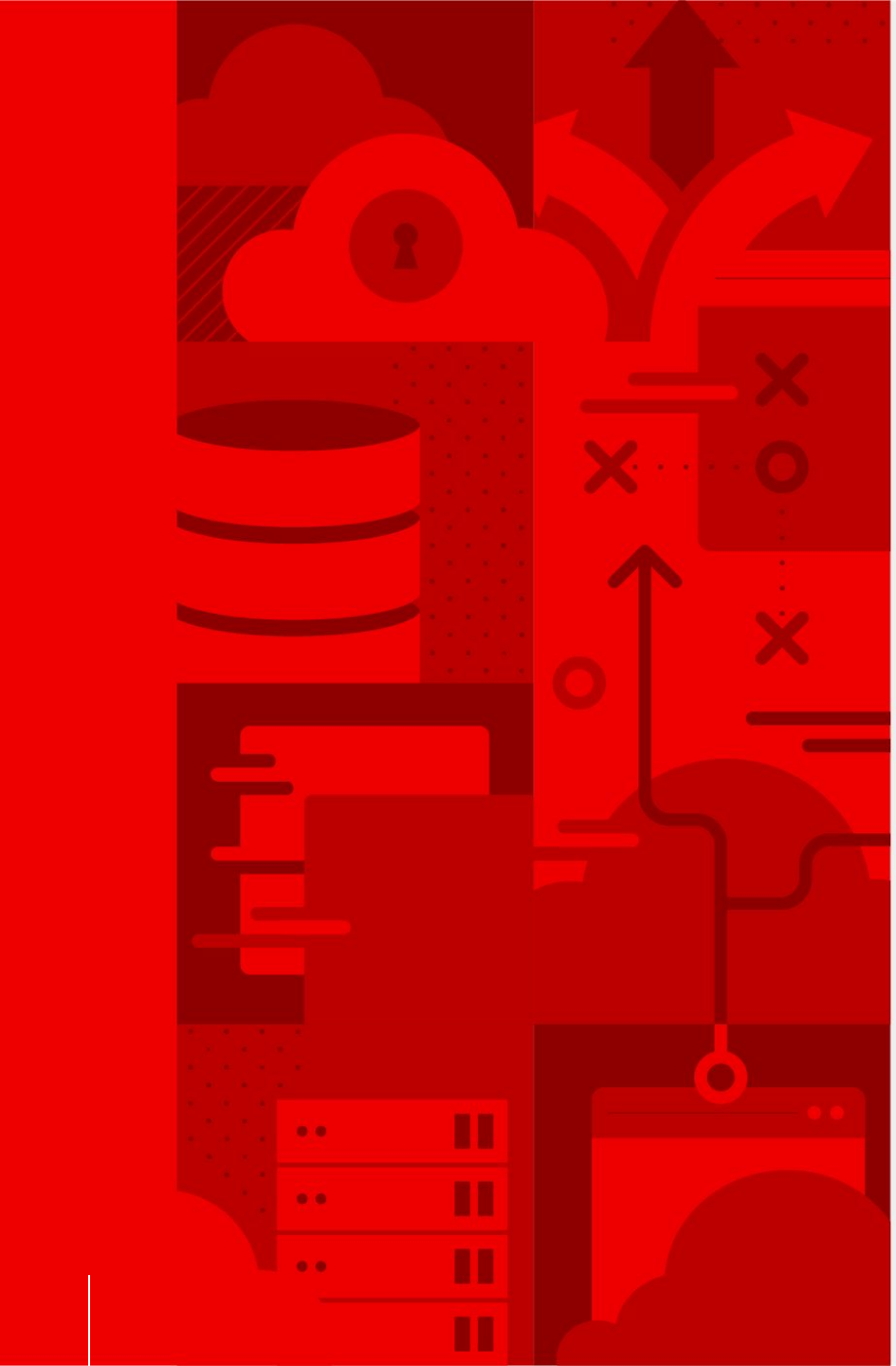


Ansible Automation: A Story from the Trenches

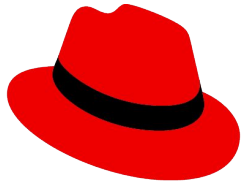
David Glaser

Senior Technical Account Manager

11/20/2019

Red Hat

Me!



Red Hat
Red Hat Certified

Engineer

David Glaser

Senior Technical Account Manager

Red Hat, Inc.
227 W. Monroe Street
Chicago, IL 60606

dglaser@redhat.com
grizz@redhat.com
Tel: 3124774368
Mobile: 81028441854
Certification number: 100-135-995

Red Hat

Ansible is a tool, automation is a skill

- Large customer with thousands of hosts
- Migrating from Chef to Ansible
- Customer team produced a set of Ansible playbooks and roles, then requested we review them
- We found a number of improvements, both obvious and subtle

Facts or no Facts?

Red Hat

Ansible Fact Gathering

Red Hat

Problem?

```
- name: RHEL Base configurations
  hosts: all
  gather_facts: no
  tasks:
  - name: checking host platform
    setup:
      filter: ansible_distribution
    register: host_distribution

  - name: checking host version
    setup:
      filter: ansible_distribution_major_version
    register: host_major_version
```

Red Hat

When to gather facts

- Ansible gathers facts by default
- Don’t gather facts when you need speed and won’t be using any of the facts on the host
- Filtering of facts happens on return, so there’s no speed up

Red Hat

Shell and Command modules

Red Hat

More shells than MarioKart!

- About 70% of the tasks were shell tasks
  - Most instances could be replaced with Ansible modules

- Shell and command modules use should be minimized
  - Not idempotent
  - Shell uses the environment of the user on the host which can be dangerous
  - Unless modified, always show up as a ‘changed’ return value
  - Harder to diagnose issues

- Use Ansible modules whenever possible
  - Make sure they are from a trusted source

Module Priority

1. Red Hat supported modules built into Ansible Engine
2. Vendor modules
3. Community modules (Ansible Galaxy, Github, etc)
4. command and shell modules

Red Hat

Know what you can Handle(er)

Red Hat

Problem?

```
- name: Create user
  user:
    name: idm
    home: "/opt/IDMfile"
    shell: /bin/ksh
    state: present
  notify:
      - restart nscd
      - Create IDMfile directory

- name: create kshrc file
  copy:
    src: dot-kshrc
    dest: /opt/IDMfile/.kshrc
    owner: idm
    group: root
    mode: '0600'
```

```
# handlers
  -    name: restart nscd
       service:
          name: nscd
          state: restarted

  -    name: Create IDMfile directory
       file:
          path: /opt/IDMfile
          owner: idm
          group: root
          mode: '0755'
          state: directory
```

Red Hat

Use Handlers wisely

- Handlers get run at the end of a play, not where they are called
- Handlers are only run on a change. If the play calling them does not result in a changed state, the handler is ignored
- Should not put anything in a play after a handler that is dependant on that handler unless you use a **meta: flush_handlers** routine

Red Hat

Keep it simple

Red Hat

Make your playbooks easy to read

Ansible should be written consistently

- Tasks can be written as a single line per parameter or multiple lines per parameter using an '=' sign. Decide on one and stick to it.

  Use

  ```
  module:
    name: value1
    state: value2
  ```

  Or

  ```
  module: name=value1 state=value2
  ```

Red Hat

Make your playbooks easy to read

Ansible should be easy to understand

- Simplify the logic used in when statements
  - Use ‘not’ only when needed
    - when: not variable == “no”
    - when: variable != “no”

Red Hat

Make your playbooks easy to read

Ansible should be easy to understand

- Simplify the logic used in when statements
  - Use ‘not’ only when needed
    - when: not variable == “no”
    - when: variable != “no”

  - Separate when statements that contain logical and(s) on separate lines
    - when: variable1 == “17” and variable2 == “Stand” and variable3 is defined
    - when:
      - variable1 == “17”
      - variable2 == “Stand”
      - Variable3 is defined

Red Hat

Just Do it

(Apologies to Nike)

Red Hat

Problem?

```
- set_fact:
    file_attr_immutable: "immutable"
- stat:
    path: "/etc/file.cfg"
  register: file_status
- name: check the file immutable and set it off
  file:
    path: "/etc/file.cfg"
    state: file
    attributes: -i
  when:  file_status.stat.exists and file_attr_immutable in file_status.stat.attributes
```

Red Hat

Automate to the end state

- Don’t put in extra tasks if you aren’t using them.
- Don’t check for a state if you are going to (possibly) reset that state
- Use tags or debug levels to execute certain tasks under certain conditions

Red Hat

Fail early and Fail often

Red Hat

Problem?

```
- name: Install/Upgrade PatchClient
  package:
    name: PatchClient
    state: latest

- name: Configure the PatchCLient
  shell: "bash /opt/PatchClient/config.sh client.dat > /dev/null"
  when: client.dat is defined
```

Red Hat

Try not to partially run tasks

- Check for error conditions (undefined variables, missing files, etc) early in a play
- Block groups of statements together so they complete or fail as a group

Red Hat

Try not to partially run tasks

- Check for error conditions (undefined variables, missing files, etc) early in a play
- Block groups of statements together so they complete or fail as a group

```
- name: Set up PatchClient
  block:
      - name: Install/Upgrade PatchClient
        package:
         name: PatchClient
         state: latest

      - name: Configure the PatchCLient
        shell: "bash /opt/PatchClient/config.sh client.dat > /dev/null"
   when: client.dat is defined
```

Red Hat

Summary

Red Hat

Summary

Automation is a process

Many different ways and levels of automation with Ansible

Use best practices and recommendations

https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html

Don’t be afraid to ask others to review

Come, come, Mr. Scott. Young minds, fresh ideas. Be tolerant. - James T. Kirk

Look at your playbooks as a whole

Don’t lose the forest for the trees

Questions

Red Hat