Red Hat Ansible Automation Platform (AAP) Policy & Governance"

Server Engineering and Operations

Table of Contents

## # Red Hat Ansible Automation Platform (AAP) Policy & Governance

# 1. Introduction

This policy and governance document establishes the foundation for Red Hat Ansible Automation Platform (AAP) implementation at My Big Company. It defines our comprehensive governance framework, encompassing user roles, role-based access control (RBAC), and permission structures that safeguard our automation workflows within Server Engineering and Operations. This document also outlines the AAP architecture, execution model, and key concepts, clearly understanding the platform’s operational structure.

## Purpose

This document serves multiple critical functions within our automation strategy. It establishes the fundamental governance structure for AAP, including organization design, role-based access control, and permission management. These elements work together to ensure secure, scalable automation practices that align with My Big Company’s operational requirements and Server Engineering and Operations’ Strategic Roadmap (Stabilize the Environment, Drive Process Efficiencies, Develop Talent, Deliver Emerging Technologies, and Manage Cost).

Our governance framework enforces strict RBAC policies through SECU security standards to ensure only authorized personnel can create, modify, or execute automation workflows. This control extends beyond basic access management through comprehensive auditing policies that track job executions, access modifications, and automation changes. These measures ensure compliance with both industry standards and our internal security policies, with integration points Splunk, BMC Helix, and Red Hat Satellite providing a centralized view of platform activities.

The successful implementation of this governance framework requires active participation across all domains: Storage, Network, Network Security, Server, Virtualization, and App Dev. Platform Architects, Engineers, and Automation Consumers must collaborate effectively while operating within their defined roles to maintain our automation environment’s security, efficiency, and operational resilience.

## Scope

* Covers user access, job execution, inventory management, and compliance policies.
* Defines how automation is developed, tested, and deployed within AAP.
* Establishes a standardized approach to execution environments (EEs) and collection management.
* Provides policies for credential security and centralized inventory governance.
* Includes logging, auditing, and security and regulatory adherence compliance measures.
* Outlines procedures for onboarding new teams and roles within AAP.
* **Excludes** custom playbook development (handled by individual teams).

## Audience

* **Platform Engineers & Administrators**: Maintain platform governance.
* **Engineers & Operators**: Execute and develop automation.
* **Security & Compliance Teams**: Audit platform usage.
* **Executive Leadership**: Oversee governance adherence, provide strategic direction, and ensure that automation aligns with broader business objectives.

# AAP Architecture & Execution Model

The job execution process follows a defined sequence that ensures security and proper resource utilization. When a job is initiated—whether through manual intervention, scheduled execution, or a webhook trigger—the Automation Controller receives and validates the request through established [‘SECU security standards’] protocols. After validation, the Controller retrieves the necessary inventory from Red Hat Satellite, Ansible Inventory and credentials from CyberArk before dispatching the job to an appropriate Execution Node.

The Execution Node then pulls the required Execution Environment (EE) from quay.io/internal-registry, creating an isolated context for the automation task. Within this environment, the job executes using predetermined playbooks and collections ansible.windows, and community.general. Throughout the process, the platform maintains comprehensive logging of all activities in Splunk, BMC Helix, and Red Hat Satellite, ensuring full auditability of the execution chain.

This section breaks down the key components of AAP. It highlights Control Nodes, which are responsible for managing automation tasks, and Execution Nodes, which carry out those tasks. We also clarify important terminology, including team roles, credentials, workflows, and execution environments.

## Definition of Key Concepts

|  |  |
| --- | --- |
| Concept | Definition |
| **Team Roles** | Defined roles such as Platform Administrators, Engineers, Operators, and Auditors, ensuring proper governance and execution of automation. |
| **Credentials** | Securely stored secrets used for authentication and authorization within automation workflows, managed via CyberArk. |
| **Organizations** | Logical separation of automation workloads (e.g., Prod, Develop) to ensure governance and controlled access. |
| **Workflows & Jobs** | A workflow is a sequence of automated tasks, while jobs are individual executions within those workflows. |
| **Job Templates** | Predefined automation jobs ensuring standardization and repeatability. |
| **Execution Environments (EEs)** | Isolated containerized environments that host automation dependencies and ensure consistent execution. |
| **Control Plane** | The management layer is responsible for UI, API interactions, job scheduling, and RBAC enforcement. |
| **Execution Plane** | The layer where automation jobs execute in isolated environments, ensuring security and workload distribution. |
| **Automation Hub** | Repository for certified Ansible collections, ensuring standardized automation package management. |
| **Automation Controller** | Central orchestration component managing job execution across multiple execution nodes. |
| **Mesh Nodes** | Distributed execution nodes used to scale automation workloads across different network segments. |
| **Network Management Zone** | A designated area where EE virtual machine hosts are deployed, guaranteeing controlled execution within network parameters. |
| **Containers & Container Images** | Lightweight, isolated execution units used to package and run automation tasks efficiently. |
| **Registries** | Storage locations for Execution Environments and container images (e.g., quay.io/internal-registry). |

# 3. AAP Organizations & Role-Based Access Model

Our AAP implementation uses a deliberate two-organization structure that balances security with operational efficiency. The develop organization is the primary automation environment, encompassing all operational stages. This consolidated approach ensures consistent governance across our deployment pipeline while maintaining appropriate access controls for each environment.

The main organization provides a dedicated space for testing and validation, explicitly separated from production workloads. This separation enables teams to safely experiment with and validate automation workflows before promoting them through Github Actions with our Build, Test, Deploy pipeline.

This model delivers several key benefits:

1. It maintains operational simplicity while supporting enterprise-scale automation across outrastructure.
2. The structure aligns naturally with our existing Github code promotion strategy.
3. Cross-team visibility is preserved through universal read access to both organizations, fostering collaboration while maintaining security boundaries through Okta, Active Directory, and SailPoint.

## Organization Structure

* **develop**: Hosts approved automation for production environments. For governance purposes, all My Big CompanySDLC environments, including Development, Test, Staging, and Production, are considered part of the main organization within the context of AAP.
* **main**: Used for testing and validating playbooks before production deployment, explicitly tied to a sandbox environment.

## Why This Model?

* **Simplicity & Scalability**: Avoids redundant team-based orgs.
* **Code & Automation Parity**: Matches Dev → Prod branching strategy.
* **Transparency**: All roles have read access to both organizations.

## Team Role Definitions

Role permissions are handled at the team level. Each team role is mapped to a SailPoint-managed Active Directory (AD) group. Teams are domain focused and aligned with specific technology stacks (My Big Company), ensuring clear ownership of automation assets. The following roles are defined within our AAP implementation: - **Authentication to AAP is exclusively through Okta**; all users authenticate via Okta before accessing the platform. - **Local platform administrator accounts have strictly limited usage**, reserved for break-glass scenarios only. - **Platform personnel do not manage access directly**. User and team access is provisioned through **SailPoint-managed AD groups** and is governed by corporate identity policies.

## Permissions Model Overview

Our permissions framework implements multiple layers of access control and governance. Platform Administrators maintain centralized control over inventories through Red Hat Satellite, Ansible Inventory, ensuring efficient resource utilization while preventing licensed host overuse. Each team receives a dedicated inventory scope aligned with their infrastructure responsibilities, with access rights determined by team-based roles in SailPoint.

Projects are organized around platform domains including Storage, Network, Network Security, Server, Virtualization, and App Dev. This structure follows a clear ownership model: teams managing specific technologies own the corresponding automation code, while integration code resides within the relevant platform scope.

To maintain security and compliance, all platform actions are logged and forwarded to Splunk, BMC Helix, and Red Hat Satellite for analysis. Our Architects and Principal Engineers enforce clear boundaries between high-risk and low-risk workflows, with future plans to implement automated approval processes for high-risk operations. While routine workflows may operate under standing CAB approval with Work Orders, higher-risk activities require explicit Change Requests (CRQs) in BMC Helix before execution.

# 4. Platform Configuration

The platform configuration establishes essential operational parameters that ensure reliability, security, and performance across our automation infrastructure. Our implementation leverages F5 Big-IP for high availability, with comprehensive integration across our technology stack.

## Authentication & Access Control

Authentication and access control operate through a multi-layered approach utilizing Okta, Active Directory, and SailPoint. This integration provides robust user lifecycle management while maintaining compliance with our security requirements. The platform strictly adheres to SECU security standards standards.

## Execution Environment Configuration

Our Execution Environment (EE) strategy emphasizes consistency and security across all automation workflows. EEs are securely hosted in quay.io/internal-registry, with standardized Python dependencies including pywinrm, requests, and pyvmomi to support our diverse automation requirements across RHEL and Windows Server environments.

The EE lifecycle follows a structured governance model:

Development Phase EEs begin in the main organization, where teams can safely develop and validate new configurations. During this phase, Platform Engineers utilize Ansible Builder to create and modify EEs, ensuring alignment with organization-wide automation standards.

Validation & Approval Architectural review is mandatory for all EE modifications. This process ensures that changes align with our technical standards and security requirements. Platform architects must approve any significant configuration changes before proceeding with production consideration. ***Significant changes include***: - Introduction of new dependencies - Changes to existing dependencies - Updates to Python versions - Changes to the base EE image - Changes to the EE configuration - Any other modifications that may impact security or stability

Production Promotion Promotion to develop requires formal change management through BMC Helix. Only approved, stable EEs are permitted in the production environment, with container image management leveraging [‘Red Hat OpenShift’, ‘Docker’] capabilities.

Future state improvements will leverage Red Hat OpenShift Quay for enhanced EE image management, introducing automated validation workflows to streamline our governance processes while maintaining security standards.

## Collections Management

Our collection management strategy ensures consistent and secure automation content across the platform. Key aspects include:

The architectural review process evaluates new collections for security implications, dependency management, and alignment with our automation standards. Version control policies ensure stability while allowing for necessary updates, with collection updates automated through quay.io/internal-registry.

## Inventory Management

Inventory control within AAP follows a centralized management model led by Platform Engineers. This approach ensures:

1. Consistent access control through RBAC-driven policies
2. Efficient resource utilization within Red Hat Satellite, Ansible Inventory
3. Prevention of licensed host overutilization through active monitoring
4. Standardized inventory structure across automation workflows

The inventory framework supports team autonomy while maintaining platform-wide governance standards, enabling efficient automation execution while preserving security boundaries.

# 5. Team Onboarding & Role Integration

## Access Management Framework

Our platform’s access management strategy integrates with Okta, Active Directory, and SailPoint to ensure comprehensive identity governance. This framework supports our diverse technical domains: Storage, Network, Network Security, Server, Virtualization, and App Dev, each with specific operational requirements and access needs.

All permissions are managed through team roles rather than individual assignments, with group membership controlled through SailPoint. Platform Engineers maintain resource assignments for specific automation components while overall identity management remains cetralized.

Platform Administrator responsibilities include: - Managing automation resources across VMware vSphere, and Red Hat OpenShift - Overseeing integration with BMC Helix, and Digital Workplace - Ensuring compliance with SECU security standards

Changes to access permissions require formal requests through BMC Helix, and Digital Workplace, maintaining our governance standards while supporting operational efficiency.

## Permissions Model

Each team role is assigned specific permissions based on their operational responsibilities.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Role | Projects | Job Templates | Inventories | Credentials | Execution Environments |
| Platform Admin | Admin | Admin | Admin | Admin | Admin |
| Platform Architect | Admin (main) / Read (develop) | Admin (main) / Read (develop) | None | Use (main) / Read (develop) | Use (main) / Read (develop) |
| Platform Engineer | Use | Edit & Execute | Read | Use | Use |
| Platform Operator | Read | Execute | None | Use | Use |
| Platform Auditor | Read | Read | Read | Read | Read |
| Virtualization Engineer | Use | Edit & Execute | None | Use | Use |
| Virtualization Operator | Read | Execute | None | Use | Use |
| Architect | Edit & Execute (main) / Read (develop) | Edit & Execute (main) / Read (develop) | None | Use (main) / Read (develop) | Use (main) / Read (develop) |
| Engineer | Use | Edit & Execute | None | Use | Use |
| Operator | Read | Execute | None | Use | Use |
| Operator | Read | Execute | None | Use | Use |
| Auditors (All) | Read | Read | Read | Read | Read |

* **Platform Administrators assign permissions for job templates and inventories** upon creation and onboarding as part of an access change request.
* **All permission changes require an access change request** to be submitted in Digital Workplace.
* **Shared items across multiple teams are not allowed** unless explicitly defined as necessary under governance policies.
* **Each team must conduct a code review before promoting code from Develop to Prod.**

# 6. Security & Compliance

Our security framework implements comprehensive controls that protect automation assets while enabling efficient operations. The platform maintains compliance with SECU security standards standards, with regular audits conducted Quarterly.

## Hardening Guidelines

Our security posture encompasses multiple layers of protection: - Execution isolation through Red Hat OpenShift, and Docker - Network security managed via Cisco Fabric, and Infoblox (DNS) - Endpoint protection through Tenable, and SentinelOne - Access control via Okta, Active Directory, and SailPoint

## Sensitive Data Management

Credential and secret management leverages CyberArk for secure storage and controlled access. This integration ensures that sensitive data remains protected throughout the automation lifecycle while remaining accessible to authorized workflows.

## Monitoring and Compliance

Our comprehensive monitoring strategy includes: - Operational metrics tracking: Automation Coverage (%), Change Lead Time, Incident MTTR, and Configuration Drift (%) - Security event monitoring through Splunk, BMC Helix, and Red Hat Satellite - Change tracking via BMC Helix - Configuration management in Ansible Automation Platform, and GitLab

Regular compliance reporting aligns with Quarterly audit cycles, ensuring continued adherence to our security standards and operational requirements.