

Advanced Scenario: Full Terraform Import of a Brownfield GCP Subnet (Including Tags)

Learning Objective

Demonstrate the complete process of converting an externally-created GCP resource into a fully Terraform-managed object — not just its main resource, but all associated metadata.

This scenario covers:

- Importing a core resource (subnet)
- Discovering hidden dependent resources
- Analysing what Terraform can see vs what it cannot see
- Importing Tag Key, Tag Value, Tag Binding
- Verifying drift
- Ending with a fully rebuildable subnet (true IAC)

Stage 1 — Import the Core Resource (Subnet)

A manually created subnet exists in GCP with the following properties:

- Name: eu-subnet
- Region: europe-west2
- CIDR: 10.0.3.0/24
- Secondary range: demo (192.168.1.0/24)
- Tag: dept: finance

Terraform configuration:

```
resource "google_compute_subnetwork" "eu_subnet" {  
  name      = "eu-subnet"  
  region    = "europe-west2"  
  ip_cidr_range = "10.0.3.0/24"
```

```
network    = google_compute_network.lab_vpc.id
```

```
secondary_ip_range {  
  range_name  = "demo"  
  ip_cidr_range = "192.168.1.0/24"  
}  
}
```

Import command:

```
terraform import google_compute_subnetwork.eu_subnet  
projects/<PROJECT_ID>/regions/europe-west2/subnetworks/eu-subnet
```

Terraform will show “No changes”, because it only tracks fields it knows about.

Stage 2 — Unexpected Drift Discovery

The subnet in the GCP console shows a tag: dept: finance.

Terraform state does not show this.

Reason:

GCP Resource Manager tags are NOT part of the subnet resource.

They exist in a separate API and must be imported separately.

Lesson:

Not everything visible in the GUI belongs to the resource you're importing.

Stage 3 — What Actually Exists

The dept: finance tag is actually three separate resources:

- Tag Key (dept)

- Tag Value (finance)
- Tag Binding (binding value to the subnet)

This structure is hidden by the GUI but exposed by Terraform and the API.

Stage 4 — Import All Linked Resources

4.1 Import Tag Key

```
gcloud tags keys list --project <PROJECT_ID>
```

```
terraform import google_tags_tag_key.dept tagKeys/<TAG_KEY_ID>
```

4.2 Import Tag Value

```
gcloud tags values list --parent=tagKeys/<TAG_KEY_ID>
```

```
terraform import google_tags_tag_value.dept_finance tagValues/<TAG_VALUE_ID>
```

4.3 Import Tag Binding

```
gcloud resource-manager tags bindings list --  
parent="//compute.googleapis.com/projects/<PROJECT_ID>/regions/europe-  
west2/subnetworks/eu-subnet" --format="value(name)"
```

```
terraform import google_tags_tag_binding.eu_subnet_dept_finance <BINDING_NAME>
```

Stage 5 — After All Imports

Once all three resources are imported, a terraform plan becomes fully clean.

Terraform now owns:

- Subnet
- Tag Key
- Tag Value
- Tag Binding

This enables full reproducibility and drift detection.

Stage 6 — Rebuild Test

Delete the subnet manually in the GCP console, then run:

```
terraform apply
```

Terraform will recreate:

- Subnet
- Secondary range
- Tag Key
- Tag Value
- Tag Binding

Result: a fully deterministic rebuild, completing the IaC lifecycle.

This scenario demonstrates:

- Multi-layer resource modelling
- Hidden drift detection
- Why imports can be complex
- Brownfield IaC onboarding
- Google's multi-API infra model
- Ensuring complete rebuild capability