Управление конфигурацией. Основные DevOps инструменты. Знакомство с Ansible

Не забудь включить запись!

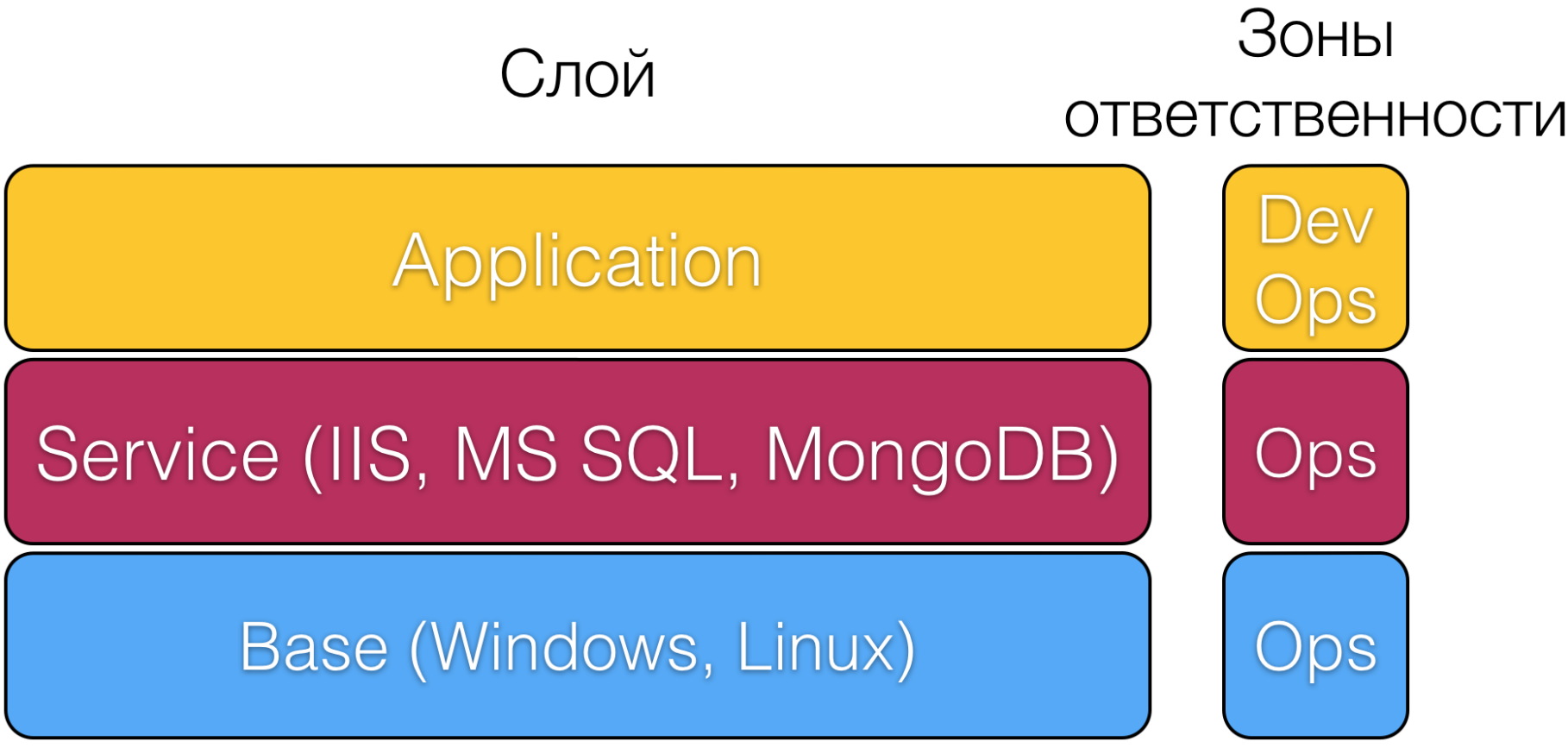
План

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

Configuration Management

- Общий фреймворк
- Повторное использование кода
- Версионирование
- Совместная работа
- Самодокументирование
- Base-Service-App модель
- Повторяемость (контроль за изменением состояния)

Base-service-app модель



IaC (Infrastructure as a Code)

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

CM tools

- Ansible
- Chef
- Puppet
- SaltStack

CM tools | Сравнение

	Chef	Puppet	Ansible	SaltStack
Code	Open-source	Open-source	Open-source	Open-source
Cloud	All	All	All	All
Infrastructure	Mutable	Mutable	Mutable	Mutable
Language	Procedural	Declarative	Procedural	Declarative
Architecture	Client/Server	Client/Server	Client-only	Client/Server

Источник: Gruntwork

Итак, Ansible!

Ansible | Основные особенности

- Текущая версия: 2.7.10 (не какая-нибудь 0.12alpha)
- Готовые модули (> 1000)
- Готовые роли на Ansible Galaxy (> 11000 ролей)
- Быстрый старт
- Отсутствие агента и минимальные требования к хостам
- Идемпотентность

> *Идемпотентность* - свойство объекта или операции при повторном применении операции к объекту давать тот же результат, что и при первом применении

Ansible | И еще это...

- Декларативный язык (YAML)
- Поддержка ОС Windows, помимо *nix
- Управление внешними и внутренними инфраструктурными сервисами (облака, системы виртуализации, СХД)
- Особое внимание уделяется управлению сетевым оборудованием (Cisco, Arista, Juniper, F5 и другие)

Язык YAML

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

Ansible | Экосистема

- Простая установка
- Тестирование модулей и ролей
- Поддержка IBM/RedHat
- Активное сообщество

Ansible | Входной порог

Для начала работы нам нужны:

- ОС Linux/Unix/MacOS
- Python 2.7+ (или Python 3)
- SSH

> В Windows 10 неплохо работает в WSL режиме

Ansible | Что плохого?

- Нельзя определять зависимости между задачами (только handlers или "черная магия")
- Не хранит состояние между запусками сценариев
- Сетевые задержки (RTT) **очень** влияют на время выполнения сценариев
- Нет реагирования на события - соответственно нет автоматического применения сценариев, самовосстановления и т.п.
- Когда узлов много (несколько сотен) - надо вкладываться в оптимизацию Playbook или переходить на AWX

> Статья оптимизацию скорости исполнения Ansible

Инфраструктурный репозиторий

В репозитории храним и версионируем:

- Библиотеки и саму версию Ansible (`requirements.txt`)
- Файл конфигурации для управляющей 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

- `./infra_repo/`
  - `ansible.cfg` - конфигурационный файл
  - `mytasks.yml` - Ansible плейбук
  - `requirements.txt` - файл зависимостей для `pip`

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

> Инфраструктурные репо в компаниях могут отличаться

Ansible config file (ansible.cfg)

В файле конфигурации содержатся следующие настройки:

- Управление плагинами
- Переопределяемые параметры и указатели
- Параметры по умолчанию
- Расположение файла inventory или скрипта dynamic inventory
- Способ подключения

Последний и актуальный файл конфигурации находится в официальном репозитории Ansible

Inventory

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

Документация по Ansible Inventory

Inventory (ini-формат)

Примеры группировки хостов:

```
[reddit_app]                              # <- Названия секций -
app01.myredditclone.internal              #    это названия групп

[reddit_front]
web01.myredditclone.internal              # <- Хосты принадлежащие группе
web02.myredditclone.internal

[vote_sharding_db]
mongo_db[01:20].myredditclone.internal    # <- Хосты можно задавать паттернами
                                          #    (и да, они могут повторяться)

[eu1_region:children]                     # <- Группы могут состоять из групп
reddit_front
reddit_app
vote_sharding_db
```

Inventory (YAML-формат)

Также Inventory можно создавать в формате YAML (или JSON с аналогичной структурой документа):

```
test:                       # <- Имя группы
  hosts:
    mail.example.com        # <- Хост в группе
  children:
    webservers:             # <- Подгруппы
      hosts:
        foo.example.com:    # <- Хосты в подгруппе
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
```

Пример создания инстанса GCE с помощью Ansible

```yaml
---                                  # <- Опциональное обозначение начала файла
- name: Create instance(s)
  hosts: localhost
  connection: local
  gather_facts: no

  tasks:
  - name: Launch instances
    # Уровни вложенности задаются двумя пробелами
    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
...                                  # <- Опциональное обозначение конца файла
```

Ansible 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 на сайте Ansible

Ansible Modules

Готовый модуль для работы с пакетным менеджером APT:

```
# Данный кусочек кода устанавливает пакет Git с помощью APT, выполнив apt update перед этим
apt:
  name: git
  state: present
  update_cache: yes
```

Ссылка на документацию модуля

Playbooks

Playbooks - это сценарии для достижения целевого состояния системы с использованием модулей Ansible

Используются для:

- Описания процесса установки и настройки ПО
- Развертывания приложений на группах хостов и настройки Base Layer
- Управления внешними сервисами

Playbooks

- Один сценарий называется *play*
- В одном *playbook* может использоваться несколько *play* сценариев с различными условиями запуска (группы хостов, стратегия выполнения, набор стартовых переменных)

Один play-сценарий

reddit_app.yml

```
- hosts: reddit_app
  remote_user: root
  tasks:
  - name: Clone project repo
    git:
      repo: 'https://github.com/express42/reddit.git'
      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/express42/reddit.git'
      dest: /srv/reddit-app

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

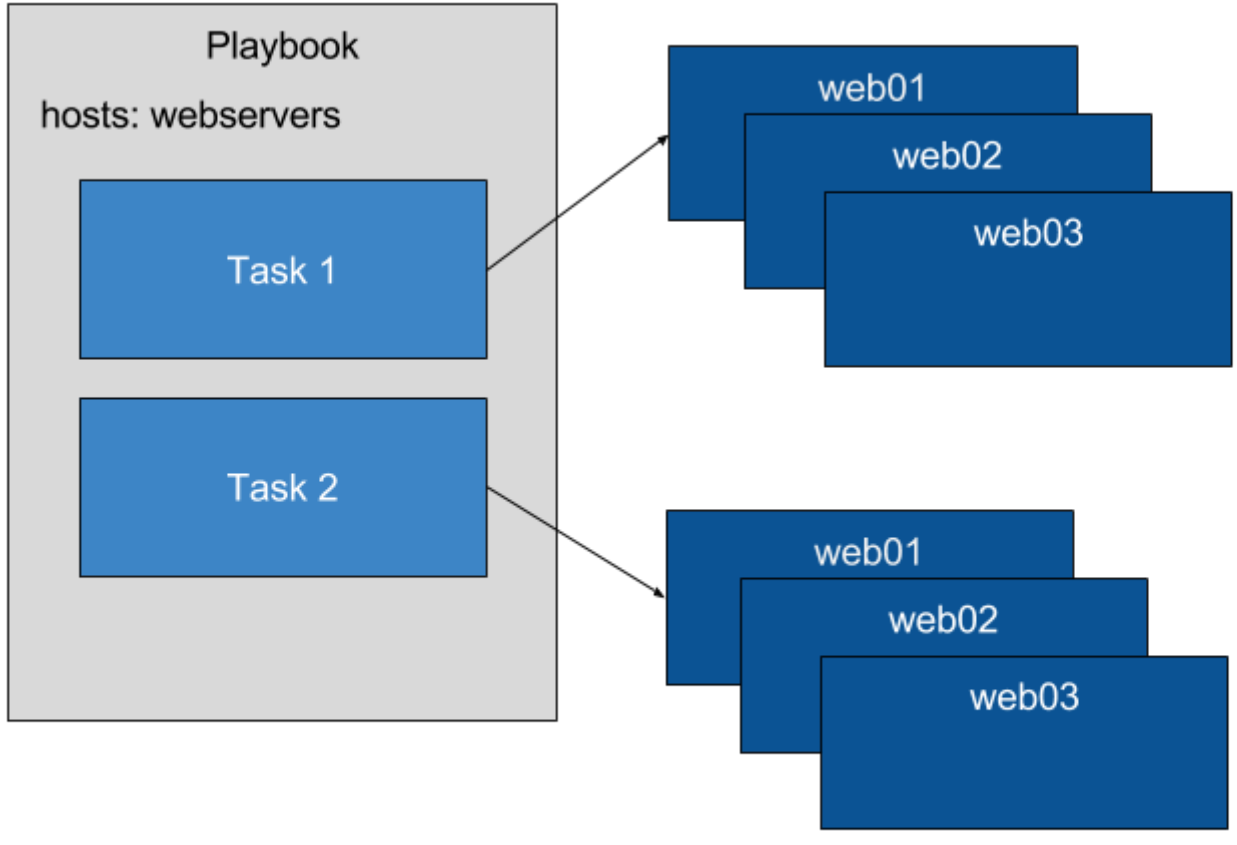
Условия выполнения play-сценариев

- `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-сценариев

- Несколько Play сценариев выполняются в том порядке, в котором они описаны
- Play сценарии могут включать в себе несколько задач.
- Задачи выполняются последовательно, по одной на всех хостах, попадающих под `host` условия (это стратегия исполнения по умолчанию, но не единственная)

Условия выполнения play-сценариев



Переиспользование плейбуков

Начиная с 2.4 добавлена возможность использовать `include_*` и `import_*` для задач и плейбуков. До этого была доступна только директива `include`.

Используя простые модели описания сценариев, можно разбивать плейбуки на несколько файлов с задачами и переиспользовать их. Это избавляет от дублирования кода сценариев и "разрастания" playbook.

> Документация по include и import

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 и списки. Их можно использовать для переменных (*key:value* параметры)

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 разработчиком переменных, существуют read-only системные переменные, которые также известны как *факты*. За это отвечает модуль `setup`.

Посмотреть все факты, которые может собрать `setup` модуль из системы можно командой:

```
$ ansible -m setup
```

Facts

Пример использования:

```
- include: install.deb.yml
  when: ansible_os_family == 'Debian'        # <- Переменная из фактов

- name: Install Systemd files
  include: inst_systemd.yml
  when: ansible_service_mgr == 'systemd'     # <- Переменная из фактов
```

Magic variables

Особые переменные

- Их имена зарезервированны
- Наиболее распространенные переменные hostvars, groups, group_names и inventory_hostname

Документация по использованию переменных

Список переменных и краткое описание

Hostvars

Hostvars - особая переменная, позволяющая при управлении одним хостом получить доступ к значениям переменных другого хоста

Пример:

```
{{ hostvars['host1']['admin_user'] }}
{{ hostvars['host1']['ansible_facts']['os_family'] }}
```

Основные антипаттерны

- Усердное использование `shell` и `command` модулей:
  - Это выручает, но нужно относиться к этому как к **workaround**
  - Всегда старайтесь переходить на вызовы модулей
- Несоблюдение четкой структуры инфраструктурного репозитория:
  - Это усложняет понимание, замедляет развитие и усложняет исправление проблем
  - Коллегам сложнее подключиться в работу на кодом
- Вызов модулей из консоли вместо использования playbook

Основные антипаттерны

Плохой пример 1 (частое использование shell, "баш скрипты на ansible"):

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

- name: Create dir
  shell: mkdir /opt

- name: Create file
  shell: touch /opt/test
```

Основные антипаттерны

Плохой пример 2 (Использование вызовов модулей из консоли вместо плейбуков):

```
$ ansible  tag_db_role_master -m command -a \
    "sudo -u postgres psql -c ‘create database myapp;’"
$ ansible  all -m command -a "mkdir /opt"
$ ansible  all -m command -a "touch /opt/test"
```

Основные антипаттерны

Плохой пример 3 (Несоблюдение четкой структуры репозитория):

```
$ ls ./my-infra-repo
env0            env3            env7            inventory1      inventory4
inventory8      playbook10.yml  playbook5.yml   playbook9.yml   test2.yml
test6.yml       utils0          utils3          utils7          env1
env4            env8            inventory10     inventory5      inventory9
playbook2.yml   playbook6.yml   test0.yml       test3.yml       test7.yml
utils1          utils4          utils8          env10           env5
env9            inventory2      inventory6      playbook0.yml   playbook3.yml
playbook7.yml   test1.yml       test4.yml       test8.yml       utils10
utils5          utils9          env2            env6            inventory0
inventory3      inventory7      playbook1.yml   playbook4.yml   playbook8.yml
test10.yml      test5.yml       test9.yml       utils2          utils6
```

Полезные ссылки

- Официальные best practice Ansible
- Что нового в Ansible 2.7
- Список модулей
- Ansible for DevOps - книга от Jeff Geerling
- Книга на русском Ansible: Up and Running
- Примеры образцов инфра репозитория: 1 и 2
- Канал в телеграме про Ansible