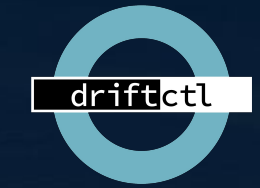


Infrastructure drift on AWS & driftctl

AWS Community Day, March 2021

MIND THE GAP

BETWEEN THE CODE AND THE PLATFORM

Story

- ❤️ GitOps
- 🥰 Users
- 😱 Drift

TL;DR

Almost everyone has experienced infrastructure drift recently.

We built driftctl to help.

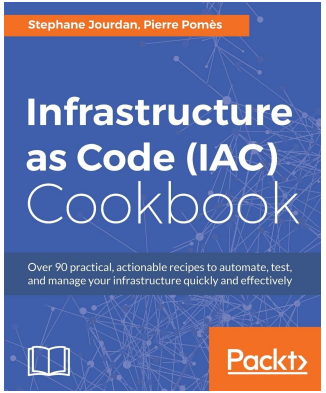
Why

- Even the best teams didn’t automate everything
- Scripts / Lambdas / Microservices are authenticated
- Customers and bosses do exist (*with admin credentials*)

whoami

Stephane Jourdan:

- @sjourdan (Twitter, GitHub, GitLab,...)
- 20 years %s/sysadmin/devops/g
- Co-founded 3 tech companies (🇨🇦 | 🇪🇺) and 1 sound studio.
- “Infrastructure-as-Code Cookbook” author
- Driftctl tool co-founder! cloudskiff/driftctl



Definition

Infrastructure Drift / ˈɪn frəˌstrʌk tʃər drɪft /

Noun

1. happens when the reality and the expectations don’t match.

Synonyms for Infrastructure Drift

1. omg

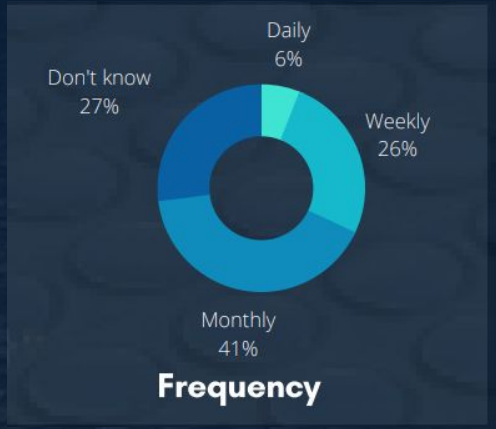
driftctl

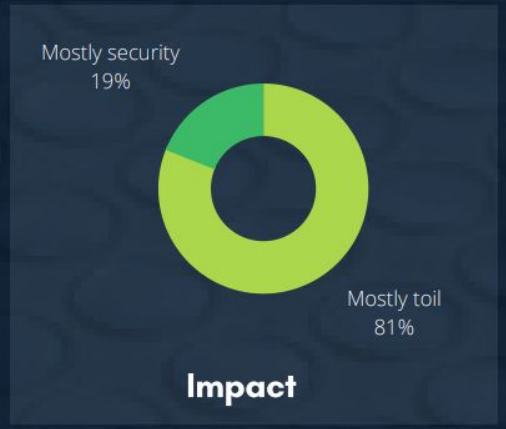
Main Causes

- Manual Changes: 96%
- Authenticated Applications: 50%
- Out-of-Sync IaC Environments: 44%

driftctl

Main Consequences





driftctl

Solutions

	Full GitOps workflow	terraform plan in a CRON job	Restrict access to the production environment	Restrict access to staging environments
Prevents developer generated drift	yes	no	partially	yes
Prevents cloud-provider generated drift	no	no	no	no
Makes drift visible	no	partially	no	no
Analyzes drift root cause	no	no	no	no
Limitations	Hard to rollout in legacy / complex environments.	Terraform plan does not "see" some changes		Decreases developer speed or requires the cost and capability of deploying sandbox environments

But...

- Even the best teams didn't automate everything
- Scripts / Lambdas / Microservices are authenticated
- Customers and bosses do exist (*with admin credentials*)

Stories!

Quick AWS IAM Story

How a simple lambda with read-only access ended up with rogue Administrative access and keys

- *without anyone noticing*

```
resource "aws_iam_user" "microservice_user" {
  name = "microservice-${data.terraform_remote_state.base.outputs.random_string}"

  tags = {
    Name = "microservice-${data.terraform_remote_state.base.outputs.random_string} User"
  }
}

resource "aws_iam_access_key" "microservice_user" {
  user = aws_iam_user.microservice_user.name
}

resource "aws_iam_user_policy_attachment" "microservice" {
  user       = aws_iam_user.microservice_user.name
  policy_arn = "arn:aws:iam::aws:policy/ReadOnlyAccess"
}
```

Quick AWS Security Group Story

How someone opened up everything to anyone on IPv4 & IPv6

- *without anyone noticing*

```
resource "aws_security_group" "supersecure" {
  name        = "supersecure"
  description = "Super Secure Security Group"

  tags = {
    Name = "Super Secure Security Group"
  }
}
```

```
resource "aws_security_group_rule" "supersecure_sg_rule_1" {
  type              = "ingress"
  from_port         = 22
  to_port           = 22
  protocol          = "tcp"
  cidr_blocks       = ["10.0.0.0/8"]
  security_group_id = aws_security_group.supersecure.id
}
```

Quick AWS S3 Story

How a scripting issue created a billing nightmare

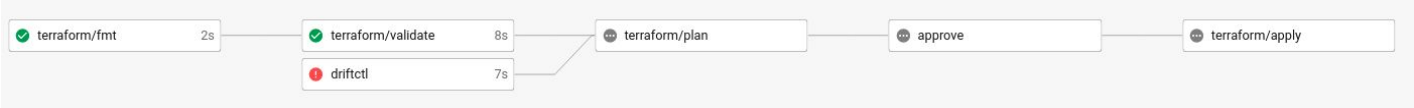
- *With only billing noticing*

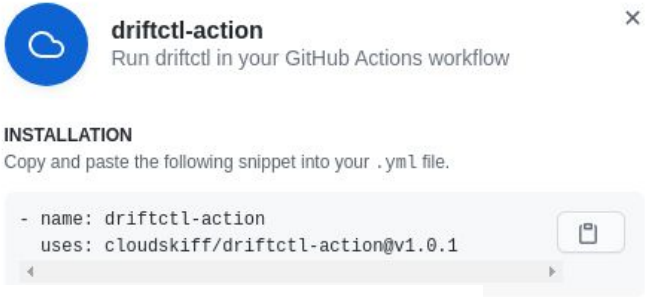
```
resource "aws_s3_bucket" "demo" {
  bucket = "${random_string.prefix.result}-demo"
  acl    = "private"
}
```

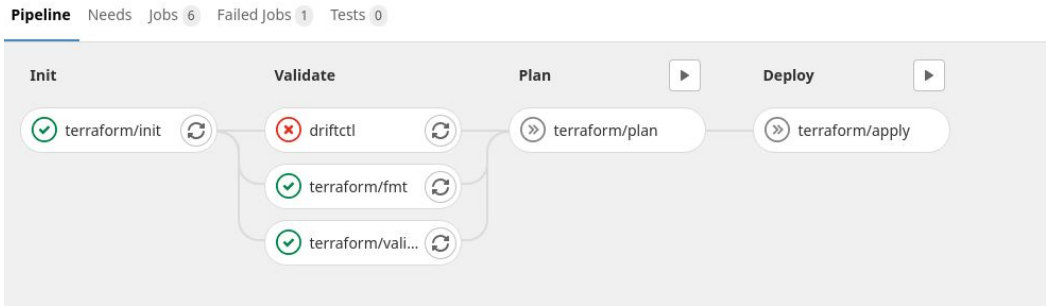
CI Integration

Show GitOps ❤️

- Circle CI Orb
- GitHub Action
- Gitlab CI
- ...







driftctl

Our own open-source solution for drift management

- AWS & GitHub Support (more to come)
- Terraform State support (local/S3/HTTP)
- Filtering & Ignore support
- Written in Go
- Apache 2.0 License

cloudskiff/driftctl

driftctl.com/d

TL;DR (Closing)

Almost everyone has experienced infrastructure drift recently.

We built driftctl to help.

MIND THE GAP

BETWEEN THE CODE AND THE PLATFORM