Puppet for Configuration Management and Automated Provisioning

Source: https://www.puppet.com/docs/puppet/6/puppet_overview

Puppet is a tool that helps you manage and automate the configuration of servers.

When you use Puppet, you define the *desired state* of the systems in your infrastructure that you want to manage. You do this by writing infrastructure code in Puppet's Domain-Specific Language (DSL) — Puppet Code — which you can use with a wide array of devices and operating systems. Puppet code is *declarative*, which means that you describe the desired state of your systems, not the steps needed to get there. Puppet then *automates* the process of getting these systems into that state and keeping them there. Puppet does this through Puppet *primary server* and a Puppet *agent*. The Puppet primary server is the server that stores the code that defines your desired state. The Puppet agent translates your code into commands and then executes it on the systems you specify, in what is called a Puppet run.

The diagram below shows how the server-agent architecture of a Puppet run works. This is also known as Puppet workflow diagram.

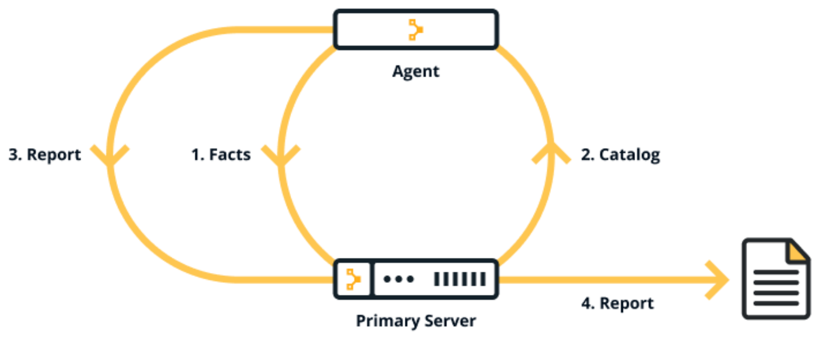

Fig. 1 Puppet Workflow – The Server-Agent Architecture

The primary server and the agent are part of the Puppet platform.

There are many benefits to implementing a declarative configuration tool like Puppet into your environment — most notably *consistency* and *automation*.

- **Consistency**. Troubleshooting problems with servers is a time-consuming and manually intensive process. Without configuration management, you are unable to make assumptions about your infrastructure — such as which version of Apache you have or whether your colleague configured the machine to follow all the manual steps correctly. But when you use configuration management, you are able to validate that Puppet applied the desired state you wanted. You can then assume that state has been applied, helping you to identify why your model failed and what was incomplete, and saving you valuable time in the process. Most importantly, once you figure it out, you can add the missing part to your model and ensure that you never have to deal with that same problem again.

- **Automation.** When you manage a set of servers in your infrastructure, you want to keep them in a certain state. If you only have to manage homogeneous 10 servers, you can do so with a script or by manually going into each server. In this case, a tool like Puppet may not provide much extra value. But if you have 100 or 1,000 servers, a mixed environment, or you have plans to scale your infrastructure in the future, it is difficult to do this manually. This is where Puppet can help you — to save you time and money, to scale effectively, and to do so securely.

Key concepts

Using Puppet is not just about the tool, but also about a different culture and a way of working. The following concepts and practices are key to using and being successful with Puppet.

Infrastructure-as-code

Puppet is built on the concept of *infrastructure-as-code*, which is the practice of treating infrastructure as if it were code. This concept is the foundation of DevOps — the practice of combining software development and operations. Treating infrastructure as code means that system administrators adopt practices that are traditionally associated with software developers, such as version control, peer review, automated testing, and continuous delivery. These practices that test code are effectively testing your infrastructure. When you get further along in your automation journey, you can choose to write your own unit and acceptance tests — these validate that your code, your infrastructure changes, do as you expect.

Idempotency

A key feature of Puppet is *idempotency* — the ability to repeatedly apply code to guarantee a desired state on a system, with the assurance that you will get the same result every time. Idempotency is what allows Puppet to run continuously. It ensures that the state of the infrastructure always matches the desired state. If a system state changes from what you describe, Puppet will bring it back to where it is meant to be. It also means that if you make a change to your desired state, your entire infrastructure automatically updates to match. Idempotency holds good for Chef as well.

Agile methodology

When adopting a tool like Puppet, you will be more successful with an *agile methodology* in mind — working in incremental units of work and reusing code. Trying to do too much at once is a common pitfall. The more familiar you get with Puppet, the more you can scale, and the more you get used to agile methodology, the more you can democratize work. When you share a common methodology, a common pipeline, and a common language (the Puppet language) with your colleagues, your organization becomes more efficient at getting changes deployed quickly and safely.

Git and version control

Git is a *version control* system that tracks changes in code. While version control is not required to use Puppet, it is highly recommended that you store your Puppet code in a Git repository. Git is the industry standard for version control, and using it will help your team gain the benefits of the DevOps and agile methodologies

When you develop and store your Puppet code in a Git repository, you will likely have multiple branches — feature branches for developing and testing code and a production branch for releasing code. You test all of your code on a feature branch before you merge it to the production branch. This process, known as Git flow, allows you to test, track, and share code, making it easier to collaborate with colleagues. For example, if someone on your team wants to make a change to an application's firewall requirements, they can create a pull request that shows their proposed changes to the existing code, which everyone on your team can review before it gets pushed to production. This process leaves far less room for errors that could cause an outage.

The Puppet platform

Puppet is made up of several packages. Together these are called the Puppet platform, which is what you use to manage, store and run your Puppet code. These packages include `puppetserver`, `puppetdb`, and `puppet-agent` — which includes *Facter* and *Hiera*.

Puppet is configured in an agent-server architecture, in which a primary node (system) controls configuration information for one or more managed agent nodes. Servers and agents communicate by HTTPS using SSL certificates. Puppet includes a built-in certificate authority for managing certificates. Puppet Server performs the role of the primary node and also runs an agent to configure itself.

Facter, Puppet’s inventory tool, gathers *facts* about an agent node such as its hostname, IP address, and operating system. The agent sends these facts to the primary server in the form of a special Puppet code file called a *manifest*. This is the information the primary server uses to compile a *catalog* — a JSON document describing the desired state of a specific agent node. Each agent requests and receives its own individual catalog and then enforces that desired state on the node it's running on. In this way, Puppet applies changes all across your infrastructure, ensuring that each node matches the state you defined with your Puppet code. The agent sends a report back to the primary server.

You keep nearly all of your Puppet code, such as manifests, in *modules*. Each module manages a specific task in your infrastructure, such as installing and configuring a piece of software. Modules contain both code and data. The data is what allows you to customize your configuration.

Using a tool called *Hiera*, you can separate the data from the code and place it in a centralized location. This allows you to specify guardrails and define known parameters and variations, so that your code is fully testable and you can validate all the edge cases of your parameters. If you have just joined an existing team that uses Puppet, take a look at how they organize their Hiera data.

All of the data generated by Puppet (for example facts, catalogs, reports) is stored in the *Puppet database* (PuppetDB). Storing data in PuppetDB allows Puppet to work faster and provides an API for other applications to access Puppet's collected data. Once PuppetDB is full of your data, it becomes a great tool for infrastructure discovery, compliance reporting, vulnerability assessment, and more. You perform all of these tasks with PuppetDB queries.

The diagram below shows how the Puppet components fit together.

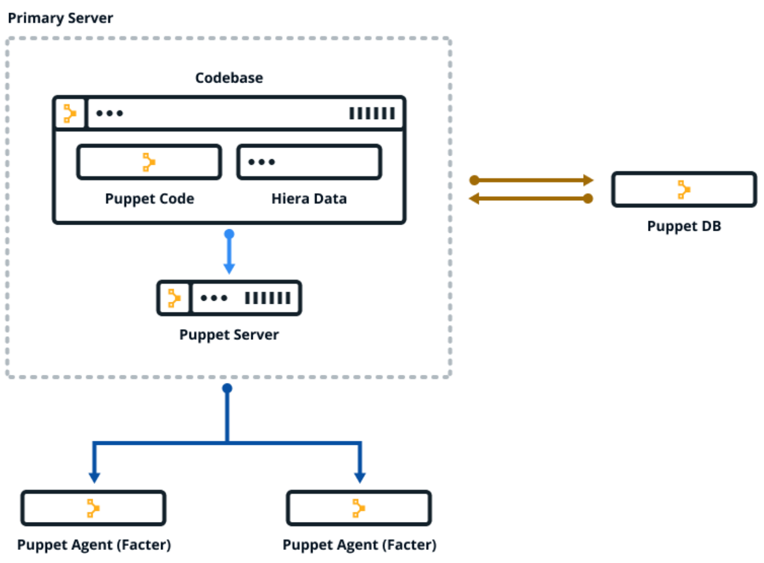

Fig. 2 Puppet Architecture and its Components

Open-source Puppet vs Puppet Enterprise (PE)

Puppet Enterprise (PE) is the commercial version of Puppet and is built on top of the Puppet platform. Both products allow you to manage the configuration of thousands of nodes. Open-source Puppet does this with desired state management. PE provides an imperative, as well as declarative, approach to infrastructure automation.

If you have a complex or large infrastructure that is used and managed by multiple teams, PE is a more suitable option, as it provides a graphical user interface, point-and-click code deployment strategies, continuous testing and integration, and the ability to predict the impact of code changes before deployment.

The Puppet ecosystem

Alongside Puppet the configuration tool, there are additional Puppet tools and resources to help you use and be successful. These make up the Puppet ecosystem

Install existing modules from Puppet Forge

Puppet Forge is a catalogue of modules created by Puppet, Puppet’s partners, and community that helps IT ops practitioners supercharge and simplify their automation processes. With step-by-step guides and tutorials, Puppet Forge provides a platform for customers to grow their skills with Puppet.

Modules manage a specific technology in your infrastructure and serve as the basic building blocks of Puppet desired state management. On the Puppet Forge, there is a module to manage almost any part of your infrastructure. Whether you want to manage packages or patch operating systems, a module is already set up for you. See each module’s README for installation instructions, usage, and code examples.

When using an existing module from the Forge, most of the Puppet code is written for you. You just need to install the module and its dependencies and write a small amount of code (known as a profile) to tie things together.

You keep nearly all of your Puppet code, such as manifests, in *modules*. Each module manages a specific task in your infrastructure, such as installing and configuring a piece of software. Modules contain both code and data. The data is what allows you to customize your configuration.

Modules serve as the basic building blocks of Puppet and are reusable and shareable.

Modules contain Puppet classes, defined types, tasks, task plans, functions, resource types and providers, and plug-ins such as custom types or facts. Modules must be installed in the Puppet modulepath. Puppet loads all content from every module in the modulepath, making this code available for use.

You can download and install modules from the Puppet Forge. The Forge contains thousands of modules written by Puppet developers and the open-source community for a wide variety of use cases. Expect to write at least a few of your own modules to meet specific needs in your infrastructure.

More info on Modules and Manifests:
https://www.puppet.com/docs/puppet/7/modules_fundamentals.html

Video: https://www.youtube.com/watch?v=WWTV7Kz41Go

Develop existing or new modules with Puppet Development Kit (PDK)

You can write your own Puppet code and modules using Puppet Development Kit (PDK), which is a framework to successfully build, test and validate your modules.

Write Puppet code with the VSCode extension

The *Puppet VSCode extension* makes writing and managing Puppet code easier and ensures your code is high quality. You can use the extension with Windows, Linux, or macOS.

Run acceptance tests with Litmus

Litmus is a command line tool that allows you to run acceptance tests against Puppet modules for a variety of operating systems and deployment scenarios. Acceptance tests validate that your code does what you intend it to do.

Use cases

Puppet Forge has existing modules and code examples that assist with automating the following use cases:

- **Base system configuration** (Including registry, NTP, firewalls, services)
- **Manage web servers** (Including apache, tomcat, IIS, nginx)
- **Manage database systems** (Including Oracle, SQL Server, MySQL, PostgreSQL)
- **Manage middleware/application systems** (Including Java, WebLogic/Fusion, IBM MQ, IBM IIB, RabbitMQ, ActiveMQ, Redis, ElasticSearch)
- **Source control** (Including Github, Gitlab)
- **Monitoring** **(**Including Splunk, Nagios, Zabbix, Sensu, Prometheus, NewRelic, Icinga, SNMP)
- **Patch management** (OS patching on Enterprise Linux, Debian, SLES, Ubuntu, Windows)
- **Package management** (Linux - Puppet integrates directly with native package managers, Windows - Use Puppet to install software directly on Windows, or integrate with Chocolatey)
- **Containers and cloud native** (Including Docker, Kubernetes, Terraform, OpenShift)
- **Networking** (Including Cisco Catalyst, Cisco Nexus, F5, Palo Alto, Barracuda)
- **Secrets management** (Including Hashicorp Vault, CyberArk Conjur, Azure Key Vault, Consul Data)

See each module's README for installation, usage, and code examples.

If you don't see your use case listed above, have a look at the following list to see what else we might be able to help you with:

- **Continuous integration and delivery of Puppet code**
    - *Continuous Delivery for Puppet Enterprise* (PE) offers a prescriptive workflow to test and deploy Puppet code across environments. To harness the full power of PE, you need a robust system for testing and deploying your Puppet code. Continuous Delivery for PE offers prescriptive, customizable work flows and intuitive tools for Puppet code testing, deployment, and impact analysis — so you know how code changes will affect your infrastructure before you

deploy them — helping you ship changes and additions with speed and confidence.

- **Incident remediation**
    - If you need to minimize the risk of external attacks and data breaches by increasing your visibility into the vulnerabilities across your infrastructure, take a look at *Puppet Remediate*. With Remediate, you can eliminate the repetitive and error-prone steps of manual data handovers between teams.
- **Integrate Puppet into your existing workflows**
    - You can integrate Puppet with your existing workflows involving other technologies including Splunk and VMware vRA.

Bolt

Bolt is an open-source orchestration tool that automates the manual work it takes to maintain infrastructure. You can use Bolt as a stand-alone tool or integrate it with Puppet. Bolt allows you to automate tasks on an as-needed basis or as part of a greater orchestration workflow.

Use Bolt to automate tasks that you perform on an as-needed basis or as part of a greater orchestration workflow. For example, you can use Bolt to patch and update systems, troubleshoot servers, deploy applications, or stop and restart services. Bolt can be installed on your local workstation and connects directly to remote targets with SSH or WinRM, so you are not required to install any agent software.

Installation: https://www.puppet.com/docs/puppet/6/server/install_from_packages

Advantages of Puppet

1. The Puppet Domain Specific Language (DSL) combined with Ruby functions helps implement complex requirements.
2. The official Puppet website offers a complete documentation on the software with good explanations and examples.
3. Puppet Forge provides a great number of modules to download and import directly in a configuration.

Disadvantages of Puppet

1. The initial configuration of Puppet is lengthy and takes time as we need to setup the Puppet master and install agents on the nodes. Chef installation is simpler and shorter.
2. Puppet’s architecture and the configuration language are complex to learn. Hence the learning curve is sharp if Ruby is new to the team members.
3. Puppet environment is complex and Puppet does not support push functionality.