Configuration Management: Chef & Puppet

Internet-scale Distributed Systems Seminar Report

Felix Wieser
TU Munich
Munich, Germany
felix.wieser@tum.de

Daniel Federschmidt
TU Munich
Munich, Germany
daniel.federschmidt@tum.de

ABSTRACT

A good report has an abstract!

KEYWORDS

Configuration Management, DevOps

1 INTRODUCTION

1.1 Infrastructure as Code

- Paradigm shift due to the complexity of modern web services
  - *Repeatability*
  - *Automation*
  - *Agility*
  - *Scalability*
  - *Reassurance*
  - *Disaster Recovery*

1.2 Why configuration management

- Deploy many applications on many machines (a.k.a. nodes)
- Update an application on all nodes simultaneously
- Simplify rollbacks
- Keep environments consistent among multiple entities (i.e. dev/testing/production)
- Keep records of all changes of the infrastructure

1.3 Properties of system configuration tools

- Input Specification
- Abstraction Mechanisms
- Modularization Mechanisms
- Relation Modeling

Input Specification

2 SYSTEM ARCHITECTURE

- Push config vs. Pull config

High-level overview.

2.1 Configuration Management

This is a citation [1].

2.1.1 Insight A. A subsubsection

2.1.2 Insight B. Another subsubsection.

2.2 Chef

Found on https://docs.chef.io/chef_overview.html

- Components
  - Chef DK (Chef Development Kit)
    - Computers running Chef DK are called Workstations
    - Creation of cookbooks
    - Test of cookbooks with Test Kitchen
      - Describe Test Kitchen here
    - Components of workstations
      - Knife
        - Interface between local chef-repo and Chef server
      - The chef-repo
        - Cookbook storage
        - "The chef-repo should be synchronized with a version control system (such as git), and then managed as if it were source code" https://docs.chef.io/workstation.html#configure-ruby-environment
      - knife.rb
        - File to specify configuration details for knife
  - Chef Server
    - Hub for configuration data (cookbooks)
    - Pull configuration: Nodes pull cookbooks from server
    - Features
      - Search any type of data that is indexed by the Chef server (features wildcards, etc.)
      - Management of
        - Nodes
        - Cookbooks and recipes
        - Roles
        - Stores of JSON data (data bags), including encrypted data
        - Environments
        - User accounts and user data
      - data bag
        - Global variable that is stored as JSON data
        - Accessible from Chef server
        - Indexed for searching
      - Policy
        - Role
          - Defines patterns and processes that exists across nodes
          - Chef client merges attributes and run-lists with assigned roles
      - Environment
        - Maps the real-life workflow to the configuration items of Chef
        - Can be associated with one or more cookbook versions
      - Run-list
        - Ordered list of roles and/or recipes
        - Items run in the order defined in the run-list

      · Can be node-specific
      · Stored as part of the node object on the Chef server
      · Maintenance with knife or Chef Automate
  - Chef client
    * Must be installed on each node
    * Performs
      · Registration and authentication of the node with the chef server
      · Building the node object
      · Synchronization of cookbooks
      · Compilation of the resource collection
      · Configuration of the node
      · Exception and notification handling
  - Ohai
    * Collects system configuration data for use within cookbooks
    * Includes many built-in plugins to detect state
    * Attributes contain: OS, Network, Memory, Disk, CPU, Kernel, host names, virtualization, etc.
  - Chef Supermarket
    * Sharing and management of community cookbooks
- Cookbooks contain
  - attributes
  - cookbook_file
  - libraries: Ruby code can be included in a cookbook
  - metadata: Stored in *metadata.rb*. Helps the server deploy the cookbooks to the nodes correctly
  - recipes
    * Authored in Ruby
    * Collection of ressources
    * Must define everything that is needed to configure the node
  - ressources
    * Describes the desired state for a configuration item
    * Describes the steps to achieve the desired state
    * Contains ressource type
    * Grouped into recipes
  - templates
    * Used to dynamically generate static text files
    * May contain Ruby
    * Intended to manage configuration files
  - tests

2.3 Puppet

- Uses a master-slave architecture
- Uses pull config
- Ressource management:
  - Manifests describe the node configuration
  - Groups of ressources can be organized into classes ⇒ i.e. config for entire application can be grouped
  - Modules combine manifests and data to improve code organization
- Server node connection via SSL works as follows:

(1) Node sends normalized data to the Puppet master
(2) Server uses this data to compile a catalog, that specifies how the node should be configured
(3) The node reports back the successful config to the master (Visible on the Puppet Dashboard)

2.4 Evaluation

Actual hard work happens here - many thoughts!

3 CONCLUSIONS

- Decision for or against a certain tool might not only be technical but depends on the structure of the organization implementing the architecture (Conways Law?)

REFERENCES

[1] Leslie Lamport. 1986. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, MA.