# lesson-06-configuration-management-ecosystem.md

# Configuration Management Ecosystem: Ansible, Puppet, Chef, SaltStack

This lesson introduces the larger configuration management ecosystem used in DevOps. Many organizations use tools such as Puppet, Chef, and SaltStack to configure servers and maintain system state across large deployments. Each tool solves similar problems but approaches them differently. Understanding these tools helps you evaluate which approach fits your environment and how they compare to the Python and Ansible workflows shown in earlier lessons.

---

## 1. What Configuration Management Tools Solve

All configuration management (CM) systems automate:

- Installing software
- Managing configuration files
- Creating required directories and permissions
- Enabling and restarting services
- Maintaining consistent state across servers
- Preventing configuration drift

Rather than writing scripts or SSH commands, these tools define *desired state* and ensure that the system matches it.

Core goals:

- **Consistency** – each server is configured identically
- **Idempotency** – running the tool repeatedly does not break anything
- **Automation** – no manual logins, edits, or repetitive setup
- **Scalability** – handle tens, hundreds, or thousands of nodes

---

## 2. Quick Comparison Table

| Feature | Ansible | Puppet | Chef | SaltStack |
|---|---|---|---|---|
| Architecture | Agentless | Agent-based | Agent-based | Agent or agentless |
| Language | YAML | DSL in Ruby | DSL in Ruby | YAML + Python |
| Idempotent | Yes | Yes | Yes | Yes |
| Execution Model | Push | Pull | Pull | Push or pull |
| Complexity | Low | Medium | Medium-high | Medium |

| Feature | Ansible | Puppet | Chef | SaltStack |
|---|---|---|---|---|
| Typical Use | Simple automation and deployment | Large-scale server fleets | Complex enterprise automation | Fast, event-driven automation |

## 3. Execution Models

### 3.1 Push Model (Ansible)

A control machine connects to all servers and pushes configuration:

- No agents required
- Uses SSH
- Simple to start
- Good for small/medium environments

```
control-node --> server1
control-node --> server2
control-node --> server3
```

### 3.2 Pull Model (Puppet, Chef)

Each server has an agent that checks in with a central master:

```
server1 --> master
server2 --> master
server3 --> master
```

Advantages:

- Scales well
- Master enforces consistent policies

Disadvantages:

- Requires an agent installation
- Requires a master server

### 3.3 Hybrid (SaltStack)

SaltStack supports:

- Push (similar to Ansible)
- Pull (similar to Puppet)
- Event-driven updates

---

## 4. Example Configurations Across Tools

To compare these tools concretely, the following examples all describe the *same task*:

**Ensure Nginx is installed, a configuration file is deployed, and the service is running.**

### 4.1 With Ansible (YAML)

```yaml
---
- hosts: webservers
  become: yes

  tasks:
    - name: Install nginx
      apt:
        name: nginx
        state: present

    - name: Upload configuration
      copy:
        src: nginx.conf
        dest: /etc/nginx/nginx.conf

    - name: Restart nginx
      systemd:
        name: nginx
        state: restarted
```

### 4.2 With Puppet (DSL in Ruby)

```puppet
package { 'nginx':
  ensure => installed,
}

file { '/etc/nginx/nginx.conf':
  ensure  => file,
  source  => 'puppet:///modules/nginx/nginx.conf',
  require => Package['nginx'],
}

service { 'nginx':
  ensure    => running,
  enable    => true,
  subscribe => File['/etc/nginx/nginx.conf'],
}
```

### 4.3 With Chef (Ruby DSL)

```
package 'nginx'

cookbook_file '/etc/nginx/nginx.conf' do
  source 'nginx.conf'
  notifies :restart, 'service[nginx]'
end

service 'nginx' do
  action [:enable, :start]
end
```

### 4.4 With SaltStack (YAML)

```
nginx:
  pkg.installed: []

/etc/nginx/nginx.conf:
  file.managed:
    - source: salt://nginx/nginx.conf
    - require:
      - pkg: nginx

nginx-service:
  service.running:
    - name: nginx
    - enable: True
    - watch:
      - file: /etc/nginx/nginx.conf
```

---

## 5. When to Choose Each Tool

### 5.1 Ansible

Choose Ansible when you want:

- No agents installed on servers
- Simple configurations
- Easy integration with GitHub Actions
- Ad-hoc command capabilities
- Manage small to medium server fleets

Best for deployments, automation tasks, and cloud orchestration.

### 5.2 Puppet

Choose Puppet when you need:

- Ongoing management of a large fleet

- Strict, enforced consistency
- A central authority (Puppet Master)
- Strong policy enforcement

Enterprises often use Puppet for wide-scale server configuration.

### 5.3 Chef

Choose Chef when:

- You prefer a full programming language (Ruby) for logic
- You need complex orchestration
- You operate in an environment with Chef expertise

Best for complex interdependent configuration logic.

### 5.4 SaltStack

Choose SaltStack when:

- You need very fast remote execution
- You prefer event-driven infrastructure
- You want both push and pull models

SaltStack excels in automation environments requiring speed and flexibility.

---

## 6. Where These Tools Fit in CI/CD Pipelines

Configuration management tools typically appear in one of these stages:

### During Deployment (Ansible)

- Code is pushed
- A workflow triggers a playbook
- Application is deployed and configured
- No long-term agent needed

### During Provisioning (Puppet, Chef)

- New servers boot
- Agents register to a master
- Configuration is applied automatically
- Designed for 24/7 consistency

### Hybrid and Post-Deployment Tasks (Salt)

- Events trigger configuration changes
- Can push commands to many servers instantly
- Useful for environments requiring rapid updates

---

## 7. Infrastructure as Code (IaC) vs Configuration Management

Many organizations use:

- **Terraform** or **CloudFormation** for *infrastructure provisioning*
- **Ansible / Puppet / Chef / SaltStack** for *configuration*

Infrastructure as Code tools build servers.
Configuration management tools configure them.

Simple distinction:

```
Terraform: "Make me 3 servers."
Ansible:   "Install Nginx and deploy my app on those servers."
```

---

## 8. Summary

This lesson covered the major configuration management tools used in modern DevOps environments:

- **Ansible** uses SSH and is simple and agentless.
- **Puppet** and **Chef** use agents and central masters for high consistency.
- **SaltStack** offers fast, scalable, event-driven automation.

Each tool automates idempotent configuration and reduces the need for manual scripting. Understanding their differences helps you design scalable, maintainable DevOps workflows.