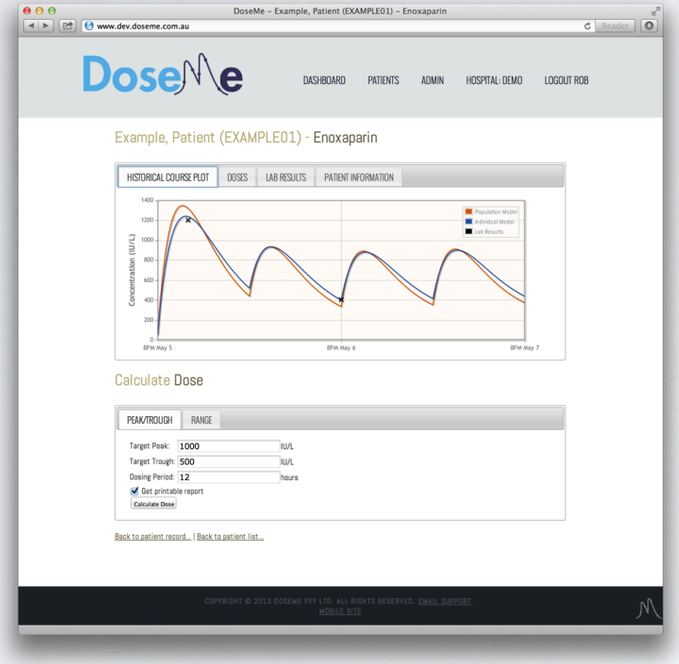
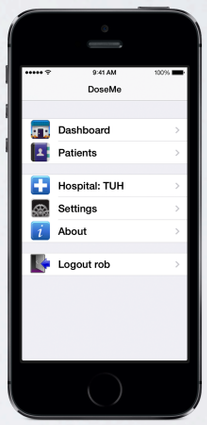
DoseMe

DevOps in a Regulated World
aka 'Ansible, AWS, and Jenkins'





www.doseme.com.au

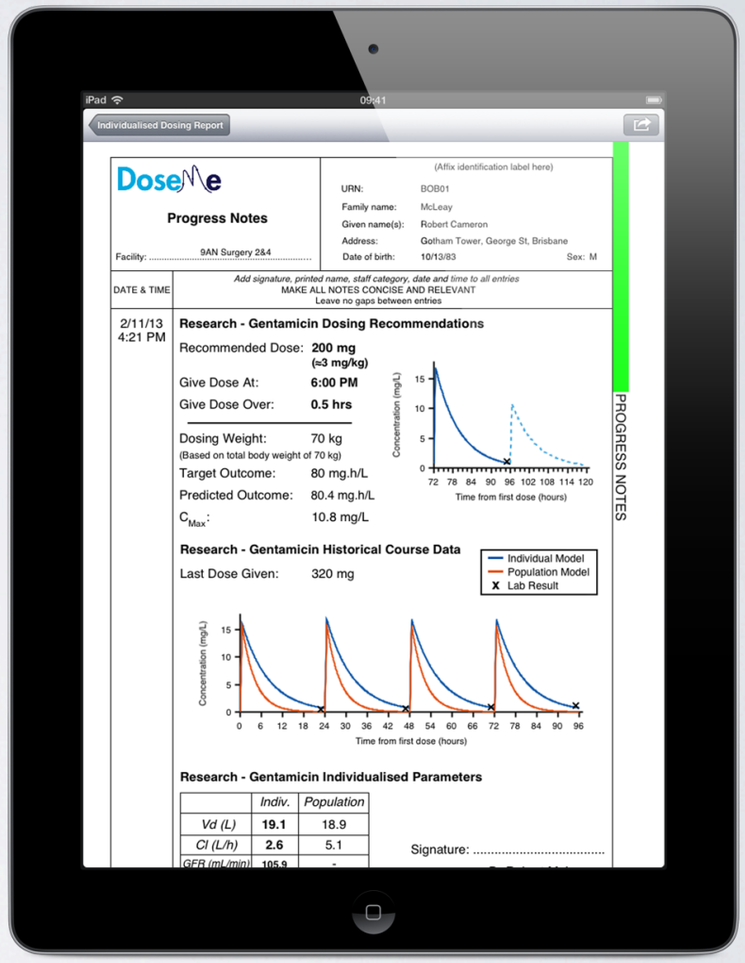
```
$ git clone git@bitbucket.org:doseme/ansible-aws-talk.git
```

Overview

- What is DoseMe?
- DevOps and medical devices
- Ansible Overview
- Ansible and AWS
- Our technology decisions and the results

What Is DoseMe?

- DoseMe: simple dose-individualisation, with:
  - iPad, iPhone, web and mobile web.
  - GP patient management software
    - HL7 integration (e.g. Pathology)
- Can dose many classes of drugs.
- Dosing individually:
  - **Increases the childhood leukaemia survival rate by 15%.**
  - **Save an average $2,500/patient on aminoglycosides.**



How Does DoseMe Work?

- It is your first day in a new job, in a new location, how much time would you give yourself to drive to work?

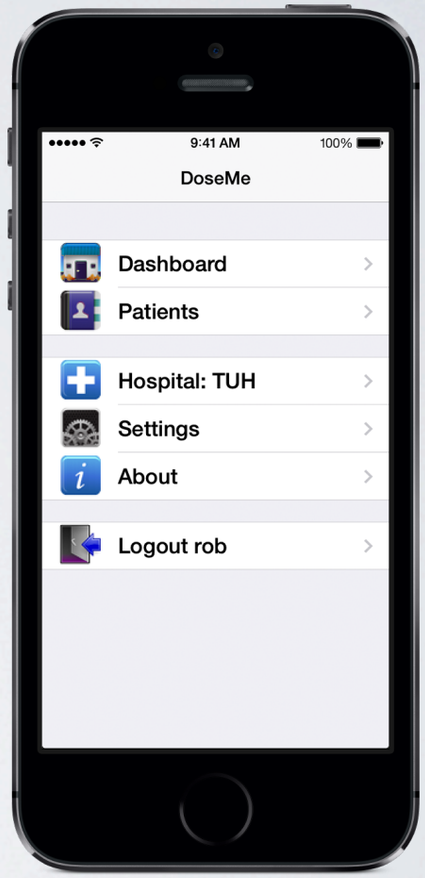
- You didn't get fired on your first day… what influences your decision on when to leave home on your second day of work?

- We can calculate your next drug dose the same way - using mathematical models of drug absorption/clearance fit to your prior data.

Interacting Medication (Drug or Class)	Risk of bleeding ↑	↓
Aminoglutethimide		++
Amiodarone	+++	
Amoxycillin	++	
Anabolic Steroids/Androgens e.g. nandrolone, oxandralone	+++	
Anticoagulants/Antiplatelets *e.g. low molecular weight heparin (enoxaparin – Clexane®), clopidogrel, aspirin, Abciximab (ReoPro®), dipyridamole, heparin, tirofiban*	+++	
Antithyroid agents *e.g. carbimazole, propylthiouracil*		++
Aprepitant		+++
Azathioprine/Mercaptopurine		++
Capecitabine	++	
Carbamazepine		+
Cephalosporins *e.g. cephazolin*	+	
Cholestyramine		++
Ciprofloxacin	+++	
Cyclosporin		+
Danazol	++	
Dicloxacillin		++
Disulfiram	++	
Fibrates *e.g. fenofibrate, gemfibrozil*	++	
5-Fluorouracil	++	
Griseofulvin		+
Imidazole antifungals *e.g. ketaconazole, miconazole*	++	
Isoniazid	+	
Leflunomide	++	
Macrolides *e.g. azithromycin, clarithromycin, erythromycin, roxithromycin*	+++	
Metronidazole	+++	
Moxifloxacin	++	
Norfloxacin	+++	
NSAIDs/COX-2 inhibitors *e.g. naproxen, celecoxib, ibuprofen*	+++	
Paracetamol *(if taking greater than 3.5 - 7g/week)*	+++	
Phenobarbitone		+++
Phenytoin	+++ (initially)	+ (long-term)
Proton Pump Inhibitors *e.g. omeprazole, esomeprazole, pantoprazole*	++	
Quetiapine	++	
Quinidine	++	

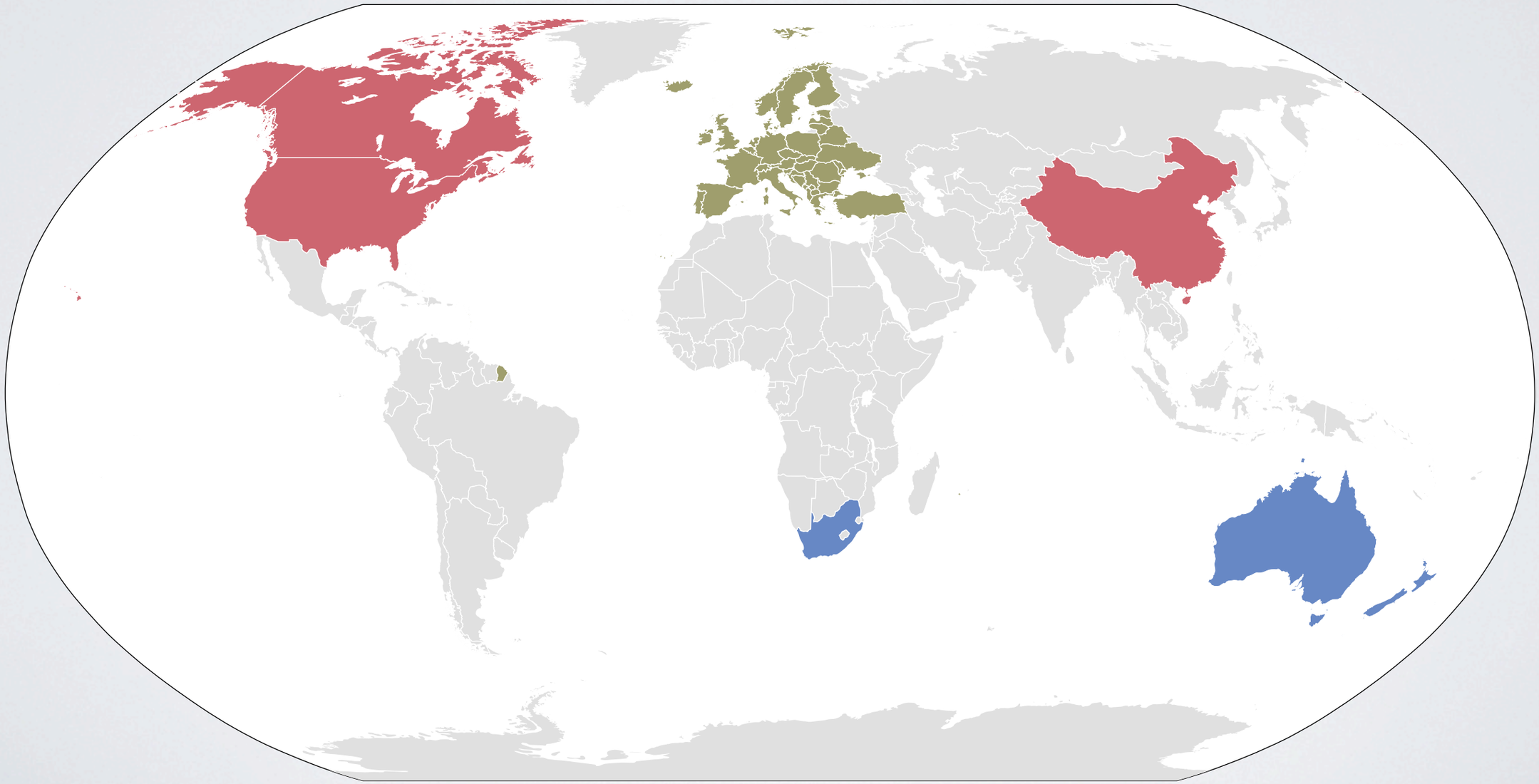
What's A Medical Device?

NEW!



DoseMe - A Medical Device?

- Not currently a medical device
- Lower classification medical device
- Higher classification medical device



How Is This Relevant To DevOps?

- Medical device guidelines require:
  - Traceability of complaints / features / fixes
  - Reproducible production environment / ‘manufacturing’
  - Consistent, maintained, and recorded production environment
  - Maintained records of production ‘manufacturing runs’
  - Regular and repeatable QA/QC testing

JIRA

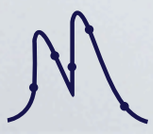
ANSIBLE

ANSIBLE + Jenkins

Jenkins

What Is Ansible?

- "IT Orchestration Engine" - Config management, deployment.
- Like CFEngine, or Puppet, except:
  - Dead-simple syntax (YAML), executed in order
  - Written in Python
  - Agentless
  - Secure - uses SSH
  - Free and Open Source (Top 10 Python Project, GitHub)
  - Backed by a commercial company (AnsibleWorks)



What Can You Do? (Modules)

- Pretty much anything:

accelerate
acl
add_host
airbrake_deployment
apt
apt_key
apt_repository
arista_interface
arista_l2interface
arista_lag
arista_vlan
assemble
assert
async_status
at
authorized_key
bigip_monitor_http
bigip_monitor_tcp
bigip_node
bigip_pool
bigip_pool_member
boundary_meter
bzr
campfire
cloudformation
command
copy
cron
datadog_event
debug
digital_ocean
django_manage
dnsmadeeasy
docker
docker_image
easy_install
ec2
ec2_ami
ec2_eip
ec2_elb
ec2_elb_lb
ec2_facts
ec2_group
ec2_key
ec2_tag
ec2_vol
ec2_vpc
ejabberd_user
elasticache
facter
fail
fetch
file
filesystem
fireball
firewalld
flowdock
gc_storage
gce
gce_lb
gce_net
gce_pd
gem
get_url
git
github_hooks
glance_image
group
group_by
grove
hg
hipchat
homebrew
hostname
htpasswd
include_vars
ini_file
irc
jabber
jboss
kernel_blacklist
keystone_user
lineinfile
linode
lvg
lvol
macports
mail
modprobe
mongodb_user
monit
mount
mqtt
mysql_db
mysql_replication
mysql_user
mysql_variables
nagios
netscaler
newrelic_deployment
nova_compute
nova_keypair
npm
ohai
open_iscsi
openbsd_pkg
openvswitch_bridge
openvswitch_port
opkg
osx_say
ovirt
pacman
pagerduty
pause
ping
pingdom
pip
pkgin
pkgng
pkgutil
portinstall
postgresql_db
postgresql_privs
postgresql_user
quantum_floating_ip
quantum_floating_ip_assoc
iate
quantum_network
quantum_router
quantum_router_gateway
quantum_router_interface
quantum_subnet
rabbitmq_parameter
rabbitmq_plugin
rabbitmq_policy
rabbitmq_user
rabbitmq_vhost
raw
rax
rax_clb
rax_clb_nodes
rax_dns
rax_dns_record
rax_facts
rax_files
rax_files_objects
rax_keypair
rax_network
rax_queue
rds
redhat_subscription
redis
rhn_channel
rhn_register
riak
route53
rpm_key
s3
script
seboolean
selinux
service
set_fact
setup
shell
slurp
stat
subversion
supervisorctl
svr4pkg
swdepot
synchronize
sysctl
template
unarchive

All of these are in core

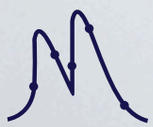
Installation

- In this talk, we use the devel branch of ansible (currently required for some AWS features).

```
$ sudo pip install paramiko PyYAML jinja2 httplib2
$ git clone git://github.com/ansible/ansible.git
$ cd ansible
$ git checkout devel

$ source ./hacking/env-setup
OR
$ sudo python setup.py install
```

- We also need boto installed to talk to EC2

```
$ sudo pip install boto
```



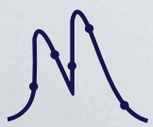
Inventory And Selecting Hosts

- Simple ini-style definitions, grouping hosts:

```
# file: server-inventory

[local]
localhost

[www]
www01.example.com
www02.example.com

[launched]
# Empty - We'll discuss later
```

- Ad-hoc commands let you select hosts or groups, using logical operators:

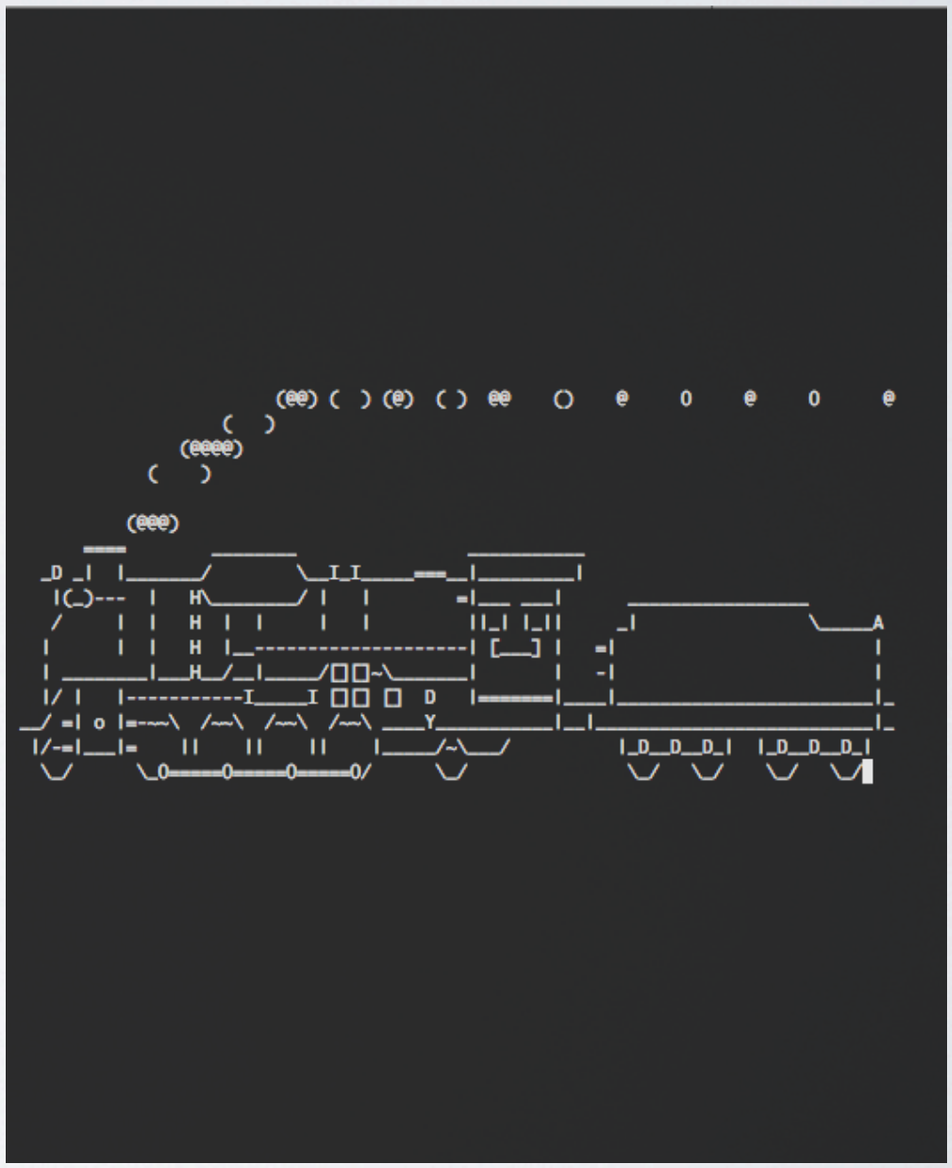
```
$ ansible -m ping www
$ ansible -m ping www:!local
```



A Simple Ad-Hoc Task

- Guaranteed to annoy your co-workers:

```
$ ansible all -i inventory_file \
   -m apt pkg=sl state=installed
```



Ansible In Production - Playbooks

- How you use ansible in production.
- Executed in order.
- Files can include other files.

```
---
# Example Playbook
- hosts: all
  vars_files:
     - vars/common.yml
  tasks:
     - include: tasks/common.yml
  handlers:
     - include: handlers/common.yml
```

Ansible In Production - Roles

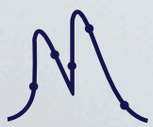
- DRY:
  - Roles allow you to reuse:
    - Tasks
    - Variables
    - Handlers
    - Files
    - Templates

Ansible In Production - Roles

```
roles/
     common/
          tasks/
          files/
          templates/
          handlers/
          vars/
          meta/
 www/
          tasks/
          files/
          templates/
          handlers/
          vars/
          meta/
```

- A `main.yml` in all of these folders will be included via a simple definition:

```
---
- hosts: www
  remote_user: admin
  sudo: True
  roles:
     - common
     - www
```



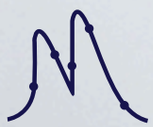
Ansible & AWS

- A sample playbook is available via git at:

```
$ git clone git@bitbucket.org:doseme/ansible-aws-talk.git
```

https://bitbucket.org/doseme/ansible-aws-talk/

- Demonstrates configuring:
  - VPC and two security groups
  - Two EC2 instances (in VPC)
  - RDS (in VPC)
  - Elastic Load Balancer
  - Installs Apache, Vim, Postfix, screen, and sudo on instances.



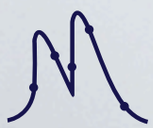
Ansible & AWS - Authentication

- Firstly, create an IAM role - e.g. belonging to a power-user group. Download the credentials, then:

- Add to $HOME/.boto:

```
[Credentials]
aws_access_key_id = <Access Key>
aws_secret_access_key = <Secret Key>
```

- Now we're ready to deploy!



Deploying EC2 Instances - 1/3

```
 - name: Provision EC2 instances - zone 1
local_action:
      module: ec2
      state: present
      id: "{{ idempotent_id }}"
      region: "{{ aws_region }}"
      keypair: "{{ aws_keypair }}"
      group: ansibleDeployedDefault
      instance_type: "{{ aws_instanceType }}"
      image: "{{ aws_image }}"
      vpc_subnet_id: "{{ vpc.subnets[0].id }}"
register: ec2
```

Deploying EC2 Instances - 2/3

- If you're running Ansible every hour, last thing that you want is to deploy another complete set of servers on every run.

```
{{ idempotent_id }}
```

- Can represent either one or a set of instances - do not reuse.
- See here for a guide:

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Run_Instance_Idempotency.html

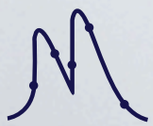
Deploying EC2 Instances - 3/3

```
 - name: Provision EC2 instances - zone 1
local_action:
      module: ec2
      state: present
      id: "{{ idempotent_id }}"
      region: "{{ aws_region }}"
      keypair: "{{ aws_keypair }}"
      group: ansibleDeployedDefault
      instance_type: "{{ aws_instanceType }}"
      image: "{{ aws_image }}"
      vpc_subnet_id: "{{ vpc.subnets[0].id }}"
register: ec2
```

VPCs And Security Groups

```
  - name: "Provision VPC in
  {{ aws_region }}"
local_action:
      module: ec2_vpc
      state: present
      cidr_block: 10.1.0.0/16
      subnets:
            - cidr: 10.1.1.0/24
              az: us-east-1a
            - cidr: 10.1.2.0/24
              az: us-east-1d
internet_gateway: True
route_tables:
            - subnets:
                  - 10.1.1.0/24
                  - 10.1.2.0/24
              routes:
                  - dest: 0.0.0.0/0
                    gw: igw
region: "{{ aws_region }}"
wait: yes
register: vpc
```

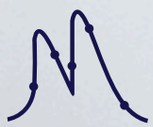
```
- name: Provision ec2 security group
  local_action:
        module: ec2_group
        state: present
        name: ansibleDeployedDefault
        description: "Default -http"
        vpc_id: "{{ vpc.vpc_id }}"
        region: "{{ aws_region }}"
        rules:
          - proto: tcp
            from_port: 80
            to_port: 80
            cidr_ip: 0.0.0.0/0
```



VPCs And Security Groups

- Each Ansible task returns a data-structure of useful information, which we can register to a variable and then use.
- -v will show you the returned output from a task in JSON.
- The docs aren't terribly clear on the data returned, so developing Ansible playbooks requires liberal use of -v.

```
    rob@zazu:~/projects/git/doseme-config-warfarin-aws$ ansible-playbook -i server_inventory site.yml
    -v

PLAY [local] ******************************************************************

TASK: [aws | Provision VPC in us-east-1] **************************************
ok: [localhost] => {"changed": false, "item": "", "subnets": [{"az": "us-east-1d", "cidr":
"10.2.2.0/24", "id": "subnet-e9f8f49d"}, {"az": "us-east-1a", "cidr": "10.2.1.0/24", "id":
"subnet-40371e06"}], "vpc": {"cidr_block": "10.2.0.0/16", "dhcp_options_id": "dopt-d8d7ddba", "id":
```

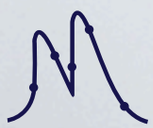


Provisioning An ELB

- Unfortunately, even devel branch Ansible doesn't yet support creating Elastic Load Balancers in VPCs (It does support adding hosts to them).

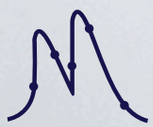
- We're working on a patch internally, and hope to have this working and submitted soon!

- There's examples in the talk code repository, but I won't discuss this here.



Provisioning A Multiple A-Z RDS

```
 - name: Provision RDS
local_action:
      module: rds
      command: create
      region: "{{ aws_region }}"
      multi_zone: yes
      subnet: "{{ aws_db_subnetgroup }}"
      vpc_security_groups: "{{ rdsvpc.group_id }}"
      instance_name: "{{ aws_db_name }}"
      db_engine: "{{ aws_db_engine }}"
      size: "{{ aws_db_size }}"
      instance_type: "{{ aws_db_instanceType }}"
      username: "{{ aws_db_username }}"
      password: "{{ aws_db_password }}"
register: rds
```



Ok, So Now What?

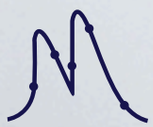
- Remember the empty host group "launched"?

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

- Even though it was empty earlier, we still assign a role to it:

```
- hosts: launched
  remote_user: admin
  sudo: True
  roles:
    - role: common
```

- Result: Role is applied – instance deployed and configured!



How Do We Automate Running Of Ansible Configurations?

- Not going to discuss Jenkins in detail, but we can trigger jobs to run based on:
  - Git/Hg/SVN Commits
  - Periodically
  - As a dependent task
  - Manually

Jenkins

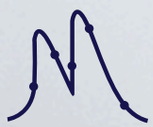
Plugging This Into Jenkins

- We use Perl unit tests to test our playbooks in Jenkins upon commit - the test script could be as simple (and bad) as:

```
#!/usr/bin/perl
use Test::More tests => 1;

my $result = `ansible-playbook -i servers --syntax-check site.yml`;
chomp $result;
ok($result eq 'Playbook Syntax is fine', "Syntax check of playbook");
```

- This can then be run in Jenkins as:

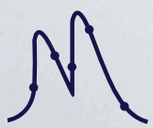
```
prove --formatter=TAP::Formatter::JUnit > jenkins-${JOB_NAME}-${BUILD_NUMBER}-junit.xml
```



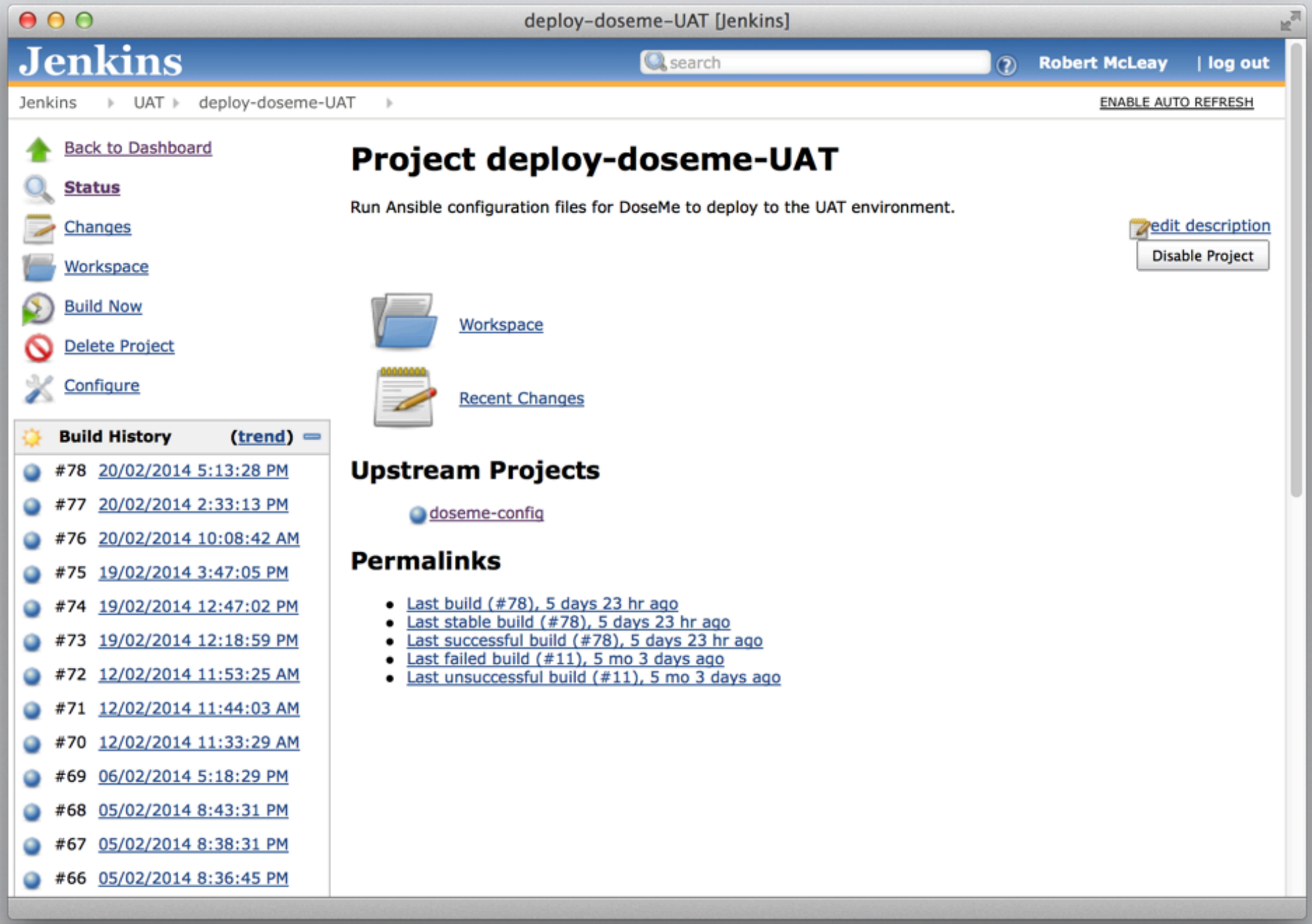
Continuous (Server) Deployment

- Following a configuration check in Jenkins, we then run the playbook. This gives us a record of:
  - Changes made to the playbook (by who and when)
  - Any changes made to a host.
  - Full console output of running Ansible.
  - Summary of actions taken - e.g.

```
PLAY RECAP ********************************************************************
yyy.zzz.doseme.com.au : ok=37   changed=2    unreachable=0    failed=0

Finished: SUCCESS
```

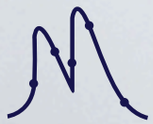


Continuous (Server) Deployment



Challenges - Ansible, AWS, & Jenkins

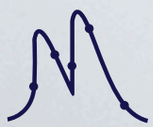
- We started using Ansible early (pre-1.0, from memory)
  - The syntax has changed (even how you use variables!)
  - Roles didn't exist then (DRY was harder)
- Still missing support for some features:
  - The ec2_elb_lb module doesn't support VPCs.
  - rds and ec2 modules in release are lacking some features.
- We can't do continual deployment to production.



Benefits Of Ansible We've Found

- We don't use Ruby, so we don't need to deploy Ruby.
- We don't have to worry about agent deployment.
- The Python modules are relatively easily extendable.

- It was easy to start with a simple playbook and grow it as the company has – no complex setup required to begin.



Questions?

```
$ git clone git@bitbucket.org:doseme/ansible-aws-talk.git
```

http://docs.ansible.com

http://boto.readthedocs.org/en/latest/ec2_tut.html