

Configuration Management



Objective

- Understand the necessity of configuration management (CM) and its role in infrastructure automation.
- Learn the benefits of CM, including consistency, scalability, and disaster recovery.
- Compare popular CM tools (Chef, Puppet, Ansible) and their use cases.
- Understand why Ansible is widely preferred for configuration management.





Explaining the traditional manual infrastructure setup process and its limitations

Let's See

The traditional manual infrastructure setup process involves several steps:

1. Provisioning: This includes acquiring and setting up servers, network equipment, and other infrastructure components. It involves installing operating systems, configuring IP addresses, hostnames, firewalls, and DNS settings, and connecting these components to a network.
2. Configuration: After provisioning, the infrastructure is customized by installing dependencies, updating software, setting up logging, and configuring databases.



Let's See

The traditional manual approach to infrastructure setup has several limitations:

- Time-Consuming and Error-Prone
- Limited Scalability
- Inconsistent Environments
- High Maintenance Costs
- Security Risks





Discussing automation in CM and its role in DevOps workflows

Let's Discuss

Automation in Configuration Management

Automation in configuration management involves using tools and scripts to manage and maintain consistent configurations across different environments. This includes:

- Infrastructure as Code (IaC)
- Configuration Automation Tools
- Version Control Systems



Let's Discuss

In DevOps workflows, automation in configuration management plays a crucial role by:

- Streamlining Infrastructure Management
- Enhancing Agility and Efficiency
- Improving Reliability and Scalability
- Integrating with CI/CD Pipelines





**Providing examples of
configuration drift and how
CM tools prevent it**

Example of configuration drift

Examples of Configuration Drift

1. Manual Changes
2. Software Updates
3. Temporary Fixes

How CM Tools Prevent Configuration Drift

1. Infrastructure as Code (IaC)
2. Configuration Management Tools
3. Monitoring and Alerts
4. Version Control





Explaining the major advantages of using CM

Advantages

1. **Consistency:** CM ensures uniform environments across different servers by maintaining consistent configurations, reducing errors and inconsistencies that can arise from manual setup.
2. **Scalability:** It allows for easy replication of configurations across multiple servers, making it simpler to scale infrastructure without manually configuring each server.
3. **Disaster Recovery:** CM facilitates quick system restoration from backups and predefined configurations, ensuring business continuity by minimizing downtime during failures.



**Demonstrating an
example where
automation saves time**

Let's do it

Example: Automating Server Deployment with Ansible

Scenario

A company needs to deploy 20 new web servers with identical configurations:

- Operating System: Ubuntu 22.04
- Web Server: Apache
- Database: MySQL
- Security: Specific firewall rules

Manual Process

Manually deploying these servers would involve:

- Time-Consuming
- Error-Prone

Let's do it

Automated Process with Ansible

1. Create a Playbook: Define a single Ansible playbook that outlines the desired configuration for all servers.

```
---
- name: Deploy Web Server
  hosts: webservers
  become: yes

  tasks:
    - name: Install Apache
      apt:
        name: apache2
        state: present

    - name: Install MySQL
      apt:
        name: mysql-server
        state: present

    - name: Configure Firewall
      ufw:
        rule: allow
        port: 80
```

Let's do it

2. Run the Playbook: Execute the playbook across all 20 servers simultaneously using Ansible's command-line interface.

```
ansible-playbook -i inventory deploy_web_server.yml
```

3. Monitor and Verify: Use monitoring tools to verify that all servers are correctly configured and functioning as expected.

Take A 5-Minute Break!



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





Introducing the three major CM Tools

CM Tools

1. Chef

- Language: Ruby-based
- Architecture: Agent-based, requiring a dedicated Chef server
- Key Features: Declarative configuration, extensive resource management, and compliance automation. Chef adapts to infrastructure needs and provides scalability and consistency across environments.

CM Tools

2. Puppet

- Language: Declarative language
- Architecture: Agent-master model, with Puppet agents communicating with a central Puppet server.
- Key Features: Suitable for large enterprise environments, Puppet offers idempotence, resource abstraction, and modularity. It is ideal for managing complex configurations and ensuring compliance.

CM Tools

3. Ansible

- Language: YAML-based
- Architecture: Agentless, simplifying setup and reducing operational overhead
- Key Features: Easy to learn and use, Ansible provides idempotency, extensibility, and support for multiple platforms. It is favored for its simplicity and flexibility in automating IT tasks.

Pop Quiz

Q. Which CM tool is YAML-based, agentless, and simple to set up and use?

A

Ansible

B

Chef

Pop Quiz

Q. Which CM tool is YAML-based, agentless, and simple to set up and use?

A

Ansible

B

Chef



**Comparing features,
advantages, and use
cases of each tool**

Comparing CM Tools

Comparison of Chef, Puppet, and Ansible

Features:

Tool	Architecture	Language	Ease of Setup	Ease of Use
<u>Chef</u>	Master-Agent	Ruby	Complex	Intermediate
<u>Puppet</u>	Master-Agent	Ruby, Puppet DSL	Complex	Experienced
<u>Ansible</u>	Agentless	Python, YAML	Easy	Simple

Comparing CM Tools

Advantages and Use Cases

Chef:

- Advantages: Robust for complex configurations, good for large enterprises.
- Use Cases: Ideal for environments requiring extensive customization and control.

Puppet:

- Advantages: Suitable for large-scale deployments, declarative language simplifies management.
- Use Cases: Best for environments where consistency and compliance are critical.

Ansible:

- Advantages: Easy to set up and use, agentless architecture reduces overhead.
- Use Cases: Suitable for smaller teams or environments prioritizing simplicity and rapid deployment.



**Discussing why Ansible
if often preferred**

Let's Discuss

Why Ansible is Often Preferred

- Ansible is frequently chosen over other configuration management tools due to several key advantages:
 1. Agentless Architecture:
 - No Additional Software Required
 - Simplified Deployment and Maintenance
 2. Human-Readable YAML Syntax:
 - Easy to Understand and Write
 - Accessibility
 3. Simple and Lightweight:
 - Fast Learning Curve
 - Efficient Performance



Time for Case Study

Important

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

Thanks



SKILLS

!

