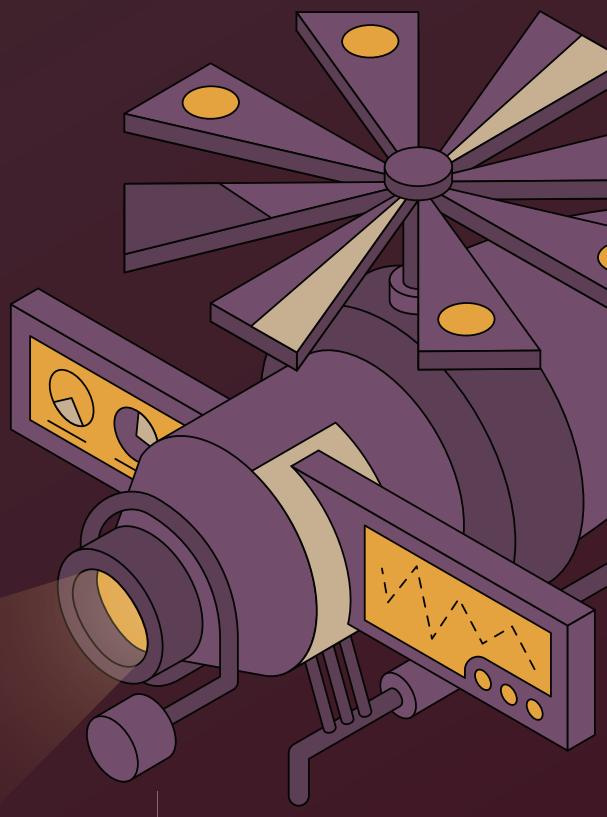
XLAB Steampunk

Quarterly AI Model Leaderboard

Which AI model should your team use for Ansible



Evaluating AI-generated Ansible playbooks across three scenarios

Q3 2026

Contents

Executive summary 3

The AI Model Leaderboard Q3 2026 findings 5

Key findings

Not all errors are equal

claude-sonnet-5's recurring blind spot

Why this matters 7

Appendix 8

Methodology

Results

About Steampunk Spotter 13

Executive summary

We tested three AI models on how well they write Ansible playbooks: DeepSeek-V4-Flash, claude-sonnet-5, and gpt-5.6-terra. Each one got the same three scenarios, starting with a simple website setup and ending with a full Ansible version upgrade. Every generated playbook was then scanned by Steampunk Spotter, a shift-left governance and AI guardrail platform built to catch errors, warnings, and risky patterns in AI-generated Ansible and Terraform code.

3
AI models

3
scenarios

Spotter found **140 errors, 94 warnings, and 221 hints across the three models and three scenarios**. One model especially struggled. Claude-sonnet-5's biggest problem, by far, was using short module names instead of fully qualified ones. This showed up 12 times in the first scenario, 17 times in the second scenario, and 27 times in the third scenario. It is the most consistent and most unexpected finding: the model that's supposed to be really capable made this mistake in every single scenario.

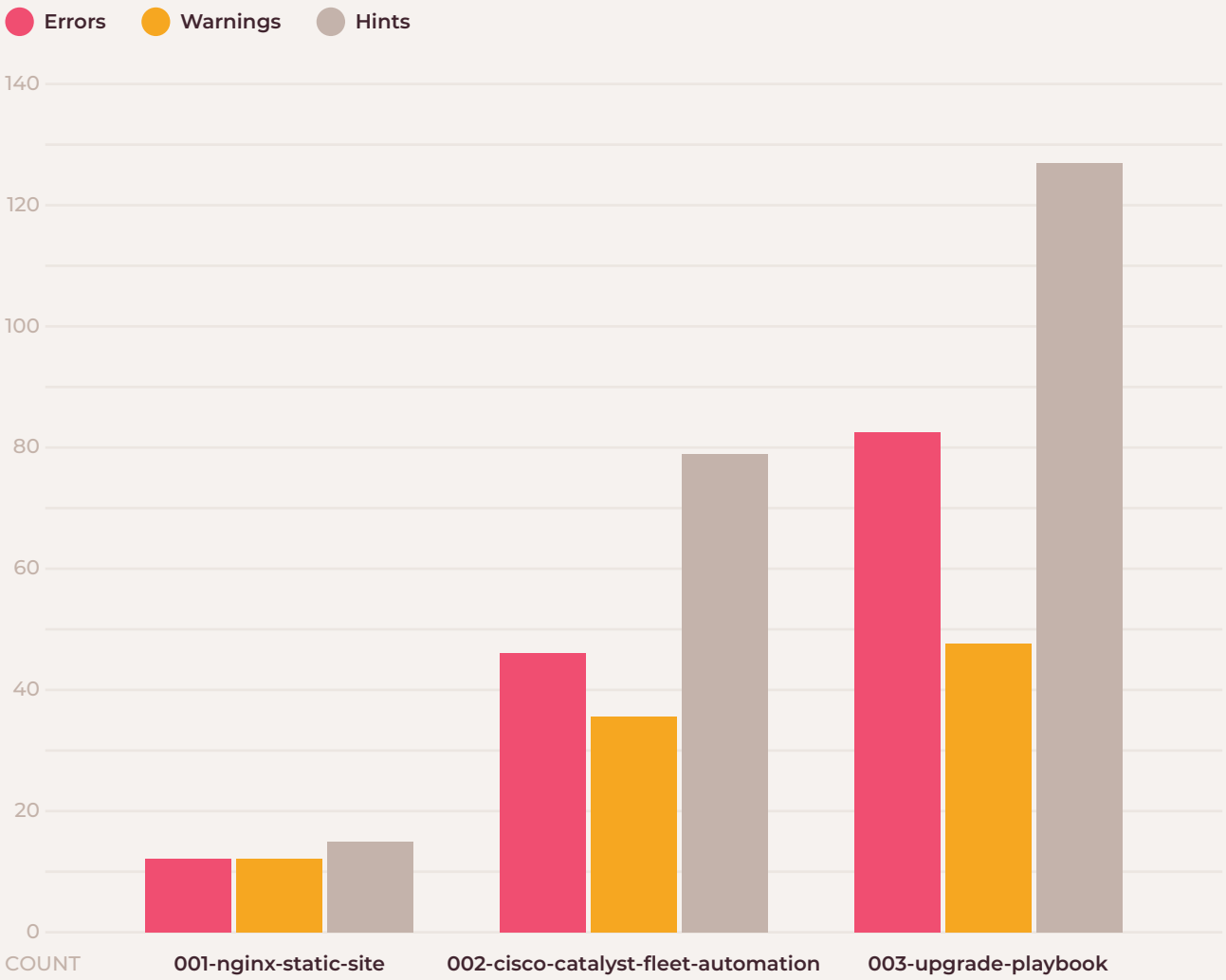
140
errors

94
warnings

221
hints

For this report, we did not test AI-generated code against organizational standards, compliance requirements, or security guidelines. We also didn't perform any reruns. Both omissions likely produced better results than an organization can expect in practice when writing and updating code with AI. We'll address these limitations in our next quarterly report.

Spotter findings by scenario (all models combined)



One issue had nothing to do with which model we picked. All three models triggered the same warning about a changed parameter default in the Ansible 2.4 to 2.14 upgrade scenario.

This is a general AI model limitation. None of these models are built specifically for Ansible, so none of them can be expected to track every collection, module, and version-specific interface that exists. Argument specs change over time too, and a model trained before a given change simply won't know about it.

The AI Model Leaderboard Q3 2026 findings

The table below shows Spotter's findings for each model and scenario. Scenarios increase in complexity, from 001, the easiest, to 003, the hardest. Scenario 001's prompt was short and direct, scenario 002's was highly detailed, and scenario 003's author had less Ansible experience but still wrote an explicit task description.

SCENARIO	MODEL	ERRORS	WARNINGS	HINTS
001 Nginx static site	DeepSeek-V4-Flash	0	2	2
	claude-sonnet-5	12	7	8
	gpt-5.6-terra	0	3	5
002 Cisco Catalyst fleet automation	DeepSeek-V4-Flash	8	10	26
	claude-sonnet-5	38	17	26
	gpt-5.6-terra	0	7	27
003 Ansible 2.4 to 2.14 upgrade	DeepSeek-V4-Flash	45	19	53
	claude-sonnet-5	34	18	34
	gpt-5.6-terra	3	11	40

Key findings

140 errors and more than 200 warnings and hints across all three models

Across all three models and scenarios, Spotter caught a significant number of issues that AI generated code missed.

FQCN non-compliance is claude-sonnet-5's biggest weakness

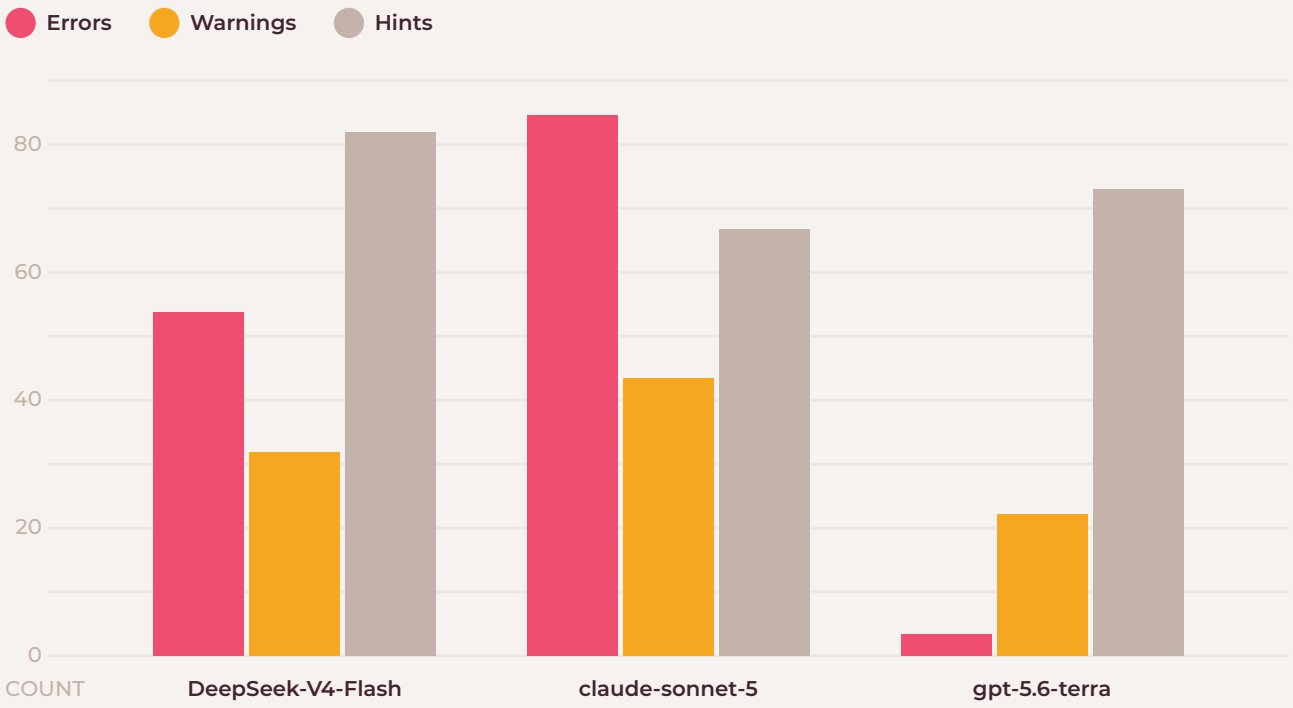
It showed up in every scenario: 12 times in 001, 17 times in 002, and 27 times in 003, the largest single error category for that model.

6 of 9 runs returned at least one error

Only gpt-5.6-terra came back error-free in more than one scenario. However, it still produced more than 90 warnings and hints across all three scenarios.

The full list of findings, by code and scenario, is in the appendix.

Spotter findings by model (all scenarios combined)

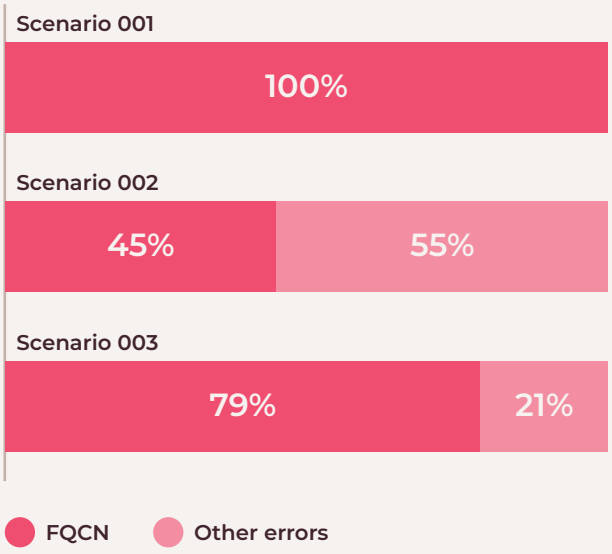


Not all errors are equal

The FQCN issue (using a short module name instead of a fully qualified one) shows up often. A short prompt addition can reduce how often it happens, but model output is probabilistic, so mistakes will still happen at random. Spotter catches and can fix this kind of error deterministically, every time, regardless of how the prompt is worded.

This gap showed up clearly in scenario 003, where the prompt explicitly named Ansible 2.14 as the target version, a version where FQCN is the expected standard. Even with that instruction stated outright, the models still made the mistake.

Errors such as "not a valid parameter for the module", "a sub-parameter placed in the wrong spot", "a parameter value that does not match any of the allowed choices", "Spotter cannot find information on the module", and "the collection version in requirements.yml does not exist" point to deeper problems and deserve more attention.



Warnings also shift in importance: the W1800 warning (the default value for a parameter changed between different versions of an Ansible module) matters more in scenario 003, since that scenario is specifically about a version upgrade.

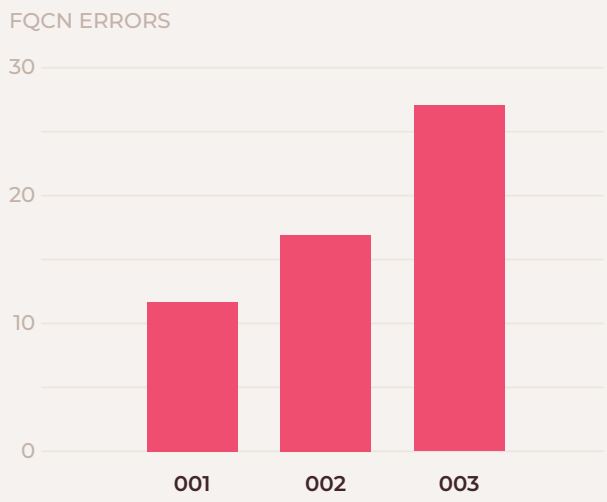
claude-sonnet-5's recurring blind spot

Across all three scenarios, claude-sonnet-5's errors are dominated by “missing fully qualified collection names”. That one issue is behind all 12 errors in scenario 001, 17 of the 38 in scenario 002, and 27 of the 34 in scenario 003. It shows up as the model's biggest error category.

This issue can improve with a short prompt instruction, but it will not go away completely. Model output is probabilistic, so mistakes can still slip through at random, even when the instruction is explicit.

Scenario 003 is proof of this: the prompt directly named Ansible 2.14 as the target version, where FQCN is the expected standard, and claude-sonnet-5 still made the mistake.

claude-sonnet-5: FQCN errors by scenario



Why this matters

Deployment readiness

All model-and-scenario combinations came back with errors, warnings and hints. That's a strong argument for putting a deterministic review step between AI-generated Ansible code and anything that touches production.

Upgrades carry risk no matter the model

All three models triggered the same warning on the Ansible 2.4 to 2.14 upgrade scenario. None of these models are built specifically for Ansible, so none of them can be expected to track every collection, module, and version-specific interface that exists. Argument specs change over time too, and a model trained before a given change simply won't know about it. Teams running AI-assisted version upgrades should expect this class of warning no matter which model they use. Another limitation is that popular AI models can't consistently enforce an organization's requirements, compliance, and security standards.

Reputation isn't a guardrail

Being a more capable model didn't help here. claude-sonnet-5 had the most errors of the three in two out of three scenarios, and its FQCN mistakes persisted even in scenario 003, where the prompt explicitly named Ansible 2.14 as the target version, a version where FQCN is the expected standard. If a model can still get this wrong with the answer spelled out in the prompt, reputation isn’t something you can count on to catch it.

Appendix

Methodology

Step 1

Three scenarios were used, increasing in complexity, each written by a different person with a different level of Ansible experience.

Step 2

Three models, DeepSeek-V4-Flash, claude-sonnet-5, and gpt-5.6-terra, each generated a playbook for every scenario in a single pass.

Step 3

Steampunk Spotter then scanned each generated playbook and reported errors, warnings, and hints. Where a run failed to complete or did not parse correctly, it was rerun until a complete result was captured for comparison.

Scenario 001: Nginx static site

This was the simplest task. The prompt was short and stated exactly what was needed with little extra detail.

> "My task is to expose index.html inside directory /var/www/html on port 8000. Would like to use nginx server. Please also ensue that page is available after command are executed. If it helps - inside ansible use/create template file for nginx file in which port must be variable. I can fix other details."

Scenario 002: Cisco Catalyst fleet automation

This task was larger in scope. The prompt was very detailed and described a ten switch rollout across two site roles.

> "We're going to automate a fleet of 10 Cisco Catalyst switches across two sites: 5 Catalyst 9300 switches acting as core/distribution, and 5 Catalyst 9200 switches acting as access-layer switches. All switches run IOS-XE. Use the cisco.ios collection for all platform-specific tasks (this is the correct collection for IOS-XE-based Catalyst switches — not cisco.nxos, which is for Nexus data-center switches, or cisco.iosxr, which is for service-provider routers). Build playbooks that do the following, in order:
>
> 1. Patch/firmware install — install the latest approved IOS-XE image on each switch.
> 2. Port configuration — configure/open the specific access ports required per switch role (core vs. access).
> 3. Staged reboot — reboot switches in a controlled, non-disruptive order (not all at once), and confirm each switch comes back online before moving to the next."

Scenario 003: Ansible 2.4 to 2.14 upgrade

This scenario involved upgrading a full set of playbook files.

"We want to upgrade the following playbooks from Ansible version 2.4 to Ansible version 2.14. The playbooks are confirmed to be compatible with the newer version and should maintain their original functionality after the upgrade. Below are the files, compatible with Ansible 2.4, that need to be upgraded to Ansible 2.14. For each file, a relative path is given, and files are split using a === FILE: relative/path === delimiter.

Extra instructions:

- Preserve every listed file and relative path. Do NOT rename the files.
- Preserve behaviour and functionality: make only the changes required for Ansible 2.14 compatibility."

Results

SCENARIO	MODEL	ERRORS	WARNINGS	HINTS
001 Nginx static site	DeepSeek-V4-Flash	0	2	2
	claude-sonnet-5	12	7	8
	gpt-5.6-terra	0	3	5
002 Cisco Catalyst fleet automation	DeepSeek-V4-Flash	8	10	26
	claude-sonnet-5	38	17	26
	gpt-5.6-terra	0	7	27
003 Ansible 2.4 to 2.14 upgrade	DeepSeek-V4-Flash	45	19	53
	claude-sonnet-5	34	18	34
	gpt-5.6-terra	3	11	40

Scenario: 001-nginx-static-site

MODEL
DeepSeek-V4-Flash

ERRORS
No findings.

WARNINGS
- **W1800** (x2) — Default value for this parameter changed between module versions.

MODEL
claude-sonnet-5

ERRORS
- **E903** (x12) — Use a fully-qualified name instead of short name.

WARNINGS
- **W1800** (x6) — Default value for this parameter changed between module versions.
- **W2603** (x1) — External Python imports in module.

MODEL
gpt-5.6-terra

ERRORS
No findings.

WARNINGS
- **W1800** (x3) — Default value for this parameter changed between module versions.

Scenario: 002-cisco-catalyst-fleet-automation

MODEL
DeepSeek-V4-Flash

ERRORS
- **E001** (x7) — Not a valid parameter for module.
- **E900** (x1) — Cannot find module information in the database.

WARNINGS
- **W1800** (x7) — Default value for this parameter changed between module versions.
- **W2603** (x2) — External Python imports in module.
- **W700** (x1) — Use of 'ignore_errors' is discouraged. Consider using 'failed_when' instead.

MODEL
claude-sonnet-5

ERRORS
- **E1902** (x21) — Collection version from requirements.yml does not exist.
- **E903** (x17) — Use a fully-qualified name instead of short name.

WARNINGS
- **W1800** (x9) — Default value for this parameter changed between module versions.
- **W2603** (x4) — External Python imports in module.
- **W700** (x4) — Use of 'ignore_errors' is discouraged. Consider using 'failed_when' instead.

MODEL
gpt-5.6-terra

ERRORS
No findings.

WARNINGS
- **W1800** (x6) — Default value for this parameter changed between module versions.
- **W2603** (x1) — External Python imports in module.

Scenario: 003-upgrade-playbook

MODEL
DeepSeek-V4-Flash

ERRORS
- **E903** (x39) — Use a fully-qualified name instead of short name.
- **E601** (x3) — Task parameter is set to a value that does not match any of the possible choices.
- **E001** (x2) — Not a valid parameter for module.
- **E010** (x1) — A given sub-parameter should be specified inside a specific parameter.

WARNINGS
- **W1800** (x12) — Default value for this parameter changed between module versions.
- **W2602** (x4) — Unknown Python imports in module.
- **W2603** (x2) — External Python imports in module.
- **W1100** (x1) — Use of 'with_items' is discouraged. Consider using 'loop' instead.

MODEL
claude-sonnet-5

ERRORS
- **E903** (x27) — Use a fully-qualified name instead of short name.
- **E1902** (x7) — Collection version from requirements.yml does not exist.

WARNINGS
- **W1800** (x12) — Default value for this parameter changed between module versions.
- **W2602** (x4) — Unknown Python imports in module.
- **W2603** (x1) — External Python imports in module.
- **W1100** (x1) — Use of 'with_items' is discouraged. Consider using 'loop' instead.

MODEL
gpt-5.6-terra

ERRORS
- **E903** (x1) — Use a fully-qualified name instead of short name.
- **E601** (x1) — Task parameter is set to a value that does not match any of the possible choices.
- **E001** (x1) — Not a valid parameter for module.

WARNINGS
- **W1800** (x5) — Default value for this parameter changed between module versions.
- **W2603** (x4) — External Python imports in module.
- **W2602** (x2) — Unknown Python imports in module.

XLAB Steampunk

STEAMPUNK SPOTTER FOR ENTERPRISE

The governance and AI guardrail for Ansible and Terraform

Spotter applies shift-left verification and enforcement of your standards to every Ansible Playbook and Terraform configuration before execution – whether written by your team or AI.

- Trusted by Red Hat
- RHEL certified
- ISO certified
- CIS compliant
- On-prem & cloud

USE CASES

SECURITY AND COMPLIANCE

Enhance Security and Compliance

Identify and prevent security and compliance risks before they impact your automation.

Zero
High-Risk Issues After Remediation

AI VERIFICATION

Verify AI-assisted content

Validate AI-generated Ansible Playbooks and Terraform code against the same standards as scripts written by your team.

82%
Fewer Playbook Errors

UPGRADE AND MIGRATION

Upgrade and Migrate Scripts

Automate and accelerate every step of your upgrade and migration process.

8x
Faster

SCRIPT STANDARDIZATION

Enable Script Standardization

Ensure your scripts follow best practices and stay maintainable, secure, and scalable.

70%
Productivity Boost

FROM STANDARDS TO ENFORCEMENT

One governance layer for your automation

1 DEFINE
Set the standards your automation must meet

Custom policies | 200+ checks

2 VALIDATE
Check every change against your standards

Script Scanning | Compliance Validation | Supply Chain Management | SBOM&CVE Analysis

3 ENFORCE
Gate changes before they reach production

Integrations

4 PROVE
See what passed, what failed, and why

Analytics and Reporting

Want to learn more how Spotter can fit into your AI workflow?

Book a demo