redhat.

ANSIBLE INTRODUCTION WORKSHOP

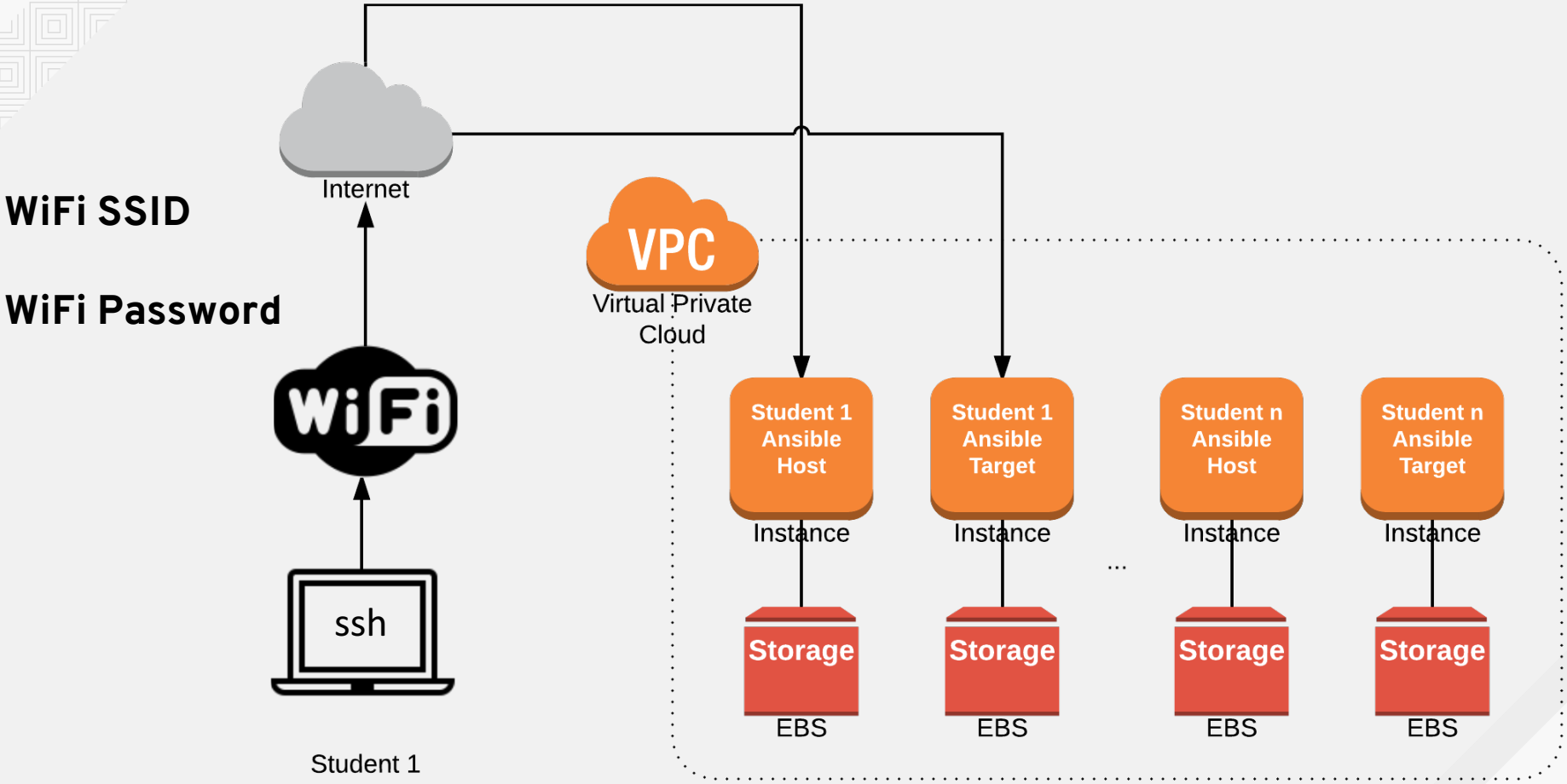
An introduction to Ansible using AWS

Jason Callaway
Red Hat Principal Solutions Architect
jcallawa@redhat.com | @jasoncallaway | jasoncallaway.com

AGENDA

Speak up if this isn't what you thought it would be!

1. Review of workshop environment and how to connect
2. Remote administration, file management, package management
3. Fancy config options for Ansible and SSH
4. Playbooks
5. Finding the right module
6. EC2 dynamic inventory
7. Roles and Ansible Galaxy
8. How to scale it and make it enterprise-ready


WiFi SSID
WiFi Password
Internet
VPC
Virtual Private Cloud
WiFi
ssh
Student 1
Student 1 Ansible Host
Student 1 Ansible Target
Student n Ansible Host
Student n Ansible Target
Instance
Instance
Instance
Instance
...
Storage
Storage
Storage
Storage
EBS
EBS
EBS
EBS

ACCESSING THE WORKSHOP ENVIRONMENT

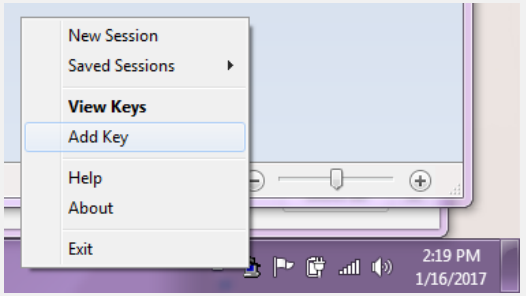
Everything happens over SSH

- Download the keys from https://jasoncallaway.com/workshop
  - Username: redacted, Password: redacted
  - Windows: redacted
  - Linux and macOS: redacted
- Linux and macOS: No additional configuration required, use default terminal application
- Windows, download putty.zip

SETTING UP PUTTY ON WINDOWS

Step 1 of 3

1. Extract putty.zip
2. Double-click PAGENT.EXE
3. Right-click the PuTTY Agent icon in the system tray
4. Select Add Key
5. Find and select the ansible_workshop.ppk file that you downloaded
6. It will seem like nothing happened, but don't worry, it was added



SETTING UP PUTTY ON WINDOWS

Step 2 of 3

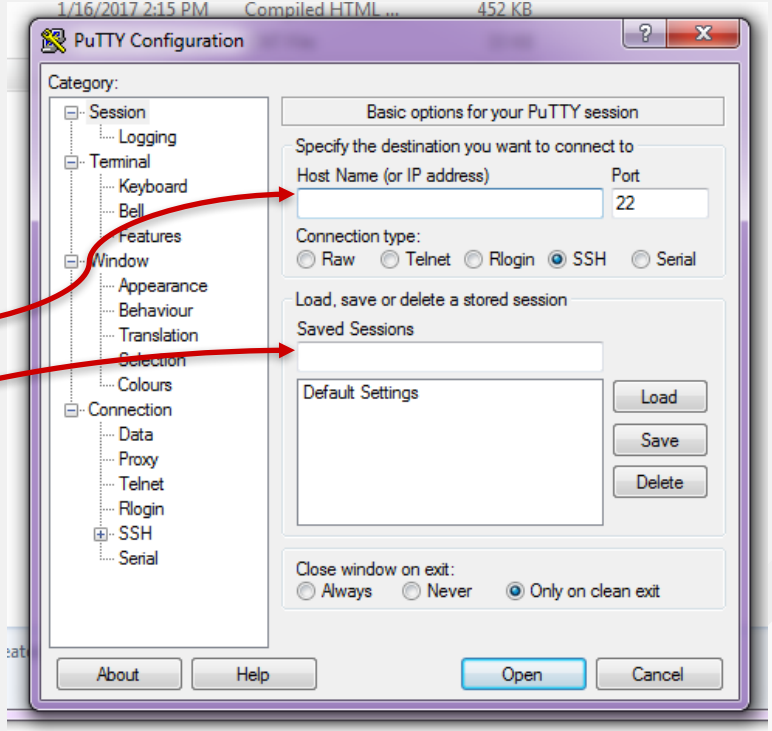
1. Double-click PUTTY.EXE
2. Enter the following into the Host Name field:

```
ec2-user@student-1.workshop.rhtps.io
```

3. Give it a name in the Saved Sessions field

```
student-1
```

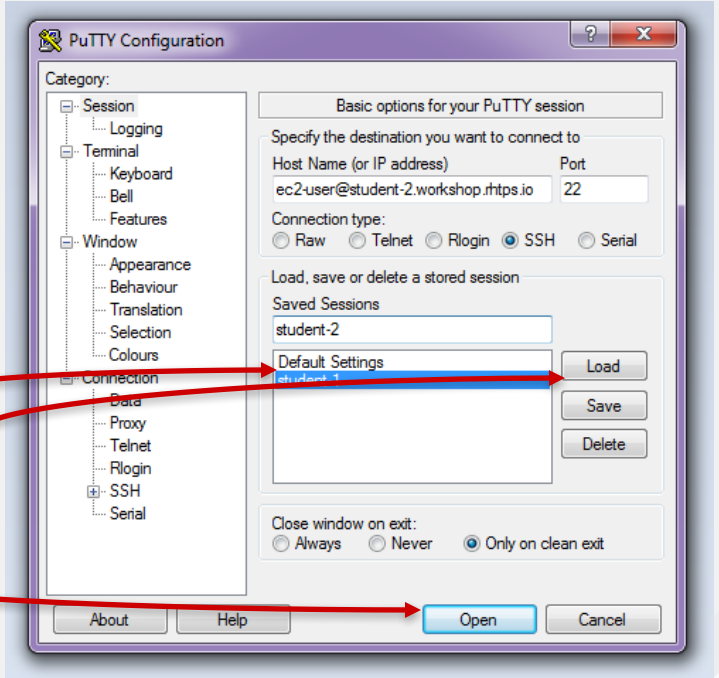
4. Click Save



SETTING UP PUTTY ON WINDOWS

Step 3 of 3

1. Repeat the previous step for the next student-n (i.e., 1 and 2, 3, and 4, etc…)
2. To open SSH sessions, select a saved session name
3. Click Load
4. Click Open



SSH-ING FROM LINUX OR MACOS

Step 1 of 1

- On macOS
  - Type ⌘-space
  - Type “terminal”
  - Enter
- On Linux, open a terminal
- Use this command for both

```
ssh -i ~/Downloads/ansible_workshop.pem ec2-user@student-1.workshop.rhtps.io
```

TERMIAL MULTIPLEXING

Never worry about timeouts

- TMUX is your friend
- CTRL-B then D to disconnect

```
[root@ip-192-168-0-150 ~]# tmux new-session -s myname
```

- To reconnect

```
[root@ip-192-168-0-150 ~]# tmux attach -t myname
```

INSTALLING ANSIBLE

Ansible Core installation

- Ansible Core is free. You get it by being a RHEL customer.
- It's best-effort support.
- You can install it from EPEL.

```
$ ssh -i ~/Downloads/ansible_workshop.pem ec2-user@student-0.workshop.rhtps.io
(type yes to accept the ssh fingerprint)
[ec2-user@ip-192-168-0-150 ~]$ sudo su -
[root@ip-192-168-0-150 ~]# ls /root/cheats
[root@ip-192-168-0-150 ~]# yum install -y ansible
```

REMOTE ADMINISTRATION

Running adhoc commands

- First, you set up your hosts file

```
[root@ip-192-168-0-150 ~]# echo "student-1.workshop.rhtps.io" > /etc/ansible/hosts
[root@ip-192-168-0-150 ~]# echo "student-2.workshop.rhtps.io" >> /etc/ansible/hosts
```

- Now let's run some commands

```
[root@ip-192-168-0-150 ~]# ssh-agent bash
[root@ip-192-168-0-150 ~]# ssh-add ./ansible_workshop.pem
[root@ip-192-168-0-150 ~]# ansible all -u ec2-user -a "whoami"
[root@ip-192-168-0-150 ~]# ansible all -u ec2-user --become -a "whoami"
```

FANCY CONFIG OPTIONS

Configuring Ansible and the SSH client

- /etc/ansible/ansible.cfg and ./ansible.cfg

```
[defaults]
remote_user = ec2-user
host_key_checking = False
```

- ~/.ssh/config

```
Host *
  User ec2-user
  ForwardAgent yes
  StrictHostKeyChecking no
  IdentityFile ~/path/to/key.pem
```

- http://docs.ansible.com/ansible/intro_configuration.html#getting-the-latest-configuration

REMOTE ADMINISTRATION

Using modules adhoc

- **Install a package**

```
[root@ip-192-168-0-150 ~]# ansible all --become -m yum -a "name=vim state=present”
```

- **Make a directory**

```
[root@ip-192-168-0-150 ~]# ansible all --become -m file -a ”path=/etc/ansible
state=directory"
```

- **Manage files**

```
[root@ip-192-168-0-150 ~]# ansible all --become -m copy \
  -a "src=/etc/ansible/hosts dest=/etc/ansible/hosts owner=root mode=644"
```

- **Why didn’t we specify `-u ec2-user` that time?**

PLAYBOOKS

Ansible's configuration, deployment, and orchestration language

- Example playbook (1 of 2)

```
# cd cheats; ansible-playbook example.yml
---
- hosts: webservers
  become: yes
  vars:
    my_name: jason
  tasks:
    - name: ensure apache is at the latest version
      yum: name=httpd state=latest
    - name: start the httpd service
      service: name=httpd state=started enabled=yes
    - name: populate index.html
      template: src=index.j2 dest=/var/www/html/index.html owner=apache group=apache mode=644
```

PLAYBOOKS

Ansible's configuration, deployment, and orchestration language

- Example playbook (2 of 2)

```
# ansible-playbook example.yml
```

- Why didn't that do anything? You need a webservers group in /etc/ansible/hosts

```
[webservers]
```

- Re-run, verify that it worked.
- It's ok to re-run as many times as you want. Idempotence!
- Open a browser, go to http://student-n.workshop.rhtps.io

PLAYBOOKS - EXERCISE

Enhance your playbook

- Add the following capability
  - Install the firewalld service
    - Hint: when looking for a module, Google “ansible module-name”
  - Start the service and make it persist
  - Add the http service to firewalld
- Extra credit
  - Add htaccess protection to /var/www/html
    - Hints: install python-passlib on target systems, you’ll to use the following modules: yum, htpasswd, copy, file, replace, lininfile, service

DYNAMIC INVENTORIES

You don’t have to maintain a static hosts file

http://docs.ansible.com/ansible/intro_dynamic_inventory.html#example-aws-ec2-external-inventory-script

- Ability to talk to the C2S AWS APIs is super-helpful
- You can refer to hosts by tags
- Must happen from inside the AWS boundary, or you’ll have to deal with CAP
- Demo

REDHATGOV NIST 800-53 ROLE

Quickly STIG your instances

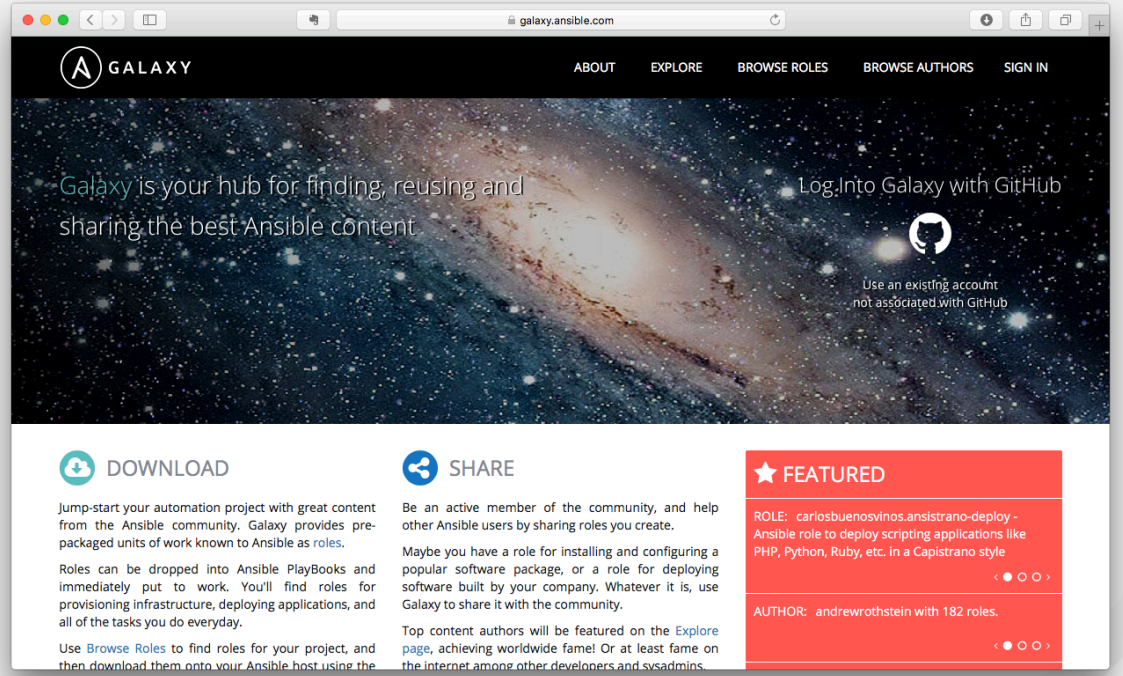
- Applying the STIG to implement NIST 800-53 security controls is a snap with the RedHatGov 800-53 Role
- https://github.com/RedHatGov/ansible-role-800-53

```
---
-  hosts: webservers
   become: yes
   gather_facts: yes
   roles:
     - ansible-role-800-53
# ansible-playbook 80053.yml
```

ANSIBLE GALAXY

yum for Ansible

Check out:
galaxy.ansible.com/rhtps



TOWER EXPANDS AUTOMATION TO YOUR ENTERPRISE.

CONTROL

Scheduled and centralized jobs

KNOWLEDGE

Visibility and compliance

DELEGATION

Role-based access and self-service

SIMPLE

Everyone speaks the same language

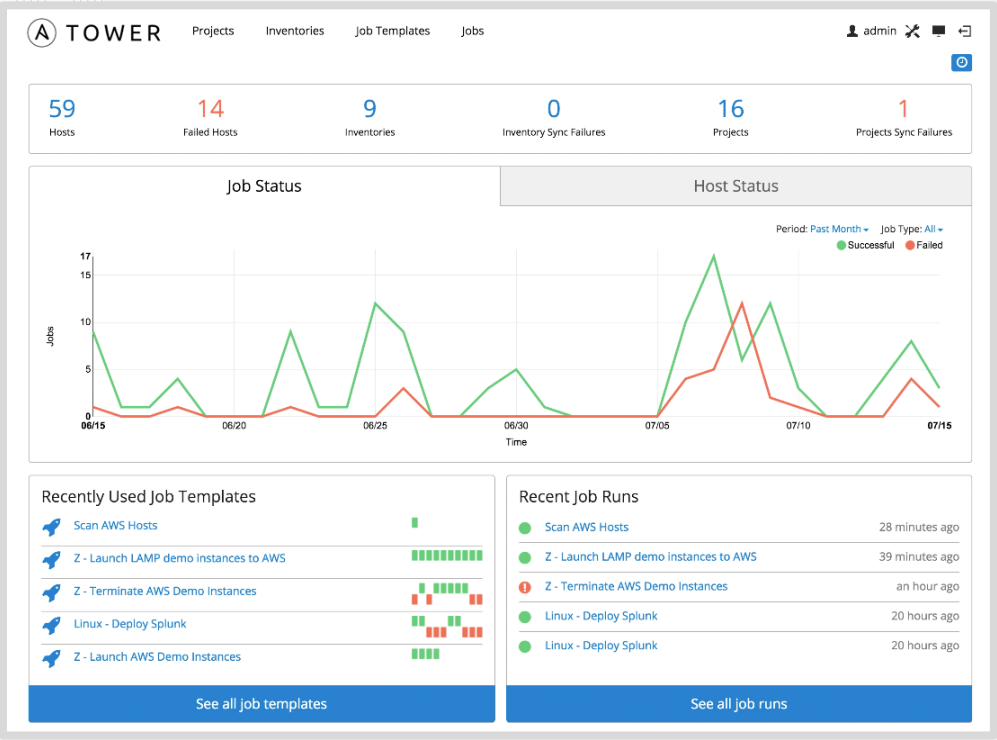
POWERFUL

Designed for multi-tier deployments

AGENTLESS

Predictable, reliable, and secure

AT ANSIBLE'S CORE IS AN **OPEN-SOURCE** AUTOMATION ENGINE.



Ansible tower is an **enterprise framework** for controlling, securing and managing your Ansible automation – with a **UI and restful API.**

- **Role-based access control** keeps environments secure, and teams efficient.
- Non-privileged users can **safely deploy** entire applications with **push-button deployment** access.
- All Ansible automations are **centrally logged,** ensuring **complete auditability and compliance.**

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos