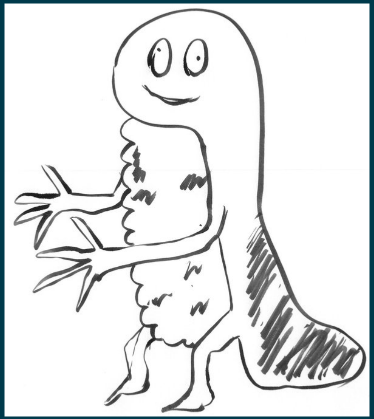
Ansible - Automation for Everyone!

Introduction about Ansible Core

Hideki Saito
Software Maintenance Engineer/Tower Support Team
2017.06

Who am I

- Hideki Saito <hsaito@redhat.com>
  - Software Maintenance Engineer / Red Hat
  - Work for Ansible Tower Support Team
  - I love Ansible, OpenStack and Beer :)
  - Twitter: @saito_hideki



Agenda

- Ansible Core Introduction
- Demo’s
  - Let’s play with Ansible Core
    - Getting Started
    - Ad-Hoc command
    - Playbooks
- Ansible Tower by Red Hat

Motivation and Proposition

Automate routine work to operate IT system.

- Let's start with where we can automate easily.
- Let's start automation using script language that anyone can easily understand.

- Education and training for programming take a lot of time.
- The IT system includes various kinds of hardware / software.

AUTOMATION FOR EVERYONE

- Ansible is an IT automation tool
- Goals are simplicity and ease-of-use
- Managing target via SSH transportation
- Management steps is written by YAML
- New release is provided approximately every 2 months

Ansible Core

Ansible Core is command-line IT automation Tool and libraries

Introduce following components of Ansible Core:

1. Command Line Tools
2. Playbooks
3. Inventory
4. Modules
5. Plugins

COMMAND LINE TOOLS

Ansible Core contains some command line tools. Following 2 commands are able to control your target hosts.

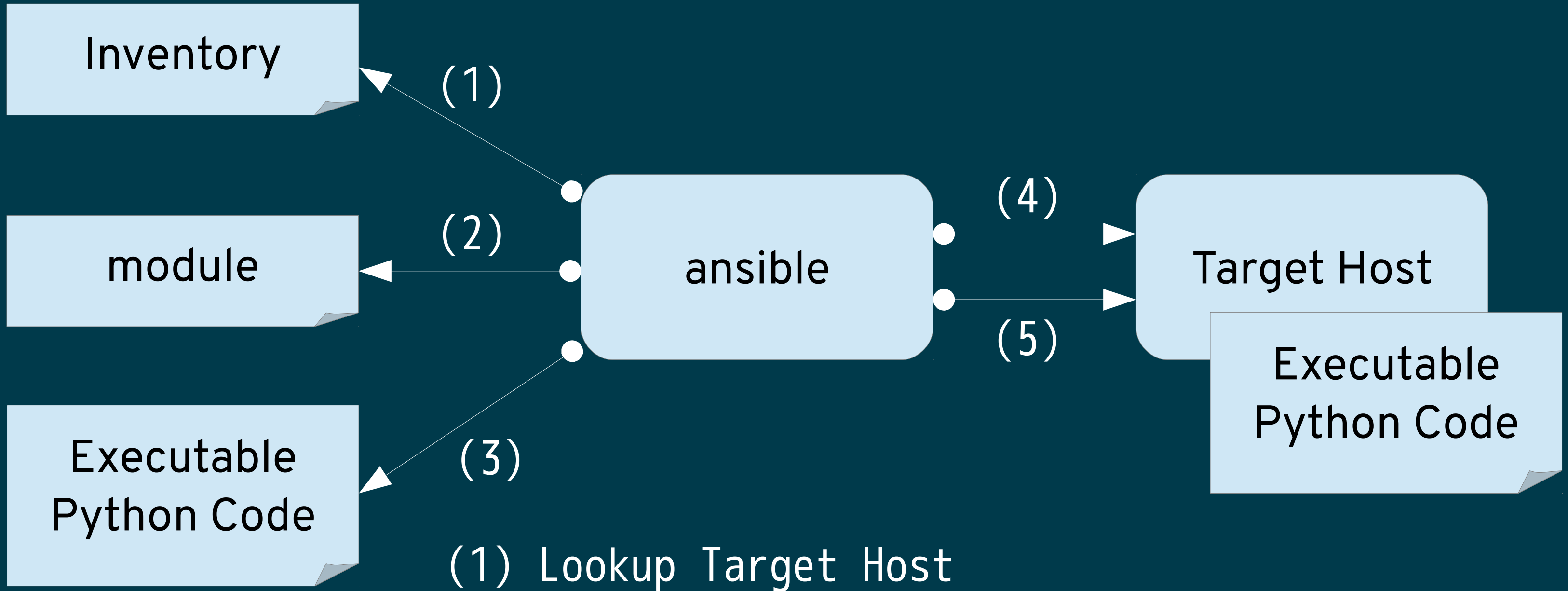
1. ansible command

```
[Usage] ansible %Target% -i %Inventory% -m %Module%
$ ansible www -i inventory -m ping

[Usage] ansible %Target% -i %Inventory% -a %Ad-Hoc Command%
$ ansible www -i inventory -a  "/sbin/reboot"
```

2. ansible-playbook command

```
[Usage] ansible-playbook -i %Inventory% %Playbook%
$ ansible-playbook -i inventory playbook.yml
```

COMMAND MECHANISM



```
(1) Lookup Target Host
(2) Read Module
(3) Generate executable code from Module
(4) Copy Executable python code to via SCP
(5) Execute python code on Target Host
```

PLAYBOOKS

Playbooks are Ansible’s configuration, deployment, and orchestration language. You can write Playbooks easily by YAML.

```yaml
---
- hosts: www
  vars:
    new_name: ansible-host1
  tasks:
  - name: get hostname
    shell: hostname
    register: result
  - name: set hostname
    hostname:
      name: "{{ new_name }}"
    notify: show hostname
  handlers:
  - name: show hostname
    debug:
      msg: "before={{ result.stdout }} after={{ new_name }}"
```

INVENTORY (STATIC)

Ansible is able to working against multiple system at the same time.

You can select portions of systems listed in the inventory at running time.

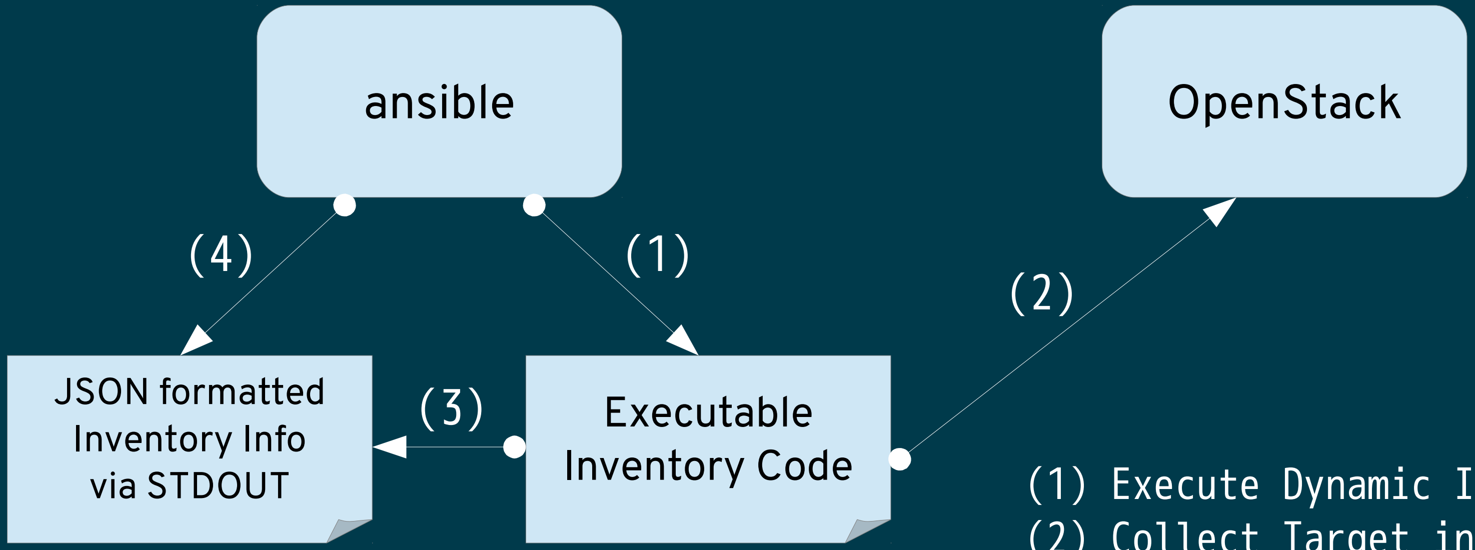
```
[localhost]
127.0.0.1

[staging]
192.168.0.1
192.168.0.2

[production]
www1.example.com
www2.example.com

[vars:local]
ansible_connection=local
```

INVENTORY (DYNAMIC)

Ansible easily supports all of these options via an external inventory system.

For example: OpenStack, AWS, GCE or something like that.

You can look these dynamic inventories at https://goo.gl/knXn3c


ansible
OpenStack
(4)
(1)
(2)
(3)
JSON formatted Inventory Info via STDOUT
Executable Inventory Code

```
(1) Execute Dynamic Inventory
(2) Collect Target information
(3) Output Inventory to STDOUT
(4) Read Inventory Information
```

MODULES (1)

Ansible has a lot of modules that can be executed directly on remote hosts or through Playbooks. You can see module index at https://goo.gl/yCGC4U

Group	Target	Group	Target
Cloud	AWS, GCE, Azure, OpenStack etc...	File	file, template, stat, unarchive etc...
Clustering	K8S, Pacemaker etc...	Identity	FreeIPA, OpenDJ
Commands	command, shell, expect etc...	Inventory	Add group and host to inventory
Crypto	openssl	Messaging	RabbitMQ
Database	MySQL, PostgreSQL, MSSQL etc ...	Monitoring	datadog, logstash, nagios etc...

MODULES (2)

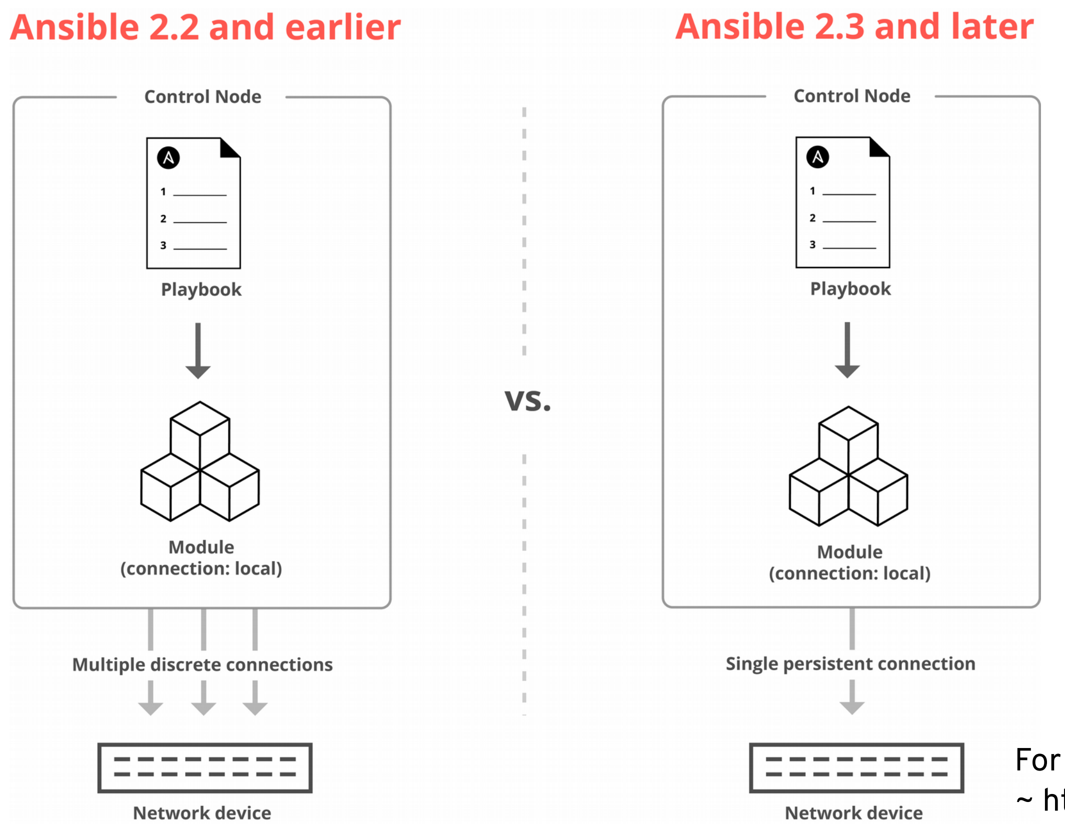
Group	Target
Net Tools	haproxy, nmcli, ldap, get_url etc…
Network	Bigswitch, Cumulus, Eos, IOS. Junos etc …
Notification	hipcat, irc, slack etc…
Packaging	rpm, yum, npm, apt etc…
Remote management	HP iLO, IPMI etc…
Source control	git, github, gitlab, hg, subversion etc …

Group	Target
Storage	NetApp, zfs etc…
System	user, group, service, puppet :) etc…
Utilities	Helper, Logic
Web infrastructure	apache, nginx, tower etc…
Windows	IIS, acl, package etc…

PLUGINS

Plugins are pieces of code that augment Ansible's core functionality.

You can easily write your own. Please see: https://goo.gl/ZQ9hvb



For example: connection plugin
~ https://goo.gl/rLha4L ~

DEMO’S

- Getting Started
  - Installation
- Ad-Hoc command
- Playbooks

Simple can be harder than complex. You have to work hard to get your thinking clean to make it simple.

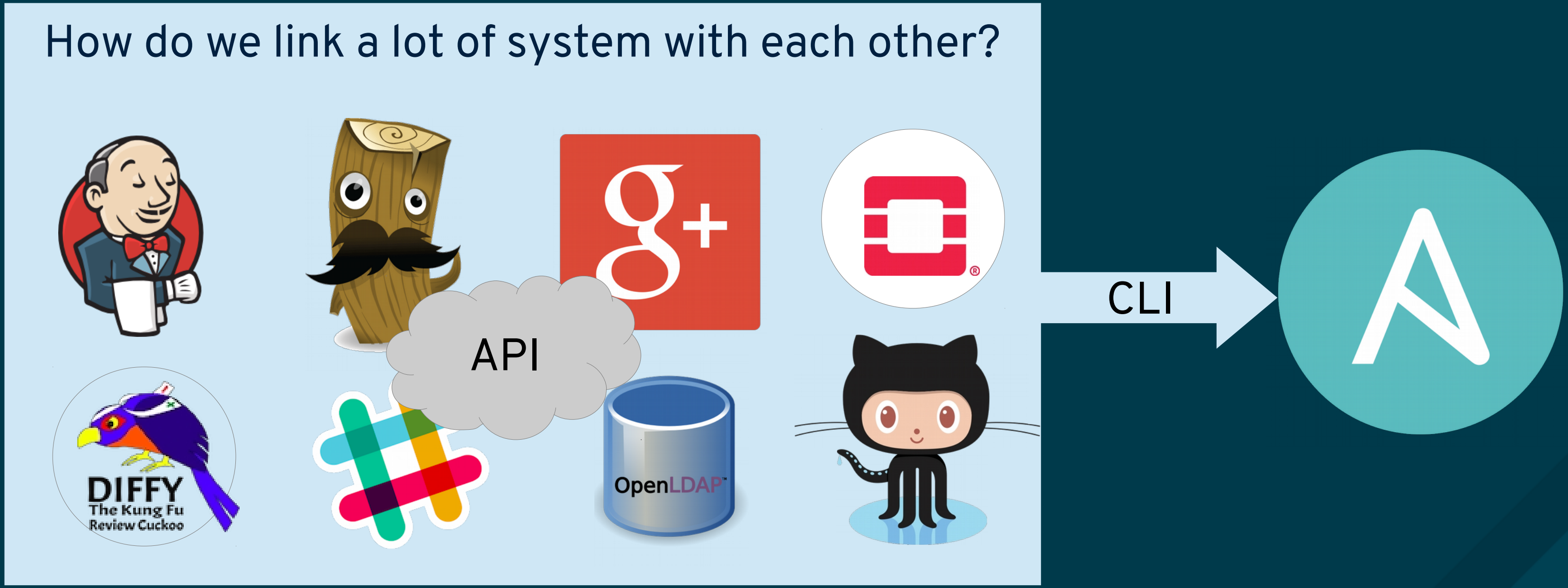
But it's worth it in the end because once you get there, you can move mountains.

~ Steve Jobs ~

Beyond the Core

What should we do the next-step?

Building an IT automation process as simple as possible. But ANSIBLE Core does not provide enough functions to advance IT automation to the next step. It does not provide API based control mechanism.


How do we link a lot of system with each other?
API
DIFFY
The Kung Fu Review Cuckoo
OpenLDAP
CLI

Ansible Tower by Red Hat

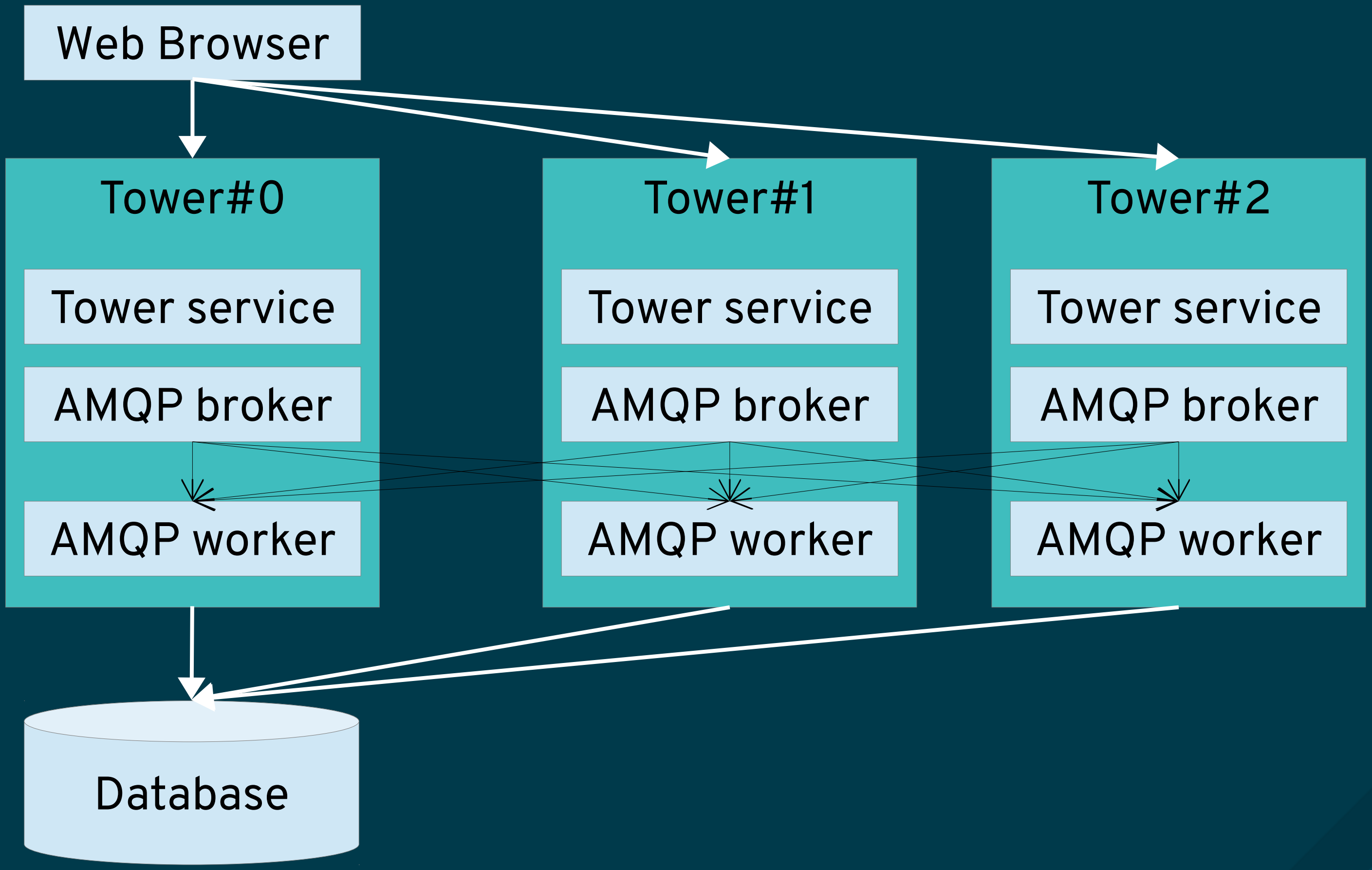
Ansible Tower is a web-based solution that is It’s designed to be the hub for all of your automation tasks.

Introduce following Tower functions:

1. Overview
2. Job Template / Work-flow /Callback
3. Web based Dashboard
4. RESTful API
5. Isolation, Consolidation and Cooperation

Ansible Tower - Architecture Design

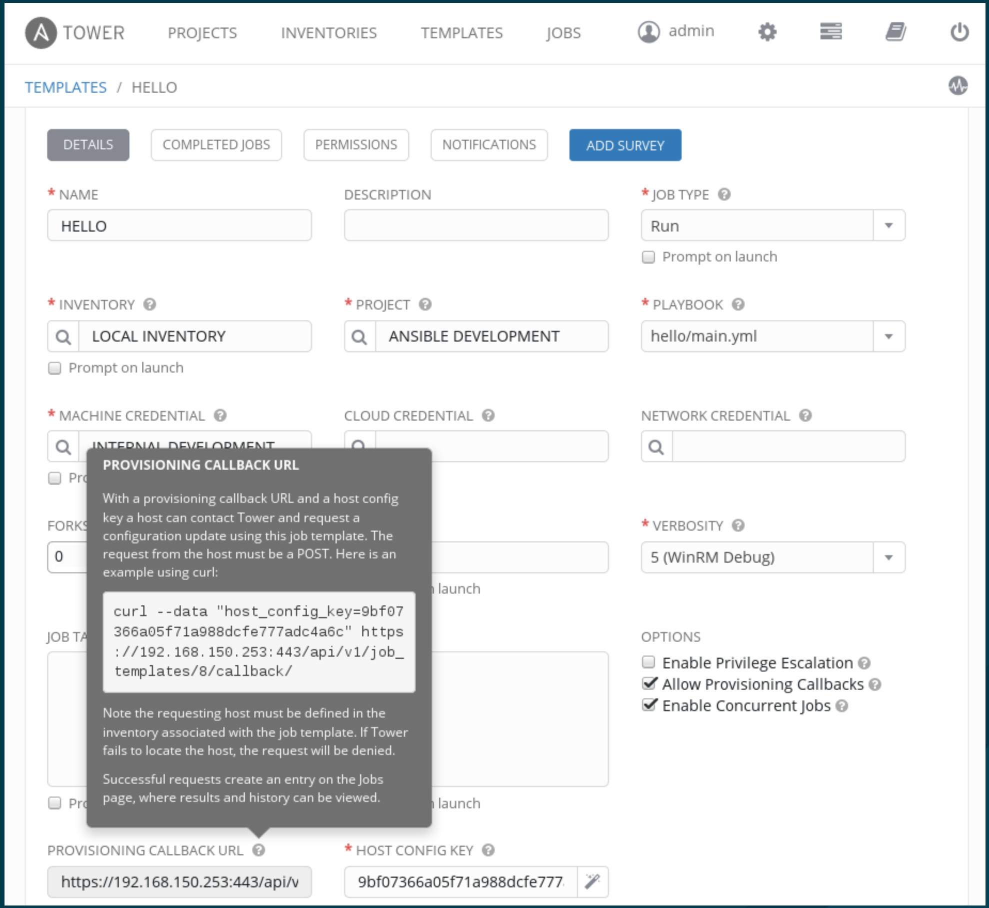
What’s the Ansible Tower



Execute Job/Workflow/Callback

Ansible Tower runs a playbook as a Job.

- Job Template
  - Jobs can be run periodically.
- Workflow
  - Jobs can combine as a workflow
- Callback URL
  - Jobs can launch from Target via callback url



* NAME: HELLO	DESCRIPTION:	* JOB TYPE: Run ☐ Prompt on launch
* INVENTORY: LOCAL INVENTORY ☐ Prompt on launch	* PROJECT: ANSIBLE DEVELOPMENT	* PLAYBOOK: hello/main.yml
* MACHINE CREDENTIAL: INTERNAL DEVELOPMENT	CLOUD CREDENTIAL:	NETWORK CREDENTIAL:
FORKS: 0		* VERBOSITY: 5 (WinRM Debug)
JOB TA		OPTIONS ☐ Enable Privilege Escalation ✓ Allow Provisioning Callbacks ✓ Enable Concurrent Jobs
PROVISIONING CALLBACK URL: https://192.168.150.253:443/api/v	* HOST CONFIG KEY: 9bf07366a05f71a988dcfe777	

PROVISIONING CALLBACK URL

With a provisioning callback URL and a host config key a host can contact Tower and request a configuration update using this job template. The request from the host must be a POST. Here is an example using curl:

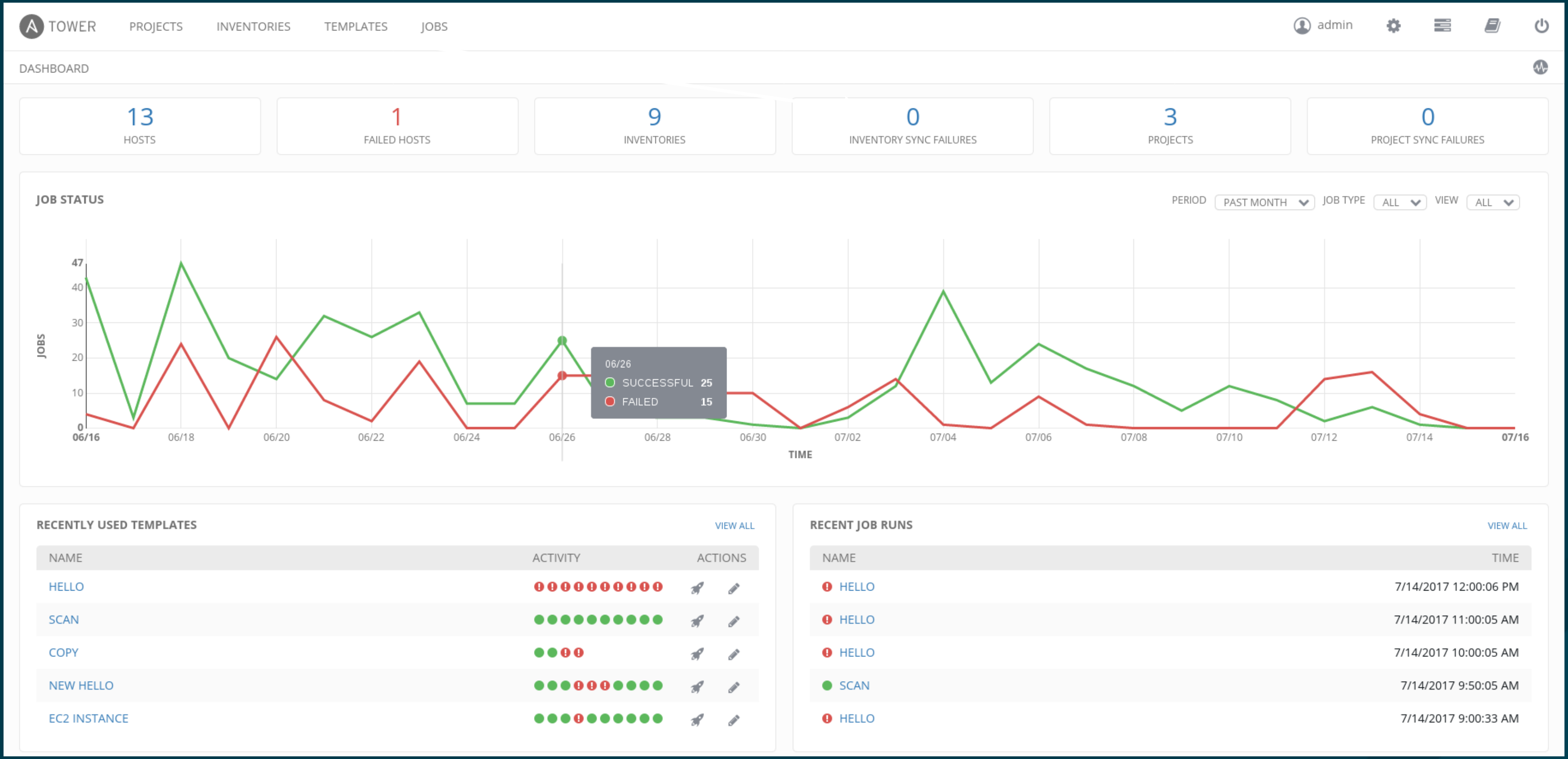
```
curl --data "host_config_key=9bf07
366a05f71a988dcfe777adc4a6c" https
://192.168.150.253:443/api/v1/job_
templates/8/callback/
```

Note the requesting host must be defined in the inventory associated with the job template. If Tower fails to locate the host, the request will be denied.

Successful requests create an entry on the Jobs page, where results and history can be viewed.

Web based Dashboard

Visualization of job execution result.



13	1	9	0	3	0
HOSTS	FAILED HOSTS	INVENTORIES	INVENTORY SYNC FAILURES	PROJECTS	PROJECT SYNC FAILURES

JOB STATUS

PERIOD: PAST MONTH JOB TYPE: ALL VIEW: ALL

RECENTLY USED TEMPLATES VIEW ALL

NAME	ACTIVITY	ACTIONS
HELLO	❗❗❗❗❗❗❗❗❗❗	
SCAN	●●●●●●●●●●	
COPY	●●❗❗	
NEW HELLO	●●●❗❗❗●●●●	
EC2 INSTANCE	●●●❗●●●●●●	

RECENT JOB RUNS VIEW ALL

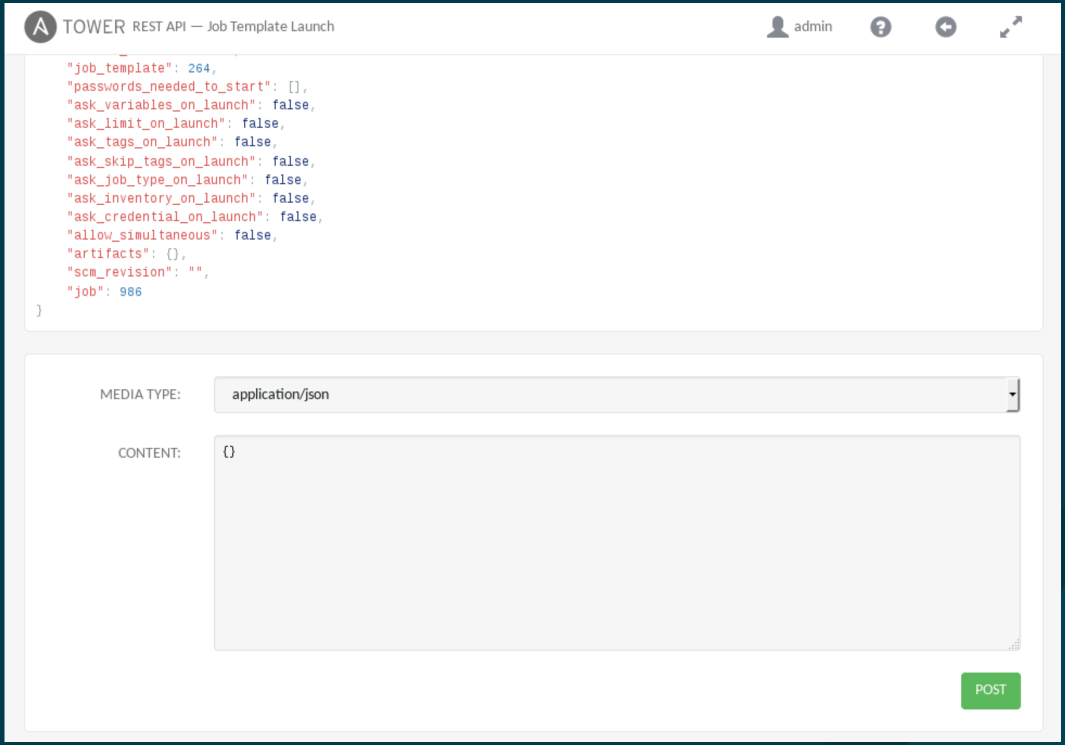
NAME	TIME
❗ HELLO	7/14/2017 12:00:06 PM
❗ HELLO	7/14/2017 11:00:05 AM
❗ HELLO	7/14/2017 10:00:05 AM
● SCAN	7/14/2017 9:50:05 AM
❗ HELLO	7/14/2017 9:00:33 AM

RESTful API

You can manage Tower server via RESTful API

If you want to manage Tower from other external IT system, you can use API!

- Access https://tower/api/v1/
- Manage Tower settings
- Launch Job template
- etc..



```
    "job_template": 264,
    "passwords_needed_to_start": [],
    "ask_variables_on_launch": false,
    "ask_limit_on_launch": false,
    "ask_tags_on_launch": false,
    "ask_skip_tags_on_launch": false,
    "ask_job_type_on_launch": false,
    "ask_inventory_on_launch": false,
    "ask_credential_on_launch": false,
    "allow_simultaneous": false,
    "artifacts": {},
    "scm_revision": "",
    "job": 986
}
```

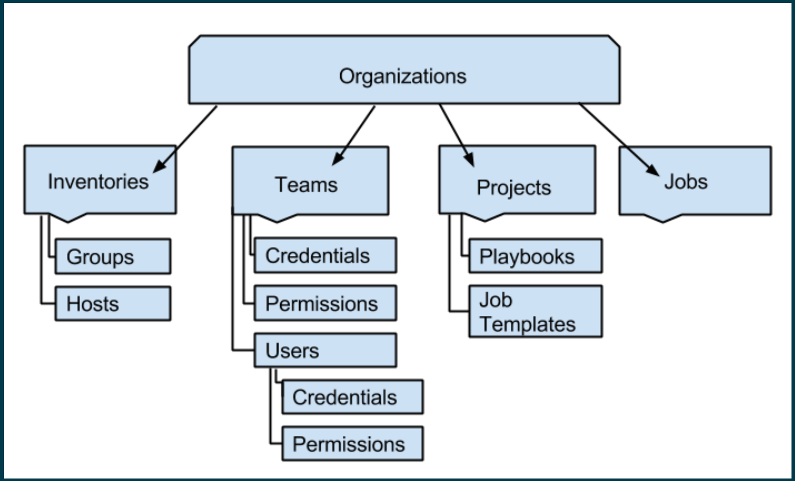
MEDIA TYPE: application/json

CONTENT: {}

Isolation, Consolidation and Cooperation

Isolation of authority, consolidation management, and using external systems.

- Role Based Access Control
  - Organization, Project, User, Team
- Integrates with LDAP, AD, and other IAM
- Logging aggregation with other system
- Job isolation via namespace and chroots
- etc…



If you want to proceed to the next step,

I believe Ansible Core and Tower will help you.

redhat

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos