

## Case Study

# Infrastructure Technology Management

Ansible & Github

### The Client

An Italian multinational banking group headquartered in Milan. Listed as one of the most important 30 financial institutions worldwide.

### The Challenge

The client was facing inefficiencies in the validation and deployment of Ansible playbooks and roles across multiple teams and repositories. Manual linting and testing were time-consuming, prone to human error, and lacked standardization, leading to inconsistent code quality and delayed delivery cycles. Additionally, the absence of a robust automation pipeline made it difficult to maintain compliance and reliability across environments.

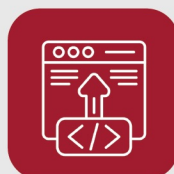
### Business Outcome

**Increased Efficiency & Reduced Manual Effort:** The client achieved significant time savings by eliminating manual linting and testing processes, freeing up engineering resources for higher-value activities

**Enhanced Code Quality & Compliance:** Automated linting and Molecule tests ensured that all Ansible code adhered to industry best practices and internal standards, reducing the risk of configuration drift and production issues.

**Improved Developer Experience:** Teams benefited from immediate feedback on code issues within the development lifecycle, leading to quicker resolution of defects and smoother code reviews.

**Standardized Workflow Across Teams:** Established a uniform, scalable, and repeatable testing framework that could be easily adopted across all teams and projects within the organization.



OVER **120%**  
Deployment rate

## Technical Solution

### Automating Playbook and Role Testing using Ansible-lint

#### Solution

Developed a fully automated solution leveraging Ansible-lint integrated with GitHub APIs

#### Implementation

Created a modular playbook that dynamically connects to GitHub, retrieves all relevant YAML/Ansible files from specified repositories and organizational units

#### Logic

Implemented logic to parse directory structures and selectively lint only Ansible-specific YAML files, skipping irrelevant configurations

#### Integration

Integrated linting into a scheduled automation process that runs periodically or on specific triggers (e.g., code commit).

#### Validation

The linting results were captured and stored at repository level for visibility by developers and release managers for desired actions

#### Result

This automation not only identified syntax issues but also enforced best practices and policy compliance across all Ansible codebases

### CI/CD Pipeline with Molecule Testing

#### Solution

Architected and implemented a CI/CD pipeline using GitHub Actions to execute Molecule testing frameworks.

#### Logic

The pipeline was designed to automatically trigger upon code commits, pull requests, or manual invocations

#### Integration

Integrated Molecule with drivers such as Docker to enable fast and isolated testing environments for Ansible roles and playbooks

#### Customized pipeline

Each pipeline stage executed distinct test phases:

- **Lint Phase:** Ansible-lint check as an early gate.
- **Dependency Phase:** Automatic dependency resolution and installation via Ansible Galaxy or internal artifact repositories
- **Converge Phase:** Applied the role/playbook to a test container
- **Verify Phase:** Custom verification tasks using Testinfra to assert the final state of the infrastructure
- **Cleanup Phase:** Removal of test containers and resources

#### Validation

Pipeline also included artifact retention (logs, reports) and notifications (Email) upon completion, ensuring feedback loops were fast and actionable.