Module 3: Secrets and Sensitive Data Management

Securing Terraform Code, State, and CI/CD Pipelines

Module Overview

What This Module Covers

- Handling sensitive data in Terraform projects
- Preventing credential leaks in code and pipelines
- Integrating secure secret managers
- Protecting Terraform state and logs

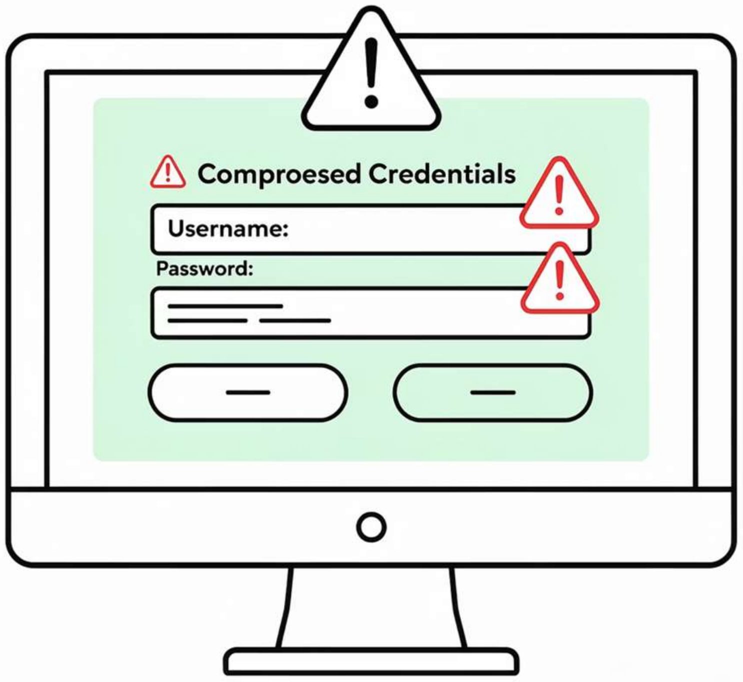
This module focuses on one of the biggest real-world Terraform risks – exposed credentials. We'll learn how professional DevOps teams securely handle secrets in infrastructure automation.

Learning Objectives

By the end of this module, you will be able to:

- *Identify risks of hard-coded secrets*
- *Retrieve secrets securely from AWS Secrets Manager*
- *Use HashiCorp Vault with Terraform*
- *Inject secrets securely via Jenkins pipelines*
- *Implement secret rotation best practices*
- *Prevent sensitive values from leaking into state files*

Emphasize that secret management is not optional in production – it's a security requirement.



Why Do We Need Secret Management?

Real-World Scenario

A DevOps team:

- Hard-codes DB passwords in Terraform
- Pushes code to GitHub
- Uses same credentials across environments

Result:

● Password leaks	● Compliance violations	● High security risk

Most breaches happen because secrets are stored in code or logs.

Problems Without Secure Secrets

Common Issues

Secrets committed to Git

Credentials permanently stored in version control history

Passwords visible in Terraform state

State files contain plain-text sensitive values

Credentials printed in Jenkins logs

Build logs expose secrets to anyone with access

Manual secret rotation downtime

Updating credentials requires code changes and redeployment

Audit failures

Unable to track who accessed secrets and when

Even experienced teams accidentally expose secrets if proper systems aren't in place.

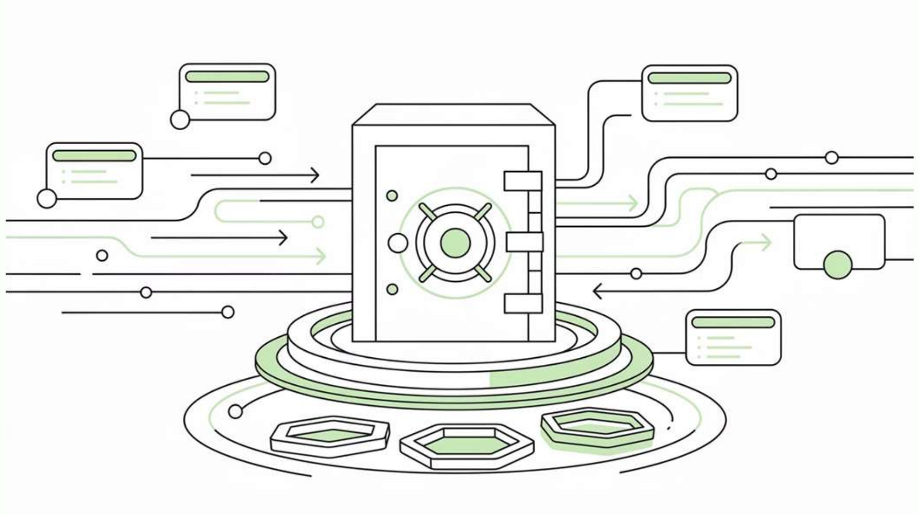
Why Basic Approaches Fail

What People Try (and why it fails)

Method	Why It's Not Safe
Plain Terraform variables	Stored in state file
.tfvars files	Often committed to Git
Environment variables	Can appear in logs
Hard-coding	Visible everywhere

Important: Terraform state is a hidden danger – it can contain plain-text secrets.

How Secure Secret Management Solves This



Secrets retrieved at runtime

No credentials in source code

Temporary pipeline injection

Centralized rotation

Compliance friendly

Secret managers act as a secure vault, not a config file.

Risk of Hard-Coded Secrets

Bad Practice Example

```
variable "db_password" {
  default = "P@ssw0rd123"
}
```

Where This Password Appears

Terraform state

Git history

Plan output

Logs

Critical: This is one of the most common beginner mistakes.

Using AWS Secrets Manager

Terraform can fetch secrets securely

```
data
"aws_secretsmanager_secret_version"
"db" {
  secret_id =
"prod/database/password"
}
```

Terraform is not storing the password – it retrieves it dynamically.

Using the Secret in a Resource

```
resource "aws_db_instance" "main" {
  engine          = "mysql"
  username        = "admin"
  password        = data.aws_secretsmanager_secret_version.db.secret_string
  instance_class  = "db.t3.micro"
}
```

Key Point: The secret never exists in your Git repository.

Using HashiCorp Vault

```
data "vault_generic_secret" "db" {
  path = "secret/data/prod/db"
}

locals {
  db_password =
data.vault_generic_secret.db.data["
password"]
}
```

Vault is powerful for dynamic credentials and strict access policies.

Injecting Secrets via Jenkins

```
withCredentials([string(credentialsId: 'db-password', variable: 'DB_PASS')]) {
  sh '''
    terraform apply -var="db_password=$DB_PASS" -auto-approve
  '''
}
```

Security Benefits

✓ *Masked in logs*

✓ *Not stored in code*

✓ *Temporary access*

Credential Rotation

Best Practice Flow

01

Rotate secret in Secrets Manager / Vault

02

Jenkins fetches updated value

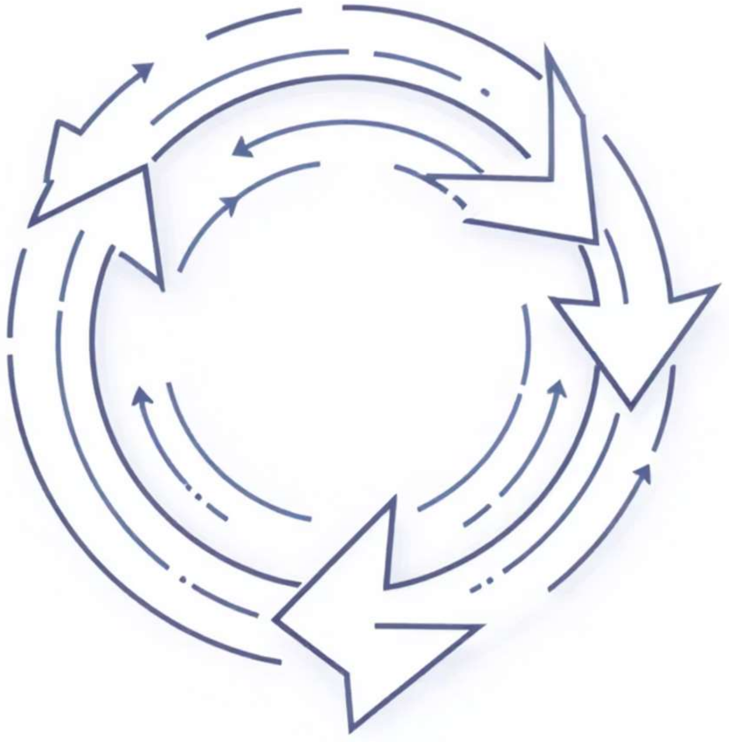
ĆĊ

Terraform uses new credential

04

No code changes required

Rotation reduces long-term exposure risk.



Practical Example: Secure EC2 Deployment

Step 1: Fetch Secret from AWS Secrets Manager

```
data "aws_secretsmanager_secret_version" "app_secret" {
  secret_id = "prod/app/api_key"
}
```

This retrieves the latest version of the API key stored in AWS Secrets Manager.

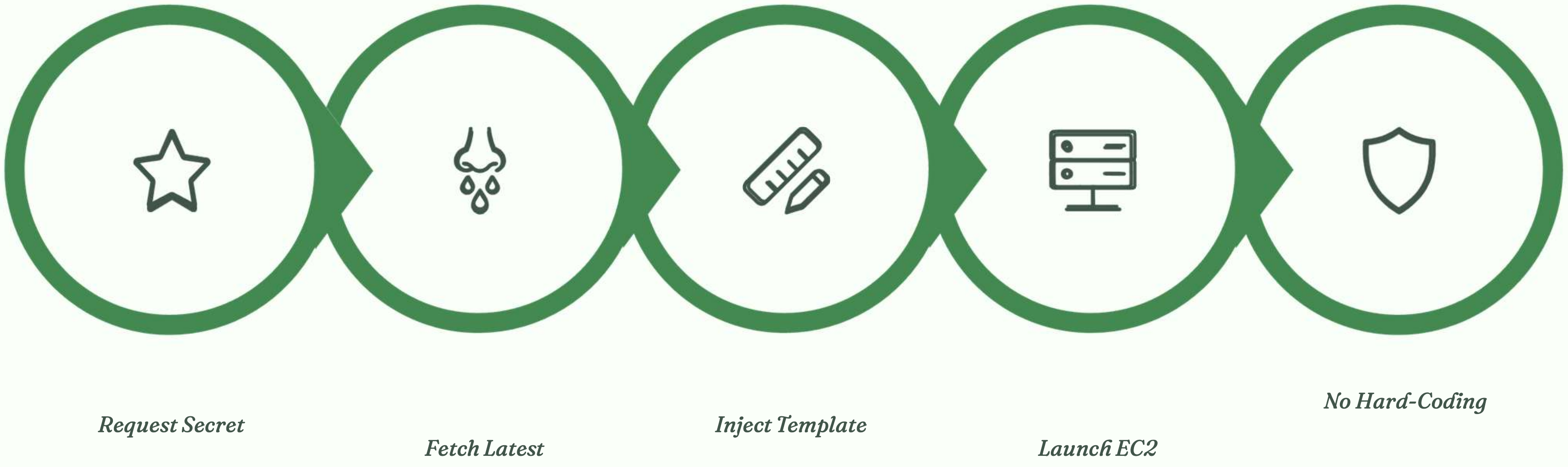
Inject Secret into User Data

Step 2: Use Secret in EC2 Instance

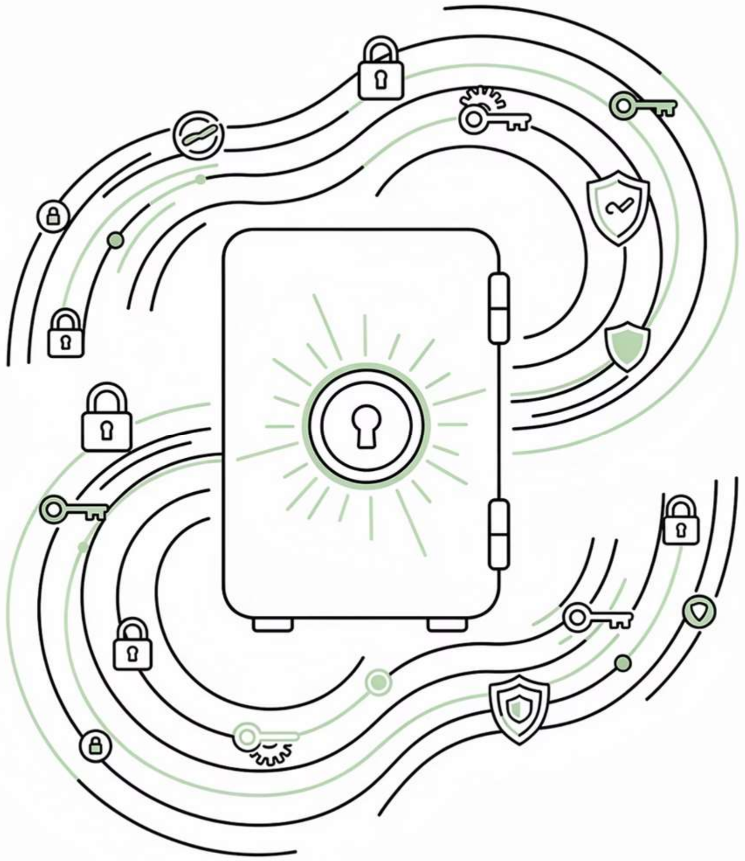
```
resource "aws_instance" "app_server" {
  ami           = "ami-12345678"
  instance_type = "t3.micro"

  user_data = templatefile("${path.module}/userdata.sh", {
    api_key = data.aws_secretsmanager_secret_version.app_secret.secret_string
  })
}
```

The secret is securely injected into the EC2 instance configuration at launch time.

Step-by-Step Flow



This complete workflow ensures that secrets are never hard-coded, stored in Git, or exposed in logs – maintaining security throughout the entire deployment process.



Combining with Other Terraform Concepts & Best Practices

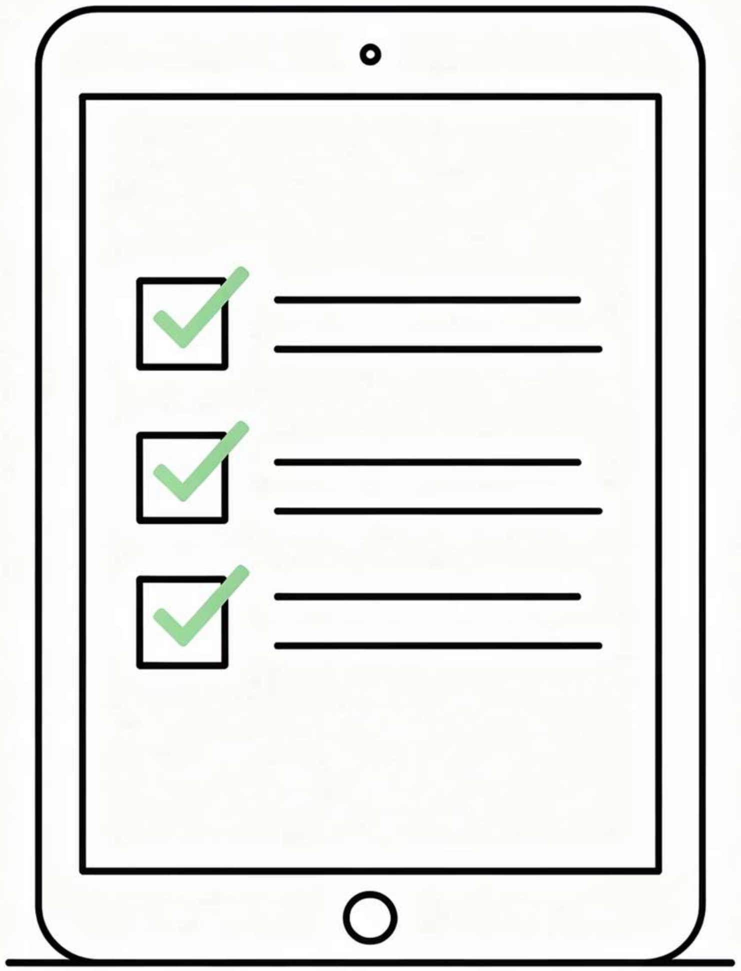
Combining with Other Terraform Concepts

Secrets work with:

- *Variables*
- *Locals*
- *for_each*
- *Dynamic blocks*
- *Remote state encryption*

Common Mistakes

- ❌ *Secrets in .tfvars*
- ❌ *Echoing secrets in pipelines*
- ❌ *Public Terraform plan logs*
- ❌ *Not marking variables as sensitive = true*
- ❌ *Unencrypted state storage*



Key Takeaways

- ✓ *Never hard-code secrets*
- ✓ *Use Secrets Manager or Vault*
- ✓ *Inject at runtime via CI/CD*
- ✓ *Mask logs*
- ✓ *Rotate credentials*
- ✓ *Protect state files*

Knowledge Check

Q1: Where should credentials be stored?

A. Terraform variable

B. GitHub

C. Secret manager

D. Jenkins logs

Q2: Jenkins credentials binding does what?

A. Stores passwords in code

B. Secure runtime injection

C. Prints secrets

D. Saves to state

Answers

1

C

2

B

Key Terms

SECRET MANAGER

CREDENTIAL ROTATION

SENSITIVE VARIABLE

VAULT

TERRAFORM STATE

Documentation Links

- Terraform Sensitive Variables
- AWS Secrets Manager Provider
- Vault Provider
- Jenkins Credentials Plugin

If you want, I can next:

✅ Convert this into proper PPT-ready text formatting

✅ Add diagram slide ideas

✅ Create a lab exercise slide section