

# Módulo 3: Automation in the Cloud

---

## Introducción

En este módulo se explora cómo la **automatización** se aplica en entornos **cloud (nube)** para gestionar recursos de manera eficiente, reproducible y escalable. Aprenderás cómo las herramientas modernas permiten desplegar infraestructura y servicios con scripts o archivos de configuración, eliminando tareas manuales y errores humanos.

---

## Conceptos clave

### ¿Qué es la automatización en la nube?

- Es el uso de **scripts o herramientas** para **provisionar, configurar y mantener** recursos en la nube (máquinas virtuales, redes, almacenamiento, etc.).
  - Se basa en principios como **Infrastructure as Code (IaC)** y **Declarative Configuration**.
- 

## Infrastructure as Code (IaC)

- Permite describir la infraestructura (VMs, redes, balanceadores, etc.) con **archivos de texto versionables**.
  - Facilita:
    - Reproducibilidad.
    - Escalabilidad.
    - Auditoría de cambios.
  - Herramientas comunes:
    - **Terraform** (HashiCorp)
    - **AWS CloudFormation**
    - **Google Cloud Deployment Manager**
    - **Ansible, Puppet, Chef** (para configuración post-provisionamiento)
- 

## Declarative vs Imperative Automation

Enfoque	Descripción	Ejemplo
Imperativo	Decís <i>cómo</i> llegar al estado deseado paso a paso.	Script Bash que instala software línea por línea.
Declarativo	Describís <i>qué</i> querés lograr, y la herramienta decide cómo hacerlo.	Archivo YAML en Terraform que declara una instancia con ciertas propiedades.

---

## Herramientas y servicios populares

- **AWS, Google Cloud Platform (GCP), Azure:** todos soportan IaC.
  - **Terraform:** lenguaje declarativo para múltiples proveedores cloud.
  - **Ansible:** ideal para automatización de configuraciones y despliegues.
  - **Puppet y Chef:** más usados en entornos corporativos grandes.
- 

## 💡 Ejemplo conceptual

```
# Terraform: crear una VM en Google Cloud
resource "google_compute_instance" "vm_instance" {
  name          = "mi-servidor"
  machine_type  = "e2-medium"
  zone          = "us-central1-a"

  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-12"
    }
  }

  network_interface {
    network = "default"
    access_config {}
  }
}
```

➡ Este código declara **qué** máquina querés (no cómo instalarla paso a paso). Terraform se encarga de crear, configurar y mantener ese estado.

---

## 🔄 Ventajas de la automatización en la nube

- 🛠 **Consistencia** entre entornos (producción, pruebas, desarrollo).
  - ⚡ **Despliegue rápido** y escalable.
  - 📖 **Historial de cambios** mediante control de versiones.
  - 🔄 **Reutilización** de scripts e infraestructura.
  - 💰 **Optimización de costos** automatizando apagado de instancias inactivas.
- 

## 🔗 Integración con CI/CD

La automatización cloud se integra con **pipelines de CI/CD**, permitiendo:

1. Probar código automáticamente.
  2. Desplegar aplicaciones a entornos cloud sin intervención manual.
  3. Validar infraestructura antes de aplicar cambios (por ejemplo, con **terraform plan**).
- 

## ⚙ Comandos o prácticas comunes

- `terraform init` → inicializa el entorno.
- `terraform plan` → muestra los cambios que se harán.
- `terraform apply` → ejecuta la creación o modificación.
- `gcloud compute instances create ...` → ejemplo imperativo en GCP CLI.
- `ansible-playbook archivo.yml` → ejecuta configuración declarada.

---

## Resumen del módulo

Tema	Concepto clave
Automatización cloud	Uso de scripts y archivos para gestionar recursos.
IaC	Descripción declarativa y versionable de infraestructura.
Terraform / Ansible	Herramientas principales del ecosistema.
Declarativo vs Imperativo	Dos paradigmas de automatización.
CI/CD + Cloud	Integración para despliegue continuo.

---

## Referencias

- Google Cloud Docs: <https://cloud.google.com/docs>
- Terraform: <https://www.terraform.io/>
- Ansible: <https://docs.ansible.com/>
- Puppet: <https://puppet.com/docs/>