosts, monitoring systems
Application	Business logic and application servers (managed by Ansible)
Database	Backend storage — SQL, NoSQL, secrets management
Internet	External connectivity — cloud APIs, package mirrors, Red Hat CDN

AAP Component Placement Across Security Zones

Recommended Zone Placement

Controller

→ Management Zone

Execution Nodes

→ Automation Zone

Managed Servers

→ Application Zone

PostgreSQL DB

→ Database Zone

Required Traffic Flows

All flows below must be **explicitly permitted** in firewall policy. There are no implicit trusts between zones.

- Controller → Execution Nodes (port 443 / SSH)
- Execution Nodes → Managed Hosts (SSH port 22 / WinRM 5985/5986)
- Controller → Database (PostgreSQL port 5432)
- Controller → Automation Hub (port 443)

Document every required flow before submitting firewall change requests. Undocumented flows are the #1 cause of post-deployment failures.

Firewall Design Considerations for Automation

Automation changes the nature of network traffic. Firewall infrastructure must be designed to handle the operational demands of large-scale AAP deployments.

Frequent Rule Changes

Automation workflows introduce new endpoints and API targets continuously. Static firewall policies become bottlenecks.

Multiple Outbound API Calls

Cloud provisioning, secrets retrieval, and inventory sync all generate outbound HTTPS traffic that must be tracked and permitted.

High Concurrency & Throughput

Parallel job execution multiplies connection counts. Firewalls must handle stateful inspection at scale without becoming a bottleneck.

Logging & Change Tracking

All rule changes and traffic flows must be logged for compliance, audit trails, and incident response.

Automation-Specific Firewall Design Principles

❌ Anti-Patterns to Avoid

Hardcoded IP Rules

Break when IPs change; impossible to maintain at scale

Manual Rule Entry

Error-prone, slow, and untraceable — the opposite of automation

No Rollback Capability

A bad rule push with no undo path can take down production

✅ Best Practices to Implement

Rule Tagging

Tag rules by owner, environment, and ticket — enables auditability

Version Control

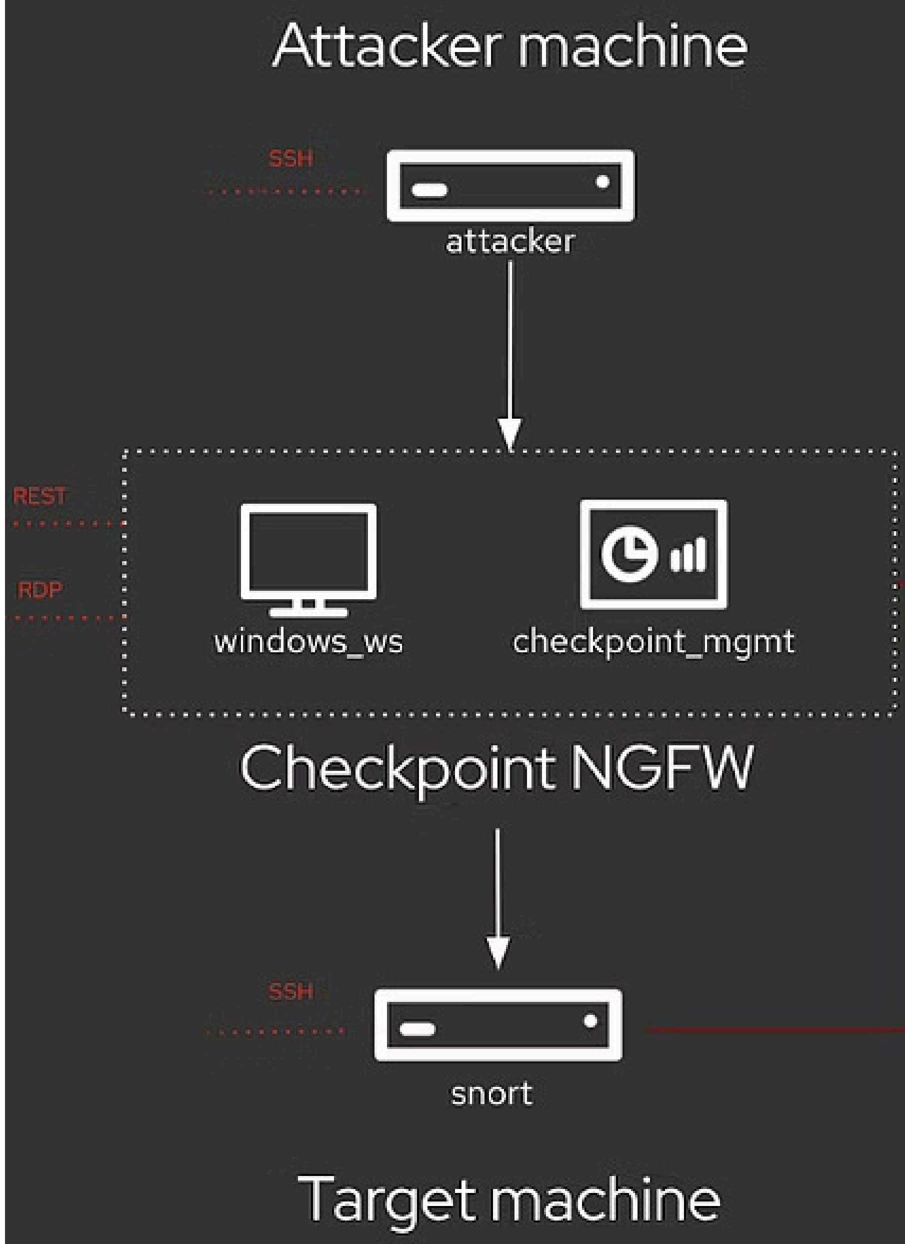
Store firewall policy in Git — enables peer review and history

Automated Validation

Use Ansible to test connectivity after every rule change before closing change tickets

Firewall Automation Using Ansible

Ansible transforms firewall management from a manual, ticket-driven process into a repeatable, auditable, version-controlled workflow — enabling consistent policy enforcement at enterprise scale.



Firewall Automation Concepts with Ansible

What Ansible Can Automate

- Firewall rule creation and deletion
- NAT rule management
- AWS Security Group updates
- Network ACL modifications
- VLAN configuration changes

Why Automate Firewall Management?

Faster Deployment

Rules applied in seconds, not days

Reduced Human Error

Playbooks enforce consistent syntax and logic

Policy Consistency

Identical rules across all environments — dev, staging, prod

Audit Trail

Every change tracked in version control with who, what, when

Supported Vendors

Cisco

Palo Alto

Fortinet

Check Point

Firewall Rule Automation: Conceptual Example

Ansible Task: Allow HTTPS Traffic

```
- name: Allow HTTPS from App Subnet
  firewall_rule:
    source: 10.10.10.0/24
    destination_port: 443
    protocol: tcp
    action: allow
    state: present
    comment: "AAP Controller API access"
```

This task can be run idempotently — it applies the rule if missing and makes no change if it already exists.

Key Takeaways

1 Networking is foundational

DNS, NTP, IP addressing, and ports must all be correct before AAP deployment begins.

2 Zones enforce security

Place every AAP component in the right security zone with explicit, documented firewall flows.

3 Automate the firewall itself

Use Ansible to manage firewall rules — faster, consistent, and auditable at enterprise scale.