Ansible basics

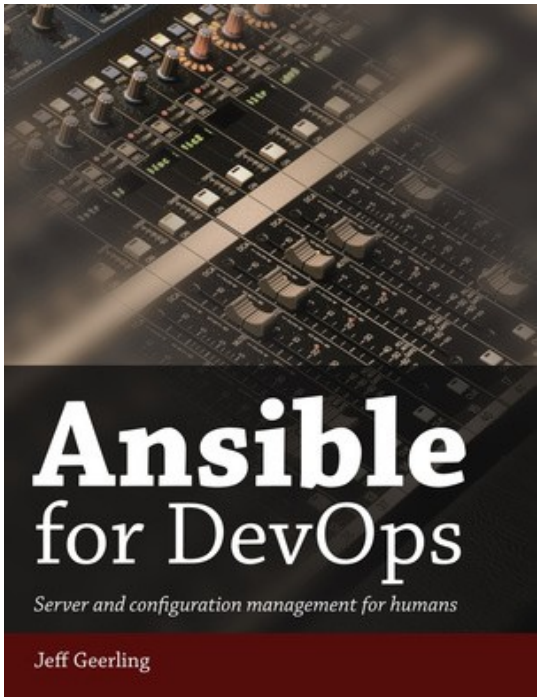
10.6.2016

Martin Milata <mmilata@srck.net>

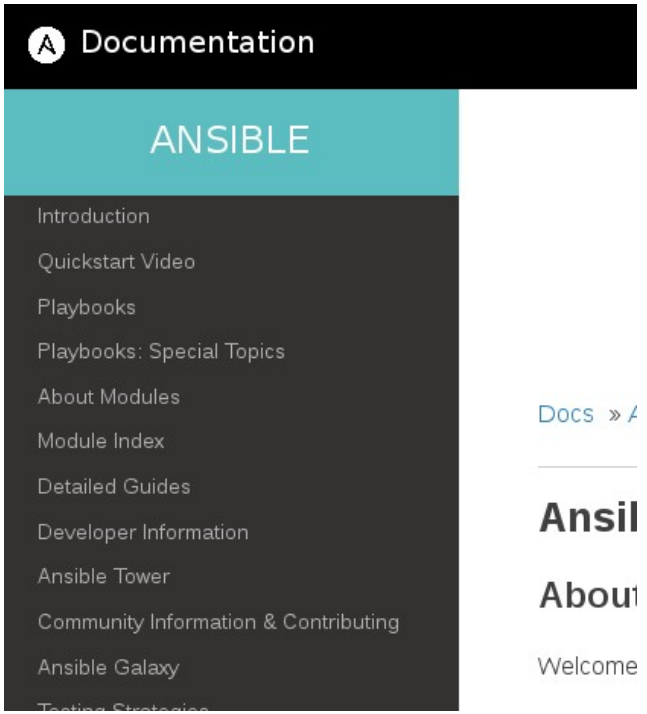
Intro

- Who am I?
- Who develops Ansible?
- Hands-on exercises
- Ask questions!

Sources



https://leanpub.com/ansible-for-devops



https://docs.ansible.com/ansible/

What is it?

- Configuration management tool
- Set up machine based on a text description (not really a programming language)
- Radically simple
- Push model
- Connects through SSH, only requires Python

Similar software

- Puppet
- Chef
- Salt
- CFEngine
- Fabric/capistrano

Why use it?

- Advantages over configuring servers manually?
- Shell scripts?
- Storing infrastructure configuration in version control (git)
- Reusing someone else's configuration
- Development & testing on local VMs

Hands-on 1

- Try logging in to your VM
- Make sure python is installed
- Run ssh-keygen if needed
- ssh-copy-id -p X0022 user@1.2.3.4
- Install ansible package on your host

Hands-on 1: inventory file

Edit /etc/ansible/hosts to contain single line:

```
1.2.3.4 ansible_ssh_user=user ansible_ssh_port=X0022 ansible_sudo=true
```

Hands-on 1: ad-hoc commands

- `ansible all -m ping`
- `ansible all -m shell -a “date”`
- `ansible all -m user -a “name=guest state=present shell=/bin/zsh”`
- `ansible all -m user -a “name=guest state=absent”`
- `ansible all -m apt -a “name=ntp state=present”`
- `ansible all -m service -a “name=ntp state=started enabled=yes”`
- `ansible all -m setup -v | less`

Ansible modules

- http://docs.ansible.com/ansible/list_of_all_modules.html
- Quick reference: `ansible-doc <module-name>`
- Interesting modules: copy, cron, file, git, group, hostname, htpasswd, ini_file, lineinfile, mount, postgresql_user, sysctl, wait_for …
- Modules for running arbitrary commands: `command`, `shell`
- You can write your own

Playbooks

- Playbooks contain list of ansible module calls (“tasks”)
- And more
- Ansible uses YAML for all its files

YAML - scalars

- Strings: Hi!, “Hello”
- Numbers: 42, 0.1337
- Booleans: true, no, off

YAML – lists and records

```
- Apple
- Orange
- Strawberry
- Mango
```

```
name: John Doe
job: Sysadmin
city: Brno
```

YAML - example

```
---
# An employee record
name: Example Developer
job: Developer
city: Brno
employed: True
foods:
    - Apple
    - Orange
    - Strawberry
    - Mango
languages:
    ruby: Expert
    python: Expert
    dotnet: Beginner
```

Hands-on 2

- Please clone example playbook from github:
- `git clone https://github.com/mmilata/ansible-workshop`
- `cd example1`

- To run it:
- `ansible-playbook playbook.yml -v`

Idempotence

- = Doing something multiple times has the same result as doing it only once
- Playbooks should be idempotent
- Ansible modules are idempotent (except command and shell)
- Check mode

Playbooks - variables

- Command line
- Defined in playbooks
- group_vars/host_vars
- Facts (incl. set_fact)
- Role defaults

Playbooks - loops

```
- name: add several users
  user: name={{ item }} state=present groups=wheel
  with_items:
    - testuser1
    - testuser2

- name: add several users
  user: name={{ item }} state=present groups=wheel
  with_items: “{{ user_list }}”

- name: add several users
  user: name={{ item.name }} state=present groups={{ item.groups }}
  with_items:
    - name: testuser1
      groups: wheel
    - name: testuser2
      groups: root

- name: add several users
  user: name={{ item.key }} state=present groups={{ item.value }}
  with_dict:
    testuser1: wheel
    testuser2: root

- copy: src={{ item }} dest=/etc/fooapp/ owner=root mode=600
  with_fileglob:
    - /data/files/fooapp/*
```

Playbooks - conditions

```
- copy: src=x dest=y

  when: something is defined

...

  when: something >= 5

...

  when: (something in [3,4,5]) and
        otherthing is defined
```

```
- name: list contents of directory
  command: ls mydir
  register: contents

- name: check contents for emptiness
  debug: msg="Directory is empty"
  when: contents.stdout == ""
```

Organizing playbooks

- Includes

```
tasks:
- include: database.yml

- include: kerberos.yml
  when: setup_kerberos is defined

- include: add_user.yml user=alice
- include: add_user.yml user=bob
```

- Roles

```
roles:
- basic_setup
- postfix
- nagios_client
- some_app
```

Roles

- Reusable pieces of ansible code
- Self-contained
- Customizable by using variables

```
role_name/
  defaults/
    main.yml
  files/
    foo.conf
    ...
  handlers/
    main.yml
  meta/
    main.yml
  tasks/
    main.yml
  templates/
    bar.cfg.j2
    ...
  vars/
    main.yml
```

Hands-on 3

- `cd ansible-workshop/example2`
- `ansible-playbook site.yml -v`

```
group_vars/
  all
roles/
  apache/
    defaults/
    handlers/
    tasks/
    templates/
  mariadb/
    handlers/
    tasks/
  mediawiki/
    defaults/
    tasks/
site.yml
```

Ansible Galaxy

- https://galaxy.ansible.com/
- `ansible-galaxy` command
- Can be used with private git repositories too
- Always review third-party roles!
- http://debops.org/

```
$ cat rolesfile.yml
 - src: yatesr.timezone

 - src: https://github.com/mmilata/ansible-role-apache
   version: v1.2
   name: apache
$ ansible-galaxy install -r requirements.yml -p roles/
```

Templates

- http://jinja.pocoo.org/docs/dev/templates/

```
{% if allow_subnet is defined %}
Require ip {{ allow_subnet }}
{% else %}
Require local
{% endif %}
```

```
{% for repo in yum_repos %}
[{{ repo.name }}]
baseurl = {{ repo.url }}
enabled = {{ repo.enabled }}
{% endfor %}
```

```
secrethash = “{{ 'test1'| hash('sha1') }}”
enabled = {{ foo_enabled | default(“no”) }}
```

Inventory

```
[app-brq]
app1.brq.example.org
app2.brq.example.org

[app-bts]
app1.bts.example.org

[db-brq]
db.brq.example.org

[db-bts]
db.bts.example.org

[app:children]
app-brq
app-bts

[db:children]
db-brq
db-bts
```

```
site.yml
appservers.yml
dbservers.yml
inventory
roles/
  ...
group_vars/
  all
  app
  app-bts
  db
host_vars/
  app1.brq.example.org
```

Thank you!
Questions?