# **Lab: Terraform + Console Automation**

**Lab:** Provision Redpanda/ Kafka resources via Terraform

**Tasks:** Write Terraform for **topic creation**, **ACLs**, and **security policies**

**Goal:** Automate admin tasks as **infrastructure-as-code** and validate in Console

## Purpose of the Lab

Enable learners to **codify Redpanda administration as Infrastructure-as-Code (IaC)** so topics, ACLs, and security policies are created, reviewed, and reproduced consistently across environments. By writing Terraform for **topic configuration** (partitions, retention, cleanup) and **least-privilege ACLs** (producer/consumer + group scope), participants practice:

* **Provider setup & parameterization:** securely wiring SASL/TLS auth and using variables instead of hardcoded values.
* **Idempotent provisioning:** using plan/apply/destroy to create and update resources predictably.
* **Security-as-code:** embedding least-privilege policies in versioned HCL for auditability and peer review.
* **Drift detection & reconciliation:** spotting Console-side changes in terraform plan and restoring the desired state.

The outcome is a repeatable, auditable workflow that reduces manual Console errors and scales cleanly from dev to prod.

## **Learning outcomes**

By the end, learners can:

* Configure the **Kafka Terraform provider** to talk to a SASL/TLS Redpanda cluster.
* Declare **topics** with config (retention, cleanup policy, partitions, RF).
* Declare **ACLs** for **least privilege** (producer role, consumer role + group scope).
* Run terraform plan/apply, validate in **Console**, and observe **drift detection**.

## Prerequisites

* Terraform **v1.3+** installed.
* Python labs done (you have a working bootstrap + credentials).
* Two service accounts exist (from Lab 5):
  + svc\_orders\_producer (password known)
  + svc\_orders\_consumer (password known)
* Cluster is reachable on SASL\_SSL, SCRAM-SHA-512.

**Never hardcode secrets** in HCL. Use environment variables or **terraform.tfvars** that you keep out of git.

## Project layout

Create a new folder, e.g., **tf-redpanda-lab**:

|  |
| --- |
| tf-redpanda-lab/  providers.tf  variables.tf  main.tf  outputs.tf  terraform.tfvars.example |

## **Note: svc\_tf\_admin\_or\_reader** in the Terraform example is just a **placeholder** for “the user Terraform will log in as

Use an existing account that already has enough permissions (e.g., your admin user).

## 

## **Provider & variables**

**providers.tf**

Defines Terraform’s dependencies and configures the Kafka (Redpanda) provider. It pins the Mongey/kafka provider version and tells Terraform how to connect to your cluster (bootstrap address, TLS on, SASL username/password, SCRAM-SHA-512). In short: this file teaches Terraform how to talk to Redpanda.

|  |
| --- |
| terraform {  required\_version = ">= 1.3.0"  required\_providers {  kafka = {  source = "Mongey/kafka"  version = "~> 0.7"  }  } }  provider "kafka" {  bootstrap\_servers = [var.bootstrap]  tls\_enabled = true   sasl\_username = var.username  sasl\_password = var.password  sasl\_mechanism = "scram-sha256" # <-- use lowercase } |

**variables.tf**

Declares all the inputs your config needs so nothing is hardcoded: bootstrap host:port, provider username/password (marked sensitive), the producer/consumer principals, the controlled topic name, allowed consumer-group prefix, and topic tuning (partitions, replication factor, retention, cleanup policy). In short: this file defines the knobs you can set per environment.

|  |
| --- |
| variable "bootstrap" {  type = string  description = "Kafka/Redpanda bootstrap host:port" }  variable "username" {  type = string  description = "User for provider auth (can be any account with read metadata)"  sensitive = true }  variable "password" {  type = string  description = "Password for provider auth"  sensitive = true }  # ACL principals (service accounts created in Console) variable "producer\_principal" {  type = string  description = "ACL principal for producer (e.g., User:svc\_orders\_producer)"  default = "User:svc\_orders\_producer" }  variable "consumer\_principal" {  type = string  description = "ACL principal for consumer (e.g., User:svc\_orders\_consumer)"  default = "User:svc\_orders\_consumer" }  # Controlled topic + group policy variable "topic\_name" {  type = string  default = "secured\_orders"  description = "Topic controlled by Terraform" }  variable "group\_prefix" {  type = string  default = "cg-secured-orders-"  description = "Allowed consumer group prefix" }  # Topic configuration variable "partitions" {  type = number  default = 6 } variable "replication\_factor" {  type = number  default = 3 } variable "retention\_ms" {  type = number  default = 604800000 # 7 days } variable "cleanup\_policy" {  type = string  default = "delete" # or "compact" } |

**terraform.tfvars example** (do **not** commit a real one with credentials)

|  |
| --- |
| bootstrap = "d2heugc8ljef72usu9gg.any.ap-south-1.mpx.prd.cloud.redpanda.com:9092" username = "svc\_tf\_admin\_or\_reader" password = "REDACTED"  # If your provider user is not a powerful admin, that's fine; # we only need metadata read to apply topic+ACL resources.  producer\_principal = "User:svc\_orders\_producer" consumer\_principal = "User:svc\_orders\_consumer"  topic\_name = "secured\_orders" group\_prefix = "cg-secured-orders-" partitions = 6 replication\_factor = 3 retention\_ms = 604800000 cleanup\_policy = "delete" |

Alternatively, use environment variables:

|  |
| --- |
| export TF\_VAR\_bootstrap="host:9092" export TF\_VAR\_username="svc\_tf\_admin\_or\_reader" export TF\_VAR\_password="\*\*\*" |

## 

## **Resources: Topic + ACLs**

**main.tf**

Implements the actual infrastructure: creates the kafka\_topic with your configs and attaches least-privilege kafka\_acl rules—producer gets Write + Describe on the topic; consumer gets Read + Describe on the topic and Read on a constrained consumer-group (literal or prefixed). Optional idempotent write ACL can be added. In short: this file is the blueprint that builds the topic and its security policy.

|  |
| --- |
| # Topic with custom config resource "kafka\_topic" "topic" {  name = var.topic\_name  partitions = var.partitions  replication\_factor = var.replication\_factor   config = {  "cleanup.policy" = var.cleanup\_policy  "retention.ms" = tostring(var.retention\_ms)  # Optional: tune segment size to observe faster retention:  # "segment.bytes" = "134217728" # 128MB  } }  # --- Least-privilege ACLs ---  # Producer: WRITE + DESCRIBE on topic resource "kafka\_acl" "producer\_write" {  resource\_type = "Topic"  resource\_name = kafka\_topic.topic.name  acl\_principal = var.producer\_principal  acl\_host = "\*"  acl\_operation = "Write"  acl\_permission\_type = "Allow" }  resource "kafka\_acl" "producer\_describe" {  resource\_type = "Topic"  resource\_name = kafka\_topic.topic.name  acl\_principal = var.producer\_principal  acl\_host = "\*"  acl\_operation = "Describe"  acl\_permission\_type = "Allow" }  # Consumer: READ + DESCRIBE on topic resource "kafka\_acl" "consumer\_read" {  resource\_type = "Topic"  resource\_name = kafka\_topic.topic.name  acl\_principal = var.consumer\_principal  acl\_host = "\*"  acl\_operation = "Read"  acl\_permission\_type = "Allow" }  resource "kafka\_acl" "consumer\_describe" {  resource\_type = "Topic"  resource\_name = kafka\_topic.topic.name  acl\_principal = var.consumer\_principal  acl\_host = "\*"  acl\_operation = "Describe"  acl\_permission\_type = "Allow" }  # Consumer GROUP ACL (prefix) to restrict which groups the consumer can join resource "kafka\_acl" "consumer\_group\_read" {  resource\_type = "Group"  resource\_name = var.group\_prefix # prefix pattern  acl\_principal = var.consumer\_principal  acl\_host = "\*"  acl\_operation = "Read"  acl\_permission\_type = "Allow" }  # Optional: IdempotentWrite for producer (only if you enable idempotence in producer clients) # resource "kafka\_acl" "producer\_idempotent" { # resource\_type = "Cluster" # resource\_name = "kafka-cluster"# # acl\_principal = var.producer\_principal # acl\_host = "\*" # acl\_operation = "IdempotentWrite" # acl\_permission\_type = "Allow" # } |

**outputs.tf**

|  |
| --- |
| output "topic" {  value = kafka\_topic.topic.name }  output "producer\_principal" {  value = var.producer\_principal }  output "consumer\_principal" {  value = var.consumer\_principal }  output "group\_prefix" {  value = var.group\_prefix } |

## Run the Terraform workflow

In the tf-redpanda-lab folder:

|  |
| --- |
| terraform init |

**Expected (truncated):**

|  |
| --- |
| Initializing the backend... Initializing provider plugins... - Finding Mongey/kafka versions matching "~> 0.7"... - Installing Mongey/kafka v0.7.x... Terraform has been successfully initialized! |

Then:

|  |
| --- |
| # either rely on TF\_VAR\_\* env vars or create terraform.tfvars terraform plan \  -var 'bootstrap=d2heugc8ljef72usu9gg.any.ap-south-1.mpx.prd.cloud.redpanda.com:9092' \  -var 'username=svc\_tf\_admin\_or\_reader' \  -var 'password=REDACTED' |

**Expected plan highlights:**

|  |
| --- |
| Plan: 6 to add, 0 to change, 0 to destroy.   + resource "kafka\_topic" "topic" { name = "secured\_orders" ... }   + resource "kafka\_acl" "producer\_write" (Topic/LITERAL/Write Allow)  + resource "kafka\_acl" "producer\_describe" (Topic/LITERAL/Describe Allow)  + resource "kafka\_acl" "consumer\_read" (Topic/LITERAL/Read Allow)  + resource "kafka\_acl" "consumer\_describe" (Topic/LITERAL/Describe Allow)  + resource "kafka\_acl" "consumer\_group\_read" (Group/PREFIXED/Read Allow) |

Apply:

|  |
| --- |
| terraform apply -auto-approve \  -var 'bootstrap=d2heugc8ljef72usu9gg.any.ap-south-1.mpx.prd.cloud.redpanda.com:9092' \  -var 'username=svc\_tf\_admin\_or\_reader' \  -var 'password=REDACTED' |

**Expected (truncated):**

|  |
| --- |
| kafka\_topic.topic: Creating... kafka\_topic.topic: Creation complete after 1s [id=secured\_orders] kafka\_acl.producer\_write: Creation complete kafka\_acl.producer\_describe: Creation complete kafka\_acl.consumer\_read: Creation complete kafka\_acl.consumer\_describe: Creation complete kafka\_acl.consumer\_group\_read: Creation complete  Apply complete! Resources: 6 added, 0 changed, 0 destroyed.  **Outputs:** topic = "secured\_orders" producer\_principal = "User:svc\_orders\_producer" consumer\_principal = "User:svc\_orders\_consumer" group\_prefix = "cg-secured-orders-" |

## **Validate in Redpanda Console**

* **Topics → secured\_orders → Configuration**
  + Verify cleanup.policy and retention.ms match your HCL.
* **Security → ACLs**
  + Filter by svc\_orders\_producer and svc\_orders\_consumer.
  + You should see:
    - Producer: Topic secured\_orders **WRITE Allow**, **DESCRIBE Allow**
    - Consumer: Topic secured\_orders **READ Allow**, **DESCRIBE Allow**
    - Consumer Group: cg-secured-orders- (Prefixed) **READ Allow**

## **Functional verification (quick)**

Use your earlier test scripts:

**Producer user** (svc\_orders\_producer):

|  |
| --- |
| export RP\_USERNAME="svc\_orders\_producer" export RP\_PASSWORD="\*\*\*\*" python ../acl\_producer\_test.py |

|  |
| --- |
| # Expected: [producer] OK: secured\_orders-<p>@<offset> |

**Consumer user** (svc\_orders\_consumer):

|  |
| --- |
| export RP\_USERNAME="svc\_orders\_consumer" export RP\_PASSWORD="\*\*\*\*" python ../acl\_consumer\_test.py |

|  |
| --- |
| # Expected: reads the produced message (or "polled 0 messages" if no new) |

Try a **negative test**: change topic in the producer test to unapproved\_topic → should fail with TopicAuthorizationFailed.  
Try a rogue **group** in consumer test → should fail with GroupAuthorizationFailed.

## Drift detection demo (teaching moment)

1. In **Console**, edit secured\_orders → change retention.ms to 3600000 (1 hour) and **Save**.
2. Back in Terraform:

|  |
| --- |
| terraform plan |

**Expected:** Terraform shows a **change** on kafka\_topic.topic.config["retention.ms"] from 604800000 → 3600000.  
3) Reconcile by terraform apply to push your desired state back—this is **drift correction**.

## Cleanup (optional)

Destroy all lab resources managed by Terraform:

|  |
| --- |
| terraform destroy -auto-approve |

**Expected:**

Destroy complete! Resources: 6 destroyed.

This removes the topic and ACLs; confirm no one else depends on them.

## 

## Troubleshooting

* **NoBrokersAvailable (during plan/apply):** Bootstrap or SASL/TLS config missing in provider (check vars/env).
* **TopicAuthorizationFailed (on topic create):** The **provider user** may not have privilege to create topics.
  + Workaround: have an admin pre-create the topic (or run provider with an admin account).
* **ACLs appear in wrong cluster:** Ensure you’re applying to the same cluster shown in Console (breadcrumb).
* **Consumer still blocked:** Make sure **both Topic READ** and **Group READ** (literal or prefixed) are present for the **same** principal.
* **Provider TLS issues:** If your environment uses custom CAs, you may need a truststore; otherwise keep tls\_enabled = true with public CA.

## Success criteria (checklist)

* terraform apply created the **topic** and **ACLs**.
* Console shows the same policies you declared in HCL.
* Producer can **only** write secured\_orders; fails elsewhere.
* Consumer can **only** read secured\_orders; can **only** join groups with prefix cg-secured-orders-.
* terraform plan surfaces **drift** if you change settings in Console.