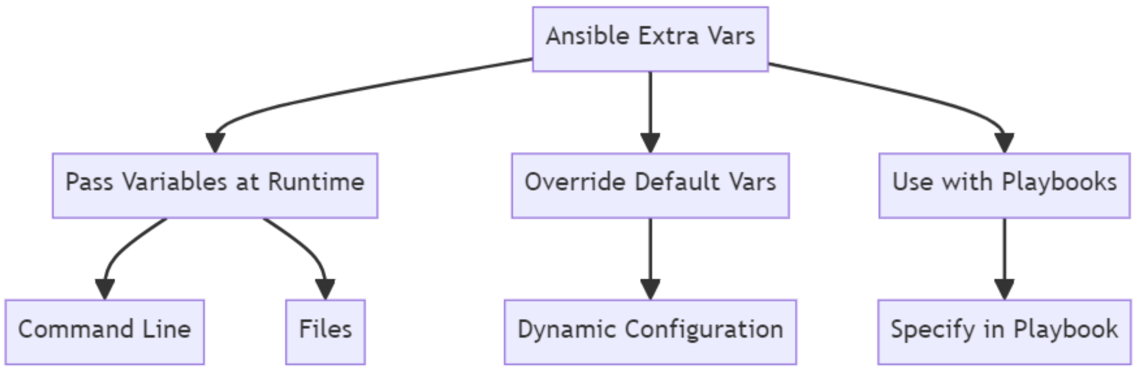
Ansible Cheat Sheet with Real-Time Use Cases - Part 3 🚀

Introduction:

These advanced scenarios cover a range of use cases from environment-specific configurations to CI/CD pipeline integration and advanced error handling. Feel free to adapt and customize them based on your specific automation needs.

41. Multi-Environment Configuration

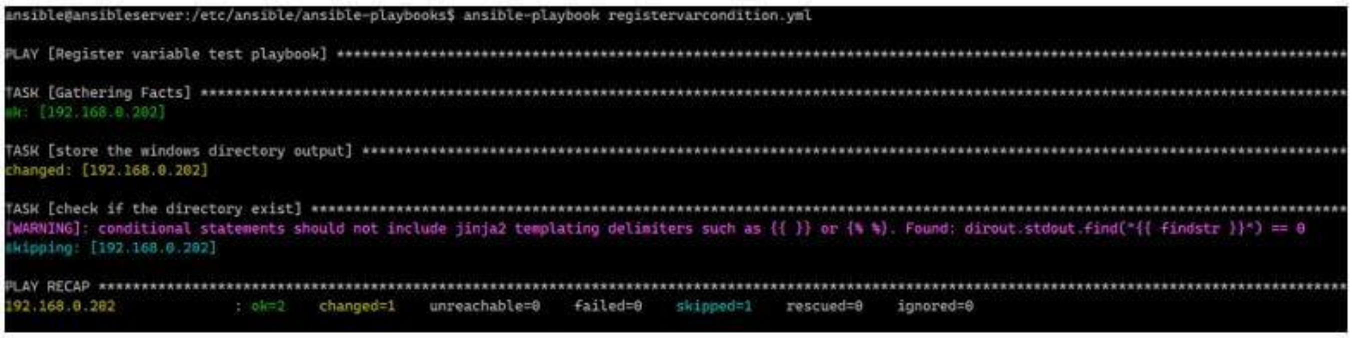
Command: `--extra-vars`



Use Case: Dynamically configure settings for different environments during playbook execution.

```
ansible-playbook deploy.yaml --extra-vars "env=staging"
```

42. Conditional Task Execution



```
ansible@ansibleserver:/etc/ansible/ansible-playbooks$ ansible-playbook registervarcondition.yml

PLAY [Register variable test playbook] ****************************************************************

TASK [Gathering Facts] ********************************************************************************
ok: [192.168.0.202]

TASK [store the windows directory output] *************************************************************
changed: [192.168.0.202]

TASK [check if the directory exist] *******************************************************************
[WARNING]: conditional statements should not include jinja2 templating delimiters such as {{ }} or {% %}. Found: dirout.stdout.find("{{ findstr }}") == 0
skipping: [192.168.0.202]

PLAY RECAP ********************************************************************************************
192.168.0.202              : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

Command: `when`

Use Case: Execute tasks conditionally based on certain criteria.

```
- name: Perform task only on Linux
  command: /path/to/linux_script.sh
  when: "'Linux' in ansible_os_family"
```

43. Docker Container Orchestration

Command: `docker_container`

Use Case: Orchestrate Docker containers with Ansible for seamless deployment and management.

```
- name: Start Docker container
  docker_container:
    name: my_container
    image: my_image:latest
    state: started
```

44. Managing SSH Keys

Command: `authorized_key`

Use Case: Manage SSH keys on remote hosts for secure access.

```
- name: Add SSH key for user
  authorized_key:
    user: my_user
    key: "{{ lookup('file', '~/.ssh/id_rsa.pub') }}"
```

45. System Monitoring Setup

Command: `cron`

Use Case: Schedule periodic system monitoring tasks with Ansible.

```
- name: Schedule monitoring script
  cron:
    name: "Run Monitoring Script"
    minute: "0"
    hour: "*/2"
    job: "/path/to/monitoring_script.sh"
```

46. Service Discovery Integration

Command: `uri`

Use Case: Integrate Ansible with service discovery tools for dynamic inventory updates.

```
- name: Register server with Consul
  uri:
    url: "http://consul-server:8500/v1/agent/service/register"
    method: POST
    body: '{"ID": "web-server", "Name": "web", "Port": 80}'
```

47. Custom Security Policies

Command: `lineinfile`

Use Case: Enforce custom security policies by modifying specific lines in configuration files.

```
- name: Set custom security policy
  lineinfile:
    path: /etc/security/policy.conf
    line: "ENFORCE_CUSTOM_POLICY=yes"
```

48. CI/CD Pipeline Integration (GitLab)

Command: Use Ansible playbooks in GitLab CI/CD pipelines.

Use Case: Seamless integration of Ansible with GitLab for continuous deployment.

```
stages:
  - deploy

deploy:
  script:
    - ansible-playbook deploy.yaml
```

49. Managing Environment Variables

Command: `environment`

Use Case: Set environment variables for tasks during playbook execution.

```
- name: Run script with custom environment
  command: /path/to/script.sh
  environment:
    CUSTOM_VAR: "value"
```

50. Advanced Error Handling

Command: `block`, `rescue`, `always`

Use Case: Implement advanced error handling with try-catch blocks.

```
- name: Advanced Error Handling
  block:
    - name: Attempt a risky task
      command: /path/to/risky_script.sh
  rescue:
    - name: Handle errors gracefully
      debug:
        msg: "Task failed, but we caught it!"
  always:
    - name: Always run cleanup task
      command: /path/to/cleanup_script.sh
```

51. Network Device Configuration

Command: `ios_command` (for Cisco devices)

Use Case: Automate configuration changes on network devices.

```
- name: Configure Cisco router
  ios_command:
    commands:
      - interface GigabitEthernet0/0
      - ip address 192.168.1.1 255.255.255.0
  delegate_to: network_device
```

52. Cloud Resource Provisioning (AWS)

Command: `ec2_instance`

Use Case: Dynamically provision EC2 instances on AWS.

```
- name: Launch EC2 instance
  ec2_instance:
    name: my_instance
    instance_type: t2.micro
    image: ami-12345678
    key_name: my_key
    count: 1
    tags:
      - key: Name
        value: MyInstance
```

53. Custom Module Development

Command: `ansible.module_utils`

Use Case: Develop custom Ansible modules to extend functionality.

```
# Custom module_utils/my_module.py
def my_custom_function():
    return "Hello from my custom module!"

# Playbook using the custom module
- name: Use custom module
  debug:
    msg: "{{ lookup('module_utils', 'my_module.my_custom_function') }}"
```

54. HashiCorp Vault Integration

Command: `hashi_vault`

Use Case: Integrate Ansible with HashiCorp Vault for secure secret management.

```
- name: Retrieve secret from Vault
  hashi_vault:
    path: secret/data/myapp
    field: api_key
  register: vault_result

- name: Use secret in playbook
  debug:
    msg: "API Key: {{ vault_result.secret }}"
```

55. Windows Server Configuration

Command: `win_command`, `win_shell`

Use Case: Configure settings on Windows servers.

```
- name: Set Windows registry key
  win_shell: New-ItemProperty -Path 'HKLM:\Software\MyApp' -Name 'Setting'
-Value '123' -PropertyType 'String'
```

56. Custom Logging and Auditing

Command: `log_callback`

Use Case: Implement a custom callback plugin for detailed logging.

```
export ANSIBLE_STDOUT_CALLBACK=my_custom_logging_callback
ansible-playbook deploy.yaml
```

57. Container Orchestration (Kubernetes)

Command: `k8s`

Use Case: Deploy and manage applications in Kubernetes clusters.

```
- name: Deploy app in Kubernetes
  k8s:
    state: present
    definition:
      apiVersion: v1
      kind: Pod
      metadata:
        name: mypod
      spec:
        containers:
          - name: mycontainer
            image: myimage:latest
```

58. Git Repository Management

Command: `git_config`

Use Case: Configure Git settings for automation.

```
- name: Set Git configuration
  git_config:
    key: user.email
    value: "ansible@company.com"
  delegate_to: localhost
```

59. Service Scaling in Cloud (Azure)

Command: `azure_rm_virtualmachine`

Use Case: Dynamically scale the number of virtual machines in Azure.

```
- name: Scale VM instances in Azure
  azure_rm_virtualmachine:
    resource_group: my_resource_group
    name: my_vm
    vm_size: Standard_DS2_v2
    count: 3
    state: present
```

60. Dynamic Inventory with Service Discovery

Command: `script` (for custom dynamic inventory script)

Use Case: Develop a custom dynamic inventory script for fetching host information.

```
ini
# Custom inventory script (custom_inventory.py)
#!/usr/bin/env python
print('{"web_servers": ["server1", "server2"], "app_servers": ["server3"],
"all": {"children": ["web_and_app_servers"]}}')
bash
ansible-inventory -i custom_inventory.py --list
```

Conclusion:

In this Troubleshooting scenarios we have cover a range of advanced use cases, including network device configuration, cloud resource provisioning, custom module development, and integration with external tools like HashiCorp Vault and Kubernetes. Feel free to adapt these examples to suit your specific automation requirements.