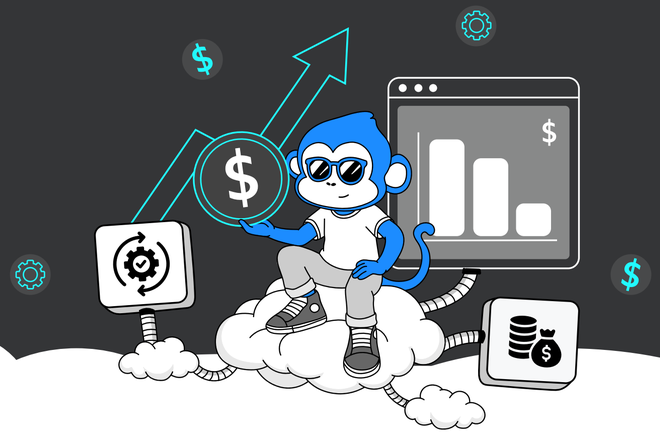
Optimize AWS Costs with Terraform:
What You Need to Know



Section 1: Why It Matters

Cloud Cost Challenges on AWS:

- Overprovisioned EC2 and RDS instances
- Forgotten resources (EBS, S3, Lambda)
- No pre-deployment visibility into spend
- Manual clean-up and no drift detection

Section 2: Terraform-Based Solutions

- ✅ **Codify Cost-Saving Defaults**
  Use `t3.micro`, `GLACIER`, and tagging policies in code.
- ✅ **Automate Cleanups**
  Apply `lifecycle` rules to delete/transition resources.
- ✅ **Conditional Deployments**
  Avoid waste in non-prod with `count + env` variables.
- ✅ **Pre-Deploy Cost Estimates**
  Use **Infracost** in CI pipelines to avoid surprises.
- ✅ **Budget Alerts as Code**
  Set up `aws_budgets_budget` in Terraform for spend limits.
- ✅ **Spot & Auto Scaling**
  Mix On-Demand + Spot in `autoscaling_group` configs.

Section 3: Top Tools for Terraform Workflows

Tool	Purpose
Infracost	Cost diff in Terraform plans
ControlMonkey	Guardrails, drift detection, optimization
AWS Budgets	Budget enforcement as code
AWS Cost Explorer	Trend visualization
Compute Optimizer	EC2/ASG right-sizing advice

Section 4: Pro Tips

- Always run `terraform plan` before apply
- Use `-var` and `.tfvars` to simulate environments
- Build reusable cost-aware **modules**
- Automate drift detection + tagging with ControlMonkey