

Курс «DevOps. Системный инженер»

Занятие 11.

Configuration management. Ansible

Преподаватель

Пиотух Валерий Владимирович



1. Что такое Configuration Management, цели, инструменты.
2. Ansible. Описание, основные функции.
3. Демо: настройка виртуальных машин (Virtualbox) с помощью Ansible.

Управление конфигурацией — это процесс проектирования систем, позволяющий обеспечить единообразие атрибутов продукта на протяжении всего жизненного цикла. В технологическом мире управление конфигурацией представляет собой процесс управления ИТ, который используют для отслеживания отдельных элементов конфигурации ИТ-системы, таких как программное обеспечение, сервер или кластер серверов.

Управление конфигурацией программного обеспечения — это процесс системного проектирования, предназначенный для отслеживания изменений в конфигурационных метаданных программных систем.

Управление конфигурацией помогает техническим командам создавать стабильные и надежные системы с помощью инструментов, которые автоматически управляют обновлениями конфигурационных данных и отслеживают их. Сложные программные системы состоят из компонентов, различающихся по размеру и сложности. В качестве конкретного примера приведем архитектуру микрослужб. Каждая служба в такой архитектуре использует конфигурационные метаданные для собственной регистрации и инициализации. Вот несколько примеров конфигурационных метаданных программного обеспечения.

- Спецификации распределения ресурсов вычислительного оборудования для ЦП, оперативной памяти и т. д.
- Конечные точки, которые определяют внешние подключения к другим службам, базам данных или доменам.
- Секреты, такие как пароли и ключи шифрования.

С самого начала конфигурационные данные тяжело поддаются обработке и легко могут выпасть из поля зрения. Они не являются настоящим кодом и поэтому не сразу попадают в систему контроля версий. Их также нельзя отнести к настоящим данным, поэтому они не хранятся в первичной базе данных. Традиционное и маломасштабное системное администрирование обычно выполняется с помощью набора скриптов и специальных процессов. Конфигурационные данные нередко остаются без внимания, но они крайне важны для работы системы.

Рост количества облачных инфраструктур привел к разработке и внедрению новых моделей управления инфраструктурой. Развертывание сложных облачных системных архитектур и управление ими выполняется с помощью файлов с конфигурационными данными. Новые облачные платформы позволяют командам указывать необходимые аппаратные ресурсы и сетевые подключения с помощью файлов данных, таких как YAML. Затем эти файлы данных считываются, а настроенная инфраструктура становится доступна в облаке.

Инструменты configuration management

06



Terraform

'SALTSTACK



puppet



git



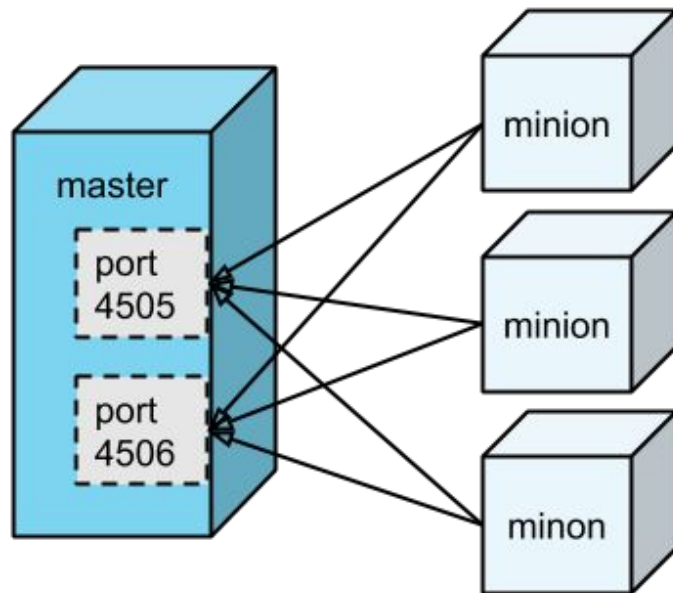
CHEF

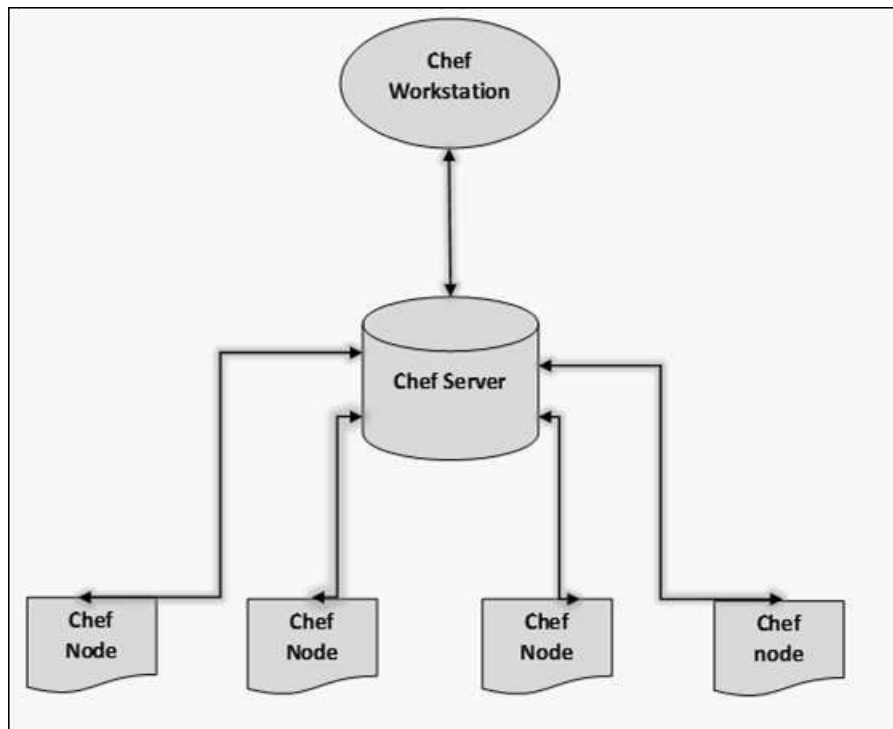


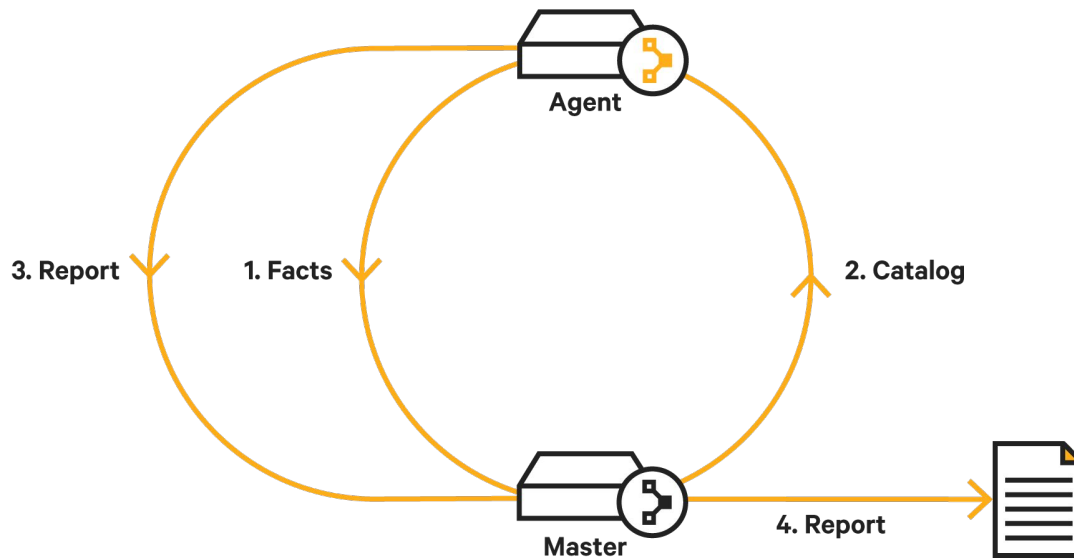
ANSIBLE



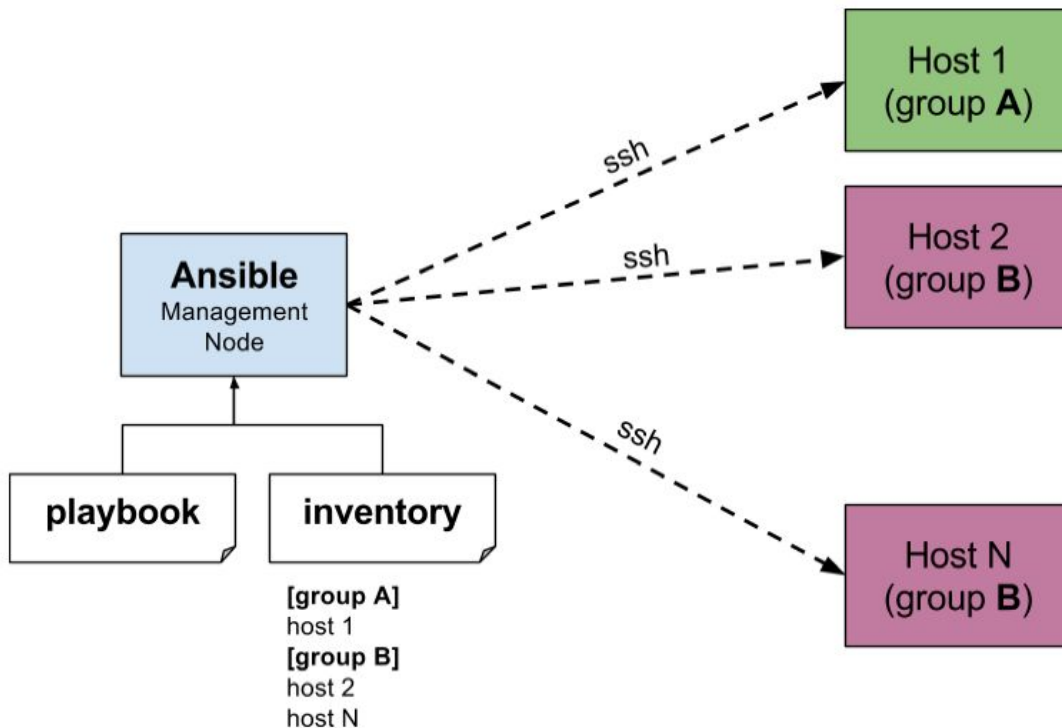
docker







Ansible



Ansible — это программное решение для удаленного управления конфигурациями. Оно позволяет настраивать удаленные машины. Главное его отличие от других подобных систем в том, что Ansible использует существующую инфраструктуру SSH, в то время как другие (chef, puppet, и пр.) требуют установки специального окружения.



A N S I B L E

Установка и настройка Ansible

```
$ sudo apt-add-repository ppa:ansible/ansible
```

```
$ sudo apt update && sudo apt install ansible -y
```

```
$ ansible --version
```

```
$ ssh-keygen
```

* для удобства можно поменять имена вл командой `sudo hostnamectl set-hostname *new-hostname*`

Далее нужно положить публичный ключ мастера на остальные машины в файл `~/.ssh/authorized_keys`

Ansible inventory

Inventory - файл, в котором содержится информация о хостах, с которыми работает Ansible. По умолчанию inventory файл находится по пути `/etc/ansible/hosts` либо задается с помощью параметра `-i`

Пример:

```
$ cat ansible/hosts
```

```
10.10.0.6
```

```
ansible-1 ansible_ssh_host=10.10.0.8 ansible_ssh_user=vagrant
```

Вместе с установкой Ansible устанавливается также большое количество модулей (библиотека модулей).

Модули отвечают за действия, которые выполняет Ansible. При этом каждый модуль, как правило, отвечает за свою конкретную и небольшую задачу.

Модули можно выполнять отдельно или собирать в определенный сценарий (play), а затем в playbook.

Примеры Ansible модулей

Ping

```
$ ansible -i ansible/inventory/hosts -m ping all
```

Shell

```
$ ansible -i ansible/inventory/hosts -m shell -a 'uname -a' all
```

Copy

```
$ ansible -i ansible/inventory/hosts -m copy -a  
'src=/home/vagrant/ansible/inventory/hosts dest=/tmp/' all
```

Setup (сборка фактов об узлах)

```
$ ansible -i ansible/inventory/hosts -m setup all  
*-a 'filter=ansible_memtotal_mb'
```


Группировка хостов

```
$ cat ansible/hosts
```

```
[db]
```

```
ansible-1 ansible_ssh_host=10.10.0.6 ansible_ssh_user=vagrant
```

```
[web]
```

```
ansible-2 ansible_ssh_host=10.10.0.8 ansible_ssh_user=vagrant
```

```
$ ansible -i ansible/hosts -m ping db
```

Ansible playbook

Playbook в Ansible - это способ отправки команд на удалённые компьютеры с помощью скриптов. Вместо того, чтобы индивидуально использовать команды для удалённой настройки компьютеров из командной строки, вы можете настраивать целые сложные среды, передавая скрипт одной или нескольким системам.

Ansible playbook

```
$ cat ansible/playbook.yml
```

```
---
```

```
- name: Setting up our VMs
  gather_facts: true
  hosts: all
```

```
tasks:
```

```
  - name: Create group "devops"
    ansible.builtin.group:
      name: devops
      state: present
      gid: 1400
      become: yes
```

```
  - name: Add user "belhard"
    ansible.builtin.user:
      name: belhard
      group: devops
      shell: /bin/bash
      comment: "Belhard DevOps user"
      state: present
      uid: 1500
      become: yes
```

Добавим sudo без пароля и свой локальный SSH ключ id_rsa.pub

- name: Add belhard user to the sudoers
copy:
 dest: "/etc/sudoers.d/belhard"
 content: |
 belhard ALL = NOPASSWD: ALL
 belhard ALL = (ALL) NOPASSWD:ALL
become: yes

- name: Add local SSH key
copy:
 dest: "/home/belhard/.ssh/authorized_keys"
 content:
 !!!сюда вставьте свой SSH public key с локальной машины!!!
become: yes

Установим java на хост slave-1

```
- name: install java
  apt:
    update_cache: yes
    pkg:
      - unzip
      - zip
      - openjdk-11-jdk
    autoclean: yes
    autoremove: yes
  when:
    - ansible_hostname == "slave-1"
  become: yes
```

Установим nginx на хост slave-2

- name: update
 apt: update_cache=yes
 become: yes

- name: Install Nginx
 apt: name=nginx state=latest
 when:
 - ansible_hostname == "slave-2"
 become: yes
 notify:
 - restart nginx

handlers:

- name: restart nginx
 service: name=nginx state=reloaded
 become: yes

Ansible playbook

Установим Docker отдельным playbook'ом

```
git clone https://github.com/do-community/ansible-playbooks
```

```
cd ansible-playbooks/docker_ubuntu1804
```

```
ansible-playbook -i ~/ansible/hosts playbook.yml
```

**Спасибо
за внимание!**

