Red Hat
Ansible
Automation

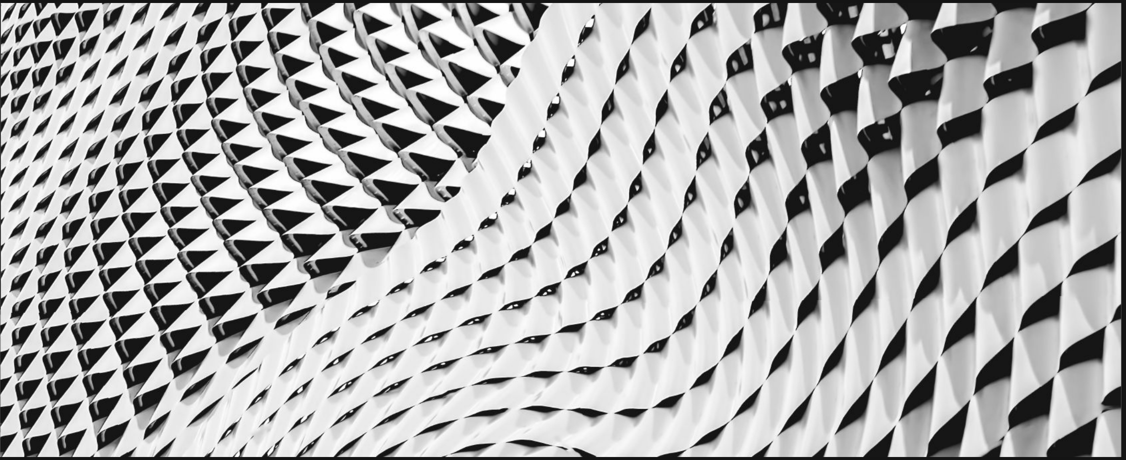
Ansible Best Practices, Part 1

How to evolve Ansible Automation

Andreas Stolzenberger
EMEA technical Partner Enablement Manager
ast@redhat.com

Red Hat

How to use



Ansible in Style

- Treat Playbooks as any other code
- Version Control
- Comments
- Variable Names
- Playbook Names
- Directory Structure
- Variable usage

Do it with style

- Create a style guide for consistency:
  - Tagging
  - Whitespace
  - Naming of Tasks, Plays, Variables, and Roles
  - Directory Layouts
- Enforce the style

  example: https://goo.gl/JfWBcW

  https://docs.adfinis-sygroup.ch/public/ansible-guide/styling_guide.html

Proper variable names can make plays more readable and avoid variable name conflicts

```
a: 25
data: ab
data2: abc
id: 123
```

```
apache_max_keepalive: 25
apache_port: 80
tomcat_port: 8080
```

Avoid collisions and confusion by adding the role name to a variable as a prefix.

```
apache_max_keepalive: 25
apache_port: 80
tomcat_port: 8080
```

MAKE YOUR PLAYBOOK READABLE

NO!

```
- name: install telegraf
  yum: name=telegraf-{{ telegraf_version }} state=present update_cache ...
  notify: restart telegraf

- name: start telegraf
  service: name=telegraf state=started
```

Yes!

```
- name: install telegraf
  yum:
    name: “telegraf-{{ telegraf_version }}”
    state: present
    update_cache: yes
    enablerepo: telegraf
  notify: restart telegraf

- name: start telegraf
  service:
    name: telegraf
    state: started
```

Exhibit A

```
- hosts: web
  tasks:
  - yum:
      name: httpd
      state: latest

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

```
PLAY [web]
********************************

TASK [setup]
********************************
ok: [web1]

TASK [yum]
********************************
ok: [web1]

TASK [service]
********************************
ok: [web1]
```

Exhibit B

```
- hosts: web
  name: installs and starts apache

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

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

```
PLAY [install and starts apache]
********************************

TASK [setup]
********************************
ok: [web1]

TASK [install apache packages]
********************************
ok: [web1]

TASK [starts apache service]
********************************
ok: [web1]
```

Evolve your Ansible code

Start with a basic playbook and static inventory

Refactor and modularize later

Move information to variables

Move credentials to Environment

Evolve from Core to Tower

Split complex Playbooks to Workflows

Evolution Example 1: Core Code

```
---
- hosts: localhost

  vars:
    service_account_email:
ansible@myproject.iam.gserviceaccount.com
    credentials_file: myproject-12345678.json
    project_id: myproject-12345678

  tasks:
    - name: create multiple instances
      gce:
        instance_names:
webserver1,webserver2,database,loadbalancer
        zone: us-central1-a
        machine_type: n1-standard-1
        image: rh74gce
        state: present
        service_account_email: "{{ service_account_email }}"
        credentials_file: "{{ credentials_file }}"
        project_id: "{{ project_id }}"

      register: gce
… (use gce.instance_data.name etc. to generate static
inventory )
```

Two Plabooks, same outcome

Credentials

Will move to Vault and the Environment

Module Paramters

Will move to Tempalte Variables

Register Output

To create static inventory moves to dynamic inventory

Evolution Example 1: Tower Code

```
---
- hosts: localhost
  gather_facts: no

  tasks:
    - name: handle instance
      gce:
        instance_names: "{{ gce_instance }}"
        zone: "{{ gce_zone }}"
        machine_type: "{{ gce_machine }}"
        image: "{{ gce_image }}"
        state: "{{ gce_state }}"
```

Two Plabooks, same outcome

Credentials

Delievered from Vault, not inside the Playbook

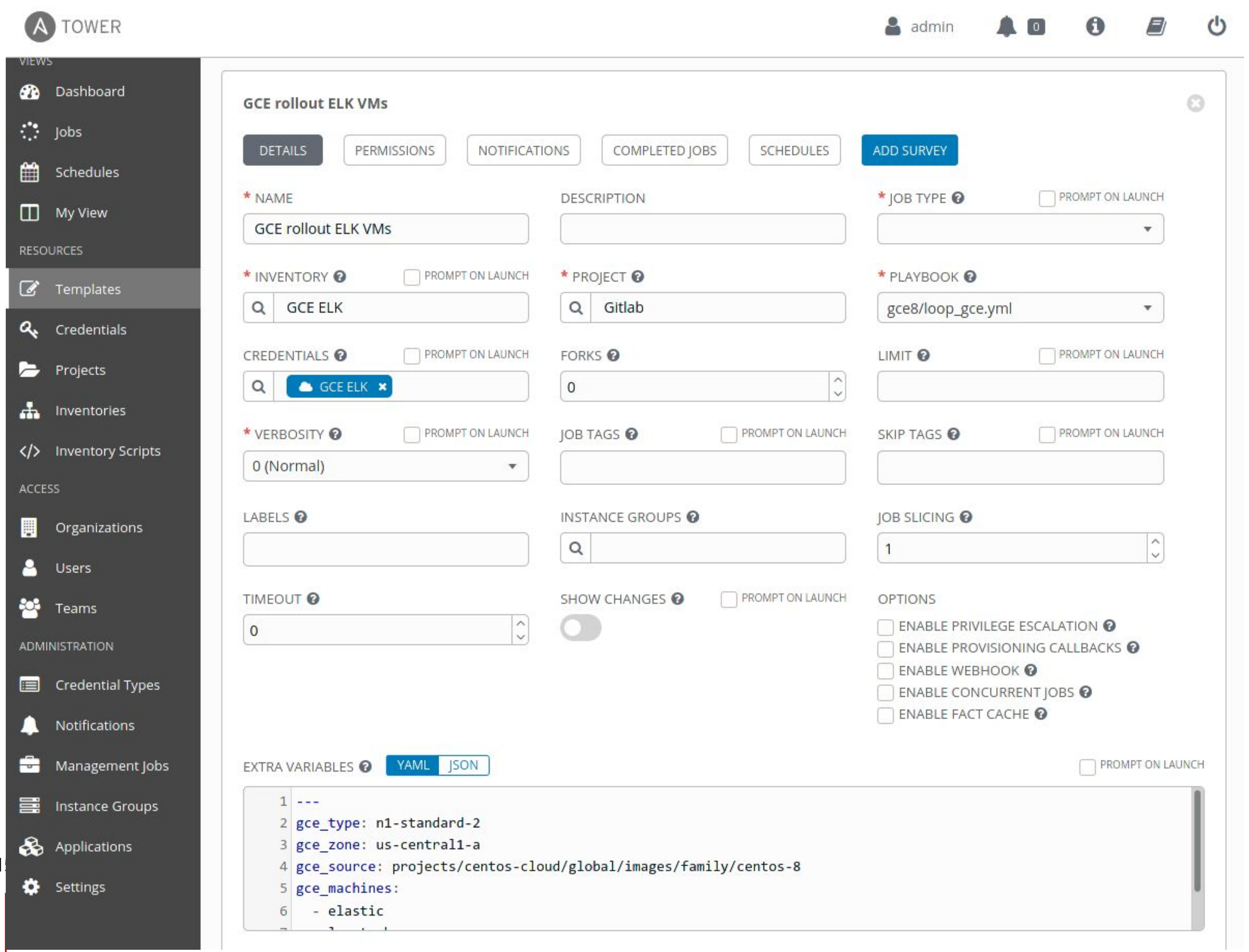
Module Paramters

Defined in Template Variable section or queried through a survey

Register Output

Not needed. Inventory is created dynamically.

Evolution Example 1: Tower Code



GCE rollout ELK VMs

* NAME: GCE rollout ELK VMs	DESCRIPTION:	* JOB TYPE ☐ PROMPT ON LAUNCH:
* INVENTORY ☐ PROMPT ON LAUNCH: GCE ELK	* PROJECT: Gitlab	* PLAYBOOK: gce8/loop_gce.yml
CREDENTIALS ☐ PROMPT ON LAUNCH: GCE ELK	FORKS: 0	LIMIT ☐ PROMPT ON LAUNCH:
* VERBOSITY ☐ PROMPT ON LAUNCH: 0 (Normal)	JOB TAGS ☐ PROMPT ON LAUNCH:	SKIP TAGS ☐ PROMPT ON LAUNCH:
LABELS:	INSTANCE GROUPS:	JOB SLICING: 1
TIMEOUT: 0	SHOW CHANGES ☐ PROMPT ON LAUNCH	OPTIONS

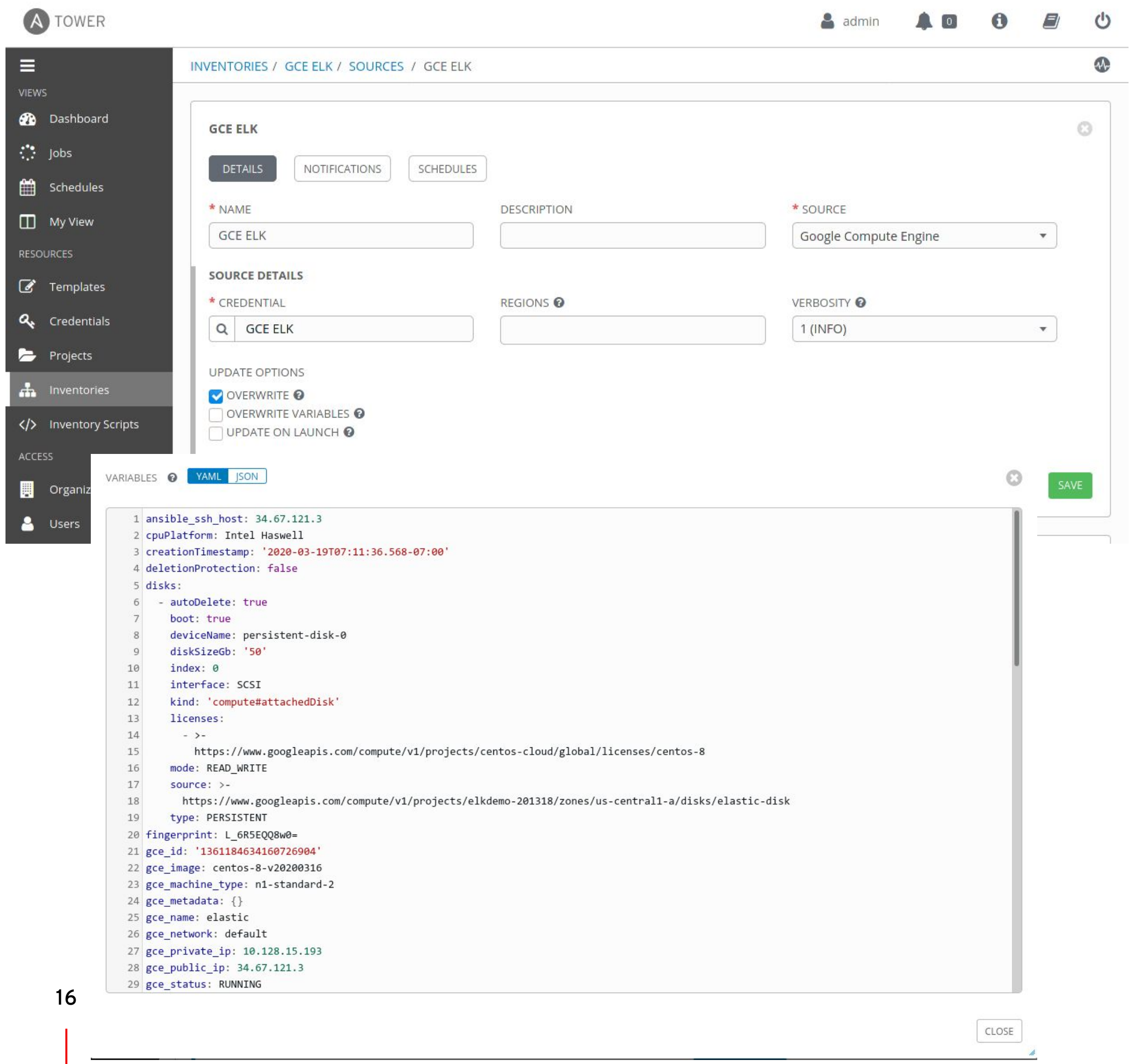
OPTIONS
- ☐ ENABLE PRIVILEGE ESCALATION
- ☐ ENABLE PROVISIONING CALLBACKS
- ☐ ENABLE WEBHOOK
- ☐ ENABLE CONCURRENT JOBS
- ☐ ENABLE FACT CACHE

EXTRA VARIABLES ☐ PROMPT ON LAUNCH

```yaml
---
gce_type: n1-standard-2
gce_zone: us-central1-a
gce_source: projects/centos-cloud/global/images/family/centos-8
gce_machines:
  - elastic
```

- Playbook controlled by variable Declaration
- Variables passed on by Template Definition in Tower
- Playbook reusable for many purposes

Use dynamic & smart inventories



GCE ELK

* NAME	DESCRIPTION	* SOURCE
GCE ELK		Google Compute Engine

SOURCE DETAILS

* CREDENTIAL	REGIONS	VERBOSITY
GCE ELK		1 (INFO)

UPDATE OPTIONS

- [✓] OVERWRITE
- [ ] OVERWRITE VARIABLES
- [ ] UPDATE ON LAUNCH

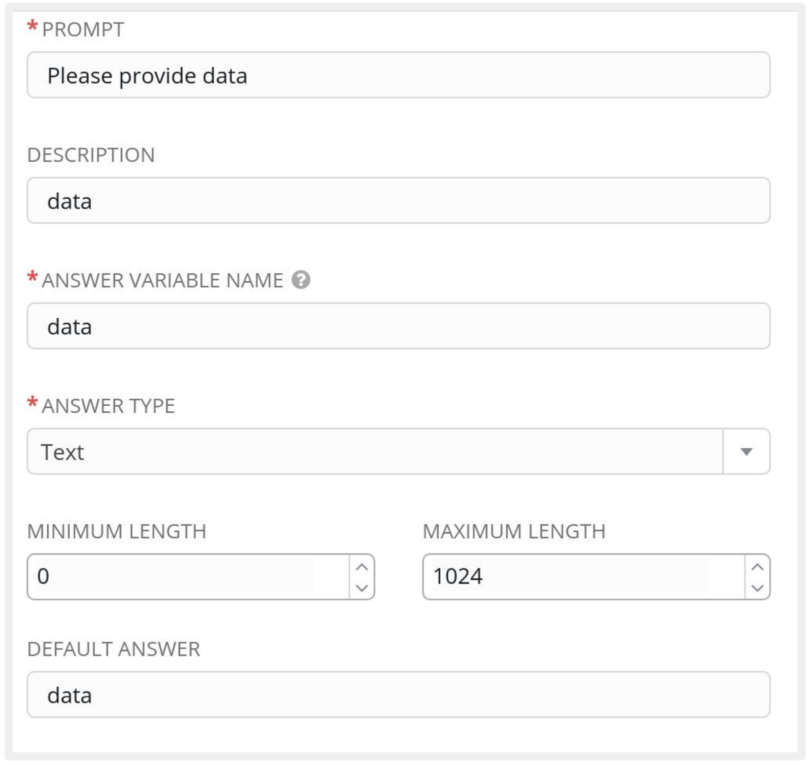
VARIABLES

```yaml
ansible_ssh_host: 34.67.121.3
cpuPlatform: Intel Haswell
creationTimestamp: '2020-03-19T07:11:36.568-07:00'
deletionProtection: false
disks:
  - autoDelete: true
    boot: true
    deviceName: persistent-disk-0
    diskSizeGb: '50'
    index: 0
    interface: SCSI
    kind: 'compute#attachedDisk'
    licenses:
      - >-
        https://www.googleapis.com/compute/v1/projects/centos-cloud/global/licenses/centos-8
    mode: READ_WRITE
    source: >-
      https://www.googleapis.com/compute/v1/projects/elkdemo-201318/zones/us-central1-a/disks/elastic-disk
    type: PERSISTENT
fingerprint: L_6R5EQQ8w0=
gce_id: '1361184634160726904'
gce_image: centos-8-v20200316
gce_machine_type: n1-standard-2
gce_metadata: {}
gce_name: elastic
gce_network: default
gce_private_ip: 10.128.15.193
gce_public_ip: 34.67.121.3
gce_status: RUNNING
```

- Combine multiple inventory types
- Let Tower take care of syncing and caching
- Use smart inventories to group nodes

Red Hat

Use surveys to get variable values



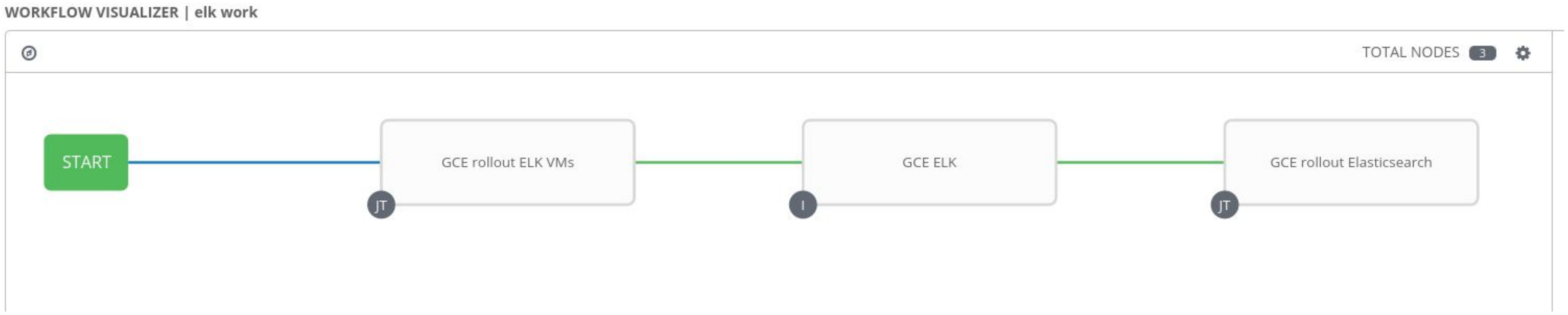
* PROMPT	Please provide data
DESCRIPTION	data
* ANSWER VARIABLE NAME	data
* ANSWER TYPE	Text
MINIMUM LENGTH	0
MAXIMUM LENGTH	1024
DEFAULT ANSWER	data

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

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

Multiple playbooks can be combined into one workflow



Step 1:
Create VMs on Cloud

Step 2:
Sync Cloud Inventory

Step 3:
Roll out Application

Use Workflows to abstract “Layers” of your Automation. Step 1 & 2 are Cloud speciffic, Step 3 can be Cloud agnostic and may be reused with different Clouds.

POWERFUL BLOCKS

Blocks can help in organizing code, but also enable rollbacks or output data for critical changes.

```
- block:
    copy:
      src: critical.conf
      dest: /etc/critical/crit.conf
    service:
      name: critical
      state: restarted
  rescue:
   command: shutdown -h now
```

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
```

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

Send all logs from Tower to central logging

- Splunk, Loggly, ELK, REST
- Send results from Ansible runs - but also from Tower changes

Give inventory nodes human-meaningful names rather than IPs or DNS hostnames.

```
10.1.2.75
10.1.5.45
10.1.4.5
10.1.0.40

w14301.acme.com
w17802.acme.com
w19203.acme.com
w19304.acme.com
```

```
db1 ansible_host=10.1.2.75
db2 ansible_host=10.1.5.45
db3 ansible_host=10.1.4.5
db4 ansible_host=10.1.0.40

web1 ansible_host=w14301.acme.com
web2 ansible_host=w17802.acme.com
web3 ansible_host=w19203.acme.com
web4 ansible_host=w19203.acme.com
```

Group hosts for easier inventory selection and less conditional tasks -- the more the better.

```
[db]
db[1:4]

[web]
web[1:4]
```

```
[east]
db1
web1
db3
web3

[west]
db2
web2
db4
web4
```

```
[dev]
db1
web1

[testing]
db3
web3

[prod]
db2
web2
db4
web4
```

Use dynamic sources where possible. Either as a single source of truth - or let Ansible unify multiple sources.

- Stay in sync automatically
- Reduce human error
- No lag when changes occur
- Let others manage the inventory

Red Hat

Know where your variables are

- Find the appropriate place for your variables based on what, where and when they are set or modified
- Separate logic (tasks) from variables and reduce repetitive patterns
- Do not use every possibility to store variables - settle to a defined scheme and as few places as possible (see Styleguide)

ROLES,COLLECTIONS AND GALAXIES

A Role is like a “function()” in other languages

```
site.yml
webservers.yml
fooservers.yml
roles/
    common/
        tasks/
        handlers/
        files/
        templates/
        vars/
        defaults/
        meta/
    webservers/
        tasks/
        defaults/
        meta/
```

Directory Structure

Separates Ansible Code into

- Tasks
- Variables
- Files
- Templates
- Handlers
- Additional Data

Create Role by funktional Tasks, like

- Role: DB2/Server
- Role: Webserver

Red Hat

Roles enable you to encapsulate your operations.

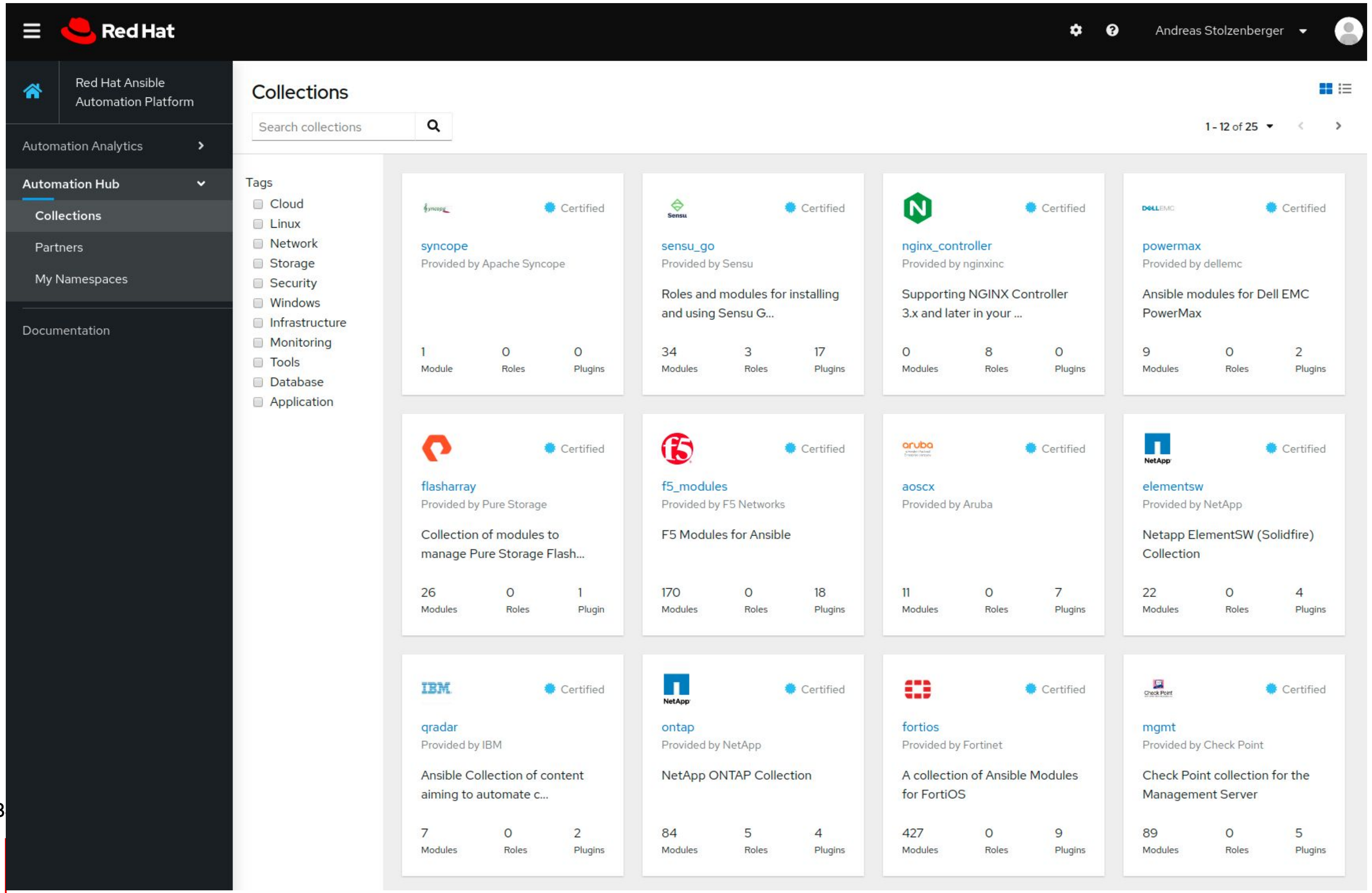
```
webservers.yml:

---
- hosts: webservers
  roles:
     - common
     - webservers
```

- Like playbooks -- keep roles purpose and function focused
- Store roles each in a dedicated Git repository
- Limit role dependencies
- Reusable in many Templates
- Simplyfies Main Code

Get collections from Red Hat Automation Hub, Inspect Roles from Galaxy

- Collections can contain roles, and other other code like modules as well
- Galaxy provides thousands of roles and collections
- Quality varies drastically
- Prefer Red Hat supported Collections over Upstream Galaxy
- Commit to Galaxy if you have Roles to share

Red Hat Automation Hub



Red Hat

Andreas Stolzenberger

Red Hat Ansible Automation Platform

Automation Analytics

Automation Hub

Collections

Partners

My Namespaces

Documentation

Collections

Search collections

1 - 12 of 25

Tags

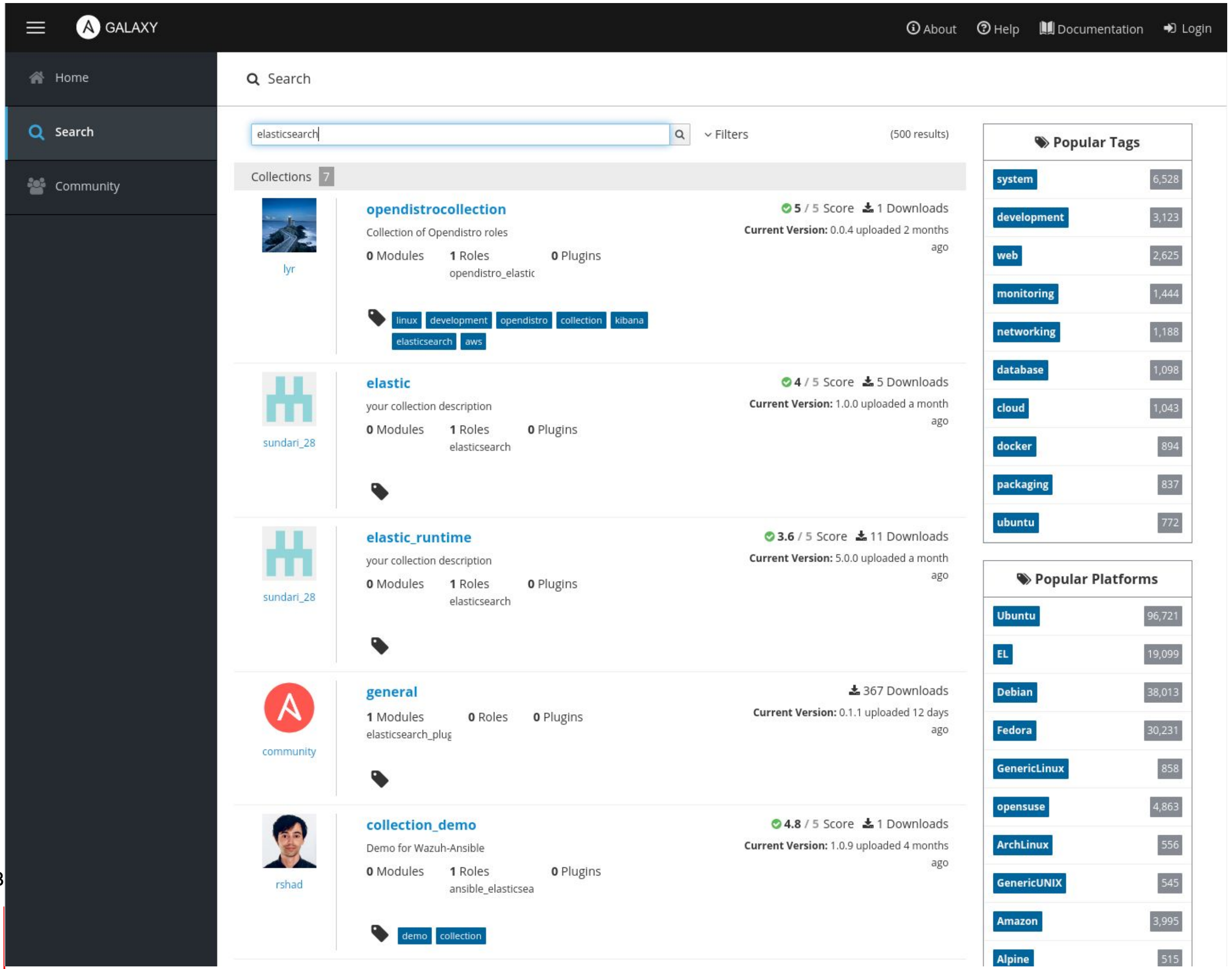
- Cloud
- Linux
- Network
- Storage
- Security
- Windows
- Infrastructure
- Monitoring
- Tools
- Database
- Application

Collection	Provider	Description	Modules	Roles	Plugins
syncope (Certified)	Provided by Apache Syncope		1 Module	0 Roles	0 Plugins
sensu_go (Certified)	Provided by Sensu	Roles and modules for installing and using Sensu G...	34 Modules	3 Roles	17 Plugins
nginx_controller (Certified)	Provided by nginxinc	Supporting NGINX Controller 3.x and later in your ...	0 Modules	8 Roles	0 Plugins
powermax (Certified)	Provided by dellemc	Ansible modules for Dell EMC PowerMax	9 Modules	0 Roles	2 Plugins
flasharray (Certified)	Provided by Pure Storage	Collection of modules to manage Pure Storage Flash...	26 Modules	0 Roles	1 Plugin
f5_modules (Certified)	Provided by F5 Networks	F5 Modules for Ansible	170 Modules	0 Roles	18 Plugins
aoscx (Certified)	Provided by Aruba		11 Modules	0 Roles	7 Plugins
elementsw (Certified)	Provided by NetApp	Netapp ElementSW (Solidfire) Collection	22 Modules	0 Roles	4 Plugins
qradar (Certified)	Provided by IBM	Ansible Collection of content aiming to automate c...	7 Modules	0 Roles	2 Plugins
ontap (Certified)	Provided by NetApp	NetApp ONTAP Collection	84 Modules	5 Roles	4 Plugins
fortios (Certified)	Provided by Fortinet	A collection of Ansible Modules for FortiOS	427 Modules	0 Roles	9 Plugins
mgmt (Certified)	Provided by Check Point	Check Point collection for the Management Server	89 Modules	0 Roles	5 Plugins

Red Hat and Vendor supported Collections consisting of:

- Plug-Ins
- Modules
- Roles

Ansible Galaxy



GALAXY

About Help Documentation Login

Home

Search

Community

Search

elasticsearch

Filters

(500 results)

Collections 7

opendistrocollection — lyr
Collection of Opendistro roles
0 Modules 1 Roles opendistro_elastic 0 Plugins
5 / 5 Score 1 Downloads
Current Version: 0.0.4 uploaded 2 months ago
linux development opendistro collection kibana elasticsearch aws

elastic — sundari_28
your collection description
0 Modules 1 Roles elasticsearch 0 Plugins
4 / 5 Score 5 Downloads
Current Version: 1.0.0 uploaded a month ago

elastic_runtime — sundari_28
your collection description
0 Modules 1 Roles elasticsearch 0 Plugins
3.6 / 5 Score 11 Downloads
Current Version: 5.0.0 uploaded a month ago

general — community
1 Modules elasticsearch_plug 0 Roles 0 Plugins
367 Downloads
Current Version: 0.1.1 uploaded 12 days ago

collection_demo — rshad
Demo for Wazuh-Ansible
0 Modules 1 Roles ansible_elasticsea 0 Plugins
4.8 / 5 Score 1 Downloads
Current Version: 1.0.9 uploaded 4 months ago
demo collection

Popular Tags	
system	6,528
development	3,123
web	2,625
monitoring	1,444
networking	1,188
database	1,098
cloud	1,043
docker	894
packaging	837
ubuntu	772

Popular Platforms	
Ubuntu	96,721
EL	19,099
Debian	38,013
Fedora	30,231
GenericLinux	858
opensuse	4,863
ArchLinux	556
GenericUNIX	545
Amazon	3,995
Alpine	515

Community developed and maintained Roles:

- Inspirational
- Open Source
- Unsupported
- Might already have solved your problem

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