Архитектура вычислительных систем

Управление инфраструктурой

Романюта Алексей Андреевич

alexey-r.98@yandex.ru

Кафедра вычислительных систем
Сибирский государственный университет телекоммуникаций и информатики

Управление конфигурацией

- Любой сервис, приложение имеет конфигурацию, например, конфигурацию системы для его запуска – systemd-unit, windows-service
- Конфигурации приложения, параметры запуска, размещения отличаются для каждого отдельно взятого приложения
- С ростом количества приложений и их разновидностей, усложняется процесс поддержки, внесения изменений и обновления.

Автоматизация управления конфигурациями

- Все можно описать кодом – подход *as code* (Infrastructure as Code – IaC)
- Коллективная разработка
- Сведение возможных ошибок в процессе к минимуму
- Пишется один раз для всех экземпляров одного типа приложения
- «Лень», Etc…

Свойства систем управления конфигурациями

- Все описано сценариями – playbook, recipe, runbook
- Идемпотентность – отсутствие изменений конфигурации не вызывает дополнительных действий

Ansible

- Open-source
- Agentless – не требует установки сервера управления и агента на клиентов
- Python/yaml based – основная часть написана на языке python, для описания сценариев используется yaml.
- Позволяет добавлять пользовательские модули (На любом языке программирования)
- Использует в качестве транспорта до клиентов протокол SSH

Сущности ansible

- Inventory – Описание хостов и их групп. Так же описывает параметры и значения, применимые к отдельным хостам или группе (host/group vars) Описывает «Где» запускать
- Playbook – Набор задач, ролей для применения. Описывает «Что» и «где» запускать
- Roles – атомарный набор сценариев, содержащий необходимые части для развертывания и управления приложением

Ansible конфигурация репозитория

```
[defaults]
display_skipped_hosts = False
; host_key_checking = False
roles_path = roles
library = library
interpreter_python = /usr/bin/python3
; log_path = ansible.log
ansible_managed = "Managed by Ansible"
remote_tmp = /tmp
inventory = inventory/csc
vault_password_file = .ansible_vault_pass
pipelining = True
callbacks_enabled = default, timer, profile_tasks

[ssh_connection]
ssh_args = -o ControlMaster=auto -o ControlPersist=60s -o
StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null
control_path = %(directory)s/%%h-%%p-%%r
```

```
inventory/
  csc/
  cscdef/
playbooks/
library/
roles/
.ansible_vault_pass
```

Ansible inventory

- Возможные форматы
  - INI
  - JSON
  - YAML

INI формат используется в подавляющем большинстве инструкций в Сети.
Однако, вместе с JSON принят сообществом как сложно читаемый и неудобный при ручном редактировании

inventory.ini:

```
mail.example.com

[webservers]
foo.example.com a=1
bar.example.com a=2
```

inventory.yml:

```
---
servers: # группа servers
  hosts:
    s1: # сервер s1
      ansible_host: 1.2.3.4
    s2:
      ansible_host: 1.2.3.5
      ansible_port: 2222
```

Ansible inventory

- Каждый хост может принадлежать нескольким группам хостов
  Можно поместить каждый хост более чем в одну группу. Например, рабочий веб-сервер в центре обработки данных A1 может быть включен в группы с именами *prod*, *a1* и *webservers*. Вы можете создавать группы, которые отслеживают:
  - Что — приложение, стек или микросервис (например, серверы баз данных, веб-серверы и т. д.).
  - Где — центр обработки данных или регион (например, восток, запад).
  - Окружения — Этап разработки, чтобы избежать тестирования на «продакшн» ресурсах (например, *prod*, *test*).

Ansible inventory: переменные

- Переменные так же могут быть в одном из форматов – ini, json, yaml
- Делятся на два типа: host и group
- Т.к. сервера могут быть объединены в группы – group vars устанавливает переменные для каждого хоста в группе
- Host vars устанавливает переменную для конкретных хостов
- Host vars имеет приоритет выше group vars

Ansible: переменные

- Переменные могут быть определены в нескольких местах
  - Inventory group/host
  - В defaults/vars роли
  - При запуске сценария как extra-vars
  - В playbook

Если одна переменная определена в нескольких местах – как определяется её итоговое значение?

Ansible: переменные

1. command line values (for example, -u my_user, these are not variables)
2. role defaults (defined in role/defaults/main.yml) 1
3. inventory file or script group vars
4. inventory group_vars/all
5. playbook group_vars/all
6. inventory group_vars/*
7. playbook group_vars/*
8. inventory file or script host vars
9. inventory host_vars/*
10. playbook host_vars/*
11. host facts / cached set_facts
12. play vars
13. play vars_prompt
14. play vars_files
15. role vars (defined in role/vars/main.yml)
16. block vars (only for tasks in block)
17. task vars (only for the task)
18. include_vars
19. set_facts / registered vars
20. role (and include_role) params
21. include params
22. extra vars (for example, -e "user=my_user")(always win precedence)

Если одна переменная определена в нескольких местах – как определяется её итоговое значение? - *приоритет переменных*[1]
Списки/Словари? – полное переопределение в соответствии с приоритетом

[1] Приоритет переменных Ansible // https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#understanding-variable-precedence

Ansible playbook

- Задача playbook – определить «что» запускать и «где»

freeipa-server.yml:

```
---
- name: Converge freeipa # Имя этапа
  become: true # Необходимо ли повышение прав
  serial: 1 # Запуск только для 1 хоста в одно время
  hosts: # «Где» запускать
    - ipaserver
  roles: # «Что» запустить
    - role: docker_install # Вызов роли docker_install
      tags: docker
    - role: freeipa_server # Вызов роли freeipa_server
      tags:
        - "freeipa"
  post_tasks: # «Что» запустить после основных этапов
    - name: Wait for port 443 to become open
      ansible.builtin.wait_for:
        host: "{{ freeipa_server_ip }}"
        port: 443
        delay: 5
        timeout: 300
      tags: ["freeipa"]
```

Ansible roles

- Роль – атомарная сущность, задача которой описывать сценарии выполнения для конкретного приложения/сервиса
- Ansible-galaxy – хранилище ролей или коллекций ansible от сообщества

Каждая роль содержит:
defaults – набор переменных и их значений по умолчанию. Имеют самый низкий приоритет

files – набор файлов, необходимых для приложения

handlers – набор сценариев, запускаемых «по требованию»

meta – информация о роли, спецификация аргументов

tasks – сценарии выполнения для данной роли

templates – набор шаблонов jinja2 для генерации конфигурационных файлов

tests – сценарии тестирования ролей

vars – переменные, которые могут зависеть от целевых систем или других факторов

Ansible: основные модули

Пример встроенных модулей

- ansible.builtin.copy
- ansible.builtin.template
- ansible.builtin.apt, ansible.builtin.dnf, ansible.builtin.pkg
- ansible.builtin.meta
- ansible.builtin.lineinfile
- ansible.builtin.blockinfile
- ansible.builtin.systemd

Пример модуля сторонних коллекций

- community.docker.docker_compose_v2

Ansible роль: пример

- Для примера, напишем роль установки atop
- Приложение ставится из apt репозиториев
- Частичное управление конфигурацией
- Параметризируем параметры:
  - LOGINTERVAL
  - LOGGENERATIONS

/etc/default/atop:

```
# /etc/default/atop
# see man atoprc for more
possibilities to configure atop
execution

LOGOPTS="-R"
LOGINTERVAL=600
LOGGENERATIONS=28
LOGPATH=/var/log/atop
```

Сгенерировать основные файлы роли (важно распложить их в директории, определенной roles_path)

```
ansible-galaxy role init atop
```

Ansible роль: пример

defaults/main.yml

```
---
# defaults file atop_install
atop_install_log_interval: 10
atop_install_log_generations: 10
```

handlers/main.yml

```
---
# handlers file for
atop_install
- name: "Restart atop"
  ansible.builtin.systemd:
    daemon_reload: true
    enabled: true
    state: "restarted"
    name: "atop"
```

tasks/main.yml

```
---
- name: "[Debian] Install atop package"
  ansible.builtin.apt:
    name: atop
    state: present
    update_cache: true

- name: "Configure atop"
  ansible.builtin.blockinfile:
    path: /etc/default/atop
    block: |
      LOGINTERVAL={{ atop_install_log_interval }}
      INTERVAL={{ atop_install_log_interval }}
      LOGGENERATIONS={{ atop_install_log_generations }}
    marker: "# {mark} ANSIBLE MANAGED BLOCK for atop role"
    create: true
    mode: "0644"
  notify: Restart atop
```

Ansible: генерация файлов конфигурации

```
- name: "Configure atop"
  ansible.builtin.blockinfile:
    path: /etc/default/atop
    block: |
      LOGINTERVAL={{ atop_install_log_interval }}
      INTERVAL={{ atop_install_log_interval }}
      LOGGENERATIONS={{ atop_install_log_generations }}
    marker: "# {mark} ANSIBLE MANAGED BLOCK for atop-install role"
    create: true
    mode: "0644"
  notify: Restart atop
```

```
- name: "Generate config file"
  ansible.builtin.template:
    src: "templates/auditd.conf.j2"
    dest: "/etc/audit/auditd.conf"
    mode: "0640"
  notify: Restart auditd
```

```
- name: "Generate immutable flag file"
  ansible.builtin.copy:
    content: |
      # Managed by Ansible
      -e 2
    dest: "/etc/audit/rules.d/99-immutable.conf"
    mode: "0640"
```

Ansible: генерация файлов конфигурации

```
- name: "Add audit=1 to grub cmdline"
  ansible.builtin.lineinfile:
    state: present
    dest: /etc/default/grub
    backrefs: true
    regexp: '^(GRUB_CMDLINE_LINUX=(?!.* audit)\"[^\"]+)(\".*)'
    line: '\1 audit=1\2'
  notify: Update grub
```

```
- name: "Generate immutable flag file"
  ansible.builtin.copy:
    src: "99-immutable.conf"
    dest: "/etc/audit/rules.d/99-immutable.conf"
    mode: "0640"
```

Ansible: шаблоны

- Используются, когда файл конфигурации делается полностью IaC-managed
- Используется шаблонизатор jinja2

```
- name: "Generate config file"
  ansible.builtin.template:
    src: "templates/auditd.conf.j2"
    dest: "/etc/audit/auditd.conf"
    mode: "0640"
  notify: Restart auditd
```

```
auditd_config: {}
auditd_config_default:
  local_events: "yes"
  write_logs: "yes"
  log_file: /var/log/audit/audit.log
  log_group: adm
```

templates/auditd.conf.j2

```
#jinja2: lstrip_blocks: True
{{ ansible_managed | comment('plain') }}
{% set config = auditd_config_default |
      combine(auditd_config, recursive=True, list_merge='append') %}

{% for k,v in config.items() %}
{{ k }} = {{ v }}
{% endfor %}
```

Позволяет определить default-конфиг в роли и переопределить его в переменных выше по приоритету. Combine – функция объединения map

Ansible: шаблоны

```
#jinja2: lstrip_blocks: True
{{ ansible_managed | comment('plain') }}
global:
  scrape_interval: {{ prometheus_scrape_interval }}
  evaluation_interval: {{ prometheus_evaluation_interval }}
  external_labels:
    {{ prometheus_external_labels | to_nice_yaml(indent=2) | indent(4) }}

rule_files:
  - "/etc/prometheus/prometheus-rules.yml"

alerting:
  alertmanagers:
    {{ prometheus_alertmanager_config | to_nice_yaml(indent=2) | indent(4) }}

scrape_configs:
{% for job in prometheus_scrape_configs %}
  - job_name: {{ job.name }}
    scrape_interval: {{ job.scrape_interval | default("10s") }}
    scheme: {{ job.scheme | default("http") }}
    metrics_path: {{ job.metrics_path | default("/metrics") }}
    {% if job.params is defined %}
    params:
      {{ job.params | to_nice_yaml(indent=2) | indent(6) }}
    {% endif %}
    {% if job.relabel_configs | default([]) | length > 0 %}
    relabel_configs:
      {{ job.relabel_configs | to_nice_yaml(indent=2) | indent(6) }}
    {% endif %}
    {{ job.configs | to_nice_yaml(indent=2) | indent(4) }}
{% endfor %}
```

Ansible: vault

- Одна из практик разработки – не добавлять в изменения и коммиты секретных данных – токены, пароли и тд в открытом виде
- Можно добавлять в зашифрованном, но пароль так же держать отдельно
- Ansible имеет подсистему vault
- Best practice – размещать секреты в отдельный файл переменных (в inventory и тп) и шифровать весь файл. Файлу дать суффикс ‘–vault’ (vars-vault.yml)

```
ansible.cfg
[defaults]
vault_password_file = .ansible_vault_pass
```

```
.gitignore
.ansible_vault_pass
```

```
inventory/
  csc/
  cscdef/
playbooks/
library/
roles/
.ansible_vault_pass
```

Ansible: vault

inventory/host_vars/myhost/vars.yml

```
---
my_secret_var: "{{ vault_my_secret_var }}"
```

inventory/host_vars/myhost/vars-vault.yml

```
---
vault_my_secret_var: "secretvalue"
```

```
ansible-vault encrypt inventory/host_vars/myhost/vars-vault.yml
```

inventory/host_vars/myhost/vars-vault.yml

```
$ANSIBLE_VAULT;1.1;AES256
653866336538356431663161306638613530333166613965306537356232636461616239346164 33…
```

Live section

Романюта Алексей Андреевич

alexey-r.98@yandex.ru

Кафедра вычислительных систем
Сибирский государственный университет телекоммуникаций и информатики