Cornell University

Student Services IT

Class Roster

Moving to Amazon Web Services with Ansible

ANSIBLE

by Red Hat®

amazon web services™

Jenkins

Supported Customers

Division of Student and Campus Life (SCL)

Athletics & Physical Education	Dean of Students
Career Services	Gannett Health Services
Chimes	Public Service Center
Campus Life Enterprise Services (Cornell Store, Housing, Dining...)	Student Disability Services
Cornell Commitment	... and more!

Vice Provost for Enrollment

University Registrar	Student Employment
Financial Aid	Undergraduate Admissions
Graduate School	... and more!

Class Roster

- Official schedule of classes
  classes.cornell.edu

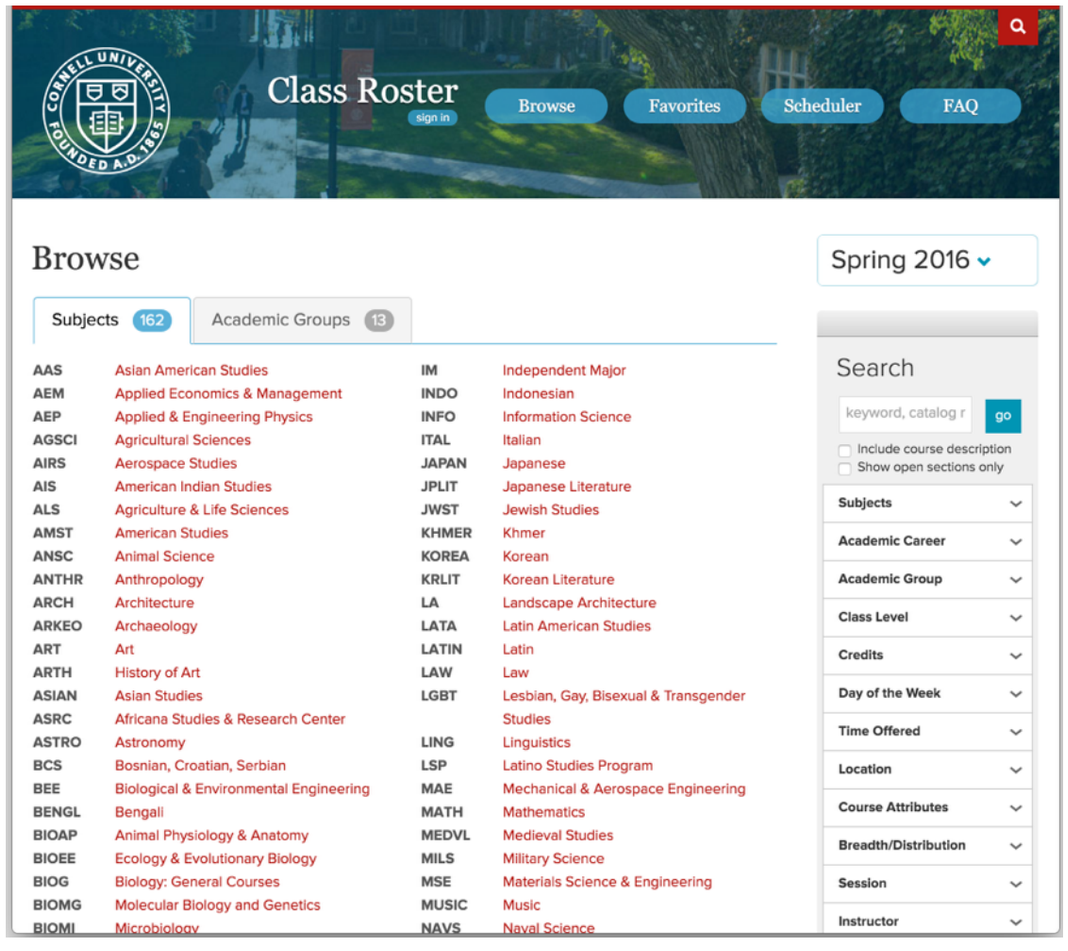
- Office of the University Registrar

- October 2014
  Developed and launched on-premises

- February 2016
  AWS migration

- March 2016
  Major upgrade - "Scheduler"

- Usage - pre-enrollment spikes



Class Roster

Before AWS

- CIT Server Farm - 3 extra-tier Prod VMs [2 CPU, 8GB RAM]
  - 2 app servers - PHP 5-5, httpd-2.2, mod_fcgid
  - 1 db server - MySQL 5.6 + MongoDB 2.6 (adjacent)
- Big IP Load Balancer
- Failover to SSIT Sorry Services on CIT Static Hosting
- Deployments (manual) via homegrown bash script
- Monitoring - OpsView w/limited notifications
- Logging - local VMs only

Class Roster

AWS - Feb 2016

- Dual account approach - dev/test & prod isolated
- Production
  - 2+ app servers - Apache 2.4, CUWebAuth 2.4, PHP-FPM 5.6 [c4.large]
    - ELB; Auto Scaling Group; scheduled ASG actions
  - 1 RDS db server - MySQL 5.6 [db.m4.large]
  - 1 db server - MongoDB 3.2 - [m4.large]
- Route 53 w/failover records to our new HA "StaticWeb" service
  - ELB; 2 x t2.nano
- Jenkins - Automatic deployments (w/Ansible)
- Monitoring - Amazon CloudWatch & Pingdom
- Logging - PaperTrail & ELB Logging to S3
- + "standard" Cloudification services setup - NAT Gateway, CloudTrail, VPN, etc.

Ansible & Class Roster

- Provision Most* AWS Resources
  - Maintain Security Groups
  - Launch EC2 & RDS Instances
  - Create ELBs
  - Maintain R53 Records
  - Create AMIs
  - Create Launch Configs
  - Create/Update ASGs
  - Dev/Test Start/Stop
- Configuration Management - EC2 Instances
- App Deployments & Continuous Delivery
- Patching*

Class Roster

What *is* Ansible?

- IT Automation Engine
  - nodes + modules
- Python + YAML
- Static + Dynamic Inventory
- Playbooks, Plays, Roles, Tasks
- Modules, Plugins, Filters
  - Extensive out of the box; easily extendable
- Templates - Jinja2
- Vault
- Ansible Galaxy - community shared roles

Why Ansible?

- Low barrier to entry
- Agentless
- Modern, powerful, and growing community
- "Batteries Included" (450+ modules) + Ansible Galaxy
- Broad compatibility*
- Infrastructure as code

Getting Started

Many ways to install, one of easiest is with pip

```
sudo easy_install pip
sudo pip install ansible
```

boto/aws-cli compatible credentials

Inventory Options

1. Static

   - Nodes (hosts) and groups defined in config file

2. Dynamic

   - Collected via script w/optional cache, ex. ec2.py

3. Static + Dynamic

   - Best of both

localhost is a node too -- useful for AWS modules

Class Roster

Playbook, Plays, Tasks, Roles

```
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: ec2-user

  pre_tasks:

  roles:

  post_tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes

  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

Class Roster

Playbook, Plays, Tasks, Roles

Play

```
- hosts: webservers
  roles:
  - ec2-config

  - phpweb-config
```

Role

Playbook

```
- hosts: imageservers
  pre_tasks:
  - name: install imagick
    yum: name=imagick state=latest

  - name: set file ownership
    file:
      path:  /etc/foo.conf
      state: touch
      mode:  "u=rw,g=r,o=r"

  roles:
  - httpd
```

Task

Task

Class Roster

Modules, Plugins, Filters

- 450+ core/extra modules

```
cron
ec2
ec2_ami
git
group
s3
template
user
yum
```

- custom modules & filters

```
ec2_elb_tag
ec2_vol_find_volume_id
```

Class Roster

Task

roles/phpweb-config/tasks/sites.yml

```
...
- name: Get Holding ID Keytabs from S3
  s3:
    bucket: "{{ ansible_s3_bucket }}"
    object: "/cuwebauth/keytabs/https.{{ item.value.servername }}.keytab"
    dest: "/etc/httpd/conf/https.{{ item.value.servername }}.keytab"
    mode: get
  with_dict: "{{ holding_ids }}"
...
```

group_vars/tag_Name_web_demo/vars.yml

```
holding_ids:
  classes:
    group: classes
    comment: "Class Roster Demo"
    fpm_port: 9000
    servername: ansible-demo.training.cucloud.net
```

Cornell University

Class Roster

Live Demo

1. Provision two EC2 Instances in sep AZs
2. Configure instances
3. Deploy an app
4. ELB + Route 53

Cornell University

Class Roster

… and more

- Playbook check mode, tags, limit
- Ad-hoc commands
- Ansible Tower - commercial "central dashboard" offering

Cornell University

Class Roster

Questions?

github.com/ericgrysko/ansible-classroster-presentation