Ansible Pull

Ansible Advanced

Ansible Pull

- Ansible is typically known for its push-based architecture, where the control node pushes configuration and commands to the managed nodes
- However, Ansible also offers a pull-based approach, known as **`ansible-pull`**, which essentially reverses this process
- **`ansible-pull`** is a command provided by Ansible that allows managed nodes to pull their configurations from a central repository and apply them locally
- Instead of the control node pushing configurations to the managed nodes, the nodes themselves will fetch their configurations and apply them.

Key Features and Benefits

- Decentralized Execution
  - With ansible-pull, there's no need for a central control node to manage configurations
  - Each node is responsible for pulling its own configuration
- Scalability
  - This model can be more scalable for environments with a large number of nodes, as it avoids potential bottlenecks associated with a single control node pushing to many managed nodes
- Flexibility
  - Nodes can be configured to pull updates at different intervals or times, allowing for staggered updates across the infrastructure
- Self-healing
  - Nodes can be set up to periodically pull their configurations, ensuring that they auto-correct any drift from the desired state
- Use of Version Control
  - ansible-pull typically pulls playbooks from a version control system (like Git), ensuring that nodes are always using the latest configurations and allowing for easy tracking of changes

How to use

- For example, to pull a playbook from a Git repository and apply it:

```
ansible-pull -U https://github.com/username/my_ansible_repo.git
```

- Workflow:
  - A node runs the **`ansible-pull`** command, which clones (or updates) a specified Git repository to a local directory
  - **`ansible-pull`** looks for a playbook in the repository
  - By default, it looks for a playbook named **`localhost.yml`**, but you can specify a different name using the **`-d`** option.
  - The playbook is executed locally on the node, applying the desired configuration

Common Use Cases

- Edge Devices
  - For devices or nodes that are distributed geographically and might not always be reachable for centralized management
- Cloud Auto-scaling
  - In cloud environments where new instances are dynamically created, these instances can be configured to run ansible-pull upon startup to fetch and apply their configurations
- Periodic Configuration Checks
  - Nodes can be set up to run ansible-pull via cron jobs or scheduled tasks to periodically check and ensure they are in the desired state

Push vs. Pull

- Initiation
  - Push: Control node initiates and pushes configurations to nodes
  - Pull: Nodes self-initiate and pull configurations from a repository
- Architecture
  - Push: Centralized
  - Pull: Decentralized
- Connection Management
  - Push: Control node manages connections to all nodes
  - Pull: Nodes individually connect to the central repository

Push vs. Pull

- Best Suited For
  - Push: Regular configuration management, dynamic environments, centralized logging
  - Pull: Edge devices, cloud auto-scaling, disconnected or occasionally connected environments
- Configuration Source
  - Push: Playbooks and roles stored on the control node or fetched from a repository
  - Pull: Typically a version control system like Git
- Execution Control
  - Push: Granular control from the control node, allowing targeted runs
  - Pull: Nodes run based on schedule or triggers, with less granular control from a central location