Red Hat
Ansible
Automation

Ansible Best Practices, Part 4

Execute, verify, optimize and scale

EXECUTING THE ANSIBLE COMMANDS

How to execute

Ansible provides multiple switches for command line interaction and troubleshooting.

```
-vvvv
--step
--check
--diff
--start-at-task
--limit
```

Ansible has switches to show you what will be done

```
Use the power of included options:
    --list-tasks
    --list-tags
    --list-hosts
    --syntax-check
```

QUICKLY LAUNCH WITHOUT INVENTORY

If there is a need to launch something without an inventory - just do it!

- For single tasks - note the comma:

```
ansible all -i neon.qxyz.de, -m service -a
"name=redhat state=present"
```

- For playbooks - again, note the comma:

```
ansible-playbook -i neon.qxyz.de, site.yml
```

CHECK IMMEDIATELY WHAT WAS DONE

Don’t just start services -- use smoke tests

```
- name: check for proper response
  uri:
    url: http://localhost/myapp
    return_content: yes
  register: result
  until: '"Hello World" in result.content'
  retries: 10
  delay: 1
```

Try to avoid the command module - always seek out a module first

```
- name: add user
  command: useradd appuser

- name: install apache
  command: yum install httpd

- name: start apache
  shell: |
    service httpd start && chkconfig
httpd on
```

```
- name: add user
  user:
    name: appuser
    state: present

- name: install apache
  yum:
    name: httpd
    state: latest

- name: start apache
  service:
    name: httpd
    state: started
    enabled: yes
```

If managed files are not marked, they might be overwritten accidentally

- Label template output files as being generated by Ansible
- Use the ansible_managed** variable with the comment filter

```
{{ ansible_managed | comment }}
```

Root access is harder to track than sudo - use sudo wherever possible

- Don't run as root
- But login and security reasons often request non-root access
- Use become method - so Ansible scripts are executed via sudo (sudo is easy to track)
- Best: create an Ansible only user
- Don't try to limit sudo rights to certain commands - Ansible does not work that way!

DEBUG YOUR PROBLEM

Check logging on target machine

```
ansible-node sshd[2395]: pam_unix(sshd:session): session
  opened for user liquidat by (uid=0)
ansible-node ansible-yum[2399]: Invoked with name=['httpd']
  list=None install_repoquery=True conf_file=None
  disable_gpg_check=False state=absent disablerepo=None
  update_cache=False enablerepo=None exclude=None
```

How to keep the code executed on the target machine

Look into the logging of your target machine

```
$ ANSIBLE_KEEP_REMOTE_FILES=1 ansible target-node -m yum
  -a "name=httpd state=absent"
```

Execute with:

```
$ /bin/sh -c 'sudo -u $SUDO_USER /bin/sh -c
  "/usr/bin/python /home/liquidat/.ansible/tmp/..."
```

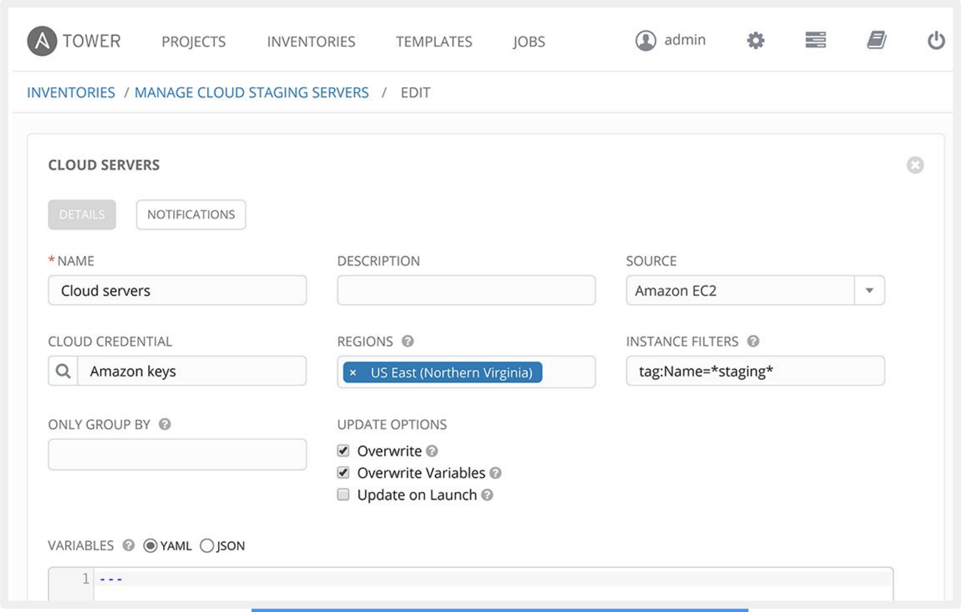
Debugging tasks can clutter the output, apply some housekeeping

```
- name: Output debug message
  debug:
    msg: "This always displays"

- name: Output debug message
  debug:
    msg: "This only displays with ansible-playbook -vv+"
    verbosity: 2
```

How to use in real life

Use dynamic & smart inventories



INVENTORIES / MANAGE CLOUD STAGING SERVERS / EDIT

CLOUD SERVERS

* NAME	DESCRIPTION	SOURCE
Cloud servers		Amazon EC2
CLOUD CREDENTIAL	REGIONS	INSTANCE FILTERS
Amazon keys	× US East (Northern Virginia)	tag:Name=*staging*
ONLY GROUP BY	UPDATE OPTIONS	
	☑ Overwrite ☑ Overwrite Variables □ Update on Launch	

VARIABLES ● YAML ○ JSON

```
---
```

- Combine multiple inventory types
- Use Fact Caching to keep system Details
- Let Tower take care of syncing and caching
- Use smart inventories to group nodes based on search filters

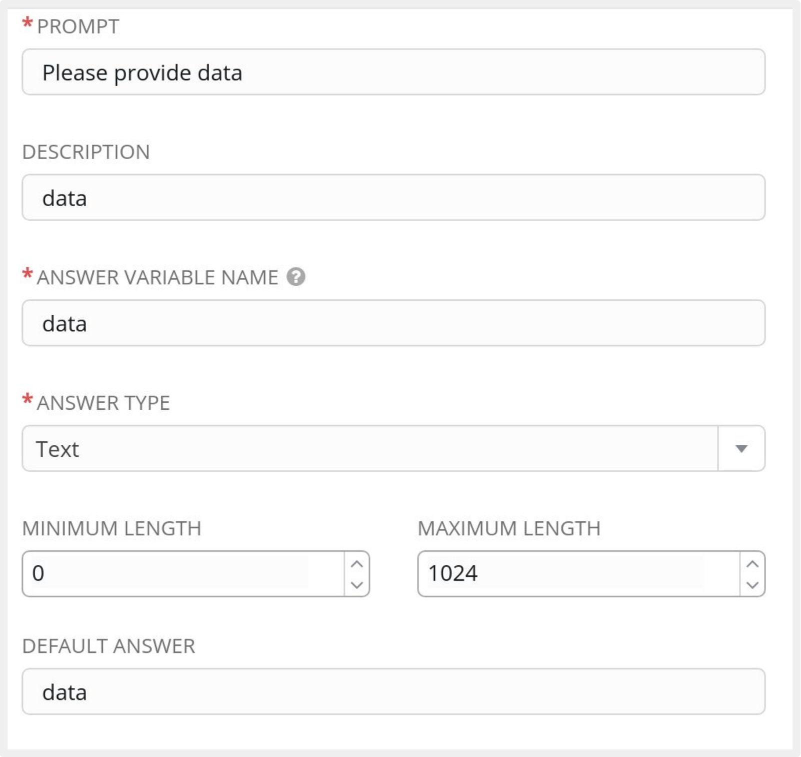
Tower job templates provide multiple options - use them wisely

- Keep jobs simple, focussed - as playbooks or roles
- Add labels to them to better filter
- For idempotent jobs, create “check” templates as well - and let them run over night
- Combine with notifications - and get feedback when a “check” failed

Multiple playbooks can be combined into one workflow

- Simple jobs, complex workflows
- React to problems via workflow
- Combine playbooks of different teams, different repositories
- Re-sync inventories during the play
- Re-sync inventories on workflow start

Use surveys to get variable values



* PROMPT

Please provide data

DESCRIPTION

data

* ANSWER VARIABLE NAME

data

* ANSWER TYPE

Text

MINIMUM LENGTH

0

MAXIMUM LENGTH

1024

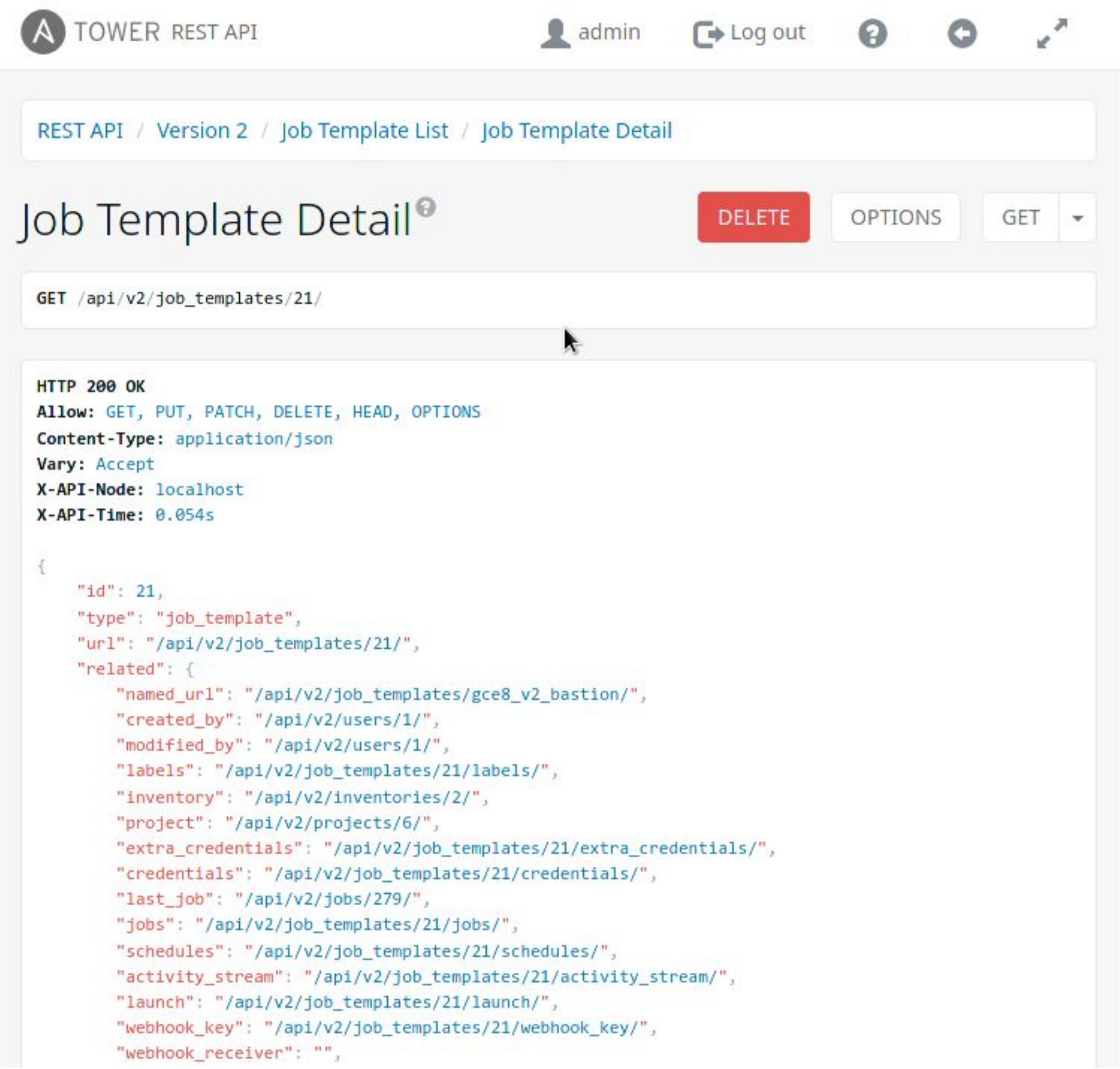
DEFAULT ANSWER

data

- Use good, meaningful variable names
- Provide a default choice
- Multiple choice > free text
- If answer not required - do you really need it at all?

Tower provides tenants, teams, and users - use them for separation

- Provide automation to others without exposing credentials
- Let others only see what they really need
- Use personal view instead of full Tower interface


TOWER REST API
admin
Log out
REST API / Version 2 / Job Template List / Job Template Detail
Job Template Detail
DELETE
OPTIONS
GET

```
GET /api/v2/job_templates/21/
```

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
X-API-Node: localhost
X-API-Time: 0.054s

{
    "id": 21,
    "type": "job_template",
    "url": "/api/v2/job_templates/21/",
    "related": {
        "named_url": "/api/v2/job_templates/gce8_v2_bastion/",
        "created_by": "/api/v2/users/1/",
        "modified_by": "/api/v2/users/1/",
        "labels": "/api/v2/job_templates/21/labels/",
        "inventory": "/api/v2/inventories/2/",
        "project": "/api/v2/projects/6/",
        "extra_credentials": "/api/v2/job_templates/21/extra_credentials/",
        "credentials": "/api/v2/job_templates/21/credentials/",
        "last_job": "/api/v2/jobs/279/",
        "jobs": "/api/v2/job_templates/21/jobs/",
        "schedules": "/api/v2/job_templates/21/schedules/",
        "activity_stream": "/api/v2/job_templates/21/activity_stream/",
        "launch": "/api/v2/job_templates/21/launch/",
        "webhook_key": "/api/v2/job_templates/21/webhook_key/",
        "webhook_receiver": "",
```

Trigger Automation from outside of Tower

- Tower API v2 can do everything available in UI
- Webhooks (Github or Gitlab)
- Provisioning Callback

Example Workflow:

*Provision VMs - *wait* -Provision App*

Instead:

Provision VMs with Callback URL for Provision App

Tower can send notifications if a job succeeds, fails or always - as mail, IRC, web hook, and so on

- Let Tower notify you and your team if something breaks
- Send mails/web-hooks automatically to a ticket systems and monitoring if there is a serious problem

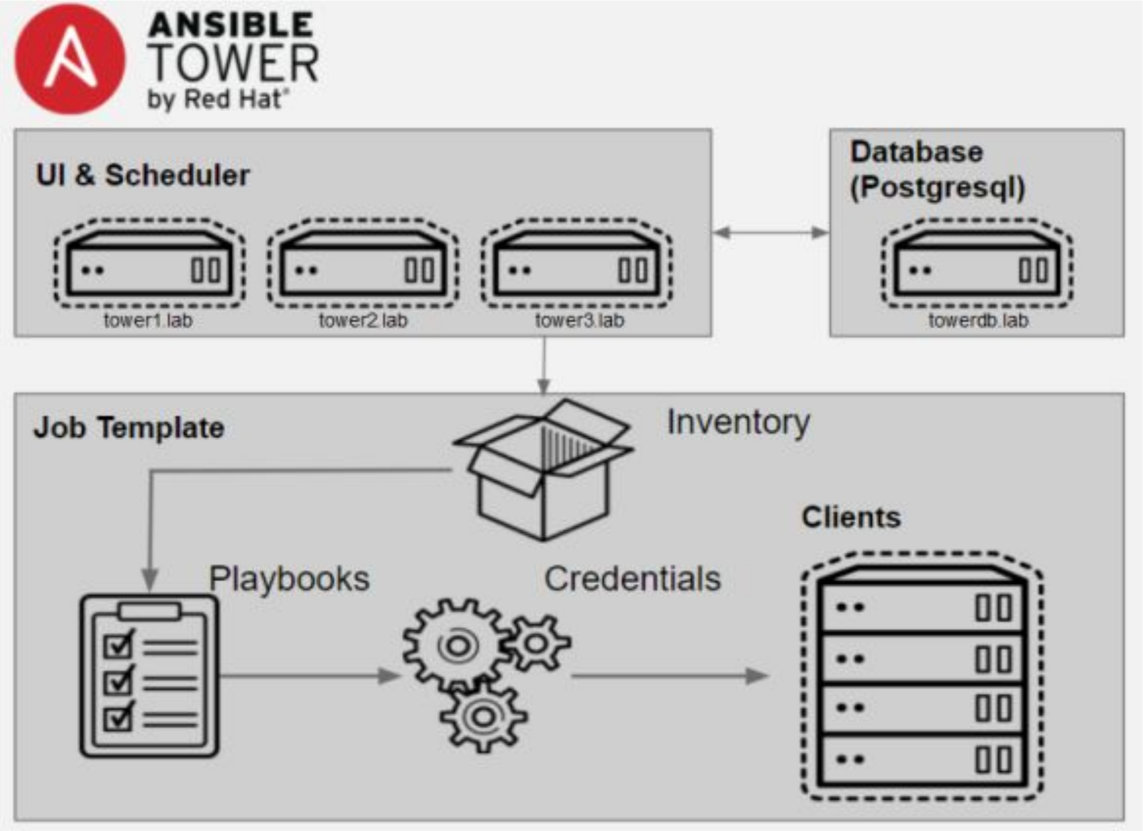
Don’t change Towers vEnv, create custom vEnvs if needed

- Playbooks on Tower run in vEnv on `/var/lib/awx/venv/ansible`
- vEnv contains supported plugIns and modules
- Creation of custom vEnvs possible
- Assignement of vEnv using Project, Organization or Inventory
- Do not overwrite system vEnv
- See also: containerzied Execution with custom PODs

Tower can be easily set up HA - and for restricted networks, deploy isolated nodes

- Make Tower HA - it is easy! (Well, except the DB part maybe....)
- For distant or restricted networks, use isolated nodes

Tower Clustering



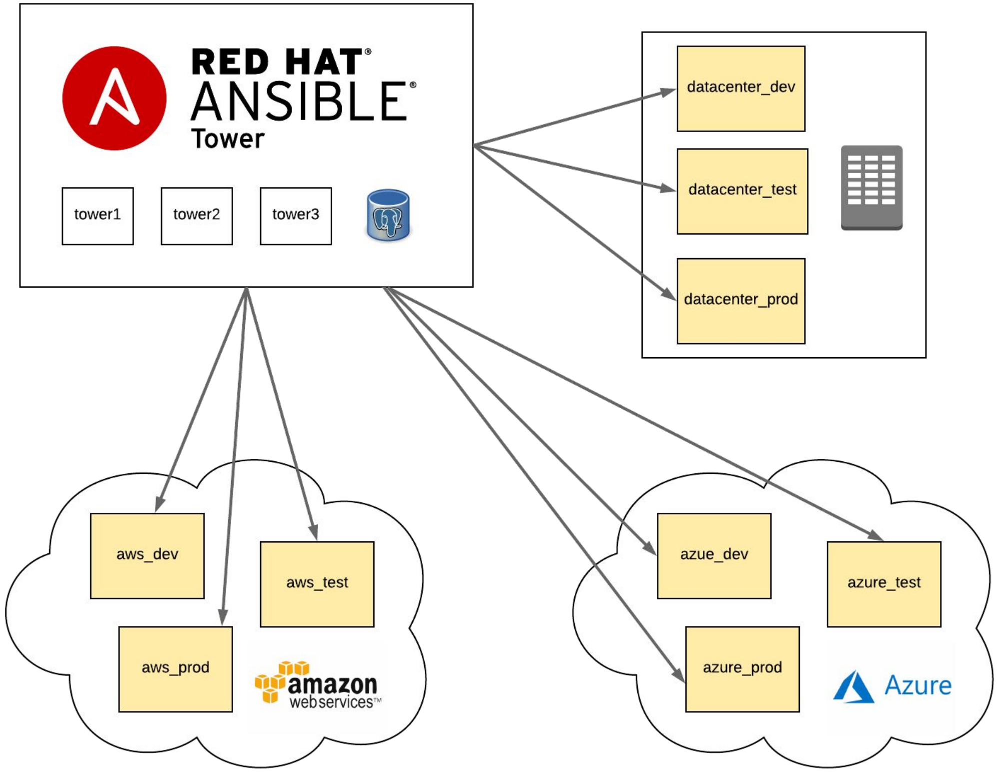
Automated Cluster Setup with Cluster Information in Tower Setup Inventory

```
[tower]
tower1.lab
tower2.lab
tower3.lab

[database]
Towerdb.lab
```

Setup does not deal with Postgresql HA, see Postgres Documentation for HA Options

Isolated Nodes



Inventory

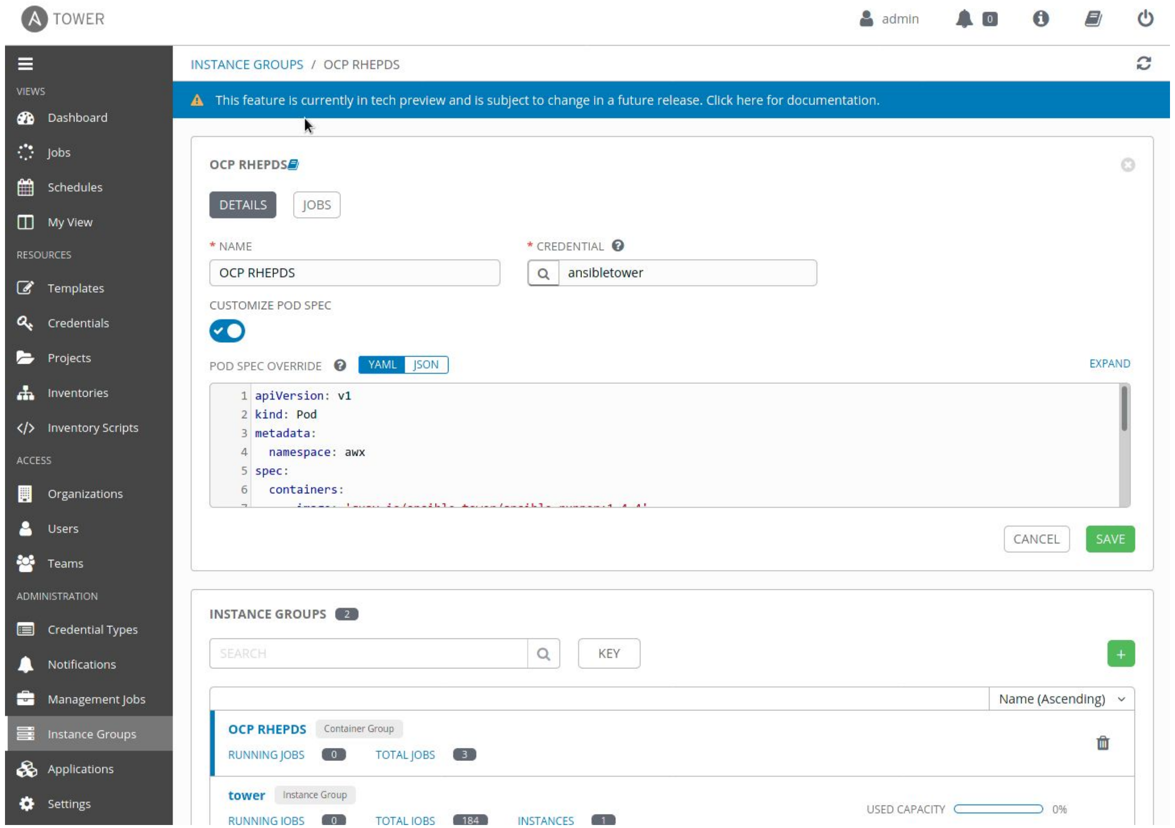
```
[tower]
tower1.nublar.mega.corp
tower2.nublar.mega.corp
tower3.nublar.mega.corp

[isolated_group_datacenter_dev]
dev-gw1.datacenter.mega.corp controller=tower
dev-gw2.datacenter.mega.corp controller=tower

[isolated_group_aws_dev]
dev-gw1.aws.mega.corp controller=tower
dev-gw2.aws.mega.corp controller=tower

[isolated_group_azure_dev]
dev-gw1.azure.mega.corp controller=tower
dev-gw2.azure.mega.corp controller=tower
```

Scale out with Openshift



INSTANCE GROUPS / OCP RHEPDS

⚠ This feature is currently in tech preview and is subject to change in a future release. Click here for documentation.

OCP RHEPDS

DETAILS | JOBS

* NAME: OCP RHEPDS

* CREDENTIAL: ansibletower

CUSTOMIZE POD SPEC: ●

POD SPEC OVERRIDE: YAML | JSON — EXPAND

```
apiVersion: v1
kind: Pod
metadata:
  namespace: awx
spec:
  containers:
    - image: 'quay.io/ansible-tower/ansible-runner:1.4.4'
```

CANCEL | SAVE

INSTANCE GROUPS 2

SEARCH | KEY | +

Name (Ascending)

OCP RHEPDS — Container Group — RUNNING JOBS 0 — TOTAL JOBS 3

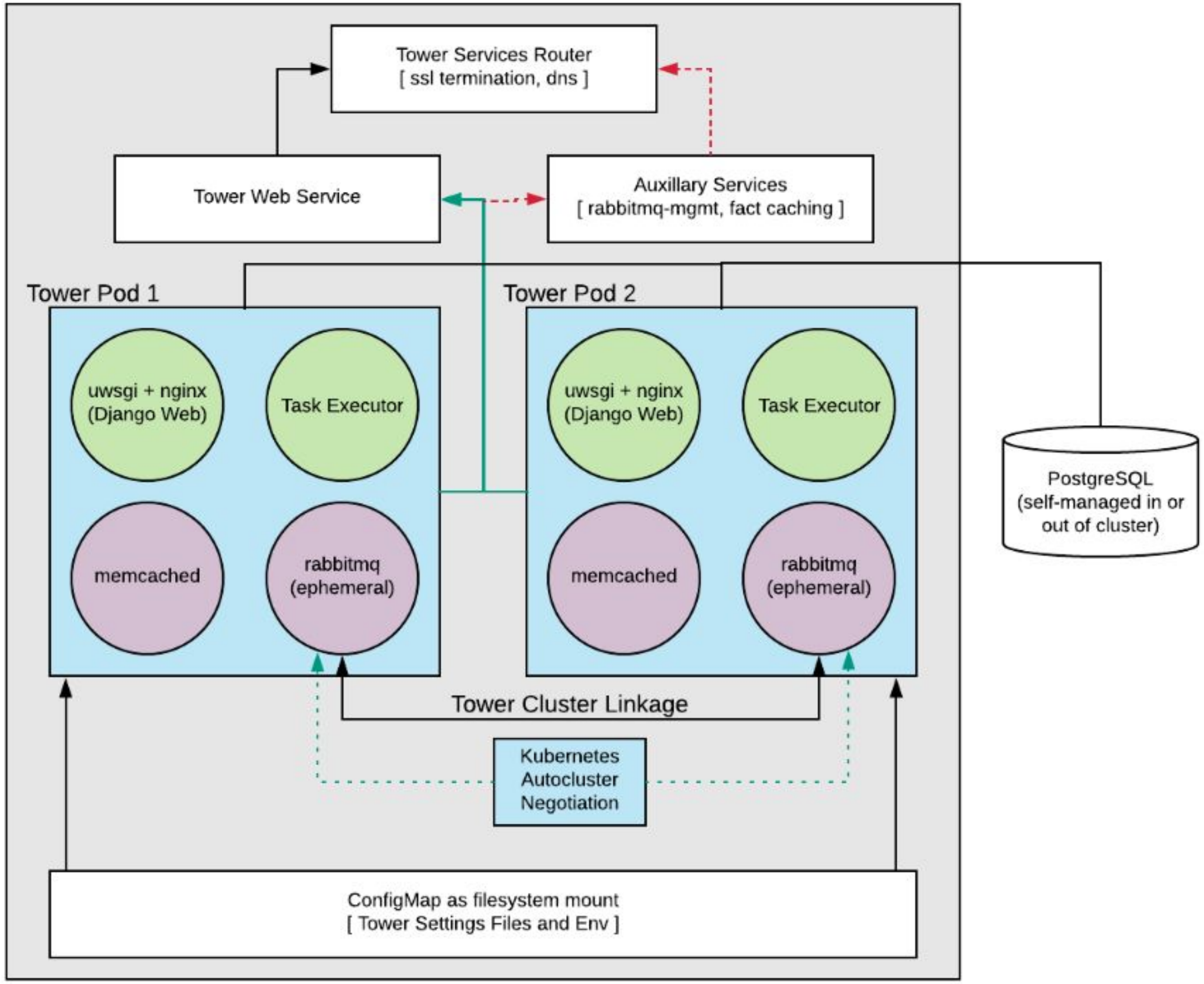
tower — Instance Group — RUNNING JOBS 0 — TOTAL JOBS 184 — INSTANCES 1 — USED CAPACITY 0%

Requirements

- Openshift Cluster
- Tower Service Account
- Assigned Project
- Pod Definition

```
apiVersion: v1
kind: Pod
metadata:
  namespace: awx
spec:
  containers:
    - image: 'quay.io/ansible-tower/ansible-runner:1.4.4'
      tty: true
      stdin: true
      imagePullPolicy: Always
      args:
        - sleep
        - infinity
```

Next Generation Tower runs containered



Tech Preview in Tower 3.6. and up

Further containerization (planned)
Separate PODs for

- UI
- Execute
- Memchached
- Redis (probably to replace rabbitmq)

Further Details to be presented on Ansiblefest 2020,
October 12-14 in San Diego

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat