RED HAT CONSULTING

Red Hat

Ansible Tower and Use Cases Review

- Concept Review
- Customer Use Cases

Prepared by:

MATT NOLAN
PRINCIPAL ARCHITECT
STP - Cloud Practice
mnolan@redhat.com

Ansible Tower Concepts

- **Modules**
    - Ansible ships with a number of modules (*the 'module library'*) that can be executed directly on remote hosts or through playbooks.
    - Modules can control system resources, like services, packages, or files (anything really), or handle executing system commands.
- **Modules Guide:**
    - https://docs.ansible.com/ansible/latest/user_guide/modules.html

----

- **Playbooks**
    - Playbooks are Ansible's configuration, deployment, and orchestration language to describe a policy you want your remote systems to enforce, or a set of steps in an automated process.
    - If Ansible modules are the tools in your workshop, playbooks are your instruction manuals, and your inventory of hosts are your raw material.
- **Playbooks Guide:**
    - https://docs.ansible.com/ansible/latest/user_guide/playbooks.html

----

- **Roles**
    - Roles are ways of automatically loading certain vars_files, tasks, and handlers based on a known file structure. Grouping content by roles also allows easy sharing of roles with other users.
- **Roles Guide:**
    - https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

----

- **Projects**
    - A Project is a logical collection of Ansible playbooks, represented in Tower.

    - You can manage playbooks and playbook directories by either placing them manually under the Project Base Path on your Tower server, or by placing your playbooks into a source code management (SCM) system supported by Tower, including Git, Subversion, Mercurial, and Red Hat Insights.
- **Projects Guide:**
    - **https://docs.ansible.com/ansible-tower/latest/html/userguide/projects.html**

- **Job Templates**
    - A job template is a definition and set of parameters for running an Ansible job. Job templates are useful to execute the same job many times. Job templates also encourage the reuse of Ansible playbook content and collaboration between teams. While the REST API allows for the execution of jobs directly, Tower requires that you first create a job template.
- **Job Templates Guide:**
    - https://docs.ansible.com/ansible-tower/latest/html/userguide/job_templates.html

----

- **Workflows**
    - Workflows allow you to configure a sequence of disparate job templates (or workflow templates) that may or may not share inventory, playbooks, or permissions. However, workflows have ‘admin’ and ‘execute’ permissions, similar to job templates. A workflow accomplishes the task of tracking the full set of jobs that were part of the release process as a single unit.
- **Workflows Guide:**
    - https://docs.ansible.com/ansible-tower/latest/html/userguide/workflows.html

----

- **Inventories**
    - An Inventory is a collection of hosts against which jobs may be launched, the same as an Ansible inventory file. Inventories are divided into groups and these groups contain the actual hosts. Groups may be sourced manually, by entering host names into Tower, or from one of Ansible Tower’s supported cloud providers.
- **Inventories Guide:**
    - https://docs.ansible.com/ansible-tower/latest/html/userguide/inventories.html

----

Storage Automation

Supporting Modules:

- https://docs.ansible.com/ansible/latest/modules/list_of_storage_modules.html

Cloud Automation

Use Cases:

- **Amazon**
    - Manage AWS Gateway API's
    - Create, Copy, Terminate, Start, Stop, Snapshot and Register EC2 instances
    - Create, Delete, Autoscale Groups
    - AMI Search and Fact Gathering on EC2
    - Manage S3 Buckets and Objects and Configure Security Policies
    - Create/Delete and obtain facts for Cloudformation
    - Create/Delete, gather facts and manage subnets for AWS VPC
    - Manage ecs (elastic container service) clusters and services
    - Manage cache clusters, security groups, subnet groups and snapshots for Amazon Elasticache
    - Gather facts, create/destroy and manage AWS ELB groups and Instances
    - Execute Lambda functions
    - Gather facts, manage subnet groups and create/delete/modify Amazon Redshift
    - Add entries for Routes and manage health checks

- **Azure**
    - Create/Terminate & manage virtual machines and container services
    - Gather DNS record set facts and manage Azure DNS zones
    - Gather facts and manage Azure load balancers
    - Manage Azure virtual networks, subnets and network interfaces
    - Manage Azure resource and security groups
    - Manage Azure Public IP Addresses
    - Manage Azure Function Apps
    - Manage Azure Storage Accounts
    - Gather facts and manage policy for VM Scale Set facts

- **Google**
    - Create / Terminate and manage GCE instances
    - Utilize GCE Image Resources
    - Create / Destroy and manage GCE load-balancer resources
    - Create and remove zones in Google Cloud DNS

    - Create / Update / Destroy Healthcheck, Target Proxy, URL Map, Forwarding Rule, and Backend Services
    - Create / Update / Destroy a managed instance group

Supporting Modules:

- https://docs.ansible.com/ansible/latest/modules/list_of_cloud_modules.html

Windows Automation:

Use Cases:

- **OS Hardening**
- **Antivirus Configuration**
- **Installing Software**
  There are three main ways that Ansible can be used to install software:
  - Using the win_chocolatey module
    - This sources the program data from the default public Chocolatey repository.
    - Internal repositories can be used instead by setting the source option.
    - The win_chocolatey module is recommended since it has the most complete logic for checking to see if a package has already been installed and is up-to-date.
  - Using the win_package module
    - This installs software using an MSI or .exe installer from a local/network path or URL.
  - Using the win_command or win_shell module to run an installer manually.

  - **NOTE:** Some installers like Microsoft Office or SQL Server require credential delegation or access to components restricted by WinRM. The best method to bypass these issues is to use become with the task. With become, Ansible will run the installer as if it were run interactively on the host.

- **Installing Updates**
  - The win_updates and win_hotfix modules can be used to install updates or hotfixes on a host.
  - The module win_updates is used to install multiple updates by category
  - The module win_hotfix can be used to install a single update or hotfix file that has been downloaded locally.
  - The win_hotfix module has a requirement that the DISM PowerShell cmdlets are present.

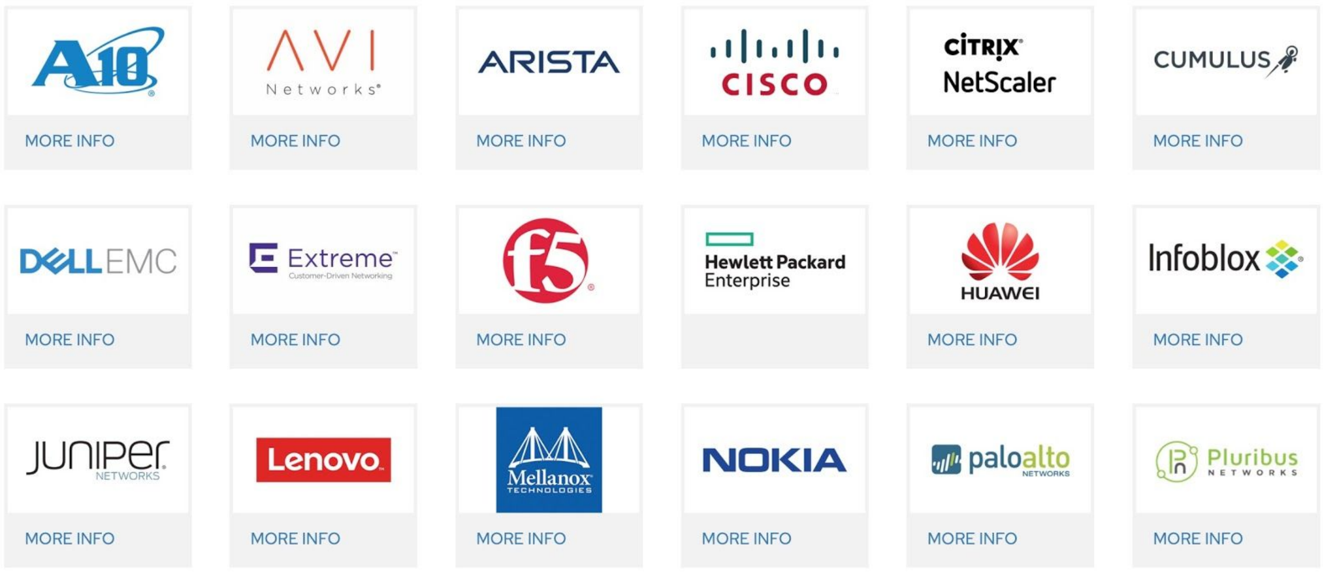
- **Set Up Users and Groups**
  Ansible can be used to create Windows users and groups both locally and on a domain.
  - **Local**
    - The modules win_user, win_group and win_group_membership manage Windows users, groups and group memberships locally
  - **Domain**
    - The modules win_domain_user and win_domain_group manages users and groups in a domain.

- <u>Running Commands</u>
  In cases where there is no appropriate module available for a task, a command or script can be run using the following modules:
  - win_shell
    - The win_shell module is used to execute commands within a shell.
  - win_command
    - The win_command module is used to execute a command which is either an executable or batch file
  - raw
    - The raw module simply executes a Powershell command remotely.
    - Since raw has none of the wrappers that Ansible typically uses, become, async and environment variables do not work.
  - script
    - The script module executes a script from the Ansible controller on one or more Windows hosts.
    - Like raw, script currently does not support become, async or environment variables.
  - <u>Choosing Command or Shell</u>
    - The win_shell and win_command modules can both be used to execute a command or commands.
    - The win_shell module is run within a shell-like process like PowerShell or cmd, so it has access to shell operators like <, >, |, ;, &&, and ||.
    - Multi-lined commands can also be run in win_shell.
    - The win_command module simply runs a process outside of a shell.
    - It can still run a shell command like mkdir or New-Item by passing the shell commands to a shell executable like cmd.exe or PowerShell.exe.
    - Some commands like mkdir, del, and copy only exist in the CMD shell. To run them with win_command they must be prefixed with cmd.exe /c.

- <u>Developing Windows Modules</u>
  - Because Ansible modules for Windows are written in PowerShell, the development guides for Windows modules differ substantially from those for standard standard modules.
  - Please see Windows module development walkthrough for more information.

<u>Supporting Docs:</u>
- User Guide
- Working With Playbooks
- Best Practices
- List of Windows Modules

Network Automation:



Use Cases

- Manage and Configure Load Balancers (LTMs / GTMs)
  - F5, Citrix, AVI and more
- Manage virtual servers / members / pools
- Manage and configure physical route and switch appliances
  - Arista, Cisco, Juniper and many more
- Manage and Configure Firewall Appliances
  - Cisco, Palo Alto, Arista, etc..
- Manage and configure SDN controllers (Cisco ACI, Cisco NSO, Arista Cloudvision)
- Collect Facts from all Network Devices
- Network Device Configuration Management
  - Verify Configuration
  - Apply Configuration
- Manage Network Device Configuration Backups
- Manage IPAM to Get and Release IP Addresses
- Manage device firmware upgrades
- Configuration Management of routers, switches
- Configuration backups
- ServiceNow integration
- Workflow Integration
  - Integrate switch port activations, adding VLANs, verifying interface speed/duplex setting checks into existing IT workflows

Supporting Modules:

- https://docs.ansible.com/ansible/latest/modules/list_of_cloud_modules.html\

DevOps & CI/CD:

Use Cases:

- Integrate directly with GIT, BitBucket, Jenkins, Team City, Bamboo, Travis CI for automated continuous deployment
- Rollback and forward application versions on demand with an automated deployment process
- Write stateful automation as code to manage application environments which can be stored alongside application source code
- Deployment strategy automation
  - canary, blue-green, rolling, etc

Supporting Links:

- https://www.ansible.com/use-cases/continuous-delivery

Security and Compliance:

Rationale:

- Define your systems for security in code
- Define and secure any part of the system, whether it's setting firewall rules, locking down users and groups, or applying custom security policies.
- Once you've defined your security configuration, you need to be able to verify it and verify it on a consistent basis.
- Ansible's idempotent nature means you can repeatedly apply the same configuration, and it will only make the necessary changes to put the system back into compliance.
- By investigating these runs, you can easily see where changes are needed.
- When the need arrives to perform a one-off task like quickly applying a security patch from a vendor, Ansible's command support allows you to get things done across your infrastructure with one simple command.

Use Cases:

- Apply regulatory security baselines
  - STIGs (*Security Technical Implementation Guides*)
  - PCI DSS (*Payment Card Industry Data Security Standard*)
- Network device hardening

- Validation and Remediation
- Enforcing security baseline standards
- System state tracking with Ansible Tower
- Incident notifications and response
- Package management
- Patch management
- OS hardening to a security compliance baseline at provisioning time for consistency and OS immutability
- Infrastructure as Code and Security as Code
- Repeatability, ability to share and verify and help with passing security and compliance audits

Supporting Modules:

- https://www.ansible.com/use-cases/security-and-compliance

Additional References:

- **Website:** Ansible General GitHub site
    - https://github.com/ansible
- **Purpose:**
    - Ansibles open source GitHub site for miscellaneous resources

----

- **Website:** Galaxy
    - **https://galaxy.ansible.com**
- **Purpose:**
    - Galaxy is a free site for finding, downloading, and sharing community developed roles.
    - Downloading roles from Galaxy is a great way to jumpstart your automation projects

----

- **Github:** Molecule
    - https://github.com/ansible/molecule
- **Purpose:**
    - Molecule is designed to aid in the development and testing of Ansible roles.
    - Molecule provides support for testing with multiple instances, operating systems and distributions, virtualization providers, test frameworks and testing scenarios.
    - Molecule encourages an approach that results in consistently developed roles that are well-written, easily understood and maintained.
- **Documentation:**
    - https://molecule.readthedocs.io/.

- **Github: Ansible Inventory Plugins**
    - https://github.com/ansible/ansible/tree/devel/lib/ansible/plugins/inventory
- Purpose:
    - Good resource for getting Ansible dynamic inventory scripts
    - Several scripts are available depending on the inventory source being targeted

----

GITHUB
github.com/ansible/ansible-examples

LAMP + HAPROXY + NAGIOS
github.com/ansible/ansible-examples/tree/master/lamp_haproxy

WINDOWS
github.com/ansible/ansible-examples/tree/master/windows

SECURITY COMPLIANCE
github.com/ansible/ansible-lockdown

NETWORK AUTOMATION
ansible.com/linklight
github.com/network-automation

