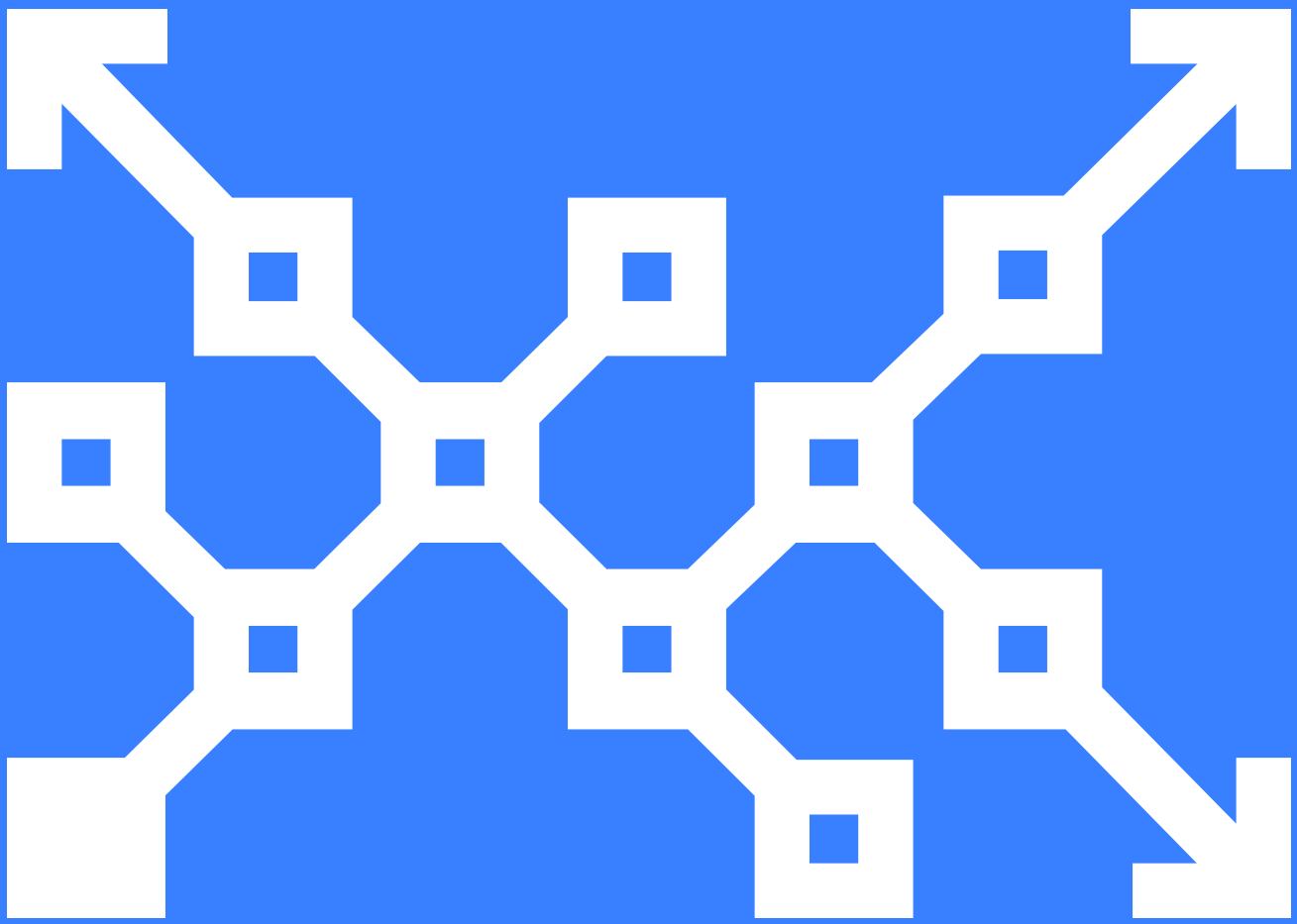
WHITE PAPER

Compliance: Comparing Puppet and Ansible



puppet

Summary

In many ways, Puppet and Ansible are quite complementary. When it comes to ensuring compliance with security policies, however, Puppet is the only tool that can guarantee configurations are being enforced in a unified way, and the only tool that can report on everything that's being managed on hosts — including configurations not managed by Puppet.

The following table contains the key capabilities that organizations look for when deciding on a solution for managing and enforcing security policy configurations. These capabilities will be addressed throughout this document.

Ansible and Puppet compliance comparison

Capabilities	Puppet	Ansible
Corrective vs. intentional change reporting • **What it is:** Knowing if a change was made because of configuration drift, or whether it was an intentional change to the desired configuration. • **Why it matters:** It's easier to assess just the unintentional changes. That means less time gets spent analyzing security and/or policy compliance issues.	✔ Puppet can identify whether a change was made because the system had drifted out of compliance, or because the Puppet code was intentionally changed.	✖ Ansible treats all system changes the same. It cannot identify why the change was made, so you cannot know if the change was made because the system was out of compliance, or for a different reason.

Capabilities	Puppet	Ansible
Desired-state conflict detection • **What it is:** Knowing when two bits of infrastructure code are attempting to manage the same configuration differently. • **Why it matters:** Managing resource configurations in multiple different ways complicates audit reporting, and could potentially mask a vulnerability. Puppet prevents this.	✓ Puppet detects whether two different pieces of Puppet code are attempting to manage the same configuration differently. Puppet does this prior to any work being done on the system, preventing the system from being left in a noncompliant state. Puppet does not require a compliance scanning tool to verify that configurations are in a compliant state.	✗ It is not possible to know whether the system is in a compliant state if you're using only Ansible.You need a separate compliance scanning tool to guarantee compliance. Multiple Ansible playbooks can silently manage the same resource differently. For example, if one playbook says OpenSSL should be version 1.1.0f and another says it should be 1.1.0d, the last playbook run wins. Even the same Ansible playbook can manage the package differently at different parts of the playbook. If multiple playbooks are managing the same system, configuration conflicts can be nearly impossible to detect.
Single source of truth • **What it is:** A single place to see all the infrastructure code that is ever run on a system or systems. • **Why it matters:** Ensuring compliance requires knowledge of all configurations being managed on a system over time. This can be accomplished only if you can inventory all the infrastructure code being applied to it.	✓ Puppet's standard mode of operation is to have all the Puppet code that is available for an environment live on the Puppet master. This means that all the Puppet code being applied to systems can be read in one place — the Puppet code repository. Puppet also tracks the version of Puppet code applied to any system, so you can always track precisely which code was applied to any system, whenever you want.	✗ Ansible's standard mode of operation is to have a playbook for each task or part of the system to be managed. Because users often run playbooks from their workstations or from a continuous delivery pipeline, you must track several sources of truth in order to know precisely which code goes into the management of any given system.

Capabilities	Puppet	Ansible
Access audits • **What it is:** Knowing who or what has accessed the system to initiate a system scan or change. • **Why it matters:** Knowing who or what has access to a system, especially with change permissions, is critical to ensuring that only those have who should authorized access actually have it.	✔ Puppet centrally controls access to systems under Puppet management. Changes are applied through the Puppet code repository, or through the Puppet node classifier. Both of these give you a full log for checking access and auditing change.	✖ Ansible operates over SSH. Anything, or anyone, can make changes to a system over SSH. If all that's required to manage a host is SSH access, you can't reliably know who or what is making changes in all cases.
Unmanaged configuration reporting • **What it is:** Knowing which packages are not being managed by Puppet and identifying those that may be vulnerable to security risks. • **Why it matters:** Comprehensive visibility into the software packages you have running (whether managed by Puppet or not) provides the insight you need to identify security issues faster.	✔ Puppet Enterprise can list all the packages in your infrastructure, from all package managers, including custom ones. Puppet Enterprise shows the various versions installed on all systems, so you can identify where vulnerabilities exist and bring those systems under management.	✖ Ansible does not have the ability to inspect and report on any configurations on a system that Ansible isn't actually managing.
Centralized and configuration-centric reporting • **What it is:** Reporting that gives you a central place to gain awareness of what's changing, viewed through the lens of configurations managed in your infrastructure. You don't have to decipher generic run reports. • **Why it matters:** To guarantee that configurations are being managed according to policy, you need to know which configurations are being managed, where they're changing, and why they're changing.	✔ All Puppet runs send their reports back to the central master. The master sorts any changes into different contexts such as role, host, or configuration. You know at a glance how pervasive a change was, and how it relates to other configurations on your systems. Puppet reports are structured to enable dozens of integrations with monitoring and reporting tools such as HipChat, JIRA, Pager Duty, Datadog, Graphite, Splunk, and many more.	✖ Ansible reports change as the output of executing a playbook. Integrations with external logging systems are required to centralize, aggregate, and correlate change occurring across Ansible runs. Ansible reports are printed to the command console and have to be parsed and searched using string manipulation tools such as grep and awk. They can't be shared easily with colleagues outside the core technical teams.

Capabilities	Puppet	Ansible
Continuous enforcement • **What it is:** Continuously ensuring a configuration is in its proper state. • **Why it matters:** Drift can happen on any mutable system for myriad reasons. When a change is made, it's critical for you to know the system will not change again until a change is deliberately made, in line with business strategy or policy.	✔ Puppet's agent-based approach means the agent can continuously monitor for changes without checking into a central master for every run. Complete idempotence is a requirement for running continuously. Idempotence is built into the core of Puppet's framework.	– Running continuously requires complete idempotency. If you want your Ansible playbooks to be idempotent, then Ansible modules would have to feature idempotency. Idempotency is not built into the core of how Ansible works, so it needs to be manually written into Ansible modules and playbooks. Further, in order to test for idempotency, you have to write tests to ensure that the content is idempotent in all potential scenarios.
Unmanaged resource purging • **What it is:** Removing configurations that shouldn't be present on the system. • **Why it matters:** For certain configuration types, it's important to know that only the configurations that are supposed to be present on systems are, in fact, the only ones present. For example, old or unauthorized firewall rules should be automatically removed from systems. SSH keys and Apache vhosts are other examples.	✔ All Puppet runs send their reports back to the central master. The master sorts any changes into different contexts such as role, host, or configuration. You know at a glance how pervasive a change was, and how it relates to other configurations on your systems. Puppet reports are structured to enable dozens of integrations with monitoring and reporting tools such as HipChat, JIRA, Pager Duty, Datadog, Graphite, Splunk, and many more.	✖ Ansible reports change as the output of executing a playbook. Integrations with external logging systems are required to centralize, aggregate, and correlate change occurring across Ansible runs. Ansible's reports are printed to the command console and have to be parsed and searched using string manipulation tools such as grep and awk. They can't be shared easily with colleagues outside the core technical teams.

Managing and proving compliance

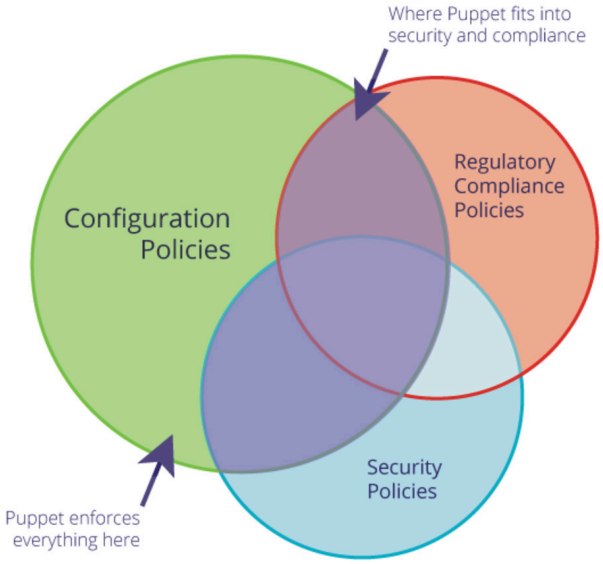
Puppet is a holistic continuous-configuration management platform. It provides a single source of truth for every configuration managed on every system in your infrastructure. Puppet provides unparalleled reporting capabilities that let you prove your systems are in a compliant state at any time, plus know when your systems fall out of compliance and when they are remediated. The powerful Puppet query language (PQL) lets you build custom reports that answer the questions your business needs to ask.

Detecting conflict

Puppet has desired-state conflict detection built right into how Puppet works. Unlike nearly every other infrastructure management tool, Puppet does not blindly execute a series of commands as it reads them. Puppet reads and understands every line of infrastructure code prior to doing any work. As a result, Puppet has the intelligence to know every desired configuration for every system, and every dependency between configurations, even if the dependencies span multiple systems.

This deep knowledge means Puppet can detect when two different parts of the infrastructure code (or more) are attempting to manage the same configuration differently. This is useful when multiple teams are writing infrastructure code, or when you have multiple contributors on the same team.

Ansible simply reads and executes instructions line by line as it reads the playbook. That means one part of a playbook can change a configuration managed earlier in the playbook. The problem is compounded if you use multiple playbooks to manage different parts of a system. You have no way of knowing if one playbook is modifying a configuration managed by an entirely different playbook. This is especially troublesome if you use a playbook to enforce compliance-related configurations, and use other playbooks to manage other aspects of the system's configuration. As soon as another playbook runs, you can no longer guarantee a system is still compliant.



Single source of truth

You need a single source of truth that clearly outlines just how any compliance-related configuration is to be managed on your infrastructure over time. With Puppet, you have just one code base per environment — and that code base is also your documentation, easily accessed and understood by auditors, or anyone else on your team.

The precision of this executable documentation stands in sharp contrast to the confusion inherent in the way tools like Ansible work: by employing a collection of playbooks, each running at various times across a subset of infrastructure systems, and often with multiple playbooks running at different times on the same system.

Continuously enforcing infrastructure state

It's never enough to simply apply compliant configurations. You must be sure that any system will remain in its compliant state until you're ready to change it again. That kind of confidence depends on multiple different capabilities coming together. You need one tool that can make changes; continuously monitor and enforce desired state over time; report on current state; and do all this at a potentially huge scale, both in terms of the number of systems and the amount of information collected.

Enforcement over time

When you deploy something, you need to know it will remain in that state until you're ready for it to change. Puppet deploys new configurations and services, and also continuously enforces their correct state. If Puppet detects configuration drift, it will automatically correct the drift, and report that both the drift and its correction took place.

Monitoring and reporting

One of the great things about managing your infrastructure as code is the ability to replicate work across hundreds or thousands of systems without any increase in human effort. **Effective monitoring at this scale is critical.** In addition to monitoring change itself, it's equally important to quickly identify where and why change occurred. You must continually verify that systems remain in the state you left them in, so you can be confident that unauthorized changes have not occurred.

Puppet Enterprise includes powerful reporting tools, so you'll always know when a single configuration has changed across 1,000 systems — or when one system experiences 1,000 changes. Puppet Enterprise tells you at a glance, in real time, how many systems are in the desired state, which require correction, and how many are out of communication. Puppet Enterprise can inform you whether a change was caused by configuration drift or a change to the infrastructure code. This ability to report on infrastructure changes, the reasons for change, and the current state — at any scale — is unique to Puppet.

Corrective vs. intentional change

Puppet Enterprise tells you whether a system change was a corrective change or an intentional change. When Ansible or Puppet Enterprise make a change to a system, it's for one of two reasons:

1. The system's configuration was changed since its deployment by something other than the automation tool.
2. The automation tool's instructions changed. We call these changes "corrective changes" vs. "intentional changes."

Only Puppet Enterprise can distinguish between the two. Puppet Enterprise informs you if a system change was required because someone or something put the system out of compliance since the last time Puppet checked the system's state, or if the system change was required because the Puppet code governing its configurations has changed.

Infrastructure at scale

If you have a large number of systems to manage, you know how important it is for a tool to work well at scale. A single Puppet server can manage up to 30,000 systems. Large enterprises like Walmart have successfully scaled Puppet Enterprise to manage more than 100,000 systems running a combination of Unix, Linux, and Windows.

Questions to ask when comparing tools

As you consider which are the right tools for your organization, it helps to ask some questions about how your organization is running now, and where you think you are going. The questions below should help you make your decision.

How many contributors to the infrastructure code base will you have?

If the answer is more than a few, you need to plan how you're going to scale everyone's contributions to the infrastructure code base. You will want to make sure your tool of choice can:

- Prevent a single configuration from being managed multiple ways by different contributors.
- Integrate with existing pipelines and testing practices.

What are your audit requirements?

If you need to be able to prove the infrastructure is always in its desired state, your new tool has to be great at managing state over time and continuously reporting when systems are in the desired state; when drift is detected and remediated; and when new infrastructure changes are deployed. Compliance audits go much more smoothly when you can audit infrastructure changes — including change approvals — in the infrastructure code base itself.

How will you identify when and where configuration drift occurred, and how long the systems were out of compliance?

One key way to ensure your configuration policies are enforced is to have a tool that continuously monitors for configurations that have drifted from compliance; automatically corrects the configuration drift; and reports on the correction.

When Puppet changes a configuration on a system, it understands if the change was due to the desired state changing (as defined in Puppet code) or due to the system changing between Puppet checks.

How do I get a list of past configuration remediation events?

Puppet tracks every managed configuration, and every change a configuration has experienced over time. Using Puppet's powerful query language), you can ask Puppet for a list of all managed configurations that have experienced a corrective change (also known as a remediation) within a set period of time. You can then get a report from your query, proving that configurations are being managed, monitored, and corrected when drift is detected.

How will you prove all the infrastructure code applied to a system or systems?

Because Puppet centralizes all the code that manages any part of the infrastructure into one code repository, it's easy to prove everything that's being managed, how it's being managed now, and how it was being managed in the past. Combined with Puppet Enterprise's reporting capabilities, you can not only report on everything that's being managed, but also prove those configurations are actually enforced over time.

Can you prove the state of a given configuration at any moment?

Puppet runs continuously and reports on what, if anything, changed. The report proves that all managed configurations are either still as you left them, or that they required a correction.

What will you give to the auditor to show which configuration policies are defined, and prove the policies are being continuously enforced?

The reporting history in Puppet Enterprise provides an auditor proof that what you're managing is in fact being managed, that you're continuously monitoring for drift, and that any drift is corrected within a very short time frame. Further, it shows that you have a fully defined change management process, that you use it, and that the process is difficult, if not impossible, to circumvent.

How do you know when a system has stopped checking for drift?

Puppet Enterprise will report when a system has stopped checking in or stopped sending regular reports. This enables you to identify where a system is not being monitored for configuration compliance, so you can take action.

What will an implementation look like at your scale?

Puppet can quickly scale to tens of thousands of systems. A single Puppet server can handle 30,000 nodes, while customers such as Walmart are managing compliant configurations continuously on over 100,000 systems.

Puppet Enterprise's orchestrator can scale just as easily, not only in terms of nodes managed, but infrastructure complexity. With Puppet, you model only the immediate dependencies an infrastructure service has on other infrastructure services. Puppet Enterprise's orchestrator then creates an environment graph to automatically determine things like order of operations and secure, automatic information sharing between systems.

Another important consideration is organizational scale. The tool you select will need to address multiple individual contributors and multiple teams that could manage any given system. This is where features like desired-state conflict detection become so critical.

How will you test proposed changes to your infrastructure code?

As you add systems, the potential for code changes to cause unintended changes grows. Puppet code can be tested using rspec, and those tests can be run in a continuous integration (CI) environment. You can use Beaker, Puppet's testing framework, to automatically test Puppet code, ensuring both code quality and that your Puppet code meets compliance requirements — all before it ever gets to production.

How will you manage access control for your configuration management system?

Puppet Enterprise manages access control within its console, including a rich role-based access control (RBAC) system that can be used to grant specific permissions where required. Ansible requires a user on every system to have SSH permission and (in the common case) sudo access, so they can run arbitrary commands as an administrator. In many environments, this user needs to be managed with the same compliance regime as other interactive users, including credential rolling schedules and auditing, and those changes need to be orchestrated with the Ansible control system to keep from breaking its access. In high-security environments, there may be a requirement for detailed logging and monitoring of what this user does on each system.

We want to hear from you.

We hope we've helped you understand the differences between Ansible and Puppet Enterprise, and that we've given you some good questions that you and your team can ask yourselves to determine which solution is right for you.

If you have any concerns or questions that this document didn't answer, or if you find anything in this document is inaccurate, please reach out to your Puppet representative. We will be happy to help.

puppet