Driftctl hands-on demo!

SRE Paris Meetup January 2021

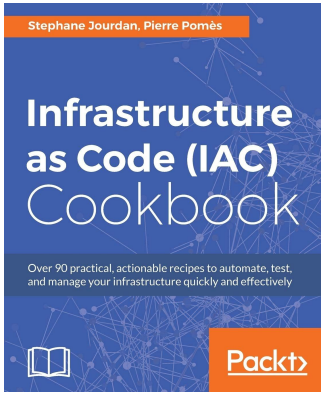
MIND THE GAP

between the code and the platform

whoami

Stephane Jourdan:

- @sjourdan (Twitter, GitHub, GitLab,...)
- 20 years (Dev)Ops
- Co-founded 3 tech companies (CAN/EU) and 1 sound studio.
- “Infrastructure-as-Code Cookbook” author
- Driftctl tool co-founder! GitHub: cloudskiff/driftctl



Agenda

1. Why driftctl
2. Demo!

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

How It Started

- We love and use Terraform (heavily)
- We learned that “Reality” > “Top Security Policy”
- We love to sleep well, too

Situation Report

- Changes happen outside of IaC (lambdas, scripts)
- me@mylaptop: $ ~/bin/terraform apply
- My [“boss”, “customer”, “coworker”] needs full console access for ${REASON} (and use it)
- Broken processes & incomplete commits
- Constraints: Skills, Time, Coverage
- IaC tests: low adoption

But...Terraform plan/apply!

- Terraform is an excellent provisioner.
- Provisioners are not tool meant to manage what's outside their scope.
- Security / compliance tools are another type of software.

```
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
```

driftctl

Stories!

Drifts Not “Seen”

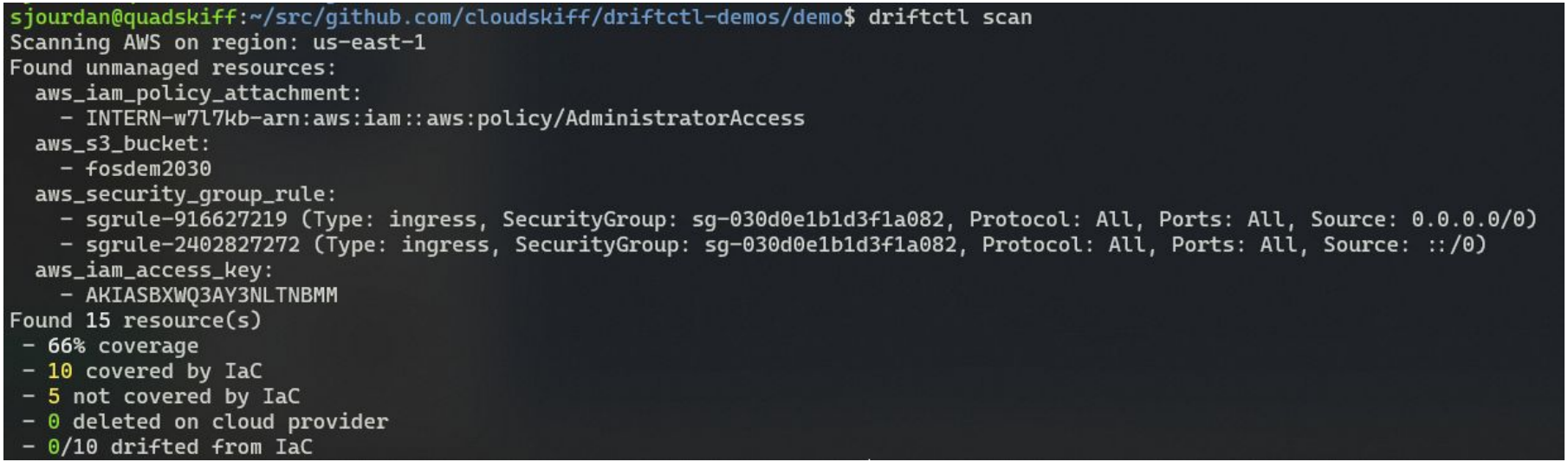
How an intern with read-only access ended up with rogue Administrative IAM access and keys

- *without anyone noticing*

```
resource "aws_iam_user" "intern_user" {
  name = "INTERN-${random_string.prefix.result}"

  tags = {
    Name = "INTERN-${random_string.prefix.result} User"
  }
}

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

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

Driftctl Output

- Resources (Unmanaged, Deleted, Drifted)
- Coverage
- Details
- JSON too



Drifts Not “Seen”

How someone opened everything to anyone on IPv4 & IPv6 on a Security Group

- *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
}
```

“Modified TFState” Drifts

How a scripting issue created an S3 billing nightmare

- *With only billing noticing*

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

```
Found drifted resources:
  - qahxxw-demo (aws_s3_bucket):
    ~ Versioning.0.Enabled: false ⇒ true
```

“Modified TFState” Drifts

How an EC2 VM was stopped and forgotten

- *without anyone noticing*

```
resource "aws_instance" "demo_instance_1" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"

  tags = {
    Name = "demo_instance_1"
  }

  volume_tags = {
    Name = "rootVol"
  }

  root_block_device {
    volume_type           = "gp2"
    volume_size           = 20
    delete_on_termination = true
  }
}
```

“Side-Effect” Drifts

How manually changing a VPC setting affected Security Group rules too

- *without anyone noticing*

Bonus: also modifies the TFState

“Manual” Drifts

How someone or something created uncontrolled resources (like many S3 buckets)

- *without anyone noticing*

```
// call S3 to create the bucket
s3.createBucket(bucketParams, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Location);
  }
});
```

“Workflow” Drifts

How automated scripts may impact TF as intended, but you’d still like to know (like CI/CD deploys, scans etc.)

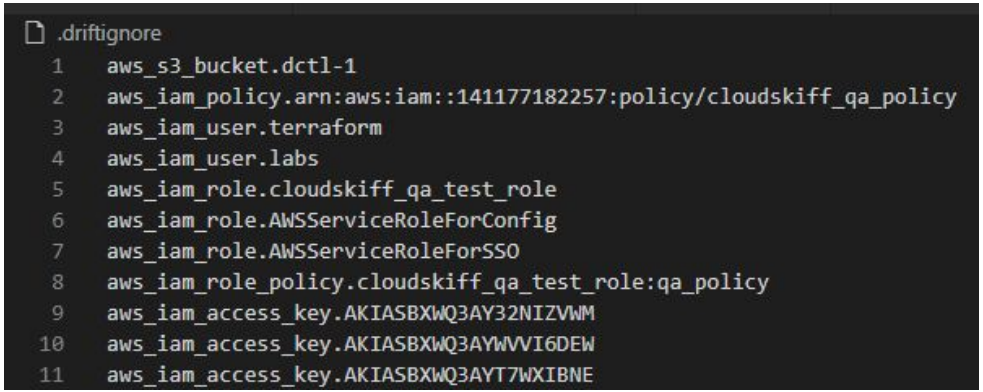
- *with just the right people noticing*


```
Found drifted resources:
  - qz8khv-lambda-demo (aws_lambda_function):
    ~ LastModified: "2021-01-15T15:41:31.183+0000" => "2021-01-15T16:23:30.327+0000"
    ~ Environment.0.Variables.VERSION: "1.0" => "2.0"
```

.driftignore

How to clean the output

- *So it's actually useful to people & processes*



Filtering

How to choose what to (not) see

- *So it's actually useful to people or processes*

```
$ driftctl scan --filter "Type=='aws_s3_bucket'"
Scanning AWS on region: us-east-1
Found unmanaged resources:
  aws_s3_bucket:
    - fosdem2031
```

JSON Output

“ driftctl | jq FTW!”

- *Some alerting script, somewhere*

```
$ driftctl scan -o json:///dev/stdout |  awk '{if(NR>1)print}' | jq '.coverage'
71
```

driftctl

Our own open-source solution for drift management

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

About driftctl

Take control of infrastructure drift

FLOSS CLI that tracks, analyzes, prioritizes, and warns of infrastructure drift

cloudskiff/driftctl

MIND THE GAP

between the code and the platform