Ansible: роли, окружения и лучшие практики

Не забудь включить запись!

План

- Организация инфраструктурного кода
- Роли в Ansible
- Работа с окружениями

Плейбуки для описания инфраструктуры

Плюсы:

- Просто писать и читать
- Просто отлаживать

Плейбуки для описания инфраструктуры

```
- name: Some playbook
  hosts: all
  gather_facts: false

  vars:
    somevar: "Hello, world"

  tasks:
    - name:
      shell: "echo {{somevar}}"
      notify:
        - hello handler

  handlers:
    - name: hello handler
      debug:
        msg: "Some useless handler"
```

Плейбуки для описания инфраструктуры

Плюсы:

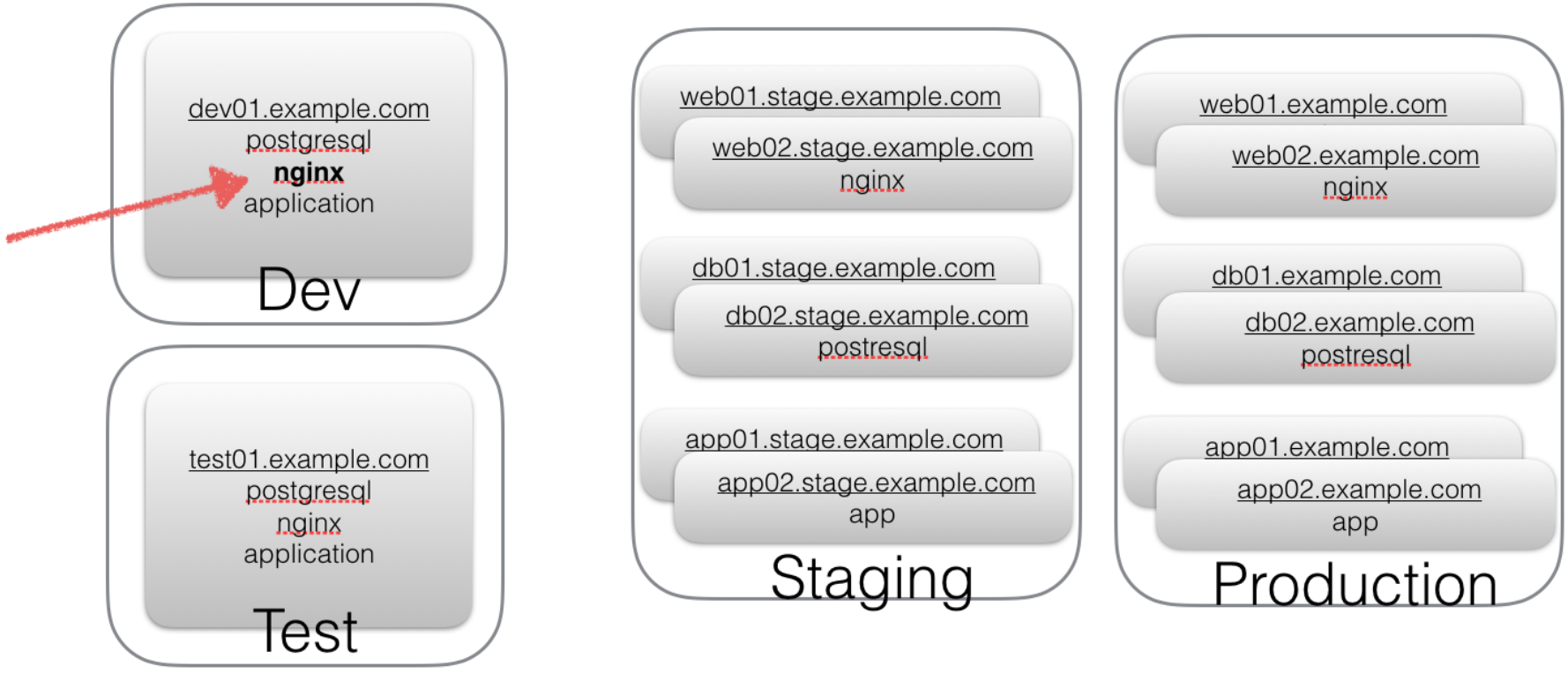
- Просто писать и читать
- Просто отлаживать

Но...

- Быстро разрастаются в размере
- Быстро разрастаются в количестве (копипаст)
- ...

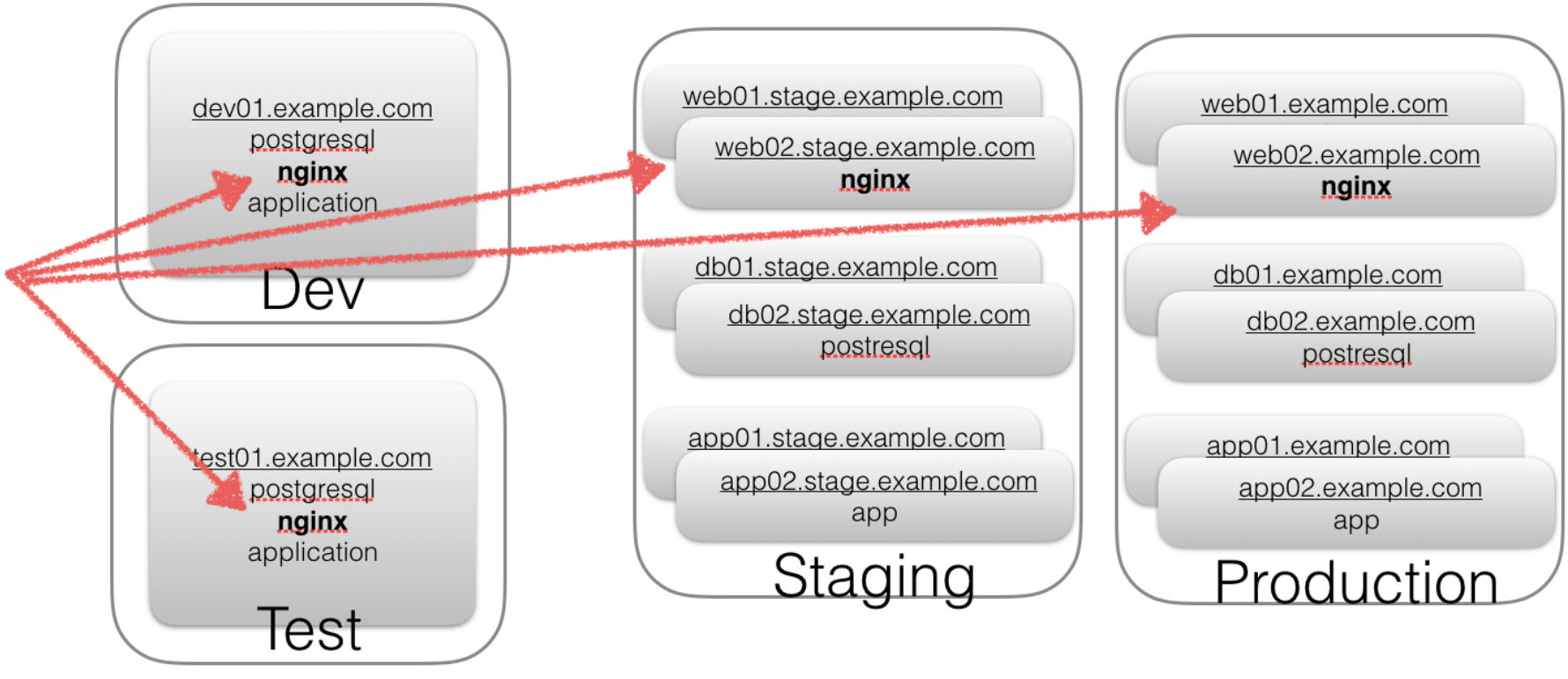
Плейбуки для описания инфраструктуры

Для новой фичи нужно изменить плейбук `nginx.yml`:



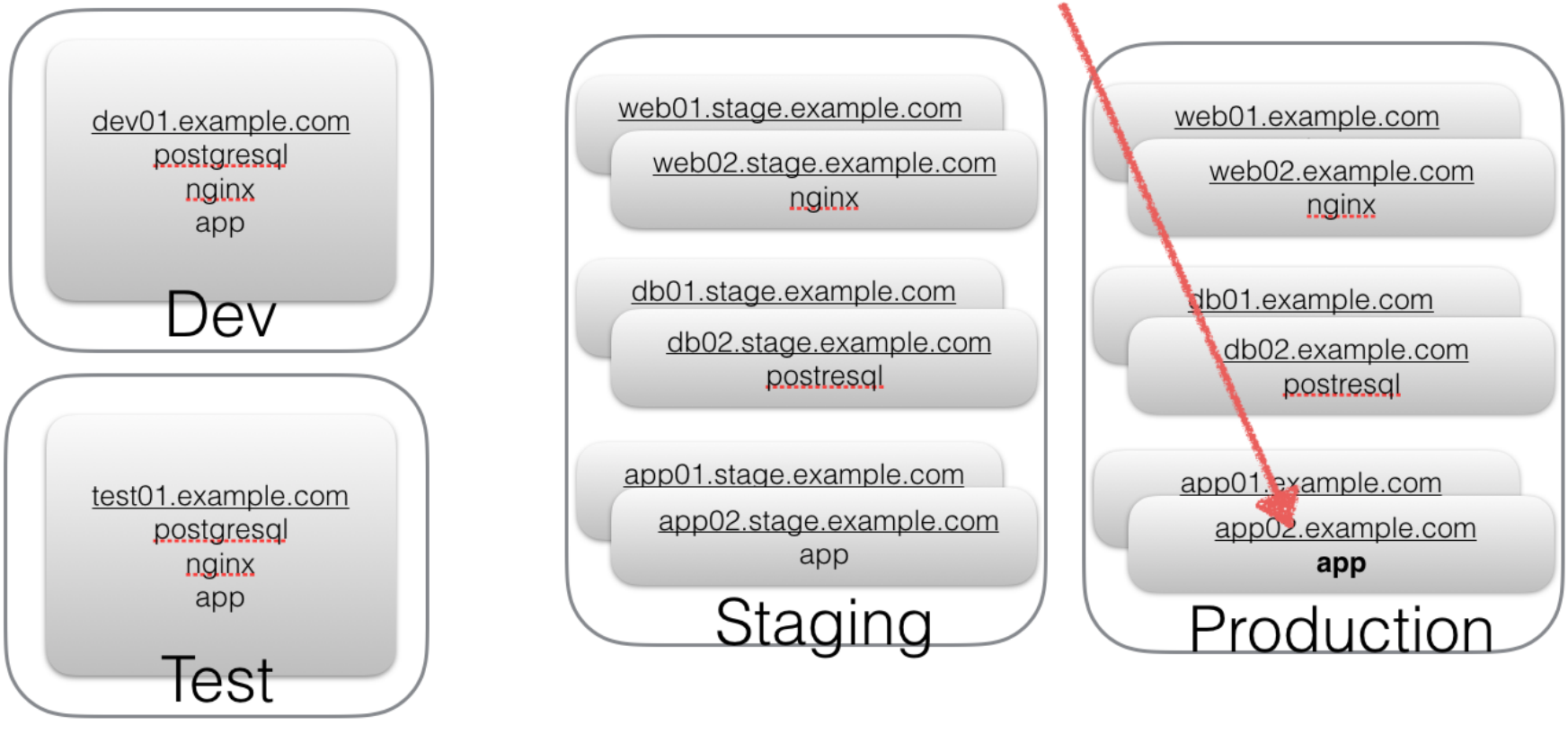
Плейбуки для описания инфраструктуры

Вот только `nginx.yml` используется не только в *dev*:



Плейбуки для описания инфраструктуры

Необходим срочный bugfix плейбука `app.yml` (только для *prod*):



Плейбуки для описания инфраструктуры

Плюсы:

- Просто писать и читать
- Просто отлаживать

Но...

- Быстро разрастаются в размере
- Быстро разрастаются в количестве (копипаст)
- Не версионируются

Плейбуки для описания инфраструктуры

Плюсы:

- Просто писать и читать
- Просто отлаживать

Но...

- Быстро разрастаются в размере
- Быстро разрастаются в количестве (копипаст)
- Не версионируются
- Не подходят как формат для распространения и переиспользования кода (нет версии, зависимостей и метаданных, зато много хардкода)

Роли Ansible

Роль в Ansible – это директория с определенной структурой и YAML-файлами

Роль содержит:

- Таски (вызовы Ansible-модулей) и хендлеры
- Наборы переменных
- Метаданные (версии и зависимости от других ролей)
- Тесты (для локальной отладки и разработки)
- Вспомогательные файлы и шаблоны

Структура роли

```
$ tree example-role
example-role
├── README.md
├── defaults
│   └── main.yml          # <- Переменные и значения по умолчанию
├── files
├── handlers
│   └── main.yml          # <-- Обработчики (aka хэндлеры)
├── meta
│   └── main.yml          # <-- Информация о роли и зависимостях
├── tasks
│   └── main.yml          # <-- Основные задачи в роли
├── templates
│   └── mongod.conf.j2    # <-- Шаблоны конфигурации
├── tests
│   ├── inventory         # <-- Сценарии и данные для тестирования
│   └── test.yml
└── vars
    └── main.yml          # <-- Внутренние переменные роли
```

Роли Ansible

- Инкапсуляция и абстракция
- Отделение кода от данных
- Единица распространения
- Единица тестирования
- Организация зависимостей

Если роль разрастается...

```
example-role
├── README.md
├── defaults
│   └── main.yml
…
├── tasks
│   ├── repository.yml          # example-role/tasks/main.yml
│   ├── install.yml             - include: repository.yml
│   ├── configure.yml           - include: install.yml
│   ├── update.yml              - include: configure.yml
│   └── main.yml
…
└── meta
    └── main.yml
```

Кросс-платформенные роли

```
example-role
├── README.md
├── defaults
│   └── main.yml
…
├── tasks                    # example-role/tasks/main.yml
│   ├── linux.yml            - include: linux.yml
│   ├── windows.yml            when: ansible_os_family == 'RedHat'
│   └── main.yml             - include: windows.yml
…                              when: ansible_os_family == 'Windows'
└── meta
    └── main.yml
```

Что остается плейбукам?

В плейбуках остается совсем немного:

собрать роли в одно целое, обозначить хосты и переменные

```
- hosts: webservers

  vars_files:
    - "{{ inventory_dir }}/credentials.yml"

  roles:
    - nginx
```

Что остается плейбукам?

```
- hosts: databases
  vars_files:
    - "{{ inventory_dir }}/credentials.yml"
  roles:
    - role: debops.lvm
      lvm_volume_groups: [ { vg = 'shared', pvs = '/dev/vdb' } ]
      lvm_logical_volumes: [ { lv='postgresql',  vg='shared', size='100%VG',
mount='/var/lib/postgresql'}]
    - role: ANXS.postgresql
    - role: Stouts.wale
      wale_aws_access_key_id: "{{aws_access_key}}"
      wale_aws_secret_access_key: "{{aws_secret_key}}"
      wale_aws_s3_prefix: "s3://project-db-backups/"
    - role: mjallday.pgbouncer
      pgbouncer:
        user: postgres
        group: postgres
        config:
          listen_addr: '0.0.0.0'
          listen_port: 6432
```

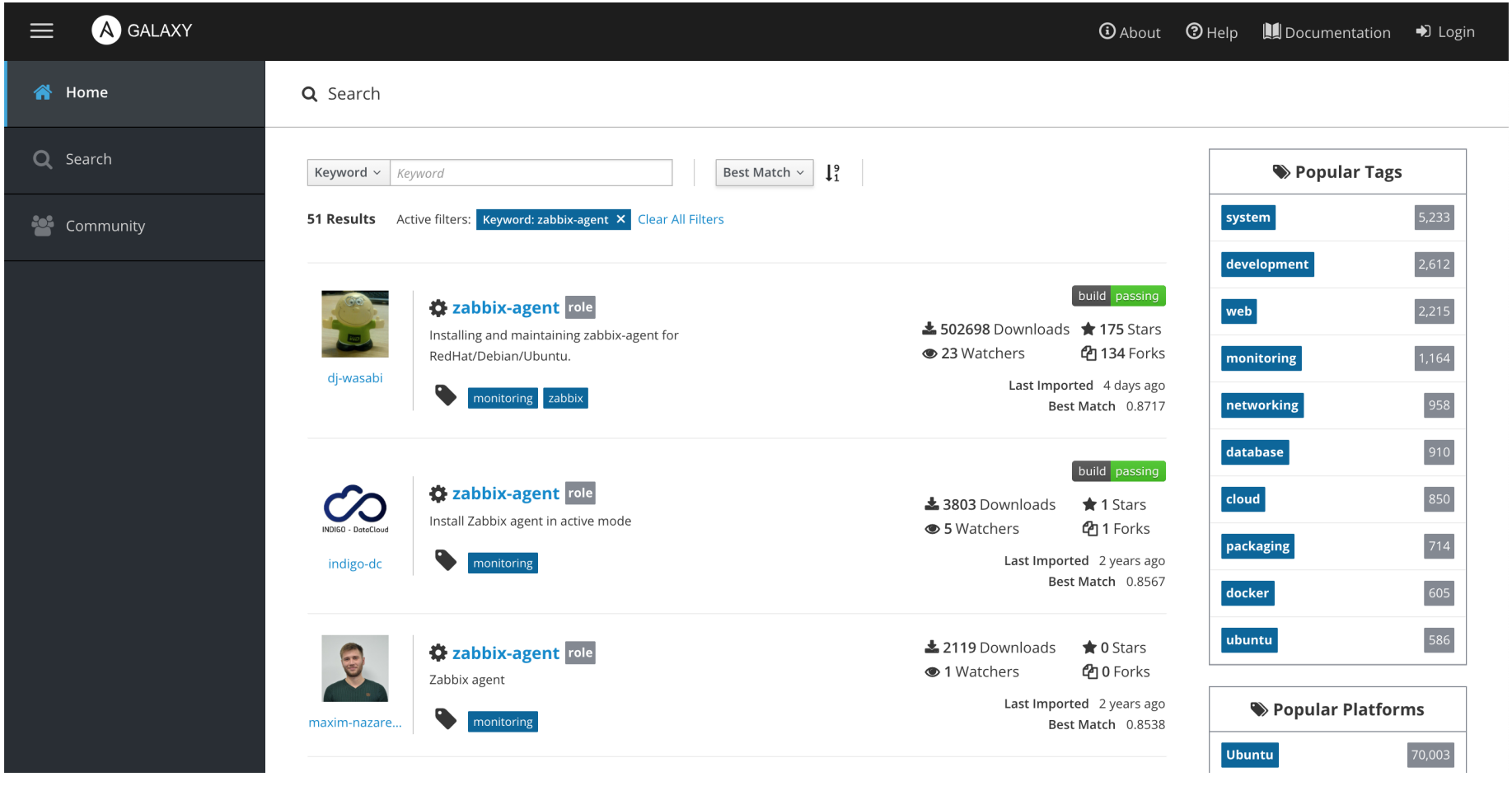
Что остается плейбукам?

```
- hosts: balancers
  connection: local
  gather_facts: no
  tasks:
    - name: balancer | show rservers status
      ios_command:
        commands: show rserver {{ item }}
        provider: "{{ balancer.connect }}"
      with_items: "{{ rservers }}"
      register: rserver_state
      tags: always
    - debug: var=rserver_state
      tags: always
    - block:
      - name: balancer | rservers out of service
        ios_config:
          parents: [ "rserver {{ item }}" ]
          lines: [ "no inservice" ]
          provider: "{{ balancer.connect }}"
        with_items: "{{ rservers }}"
        tags: outofservice
```

Ansible Galaxy

- Портал и центральный репозиторий для ролей Ansible
- `ansible-galaxy` - командная утилита для управления ролями (аналог `pip`/`npm` и подобных)
- Репозиторий насчитывает > 18000 ролей

Ansible Galaxy



Загрузка роли с Galaxy и ее использование

1. Скачать:

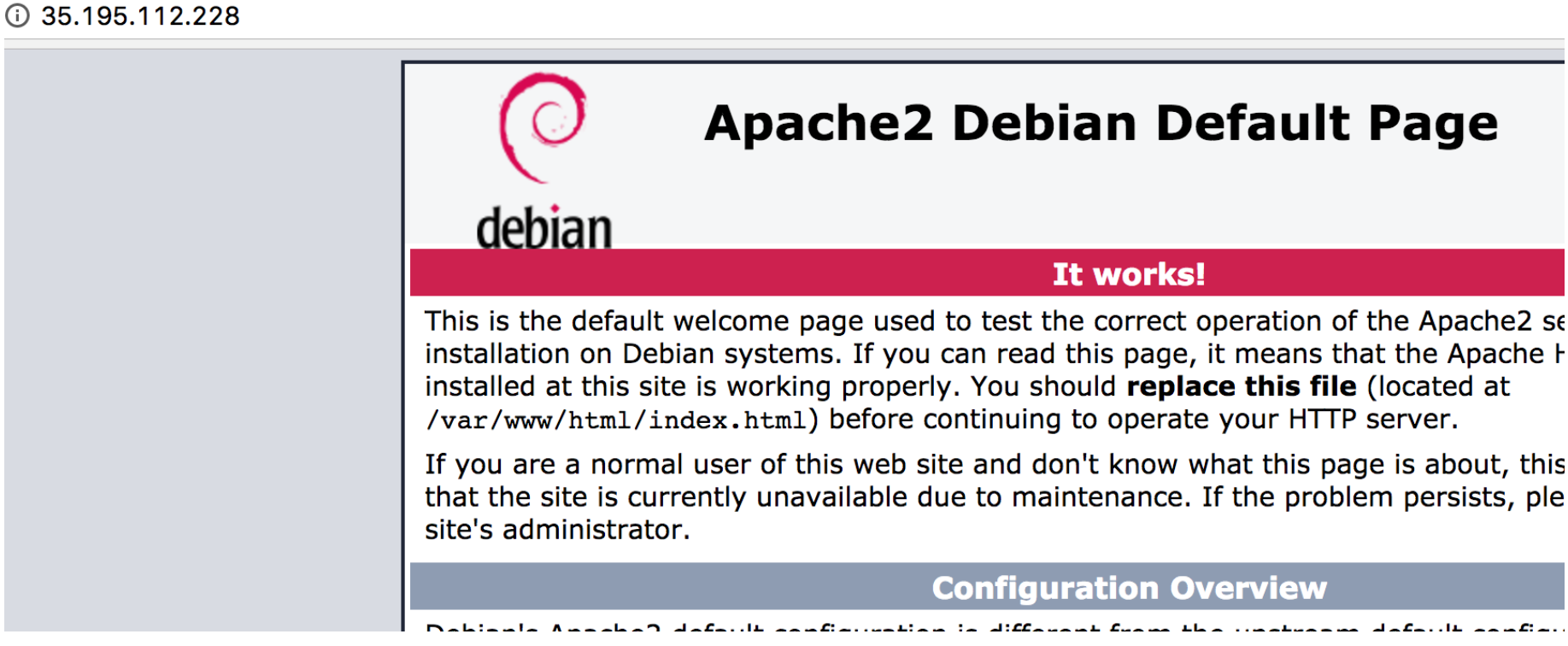
```
$ ansible-galaxy install geerlingguy.apache

- downloading role 'apache', owned by geerlingguy
- downloading from https://.../ansible-role/apache/2.1.1.tar.gz
- extracting geerlingguy.apache to ./geerlingguy.apache
- geerlingguy.apache (2.1.1) was installed successfully
```

2. Применить (пример `playbook apache.yml`):

```
- hosts: webservers
  become: true
  roles:
    - geerlingguy.apache
```

Загрузка роли с Galaxy и ее использование

```
$ ansible-playbook apache.yml      # ... и в продакшн
```



Файл с зависимостями для сценариев

Необходимые для сценария роли прописываем в `requirements.yml`:

```
- src: https://github.com/Stouts/Stouts.iptables.git
  version: 1.1.2

- src: git+https://github.com/chris/universal-tomcat
  version: f0fe23a3b1541cbc60cabe69afe401f3df07d5d3

- src: jdauphant.nginx
```

Устанавливаем все необходимое "в один клик":

```
$ ansible-galaxy install -r requirements.yml
```

Версионирование ролей

- Можно хранить все сторонние роли в своем репозитории с кодом – тогда будут проблемы при обновлении сторонних ролей
- Можно сделать по репозиторию в Git для каждой роли и `requirements.yml` для каждого окружения (например, `environments/dev/requirements.yml`)

Окружения в Ansible

Ansible нет понятия окружений (*dev*, *stage*, *prod*). Но в случае с Ansible они не нужны.

Типичное описание окружения состоит из:

- Списка серверов и их групп
- Средозависимых данных (переменные и файлы)
- Списка ролей и их версий, актуальных для окружения

Окружения в Ansible

```
environments
└── dev
    ├── group_vars
    │   ├── webservers
    │   └── databases
    ├── requirements.yml
    ├── credentials.yml
    └── inventory
```

- Получаем необходимый набор ролей:

```
$ ansible-galaxy install -r environments/<env_name>/requirements.yml
```

- Запускаем плейбук, который вызывает эти роли:

```
$ ansible-playbook -i environments/<env_name>/inventory play.yml
```

Окружения в Ansible

```
ansible.cfg
playbooks
environments
├── dev
│   ├── group_vars
│   │   ├── webservers          # Группы хостов должны быть идентичны,
│   │   └── databases           # даже если отличается схема развертывания
│   ├── requirements.yml
│   ├── credentials.yml
│   └── inventory
└── production
    ├── group_vars
    │   ├── webservers          # Группы хостов должны быть идентичны,
    │   └── databases           # даже если отличается схема развертывания
    ├── requirements.yml
    ├── credentials.yml
    └── inventory
```

Окружения в Ansible

```
ansible.cfg
playbooks
environments
├── dev
│   ├── group_vars
│   │   ├── webservers
│   │   └── databases
│   ├── requirements.yml      # Роли и их версии свои для каждого окружения
│   ├── credentials.yml
│   └── inventory
└── production
    ├── group_vars
    │   ├── webservers
    │   └── databases
    ├── requirements.yml      # Роли и их версии свои для каждого окружения
    ├── credentials.yml
    └── inventory
```

Окружения в Ansible

```
ansible.cfg
playbooks
environments
├── dev
│   ├── group_vars
│   │   ├── webservers
│   │   └── databases
│   ├── requirements.yml
│   ├── credentials.yml     # У окружений отдельные файлы с зашифрованными переменными
│   └── inventory
└── production
    ├── group_vars
    │   ├── webservers
    │   └── databases
    ├── requirements.yml
    ├── credentials.yml     # У окружений отдельные файлы с зашифрованными переменными
    └── inventory
```

Окружения в Ansible

Типовые файлы с переменными окружений:

- `environments/<env_name>/group_vars`
- `environments/<env_name>/host_vars`
- `environments/<env_name>/credentials.yml`

Подключаем в плейбуках так:

```
vars_files:
  - "{{inventory_dir}}/credentials.yml"
```

Файл конфигурации Ansible

- Настройки Ansible должны быть идентичны для команды
- Их так же необходимо версионировать
- Пример `ansible.cfg` с описанием всех параметров доступен по ссылке на GitHub

Файл конфигурации Ansible

Пример файла `ansible.cfg`:

```
[defaults]
roles_path    = ./.imported_roles:./roles
timeout = 10
vault_password_file = vault.key
retry_files_enabled = False

[ssh_connection]
ssh_args=-o ForwardAgent=yes

[diff]
always = True
context = 5
```

Хранение инфраструктурного кода

Код для нашей инфраструктуры можно хранить по-разному...

- В репозитории вместе с кодом продукта
  - Прозрачность работы разработки и эксплуатации через общую систему контроля версий и багтрекер
  - Инфраструктурный код релизится вместе с версией продукта/сервиса
- В выделенном инфраструктурном репозитории
  - Удобнее использовать разные версии кода для разных окружений (зато сложнее собрать их обратно)
  - Выделенной команде Ops проще вносить изменения

Нюансы

Повышение привилегий

Директивы `become` и `become_user` можно указывать:

- Глобально
- На плейбук
- На каждый таск (блоки и т.д.)

Пишите плейбуки и роли так, чтобы действия, которым нужен `root` **были явно выделены**

Нюансы

Пространства имен переменных

Определяя новую переменную в плейбуке или роли называйте ее так, чтобы переменная описывала свое назначение и принадлежность к роли

Например:

```
http_port: 8080                 # ❌ неправильно
shop_backend_http_port: 8080    # ✅ правильно
```

Нюансы

Загружаются все файлы в `group_vars` для хоста

Ansible загружает все переменные хоста из файлов групп, которым он принадлежит в инвентори, даже если группа в плейбуке не участвует в сценарии.

Это может привести к неоднозначности значения переменных.

Так что, используйте префиксы в именах переменных для предотвращения такой ситуации.

Особенности синтаксиса ролей

В Ansible применение ролей поддерживается в нескольких вариантах:

1. Старый вариант синтаксиса (директива `roles:`):

```
- hosts: webservers
  become: true
  roles:
    - geerlingguy.apache
```

2. Новый вариант (директивы `import_role` и `include_role`):

```
- hosts: webservers
  become: true
  tasks:
    - import_role: name=geerlingguy.apache
    - include_role: name=geerlingguy.apache
```

Особенности синтаксиса ролей

Директива `import_role` выполняется препроцессором, до запуска плейбука.

Это значит, что:

1. Не поддерживаются циклы (директива `loop:`)
2. Зато нормально работает директива `delegate_to`
3. Директива `when:` не имеет доступа к переменным групп хостов
4. Таски из роли всегда выполняются в playbook (но если есть условие `when:` мы просто увидим кучу `[SKIPPED]` тасков)

Особенности синтаксиса ролей

Директива `include_role` выполняется в процессе выполнения сценария (аналог `roles:`).

Это значит, что:

1. Она имеет доступ к актуальному состоянию переменных и фактов
2. Если условие не выполняется, то задачи роли не будут запущены
3. Мы можем использовать цикл с директивой `loop:`

```
- include_role: name="{{ item }}"
  when: ansible_local.reddit.is_host_provisioned == False
  loop: ['jdauphant.nginx', 'mongodb', 'reddit-app']
```

Особенности синтаксиса ролей

Но... вот так - не работает:

```
- include_role:
    name: nginx
  delegate_to: balancer-1
```

`delegate_to:`, `tags:` и подобные атрибуты таска `include_role` - **не применяются** для вложенных тасков.

Начиная с Ansible >= 2.7 можно воспользоваться `apply:` и "пробросить" атрибуты внутрь:

```
- include_role:
    name: nginx
    apply:
      delegate_to: balancer-1
```

Особенности синтаксиса ролей

А вот так все равно не работает 😁:

```
- hosts: appservers
  serial: 1
  pre_tasks:
    - name: Remove host from load balancing before running update
      include_role:
        name: nginx
        tasks_from: reconfigure.yml
        apply:
          delegate_to: "{{ item }}"
          become: yes
      loop: "{{ groups['loadbalancers'] }}"
  tasks:
    - name: Run appserver update
      include_tasks: app_redeploy.yml
```

Если есть идеи и решения - welcome! :)