Puppet Tech Talk Report

1. The overview and introduction of Puppet

1) The overview

Puppet is a open-source configuration management tool that is extremely powerful in deploying, configuring, managing, maintaining for a server machine. In other words, you can say that puppet can be used for an entire life of a server, starting from bootstrapping a server, to shredding and giving up a server. It provides control and enforces consistency, while allowing for any of the modifications dictated by business needs. As shortly, Puppet can automatically deliver and operate all of your software across its entire lifecycle. Doesn't like Ansible, The user describes system resources and their state, either using Puppet's declarative language or a Ruby DSL.

2) The comparison before and after using Puppet

Previous	**Present**
Manual installation	Mature tool
Login and perform inst/config change	Scalable
Not scalable	Automated way to inspect,operate,deliver
Everyone solve same problem with their own way	Version controlled

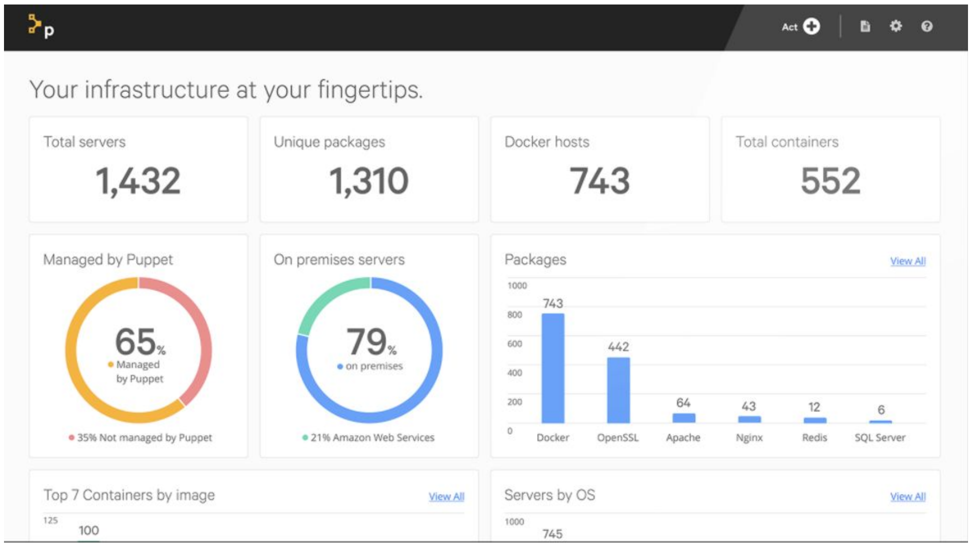
2. Products

1) Puppet Discovery

The very first step of incorporate your product into CI (Continuous Integration) is to figure out what you currently have. With the help of Puppet Discovery, we can answer questions like these:

- What's the distribution of operating systems in your on-premises environment?
- In which regions are your AWS EC2 instances running?
- What files have changed in your containers since you deployed them?

With full visibility into your infrastructure, plus meaningful insights provided in the dashboard, you can easily see what you have and what you need to automate next in the dashboard of Puppet Discovery.



2) Puppet Enterprise

Puppet Enterprise helps you scale automation broadly and deeply across your infrastructure and keep it compliant.

Puppet Enterprise helps you improve organizational productivity, go-to-market agility and collaboration across your teams while reducing tool proliferation. Puppet delivers a user-friendly, unified platform that combines a model-driven approach with imperative task execution so you can effectively manage hybrid infrastructure across its entire lifecycle. Puppet Enterprise provides you with the common language that all teams in an IT organization can use to successfully adopt DevOps practices such as version control, code review, automated testing, continuous integration and automated deployment.

Below are some features that Puppet Enterprise provides:

- One platform to enforce desired state and automate tasks
- Discovery and remediation
- Real-time reporting
- Orchestration and visual workflows
- Puppet Enterprise capabilities
- Deliver and operate everything you have.

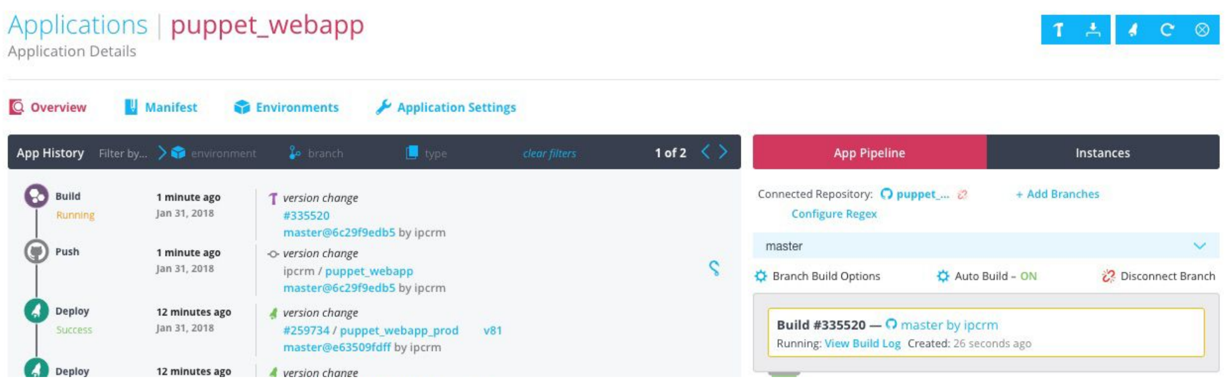
3) Puppet Pipeline

Puppet Pipelines simplifies software delivery and unifies automation silos across your Dev and Ops teams.

With Puppet Pipeline, we get end-to-end continuous delivery and automation for our entire software delivery lifecycle, from the time your developers commit application code to the deployment of those apps wherever they run.

Below are some features that you could leverage with Puppet Pipeline:

- Automatically trigger application builds on every code commit
- Visually create continuous delivery pipelines in minutes
- Easily automate application deployments
- Build and deploy containers to Kubernetes
- Integrate with DevOps tools you already use



Build Running — 1 minute ago, Jan 31, 2018 — *version change* #335520 master@6c29f9edb5 by ipcrm

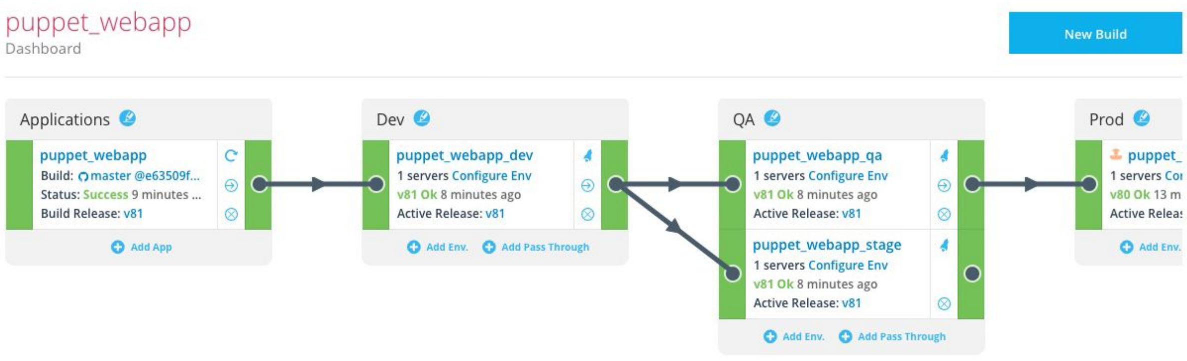
Push — 1 minute ago, Jan 31, 2018 — *version change* ipcrm / puppet_webapp master@6c29f9edb5 by ipcrm

Deploy Success — 12 minutes ago, Jan 31, 2018 — *version change* #259734 / puppet_webapp_prod v81 master@e63509fdff by ipcrm

Deploy — 12 minutes ago — *version change*

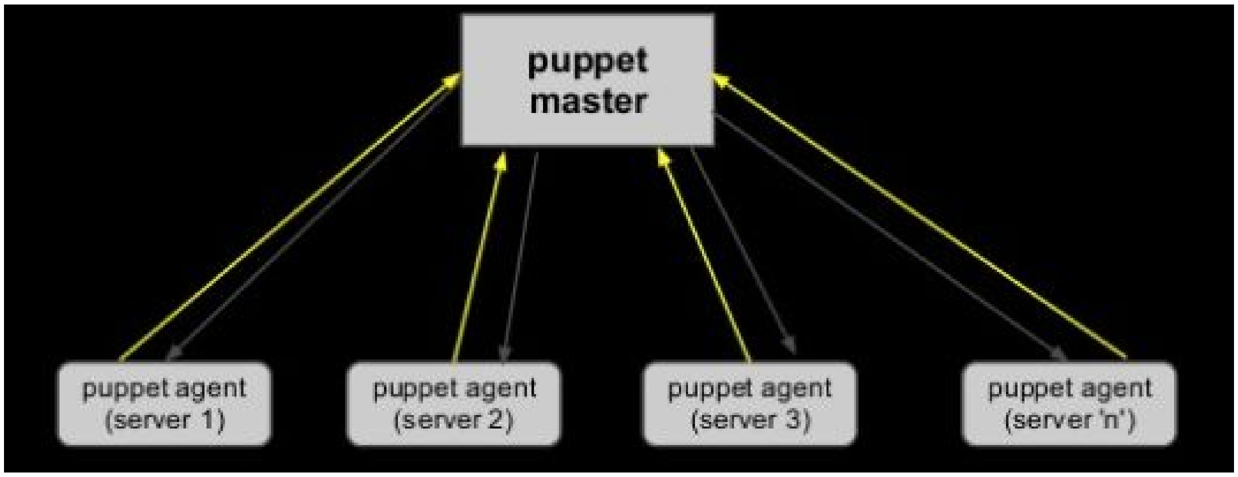
Connected Repository: puppet_... + Add Branches
Configure Regex
master
Branch Build Options — Auto Build - ON — Disconnect Branch

Build #335520 — master by ipcrm
Running: View Build Log Created: 26 seconds ago


puppet_webapp
Dashboard
New Build
Applications
puppet_webapp
Build: master @e63509f...
Status: Success 9 minutes ...
Build Release: v81
Add App
Dev
puppet_webapp_dev
1 servers Configure Env
v81 Ok 8 minutes ago
Active Release: v81
Add Env.
Add Pass Through
QA
puppet_webapp_qa
1 servers Configure Env
v81 Ok 8 minutes ago
Active Release: v81
puppet_webapp_stage
1 servers Configure Env
v81 Ok 8 minutes ago
Active Release: v81
Add Env.
Add Pass Through
Prod
puppet_
1 servers Co
v80 Ok 13 m
Active Relea
Add Env.

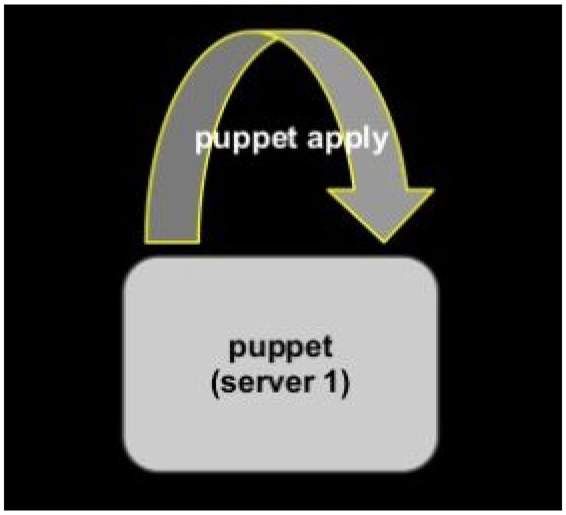
3. How Puppet works

1) **Architecture of Puppet:**
- **Agent/Master architecture:**



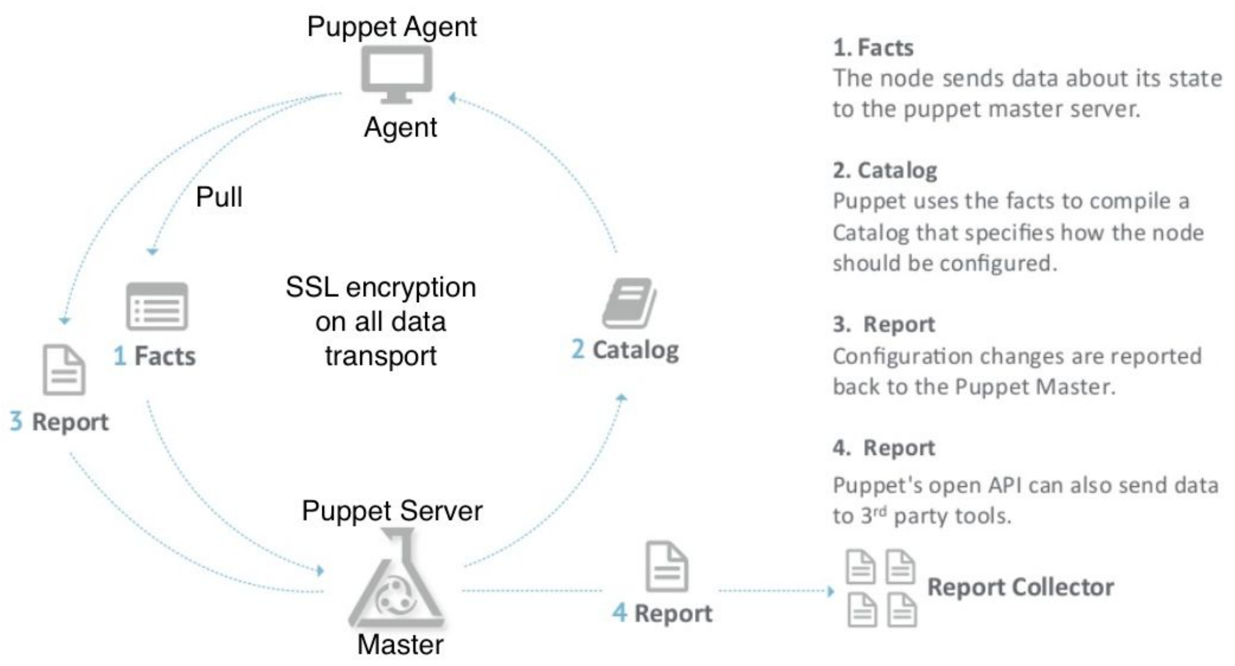
This is a client-server type of architecture where there are one or more servers running the puppet master application - puppet server. The clients or systems which need to be configured run the puppet agent application, usually as a background service. The puppet master server controls the configuration information. Each managed node agents requests its own configuration catalog from the master. This catalog is compiled by the puppet master.

- **Stand-alone architecture:**



This architecture does not have a puppet master. Each managed node has its own configuration information and compiles its own catalog. They run the Puppet Apply application, usually as a scheduled task or cron job, for changing the configuration in order to reach a desired system state.

2) Lifecycle of a puppet run (agent/master architecture):



Puppet Agent
Agent
Pull
SSL encryption on all data transport
1 Facts
2 Catalog
3 Report
Puppet Server
Master
4 Report
Report Collector
1. Facts
The node sends data about its state to the puppet master server.
2. Catalog
Puppet uses the facts to compile a Catalog that specifies how the node should be configured.
3. Report
Configuration changes are reported back to the Puppet Master.
4. Report
Puppet's open API can also send data to 3rd party tools.

The puppet agent starts a configuration pull by sending the facts collected by Facter to the puppet master and requesting a catalog. The puppet master compiles a catalog according to the current state of the host system and the desired system state as given in the configuration information. This catalog specifies how the agent should be configured. The catalog is sent back to the puppet agent. The puppet agent applies this catalog to reach the desired system state. After applying the catalog, it reports the configuration changes to the puppet master. The puppet master can then send these reports to a third party tool. Puppet has a built-in certificate authority (CA) and public key infrastructure (PKI) tools which it uses for all of its secure socket layer (SSL) communications. The client-server communication is done through port 8140.

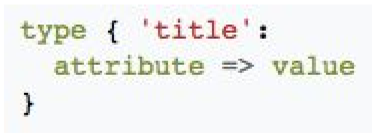
Terms explanation:

Catalog: It shows all the resources as well as any dependencies between those resources that need to be managed to reach the desired system state.

Facter: Puppet uses a utility called Facter which is used to gather information about the host system.

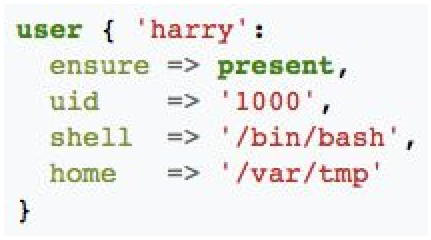
3) Configuration language:

Puppet uses its own configuration language, whose syntax is inspired by the Nagios configuration file format. Puppet follows a declarative language, so user just needs to specify 'what' state the resource needs to be in rather than describing how to bring resource to that state. This action is specified using type, title and a list of attributes as shown below:



```
type { 'title':
  attribute => value
}
```

For example, code below is the syntax to make a user harry with uid 1000, shell '/bin/bash' and home '/var/tmp':



```
user { 'harry':
  ensure => present,
  uid    => '1000',
  shell  => '/bin/bash',
  home   => '/var/tmp'
}
```

This language is written in files called puppet manifests. It has a file extension of .pp which stands for puppet policy. The main part of the language is declaring resources. All the other parts just adds a flexibility and convenience to the way resources are declared. Classes are used to group together resources. Thus, the classes can describe everything needed to configure an entire service or application. Multiple classes are combined to make a custom system role, such as 'web application worker'. The nodes are assigned classes based on their roles. This can be done using node definitions.

4. Pros and Cons

Pros:

1. ***Maturity and Stability***: Puppet has the most mature and stable interface of any configuration management tool. It has been around longer than most other tools and has hence been deployed and tested in various demanding environments. It also runs on all major operating systems.

2. ***Easy setup***: The steps for installation and initial setup is easy and supports various operating systems.

3. ***Complete Web UI***: Puppet has one of the most complete web UI with strong reporting capabilities and real-time node management

4. ***Support community***: There is a well-established and active support community through Puppet Labs.

Cons:

1. ***Puppet DSL***: Difficult for new users to learn Puppet DSL or Ruby. It is also difficult to understand existing code and implement advanced tasks which will require using Ruby-based CLI.

2. ***Complicated Code base***: Due to the DSL, the code base can grow large and became extremely complicated at a higher scale and hence makes this a bad solution to scale deployments.

3. ***Pull based***: Puppet uses a pull based model which means there will be no immediate action on changes since it follows a specified schedule for tasks.

5. Related Application/Products

Other popular configuration management tools used today as an alternative to Puppet are Ansible and Chef among others. Ansible and Chef can also be used to achieve many of the same things as Puppet but still differ from Puppet in certain ways.

1) **Ansible**
    a) ***SSH-based***: Ansible does not require installing agents on the remote nodes. It only requires installing ansible on master server which uses SSH to connect to the clients.

    b) ***Python-based***: Ansible is written in Python and allows users to script commands in YAML format which is a lot easier to learn and understand.

    c) ***Push model***: It uses a push model which means central server pushes configurations to client nodes.

2) **Chef**
    a) ***Additional Workstation***: Although Chef has a similar master-agent architecture as Puppet but it uses an additional workstation to control configurations from the master to agents.

    b) ***Code-driven approach***: Chef uses pure Ruby scripts to given a code-driven approach to configuration as opposed to the model-based approach of Puppet. This gives users more control and flexibility over their configurations.