



# ANSIBLE

“Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.”

Ansible Documentation

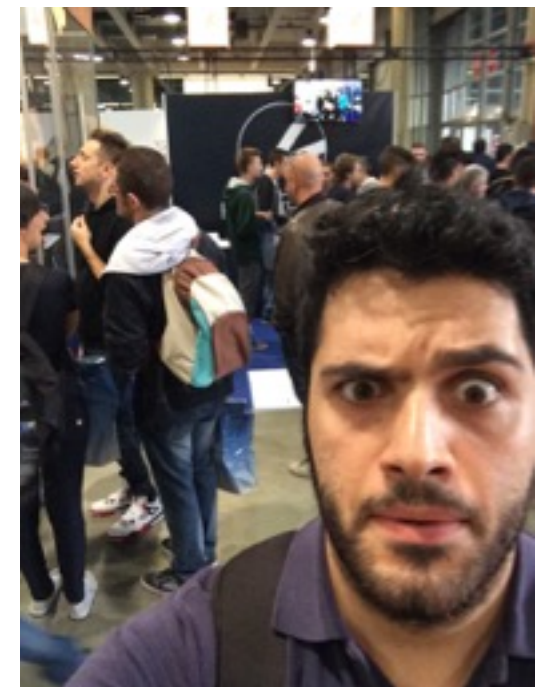
# WHOAMI

Giovanni Albero

Developer Jr.

 @albero92

  
**fazland**



# Overview

# Example

## CASE

I need an environment to develop a web application

## SOLUTION (probably)

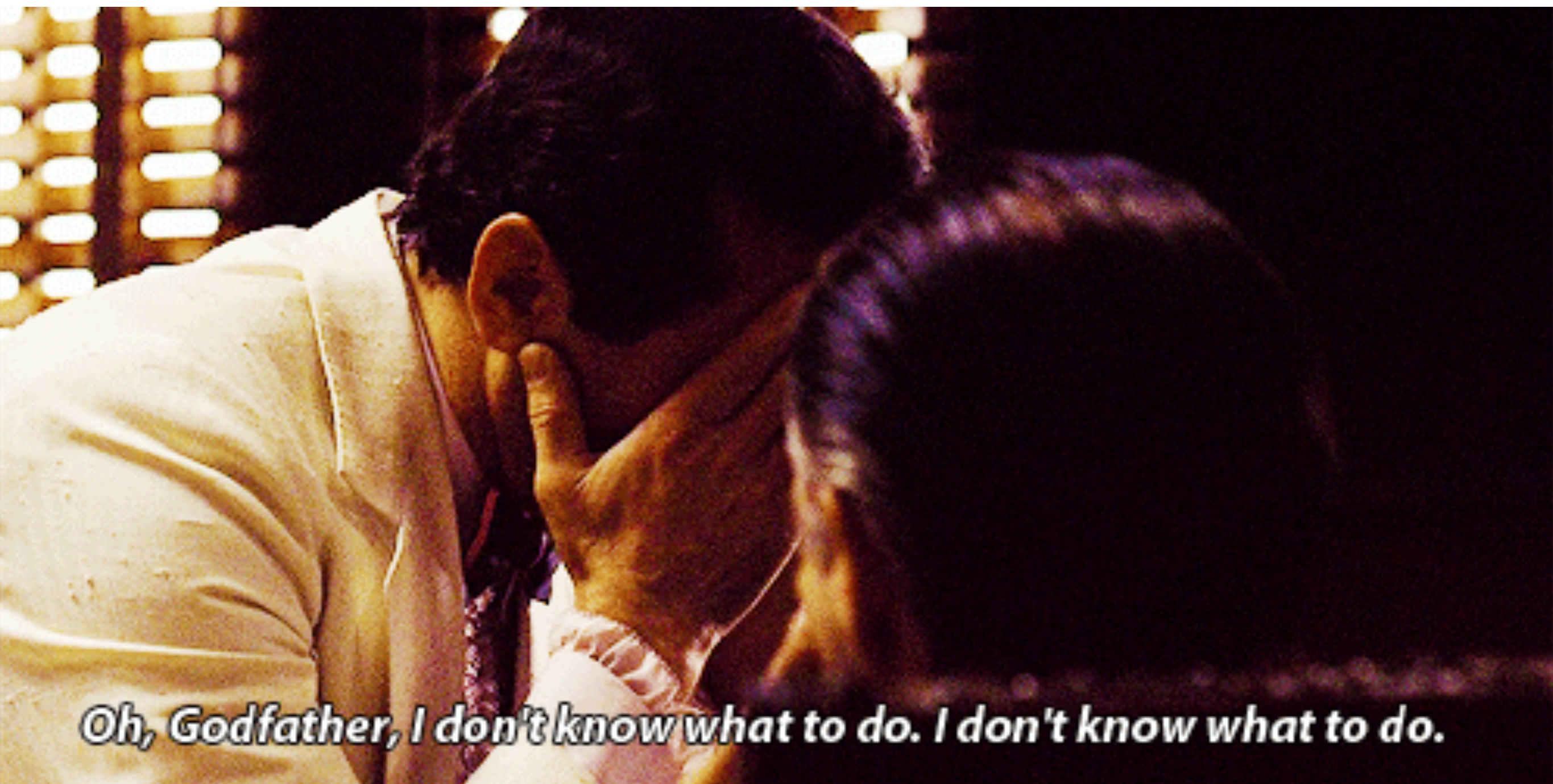
I'll create a Virtual Machine (VM), when It's ready, I'll first install nginx, then php-fpm, MySQL and so on.

...I start to work but shortly after... I've broken the VM!!!!

I have to start over, and I'll spend a lot of time to configure a machine and I don't know how many times I'll have to do it again.

But what can I do to automate this process?





*Oh, Godfather, I don't know what to do. I don't know what to do.*



A pixelated, low-resolution image of a nuclear mushroom cloud. The cloud is composed of white and light blue pixels, with a bright orange and red fireball at its base. The word "RELAX" is written in large, white, bold, sans-serif capital letters across the center of the image, superimposed over the dark, smoky part of the mushroom cloud.

**RELAX**



# There is Ansible!

what do I need?

a computer (with Unix distributions or OS X) of course!

with  python™ 2.6 or 2.7 installed

If you don't have pip installed, install it

```
$ sudo easy_install pip
```

Ansible uses the following Python modules

```
$ sudo pip install paramiko PyYAML Jinja2 httpplib2
```

And now it's time to install Ansible

```
$ sudo pip install ansible
```

Now in your mind there is a question, why is windows not in the list of OS supported?

Because you can install a Unix distribution on your pc, with a lot of benefits for your mental health :)



Used by

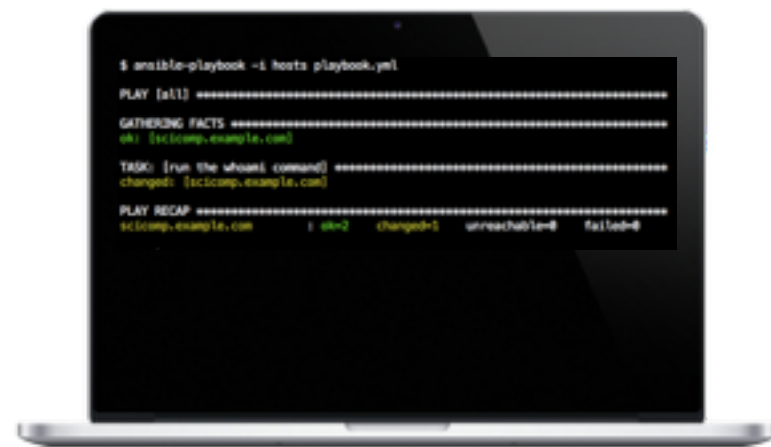


# Now we understand how Ansible works

## Ansible communicates with remote machines over SSH

From Ansible 1.3 and later (Now 1.9) it will try to use OpenSSH when possible, this enables **ControlPersist**.

There are enterprise Linux & OS (Red Hat Enterprise Linux and derivatives such as CentOS) where the version of OpenSSH may be too old to support ControlPersist, but no problem Ansible will fallback into using implementation of OpenSSH called **paramiko**. For OS X, Ubuntu, Fedora there isn't this problem.



OpenSSH



<https://developer.rackspace.com/blog/speeding-up-ssh-session-creation/>

# Structure



# Inventory

hosts is an INI-like

```
[webservers]  
foo.example.com  
bar.example.com
```

```
[dbservers]  
one.example.com  
two.example.com  
three.example.com
```

The things in brackets are group names, which are used in classifying systems and deciding what systems you are controlling at what times and for what purpose.

If you have hosts that run on non-standard SSH ports you can put the port number after the hostname

```
badwolf.example.com:5309
```

# Playbook

yaml file

---

```
- hosts: all
  sudo: yes
  tasks:
    - name: Update apt-cache
      apt: update_cache=yes
    - name: Install htop
      apt: pkg=htop state=present
```

# Variables

---

```
- hosts: all
  sudo: yes
  vars:
    package_1: curl
  tasks:
    - name: Update apt-cache
      apt: update_cache=yes
    - name: "Install {{ package_1 }}"
      apt: pkg="{{ package_1 }}" state=present
```



# Loops

---

```
- hosts: all
  sudo: yes
  tasks:
    - name: Update apt-cache
      apt: update_cache=yes
    - name: "Install {{ item }}"
      apt: pkg="{{ item }}" state=present
      with_items:
        - htop
        - curl
        - unzip
```

# Conditionals

---

```
- hosts: all
  sudo: yes
  tasks:
    - name: copy authorized_keys
      copy:
        src: authorized_keys
        dest: /home/ubuntu/.ssh/authorized_keys
        owner: ubuntu
      when: environment == "Stage"
```

# Provisioner file

## site.yml

```
---
hosts: all
sudo: yes
tasks:
  - name: install nginx
    apt: name=nginx update_cache=yes
  - name: copy nginx config file
    copy: src=files/nginx.conf dest=/etc/nginx/sites-available/default
  - name: enable configuration
    file: >
      dest=/etc/nginx/sites-enabled/default
      src=/etc/nginx/sites-available/default
      state=link
  - name: copy index.html
    template: src=templates/index.html.j2 dest=/usr/share/nginx/html/index.html mode=0644
  - name: restart nginx
    service: name=nginx state=restarted
```





# DEMO 1

## ansible-playbook

# But If I had more tasks

```
---
- hosts: servers
  vars_files:
    - vars.yml
  gather_facts: false
  sudo: true

  tasks:
    - name: Create the project directory.
      file: state=directory path={{ project_root }}

    - name: Create user.
      user: home={{ project_root }}/home/ name={{ project_name }} state=present

    - name: Update the project directory.
      file: group={{ project_name }} owner={{ project_name }} mode=755 state=directory path={{ project_root }}

    - name: Create the code directory.
      file: group={{ project_name }} owner={{ project_name }} mode=755 state=directory path={{ project_root }}/code/

    - name: Install required system packages.
      apt: pkg={{ item }} state=installed update-cache=yes
      with_items: {{ system_packages }}

    - name: Install required Python packages.
      easy_install: name={{ item }}
      with_items: {{ python_packages }}

    - name: Mount code folder.
      mount: fstype=vboxsf opts=uid={{ project_name }},gid={{ project_name }} name={{ project_root }}/code/ src={{ project_name }} state=mounted
      only_if: "$vm == 1"

    - name: Create the SSH directory.
      file: state=directory path={{ project_root }}/home/.ssh/
      only_if: "$vm == 0"

    - name: Upload SSH known hosts.
      copy: src=known_hosts dest={{ project_root }}/home/.ssh/known_hosts mode=0600
      only_if: "$vm == 0"

    - name: Upload SSH key.
      copy: src=key dest={{ project_root }}/home/.ssh/id_rsa mode=0600
      only_if: "$vm == 0"

    - name: Create the SSL directory.
      file: state=directory path={{ project_root }}/home/ssl/

    - name: Upload SSL private key.
      copy: src=files/ssl/{{ project_name }}.pem dest={{ project_root }}/home/ssl/{{ project_name }}.pem

    - name: Upload SSH public key.
      copy: src=files/ssl/{{ project_name }}.key.encrypted dest={{ project_root }}/home/ssl/{{ project_name }}.key

    - name: Change permissions.
      shell: chown -R {{ project_name }}:{{ project_name }} {{ project_root }}

    - name: Install nginx configuration file.
      copy: src=files/conf/nginx.conf dest=/etc/nginx/sites-enabled/{{ project_name }}
      notify: restart nginx

    - name: Install init scripts.
      copy: src=files/init/{{ item }}.conf dest=/etc/init/{{ project_name }}_{{ item }}.conf
      with_items: {{ initfiles }}

    - name: Create database.
      shell: {{ project_root }}/env/bin/python {{ project_root }}/code/webapp/manage.py sqlcreate --router=default | sudo -u postgres psql

  handlers:
    - include: handlers.yml

- include: deploy.yml

- hosts: servers
  vars_files:
    - vars.yml
  gather_facts: false
  sudo: true

  tasks:
    - name: Restart services.
      service: name={{ project_name }}_{{ item }} state=restarted
      with_items: {{ initfiles }}
```

it's not readable and maintainable

# Group Vars & Host Vars

You can create a separate variable file for each host and each group

*group\_vars/dbservers*

```
db_primary_host: rhodeisland.example.com
db_replica_host: virginia.example.com
db_name: widget_production
db_user: widgetuser
db_password: pFmMxcyD;Fc6) 6
rabbitmq_host: pennsylvania.example.com
```

*host\_vars/foo.example.com*

```
http_port: 80
security_port: 2555
```



# Roles

- project organization tool
- reusable components

*roles/mongodb/tasks/main.yml*

Tasks

*roles/mongodb/files/*

Holds files to be uploaded to hosts

*roles/mongodb/templates/*

Holds Jinja2 template files

*roles/mongodb/vars/main.yml*

Variables that shouldn't be overridden

# Templates

```
server {  
    charset utf-8;  
  
    listen 80 default_server;  
    server_name {{ host }};  
    root {{ wp_root }};  
  
    client_max_body_size {{ client_max_body_size }};  
    ...
```

# Templates

## In playbook

- name: Upload configuration File  
template: src=default.j2 dest="/etc/nginx/sites-available/default"

# Provisioning

```
site.yml
hosts
group_vars/
    group1
    group2
host_vars/
    hostname1
here
    hostname2
roles/
    common/
        files/
        templates/
        tasks/
        vars/
    webservers/
        ...
        ...
    applicationservers/
        ...
        ...
    databaseservers/
        ...
        ...
```

```
#master playbook
#inventory

# here we assign variables to particular groups

# if systems need specific variables, put them

# this hierarchy represents a "role"
# <-- files for use with the copy resource
# <-- files for use with the template resource
# <-- tasks file can include smaller files
# <-- variables associated with this role
```



# DEMO 2

## ansible-playbook



And on web services?  
(such as AWS)

New host file called Dynamic Inventory

remember to export  
AWS\_ACCESS\_KEY\_ID  
AWS\_SECRET\_ACCESS\_KEY

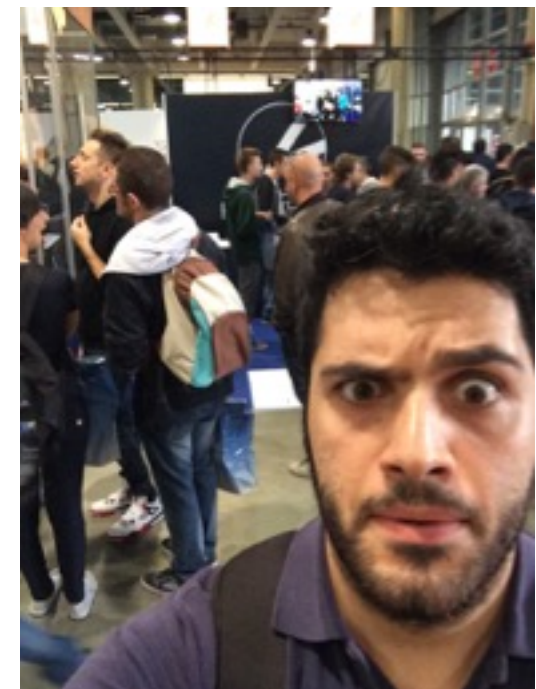
<https://raw.githubusercontent.com/ansible/ansible/devel/plugins/inventory/ec2.py>

# DEMO 3

## ansible-playbook



Thanks!



<https://github.com/giovannialbero1992/ansible>