

open source devops

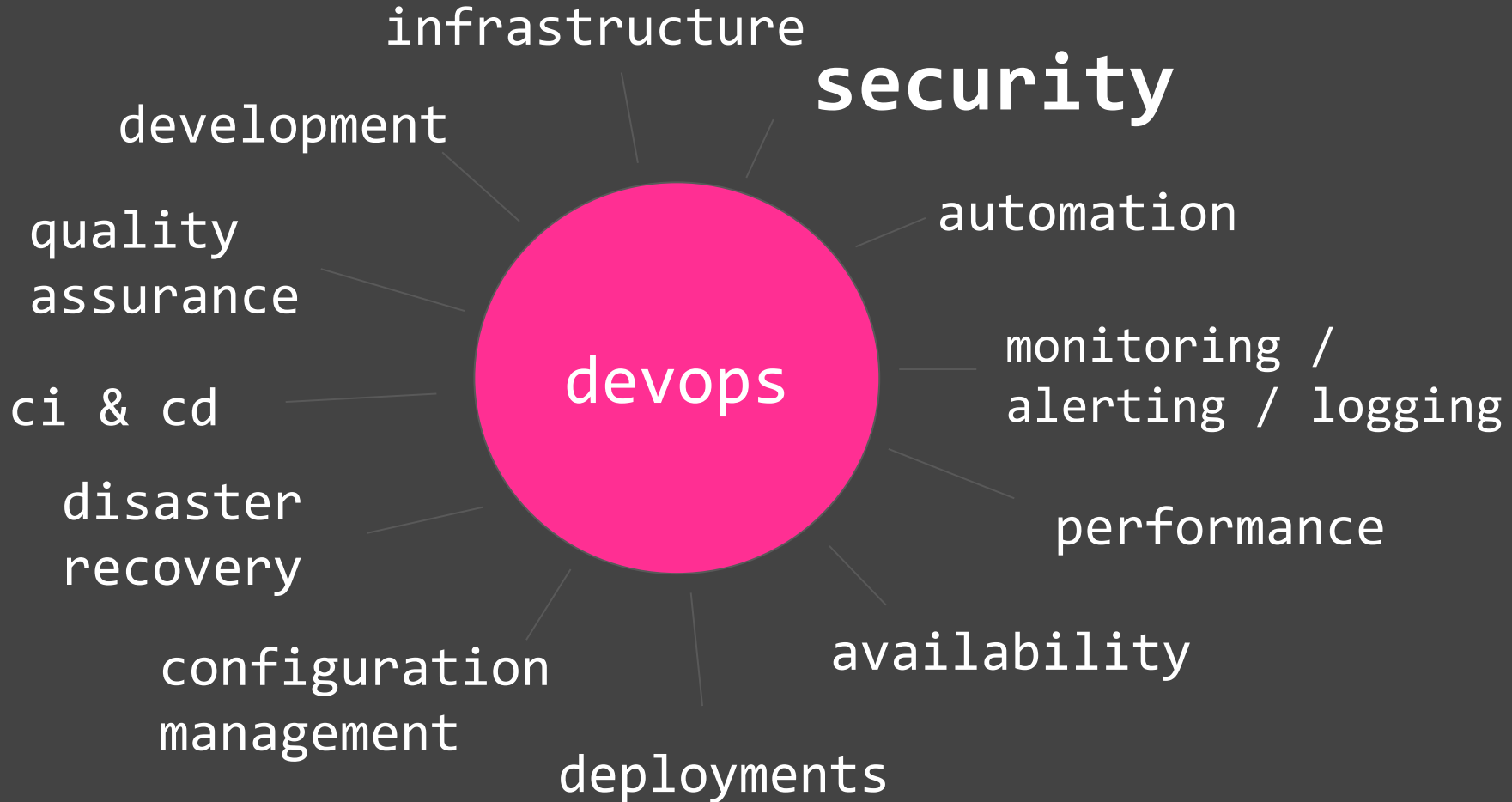
@shortxstack

intro

- whitney champion / @shortxstack
- systems architect / engineer in charleston, SC
- mom of 3
- standard nerd
- always learning
- <https://unicorns.lol>

wtf is devops?







ERMAHGERD



HashiCorp
Packer



Jenkins

graylog



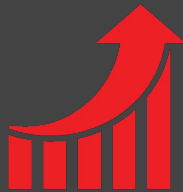
VAGRANT



mist.io



OSSEC



netdata



KOLIDE



elastic



WAZUH



build an AMI with packer, ex 1

```
{
  "variables": {
    "env": "dev",
    "build": "api",
    "timestamp": "1487793684",
    "github-commit": "a7aa8810e0ccce5989cd787851e8311a5d58d50f"
  },
  "builders": [
    {
      "type": "amazon-ec2",
      "region": "us-east-1",
      "associate_public_ip_address": true,
      "source_ami": "ami-1a2b3c4d",
      "security_group_id": "sg-1a2b3c4d",
      "instance_type": "t2.micro",
      "ssh_username": "centos",
      "ssh_private_key_file": "./key-{{user `env`}}.pem",
      "ssh_keypair_name": "key-{{user `env`}}",
      "ami_name": "unicorns-{{user `build`}}-{{user `timestamp`}}",
      "iam_instance_profile": "iam_instance_profile_admin",
      "run_tags": {
        "Name": "unicorns-{{user `env`}}-{{user `build`}}-{{user `timestamp`}}"
      },
      "run_volume_tags": {
        "Name": "unicorns-{{user `env`}}-{{user `build`}}-{{user `timestamp`}}"
      },
      "tags": {
        "Name": "unicorns-{{user `build`}}-{{user `timestamp`}}",
        "Build": "{{user `build`}}",
        "Commit": "{{user `github-commit`}}"
      }
    }
  ],
  "provisioners": [
    {
      "type": "shell",
      "inline": [
        "eval sudo \"$(aws ecr get-login --region us-east-1)\",",
        "sudo docker pull 821112832814.dkr.ecr.us-east-1.amazonaws.com/unicorns-{{user `build`}}:{{user `github-commit`}}"
      ]
    }
  ]
}
```


build an AMI with packer, ex 2

```
{
  "variables": {
    "env": "dev",
    "build": "api",
    "timestamp": "1487793684",
    "github-commit": "a7aa8810e0ccce5989cd787851e8311a5d58d50f"
  },
  "builders": [
    {
      "type": "amazon-eks",
      "region": "us-east-1",
      "associate_public_ip_address": true,
      "source_ami": "ami-1a2b3c4d",
      "security_group_id": "sg-1a2b3c4d",
      "instance_type": "t2.micro",
      "ssh_username": "centos",
      "ssh_private_key_file": "./key-{{user `env`}}.pem",
      "ssh_keypair_name": "key-{{user `env`}}",
      "ami_name": "unicorns-{{user `build`}}-{{user `timestamp`}}",
      "iam_instance_profile": "iam_instance_profile_admin",
      "run_tags": {
        "Name": "unicorns-{{user `env`}}-{{user `build`}}-{{user `timestamp`}}"
      },
      "run_volume_tags": {
        "Name": "unicorns-{{user `env`}}-{{user `build`}}-{{user `timestamp`}}"
      },
      "tags": {
        "Name": "unicorns-{{user `build`}}-{{user `timestamp`}}",
        "Build": "{{user `build`}}",
        "Commit": "{{user `github-commit`}}"
      }
    }
  ],
  "provisioners": [
    {
      "type": "ansible",
      "playbook_file": "./playbook.yml"
    }
  ]
}
```


deploy a network with ansible

- VPC
- subnets
- route tables
- ACLs
- NATs
- security groups
- ...

deploy a cloudformation stack with ansible

```
---
- hosts: localhost
  tasks:
    - name: Create my CloudFormation stack
      cloudformation:
stack_name: : "unicorn-vpc-dev"
  regionr: : "us-east-1"
  template: t: ./cf-template.json
  args:
    template_parameters:
      KeyName: unicorns-dev
  register: stack
```



cloudformation
templates!

deploy a new VM with ansible

- hosts: localhost

vars:

vm_name: my-new-vm

vm_memory: 4

tasks:

- name: create VM

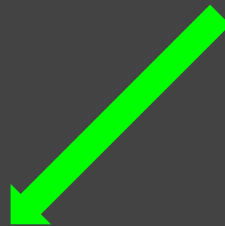
virt:

name: "{{ vm_name }}"

command: define

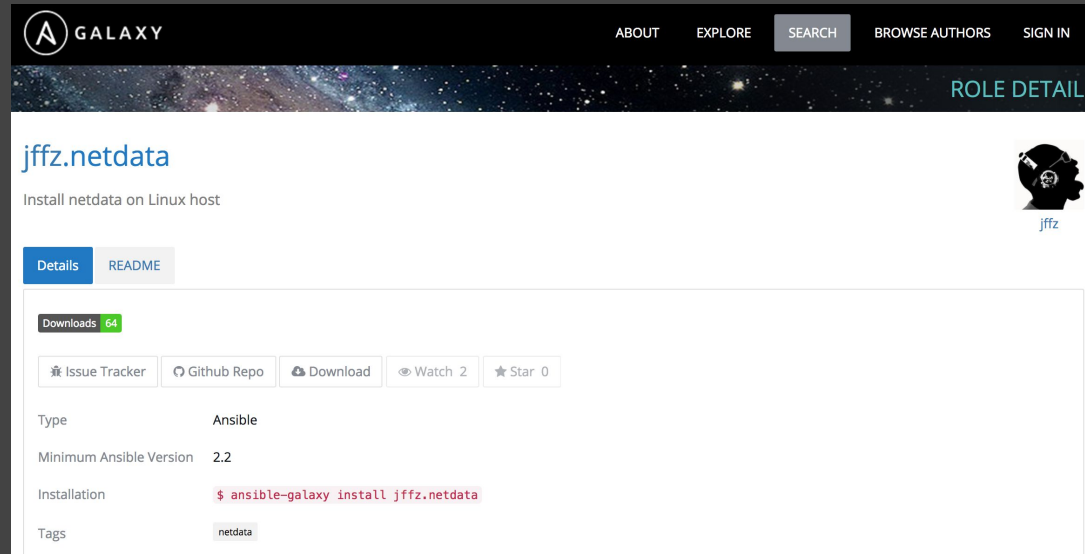
xml: "{{ lookup('template', 'vm-template.xml.j2') }}"

jinja templates!



FUN FACT:

ansible galaxy has a *ton* of playbooks and roles already written and ready to go



The screenshot shows the Ansible Galaxy interface for the role 'jffz.netdata'. The header includes the 'A GALAXY' logo and navigation links for 'ABOUT', 'EXPLORE', 'SEARCH', 'BROWSE AUTHORS', and 'SIGN IN'. A 'ROLE DETAIL' link is visible in the top right. The role name 'jffz.netdata' is displayed in blue, with a profile picture of 'jffz' to the right. Below the name, it says 'Install netdata on Linux host'. There are two tabs: 'Details' (selected) and 'README'. A green badge indicates 'Downloads 64'. Below this are buttons for 'Issue Tracker', 'Github Repo', 'Download', 'Watch 2', and 'Star 0'. The role details are listed as follows:

Type	Ansible
Minimum Ansible Version	2.2
Installation	<code>\$ ansible-galaxy install jffz.netdata</code>
Tags	netdata

FUN FACT:

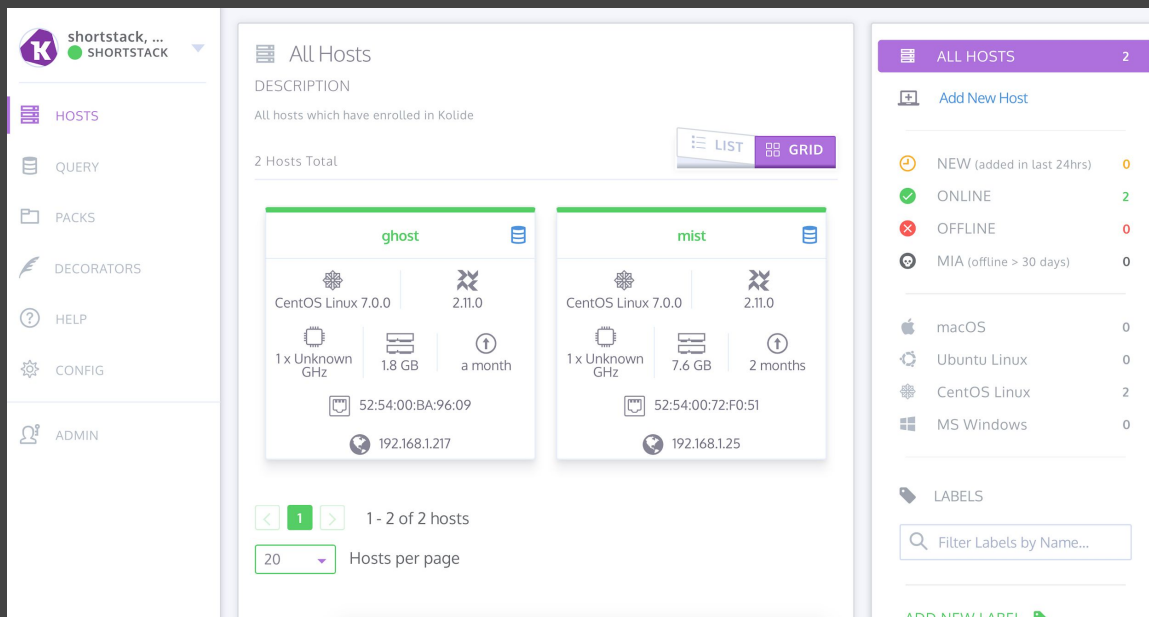
don't know how to use roles?
confused by the ansible directory
structure? FEAR NOT!

```
ansible-galaxy init $ROLE_NAME
```


deploy a kolide server

```
- hosts: kolide
  roles:
    - kolide
```

super fast.
super easy.
super shiny.



The screenshot displays the Kolide dashboard interface. On the left is a navigation sidebar with options: HOSTS, QUERY, PACKS, DECORATORS, HELP, CONFIG, and ADMIN. The main content area is titled 'All Hosts' and shows a list of 2 hosts. Two host cards are visible: 'ghost' and 'mist'. Both are running CentOS Linux 7.0.0 with kernel 2.11.0. 'ghost' has 1x Unknown GHz, 1.8 GB, and a month of uptime. 'mist' has 1x Unknown GHz, 7.6 GB, and 2 months of uptime. Both have MAC addresses starting with 52:54:00 and IP addresses 192.168.1.217 and 192.168.1.25 respectively. A right-hand sidebar shows host status statistics: NEW (added in last 24hrs) 0, ONLINE 2, OFFLINE 0, and MIA (offline > 30 days) 0. Below this is a 'LABELS' section with a search bar and a list of labels: macOS (0), Ubuntu Linux (0), CentOS Linux (2), and MS Windows (0).

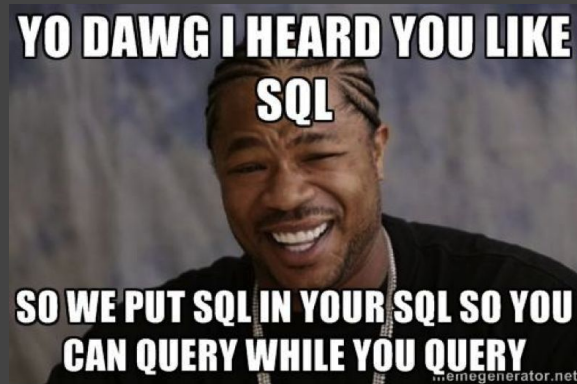
now we need osquery

New Query

Query Title

SQL

1	SELECT * FROM last;
---	---------------------



2 of 2 Hosts Returning 332 Records (0 failed) RUN

Select Targets 2 unique hosts

All Hosts X

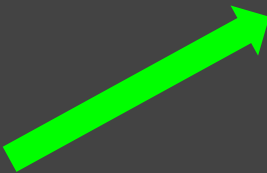
EXPORT 🗖

hostname	host	pid	time	tty	type	username
						wchampion
ghost	192.168.1.11	24417	1517845840	pts/1	7	wchampion
ghost	192.168.1.11	9941	1517845850	pts/1	7	wchampion

deploy / configure osquery daemons

```
- hosts: linux_servers
  become: yes
  become_user: root
  tasks:
    - include_vars: group_vars/agents
    - import_tasks: roles/osquery/deploy.yml
```

you can do this
from mist.io!



Add Script
You can add scripts inline or from a url. You will soon be able to schedule your scripts to run.

SCRIPT NAME
osquery

SCRIPT DESCRIPTION
install/configure osquery daemon Optional.


TYPE
Ansible Playbook Choose the type of your script. Consult the docs, on adding your scripts

SOURCE
Inline Specify the type of your script source.

Script * Copy paste your script. Make sure the script's format is aligned to the examples

ADD RESET FORM

deploy / configure openvpn server

 **openvpn**
Inline Script

[TAGS](#) [DOWNLOAD](#) [RUN ▶](#)

```
- name: "Configure OpenVPN server"
hosts: openvpn
vars:
  - env: "dev"
  - cidr: "21"

tasks:

- name: Download OpenVPN Access Server
  get_url:
    url: http://swupdate.openvpn.org/as/openvpn-as-2.5-CentOS7.x86_64.rpm
    dest: /tmp/openvpn.rpm

- name: Install OpenVPN Access Server
  shell: yum -y install /tmp/openvpn.rpm
  become: yes
  become_method: sudo

- name: Install google-authenticator
  package: name=google-authenticator state=installed
  become: yes
  become_method: sudo

- name: Change SELinux config to allow VPN tunnel
  shell: setsebool -P httpd_can_network_connect 1
  become: yes
  become_method: sudo

- name: Configure OpenVPN Access Server
  become: yes
  template:
    src: files/openvpn/config.json.j2
    dest: /usr/local/openvpn_as/etc/config.json
    owner: root
    group: root
    mode: 0644
```

ID	95cb3bfba3f9431cba20cb6d86c7e6fa
Description	install openvpn and enable 2 factor
Type	ansible

graylog

- stack
 - graylog web interface
 - elasticsearch
 - mongodb
- collector-sidecar agents on all your systems



elastic stack

- stack
 - elasticsearch
 - logstash
 - kibana
- beats log shippers on all your systems
 - filebeat, winlogbeat, etc



elastic

wazuh

- OSSEC fork
- stack
 - elasticsearch
 - logstash
 - kibana
 - wazuh kibana plugin
- OSSEC HIDS agents on all systems



tl;dr

- there are a million ways to do all of these things
- evaluate and pick the tools that are right for the job
- leverage open source where you can and recognize where you can't
- security baked in, always

the end

thank you :)