ANSIBLE

# Infrastructure as Code

## на примере Ansible

**Tinkoff**.ru

Артем Шепелев

# О чем пойдет речь?

1. Infrastructure as a Code
2. Configuration Management Software
3. Ansible Inventory
4. Ansible Ad-Hoc
5. Ansible Playbooks
6. Ansible Roles
7. Домашнее задание
8. Ссылки

1

# Infrastructure as a Code

# Infrastructure as Code (IaC) - что это?

- Это когда обычных bash-скриптов уже не хватает.
- Описание компьютерной инфраструктуры в виде кода.
- Унификация сценариев конфигурации в независимости от окружения.

# Преимущества

- Автоматизация ручных операций. Экономия времени.
- Высокая скорость выполнения операций.
- Снижение вероятности ошибок, управление чувствительными данными.

# Какие практические IaC решает задачи?

- Разворачивание стандартного окружения на хосте.
- Установка различного софта.
- Деплой приложений.
- Поддержание актуального состояния окружения.


# Configuration Management Software

# Configuration Management Software

- Ansible
- Chef
- Puppet
- Saltstack

# Почему тогда Ansible?

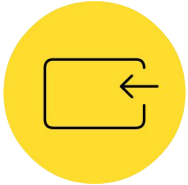
- Простота.
- Отличная документация, большое количество готовых проетков-ролей.
- Архитектура отлично подходит для обмена ролями.


# Ansible Inventory

# Основные сущности Ansible - inventory

- Static (текстовые .ini файлы)

- Dynamic (внешние сервисы)



# Static inventory

- Самый простой вариант

```
ny-prod-db01.company.com
ny-prod-db02.company.com

ny-prod-app01.company.com
ny-prod-app02.company.com
```

- Давайте усложним, введем группировку

```
[databases]
ny-prod-db01.company.com
ny-prod-db02.company.com

[applications]
ny-prod-app01.company.com
ny-prod-app02.company.com
```

# Static inventory

## Ещё чуть-чуть усложним

```
[ny:children]
ny-databases
ny-applications

[msk:children]
msk-databases
msk-applications

[applications:children]
ny-applications
msk-applications

[databases:children]
ny-databases
msk-databases
```

```
[ny-databases]
ny-prod-db01.company.com
ny-prod-db02.company.com

[ny-applications]
ny-prod-app01.company.com
ny-prod-app02.company.com

[msk-databases]
msk-prod-db01.company.com

[msk-applications]
msk-prod-app01.company.com
```

# Static inventory – различные окружения

## production.ini

```
[ny-databases]
ny-prod-db01.company.com
ny-prod-db02.company.com

[ny-applications]
ny-prod-app01.company.com
ny-prod-app02.company.com

[msk-databases]
msk-prod-db01.company.com

[msk-applications]
msk-prod-app01.company.com
```

## development.ini

```
[ny-databases]
ny-dev-db01.company.com
ny-dev-db02.company.com

[ny-applications]
ny-dev-app01.company.com
ny-dev-app02.company.com

[msk-databases]
msk-dev-db01.company.com

[msk-applications]
msk-dev-app01.company.com
```

# Dynamic Inventory

- Получение inventory из других источников
  - Consul
  - Zabbix
  - OpenStack
  - Vmware
  - Любая БД с нужным скриптом-адаптером
- Постоянно актуальное состояние inventory
- Информация о группах, переменных

# Dynamic Inventory

- Создает скрипт-адаптер к нужной backend среде, который поддерживать аргумент --list, возвращающий данные в JSON.
- JSON-документ содержит информацию об инстансах, сетевых интерфейсах, служебной мета-информации, host и group переменных

# Dynamic Inventory - Пример

```
ansible-inventory -i ~/Development/playbooks/inventory/prod.aws_ec2.yml --list
{
    "_meta": {
        "hostvars": {
            "ec2-5x-xx-x-xxx.us-west-2.compute.amazonaws.com": {
                "AmiLaunchIndex": 2,
                "Architecture": "x86_64",
                "BlockDeviceMappings": [
                    {
                        "DeviceName": "/dev/sda1",
                        "Ebs": {
                            "AttachTime": "2017-12-13T15:40:19+00:00",
                            "DeleteOnTermination": false,
                            "Status": "attached",
                            "VolumeId": "vol-0514xxx"
                        }
. . .
```


# Ansible Ad-Hoc

# Ansible Ad-Hoc

- Позволяет выполнять определенные команды над выбранными группами хостов.

```
$ ansible ny-databases -a "date" -u "admin" «
ny-prod-db01.company.com | SUCCESS | rc=0 >>
Sat Jul 21 09:48:59 EST 2018
ny-prod-db02.company.com | SUCCESS | rc=0 >>
Sat Jul 21 09:48:59 EST 2018
```

```
$ ansible ny-databases -a "echo 127.0.0.1 >> /etc/hosts" -u "admin" -K
SUDO password:
ny-prod-db01.company.com | SUCCESS | rc=0 >>
127.0.0.1 >> /etc/hosts
ny-prod-db02.company.com | SUCCESS | rc=0 >>
127.0.0.1 >> /etc/hosts
```

# Ansible Ad-Hoc

- Позволяет использовать модули ad-hoc

```
$ ansible msk-databases -m file -a "dest=/tmp/test.txt mode=0755 owner=root group=root state=touch"
-u "admin" -Kb
SUDO password:
msk-dev-db01.company.com
 | SUCCESS => {
    "changed": true,
    "dest": "/tmp/test.txt",
    "gid": 0,
    "group": "root",
    "mode": "0755",
    "owner": "root",
    "size": 0,
    "state": "file",
    "uid": 0
}
```

# Ansible Ad-Hoc

- Позволяет использовать модули ad-hoc

```
$ ansible msk-prod-web-1.company.com -m yum -a "name=nginx state=present" -u manager -Kkb
SSH password:
SUDO password[defaults to SSH password]:
msk-prod-web-1.company.com | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Loaded plugins: fastestmirror\nLoading mirror speeds from cached hostfile\n * base:
centos-mirror.rbc.ru\n * extras: dedic.sh\n * updates: dedic.sh\nResolving Dependencies\n--> Running
transaction check\n---> Package nginx.x86_64 1:1.12.2-2.el7 will be installed\n--> Processing
Dependency: nginx-filesystem = 1:1.12.2-2.el7 for package: 1:nginx-1.12.2-2.el7.x86_64\n-->
Processing Dependency: nginx-all-modules = 1:1.12.2-2.el7 for package:
1:nginx-1.12.2-2.el7.x86_64\n--> Processing Dependency: nginx-filesystem for package:
1:nginx-1.12.2-2.el7.x86_64\n--> Processing Dependency: libprofiler.so.0()(64bit) for package: ...
```


# Ansible Playbooks

# Ansible Playbook

- Перенесем ad-hoc команды в конфигурационные файлы, которые можно регулярно запускать

```
---
- hosts: webservers
  remote_user: manager
  become: yes
  tasks:
    - yum:
        name: nginx
        state: present
    - service:
        name: nginx
        state: started
        enabled: true
```

```
[webservers]
msk-prod-web-1.company.com
```

# Ansible Playbook

- Перенесем ad-hoc команды в конфигурационные файлы, которые можно регулярно запускать

```
$ ansible-playbook playbooks/webservers.yml -i inventory.ini -Kk
SSH password:
SUDO password[defaults to SSH password]:

PLAY [webservers] *************************************************************************************

TASK [Gathering Facts] ********************************************************************************
ok: [msk-prod-web-1.company.com]

TASK [yum] ********************************************************************************************
changed: [msk-prod-web-1.company.com]

TASK [service] ****************************************************************************************
changed: [msk-prod-web-1.company.com]

PLAY RECAP ********************************************************************************************
msk-prod-web-1.company.com  : ok=3    changed=2    unreachable=0    failed=0
```


# Ansible Roles

# Role

- Призвана объединять набор инструкций для конфигурации хоста: установка ПО, поддержка состояния ОС в актуальном состоянии
- Обладает необходимыми возможностями, для параметризации и многократного использования

# Role – Из чего состоит роль?

- Tasks – набор инструкций для выполнения
- Defaults – набор параметров по-умолчанию
- Handlers – набор инструкций для запуска по триггеру
- Files – набор статичных файлов
- Templates – набор файлов-шаблонов
- README.md – описание роли, параметров, требований к запуску

# Tasks

- Представляет из себя последовательный набор действий
- Имеет большое количество «модификаторов» запуска: условный запуск, запуск в цикле, выполнение другого task в случае статуса changed

# Tasks – Структура

## roles/myrole/tasks/main.yml:

```
---
- name: <task_name>
  <module_name>:
    <module_attribute1>: <value>
    <module_attribute2>: <value>
  <task_modificator>: <arguments>
```

# Tasks – Пример Apache

## roles/apache/tasks/main.yml:

```
---
- name: Install Apache Server Ubuntu
  package:
    name: apache2
    state: present # default
  when: ansible_distribution == "Ubuntu"

- name: Install Apache Server CentOS
  package:
    name: httpd
    state: present # default
  when: ansible_distribution == "CentOS"
```

# Tasks – Пример Nginx

## roles/nginx/tasks/main.yml:

```
---
- name: Install Nginx Server Ubuntu
  package:
    name: nginx
    state: present # default
```

# Tasks – Пример Loop

## roles/apache/tasks/main.yml:

```
---
- name: Install Apache Server
  package:
    name: "{{ item }}"
    state: present # default
  loop:
    - httpd
    - mod_wsgi
    - mod_ssl
    - mod_proxy_uwsgi
  when: ansible_distribution == "CentOS"
```

# Tasks – Включение других файлов tasks

## roles/nginx/tasks/main.yml:

```
---
- name: Install Nginx
  import_tasks: install.yml

- name: Configure Nginx
  import_tasks: configure.yml
```

# Defaults

- Позволяют описывать стандартные переменные для роли
- Данные переменные могут быть переопределены при последующих запусках
- Позволяет сделать роль более универсальной

# Defaults - Пример

## roles/nginx/defaults/main.yml:

```
---
nginx_version: 1.12.2
nginx_repo: http://nginx.org/packages/centos/$releasever/$basearch/
nginx_directory: /etc/nginx
nginx_vhosts_folder: vhosts.d
nginx_user: www-data
nginx_worker_proceses: 4
nginx_worker_connections: 200
nginx_sites:
  - "{{ inventory_hostname }}"
  - localhost
nginx_sites_location: /var/www/html
```

# Tasks – Install nginx

## roles/nginx/tasks/install.yml:

```
---
- name: Add repository
  yum_repository:
    name: nginx
    description: Nginx YUM repo
    baseurl: "{{ nginx_repo }}"
    gpgcheck: no
    file: external_repos

- name: Install Nginx Server
  yum:
    name: "nginx-{{ nginx_version }}"
    state: present # default
    update_cache: yes # yum update before launch
```

# Tasks – Install nginx

## roles/nginx/tasks/install.yml:

```
- name: Create group
  group:
    name: "{{ nginx_user }}"
    state: present

- name: Create user
  user:
    name: "{{ nginx_user }}"
    group: "{{ nginx_user }}"
    state: present
```

# Templates

- Шаблоны в связке с defaults позволяют добиться возможности универсально сконфигурировать приложение.
- В зависимости от заданных параметров пользователь может получить различное поведение одной и той же роли.

# Templates - Пример

## roles/nginx/templates/etc/nginx/nginx.conf.j2:

```
user {{ nginx_user }};
worker_processes {{ nginx_worker_proceses }};

events {
  worker_connections {{ nginx_worker_connections }};
}
. . .
http {
  . . .
  include {{ nginx_vhosts_folder }}/*.conf;
}
```

# Templates - Пример

## roles/nginx/templates/etc/nginx/vhosts.d/sites.conf.j2:

```
{% for site in nginx_sites %}
server {
  listen *:80;
  server_name {{ site }};
  location / {
    root {{ nginx_sites_location  }}/{{ site }};
  }
}
{% endfor %}
```

# Template module

## roles/nginx/tasks/configure.yml:

```
---
- name: Create configuration file
  template:
    src: etc/nginx/nginx.conf.j2
    dest: "{{ nginx_directory }}/nginx.conf"
    owner: root
    group: root
    mode: 644
  notify: Reload Nginx
```

# Template module

## roles/nginx/tasks/configure.yml:

```
- name: “Create {{ nginx_vhosts_folder }}”
  file:
    path: "{{ nginx_directory }}/{{ nginx_vhosts_folder }}"
    state: directory

- name: Put sites.conf
  template:
    src: etc/nginx/vhosts.d/sites.conf.j2
    dest: "{{ nginx_directory }}/{{ nginx_vhosts_folder }}/sites.conf"
    owner: root
    group: root
    mode: 644
  notify: Reload Nginx
```

# Template module

## roles/nginx/tasks/configure.yml:

```
- name: Create dirs for sites
  file:
    path: "{{ nginx_sites_location }} /{{ item }}"
    owner: "{{ nginx_user }}"
    group: "{{ nginx_user }}"
    mode: 0755
    state: directory
  loop: "{{ nginx_sites  }}"

- name: Put index.html to sites
  copy:
    src: files/localhost/index.html
    dest: "{{ nginx_sites_location }} /localhost/index.html"
    owner: "{{ nginx_user }}"
    group: "{{ nginx_user }}"
    mode: 0644

- name: Put index.html to sites
  copy:
    src: files/test-website/index.html
    dest: "{{ nginx_sites_location }} /{{ inventory_hostname }} /index.html"
    owner: "{{ nginx_user }}"
    group: "{{ nginx_user }}"
    mode: 0644
```

# Template module

## roles/nginx/tasks/configure.yml:

```
- name: Enable and run nginx
  service:
    name: nginx
    state: started
    enabled: true
```

# Handlers

- Описаний инструкций, которые выполняются в случае если task вернул статус changed
- В случае, если несколько changed tasks ссылаются на один handler – он выполнится один раз, в конце прогона роли.

# Handlers - example

## roles/nginx/handlers/main.yml:

```
---
- name: Reload Nginx
  service:
    name: nginx
    state: reloaded
```

# Playbooks

- Playbook позволяет отобразить какие роли необходимо установить на какие хосты
- Внутри playbook можно переопределять переменные

# Playbooks - Пример

- webservers.yml

```
#! /usr/bin/env ansible-playbook
---
- name: Install nginx
  hosts: webservers
  become: true
  roles:
    - nginx
```

# Создаем inventory

## inventory.ini

```
[webservers]
msk-prod-web-1.company.com
```

# Конфигурируем ansible.cfg

## ansible.cfg:

```
[defaults]
host_key_checking = False
roles_path = roles_galaxy:roles_community:roles
inventory = inventory.ini
remote_user=userxx
```

# Playbooks - Применение

```
ansible-playbook playbooks/webservers.yml -Kk
```

```
SSH password:
SUDO password[defaults to SSH password]:

PLAY [Install nginx] ***********************************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [msk-prod-web-1.company.com]

TASK [nginx : Add repository] **************************************************************************
changed: [msk-prod-web-1.company.com]

TASK [nginx : Install Nginx Server] ********************************************************************
changed: [msk-prod-web-1.company.com]

TASK [nginx : Create group] ****************************************************************************
changed: [msk-prod-web-1.company.com]

TASK [nginx : Create user] *****************************************************************************
changed: [msk-prod-web-1.company.com]

TASK [nginx : Create configuration file] ***************************************************************
changed: [msk-prod-web-1.company.com]
```

# Group vars

- Переменные group_vars позволяют переопределять defaults переменные для конкретных групп хостов
- При правильной структуре репозитория ansible вы также можете определять разные параметры для разных контуров (qa, prod, dev) для одной и той же группы хостов

# Group vars

playbooks/group_vars/webservers/vars.yml:

```
---
nginx_user: www-data
```

# Структура репозитория

```
playbooks/
   group_vars/
     webservers/
       vars.yml
   webservers.yml
roles/
   nginx/
     tasks/
     handlers/
     files/
     templates/
     vars/
     defaults/
     meta/
ansible.cfg
inventory.ini
```

# Group vars - Пример

environments/prod/group_vars/nginx/all.yml:

```
---
nginx_worker_proceses: 10
nginx_worker_connections: 1000
```

environments/dev/group_vars/nginx/all.yml:

```
---
nginx_worker_proceses: 4
nginx_worker_connections: 100
```

# Host vars

- Позволяет назначать переменные для конкретных хостов
- Является очень удобные инструментом при настройке кластеров

# Host vars - Пример

## Environments/prod/hosts

```
[nginx]
ny-prod-web-1.company.com nginx_worker_proceses=20
ny-prod-web-2.company.com nginx_worker_proceses=10
ny-prod-web-3.company.com nginx_worker_proceses=10

[postgres]
ny-prod-pgsql-1.company.com role=master
ny-prod-pgsql-2.company.com role=slave
```

# Домашнее задание

Учебные ansible-репозиторий с ролью nginx:
https://gitlab.com/tfs_s18_admin/ansible-nginx

В нем необходимо поменять (inventory.ini, ansible.cfg):

```
[webservers]
your-study-machine.host.name
```

```
[defaults]
host_key_checking = False
roles_path = roles_galaxy:roles_community:roles
inventory = inventory.ini
remote_user=user0
```

# Домашнее задание

Вам необходимо написать роли для postgresql и nginx, который будет проксировать запросы до django приложения, работающее на 8000 порту.

Установленные роли должны создавать работающее приложение, которое отдает требуемые веб-страницы.
Роль требуется оформить в формате репозитория ansible. Используйте http://bitbucket.fintech-admin.m1.tinkoff.cloud

# Полезные ссылки

- Официальная документация: https://docs.ansible.com/ansible/latest/index.html
- Best practices: https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html
- Google.com: (files|yum|template|...) ansible для документации по модулям

# Спасибо за внимание!

Артем Шепелев
a.shepelev@tinkoff.ru
Telegram: @ashepelev