
DevOps Shack

# 20 Real-Time Use-Cases of Ansible With Detailed Examples

1. **Infrastructure Provisioning**: Use Ansible to automate the provisioning of infrastructure on cloud platforms like AWS, Azure, or GCP. For example, provisioning EC2 instances on AWS:

```
- name: Provision EC2 instance
  hosts: localhost
  tasks:
    - name: Launch EC2 instance
      ec2_instance:
        key_name: mykey
        instance_type: t2.micro
        image: ami-123456
        region: us-west-2
        count: 1
        state: present
      register: ec2
```

2. **Configuration Management**: Manage configurations across multiple servers to ensure consistency and compliance. For instance, ensuring NTP service is running and configured correctly:

```
- name: Ensure NTP service is running
  hosts: all
  tasks:
    - name: Ensure NTP is installed
      yum:
        name: ntp
```

```
      state: present
  - name: Start NTP service
    service:
      name: ntpd
      state: started
      enabled: yes
```

3. **Software Installation**: Automate the installation and configuration of software packages across your infrastructure. For example, installing Apache web server:

```
- name: Install Apache web server
  hosts: webservers
  tasks:
    - name: Install Apache
      yum:
        name: httpd
        state: present
    - name: Start Apache
      service:
        name: httpd
        state: started
        enabled: yes
```

4. **Continuous Deployment**: Automate the deployment of applications to servers after successful builds. For example, deploying a Docker container:

```
- name: Deploy Docker container
  hosts: web_servers
  tasks:
    - name: Pull Docker image
      docker_image:
        name: myapp
        source: pull
    - name: Run Docker container
      docker_container:
        name: myapp_container
        image: myapp
        state: started
        ports:
          - "80:80"
```

5. **Security Hardening**: Apply security policies and configurations across servers to meet compliance standards. For example, enforcing SSH key-based authentication:

```
- name: Configure SSH key-based authentication
  hosts: all
  tasks:
    - name: Ensure SSH key is present
      authorized_key:
        user: ansible
        key: "{{ lookup('file', '/path/to/ansible.pub') }}"
        state: present
```

6. **Backup and Restore**: Automate backup and restore processes for databases and files. For example, backing up MySQL databases:

```
- name: Backup MySQL databases
  hosts: db_servers
  tasks:
    - name: Dump MySQL databases
      mysql_db:
        state: dump
        name: "{{ item }}"
        target: "/backup/{{ item }}.sql"
      with_items:
        - database1
        - database2
```

7. **Monitoring**: Integrate Ansible with monitoring tools to automate the setup and configuration of monitoring agents and services. For example, installing and configuring Prometheus node exporter:

```
- name: Install Prometheus node exporter
  hosts: monitoring_servers
  tasks:
    - name: Download node exporter
      get_url:
        url:
"https://github.com/prometheus/node_exporter/releases/download/v1.2.0/node_
exporter-1.2.0.linux-amd64.tar.gz"
        dest: "/tmp/node_exporter.tar.gz"
    - name: Extract node exporter
      unarchive:
        src: "/tmp/node_exporter.tar.gz"
        dest: "/opt/"
    - name: Start node exporter
      command: "/opt/node_exporter/node_exporter"
```

8. **Auto Scaling**: Implement auto-scaling solutions by dynamically adding or removing resources based on demand. For example, scaling EC2 instances in AWS:

```
- name: Scale EC2 instances
  hosts: localhost
  tasks:
    - name: Scale out
      ec2_scaling_policy:
        name: scale_out_policy
        adjustment_type: ChangeInCapacity
        scaling_adjustment: 1
    - name: Scale in
      ec2_scaling_policy:
        name: scale_in_policy
        adjustment_type: ChangeInCapacity
        scaling_adjustment: -1
```

9. **Configuration Drift Detection**: Use Ansible to detect and remediate configuration drift across your infrastructure. For example, detecting changes in SSH configurations:

```
- name: Check SSH configuration
  hosts: all
  tasks:
    - name: Check SSHd config
      command: diff -u <(cat /etc/ssh/sshd_config) <(cat /tmp/sshd_config)
      register: ssh_diff
    - name: Notify if changes detected
      debug:
        msg: "Changes detected in SSH configuration"
      when: ssh_diff.stdout_lines
```

10. **Patch Management**: Automate the patching process across servers to ensure they are up to date with the latest security fixes. For example, applying system updates:

```
- name: Apply system updates
  hosts: all
  tasks:
    - name: Update packages
      yum:
        name: "*"
        state: latest
```

11. **Load Balancer Configuration**: Automate the configuration of load balancers to distribute traffic across multiple servers. For example, configuring HAProxy:

```
- name: Configure HAProxy
  hosts: lb_server
  tasks:
    - name: Install HAProxy
      yum:
        name: haproxy
        state: present
    - name: Copy HAProxy configuration
      template:
        src: haproxy.cfg.j2
        dest: /etc/haproxy/haproxy.cfg
      notify:
        - Restart HAProxy
  handlers:
    - name: Restart HAProxy
      service:
        name: haproxy
        state: restarted
```

12. **Secrets Management**: Use Ansible Vault to securely manage sensitive information such as passwords and API keys. For example, encrypting a file containing sensitive data:

```
- name: Encrypt sensitive file
  hosts: localhost
  tasks:
    - name: Encrypt file
      ansible.builtin.encrypt:
        src: /path/to/sensitive_file
        dest: /path/to/sensitive_file.vault
```

13. **Database Management**: Automate database tasks such as backup, restore, and schema changes. For example, creating a MySQL database:

```
- name: Create MySQL database
  hosts: db_server
  tasks:
    - name: Create database
      mysql_db:
        name: mydatabase
```

14. **Continuous Integration**: Integrate Ansible with CI/CD pipelines to automate infrastructure deployment alongside application deployment. For example, deploying infrastructure changes using Jenkins:

```
- name: Deploy infrastructure changes
  hosts: localhost
  tasks:
    - name: Run Ansible playbook
      ansible.builtin.command: ansible-playbook infrastructure.yml
```

15. **Disaster Recovery**: Automate disaster recovery processes to minimize downtime in case of failures. For example, restoring a backup of critical data:

```
- name: Restore backup
  hosts

: db_server
  tasks:
    - name: Restore database backup
      shell: "mysql -u root -p{{ db_password }} < /path/to/backup.sql"
```

16. **Server Hardening**: Automate the implementation of security best practices to harden server configurations. For example, disabling unused services:

```
- name: Disable unused services
  hosts: all
  tasks:
    - name: Disable Telnet
      service:
        name: telnet
        state: stopped
        enabled: no
```

17. **Compliance Reporting**: Generate compliance reports to ensure that servers adhere to security policies and standards. For example, generating a CIS benchmark report:

```
- name: Generate CIS benchmark report
  hosts: all
  tasks:
    - name: Run CIS benchmark
      command: "cis-security-benchmark --level 1 --json-output
/tmp/cis_report.json"
```

18. **Log Management**: Automate log collection and analysis to monitor system health and troubleshoot issues. For example, configuring syslog forwarding:

```
- name: Configure syslog forwarding
  hosts: all
  tasks:
    - name: Configure syslog
      template:
        src: syslog-ng.conf.j2
        dest: /etc/syslog-ng/syslog-ng.conf
      notify:
        - Restart syslog-ng
  handlers:
    - name: Restart syslog-ng
      service:
        name: syslog-ng
        state: restarted
```

19. **Container Orchestration**: Use Ansible to automate the deployment and management of containers using orchestration tools like Kubernetes. For example, deploying a Kubernetes pod:

```
- name: Deploy Kubernetes pod
  hosts: localhost
  tasks:
    - name: Apply pod configuration
      k8s:
        state: present
        definition: pod.yaml
```

20. **Self-Service Provisioning**: Empower users to provision their own resources within predefined limits using Ansible Tower. For example, allowing developers to provision development environments:

```
- name: Provision development environment
  hosts: localhost
  tasks:
    - name: Launch VM
      vmware_guest:
        hostname: vcenter.example.com
        username: admin
        password: secret
        validate_certs: no
        name: dev-vm
        template: CentOS
        datacenter: DC1
        folder: /vm
        state: poweredon
```