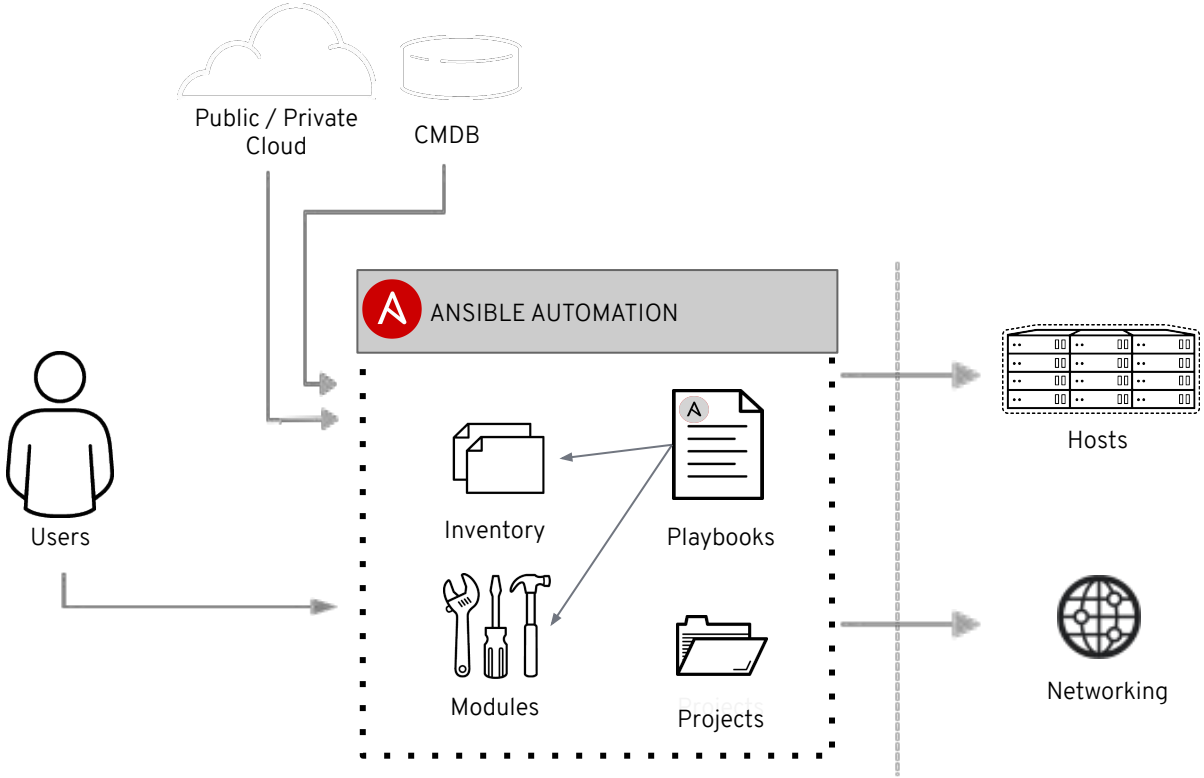
ANSIBLE

ЛУЧШИЕ ПРАКТИКИ ANSIBLE: ОСНОВЫ

Владимир Карагиоз
Red Hat
vladimir@redhat.com

КАК ЭТО ВСЕ РАБОТАЕТ?



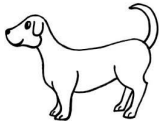
ПУТЬ ANSIBLE

Принцип 1 - СЛОЖНОСТЬ УБИВАЕТ ПРОДУКТИВНОСТЬ

ANSIBLE

Now

Нам нужна автоматизация



Нет проблем. Я Фил.
Я немного автоматизирую это.

5 years later

Эй Ты знаешь, что это такое?
^(?:(?:(?:0?[13578]|1[02])(\/|-|\.)31)\1|(?:(?:0?[13-9]|1[0-2])(\/|-|\.)(?:29|30)\2))(?:(?:1[6-9]|[2-9]\d)?\d{2})$|^(?:0?2(\/|-|\.)29\3(?:(?:(?:1[6-9]|[2-9]\d)?(?:0[48]|[2468][048]|[13579][26])|(?:(?:16|[2468][048]|[3579][26])00))))$|^(?:(?:0?[1-9])|(?:1[0-2]))(\/|-|\.)(?:0?[1-9]|1\d|2[0-8])\4(?:(?:1[6-9]|[2-9]\d)?\d{2})$
...does?

Фил написал это. Удачи, он был убит в великой войне собак-регэкспов в 2019 году.

redhat.

Принцип 2 - ЧИТАБЕЛЬНОСТЬ

ANSIBLE

```
1  - name: Install Tomcat Application server and deploy sample Java app
2    hosts: all
3    tasks:
4      - name: Ensure tomcat is installed
5        yum:
6          name: tomcat
7          state: present
8      - name: Ensure tomcat service is enabled and started
9        service:
10         name: tomcat
11         enabled: yes
12         state: started
13     - name: Download and deploy Java application
14       get_url:
15         url: https://tomcat.apache.org/tomcat-6.0-doc/appdev/sample/sample.war
16         dest: /var/lib/tomcat/webapps/sample.war
17         mode: 0777
```


redhat.

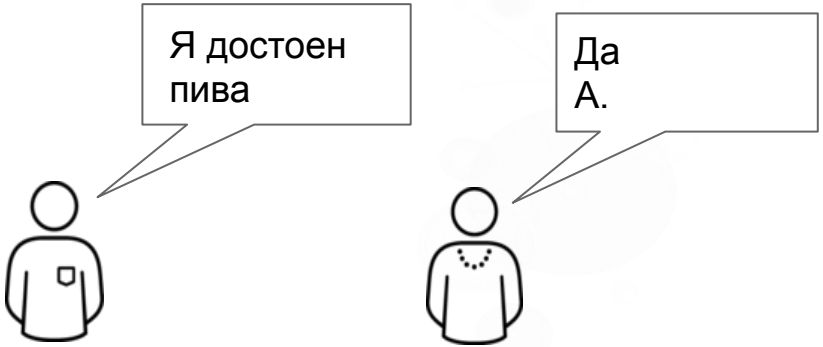
Прицпий 3 - ДУМАЙ ДЕКЛАРАТИВНО

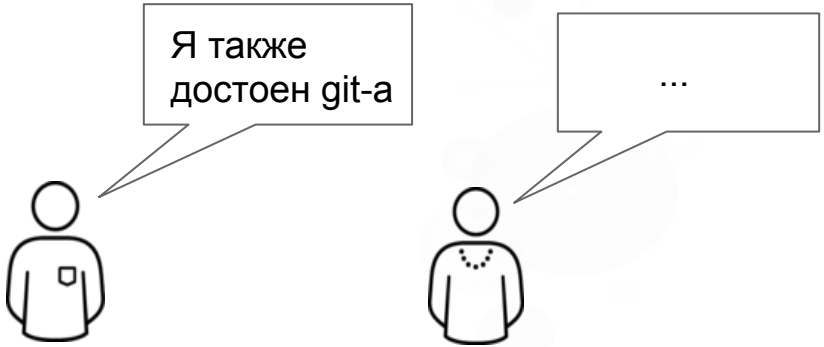
Ansible - это машина желаемых состояний.

Если вы пытаетесь «написать код» в своих плейбуках и ролях, вы настраиваете себя на неудачу. Наши плейбуке на основе YAML никогда не предназначались для программирования.

Относитесь к вашему контенту Ansible как к коду

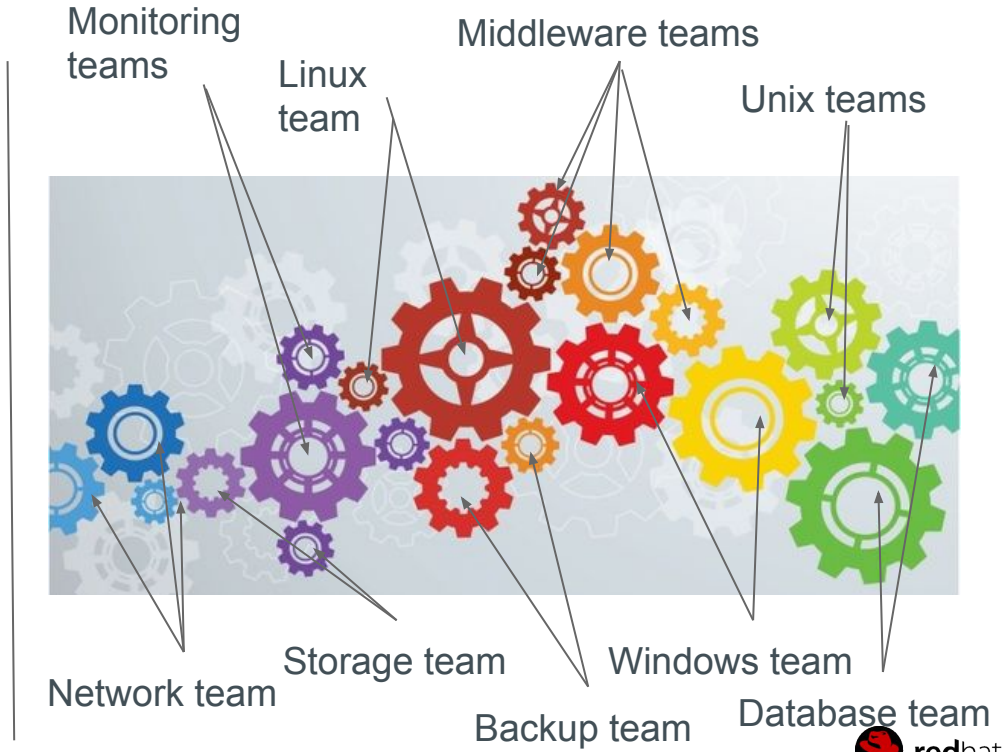
- Контроль версий вашего Ansible контента
- Начните как можно проще и делайте итерации
  - Начните с простого плейбука и статического “инвентаря”
  - Рефакторинг и модульность позже





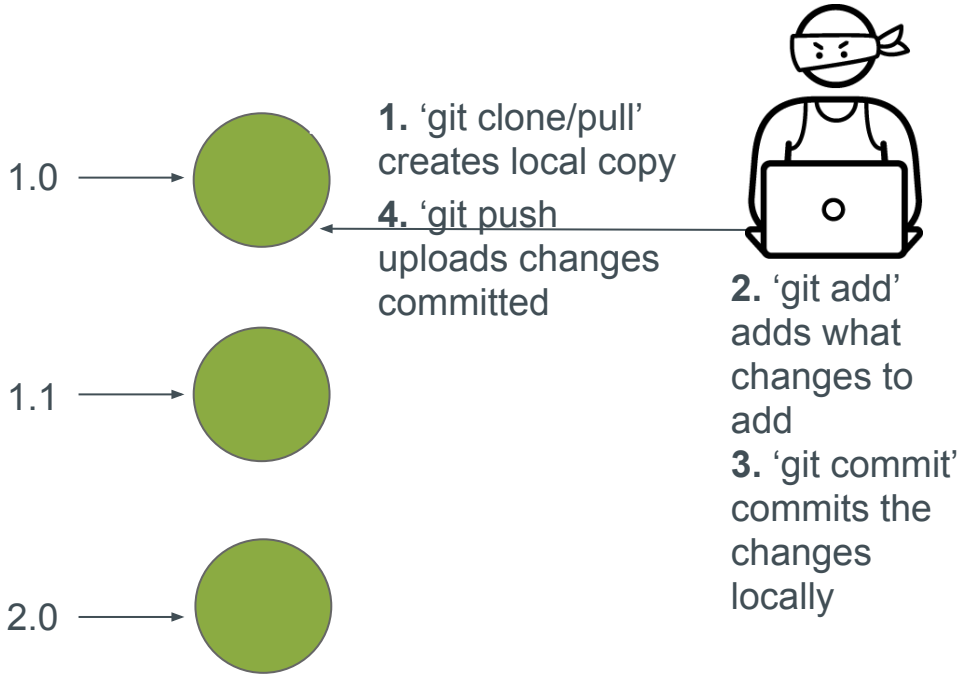
Относитесь к вашему контенту Ansible как к коду

1. **Ansible** не требует контроля версий
2. **Когда** вы масштабируете использование Ansible (для полной автоматизации) между собой будет общаться множество команд
3. **Контроль версий** был введен для решения проблем взаимодействия
4. **Git** заслужил популярность во всем мире трудным путем и является ядром многих самых популярных в мире услуг и продуктов для совместной работы.



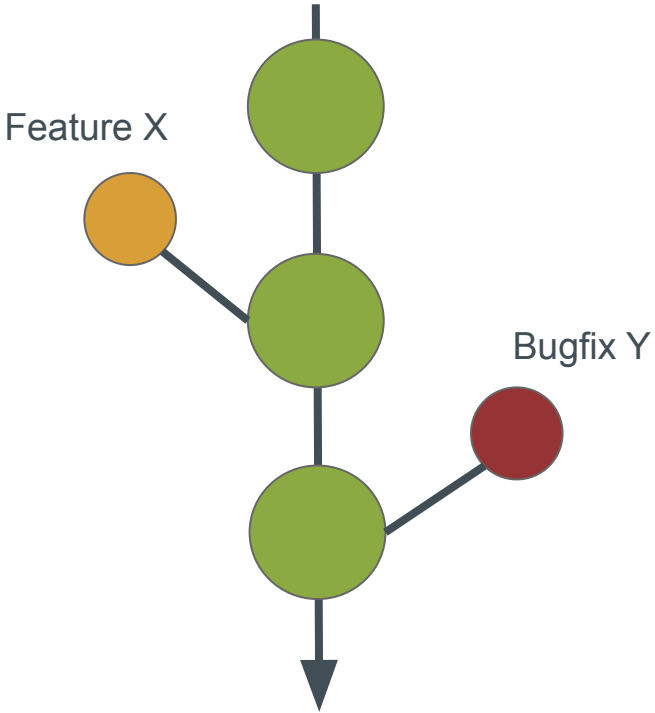
Пример: Контроль версий

1. Репозиторий **git** хранит файлы
2. Осуществляется **контроль** доступа
3. **Все** изменения во всех файлах отслеживаются
4. **Когда** вы хотите внести изменения в файл, вы сначала делаете локальную копию репозитория, затем меняете файл локально, фиксируете изменение, а затем продолжаете и говорите git, чтобы скопировать это локальное изменение в репозиторий.

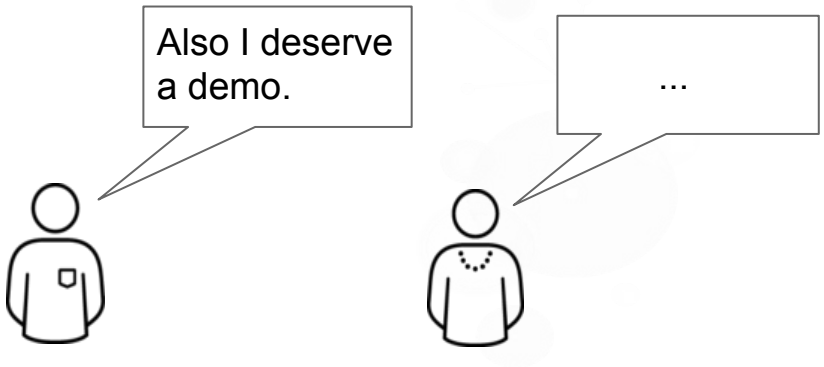


Пример: Git workflow

1. **Простой** сценарий
2. **Master** ветка всегда может пойти в релиз
3. **Ветки** используются для разработки, тестирования и багфиксинга.
4. **Тестирование.** Без него мастер ветка может быть негодной для релизов



Treat your Ansible content like code



Будьте стильными

- Создайте гид по стилю артефактов
- Консистентность в:
  - Маркировке
  - Использовании пробелов
  - Именование задач, переменных, ролей
  - Структуре папао
- Применяйте и проверяйте
- Попробуйте ansible-lint

Реализуйте процесс тестирования

Тестирование обычно включает

- Проверкку синтакса
  a. ansible-playbook --syntax-check your-playbook.yml
- Проверку стиль на “плохие практики” и поведения, которые потенциально можно улучшить
  a. ansible-lint your-playbook.yml
- Запуск плейбука или роли и проверка, что она завершена без сбоев (кэп!)
- Запуск плейбука или роли снова и проверка, что не появилось новых изменений (идемпотентность, ключевая фишка Ansible)
- Запросите API вашего приложения или проведите другой внешний тест на его функциональность
- Включите тестирование в CI/CD пайплайн для ваших плэйбуков

Read more

https://github.com/mglantz/ansible-roadshow/tree/master/labs/lab-9

МАКЕТ ПРОЕКТА: БАЗОВЫЙ


```
basic-project
├── inventory
│   ├── group_vars
│   │   └── web.yml
│   ├── host_vars
│   │   └── db1.yml
│   └── hosts
└── site.yml
```

МАКЕТ ПРОЕКТА: ОРГАНИЗАЦИЯ ПО РОЛЯМ


```
myapp
├── roles
│   ├── myapp
│   │   ├── tasks
│   │   │   └── main.yml
│   │   └── ...
│   ├── nginx
│   │   └── ...
│   └── proxy
│       └── ...
└── site.yml
```

МАКЕТ ПРОЕКТА: РАЗДЕЛЯЕМЫЕ РОЛИ


```
myapp
├── config.yml
├── provision.yml
├── roles
│   └── requirements.yml
└── site.yml
```

Давайте разумные имена вашему “инвентарю”

EXHIBIT A

```
10.1.2.75
10.1.5.45
10.1.4.5
10.1.0.40

w14301.example.com
w17802.example.com
w19203.example.com
w19304.example.com
```

EXHIBIT B

```
db1   ansible_host=10.1.2.75
db2   ansible_host=10.1.5.45
db3   ansible_host=10.1.4.5
db4   ansible_host=10.1.0.40

web1  ansible_host=w14301.example.com
web2  ansible_host=w17802.example.com
web3  ansible_host=w19203.example.com
web4  ansible_host=w19203.example.com
```

Группируйте хосты для более легкого выбора инвентаря и других условных задач - чем больше групп, тем лучше.

ЧТО

```
[db]
db[1:4]

[web]
web[1:4]
```

```
db1 = db, east, dev
```

ГДЕ

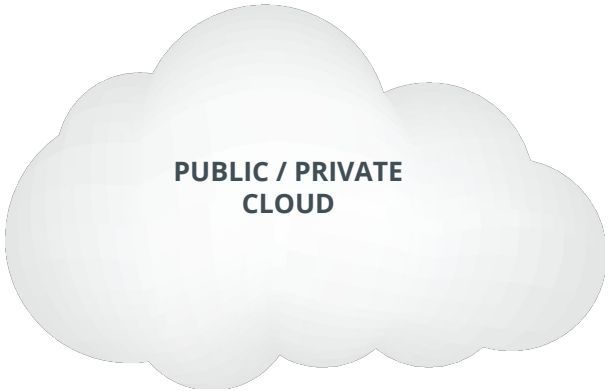
```
[east]
db1
web1
db3
web3

[west]
db2
web2
db4
web4
```

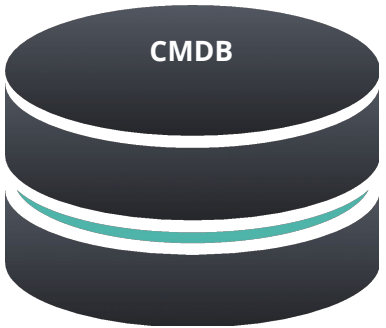
КОГДА

```
[dev]
db1
web1

[test]
db3
web3

[prod]
db2
web2
db4
web4
```

Используйте один источник истины (если он у вас есть). Даже если у вас несколько источников, Ansible может объединить их.

- Автоматическая синхронизация
- Снижение числа человеческих ошибок





Правильное именование переменных может сделать плейбуки более читабельными и избежать конфликтов имен переменных

- Используйте описательные, уникальные имена переменных (понятные человеку)
- Пользуйтесь префиксами переменных ролей, указывая “владельца” (например, имя роли или пакет)

```
apache_max_keepalive: 25
apache_port: 80
tomcat_port: 8080
```

ОТДЕЛИТЕ ЛОГИКУ ОТ ПЕРЕМЕННЫХ


```
- name: Clone student lesson app for a user
  host: nodes
  tasks:
    - name: Create ssh dir
      file:
        state: directory
        path: /home/{{ username }}/.ssh

    - name: Set Deployment Key
      copy:
        src: files/deploy_key
        dest: /home/{{ username }}/.ssh/id_rsa

    - name: Clone repo
      git:
        accept_hostkey: yes
        clone: yes
        dest: /home/{{ username }}/exampleapp
        key_file: /home/{{ username }}/.ssh/id_rsa
        repo: git@github.com:example/apprepo.git
```

EXHIBIT A

- Значения параметров и повторяющийся шаблон домашнего каталога в нескольких местах
- Работает, но можно сделать более четким и ремонтопригодным

ОТДЕЛИТЕ ЛОГИКУ ОТ ПЕРЕМЕННЫХ


```
- name: Clone student lesson app for a user
  host: nodes
  vars:
    user_home_dir: /home/{{ username }}
    user_ssh_dir: "{{ user_home_dir }}/.ssh"
    deploy_key: "{{ user_ssh_dir }}/id_rsa"
    app_dir: "{{ user_home_dir }}/exampleapp"
  tasks:
    - name: Create ssh dir
      file:
        state: directory
        path: "{{ user_ssh_dir }}"

    - name: Set Deployment Key
      copy:
        src: files/deploy_key
        dest: "{{ deploy_key }}"

    - name: Clone repo
      git:
        dest: "{{ app_dir }}"
        key_file: "{{ deploy_key }}"
        repo: git@github.com:example/exampleapp.git
        accept_hostkey: yes
        clone: yes
```

EXHIBIT B

- Значения параметров устанавливаются через значения, удаленные от задачи, и могут быть переопределены.
- Простые для понимания переменные "документируют" параметры задачи
- Проще перевести в роль

Мы используем синтаксис YAML для того, чтобы сделать читабельность кода максимальной

- Вертикальное чтение проще
- Поддержка сложных значений параметров
- Лучше работа с редакторами, которые подсвечивают синтаксис

ИСПОЛЬЗУЙТЕ НАТИВНЫЙ СИНТАКСИС YAML

NO!

```
- name: install telegraf
  yum: name=telegraf-{{ telegraf_version }} state=present update_cache=yes disab
  notify: restart telegraf

- name: configure telegraf
  template: src=telegraf.conf.j2 dest=/etc/telegraf/telegraf.conf

- name: start telegraf
  service: name=telegraf state=started enabled=yes
```

Better, but no

```
- name: install telegraf
  yum: >
      name=telegraf-{{ telegraf_version }}
      state=present
      update_cache=yes
      disable_gpg_check=yes
      enablerepo=telegraf
  notify: restart telegraf

- name: configure telegraf
  template: src=telegraf.conf.j2 dest=/etc/telegraf/telegraf.conf

- name: start telegraf
  service: name=telegraf state=started enabled=yes
```

ИСПОЛЬЗУЙТЕ НАТИВНЫЙ СИНТАКСИС YAML

Yes!

```
- name: install telegraf
  yum:
    name: telegraf-{{ telegraf_version }}
    state: present
    update_cache: yes
    disable_gpg_check: yes
    enablerepo: telegraf
  notify: restart telegraf

- name: configure telegraf
  template:
    src: telegraf.conf.j2
    dest: /etc/telegraf/telegraf.conf
  notify: restart telegraf

- name: start telegraf
  service:
    name: telegraf
    state: started
    enabled: yes
```

Имена улучшают читаемость и опыт пользователя

- Дайте всем вашим плейбукам, задачам и блокам краткие, достаточно уникальные и значимые для человека имена

EXHIBIT A

```
- hosts: web
  tasks:
  - yum:
      name: httpd
      state: latest


  - service:
      name: httpd
      state: started
      enabled: yes
```

```
PLAY [web]
*********************************

TASK [setup]
*********************************
ok: [web1]

TASK [yum]
*********************************
ok: [web1]

TASK [service]
*********************************
ok: [web1]
```

EXHIBIT B

```
- hosts: web
  name: install and start apache
  tasks:
    - name: install apache packages
      yum:
        name: httpd
        state: latest

    - name: start apache service
      service:
        name: httpd
        state: started
        enabled: yes
```

```
PLAY [install and start apache]
*********************************

TASK [setup]
*********************************
ok: [web1]

TASK [install apache packages]
*********************************
ok: [web1]

TASK [start apache service]
*********************************
ok: [web1]
```

Фокус позволяет избежать сложности

- Пусть ваши плэйбуки будут сфокусированными. Множество простых плейбуков проще, чем один гигантский плейбук со множеством условий
- Следуйте принципу Linux: делать одну вещь и делать ее хорошо

Очищайте экран от отладочной информации

- Используйте параметр verbosity, чтобы показывать информацию, когда она нужна.

```
- debug:
    msg: "This always displays"

- debug:
    msg: "This only displays with ansible-playbook -vv+"
    verbosity: 2
```

Не просто стартуйте сервисы, используйте “детекторы дыма”

```
- name: check for proper response
  uri:
    url: http://localhost/myapp
    return_content: yes
  register: result
  until: '"Hello World" in result.content'
  retries: 10
  delay: 1
```

Используйте командные модули.. экономно

- Используйте модули запуска, такие как *`shell`* или *`command`* как последнее прибежище
- Модуль *`command`* в целом более безопасны
- Модуль *`shell`* используйте для перенаправления ввода-вывода

Сначала найдите модуль

NO!

```
- name: add user
  command: useradd appuser

- name: install apache
  command: yum -y install httpd

- name: start apache
  shell: |
    systemctl start httpd && systemctl enable httpd
```

Yes :-)

```
- name: add user
  user:
    name: appuser
    state: present

- name: install apache
  yum:
    name: httpd
    state: latest

- name: start apache
  service:
    name: httpd
    state: started
    enabled: yes
```

Если все-таки приходится использовать командный модуль?

```
- hosts: all
  vars:
    cert_store: /etc/mycerts
    cert_name: my cert
  tasks:
  - name: check cert
    shell: certify --list --name={{ cert_name }} --cert_store={{ cert_store }} | grep "{{ cert_name }}"
    register: output

  - name: create cert
    command: certify --create --user=chris --name={{ cert_name }} --cert_store={{ cert_store }}
    when: output.stdout.find(cert_name)" != -1
    register: output

  - name: sign cert
    command: certify --sign  --name={{ cert_name }} --cert_store={{ cert_store }}
    when: output.stdout.find("created")" != -1
```

Создайте свой собственный модуль

```
- hosts: all
  vars:
    cert_store: /etc/mycerts
    cert_name: my cert
  tasks:
    - name: create and sign cert
      certify:
        state: present
        sign: yes
        user: chris
        name: "{{ cert_name }}"
        cert_store: "{{ cert_store }}"
```

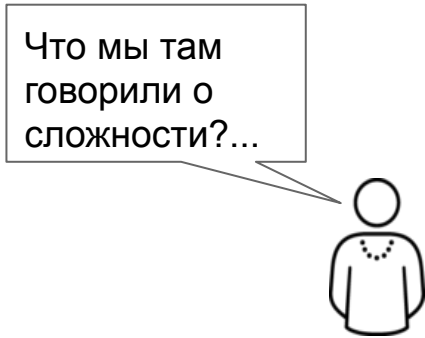
- Понятен и не технарям
- CRUD (Create, read, update and delete)

Отделите подготовку инфраструктуры от задач развертывания и настройки

```
acme_corp/
├── configure.yml
├── provision.yml
└── site.yml

$ cat site.yml
---
- import_playbook: provision.yml
- import_playbook: configure.yml
```

Jinja2 является мощным, но вам не нужно использовать всю его мощь

- Шаблоны должны быть простыми:
  - Подстановка переменных
  - Условия
  - Простые структуры управления/итерации
  - Создавайте шаблоны для ваших проблем, а не спасения мира
- Чего избегать:
  - Всего, что можно сделать напрямую в Ansible
  - Управления переменными в шаблоне
  - Обширные и сложные условия
  - Логику, построенную на встроенных именах хостов
  - Сложные вложенные циклы



Будьте осторожны при смешивании ручной и автоматической конфигурации (Или даже различные движки автоматизации...)

- Помечайте файлы, сгенерированные шаблоном, как сгенерированные Ansible

```
{{ ansible_managed | comment }}
```

Помните

- Пусть роли (как и плейбуки) фокусируются на цели и функции
- Используйте папку `roles/` для ролей в рамках одного проекта для организационной прозрачности
- Следуйте паттернам Ansible Galaxy для ролей, которые будут использоваться в нескольких проектах
- Ограничивайте зависимости ролей

Трюки и советы

- Используйте `ansible-galaxy init` для старта ваших ролей
- ...затем удалите ненужные директории и файлы-заглушки
- Используйте `ansible-galaxy` для установки ролей, в том числе и для частных
- Используйте файлы ролей (напр. `requirements.yml`) для указания внешних ролей, которые использует проект
- Всегда привязывайте роль к определенной версии, такой как тег или коммит

Средства командной строки имеют свои ограничения

- Координация между распределенными командами и организациями…
- Управление доступом к учетным данным…
- Отслеживание, аудит и автоматизация отчетности и управления…
- Самообслуживание или делегирование…
- Интеграция автоматизации с корпоративными системами…

Спасибо

Сложность убивает продуктивность
Оптимизируйте для легкости чтения
Думайте декларативно