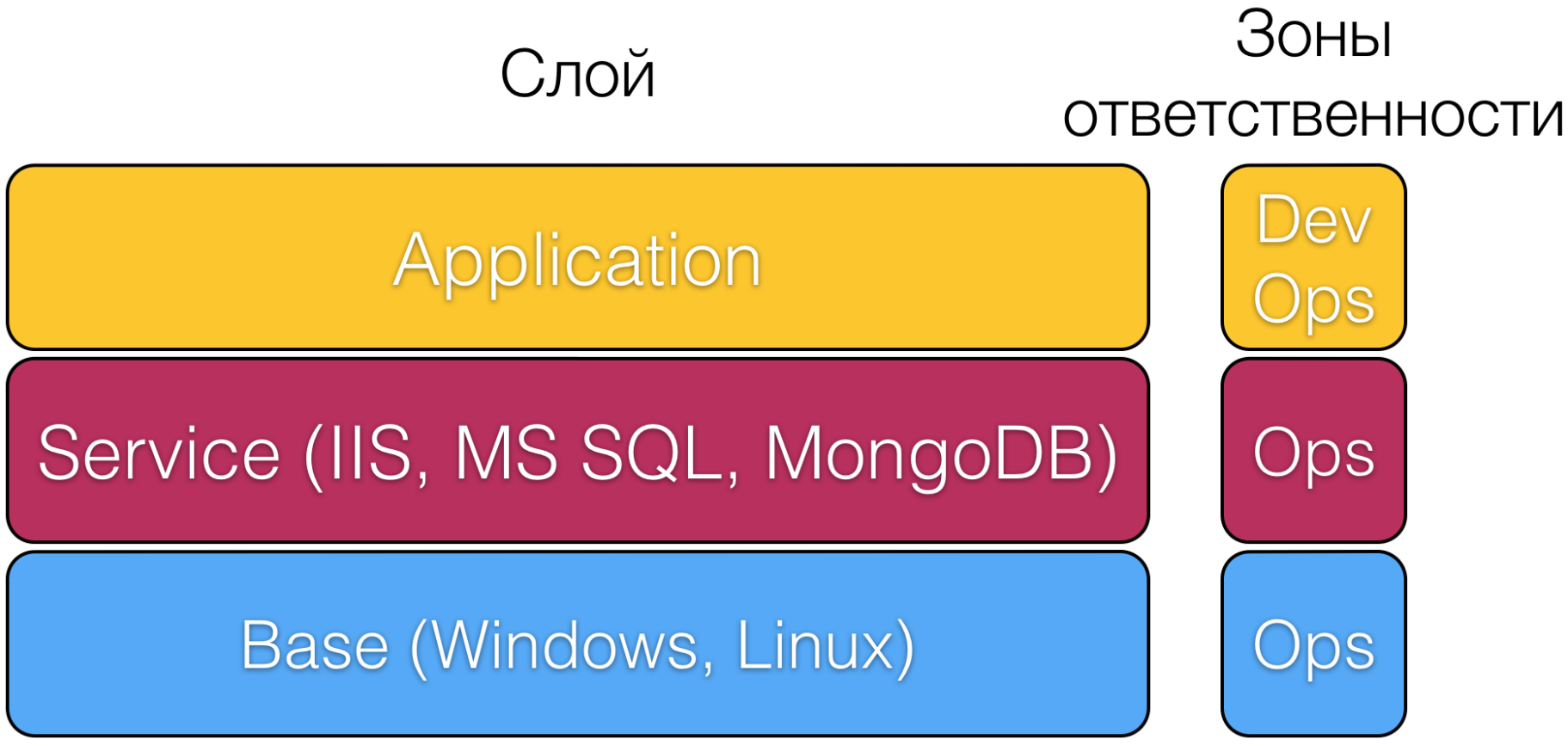
Расширенные возможности Ansible

Не забудь включить запись!

План

- Base-service-app модель и применимость Ansible
- Templates
- Handlers
- Dynamic Inventory
- Secrets
- Tags
- Debug
- Demo

Base-service-app модель



Ansible подходит для управления каждым из уровней

Управление целевым состоянием окружения

- Настройки ОС
- Установка и настройка системных пакетов и библиотек
- Управление конфигурациями
- Настройки безопастности

Деплой

- Управление каталогами для приложения и их правами
- Настройка файлов конфигурации приложений
- Обновление кодовой базы
- Актуализация библиотек приложений
- Перезапуск сервисов

Оркестрация действий

- Управление очередностью внесения изменений
- Применение изменений для небольшой части инфраструктуры

Управление облачными ресурсами

Ansible включает в себя большое количество модулей для управления облачными ресурсами. Как публичными, так и приватными:

- Amazon
- Azure
- Google
- OpenStack
- VMware
- и многие другие

Templates

- Параметризованные файлы конфигурации
- Возможность использования условий и переменных
- Возможность переиспользования конфигурации
- Описываются при помощи шаблонизатора Jinja2

Документация

Шаблоны (пример)

Файл конфигурации `templates/mongodb/mongod.conf.j2`:

```
# network interfaces
net:
  port: 27017
  bindIp: {{ mongodb_listen_ip }}        # <-- Использование переменных

# Replication params
{% if mongodb_replication_replset -%}    # <-- Условный оператор на основе
переменных
replication:
  replSetName: {{ mongodb_replication_replset }}
{% endif %}
```

Пример вызова шаблона:

```
- name: Create mongod.conf file for MongoDB
  template:
    src: templates/mongodb/mongod.conf.j2
    dest: /etc/mongod.conf
```

Handlers

- Это задачи, которые вызываются другими модулями через инструкцию `notify`. Например, перезагрузка сервиса, если изменилась конфигурация
- В норме, хэндлер вызывается, если основная задача имеет статус **CHANGED**
- Каждый вызванный хэндлер выполняется **только один раз**, *в конце выполнения play* (независимо от того, сколько раз сработал `notify` для этого хэндлера)
- На один хэндлер могут ссылаться несколько задач
- Задача-хэндлер может вызываться по нескольким событиям (через атрибут `listen`)

Handlers

- На вызов хэндлеров не влияют параметры командной строки `--tags` или `--skip-tags`
- Принудительное выполнение хендлера возможно при помощи модуля `meta`:

```
meta: flush_handlers
```

- Хэндлеры подключенных ролей вызываются *в конце сценария*, вместе с остальными (а не после применения задач роли)

Документация

Handlers: Пример

Файл `my_playbook.yml`:

```
handlers:
- name: reload mongodb  # <-- Имя задачи потом задаем в директиве notify
  service:
    name: mongod
    state: reloaded

tasks:
- name: Create mongod.conf file for MongoDB
  template:
    src: templates/mongodb/mongod.conf
    dest: /etc/mongod.conf
  notify:
    - reload mongodb    # <-- Вызовем эту задачу, если конфигурация изменится
```

Handlers: Work-around

Иногда, приходится использовать "хэндлер для бедных", чтобы обойти правила для обычных хэндлеров:

```
tasks:
- name: Create mongod.conf file for MongoDB
  template:
    src: templates/mongodb/mongod.conf
    dest: /etc/mongod.conf
  register: mongo_cfg_result

- service:
    name: mongod
    state: reloaded
  when: mongo_cfg_result.changed == True

# Тот печальный случай, когда у нас словарь с нечеткой структурой:
# when: loop_rc is defined and loop_rc.results is defined and
loop_rc.results[0].changed == True
```

Handlers

Можно использовать `include` и `import` для хэндлеров.

Пример:

```
handlers:
  - import_tasks: handlers.yml
```

Dynamic Inventory

- Позволяет работать с inventory полученным из внешних сервисов
- Решает проблему дублирования и отслеживания хостов, которые управляются через специализированное ПО
- Есть готовые скрипты для интеграции с GCP/EC2/Openstack
- Поддерживается использование внешних скриптов для получения данных из внешних систем (в формате JSON), либо вызов специальных модулей
- Для "особых случаев" есть модуль `add_host`

Dynamic + Static inventories

- Если вы хотите использовать dynamic и static инвентори вместе, то можно указать путь до папки:
- Ansible будет запускать смотреть все текстовые файлы и запускать исполняемые
- Ansible не будет использовать .ini файлы

Пример: `ansible-playbook -i path/to/inventories main.yml`

Пример:Add host

```
    - name: Add new host to our inventory.
      add_host:
        name: "{{ do.droplet.ip_address }}"
        groups: do
```

```
- hosts: do
  remote_user: ubuntu
  tasks:
    - name: Wait for ssh port become available
      local_action: "wait_for port=22 host={{ inventory_hostname }}"

    - name: install packages
      apt:
        name: "{{ packages }}"
```

Ansible Vault

- Консольная утилита для работы с секретными данными (сертификаты, приватные ключи и т.д.)
- Переменные можно подключать через `vars_files`
- Для более сложных случаев пользуйтесь Hashicorp Vault и аналогами
- Позволяет хранить в общих репозиториях файлы, в которых содержится приватная информация (пароли, токены)
- Использование зашифрованных переменных полностью прозрачно. В процессе выполнения `ansible` получает к ним доступ. Файлы на диске остаются зашифрованными.

Документация

Пример: Шифрование файла

Импорт файла с секретами:

```
vars_files:
  - secrets.yml
```

Пример содержимого файла до шифрования:

```
secret_repo_user: admin
secret_repo_pass: some_long_pass
```

Пример содержимого файла после шифрования:

```
$ANSIBLE_VAULT;1.1;AES256
633065636639316234353136313864333464326437363739653962363639666339636234653537333
663031313535626261306564653663396530343362313332 0a31643663626566343463623736373 2
....
```

Пример: Шифрование файла

Чтобы зашифровать существующий файл выполняем команды:

```
ansible-vault --vault-id dev@prompt encrypt --encrypt-vault-id dev secrets.yml
```

Файл с секретами будет запаролен, в будущем, когда мы захотим выполнить playbook с использованием `secrets.yml`, необходимо передать параметр `--ask-vault-pass`

Либо мы можем указать в `ansible.cfg` опцию **vault_password_file**, которая указывает на файл с ключом.

Пример:

```
ansible-playbook myplaybook.yml --ask-vault-pass
```

Пример: Шифрование строки

```
ansible-vault encrypt_string --vault-id dev@password --encrypt-vault-id dev
'fooooodev' --name 'the_dev_secret'
```

Результат можно вставить непосредственно в код сценария:

```
the_dev_secret: !vault |
  $ANSIBLE_VAULT;1.2;AES256;dev

30613233633461343837653833666333643061636561303338373661313838333565653635353162

32633634346237333435386534626130643336343334646660a6636336239393934393166366338636

61636237636537333938306331383339353265363239643939666639386530626330633337633833

666465633437316663 0a3637363932626664656634326139326130363039633432636231373862396330
```

Как указать ключ (Vault key):

1. Из командной строки

```
--vault-id <label>@prompt
```

2. Из plain text файла

```
--vault-id <label>@/path/to/vault.key
```

3. Из исполняемого файла

```
--vault-id <label>@key.sh
```

4. Сразу несколько

```
--vault-id messages_db@/path/to/db.key --vault-id
users_db@/path/to/users.key
```

P.S. Можно использовать `label` для того, чтобы работать с несколькими окружениями с разными ключами Vault

Тегированные задачи (Tags)

- Ansible позволяет указывать метки (tags) в плейбуках (для импортированных сценариев и задач)
- Для каждого сценария и задачи может быть установлено более одной метки
- Метки можно использовать для запуска определенных этапов сценария (ключ `--tags`)
- Либо можно пропускать части сценария по меткам (ключ `--skip-tags`)

Документация

Пример

```
# Setup and configure ruby things for reddit app
- name: "Reddit App: Configure app environment"
  hosts: all
  gather_facts: no
  become: yes

  tasks:
  - name: Install ruby and rubygems and required packages
    apt:
      <...>
    tags:
      - app_packages

  - name: Install Ruby bundler
    gem:
      <...>
    tags:
      - app_packages
```

Пример

```
- name: "Reddit App: Install mongodb"
  hosts: all
  gather_facts: no

  tasks:
  - name: Install mongodb package
    apt:
    <...>
    tags:
      - db_packages

# Setup and configure mongodb for reddit app
- name: "Reddit App: Install mongodb"
  hosts: all
  gather_facts: no

- import_playbook: gce.yml
  tags:
    - deploy
```

Пример

Выполняем определенные сценарии:

```
ansible-playbook mytask.yml --tags "app_packages, deploy"
```

Пропускаем Deploy:

```
ansible-playbook mytask.yml --skip-tags "deploy"
```

Not-so-obvious tagging

- Напомним, хэндлеры вызываются независимо от `--tags` и `--skip-tags`
- Директива `tags:` вместе с `include:` и `import_***:`
  - Тэги добавляются к каждой вложенной задаче
- Директива `tags:` вместе с `include_***:`
  - Тэги добавляются к самой задаче `include_***:`
  - Тэги вложенных задач остаются без изменений
- Тэг `always` игнорирует ключ `--skip-tags`
- Тэг `never` - таск не будет выполняться без задания `--tags`
- Несколько значений ключа `--tags` имеют логику **ИЛИ**

Поиск ошибок

- Уровень verbosity `-vvvv` (`-vvv`, `-vv`, `-v`)
- Пошаговое выполнение
- `ansible-console`
- Модуль `debug`
- Стратегия `debug` - вызывает отладчик PDB в случае проблем

Подробный вывод

`ansible -m ping localhost -vvvv`:

```
ansible 2.7.5
  config file = None
  configured module search path = [u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.15 (default, Jun 17 2018, 12:46:58)
...
META: ran handlers
<127.0.0.1> ESTABLISH LOCAL CONNECTION FOR USER: user00
<127.0.0.1> EXEC /bin/sh -c 'echo ~user00 && sleep 0'
<127.0.0.1> EXEC /bin/sh -c '( umask 77 && mkdir -p "` echo
/home/user00/.ansible/tmp/ansible-tmp-1548101889.33-95635251857245 `" && echo
ansible-tmp-1548101889.33-95635251857245="` echo
/home/user00/.ansible/tmp/ansible-tmp-1548101889.33-95635251857245 `" ) && sleep
0'
Using module file /usr/lib/python2.7/site-
packages/ansible/modules/system/ping.py
<127.0.0.1> PUT /home/user00/.ansible/tmp/ansible-local-81668UtHWhm/tmprHx0m5 TO
/home/user00/.ansible/tmp/ansible-tmp-1548101889.33-
95635251857245/AnsiballZ_ping.py
...
```

Подробный вывод (продолжение)

Одного слайда не хватило 😄

```
...
<127.0.0.1> EXEC /bin/sh -c 'rm -f -r /home/user00/.ansible/tmp/ansible-tmp-
1548101889.33-95635251857245/ > /dev/null 2>&1 && sleep 0'
localhost | SUCCESS => {
    "changed": false,
    "invocation": {
        "module_args": {
            "data": "pong"
        }
    },
    "ping": "pong"
}an
META: ran handlers
META: ran handlers
```

Пошаговое выполнение

```
root@ansible-vm:~/ansible-demo# ansible-playbook balancers.yml --step

PLAY [balancers] ***************************************************************************************************************************
Perform task: TASK: python : Install python for Ansible (N)o/(y)es/(c)ontinue: y

Perform task: TASK: python : Install python for Ansible (N)o/(y)es/(c)ontinue: *************************************************************

TASK [python : Install python for Ansible] *************************************************************************************************
ok: [balancer]
Perform task: TASK: nginx : Install nginx (N)o/(y)es/(c)ontinue: y

Perform task: TASK: nginx : Install nginx (N)o/(y)es/(c)ontinue: ***************************************************************************

TASK [nginx : Install nginx] ***************************************************************************************************************
ok: [balancer]
Perform task: TASK: nginx : Configure nginx (N)o/(y)es/(c)ontinue: y

Perform task: TASK: nginx : Configure nginx (N)o/(y)es/(c)ontinue: ***************************************************************************

TASK [nginx : Configure nginx] *************************************************************************************************************
ok: [balancer]
Perform task: TASK: nginx : Copy htpasswd file to host (N)o/(y)es/(c)ontinue: y

Perform task: TASK: nginx : Copy htpasswd file to host (N)o/(y)es/(c)ontinue: *************************************************************

TASK [nginx : Copy htpasswd file to host] **************************************************************************************************
ok: [balancer]

PLAY RECAP *********************************************************************************************************************************
balancer                   : ok=4    changed=0    unreachable=0    failed=0
```

ansible-console

```
root@ansible-vm:~/ansible-demo# ansible-console -l balancer
Welcome to the ansible console.
Type help or ? to list commands.

ubuntu@all (1)[f:5]$ ping
balancer | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ubuntu@all (1)[f:5]$ shell 'whoami'
balancer | CHANGED | rc=0 >>
ubuntu

ubuntu@all (1)[f:5]$ become yes
ubuntu@all (1)[f:5]# shell 'whoami'
balancer | CHANGED | rc=0 >>
root

ubuntu@all (1)[f:5]#
```

Модуль debug

```
vars:
  test_var: This is a test

tasks:
  - name: Show var
    debug:
      var: test_var

  - name: Show message
    debug:
      msg: "Var is {{ test_var }}"
```

Модуль debug

```
→ practice ansible-playbook vars.yml

PLAY [Ideal playbook] **********************************************************

TASK [Show var] ****************************************************************
ok: [common] => {
    "test_var": "This is a test"
}

TASK [Show message] ************************************************************
ok: [common] => {
    "msg": "Var is This is a test"
}

PLAY RECAP *********************************************************************
common                     : ok=2    changed=0    unreachable=0    failed=0

→ practice
```

Стратегия debug

```
strategy: debug

vars:
  test_var: This is a test

tasks:
  - name: Show var
    debug:
      var: test_var

  - name: Show message
    ping:
      data: "Var is {{wrong_var}}"
```

Стратегия debug

```
➜  practice ansible-playbook vars.yml

PLAY [Ideal playbook] *********************************************************

TASK [Show var] ***************************************************************
ok: [common] => {
    "test_var": "This is a test"
}

TASK [Show message] ***********************************************************
fatal: [common]: FAILED! => {"failed": true, "msg": "the field 'args' has an inva
lid value, which appears to include a variable that is undefined. The error was:
'wrong_var' is undefined\n\nThe error appears to have been in '/Users/brun/Google
 Drive 42/Devops master class/Ansible/code/practice/vars.yml': line 16, column 7,
 but may\nbe elsewhere in the file depending on the exact syntax problem.\n\nThe
offending line appears to be:\n\n\n    - name: Show message\n      ^ here\n"}
Debugger invoked
(debug) p task
TASK: Show message
(debug) p task.args
{u'data': u'Var is {{wrong_var}}'}
(debug) task.args['data']='asdf'
(debug) p vars['test_var']
u'This is a test'
(debug) r
ok: [common]

PLAY RECAP ********************************************************************
common                     : ok=2    changed=0    unreachable=0    failed=0
```

Название задачи

Аргументы

Повторить задачу

Полезные ссылки

- Про динамический инвентори на русском - тут
- Примеры динамических инвентори:
  - terraform.py
  - Terraform Inventory
  - gce.py
  - ec2.py