Управление конфигурацией. Основные DevOps инструменты.

Знакомство с Ansible.

План

- Инфраструктура как код и управление конфигурациями
- Система управления конфигурациями Ansible
- Основные элементы для работы с Ansible

IaC

- Настройка облачной инфраструктуры
- Настройка self-hosted инфраструктуры
- Настройка локального окружения
- Деплой

Configuration Management

- общий фреймворк
- повторное использование кода
- версионирование
- совместная работа
- base-app-service модель
- повторяемость (контроль за изменением состояния)

CM tools

- Ansible
- Chef
- Puppet
- Salt

Ansible

- версия 2.4 (на самом деле 2.3.2)

Ansible Changes By Release

2.4 "Dancing Days" - ACTIVE DEVELOPMENT

- готовые модули (> 1000)
- готовые роли на Ansible Galaxy (> 11000 ролей)
- быстрый старт
- отсутствие агента и минимальные требования к хостам
- идемпотентность

и еще

- декларативный язык (YAML)
- поддержка Windows
- управление внешними и внутренними инфраструктурными сервисами

Экосистема

- Простая установка
- Тестирование
- Поддержка RedHat
- Активное сообщество

Входной порог

Для рабочей машины нужны

- ОС linux/unix
- Python 2.7+
- SSH

Инфраструктурный репозиторий

Храним и версионируем

- Библиотеки и саму версию Ansible (requirements.txt)
- Файл конфигурации для управляемой машины
- Сами компоненты проекта, описанные в виде кода
- Описание и пример для запуска и отладки на локальном(не всегда) окружении разработчика
- Описание окружений

Пример инфраструктурного репозитория

```
ansible.cfg
mytasks.yml
requirements.txt
```

Более продвинутый вариант, используемый в нашей компании
https://github.com/express42/ansible-repertory

Инфраструктурные репо в компаниях могут отличаться

Ansible config file

- Управление плагинами
- Переопределяемые параметры и указатели
  - Параметры по умолчанию
  - Inventory
  - Способ подключения

Всегда последний и актуальный файл конфигурации находится в официальном репо

Inventory

- Группировка и разделение хостов
- Вложенные группы
- Inventory файлов может быть несколько
  - Inventory файл может быть динамичным и формироваться на основе каких-то внешних данных
- Предоставляет возможность формировать алиасы нашим хостам
- Предоставляет возможность переопределять параметры подключения, описанные в ansible.cfg

Inventory (ini формат)

Примеры группировки хостов

```
[reddit_app]
app01.myredditclone.internal

[reddit_front]
web01.myredditclone.internal
web02.myredditclone.internal

[vote_sharding_db]
mongo_db[01:20].myredditclone.internal

[eu1_region:children]
reddit_front
reddit_app
vote_sharding_db
```

В квадратных скобках формируем названия групп

Хосты, принадлежащие нашей группе

Хосты можно задавать по паттернам, и да, они могут повторяться

Группы из групп

Язык YAML

- Проще читать и редактировать
- Отступы для уровней вложенности
- Массивы и словари (hash, dictionary)
- Поддержка примитивов

Пример

```
---
- name: Create instance(s)
  hosts: localhost
  connection: local
  gather_facts: no

  tasks:
    - name: Launch instances
      gce:
        instance_names: created-from-ansible
        machine_type: g1-small
        image: ubuntu-1604-xenial-v20170815a
        service_account_email: 1015202452876-compute@developer.gserviceaccount.com
        credentials_file: gcp.json
        project_id: steady-tracer-178304
...
```

Пример

```
---
- name: Create instance(s)
  hosts: localhost
  connection: local
  gather_facts: no

  tasks:
    - name: Launch instances
      gce:
       instance_names: created-from-ansible
       machine_type: g1-small
       image: ubuntu-1604-xenial-v20170815a
       service_account_email: 1015202452876-compute@developer.gserviceaccount.com
       credentials_file: gcp.json
       project_id: steady-tracer-178304
...
```

Опциональные обозначения начала и конца YAML файла

Пример

```
---
- name: Create instance(s)
  hosts: localhost
  connection: local
  gather_facts: no

  tasks:
    - name: Launch instances
      gce:
        instance_names: created-from-ansible
        machine_type: g1-small
        image: ubuntu-1604-xenial-v20170815a
        service_account_email: 1015202452876-compute@developer.gserviceaccount.com
        credentials_file: gcp.json
        project_id: steady-tracer-178304
...
```

Каждый уровень вложенности отделяется двумя пробелами

Modules

- Небольшие библиотеки для выполнения и отслеживания состояния задач
  - Операции ОС
  - Команды по управлению программной частью физических устройств
  - Управление ресурсами внешних сервисов
- Основа для выполнения действий в Ansible
- Список категорий и всех модулей по ним

Modules

```
apt:
    name: git
    state: present
    update_cache: yes
```

Готовый модуль для работы с apt пакетами

Параметры модуля

Playbooks

- Сценарии для достижения целевого состояния системы с использованием модулей Ansible

- Используются для
  - описания процесса установки и настройки ПО
  - деплой
  - управления внешними сервисами

Playbooks

- Один сценарий называется play
  - В одном playbook может использоваться несколько play сценариев с различными условиями запуска

Один play сценарий

reddit_app.yml

```
---
- hosts: reddit_app
  remote_user: root
  tasks:
  - name: Clone project repo
    git:
     repo: 'https://github.com/Artemmkin/reddit'
     dest: /srv/reddit-app
```

Несколько play сценариев в одном файле

reddit_app.yml

```
---
- hosts: reddit_app
  remote_user: root
  tasks:
  - name: Clone project repo
    git:
    repo: 'https://github.com/Artemmkin/reddit'
    dest: /srv/reddit-app

- hosts: postgres_cluster_db
  remote_user: root
  tasks:
  - name: Create app db
    postgresql_db:
      name: reddit_main
      encoding: UTF-8
```

Условия выполнения плейбуков

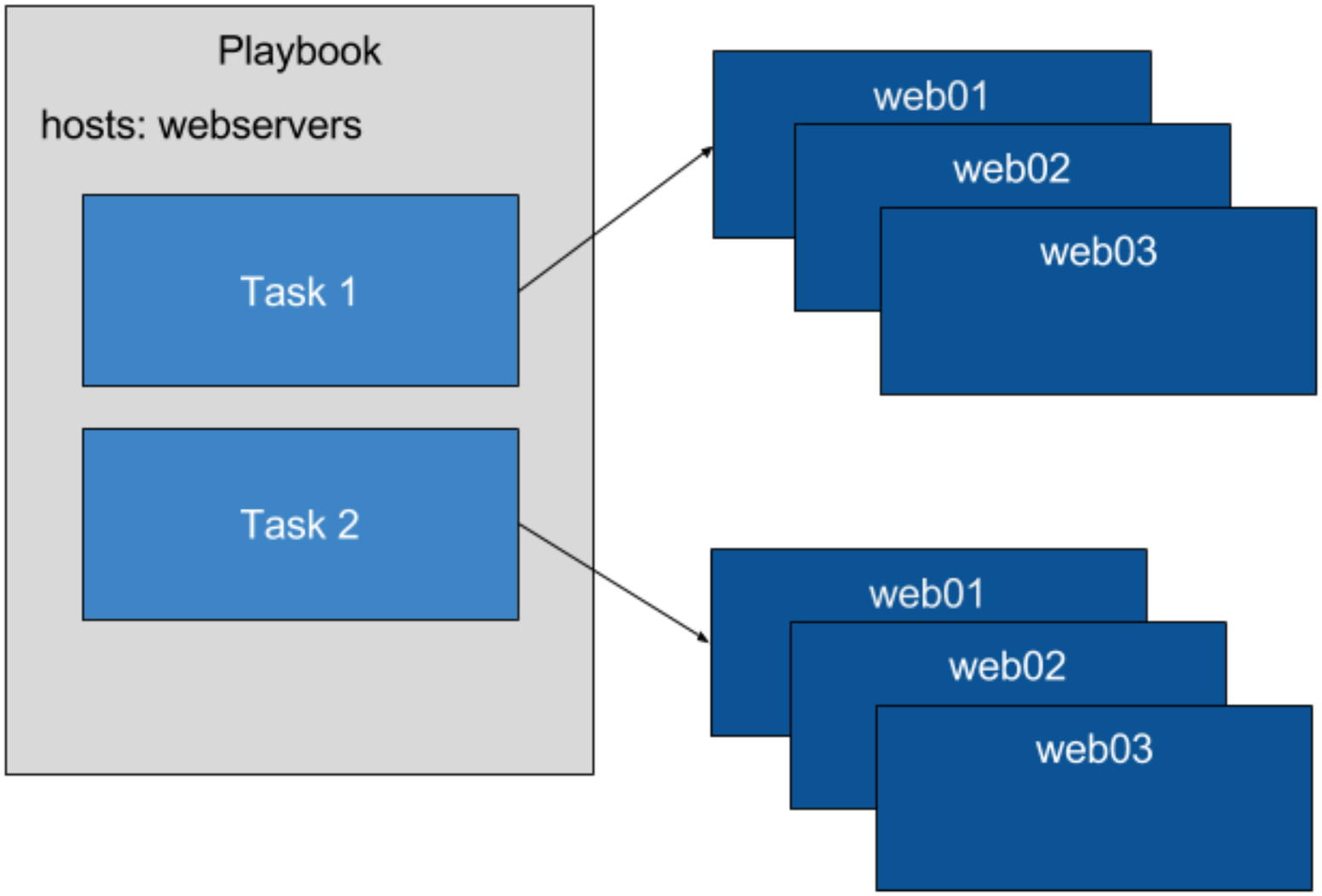
Условия выполнения

- `hosts` - хосты по условиям или группы где исполнять наши play сценарии
- `remote_user` - имя пользователя для подключения к целевым хостам
- Используется как для всего play сценария, так и для конкретных задач в нем
- Поддерживает эскалацию привилегий, команды можно выполнять от имени других пользователей

Условия выполнения play рецептов

- Несколько Play сценариев выполняются в том порядке, в котором они описаны
- Play сценарии могут включать в себе несколько задач. Задачи выполняются последовательно, по каждой на всех хостах, попадающих под host условия

Условия выполнения play рецептов



Переиспользование плейбуков

Начиная с 2.4 добавлена возможность использовать include и import для задач, для чего?

Используя простые модели описания сценариев, можно разбивать плейбуки на несколько файлов с задачами и переиспользовать их. Это избавляет от переиспользования скриптов сценария или хранения нагроможденных Playbook

Variables

- Нужны для переиспользования в различных частях описания сценариев и определения отличий

- Дополняют циклы и операторы с условиями, для определения отличий на основе переменных

- Могут использоваться почти везде, в пределах инфраструктурного репозитория

Variables

- YAML поддерживают словари, допускается использовать для переменных (k:v значения)

  Пример использования переменной в playbook

```
- hosts: reddit_app
  vars:
      active: yes
      app_path: {{ base_dir }}/reddit_app
```

Registered variables

В некоторых случаях, можно создавать переменную на основе выполненной команды или работы модуля

Эти переменные будут доступны в пределах исполняемого хоста и **могут отличаться между хостами**.

Facts

В Ansible помимо явно определенных переменных разработчиком, существуют read only системные переменные, также известны как факты. За это отвечает модуль setup.

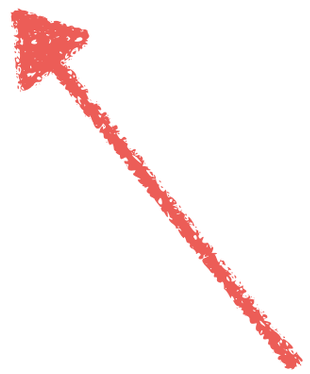
Посмотреть все факты которые может собрать setup модуль из системы и предоставлять их в виде переменных можно командой

```
$ ansible -m setup
```

Пример

```
- include: install.deb.yml

  when: ansible_os_family == 'Debian'
```



Основные антипаттерны

- Усердное использование shell модулей
  - Это выручает и нужно относиться к этому как к workaround, всегда старайтесь переходить на модули

Основные антипаттерны

Плохой пример 1

```
- name: Seed data
  shell: somescript.sh
  args:
    chdir: /opt/myapp/somedata
    creates: /opt/myapp/somedata.lock
```

Основные антипаттерны

Плохой пример 2

```
ansible  tag_db_role_master -m command -a "sudo -
u postgres psql -c ‘create database myapp;’"
```

Продолжение следует