DevOps Shack

50 Ansible Real-Time Use Cases

1. **Provisioning Servers**: Ansible can be used to automate the provisioning of servers. This includes tasks such as creating instances, configuring network settings, and installing necessary software.

```
---
- name: Provision Servers
  hosts: all
  tasks:
    - name: Create instances
      shell: |
        # Your command to create instances goes here

    - name: Configure network settings
      shell: |
        # Your command to configure network settings goes here

    - name: Install necessary software
      apt:
        name: "{{ item }}"
        state: present
      with_items:
        - package1
        - package2
```

2. **Configuration Management**: Ansible can manage configurations across multiple servers, ensuring consistency and compliance with organizational standards.

```
---
- name: Configure Servers
  hosts: web_servers
  tasks:
    - name: Copy configuration file
      copy:
        src: /path/to/local/config.conf
        dest: /etc/config.conf
      notify: restart_service
  handlers:
    - name: restart_service
      service:
        name: web_service
        state: restarted
```

3. **Continuous Deployment**: Ansible can automate the deployment of applications to various environments, ensuring quick and consistent releases.

```
---
- name: Deploy Application
  hosts: production
  tasks:
    - name: Fetch latest code
      git:
        repo: https://github.com/your/repository.git
        dest: /var/www/app
        version: master

    - name: Install dependencies
      command: /var/www/app/install.sh

    - name: Restart application
      service:
        name: your_app_service
        state: restarted
```

4. **Configuration Backup**: Ansible can automate the backup of configuration files across servers to ensure data integrity and disaster recovery.

```
---
- name: Backup Configuration
  hosts: all
  tasks:
    - name: Backup configuration files
      command: cp /etc/config.conf /etc/config.conf.backup
```

5. **User Management**: Ansible can automate user provisioning, managing SSH keys, and setting up user permissions across multiple servers.

```
---
- name: Manage Users
  hosts: all
  tasks:
    - name: Create user
      user:
        name: johndoe
        password: encrypted_password
        state: present

    - name: Add SSH key
      authorized_key:
        user: johndoe
        key: "{{ lookup('file', '/path/to/public_key.pub') }}"
        state: present

    - name: Set permissions
      file:
        path: /home/johndoe
        owner: johndoe
        group: johndoe
        mode: 0755
```

6. **Patch Management**: Ansible can automate the patching of software across servers, ensuring they are up-to-date and secure.

```
---
- name: Patch Management
  hosts: all
  tasks:
    - name: Update packages
      yum:
        name: '*'
        state: latest
```

7. **Monitoring Configuration**: Ansible can automate the configuration of monitoring agents and services across servers for centralized monitoring.

```
---
- name: Configure Monitoring
  hosts: all
  tasks:
    - name: Install monitoring agent
      yum:
        name: monitoring-agent
        state: present

    - name: Configure agent
      template:
        src: monitoring_agent.conf.j2
        dest: /etc/monitoring/agent.conf
```

8. **Database Setup**: Ansible can automate the setup and configuration of databases, including user management and schema creation.

```
---
- name: Setup Database
  hosts: db_servers
  tasks:
    - name: Install database server
      yum:
        name: mysql-server
        state: present

    - name: Start database service
      service:
        name: mysql
        state: started
```

9. **Load Balancer Configuration**: Ansible can automate the configuration of load balancers, ensuring traffic is evenly distributed across servers.

```yaml
---
- name: Configure Load Balancer
  hosts: lb_servers
  tasks:
    - name: Install load balancer software
      yum:
        name: nginx
        state: present

    - name: Copy load balancer configuration
      template:
        src: nginx.conf.j2
        dest: /etc/nginx/nginx.conf

    - name: Restart load balancer service
      service:
        name: nginx
        state: restarted
```

10. **Firewall Rules**: Ansible can automate the configuration of firewall rules, ensuring security policies are consistently applied.

```yaml
---
- name: Configure Firewall
  hosts: all
  tasks:
    - name: Allow SSH traffic
      firewalld:
        service: ssh
        state: enabled
        permanent: true
```

11. **Log Rotation**: Ansible can automate the configuration of log rotation settings to manage log files efficiently and prevent disk space issues.

```yaml
---
- name: Configure Log Rotation
  hosts: all
  tasks:
    - name: Copy logrotate configuration
      template:
        src: logrotate.conf.j2
        dest: /etc/logrotate.conf
```

12. **SSL Certificate Management**: Ansible can automate the renewal and deployment of SSL certificates across servers to ensure secure communication.

```
---
- name: Manage SSL Certificates
  hosts: web_servers
  tasks:
    - name: Renew SSL certificate
      shell: certbot renew
```

13. **Service Discovery**: Ansible can automate the registration and discovery of services in a dynamic environment using tools like Consul or etcd.

```
---
- name: Register Service
  hosts: app_servers
  tasks:
    - name: Register service with Consul
      shell: consul register service.json
```

14. **High Availability Setup**: Ansible can automate the setup of high availability configurations such as clustering and failover mechanisms.

```
---
- name: Setup High Availability
  hosts: db_servers
  tasks:
    - name: Install clustering software
      yum:
        name: pacemaker
        state: present

    - name: Configure cluster
      template:
        src: cluster.conf.j2
        dest: /etc/cluster.conf
```

15. **DNS Configuration**: Ansible can automate the configuration of DNS records across servers to manage domain resolution.

```
---
- name: Configure DNS
  hosts: dns_servers
  tasks:
    - name: Add DNS record
      shell: |
        # Your command to add DNS record goes here
```

16. **Container Orchestration**: Ansible can automate the deployment and management of containers using tools like Docker Swarm or Kubernetes.

```
---
- name: Deploy Containers
  hosts: docker_swarm
  tasks:
    - name: Deploy container
      docker_container:
        name: web_app
        image: your_image
        state: started
```

17. **Backup and Restore**: Ansible can automate the backup and restoration of data across servers to ensure data integrity and disaster recovery.

```
---
- name: Backup Data
  hosts: all
  tasks:
    - name: Backup data
      shell: |
        # Your command to backup data goes here
```

18. **Secrets Management**: Ansible can automate the retrieval and distribution of secrets securely using tools like HashiCorp Vault or Ansible Vault.

```
---
- name: Retrieve Secrets
  hosts: all
  tasks:
    - name: Fetch secret from Vault
      shell: vault read secret/password
```

19. **Performance Monitoring**: Ansible can automate the setup and configuration of performance monitoring tools to track system performance metrics.

```
---
- name: Setup Performance Monitoring
  hosts: all
  tasks:
    - name: Install monitoring agent
      yum:
        name: monitoring-agent
        state: present

    - name: Configure monitoring agent
      template:
        src: monitoring_agent.conf.j2
        dest: /etc/monitoring/agent.conf
```

20. **Compliance Checking**: Ansible can automate compliance checks against predefined security policies to ensure systems meet regulatory requirements.

```yaml
---
- name: Check Compliance
  hosts: all
  tasks:
    - name: Run compliance check
      shell: |
        # Your command to run compliance check goes here
```

21. **Configuration Templating**: Ansible can use Jinja2 templates to dynamically generate configuration files based on variables, allowing for easy customization.

```yaml
---
- name: Generate Configuration
  hosts: all
  tasks:
    - name: Generate config file
      template:
        src: config_template.j2
        dest: /etc/config.conf
      vars:
        variable1: value1
        variable2: value2
```

22. **Automated Testing**: Ansible can automate the testing of infrastructure changes to ensure they meet functional and performance requirements.

```yaml
---
- name: Run Automated Tests
  hosts: test_servers
  tasks:
    - name: Run test suite
      shell: |
        # Your command to run automated tests goes here
```

23. **Capacity Planning**: Ansible can collect system metrics and analyze them to aid in capacity planning and resource allocation.

```yaml
---
- name: Gather System Metrics
  hosts: all
  tasks:
    - name: Collect system metrics
      shell: |
        # Your command to collect system metrics goes here
```

24. **Disaster Recovery**: Ansible can automate the setup and configuration of disaster recovery environments to minimize downtime in case of failures.

```
---
- name: Setup Disaster Recovery
  hosts: dr_servers
  tasks:
    - name: Restore backup
      shell: |
        # Your command to restore backup goes here
```

25. **Workflow Automation**: Ansible can orchestrate complex workflows involving multiple tasks and dependencies across different systems.

```
---
- name: Orchestrate Workflow
  hosts: all
  tasks:
    - name: Task 1
      shell: |
        # Task 1 command goes here

    - name: Task 2
      shell: |
        # Task 2 command goes here
```

26. **Security Hardening**: Ansible can automate security hardening tasks such as disabling unused services, applying security patches, and enforcing security policies.

```
---
- name: Harden Security
  hosts: all
  tasks:
    - name: Disable unused services
      service:
        name: "{{ item }}"
        state: stopped
        enabled: no
      with_items:
        - telnet
        - ftp
        - rsh
```

27. **Custom Application Deployment**: Ansible can deploy custom applications by fetching artifacts from repositories, configuring them, and starting services.

```
---
- name: Deploy Custom Application
  hosts: app_servers
  tasks:
    - name: Fetch application artifact
      get_url:
        url: http://your_repository.com/your_app.zip
        dest: /tmp/your_app.zip

    - name: Unzip application
      unarchive:
        src: /tmp/your_app.zip
        dest: /opt/your_app

    - name: Start application service
      service:
        name: your_app_service
        state: started
```

28. **Centralized Logging**: Ansible can automate the configuration of centralized logging solutions like ELK (Elasticsearch, Logstash, Kibana) stack.

```
---
- name: Configure Centralized Logging
  hosts: log_servers
  tasks:
    - name: Install ELK stack
      shell: |
        # Your command to install ELK stack goes here
```

29. **Identity and Access Management (IAM)**: Ansible can automate the management of user accounts, groups, and permissions across multiple systems.

```
---
- name: Manage Users and Groups
  hosts: all
  tasks:
    - name: Create user
      user:
        name: johndoe
        state: present

    - name: Add user to group
      user:
        name: johndoe
        groups: admins
```

30. **Docker Image Build and Push**: Ansible can automate the building and pushing of Docker images to a registry.

```
---
- name: Build and Push Docker Image
  hosts: docker_build_server
  tasks:
    - name: Build Docker image
      docker_image:
        name: your_image
        path: /path/to/dockerfile_directory

    - name: Push Docker image to registry
      docker_image:
        name: your_image
        push: yes
        registry: your_registry
```

31. **Cross-Platform Management**: Ansible can manage heterogeneous environments with different operating systems, allowing for consistent automation across platforms.

```
---
- name: Cross-Platform Management
  hosts: all
  tasks:
    - name: Install package on CentOS
      yum:
        name: package_name
        state: present
      when: ansible_distribution == 'CentOS'

    - name: Install package on Ubuntu
      apt:
        name: package_name
        state: present
      when: ansible_distribution == 'Ubuntu'
```

32. **Version Control Integration**: Ansible playbooks can be integrated with version control systems like Git for collaboration, versioning, and change tracking.

```
---
- name: Git Integration
  hosts: all
  tasks:
    - name: Clone playbook repository
      git:
        repo: git://your_repository_url.git
        dest: /etc/ansible/playbooks
```

33. **Cloud Infrastructure Provisioning**: Ansible can automate the provisioning of cloud infrastructure resources using modules provided by cloud providers like AWS, Azure, and GCP.

```
---
- name: Provision EC2 Instances
  hosts: localhost
  tasks:
    - name: Launch EC2 instances
      ec2_instance:
        key_name: your_key
        instance_type: t2.micro
        image: ami-12345678
        count: 3
```

34. **Configuration Drift Detection**: Ansible can detect configuration drift across servers by comparing the desired state defined in playbooks with the actual state of the systems.

```
---
- name: Check Configuration Drift
  hosts: all
  tasks:
    - name: Compare configuration
      command: diff -u /etc/config.conf /etc/config.conf.bak
```

35. **Self-Service Infrastructure**: Ansible can empower users to provision and manage their own infrastructure resources through predefined automation workflows.

```
---
- name: Self-Service Provisioning
  hosts: localhost
  tasks:
    - name: Run playbook to provision resources
      include_role:
        name: provision_resources
```

36. **Cost Optimization**: Ansible can optimize cloud resource usage by automating scheduled start/stop operations for non-production environments.

```
---
- name: Schedule Start/Stop Instances
  hosts: localhost
  tasks:
    - name: Start instances during office hours
      ec2_instance:
        instance_ids: "{{ ec2_instances_to_start }}"
        state: started
      when: "ansible_date_time.hour|int >= 9 and ansible_date_time.hour|int
< 18"
```

37. **Dynamic Inventory Management**: Ansible can dynamically generate inventory based on external data sources like cloud providers, CMDBs, or custom scripts.

```
---
- name: Dynamic Inventory
  hosts: all
  tasks:
    - name: Use dynamic inventory script
      debug:
        msg: "Server {{ inventory_hostname }} has IP {{
hostvars[inventory_hostname]['ansible_host'] }}"
```

38. **Configuration Compliance Reporting**: Ansible can generate reports on configuration compliance against predefined policies to ensure adherence to standards.

```
---
- name: Generate Compliance Report
  hosts: all
  tasks:
    - name: Run compliance check
      command: compliance-check.sh
    - name: Email report
      mail:
        to: admin@example.com
        subject: Ansible Compliance Report
        body: "Please find attached the compliance report."
        attach: /path/to/compliance_report.txt
```

39. **Multi-Tier Application Deployment**: Ansible can deploy multi-tier applications consisting of web servers, application servers, and databases with coordinated configuration.

```
---
- name: Deploy Multi-Tier Application
  hosts: all
  tasks:
    - name: Deploy web servers
      include_role:
        name: web_server_role
    - name: Deploy app servers
      include_role:
        name: app_server_role
    - name: Deploy database servers
      include_role:
        name: db_server_role
```

40. **Automated Rollback**: Ansible can automate rollback procedures in case of deployment failures or issues, ensuring service availability.

```
---
- name: Automated Rollback
  hosts: all
  tasks:
    - name: Rollback application
      shell: rollback.sh
```

41. **Dynamic Scaling**: Ansible can automate the scaling of resources based on demand, ensuring optimal performance and cost efficiency.

```
---
- name: Dynamic Scaling
  hosts: scale_servers
  tasks:
    - name: Scale up servers
      shell: |
        # Your command to increase server capacity goes here
```

42. **Service Discovery and Registration**: Ansible can automate service discovery and registration with tools like Consul or etcd to enable dynamic service communication.

```
---
- name: Service Discovery
  hosts: app_servers
  tasks:
    - name: Register service with Consul
      shell: consul register service.json
```

43. **Immutable Infrastructure**: Ansible can enforce immutability by rebuilding infrastructure from scratch for each deployment, enhancing reliability and reproducibility.

```
---
- name: Immutable Infrastructure
  hosts: all
  tasks:
    - name: Deploy new version
      include_role:
        name: deploy_new_version
```

44. **Continuous Compliance Enforcement**: Ansible can continuously enforce compliance policies by automatically remediating non-compliant configurations.

```
---
- name: Continuous Compliance
  hosts: all
  tasks:
    - name: Remediate compliance violations
      include_role:
        name: remediate_compliance_violations
```

45. **Cloud Migration Automation**: Ansible can automate the migration of workloads between cloud providers or from on-premises to the cloud.

```
---
- name: Cloud Migration
  hosts: all
  tasks:
    - name: Migrate workload to AWS
      include_role:
        name: migrate_to_aws
```

46. **Disaster Recovery Testing**: Ansible can automate the testing of disaster recovery plans to ensure systems can be restored in case of failures.

```
---
- name: Disaster Recovery Testing
  hosts: dr_test_servers
  tasks:
    - name: Restore backup
      shell: |
        # Your command to restore backup for testing goes here
```

47. **Cross-Region Deployment**: Ansible can automate deployments across multiple regions to achieve high availability and fault tolerance.

```
---
- name: Cross-Region Deployment
  hosts: all
  tasks:
    - name: Deploy application to multiple regions
      include_role:
        name: deploy_to_multiple_regions
```

48. **Event-Driven Automation**: Ansible can respond to events and triggers from external systems to automate actions or workflows.

```
---
- name: Event-Driven Automation
  hosts: event_driven_servers
  tasks:
    - name: Respond to event trigger
      include_role:
        name: respond_to_event_trigger
```

49. **Infrastructure Testing**: Ansible can automate infrastructure testing by validating configurations and ensuring desired states are maintained.

```
---
- name: Infrastructure Testing
  hosts: test_servers
  tasks:
    - name: Run infrastructure tests
      shell: |
        # Your command to run infrastructure tests goes here
```

50. **Self-Healing Systems**: Ansible can automate the detection and remediation of issues to maintain system health and availability.

```
---
- name: Self-Healing Systems
  hosts: all
  tasks:
    - name: Detect and remediate issues
      include_role:
        name: detect_and_remediate_issues
```