Fake IT, until you make IT

Bas Meijer, GOTO Amsterdam 2015

Abstract

Before launching your new app, you would better be in control of your environments:

- develop & test in a production-like environment
- automate the whole enchilada with Ansible

Bassie will show how to set up a disposable development environment that mimics your production servers in a re-usable way with minimal maintenance.

Bas Meijer

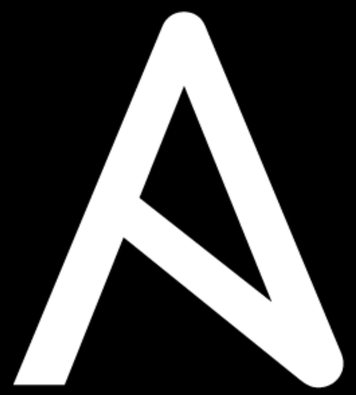
Bassie is a software developer & system engineer with decades wasted on late-night hacking.

While born before the epoch he has a keen eye for new technologies.

At the moment he is engaged with a major Dutch bank and an established European identity & access management cloud service.

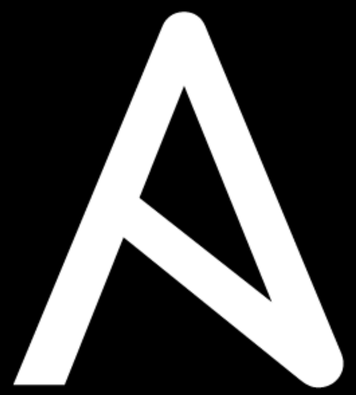
Fake IT, until you make IT

- Simple idea, but very powerful
- VM`s on laptop model production environment
- Reproducible workflows
- Automation with Ansible, Vagrant & Packer
- And yes, you can use Docker too



Why are we doing this?

- Delivery is painful
- Fear of the unknown
- Take out boring drudgery
- Kill your darlings
- Snowflakes are unique
- Humans bad at for loops
- run, Run! RUN!!

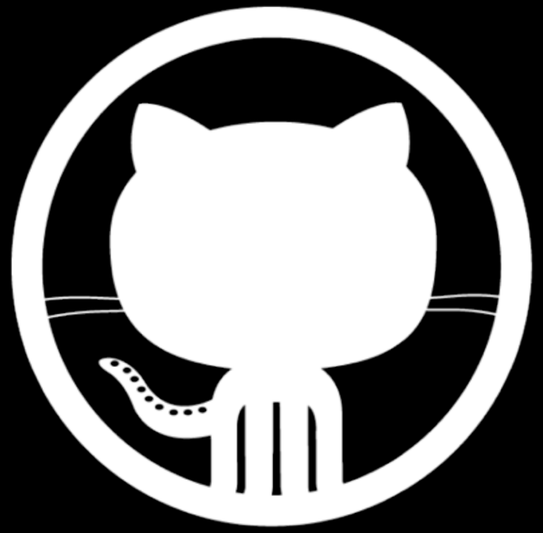


Ansible

- Easiest IT automation to use, ever.
- Minimal learning curve
- Easy audit/review/rewrite of content
- Minimal requirements: SSH & python
- No daemons, no master, no agents
- Secure, fast, scalable
- Pluggable and extensible

What do you need?

- 8Gb Ram or more, SSD
- SSH client, git client
- I use brew to get:
  - Vagrant
  - VirtualBox
  - Python
- Let`s get brewing!



Quickstart

```
git clone https://github.com/bbaassssiiee/vagransible
cd vagransible
vagrant up centos6
```

Vagrant is provider agnostic

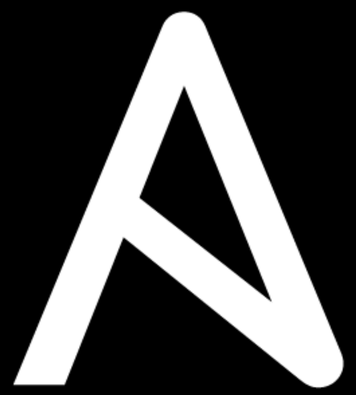
- VirtualBox
- VMWare
- Amazon Web Service
- Docker
- Microsoft Hyper-V
- …

Vagrant is provisioner agnostic

- Ansible
- Salt
- Puppet
- Chef
- bash
- …

Vagrant is what you need

- **vagrant up** - starts the machine,
  possibly downloading and caching
  the box image & provisioning the VM
- **vagrant ssh** - logs you into the VM
- **vagrant halt** - stops the VM
- **vagrant suspend** - pauses the VM
- **vagrant destroy** - trashes the VM



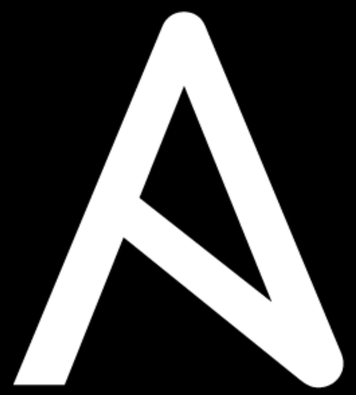
Vagrant up & running Ansible

environment

```
kreta:environment bas$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'chef/centos-6.5'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'chef/centos-6.5' is up to date...
==> default: Setting the name of the VM: PRODUCT-123
==> default: Fixed port collision for 22 => 2222. Now on port 2201.
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 => 2201 (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2201
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Warning: Connection timeout. Retrying...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Mounting shared folders...
    default: /vagrant => /Users/bas/code/environment
==> default: Running provisioner: ansible...
```

Simplest Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = “dockpack/centos6"
end
```



Provision VM instance with Ansible

```
config.vm.provision "ansible" do |ansible|

  ansible.inventory_path = "ansible.ini"
  ansible.playbook = "provision.yml"

  ansible.verbose = "vv"

end
```

Provider VirtualBox

```
config.vm.provider "virtualbox" do |vb|

  vb.gui = false
  vb.customize ["modifyvm",:id,"--memory",2048]

  vb.name = "centos6"

end
```

Free, runs on most laptops

Building your own box

packer build dockpack-centos.json

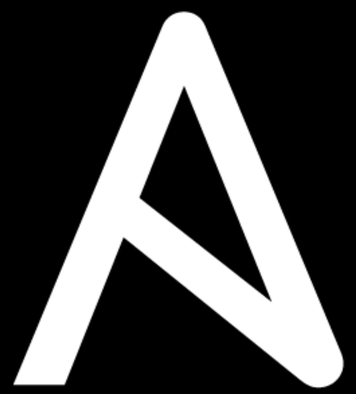
kickstart install for RedHat-like systems.

install ansible with a small shell script .

ansible does the rest in local mode.

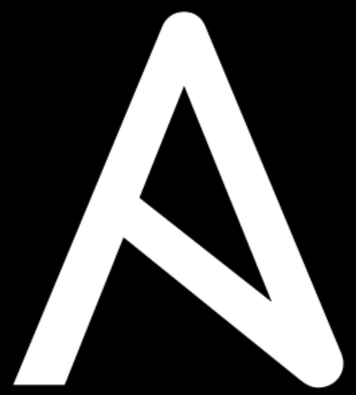
Packer creates VM images

- Builders: build a Box/image/AMI
- Providers: hypervisors for a guest VM
- Provisioners: install/configure/deploy
- All configured with a packer.json
- Utilizes RedHat kickstart.cfg



provisioning in packer json

```
"provisioners": [
  {
    "type": "shell",
    "execute_command": "echo 'vagrant' | {{.Vars}} sudo -S -E bash '{{.Path}}'",
    "override": {
      "virtualbox-iso": {
        "scripts": [
          "scripts/ansible.sh"
        ]
      }
    }
  },
  {
    "type": "ansible-local",
    "playbook_file": "packer.yml",
    "role_paths": [
      "roles/bbaassssiiee.commoncentos",
      "roles/RHEL6-STIG"
    ]
  }
```



Resources

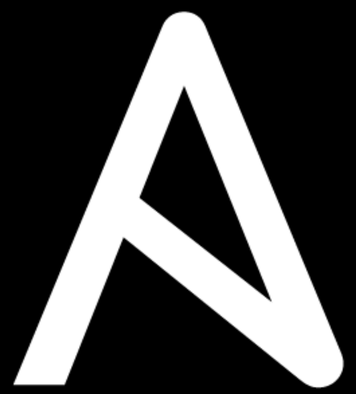
@bbaassssiiee

https://github.com/bbaassssiiee/vagransible

http://www.meetup.com/Ansible-Benelux

https://galaxy.ansible.com

http://www.vagrantup.com



ANSIBLE

GOTO Amsterdam

Bas Meijer @bbaassssiiee 2015-06-19 Amsterdam

Putting more Dev in DevOps

Working together as One team

@ikusalic

Initial state

Initial state

- Two teams

Initial state

- Two teams
- Tooling problems

Transition initiative

- One team

Transition initiative

- One team
- Rewrite?

Transition initiative

- One team
- Rewrite?
- Leveraging existing expertise

Desired tooling

- Preserve existing functionality

Desired tooling

- Preserve existing functionality
- Stateless client-server

Desired tooling

- Preserve existing functionality
- Stateless client-server
- Scala

Outcome

- One stronger team

Outcome

- One stronger team
- Knowledge transfer in both directions

Outcome

- One stronger team
- Knowledge transfer in both directions
- Rewrite was a success

Thank you

@ikusalic