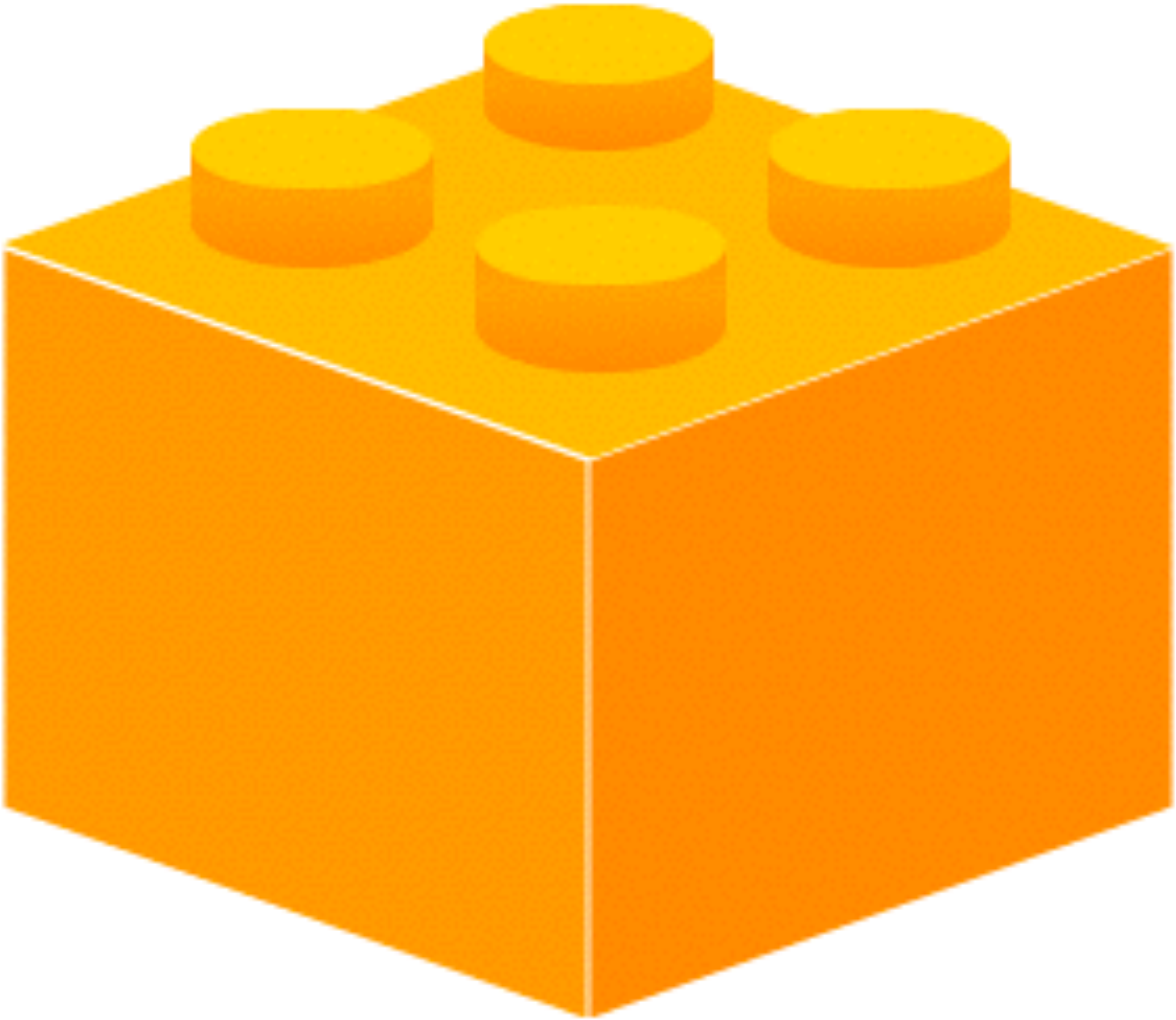
# Ansible and AWS

Advanced Amazon Web Services Meetup
November 4, 2013

Peter Sankauskas
@pas256

Answers for AWS
@Answers4AWS

We’re Back!



## About this group

# Goals

- Help make you more **AW**e**S**ome
- Learn something new
    - Share techniques, tips and tricks on using AWS
    - Share best practices
    - Share war stories
    - Share code and tools

# Assumptions

- You use AWS now
  - Don’t need to be sold on it
  - Know why it is great
- You know where the AWS documentation is
  - No “what is EC2” sessions here

*“Ask not what your meetup can do for you -*

*ask what you can do for your meetup”*

# Speakers

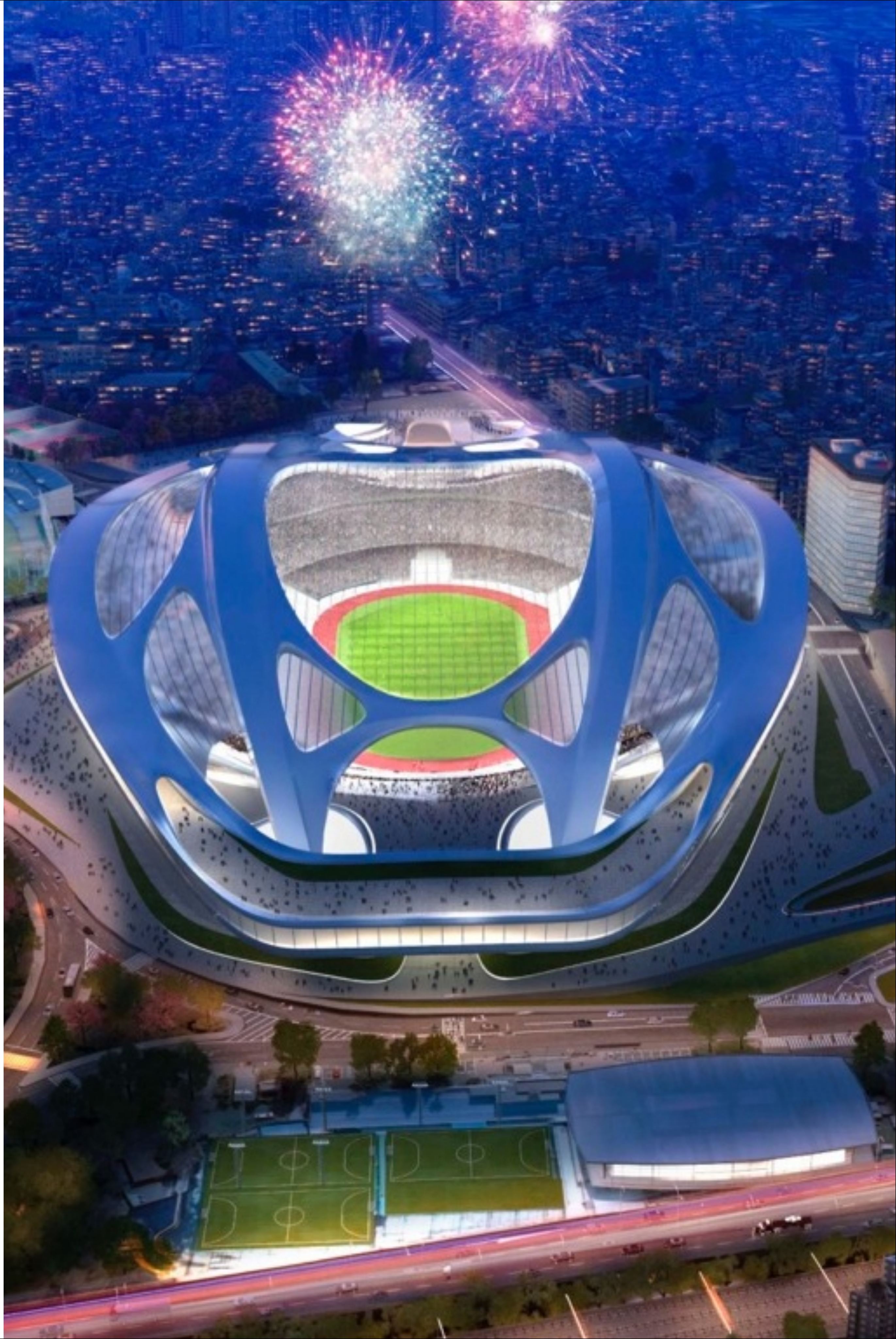
- Got something to share?
  - What are your AWS stories?
  - What problems have you solved?
  - What do you use and how do you use it?
  - Formats: 5, 10, 30 and 42 minute sessions

# Sponsors

- Want to tell us about your product/service for 10 minutes before the main presentations?
- Pay for food/drinks and you can

# Venue

- Want to host?
  - San Francisco
  - Peninsula
- You get to present too!



# About Me

**Peter Sankauskas**

- From Sydney, Australia
- Using AWS for 5 years
  - from a 12 person startup
  - to a 55,000 employee enterprise
- 2009 AWS Startup Challenge Finalist
  - Beaten by Bizo

bluepulse™

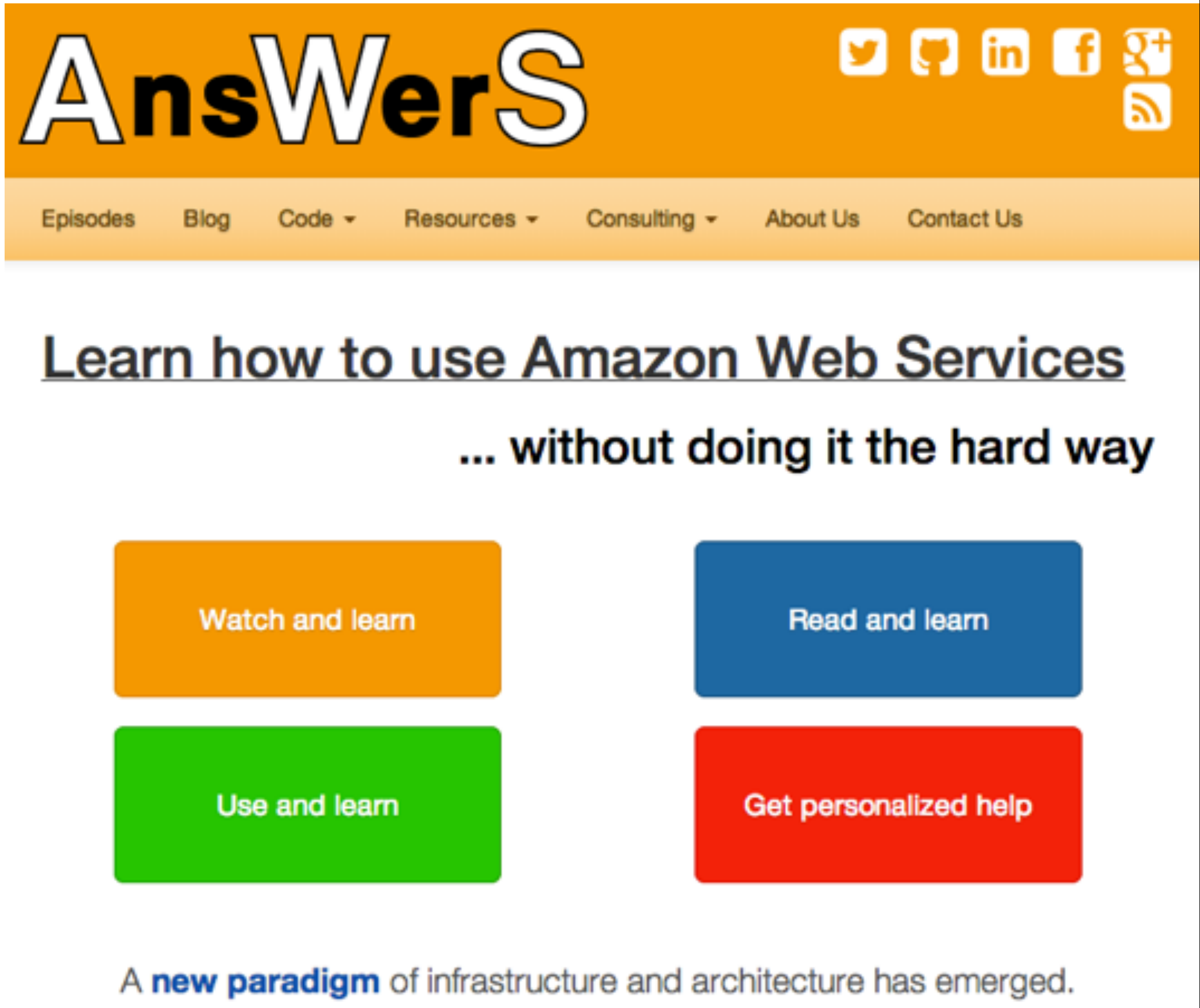
motally
mobile analytics

NOKIA
Connecting People

AdMobius

# Answers for AWS

- Episodes & Blog
  - Ansible, Reserved Instances, CloudFormation
- Code
  - Nominated for NetflixOSS Cloud Prize for Ansible Playbooks
  - Graffiti Monkey, Backup Monkey
- Personalized Help
  - Consulting services and training



AnsWerS

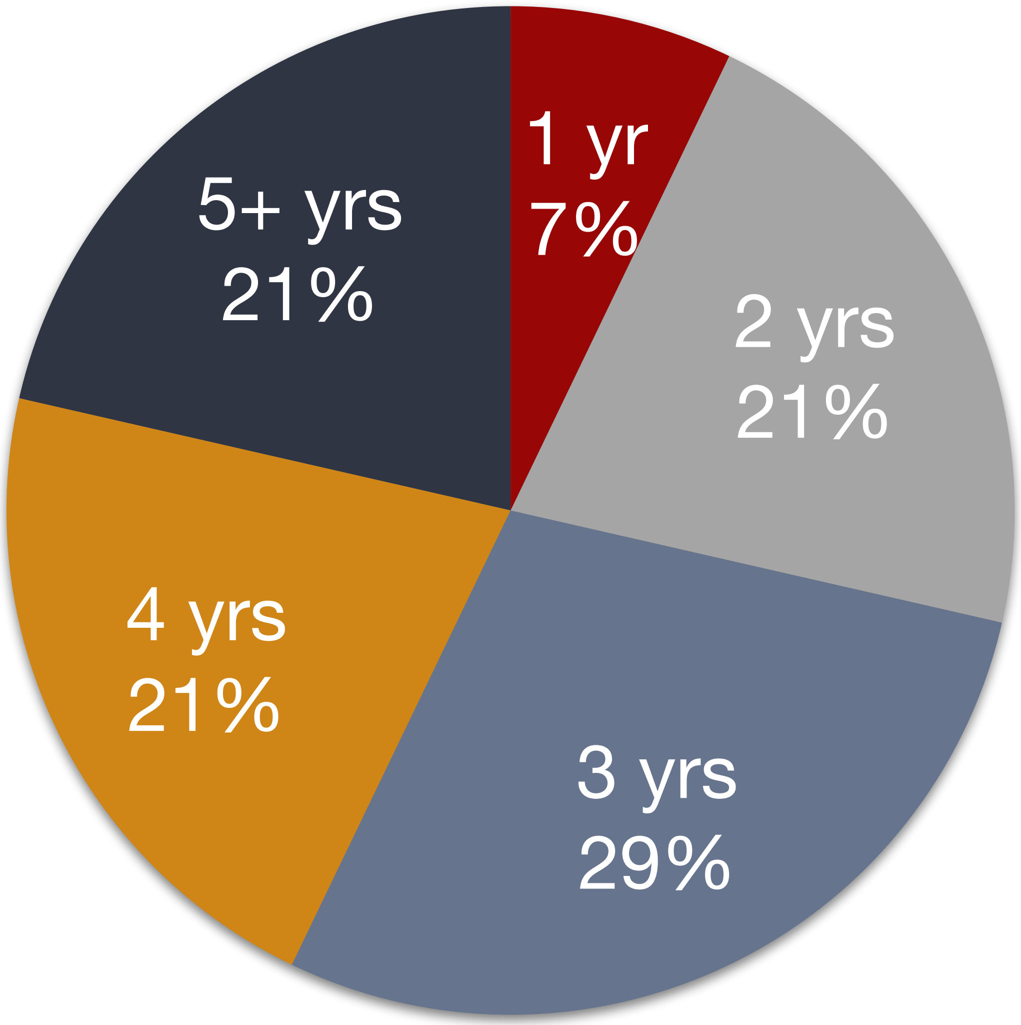
Learn how to use Amazon Web Services

... without doing it the hard way

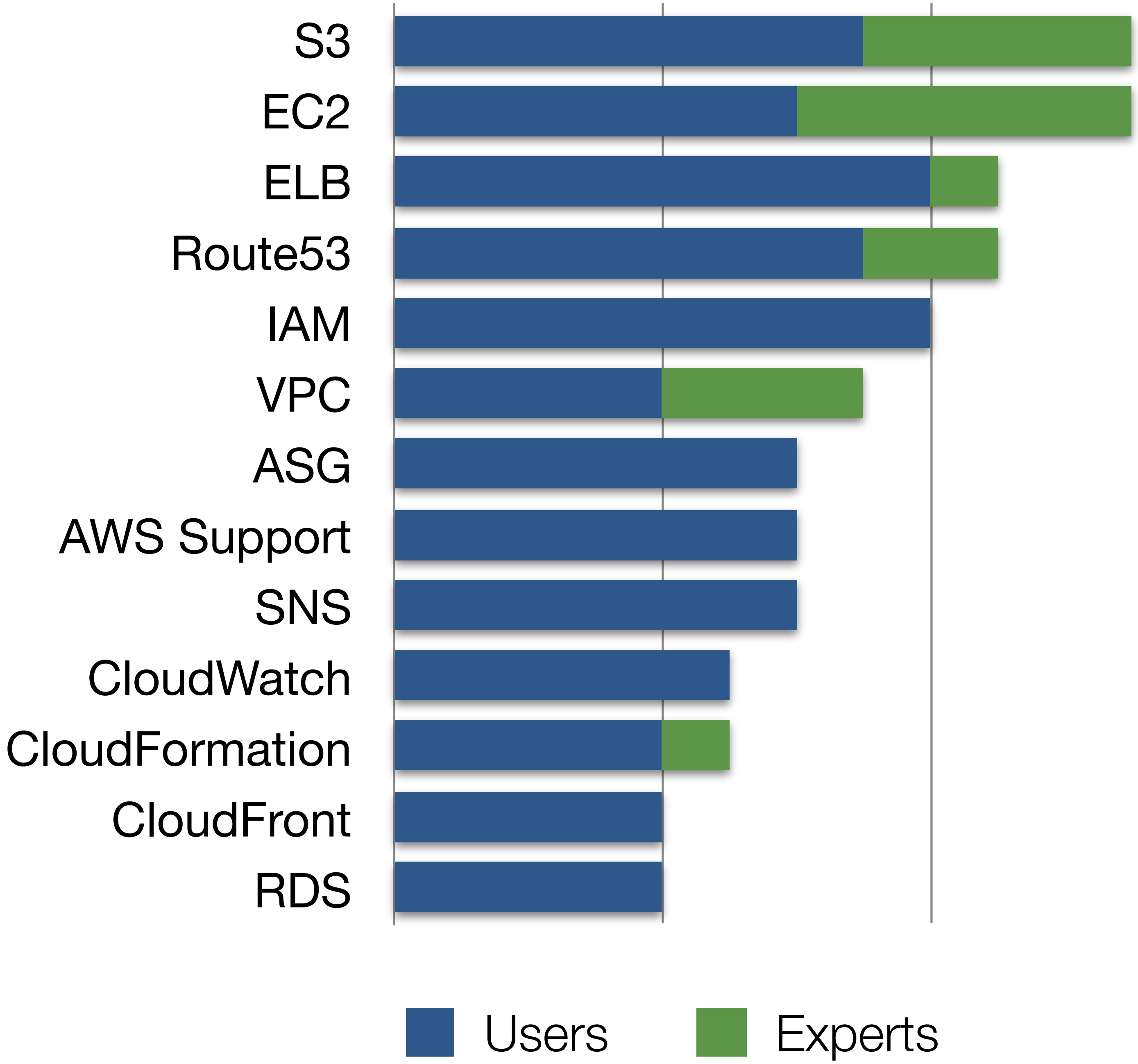
A new paradigm of infrastructure and architecture has emerged.

# Survey results

## Years of AWS Experience

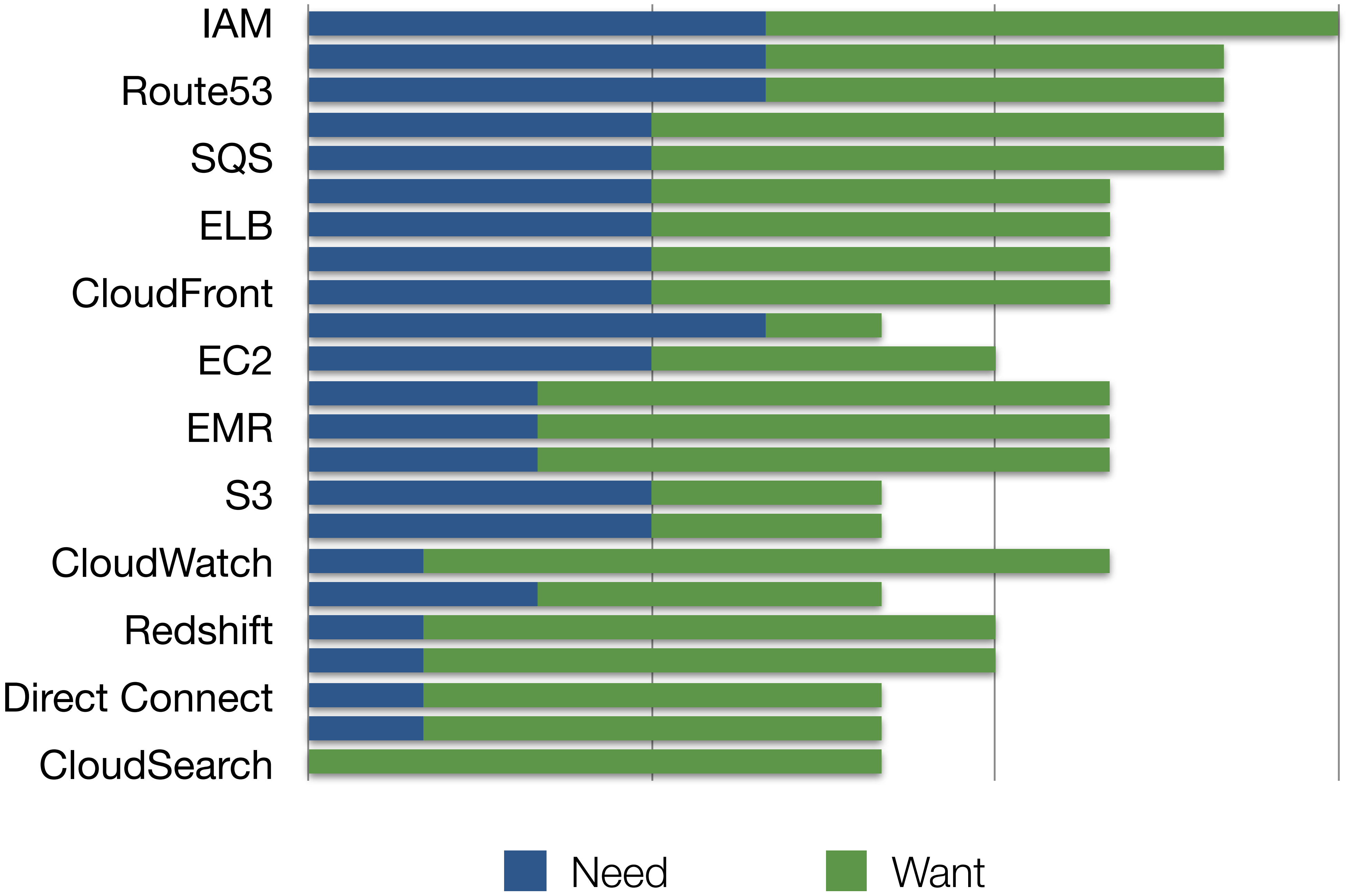


## What you use



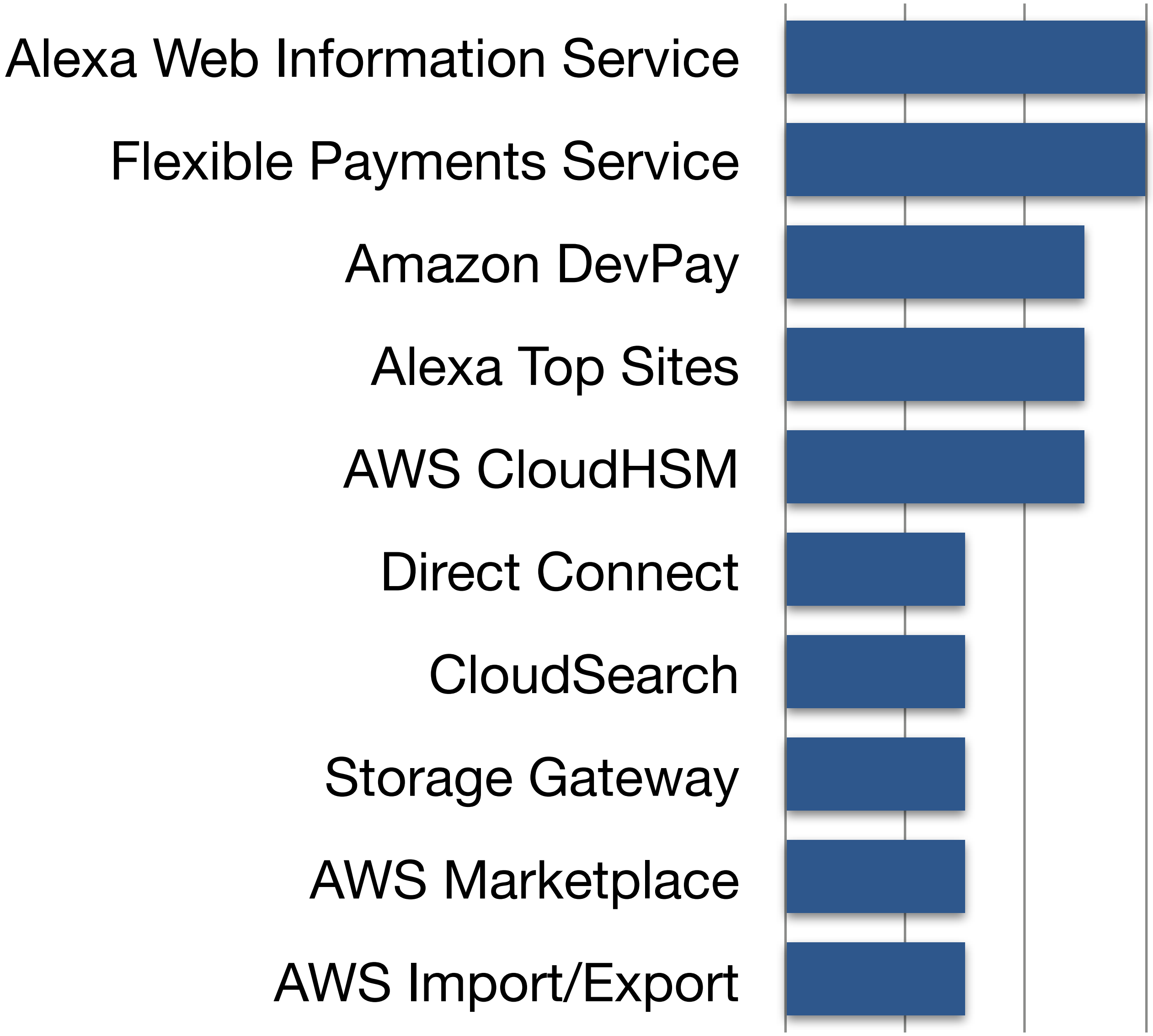
# Survey results

## Want to learn more about

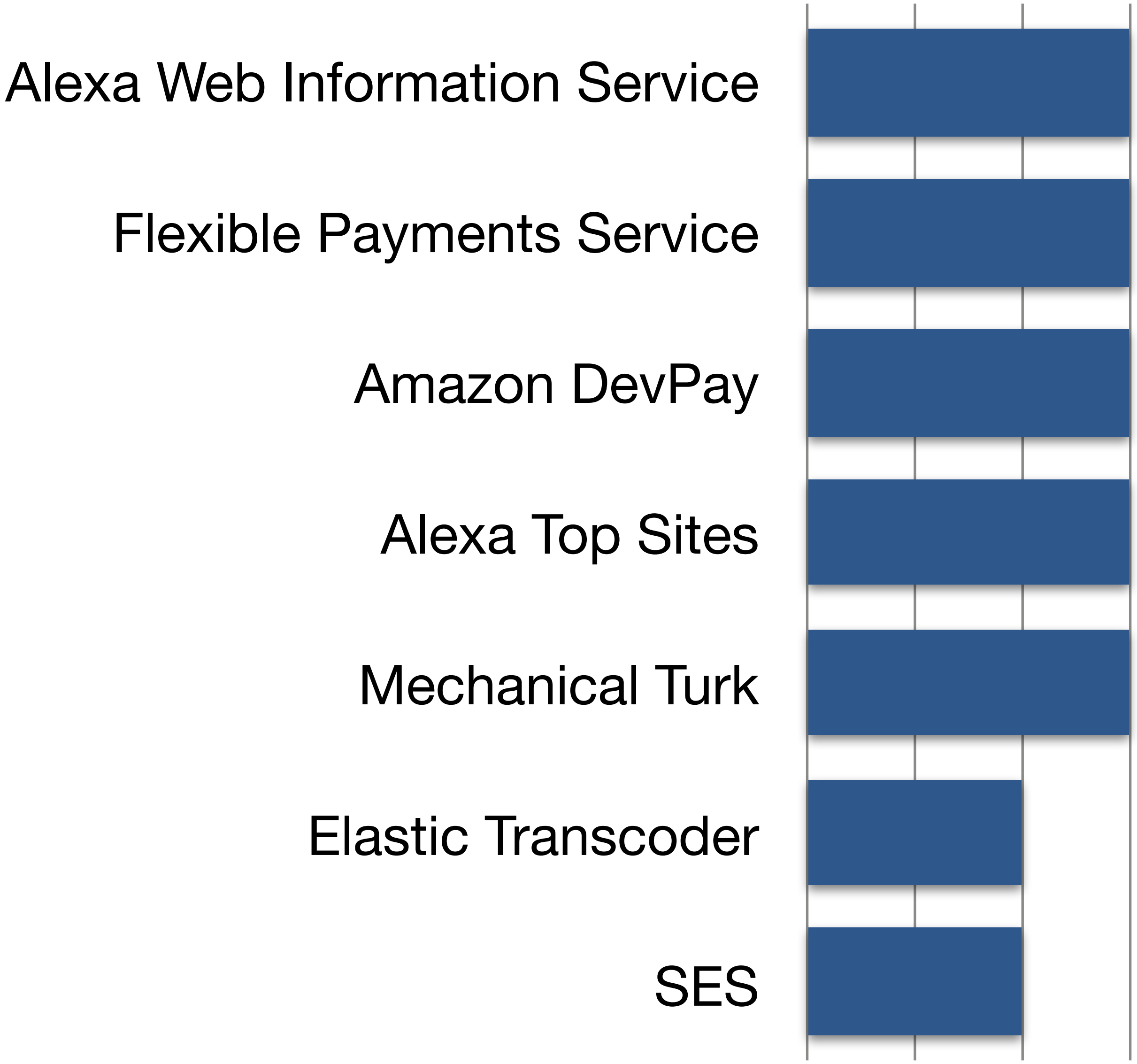


# Survey results

## Don’t know



## Don’t care



# Are you going to re:invent?

It’s sold out

# Ansible

# History

- 1st generation
  - CF Engine
- 2nd generation
  - Puppet
  - Chef
- 3rd generation
  - Ansible
  - Salt Stack

# Ansible is…

- a radically simple IT orchestration engine that makes your applications and systems easier to deploy

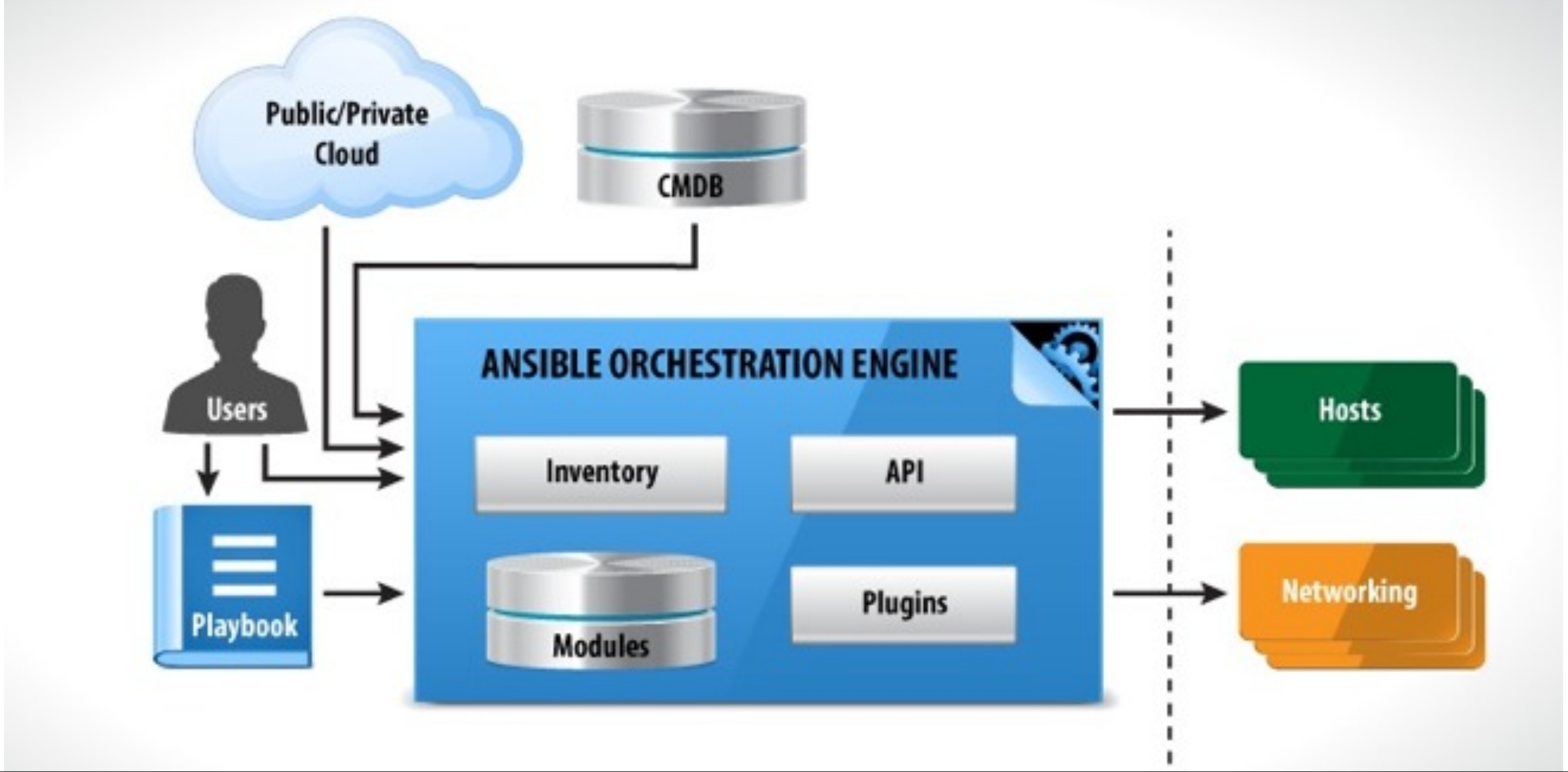
- written in Python

- secure by using SSH for connections

- agent-less

- has AnsibleWorks as a backing company

- free, open source, & available on GitHub



# Installation

From Source

```
$ sudo pip install paramiko PyYAML jinja2
$ git clone git://github.com/ansible/ansible.git
$ cd ./ansible
$ source ./hacking/env-setup
```

Using PIP

```
$ sudo pip install ansible
```

Using yum

```
$ sudo yum install ansible
```

Using apt

```
$ sudo add-apt-repository ppa:rquillo/ansible
$ sudo apt-get update
$ sudo apt-get install ansible
```

# Inventory

- List of your hosts
  - Grouped together
- Example hosts file:

```
/etc/ansible/hosts

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

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

[california]
foo.example.com
one.example.com
```

# Targeting

- Use groups in inventory to target hosts
- Combine groups to get specific
  - Use set operators
    - AND, OR, NOT
- Examples:

```
ansible -m ping webservers

ansible -m ping dbservers

ansible -m ping webservers:&california

ansible -m ping webservers:!phoenix
```

# EC2 inventory plugin

- AWS has an API which is always up-to-date
- Generate inventory off that, return JSON
- Group instances by:
  - region and availability zone
  - security groups
  - tags
  - keypairs
  - more…
- Uses boto

# boto

- Python library for AWS
  - Written by Mitch Garnaat and then hired by Amazon
    - He also writes the new AWS CLI
- Multiple ways to supply it with AWS credentials
  - Environment variables
  - IAM Role
  - `.boto` file

# .boto file

```
[Credentials]

aws_access_key_id = AKIABCDEFGHIJKLM

aws_secret_access_key = duhke3pth15aSECr3t0R3153
```

# Instance variables

- EC2 inventory script collections information about each instance
- Makes variables available to
  - plays
  - playbooks
  - templates

```
ec2_architecture
ec2_description
ec2_dns_name
ec2_id
ec2_image_id
ec2_instance_type
ec2_ip_address
ec2_kernel
ec2_key_name
ec2_launch_time
ec2_monitored
ec2_ownerId
ec2_placement
ec2_platform
ec2_previous_state
ec2_private_dns_name
ec2_private_ip_address
ec2_public_dns_name
ec2_ramdisk
ec2_region
ec2_root_device_name
ec2_root_device_type
ec2_security_group_ids
ec2_security_group_names
ec2_state
ec2_state_code
ec2_state_reason
ec2_status
ec2_subnet_id
ec2_tag_Name
ec2_tenancy
ec2_virtualization_type
ec2_vpc_id
```

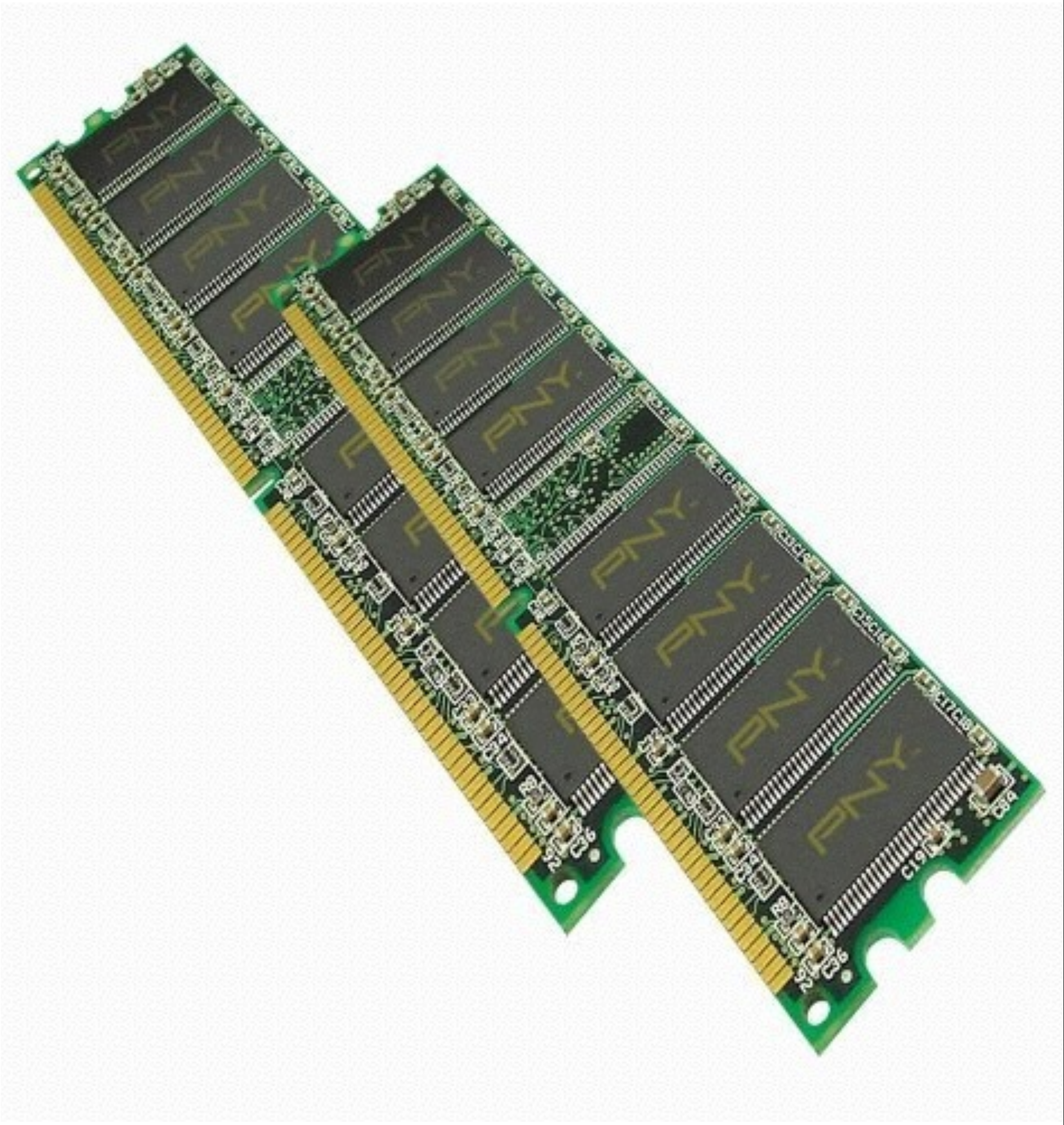
# Modules

accelerate
add_host
apt
apt_key
apt_repository
arista_interface
assemble
async_status
authorized_key
bigip_pool
campfire
cloudformation
command
copy
cron
datadog_event
debug
digital_ocean
dnsmadeeasy
easy_install
**ec2**
**ec2_ami**
**ec2_eip**
**ec2_elb**
**ec2_facts**
**ec2_group**
**ec2_tag**
**ec2_vol**
facter
fail
fetch
file
filesystem
fireball
firewalld
flowdock
gem
get_url
git
glance_image
group
group_by
hg
hipchat
homebrew
host
htpasswd
ini_file
irc
jabber
lineinfile
linode
lvg
lvol
macports
mail
modprobe
monit
mount
mqtt
mysql_db
mysql_user
nagios
netscaler
nova_compute
nova_keypair
npm
ohai
openbsd_pkg
opkg
osx_say
pacman
pagerduty
pause
ping
pingdom
pip
pkgin
pkgng
pkgutil
postgresql_db
raw
rax
rax_clb
**rds**
redis
rhn_channel
rhn_register
riak
**route53**
rpm_key
**s3**
script
selinux
service
set_fact
setup
shell
slurp
stat
subversion
supervisorctl
svr4pkg
sysctl
template
uri
user
virt
xattr
yum
zfs
zypper

# Modules

- All modules are part of core
  - No competing modules
  - No abandoned modules
- All core modules are written in Python
- You can write custom modules in any language
  - There is already helper code in Ruby

**https://github.com/ansible/ansible-for-rubyists**

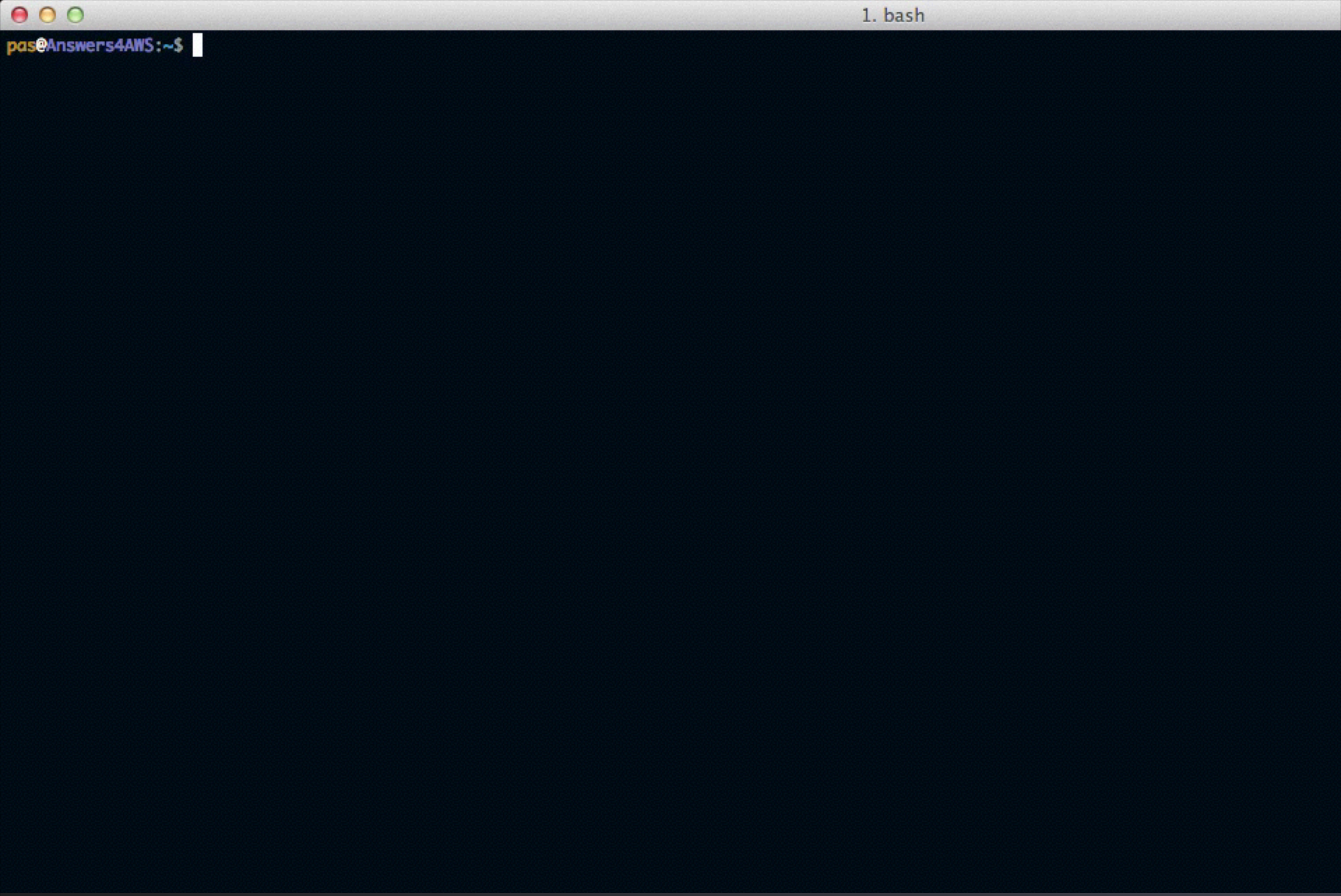


# ping

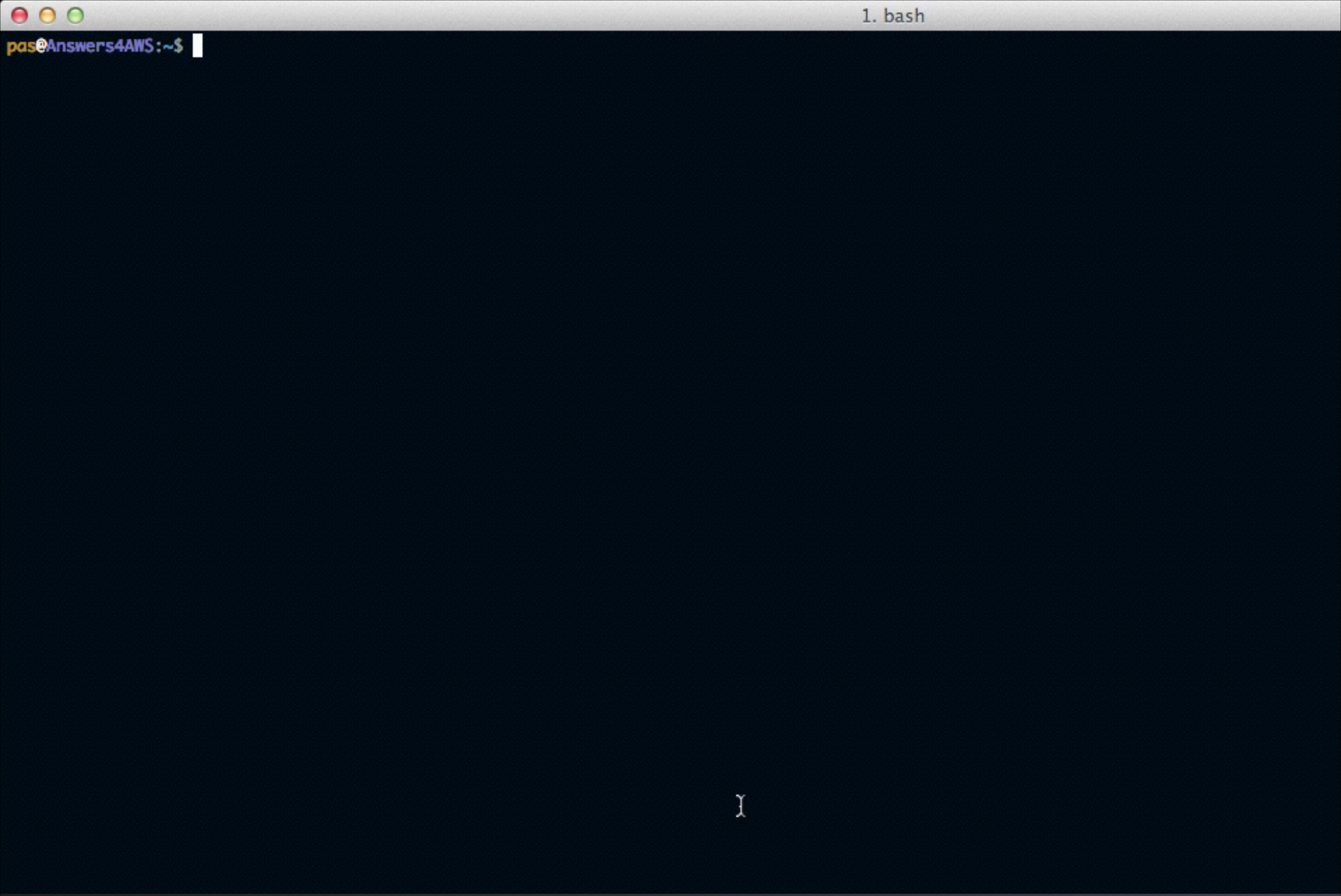
A trivial test module, this module always returns pong on successful contact. It does not make sense in playbooks, but it is useful from /usr/bin/ansible

```
# Test connection

ansible webservers -m ping
```

Demo



```
pas@Answers4AWS:~$ 
```



```
pas@Answers4AWS:~$
```

# Ad-hoc tasks

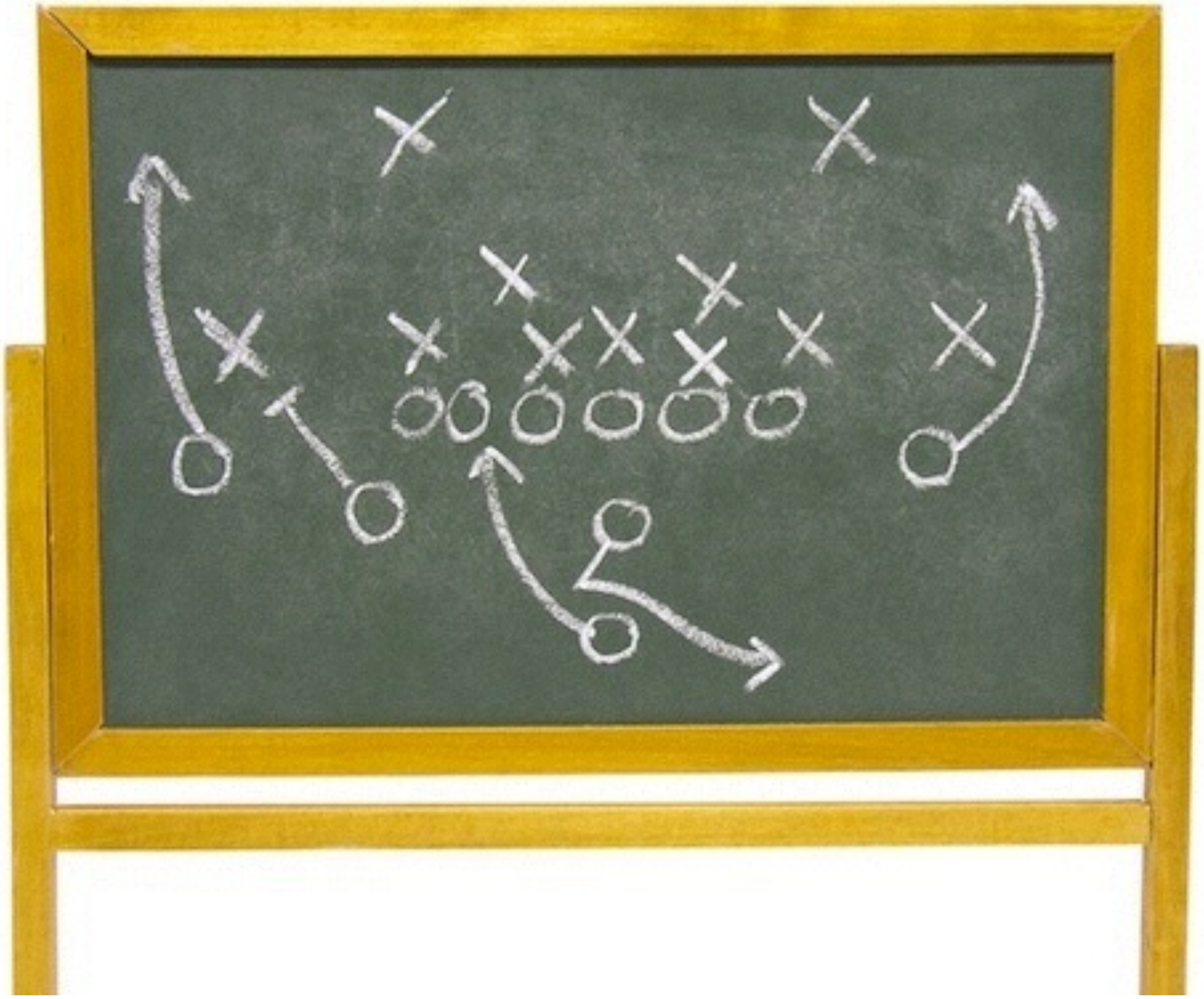
- Target the desired instance or instances
- Choose the module
- Specify the arguments

# Examples

```
ansible -m copy -a "src=script.sh
  dest=/usr/bin/script owner=root group=root
  mode=0755"  webservers

ansible -m service
  -a "name=apache state=restarted"
  --forks=2  webservers

ansible -m user -a ‘name=nsa comment="NSA"
  uid=9999’  dbservers
```

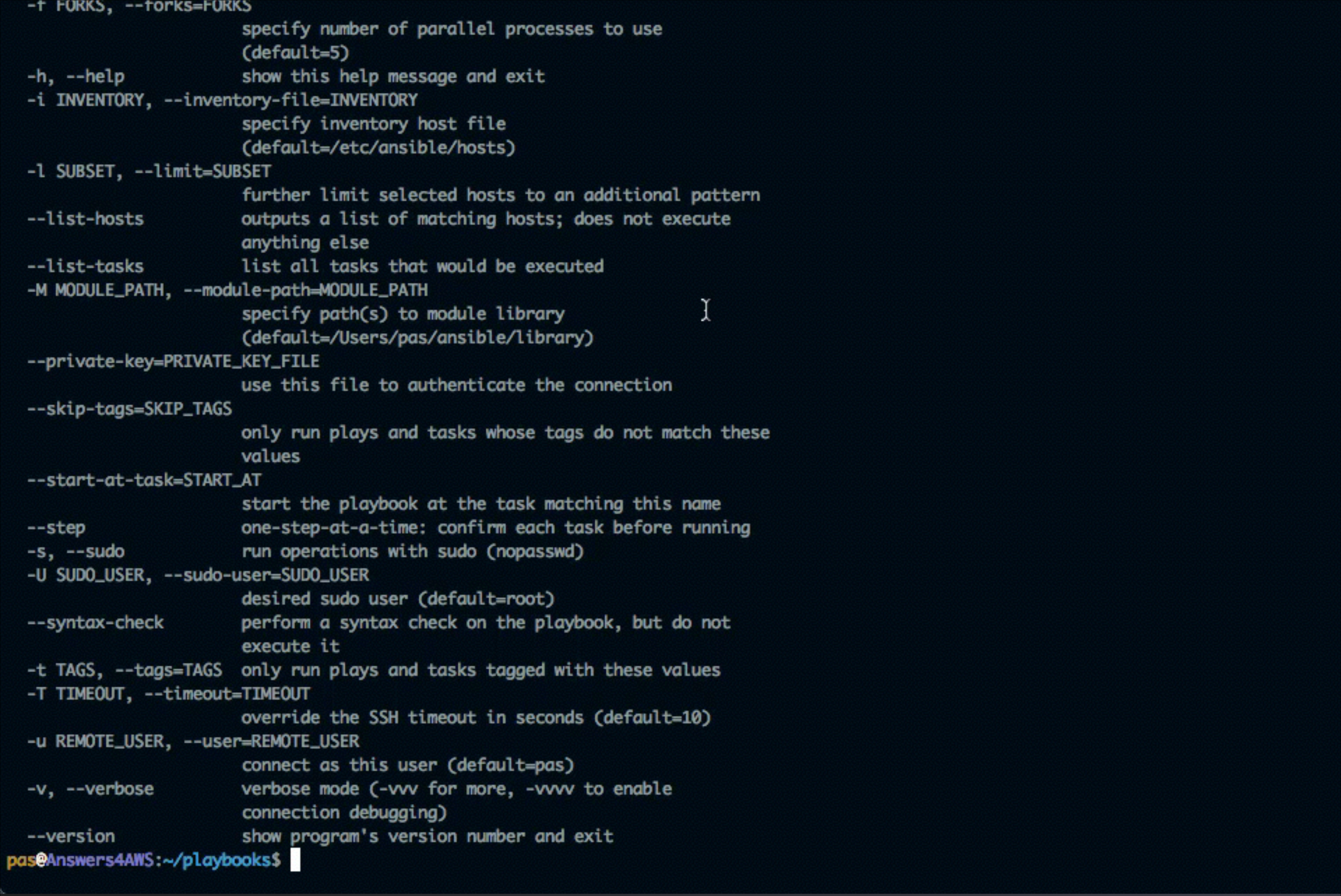
# Playbooks

- Contains one or more “plays”
- Written in YAML
  - Declare configuration
  - YAML is not code
- Executed in the order it is written
  - No dependency graph



# Install AWS command line tool

```
---
- name: Install AWS CLI
  user: ubuntu
  sudo: True
  hosts: all
  tasks:
    - name: Install Python PIP
      apt: pkg=python-pip state=latest

    - name: Install boto via PIP
      pip: name=boto state=latest

    - name: Install AWS CLI
      pip: name=awscli state=latest
```



```
  -f FORKS, --forks=FORKS
                        specify number of parallel processes to use
                        (default=5)
  -h, --help            show this help message and exit
  -i INVENTORY, --inventory-file=INVENTORY
                        specify inventory host file
                        (default=/etc/ansible/hosts)
  -l SUBSET, --limit=SUBSET
                        further limit selected hosts to an additional pattern
  --list-hosts          outputs a list of matching hosts; does not execute
                        anything else
  --list-tasks          list all tasks that would be executed
  -M MODULE_PATH, --module-path=MODULE_PATH
                        specify path(s) to module library
                        (default=/Users/pas/ansible/library)
  --private-key=PRIVATE_KEY_FILE
                        use this file to authenticate the connection
  --skip-tags=SKIP_TAGS
                        only run plays and tasks whose tags do not match these
                        values
  --start-at-task=START_AT
                        start the playbook at the task matching this name
  --step                one-step-at-a-time: confirm each task before running
  -s, --sudo            run operations with sudo (nopasswd)
  -U SUDO_USER, --sudo-user=SUDO_USER
                        desired sudo user (default=root)
  --syntax-check        perform a syntax check on the playbook, but do not
                        execute it
  -t TAGS, --tags=TAGS  only run plays and tasks tagged with these values
  -T TIMEOUT, --timeout=TIMEOUT
                        override the SSH timeout in seconds (default=10)
  -u REMOTE_USER, --user=REMOTE_USER
                        connect as this user (default=pas)
  -v, --verbose         verbose mode (-vvv for more, -vvvv to enable
                        connection debugging)
  --version             show program's version number and exit
pas@Answers4AWS:~/playbooks$
```

# DRY

- Includes
  - Reuse lists of task
- Roles
  - Reuse a set of tasks, files, variables and templates

# Includes

```
---
- name: Install AWS CLI
  user: ubuntu
  sudo: True
  hosts: all
  tasks:
    - include: install-aws-cli.yml
```

install-aws-cli.yml

```
- name: Install Python PIP
  apt: pkg=python-pip state=latest

- name: Install boto via PIP
  pip: name=boto state=latest

- name: Install AWS CLI
  pip: name=awscli state=latest
```

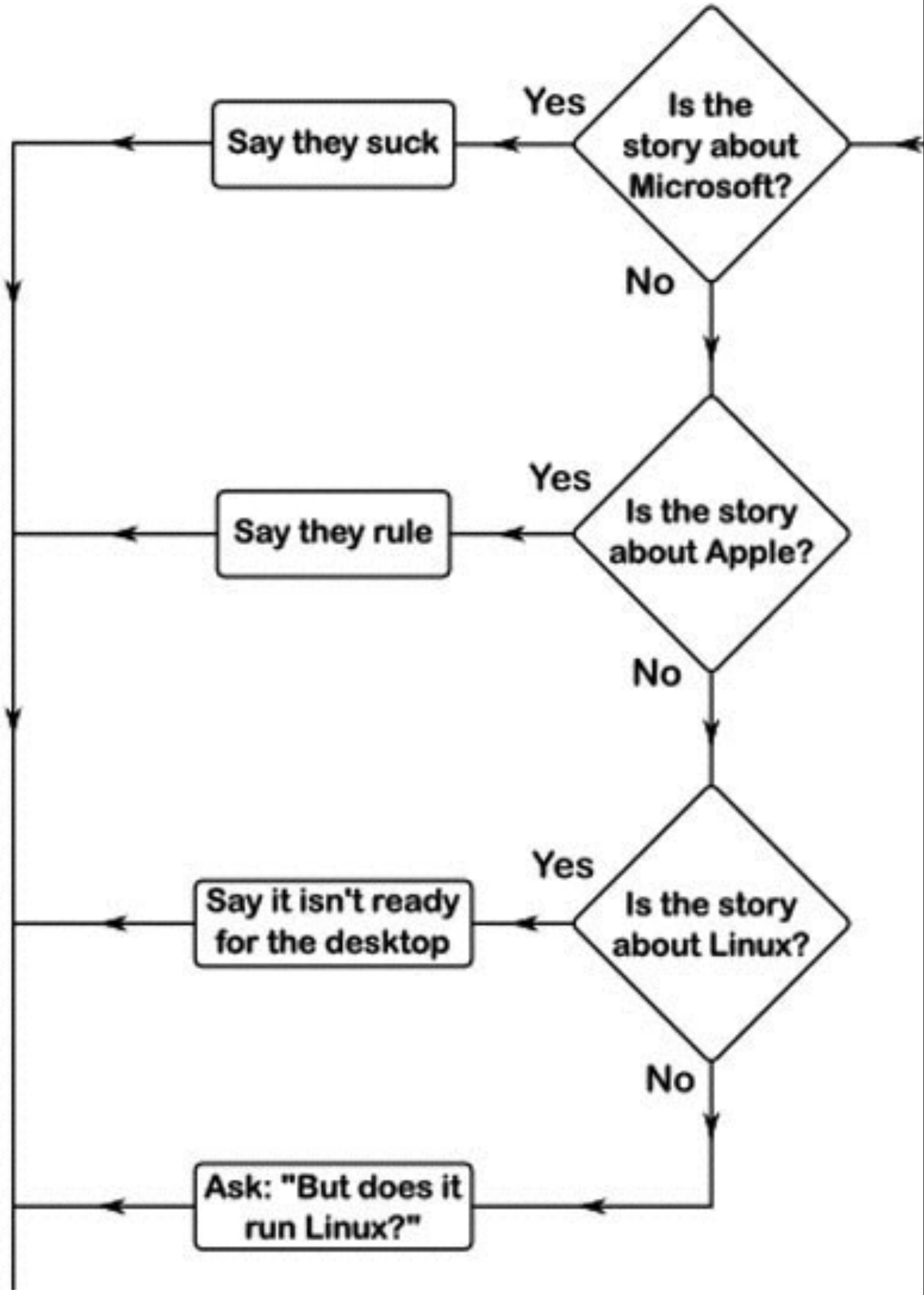
# Roles

```
---
- name: Set up web boxes
  user: ubuntu
  sudo: True
  hosts: webservers
  roles:
    - base
    - webserver
```

```
webservers.yml
dbservers.yml
roles/
   base/
     files/
     templates/
     tasks/
     handlers/
     vars/
     meta/
   webservers/
     files/
     templates/
     tasks/
     handlers/
     vars/
     meta/
```

# Conditions & Loops

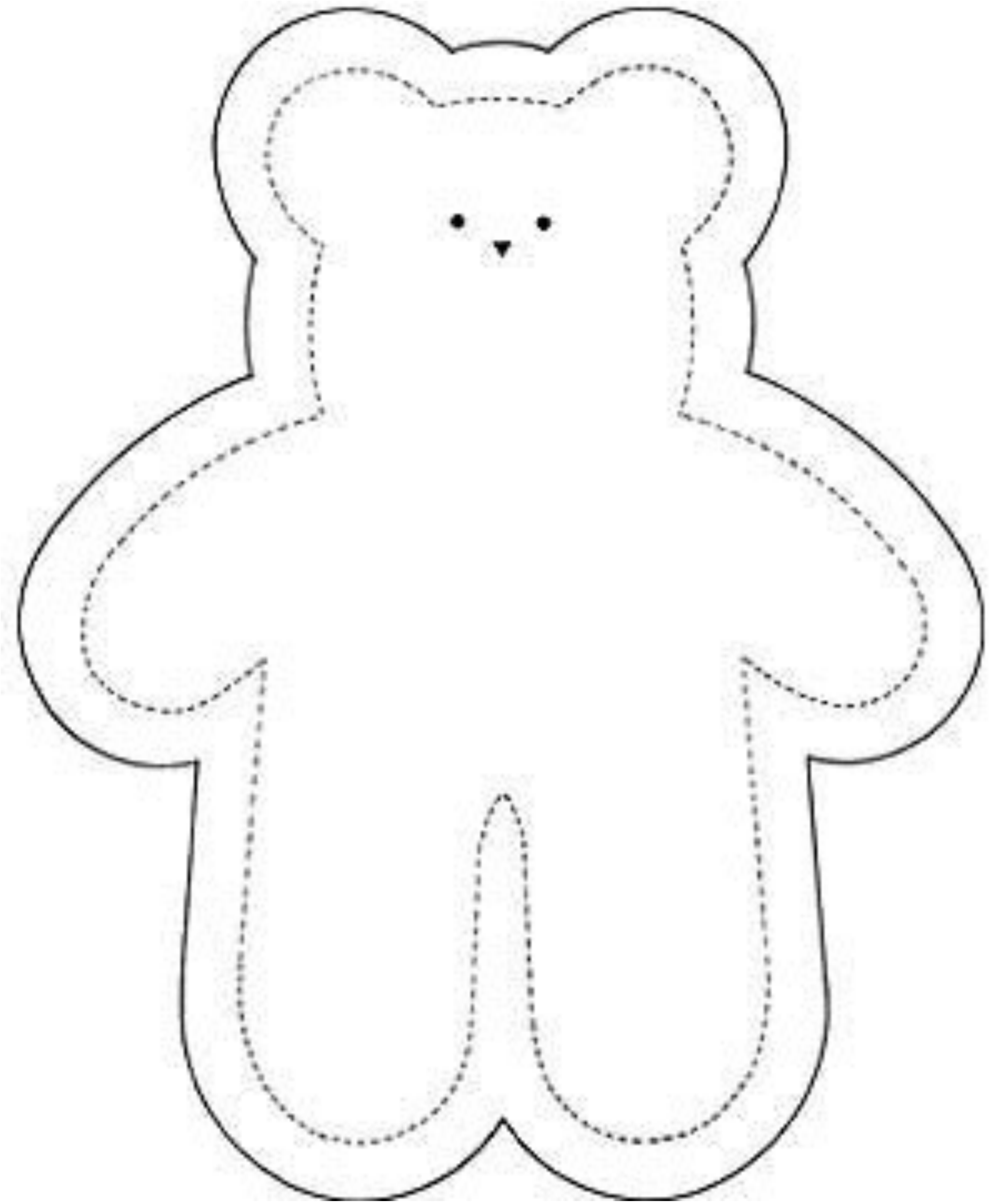
```
---
# Install everyone's favorite editors

- name: Install editor packages (apt)
  apt: pkg={{ item }} state=latest
  with_items:
    - emacs23-nox
    - emacs23-el
    - vim
  when: ansible_distribution == 'Ubuntu'


- name: Install editors packages (yum)
  yum: pkg={{ item }} state=latest
  with_items:
    - emacs
    - emacs-el
    - vim-enhanced
  when: ansible_distribution == 'Amazon'
```



Is the story about Microsoft?
Yes
Say they suck
No
Is the story about Apple?
Yes
Say they rule
No
Is the story about Linux?
Yes
Say it isn't ready for the desktop
No
Ask: "But does it run Linux?"

# Templates

- File with variable substitutions
- Same as Puppet and Chef templates
- Uses Jinja2 instead of ERB
    - {{ variable }}
    - {{ filename | md5 }}



# Provisioning

- Modules for
  - Creating security groups
  - Launching EC2 instances
  - Assigning EIPs
  - Register instances with ELBs
  - Tagging resources
  - RDS, S3 and CloudFormation

# Provisioning Playbook - 1

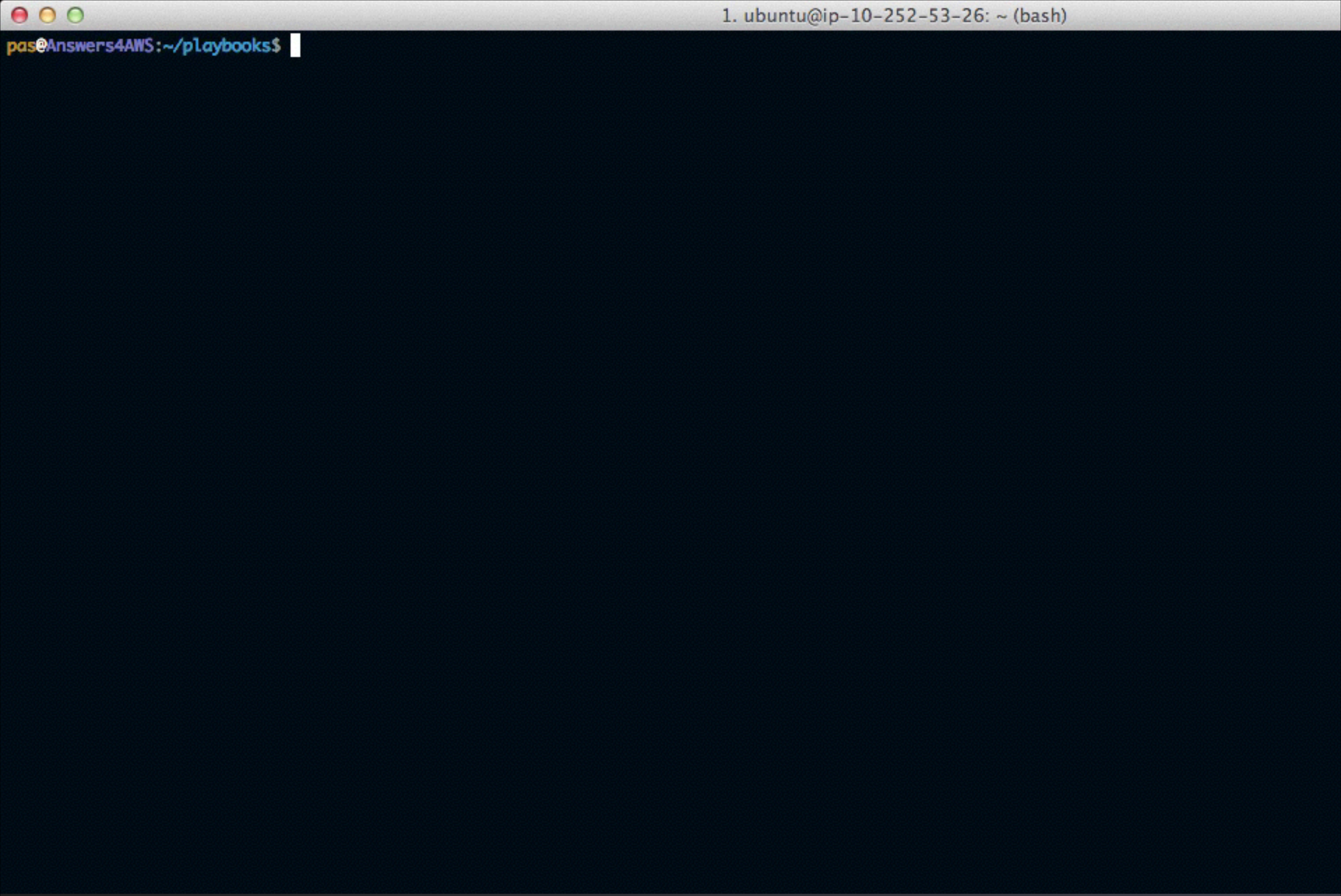
```
---
- name: Example of provisioning servers
  hosts: 127.0.0.1
  connection: local
  tasks:
    - name: Create security group
      local_action:
        module: ec2_group
        name: ep2
        description: Access to the Episode2 servers
        region: us-east-1
        rules:
          - proto: tcp
            from_port: 22
            to_port: 22
            cidr_ip: 0.0.0.0/0
```

# Provisioning Playbook - 2

```
- name: Launch instances
  local_action:
    module: ec2
    region: us-east-1
    keypair: answersforaws
    group: ep2
    instance_type: m1.small
    image: ami-8635a9b6
    count: 2
    wait: yes
  register: ec2

- name: Add EP2 instances to host group
  local_action: add_host hostname={{ item.public_ip }} groupname=ep2
  with_items: ec2.instances
```

# Provisioning Playbook - 3

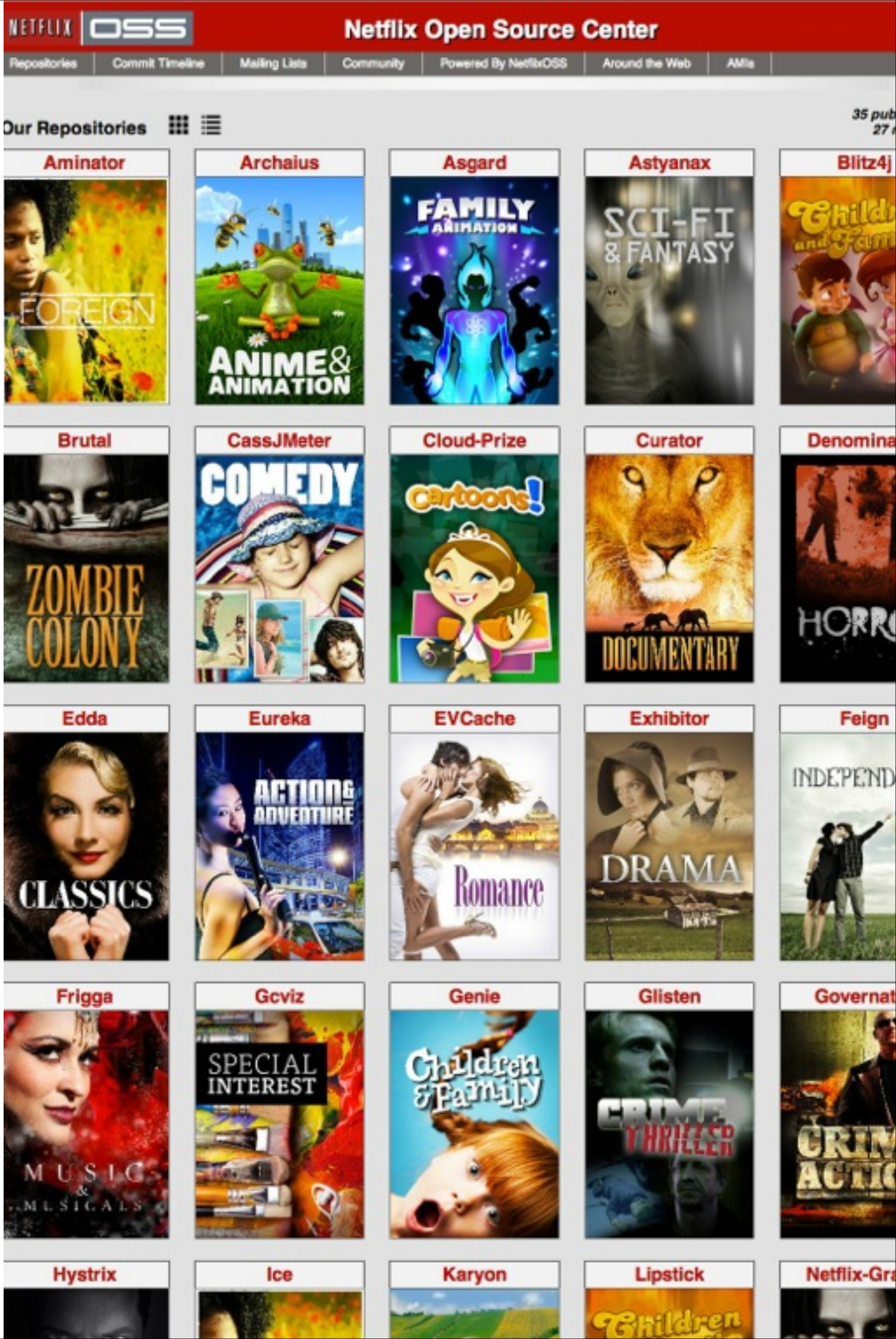
```
    - name: Add tag to instances
      local_action: ec2_tag resource={{ item.id }} state=present
      with_items: ec2.instances
      args:
        tags:
          Name: EP2

    - name: Wait for SSH to be available
      pause: minutes=1


- name: Configure provisioned servers
  hosts: ep2
  user: ubuntu
  sudo: True
  tasks:
    - include: tasks/install-awscli.yml
```



```
pas@Answers4AWS:~/playbooks$
```

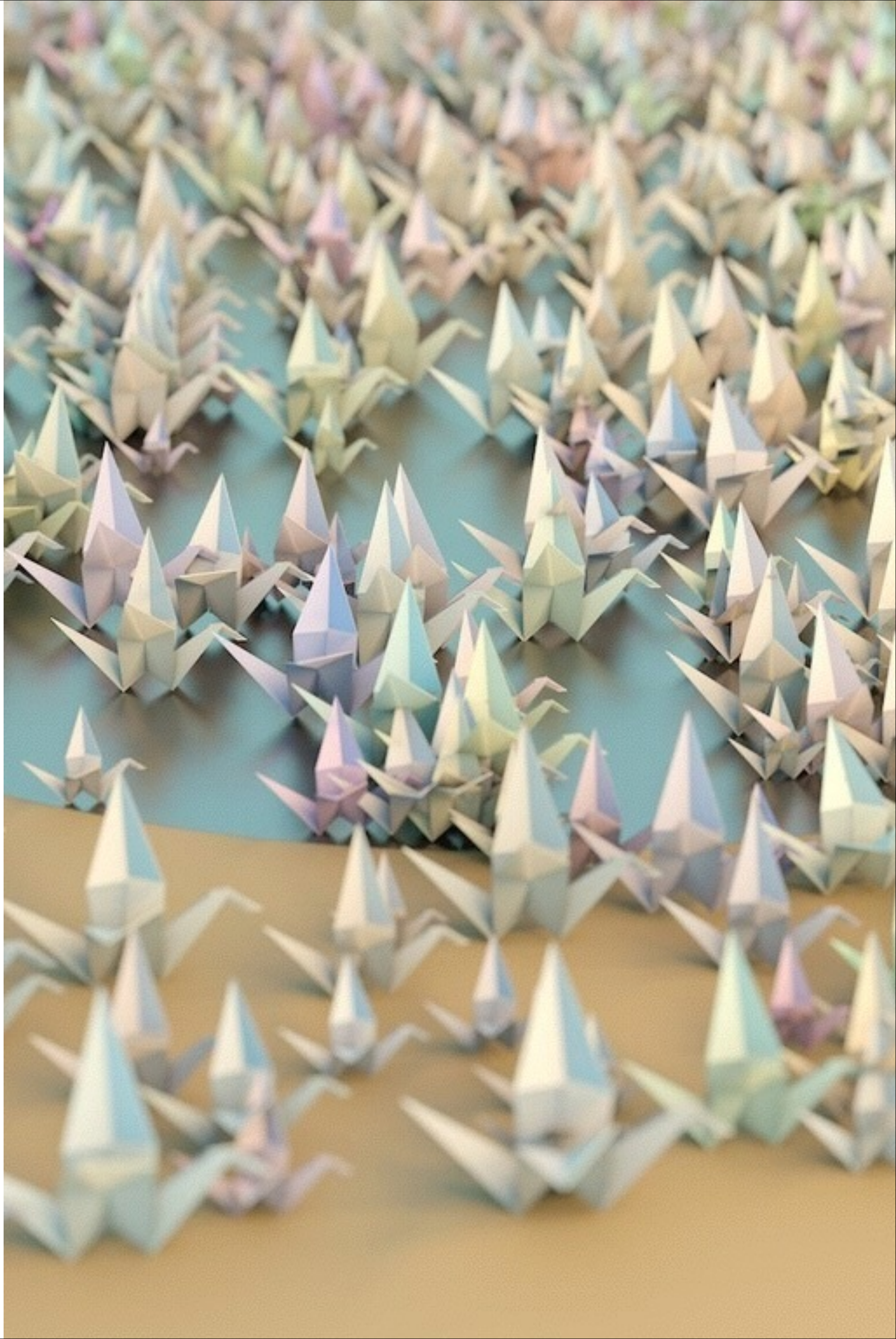
# NetflixOSS Ansible Playbooks

- Aminator
- Asgard
- Edda
- Eureka
- Genie
- Ice
- Simian Army



# AMIs

- Aminator
  - Ansible Provisioner
    - Written by me
- Packer
  - Ansible Provisioner
    - Written by Kelsey Hightower

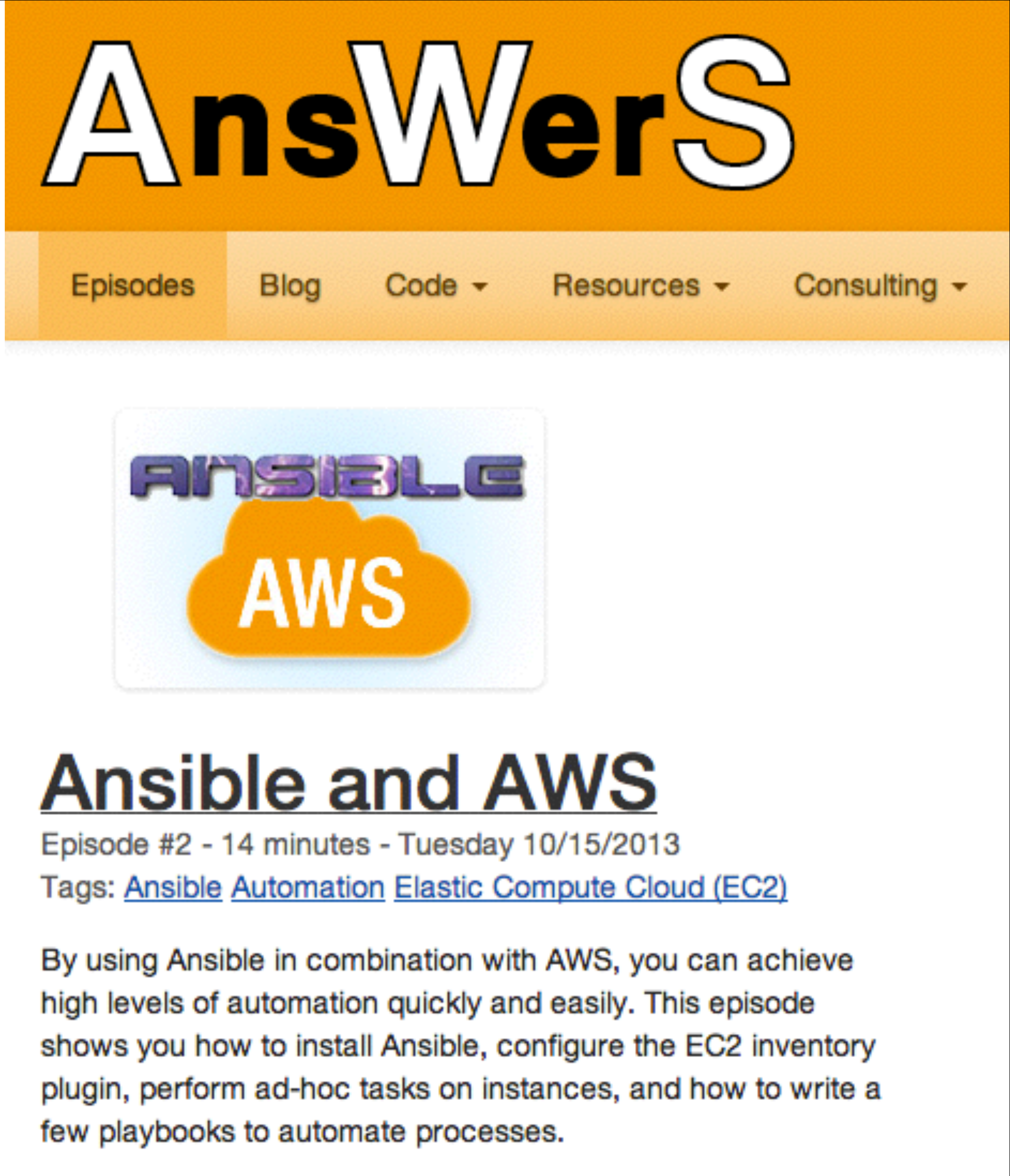


# More

- Prompts
- Tags
- Handlers
- Variable Files
- Rolling Updates
- Delegation
- Custom inventory, plugins and modules

# Video

- A condensed version of this deck in video format is available at Answers for AWS
- Look for Episode 2



## Ansible and AWS

Episode #2 - 14 minutes - Tuesday 10/15/2013

Tags: Ansible Automation Elastic Compute Cloud (EC2)

By using Ansible in combination with AWS, you can achieve high levels of automation quickly and easily. This episode shows you how to install Ansible, configure the EC2 inventory plugin, perform ad-hoc tasks on instances, and how to write a few playbooks to automate processes.

# Bonus

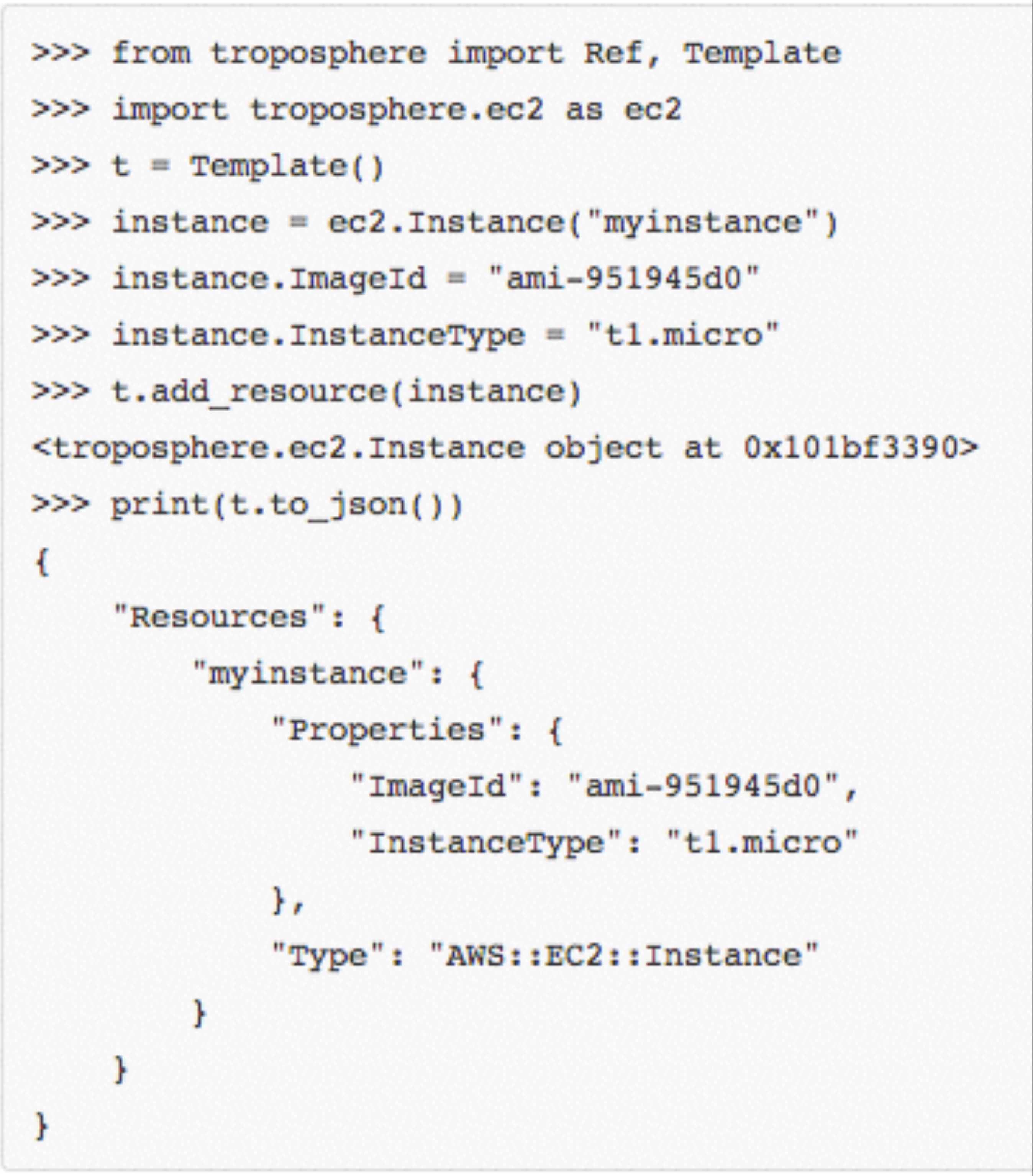
# CloudFormation

- Stack
- Templates
  - written in JSON
    - syntax errors easy
    - prone to typos
  - Checks only done at Stack creation time

# troposphere

https://github.com/cloudtools/troposphere

- API for writing CloudFormation templates
- Written in Python
- Same guy has Python API for writing IAM Policies too
  - https://github.com/cloudtools/awacs

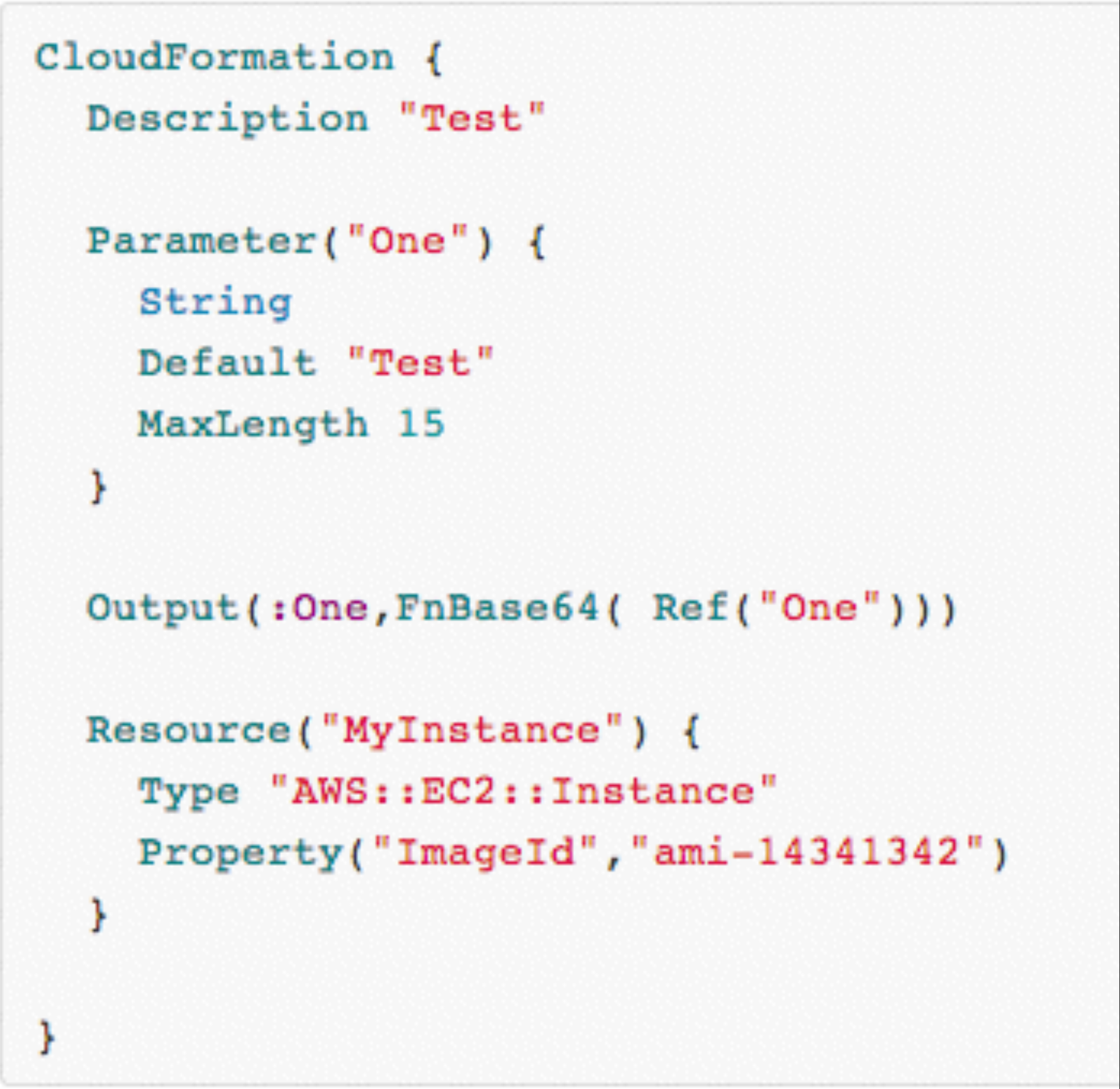


```
>>> from troposphere import Ref, Template
>>> import troposphere.ec2 as ec2
>>> t = Template()
>>> instance = ec2.Instance("myinstance")
>>> instance.ImageId = "ami-951945d0"
>>> instance.InstanceType = "t1.micro"
>>> t.add_resource(instance)
<troposphere.ec2.Instance object at 0x101bf3390>
>>> print(t.to_json())
{
    "Resources": {
        "myinstance": {
            "Properties": {
                "ImageId": "ami-951945d0",
                "InstanceType": "t1.micro"
            },
            "Type": "AWS::EC2::Instance"
        }
    }
}
```

# cfndsl

https://github.com/howech/cfndsl

- DSL for CloudFormation
- Written in Ruby



```
CloudFormation {
  Description "Test"

  Parameter("One") {
    String
    Default "Test"
    MaxLength 15
  }

  Output(:One,FnBase64( Ref("One")))

  Resource("MyInstance") {
    Type "AWS::EC2::Instance"
    Property("ImageId","ami-14341342")
  }

}
```

We are looking for Speakers, Sponsors and Venues.
Don't be shy

# *Thank you!*

Slides available at:

`http://bit.ly/ansible-aws`

Peter Sankauskas
@pas256

Answers for AWS
@Answers4AWS
`answersforaws.com`