PW SKILLS | DevOps and Cloud Computing

Advanced Ansible Usage

Objective

- Optimize Ansible task execution using parallelism and host-specific targeting.
- Use Ansible to provision and manage resources on AWS, GCP, and Azure.
- Automate monitoring, resource configuration, and backup processes using Ansible.
- Orchestrate complex deployments and integrate Ansible playbooks in CI/CD pipelines.

PW SKILLS

Explaining how Ansible handles task execution in parallel and its impact on performance

Let’s see

Ansible handles task execution in parallel by leveraging features such as forks, async, and polling. Here's a concise explanation:

1. Parallel Execution Across Hosts
2. Asynchronous Tasks
3. Serial Execution

Impact on Performance

- Improved Speed
- Resource Optimization
- Scalability

PW SKILLS

Introducing forks and how adjusting the concurrency level improves speed

Forks

In Ansible, forks determine the number of parallel connections the control node can manage simultaneously during task execution.

- By default, the fork value is set to 5, meaning tasks are executed on up to 5 hosts concurrently. Adjusting this value can significantly improve speed by increasing concurrency.
- For example, setting forks=20 allows tasks to run on 20 hosts at the same time, reducing the overall runtime for large inventories.

However, higher fork values also increase CPU and memory usage on the control host. Balancing the concurrency level based on system capacity ensures optimal performance without overloading resources.

PW SKILLS

Demonstrating limiting task execution to specific hosts for optimization

Let’s do it

To limit task execution to specific hosts for optimization in Ansible, you can use the --limit option or control execution within playbooks. Here are methods:

Using ‘—limit’ Option
The ‘—limit’ flag restricts playbook execution to specific hosts or groups:

- Run on a single host:

```
ansible-playbook playbook.yml --limit "host1"
```

- Run on multiple hosts:

```
ansible-playbook playbook.yml --limit "host1,host2"
```

- Run on a group:

```
ansible-playbook playbook.yml --limit "webservers"
```

Using ‘serial’ for Controlled Execution

The ‘serial’ keyword limits the number of hosts processed at a time:

```
- hosts: all
  serial: 5
  tasks:
    - name: Example task
      ping:
```

This executes tasks on 5 hosts at a time, optimizing resource usage.

Delegating Tasks to Specific Hosts
For individual tasks, use ‘delegate-to’ to target specific hosts:

```
tasks:
  - name: Perform task on localhost
    command: echo "Task executed"
    delegate_to: localhost
```

These methods ensure precise and efficient execution tailored to your infrastructure needs.

PW SKILLS

Discussing best practices for parallel execution in large-scale environments

Let’s discuss

- Optimize Forks Configuration
- Use Asynchronous Tasks
- Leverage Serial Execution
- Delegate Tasks Strategically
- Implement Custom Strategies
- Monitor Resource Usage

PW SKILLS

How Ansible interacts with cloud providers to automate infrastructure provisioning.

Let’s discuss

Ansible interacts with cloud providers to automate infrastructure provisioning by leveraging its agentless architecture, specialized modules, and Infrastructure-as-Code (IaC) principles. Here's a concise overview:

Key Mechanisms of Interaction:

- Agentless Design
- Modules for Cloud Integration
- Dynamic Inventory Management
- Infrastructure as Code (IaC)

Let’s discuss

Integration with Cloud Providers

Ansible can directly interact with cloud APIs to provision resources or work alongside tools like Terraform for initial resource deployment. For instance:

- Terraform deploys infrastructure.
- Ansible provisions and configures the deployed resources using playbooks.

PW SKILLS

Introducing modules for AWS, GCP, and Azure, and explain their functionalities

AWS, GCP & Azure

Ansible provides specialized modules for AWS, GCP, and Azure to automate infrastructure provisioning. Below is an introduction to these modules and their functionalities:

AWS Modules

Ansible includes a wide range of modules for AWS services, such as:

- EC2
- S3
- Cloud Formation
- IAM
- RDS
- VPC

AWS, GCP & Azure

GCP Modules

Ansible's GCP modules follow a naming convention like gcp *:

- gcp compute instance
- gcp compute network
- gcp compute disk
- gcp compute address

AWS, GCP & Azure

Azure Modules

Ansible supports Azure through modules such as:

- azure rm virtualmachine
- azure rm networkinterface
- azure rm storageaccount
- azure rm resourcegroup

PW SKILLS

Explaining how to provision and manage instances, load balancers, and networking components

Let’s see

Provisioning Instances

- **Modules:** Ansible provides cloud-specific modules like ec2 for AWS, ‘gcp compute instance’ for GCP, and ‘azure rm virtualmachine’ for Azure to create and manage virtual machines.
- **Functionality:** These modules allow users to define instance configurations (e.g., type, image ID, region) in playbooks, automating tasks such as launching VMs, assigning IPs, and configuring SSH access

Let's see

Managing Load Balancers

- **Modules:** Ansible supports load balancer modules such as 'ec2 elb' for AWS Elastic Load Balancers, 'gcp compute target pool' for GCP, and 'azure rm loadbalancer' for Azure.
- **Functionality:** These modules automate the creation and configuration of load balancers, enabling traffic distribution across instances while ensuring high availability.

Let’s see

Configuring Networking Components

- **Modules:** Networking modules like ‘ec2 vpc net’ (AWS), ‘gcp compute network’ (GCP), and ‘azure rm networkinterface’ (Azure) are used to set up networks, subnets, security groups, and routing tables.
- **Functionality:** Ansible automates network setup by defining CIDR blocks, firewall rules, and connectivity requirements in YAML playbooks.

Pop Quiz

Q. What is the primary purpose of a load balancer in cloud environments?

A

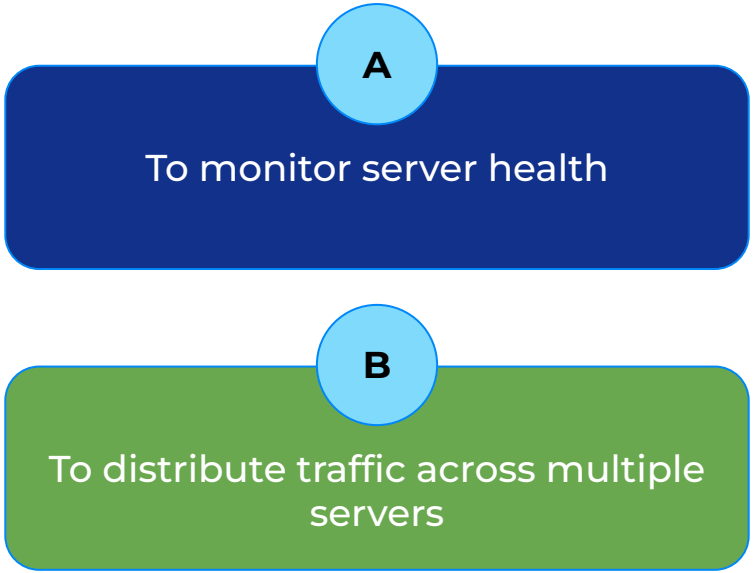
To monitor server health

B

To distribute traffic across multiple servers

Pop Quiz

Q. What is the primary purpose of a load balancer in cloud environments?



PW SKILLS

Highlighting key challenges in cloud automation and how Ansible helps overcome them.

Let’s do it

Key Challenges in Cloud Automation:

- Internet Connectivity Issues
- Security and Compliance Limitations
- Platform Lock-In
- Complexity in Multi-Cloud Management
- Technical Debt

Let’s do it

How Ansible Helps Overcome These Challenges:

- Agentless Architecture
- Security Automation
- Multi-Cloud Support
- Unified Automation
- Reducing Technical Debt

PW SKILLS

Explaining how Ansible ensures consistency across multiple hosts

Let’s do it

Ansible ensures consistency across multiple hosts through the following mechanisms:

1. Playbooks
2. Inventory Management
3. Parallel Execution
4. Task Execution Strategies
5. Patterns and Targeting

PW SKILLS

Discussing best practices for managing software installations, system configurations, and services

Let’s discuss

Software Installations:

- Use modules like dnf, apt, or win chocolatey to automate package installations and updates across hosts.
- Keep playbooks idempotent to ensure repeated executions do not cause unintended changes.
- Define software dependencies clearly in roles for modularity and reusability.

Let’s discuss

System Configurations:

- Leverage Ansible's YAML-based playbooks to maintain consistent configurations across systems.
- Use dynamic inventories to adapt to changes in infrastructure automatically.
- Implement version control for configuration files using Git to track changes effectively.

Let’s discuss

Managing Services:

- Use the service module to start, stop, enable, or disable services on managed nodes.
- Monitor service states regularly and automate recovery processes in case of failures.
- Combine provisioning and service management into a single workflow for efficiency.

PW SKILLS

Demonstrating the importance of idempotency in large-scale automation

Let’s do it

Idempotency is critical in large-scale automation as it ensures consistent, reliable, and predictable outcomes, even when operations are repeated due to failures or retries. Here’s why it matters:

1. Consistency Across Systems
2. Reliable Error Handling
3. Scalability and Resilience
4. Simplified System Design

Take A 5-Minute Break!

- Stretch and relax
- Hydrate
- Clear your mind
- Be back in 5 minutes

PW SKILLS

Explaining how Ansible automates monitoring deployments for system observability

Let’s see

Ansible automates monitoring deployments for system observability by integrating with monitoring tools and using its playbooks to deploy, configure, and manage monitoring agents and systems. Here’s how it works:

1. Agent Deployment
2. Metrics Collection
3. Alerting Configuration
4. Dashboard Setup
5. Log Management

PW SKILLS

Discussing deploying monitoring agents and configuring resource utilization tracking

Let’s discuss

Deploying Monitoring Agents

Ansible automates the deployment of monitoring agents by using predefined roles and playbooks to install and configure agents across multiple hosts. Examples include:

- Datadog
- Google Cloud Ops Agent
- New Relic Java Agent

Let’s discuss

Configuring Resource Utilization Tracking

Ansible playbooks define custom configurations for monitoring tools to collect resource utilization metrics, such as CPU, memory, and network usage. For example:

- Prometheus
- Netdata

PW SKILLS

Introducing automated backup and restore strategies using Ansible Tower or scheduled tasks.

Let's see

1. **Using Ansible Tower:**
    - Built-in Backup and Restore
    - Disaster Recovery
    - Scheduled Backups
2. **Using Scheduled Tasks with Playbooks:**
    - Synchronize Module
    - Version Control Systems
    - Custom Scheduling

PW SKILLS

Discussing real-world scenarios where automated monitoring improves operational efficiency

Let’s discuss

1. **Predictive Maintenance in Manufacturing:**
    - Companies like Siemens and General Electric use automated monitoring with IoT sensors and AI to predict machine failures before they occur. This reduces unplanned downtime, optimizes energy usage, and extends equipment lifespan by enabling timely repairs.
2. **Energy Efficiency Management:**
    - Monitoring systems adjust machinery parameters (e.g., speed, pressure) based on real-time data to optimize energy consumption. This reduces waste and lowers operational costs while minimizing environmental impact.

Let’s discuss

3. **Workflow Optimization:**
    - AI-driven monitoring identifies bottlenecks in workflows and suggests improvements. For example, process mining tools help reduce lead times and automate repetitive tasks, saving time and costs in manufacturing operations.
4. **Production Line Optimization:**
    - Automated monitoring systems, such as Yellowfin Signals, help detect anomalies in real-time on production lines. For example, a turbine blade manufacturer used these tools to identify incompatible items in workflows, preventing costly errors and improving production efficiency.

PW SKILLS

Multi-tier application deployment and how Ansible manages dependencies between services

Let’s see

Concept of Multi-Tier Application Deployment

Multi-tier applications separate functionalities into distinct layers or tiers, typically including:

- Presentation Tier
- Logic Tier
- Data Tier

This architecture enhances scalability, maintainability, and fault isolation by allowing independent development and scaling of each tier.

Let’s see

How Ansible Manages Dependencies Between Services:

1. Role-Based Structure
2. Handlers and Notifications
3. Inventory Grouping
4. Orchestration

Pop Quiz

Q. How does Ansible manage dependencies between services in multi-tier applications?

A

By using handlers and notifications to coordinate tasks

B

By manually restarting services after deployment

Pop Quiz

Q. How does Ansible manage dependencies between services in multi-tier applications?

A

By using handlers and notifications to coordinate tasks

B

By manually restarting services after deployment

PW SKILLS

How database and application servers must be deployed in a coordinated manner

Let’s see

Deploying database and application servers in a coordinated manner is essential to ensure synchronization, reliability, and optimal performance. Here’s how this coordination works:

1. Synchronization of Versions
2. Dependency Management
3. Environment-Specific Configurations
4. Minimizing Latency
5. Automated Rollbacks

PW SKILLS

Introducing best practices for integrating Ansible playbooks into CI/CD pipelines

Best practices

1. Treat Playbooks as Code
2. Modular Design
3. Automated Testing
4. Environment Consistency
5. Pipeline Integration
6. Use Dry-Run Mode

PW SKILLS

Benefits of using automation for continuous deployment in modern DevOps workflows

Benefits

1. Faster Release Cycles
2. Reduced Human Errors
3. Improved Reliability
4. Enhanced Efficiency
5. Continuous Feedback Loop
6. Scalability

PW SKILLS

Real-world scenario where Ansible automates infrastructure provisioning and application deployment via GitHub Actions

Let’s do it

1. **Infrastructure Provisioning:**

- A GitHub repository contains Terraform code to provision an EC2 instance in AWS.
- A GitHub Actions workflow is triggered upon a code push or pull request to the repository.
- The workflow runs Terraform to create the EC2 instance, ensuring it is accessible via SSH for further configuration tasks.

2. **Application Deployment:**

- After provisioning, a second job in the GitHub Actions workflow installs Ansible and runs a playbook to configure the EC2 instance.
- The playbook sets up the required software stack (e.g., web server) and deploys a test application to the instance using its public IP or DNS.

Let’s do it

3. **Workflow Integration:**

- The workflow includes Ansible linting to check playbooks for errors before execution.
- Dependencies between jobs ensure Terraform completes before Ansible begins, maintaining coordination.

PW SKILLS

Discussing challenges and solutions in integrating Ansible with Git-based workflows.

Challenges

1. **Authentication Issues:**
   Problems arise when accessing private repositories due to improper SSH key setup or incorrect repository URLs, causing tasks to hang or fail.

2. **Conflict Management:**
   Git-based workflows may encounter merge conflicts, which require manual intervention as Ansible lacks native support for automated conflict resolution.

3. **Limited Git Functionality:**
   The git module in Ansible supports cloning and pulling but does not handle operations like adding, committing, or pushing changes.

4. **Workflow Coordination:**
   Ensuring smooth integration between GitOps and Ansible workflows can be challenging, especially when managing containerized applications or complex CI/CD pipelines.

Solutions

1. **Proper SSH Key Configuration:**
   Generate SSH keys on the control machine and add the public key to the Git repository for seamless access to private repositories.
2. **Use Shell Module for Advanced Git Tasks:**
   For operations like git push, use Ansible's shell module to execute Git commands directly.
3. **Leverage Webhooks:**
   Combine Git webhooks with Ansible workflows to trigger automation tasks upon repository changes, ensuring real-time updates and synchronization.
4. **Testing and Validation:**
   Use tools like Molecule to test playbooks before execution, reducing errors in workflows involving Git repositories.

PW SKILLS

Demonstrating how automation improves scalability, repeatability, and reliability

Let’s do it

1. **Scalability:**

   Automation dynamically adjusts resources based on demand, such as scaling up during high traffic (e.g., holiday sales) and scaling down during low usage periods. This prevents resource wastage and ensures systems can handle fluctuating workloads efficiently.

2. **Repeatability:**

   Automated workflows execute tasks consistently across environments, eliminating human error and ensuring uniform results. For instance, deploying identical configurations to multiple servers using Ansible ensures predictable outcomes every time.

Let’s do it

3. **Reliability:**

   Automation reduces downtime by proactively monitoring systems, triggering alerts, and performing auto-remediation when issues arise. It ensures stable operations by handling repetitive tasks accurately and responding to failures swiftly.

PW SKILLS

Time for case study!

Important

- Complete the post-class assessment
- Complete assignments (if any)
- Practice the concepts and techniques taught in this sessio
- Review your lecture notes
- Note down questions and queries regarding this session ar
  consult the teaching assistants

Thanks !