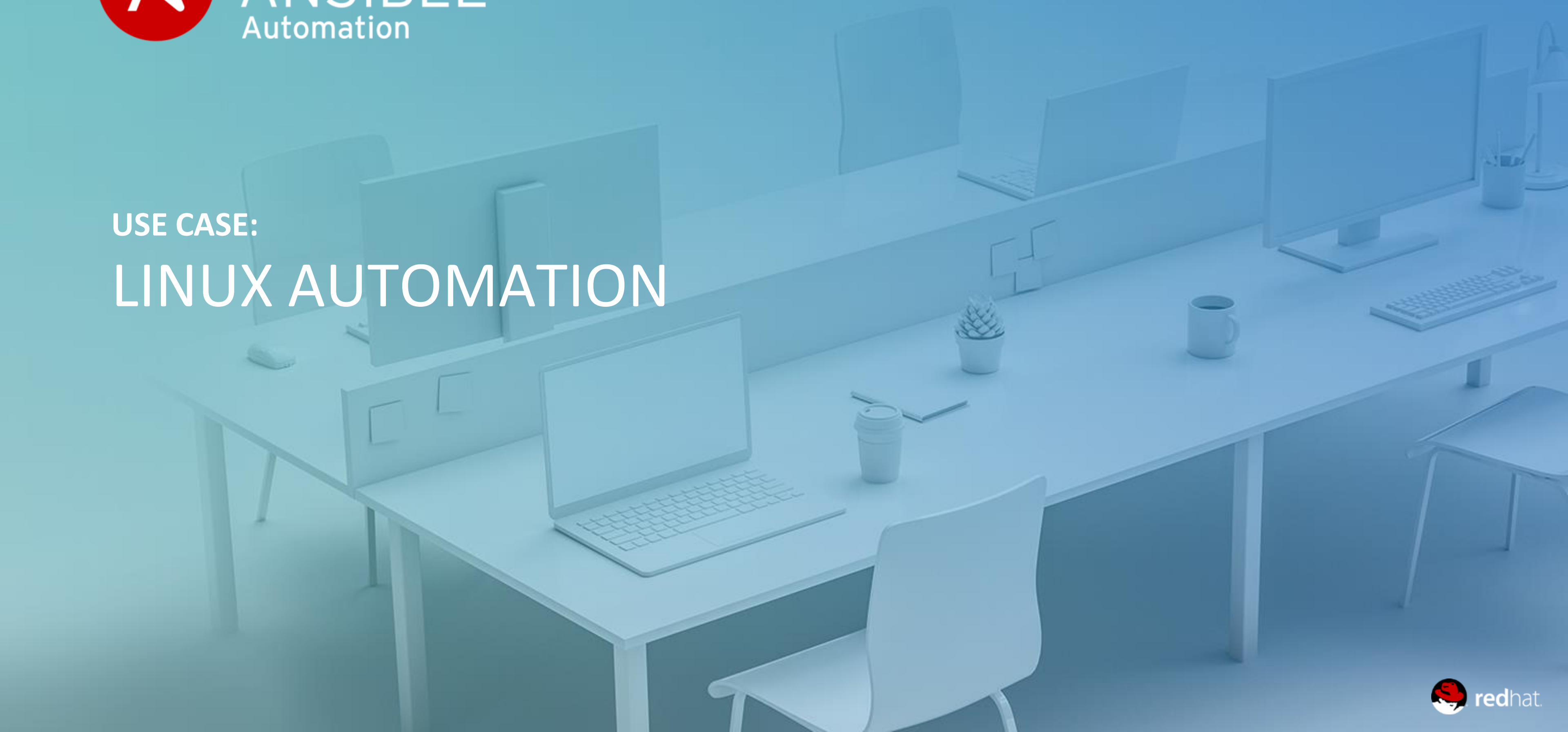
RED HAT ANSIBLE Automation

USE CASE:

LINUX AUTOMATION



LINUX AUTOMATION

150+

Linux Modules

AUTOMATE EVERYTHING LINUX

Red Hat Enterprise Linux, BSD, Debian, Ubuntu and many more!

ONLY REQUIREMENTS:
Python 2 (2.6 or later)
***or* Python 3 (3.5 or later)**

ansible.com/get-started

AUTOMATION FOR EVERYONE: **SYSTEM ADMINISTRATORS**

```yaml
---
- name: upgrade rhel packages
  hosts: rhel

  tasks:
    - name: upgrade all packages
      yum:
           name: '*'
           state: latest
```

AUTOMATION FOR EVERYONE: **SYSTEM ADMINISTRATORS**

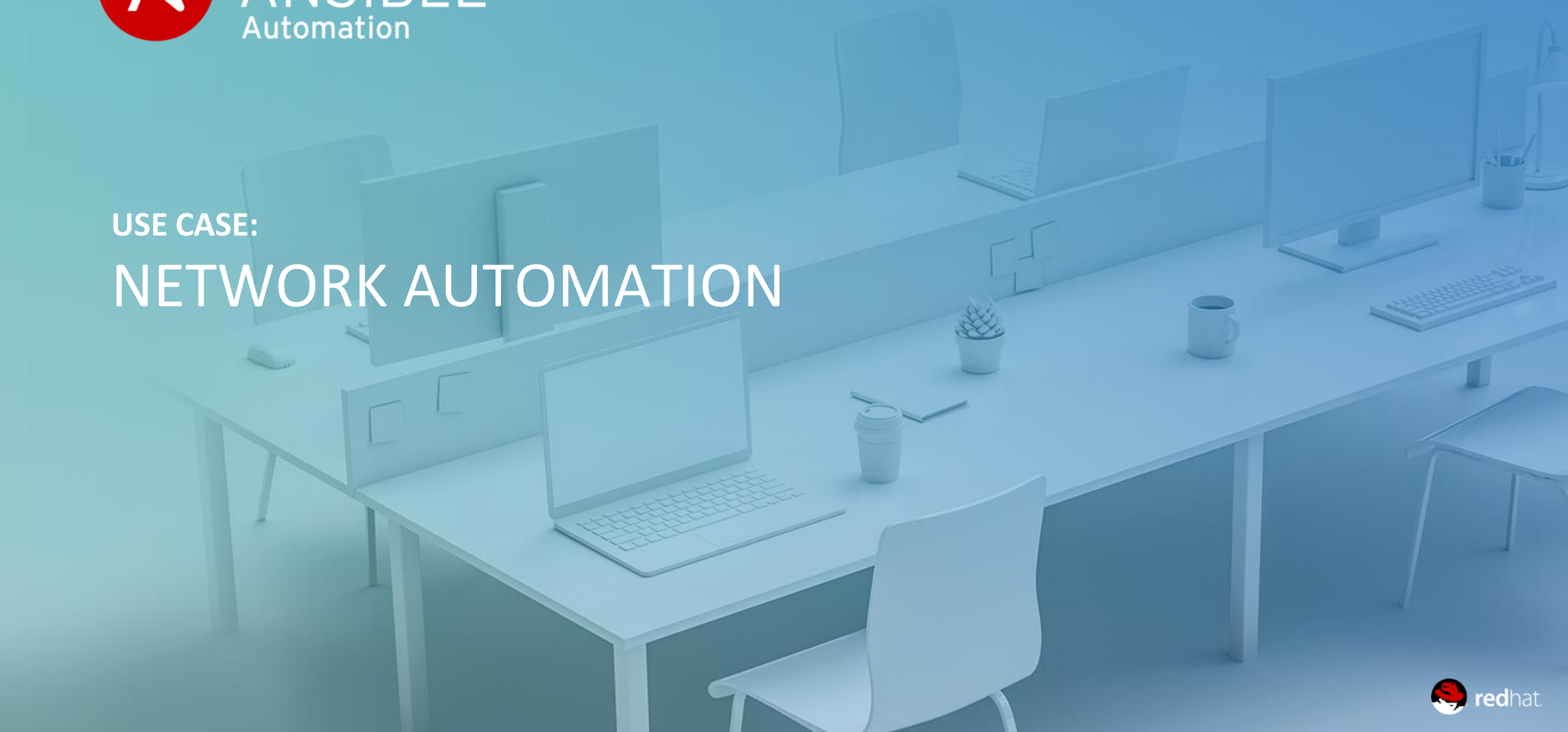
```yaml
---
- name: reboot rhel hosts
  hosts: rhel

  tasks:
    - name: reboot the machine
      reboot:
```

redhat.

AUTOMATION FOR EVERYONE: **SYSTEM ADMINISTRATORS**

```yaml
---
- name: check services on rhel hosts
  hosts: rhel
  become: yes

  tasks:
   - name: ensure nginx is started
     service:
       name: nginx
       state: started
```

redhat.



USE CASE:

NETWORK AUTOMATION

ANSIBLE NETWORK AUTOMATION

50	**700+**	**12***
Network Platforms	Network Modules	Galaxy Network Roles

ansible.com/for/networks
galaxy.ansible.com/ansible-network

**Roles developed and maintained by Ansible Network Engineering*

WHY AUTOMATE YOUR NETWORK?

PLAN AND PROTOTYPE VIRTUALLY

Use tasks as reusable building blocks

USE YOUR CURRENT DEVELOPMENT PRACTICES

Agile, DevOps, Waterfall

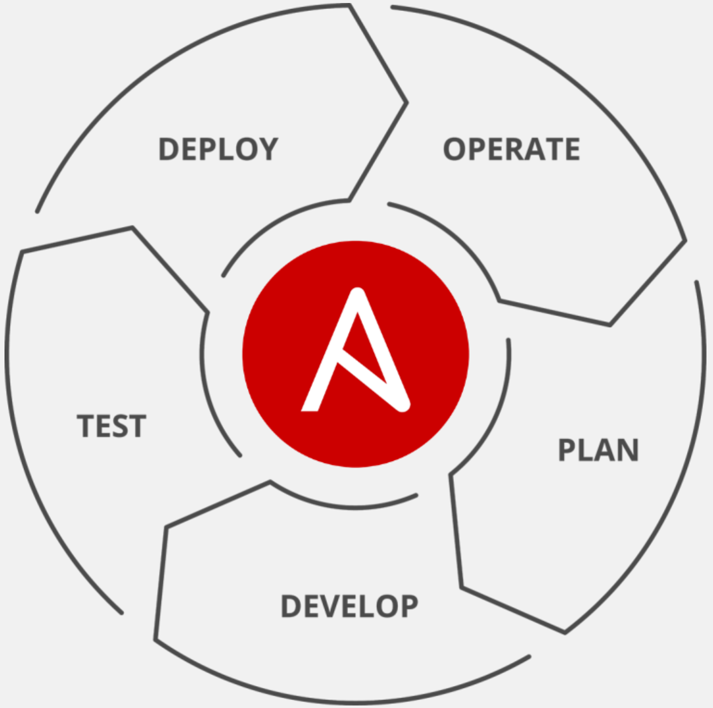
GO BEYOND THE "PING" TEST

Integrate with formal testing platforms

BE CONFIDENT DURING DEPLOYMENT

Validate changes were successful

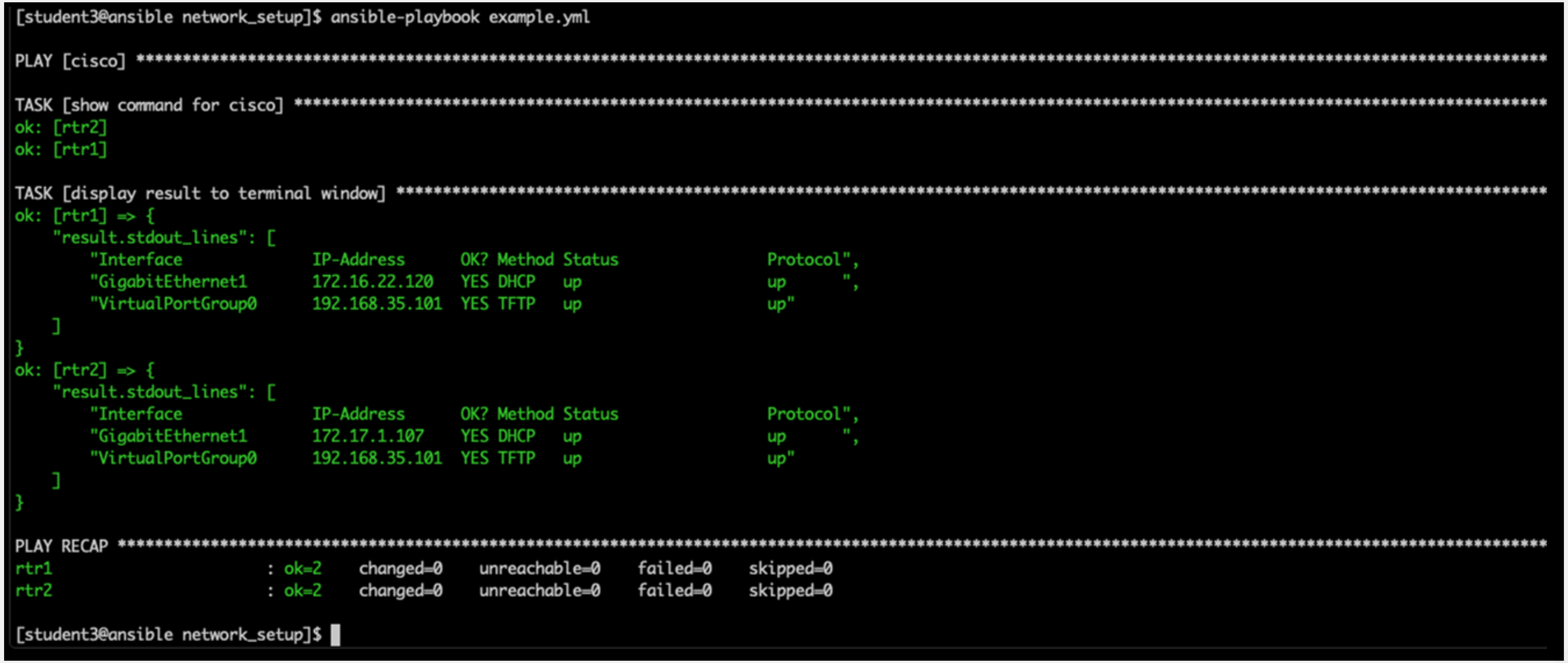
ENSURE AN ON-GOING STEADY-STATE



AUTOMATION FOR EVERYONE: **NETWORK ENGINEERS**

```
---
- hosts: cisco
  gather_facts: false
  connection: network_cli

  tasks:
       - name: show command for cisco
         cli_command:
        command: show ip int br
         register: result

       - name: display result to terminal window
         debug:
        var: result.stdout_lines
```

AUTOMATION FOR EVERYONE: **PLAYBOOK RESULTS**

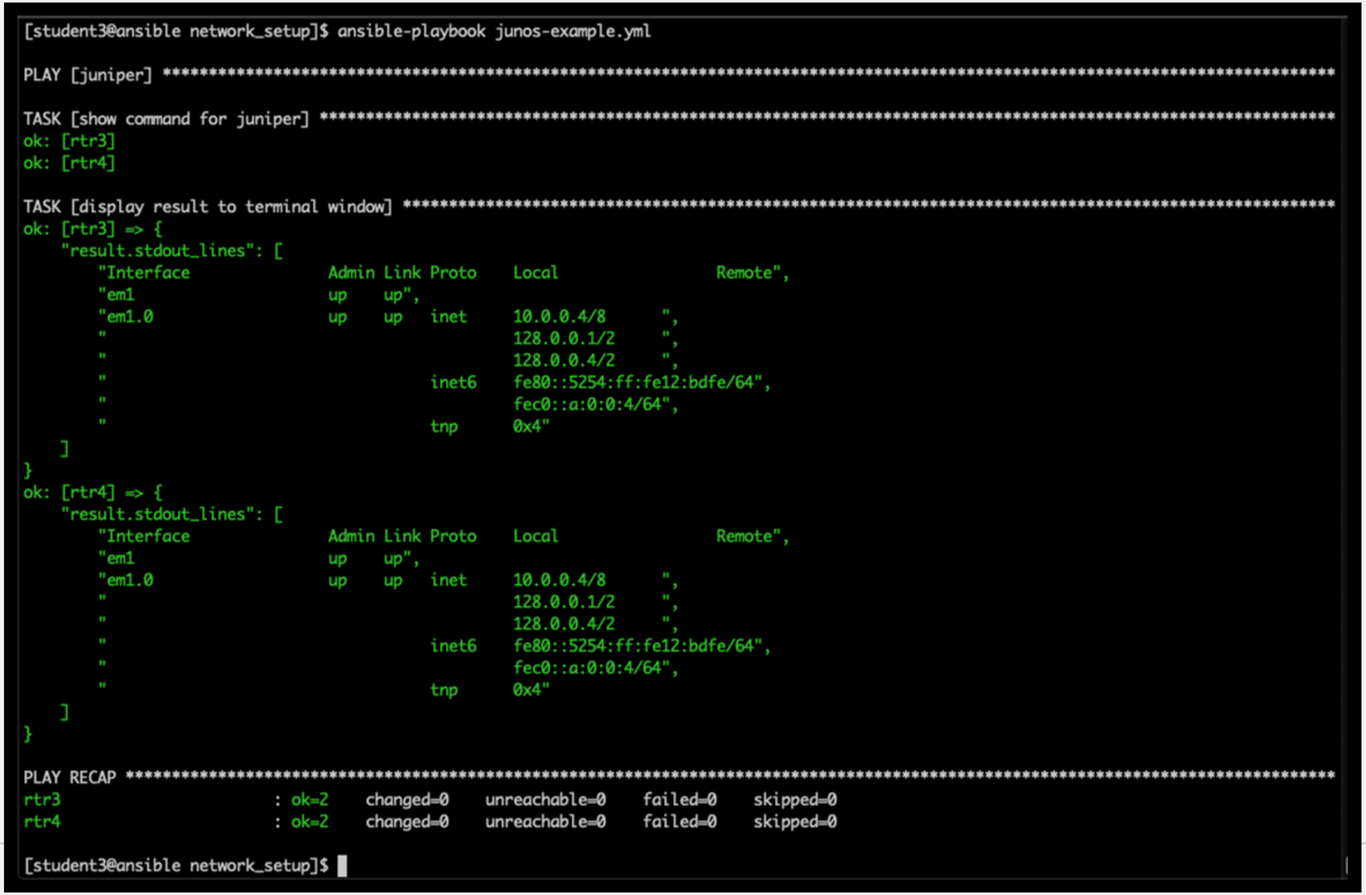


```
[student3@ansible network_setup]$ ansible-playbook example.yml

PLAY [cisco] ***************************************************************************************

TASK [show command for cisco] **********************************************************************
ok: [rtr2]
ok: [rtr1]

TASK [display result to terminal window] ***********************************************************
ok: [rtr1] => {
    "result.stdout_lines": [
        "Interface              IP-Address      OK? Method Status                Protocol",
        "GigabitEthernet1       172.16.22.120   YES DHCP   up                    up      ",
        "VirtualPortGroup0      192.168.35.101  YES TFTP   up                    up"
    ]
}
ok: [rtr2] => {
    "result.stdout_lines": [
        "Interface              IP-Address      OK? Method Status                Protocol",
        "GigabitEthernet1       172.17.1.107    YES DHCP   up                    up      ",
        "VirtualPortGroup0      192.168.35.101  YES TFTP   up                    up"
    ]
}

PLAY RECAP *****************************************************************************************
rtr1                       : ok=2    changed=0    unreachable=0    failed=0    skipped=0
rtr2                       : ok=2    changed=0    unreachable=0    failed=0    skipped=0

[student3@ansible network_setup]$
```

AUTOMATION FOR EVERYONE: **NETWORK ENGINEERS**

```
---
- hosts: juniper
  gather_facts: false
  connection: network_cli

  tasks:
      - name: show command for juniper
        cli_command:
      command: show interfaces terse em1
        register: result

      - name: display result to terminal window
        debug:
      var: result.stdout_lines
```

AUTOMATION FOR EVERYONE: **PLAYBOOK RESULTS**

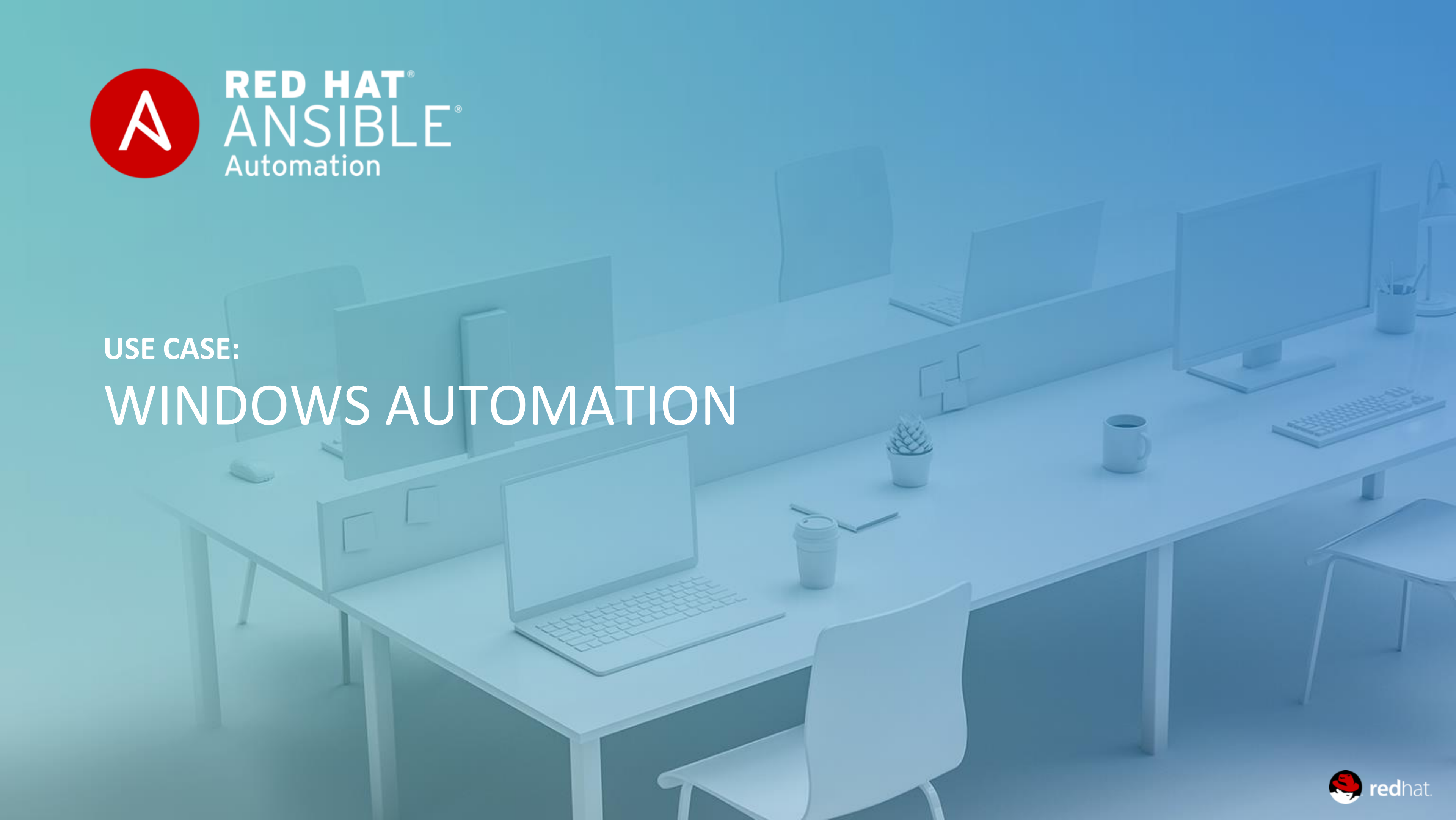


```
[student3@ansible network_setup]$ ansible-playbook junos-example.yml

PLAY [juniper] *****************************************************************************************************************************

TASK [show command for juniper] ************************************************************************************************************
ok: [rtr3]
ok: [rtr4]

TASK [display result to terminal window] ***************************************************************************************************
ok: [rtr3] => {
    "result.stdout_lines": [
        "Interface               Admin Link Proto    Local                 Remote",
        "em1                     up    up",
        "em1.0                   up    up   inet     10.0.0.4/8      ",
        "                                            128.0.0.1/2     ",
        "                                            128.0.0.4/2     ",
        "                                   inet6    fe80::5254:ff:fe12:bdfe/64",
        "                                            fec0::a:0:0:4/64",
        "                                   tnp      0x4"
    ]
}
ok: [rtr4] => {
    "result.stdout_lines": [
        "Interface               Admin Link Proto    Local                 Remote",
        "em1                     up    up",
        "em1.0                   up    up   inet     10.0.0.4/8      ",
        "                                            128.0.0.1/2     ",
        "                                            128.0.0.4/2     ",
        "                                   inet6    fe80::5254:ff:fe12:bdfe/64",
        "                                            fec0::a:0:0:4/64",
        "                                   tnp      0x4"
    ]
}

PLAY RECAP *********************************************************************************************************************************
rtr3                       : ok=2    changed=0    unreachable=0    failed=0    skipped=0
rtr4                       : ok=2    changed=0    unreachable=0    failed=0    skipped=0

[student3@ansible network_setup]$
```

RED HAT ANSIBLE Automation

USE CASE:

WINDOWS AUTOMATION



WINDOWS AUTOMATION

90+

Windows Modules

1,300+

Powershell DSC resources

ansible.com/windows

AUTOMATION FOR EVERYONE: **WINDOWS ADMINS**

```yaml
---
- name: windows playbook
  hosts: new_servers

  tasks:
  - name: ensure local admin account exists
    win_user:
      name: localadmin
      password: '{{ local_admin_password }}'
      groups: Administrators
```

AUTOMATION FOR EVERYONE: **WINDOWS ADMINS**

```
---
- name: windows playbook
  hosts: windows_machines

  tasks:
  - name: ensure common tools are installed
    win_chocolatey:
      name: '{{ item }}'
    loop: ['sysinternals', 'googlechrome']
```

AUTOMATION FOR EVERYONE: **WINDOWS ADMINS**

```
---
- name: update and reboot
  hosts: windows_servers
  tasks:
  - name: ensure common OS updates are current
    win_updates:
    register: update_result

  - name: reboot and wait for host if updates change require it
    win_reboot:
    when: update_result.reboot_required
```

redhat.

AUTOMATION FOR EVERYONE: **WINDOWS ADMINS**

```yaml
---
- name: update domain and reboot
  hosts: windows_servers
  tasks:
  - name: ensure domain membership
    win_domain_membership:
      dns_domain_name: contoso.corp
      domain_admin_user: '{{ domain_admin_username }}'
      domain_admin_password: '{{ domain_admin_password }}'
      state: domain
    register: domain_result

  - name: reboot and wait for host if domain change require it
    win_reboot:
    when: domain_result.reboot_required
```

RED HAT ANSIBLE Automation

USE CASE:

CLOUD AUTOMATION

CLOUD AUTOMATION

800+

Cloud
Modules

30+

Cloud Platforms

ansible.com/cloud

PLAYBOOK EXAMPLE: **AWS**

```
---
- name: aws playbook
  hosts: localhost
  connection: local

  tasks:
        - name: create AWS VPC ansible-vpc
          ec2_vpc_net:
      name: "ansible-vpc"
      cidr_block: "192.168.0.0/24"
      tags:
        demo: the demo vpc
    register: create_vpc
```

redhat.

PLAYBOOK EXAMPLE: **AZURE**

```
---
- name: azure playbook
  hosts: localhost
  connection: local

  tasks:
          - name: create virtual network
      azure_rm_virtualnetwork:
        resource_group: myResourceGroup
        name: myVnet
        address_prefixes: "10.0.0.0/16"
```

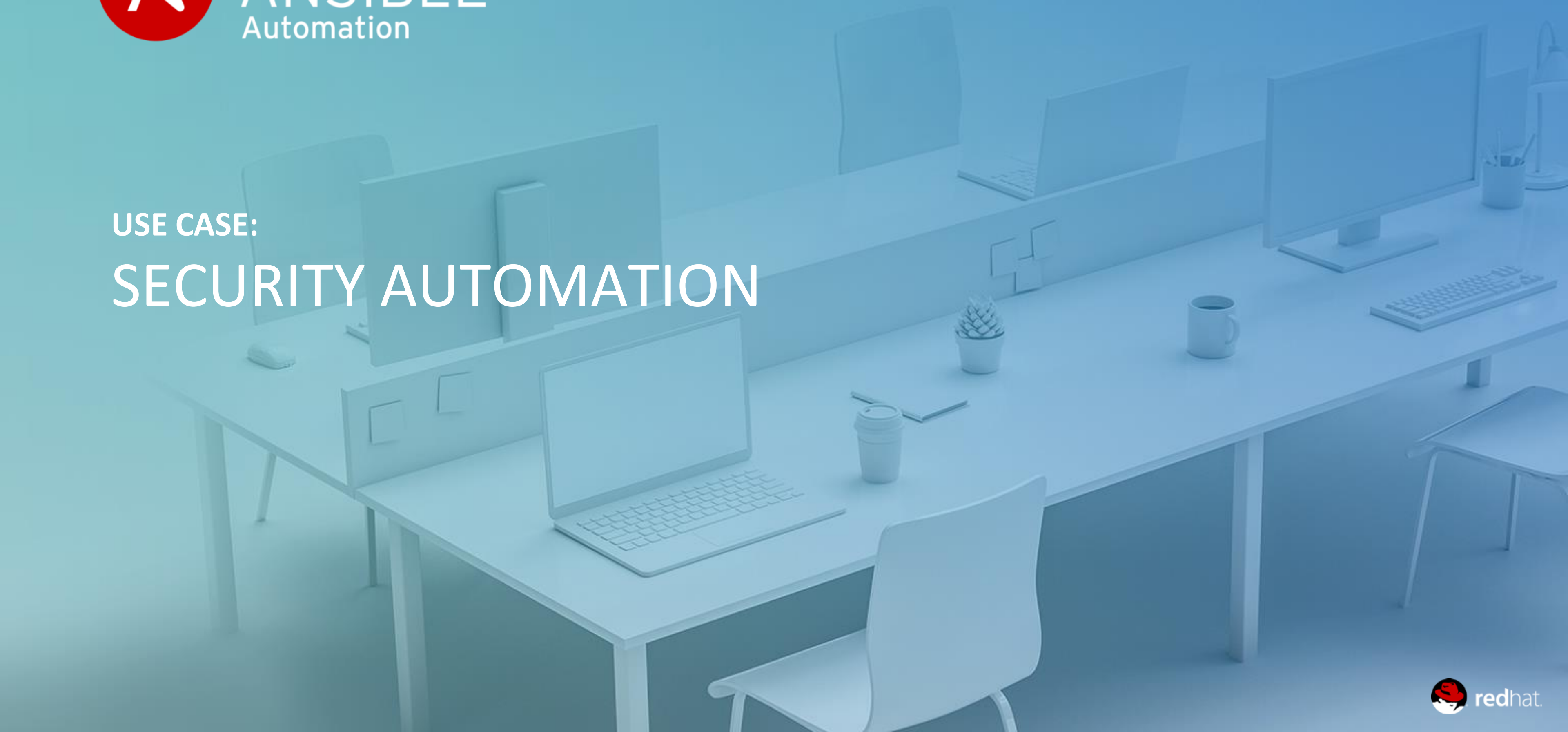
PLAYBOOK EXAMPLE: **RED HAT OPENSTACK**

```
---
- name: openstack playbook
  hosts: localhost
  connection: local

  tasks:
    - name: launch an instance
         os_server:
           name: vm1
           cloud: mordred
           region_name: ams01
           image: Red Hat Enterprise Linux 7.4
           flavor_ram: 4096
```

RED HAT ANSIBLE Automation

USE CASE:

SECURITY AUTOMATION



WHAT IS IT?

Ansible Security Automation is a supported set of Ansible modules, roles and playbooks designed to unify the security response to cyberattacks in a new way - by orchestrating the activity of multiple classes of security solutions that wouldn't normally integrate with each other.

WHAT DOES IT DO?

Through Ansible Security Automation, IT organizations can address multiple popular use cases:

- For **detection and triage of suspicious activities**, for example, Ansible can automatically enable logging or increase the log verbosity across enterprise firewalls and IDS to enrich the alerts received by a SIEM for an easier triage.
- For **threat hunting,** for example, Ansible can automatically create new IDS rules to investigate the origin of a firewall rule violation, and whitelist those IP addresses recognized as non threats.
- For **incident response,** for example, Ansible can automatically validate a threat by verifying an IDS rule, trigger a remediation from the SIEM solution, and create new enterprise firewall rules to blacklist the source of an attack.

At launch, Red Hat's Ansible security automation platform provides support for:

- **Check Point** – Next Generation Firewall (NGFW);
- **Splunk** – Splunk Security Enterprise (SE);
- **Snort**

WHO IS IT FOR?

Ansible Security Automation extends the Ansible agentless, modular and easy to use enterprise automation platform to support the following industry constituencies:

- **End-user organizations' security teams** in charge of Security Operations Centres (SOCs)
- **Managed security service providers (MSSPs)** responsible for the governance of thousands of enterprise security solutions across their whole customer base
- **Security ISVs** offering security orchestration and automation (SOAR) solutions currently using custom-made automation frameworks

AUTOMATION FOR EVERYONE: **SECURITY OPERATIONS**

```yaml
---
- name: checkpoint playbook
  hosts: checkpoint
  connection: httpapi

  tasks:
    - name: create access rule
      checkpoint_access_rule:
             layer: Network
             name: "Drop attacker"
             position: top
             source: attacker
             destination: Any
             action: Drop
```

AUTOMATION FOR EVERYONE: **SECURITY OPERATIONS**

```yaml
---
- name: checkpoint playbook
  hosts: checkpoint
  connection: httpapi

  tasks:
    - name: delete access rule
      checkpoint_access_rule:
        layer: Network
        name: "Drop attacker"
        state: absent
```

NEXT STEPS

GET STARTED

ansible.com/get-started

ansible.com/tower-trial

JOIN THE COMMUNITY

ansible.com/community

WORKSHOPS & TRAINING

ansible.com/workshops

Red Hat Training

SHARE YOUR STORY

Follow us @Ansible

Friend us on Facebook