

Production paradoksu

```
$ terraform plan  
No changes.
```

- × Production'daki firewall rule farklı.
- × Bir VM yanlış network'te.
- × Load balancer'da eksik backend var.

Terraform neden hiçbir şey görmedi?

Bu haftanın kontrolü

Terraform Drift

Don't Deploy on Friday. Verify on Friday.

“No changes” ne demek?

Terraform aslında şunu söylüyor:

Seçili workspace, state ve provider kapsamında yönettiğim alanlarda uygulanacak bir değişiklik göremiyorum.

Bu çıktı:

≠ Tüm altyapının doğru olduğunu,

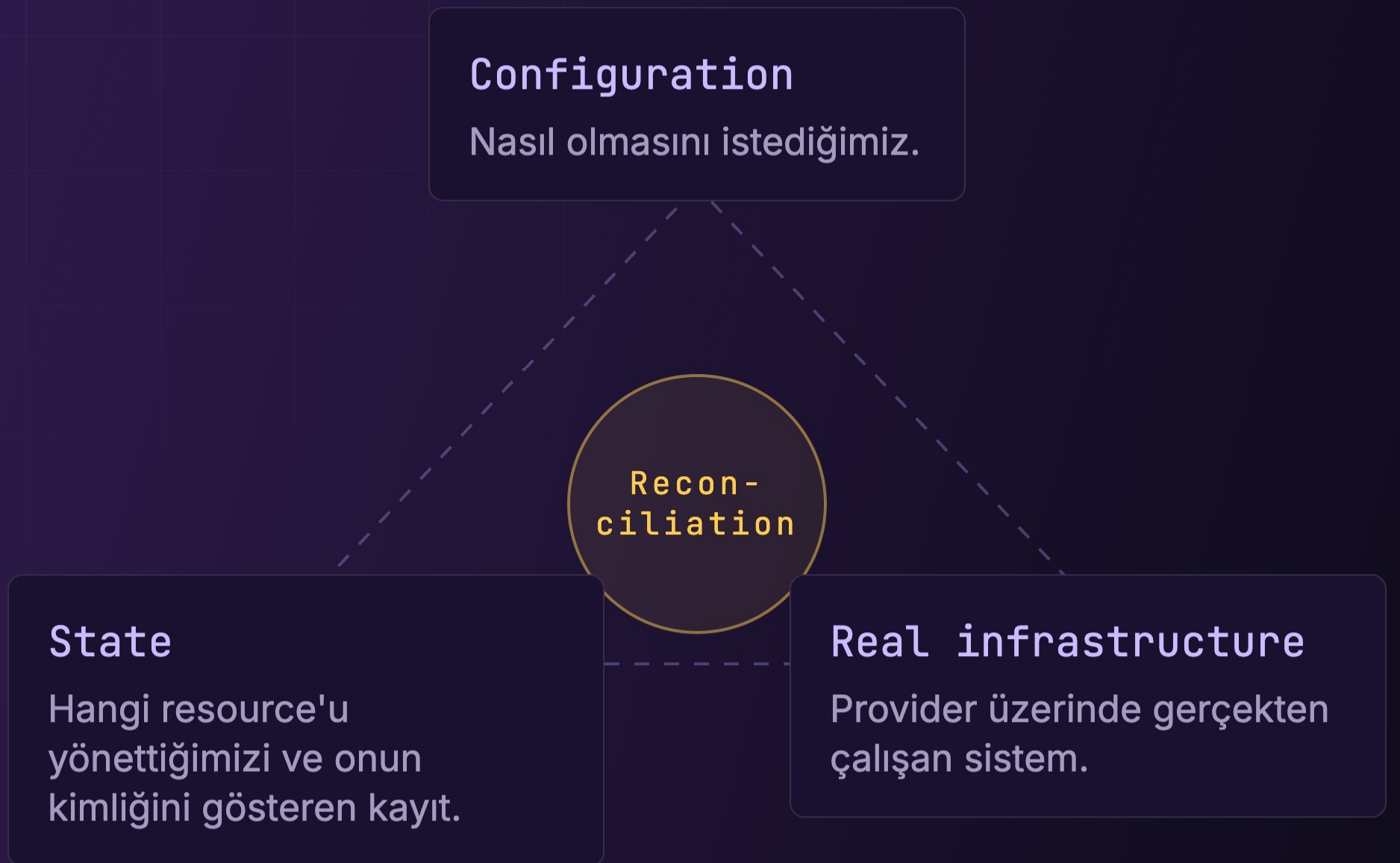
≠ Yönetilmeyen kaynak bulunmadığını,

≠ Servisin sağlıklı çalıştığını

KANITLAMAZ

Üç farklı gerçeklik

IaC ortamında tek bir source of truth yoktur:



Reliability, bu üç katmanın **sürekli uzlaştırılmasıyla** oluşur.

Drift neden görünmez kalır?

Terraform'un görüş alanı sınırlanabilir:

- 01 Yanlış workspace, account, tenant veya region kontrol ediliyordur.
- 02 Resource state içinde değildir ya da attribute `ignore_changes` kapsamındadır.
- 03 Refresh kapatılmıştır veya provider ilgili davranışı modellemiyordur.

Plan temiz olabilir.

Kapsam yine de eksik olabilir.

Yanlış refleks

“State'i refresh edelim, düzelsin.”

```
$ terraform apply -refresh-only
```

gerçek altyapıyı configuration'a döndürmez.

Sadece state'i, provider'ın bildirdiği mevcut duruma göre günceller.

Incident sırasında yapılan değişikliği configuration'a taşımadan state'e kaydetmek, kalıcı yönetim sağlamaz.

Önce hangi durumun doğru olduğuna karar verilmelidir.

Friday check

1 Doğru hedefte misiniz?

```
$ terraform workspace show  
$ terraform state list
```

Workspace, backend, provider account/tenant ve region bilgisini doğrulayın.

2 State ile gerçek kaynak farklı mı?

```
$ terraform plan -refresh-only
```

Bu komut remote resource'ları değiştirmeden drift'i incelemenizi sağlar. Planı apply etmeden önce farkın nedenini sınıflandırın.

3 Terraform neyi görmezden geliyor?

```
$ rg "ignore_changes|refresh=false" .
```

Ardından kritik resource inventory'sini provider'ın kendi CLI/API çıktısıyla karşılaştırın.

Sonucu birlikte okuyun

Refresh-only fark buldu + normal plan değişiklik önerdi

→ Gerçek altyapı configuration'dan sapmış. Değişikliği geri alın veya configuration'a taşıyın.

Refresh-only fark buldu + normal plan değişiklik önermedi

→ ignore_changes, computed attribute veya paylaşılan ownership olabilir. Bu alanın sahibini doğrulayın.

İki plan da temiz + provider çıktısı farklı

→ Yanlış scope, unmanaged resource veya provider'ın modellemediği bir kör nokta olabilir.

Drift'i yönetilebilir yapın

- 01 Drift detection'ı periyodik çalıştırın; remediation'ı otomatikleştirmeden önce değişikliği sınıflandırın.
- 02 Her `ignore_changes` için owner, gerekçe ve gözden geçirme tarihi tanımlayın.
- 03 Kritik invariant'ları Terraform dışından da doğrulayın: route, encryption, security rule, LB membership ve service readiness.

BAŞARI METRİĞİ

Drift'in oluşmaması değil, ne kadar hızlı fark edilip güvenli şekilde uzlaştırıldığıdır.

Reliability ilkesi

Code is the source of intent.
Production is the source
of evidence.

No changes, production'ın doğru olduğunu değil;
Terraform'un kendi kapsamı içinde yapacak bir değişiklik
bulamadığını söyler.

Sizin production ortamınızda Terraform'un
görmediği alanları kim doğruluyor?

Don't Deploy on Friday. Verify on Friday.

Every Friday. One Production Check.