

# Introduction to Ansible

Matt Behrens · <https://www.zigg.com/> · @zigg

DevOps West Michigan · July 14, 2014

# Outline

- Introduction
- Real examples
  - One-command development target
  - Physical test boxes
    - Building the same box locally and in the cloud
- Discussion

# Introduction

# What is Ansible?

- A “radically simple IT automation platform”
- Describe the intended system state using playbooks written in [YAML](#)
- Requires no agent on the managed machine, only a Python interpreter and an SSH server

# Installation

- `brew install ansible`
- Or [clone devel from GitHub](#)
  - Doesn't work in [virtualenv](#)\*; source `hacking/env-setup` to run out of your clone
  - Or, if you're crazy like me, maybe see [PR #6690](#)
- `brew install cowsay`

\* Ansible does more than virtualenv realistically supports, so this is not Ansible's issue

# Ad-hoc configuration

```
$ cat >hosts
[cowsay-servers]
hostname.example.com
^D
$ ansible cowsay-servers -i hosts -m yum \
-a 'name=cowsay state=present'
[[ ANSIBLING INTENSIFIES ]]
$ ssh hostname.example.com cowsay moo
```

```
< moo >
-----
```

```
 \      ^  ^
  (oo)\_____
  ( )\         )\/\
   ||-----w  ||
   ||           ||
```

# Playbooks

- Consolidate tasks and configuration knobs into a single YAML file
- Support templating—both in playbooks and in [file templates](#)—by way of Jinja2
- Can be one file, or split up into many roles (more later!)

# CAAP

(cowsay as a playbook)

---

```
- hosts: cowsay-servers
```

```
sudo: yes
```

```
tasks:
```

```
# install the cowsay package to
```

```
# provide cow-saying services
```

```
- name: install cowsay
```

```
  yum: name=cowsay state=present
```



# Running a playbook

- `ansible-playbook playbook.yml`
- `-i` to specify a local inventory file (e.g. `hosts`)
- `-k` if you have to enter a password for this run
- Use `authorized_key` to install an SSH key for future runs

# Real examples

(cows not included)

One-command  
development target

# What I'm doing

- [octothorpe](#) requires a Linux VM for development, as Asterisk *really* isn't too fond of OS X
- I don't want to maintain and publish an actual VM of my own
- Given a CentOS box and Digium's RPM repository, build a box on `vagrant up`

# How I did it

- Kept in [etc/playbook.yml](#)
- Uses [Yum repository bootstrapping](#) for EPEL, Digium repositories
- [Links /etc/asterisk](#) to work directory's etc/asterisk
- [Sets up Asterisk to restart](#) (and thus load its config) when /vagrant is mounted
- [Linked into Vagrantfile](#)

Physical test boxes

# What I'm doing

- Capacity and physical telephony testing usually requires us to build out a box from our stack of hardware
- Don't want to put disks in and hand-hold a basic Asterisk installation every time
- PXE installation option to install standard CentOS
- Run Ansible playbook to configure Asterisk and pre-install some Python developer tools

# How I did it

- Break down the job into several roles
- After installing a box, edit the local inventory to reference the boxes we want to set up
- Run the playbook on all boxes simultaneously
- Start plugging in the wires and running tests



Same box local and in  
the cloud

# What I'm doing

- Iterating on [www.zigg.com](http://www.zigg.com)
- Want to develop the box locally using Vagrant, iterating using `vagrant provision`
- Once happy, push the result up to DigitalOcean

# How I did it

- Do the majority of the work in roles (“www” only for now)
- Two playbooks contain only the Vagrant- and DigitalOcean-specific bits, pulling in the www role
- www role depends on common role, which handles server setup
- Use `ansible_python_interpreter` to use the virtualenv python for `dopy`

# How I did it

- Unfortunately, DO provisioning right now requires frequent editing of the DO playbook
- Current [digital\\_ocean](#) requires ids because DO API does; hope to use slugs in the future with DO API 2.0

# Discussion

< Thanks for being such a great audience! >

\ ^ ^  
 \ (oo) \ \_\_\_\_\_ ) \ / \  
 ( \_ ) \ \_\_\_\_\_ ) \ / \  
 | | -----w |  
 | | | |

-me | cowsay