

OpenTofu Day
NORTH AMERICA

No More Click-Ops Compliance: Postgres Audit Logging with OpenTofu

Akash Singh

#whoami

OpenTofu Day
NORTH AMERICA

Akash Singh (SKY)
Devops, MLOps and Infra at Sabi

Golden Kubestronaut
Google Summer of Code ‘24 @Keploy
Linux Foundation Mentee ‘25 @LitmusChaos
Runs the Cloud and DevOps Devroom @IndiaFoss
Lead of Student OSS Community of the year @**Point Blank**
Favourite CNCF Tool : Atlantis

/skysingh04

/skysingh04



But Sky, What is your problem with Click-Ops?

OpenTofu Day
NORTH AMERICA

Does the Devops Engineer remember…

1. **Undocumented access**

   Why the teammate has Production DB Access?

2. **Prone to drift**

   To revoke the access of the teammate after?

3. **And forget about keeping records**

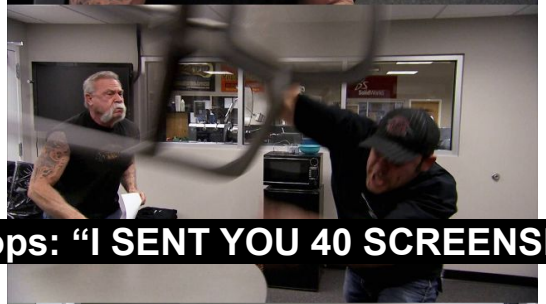
   To provide proof of this during an audit?

Now During an Audit











The solution is one OpenTofu IAC repo

01

Audit logging

pgaudit loaded and configured through the DB parameter group

02

Roles and grants

Least-privilege roles declared with the PostgreSQL provider

03

The pipeline is the evidence for audit

Pull request → Plan → Review → Apply

Biggest win : audit logging changes go through a PR

Loading pgaudit and choosing what gets logged are PR's

Turning logging off is also a PR that someone has to approve.

Easy to share during an Audit

```
resource "aws_db_parameter_group" "pg_audit"
{ name   = "pg16-audit"
  family = "postgres16"

  parameter {
    name         = "shared_preload_libraries"
    value        = "pgaudit"
    apply_method = "pending-reboot"
  }
  parameter {
    name  = "pgaudit.log"
    value = "ddl,role,write"
  }
  parameter {
    name  = "pgaudit.role"
    value = "rds_pgaudit"  # object audit
  }
}
```

Next win: every role and grant is code


```hcl
resource "postgresql_role" "app_readonly" {
  name  = "app_readonly"
  login = false
}

resource "postgresql_grant" "app_readonly" {
  database    = "payments"
  role        = postgresql_role.app_readonly.name
  schema      = "public"
  object_type = "table"
  privileges  = ["SELECT"]
}

# reads on card data get audited
resource "postgresql_grant" "audit_cards" {
  database    = "payments"
  role        = "rds_pgaudit"
  schema      = "public"
  object_type = "table"
  objects     = ["cards"]
  privileges  = ["SELECT"]
}
```

1

Access is a PR

Granting and revoking both need a reviewer.

2

Least privilege by default

Roles are NOLOGIN groups; people and apps inherit.

3

Object audit as code

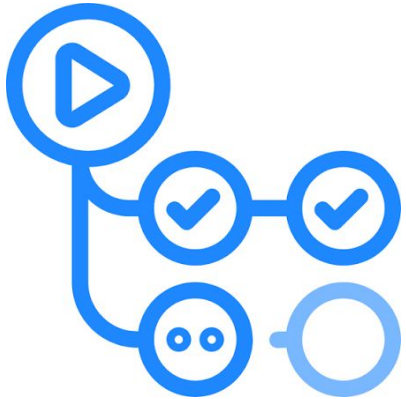
Grant the audit role SELECT, and reads get logged.

Repo Module Structure


```
db-compliance/
├── modules/
│   ├── pg-audit/          # params
│   ├── pg-roles/          # roles+grants
│   └── pg-obj-audit/      # audited tables
├── envs/
│   ├── staging/           # tfvars only
│   └── prod/
└── .github/workflows/
    ├── plan.yml           # every PR
    └── drift.yml          # nightly
```

The workflow ✨✨✨

1. **Pull request**
   Change proposed in Git
2. **`tofu plan`**
   CI posts the exact diff
3. **Review**
   A second person approves
4. **`tofu apply`**
   Only CI applies to prod





GitLab



argo

Two patterns from our experience

CRON DRIFT CHECK

To catch drift automatically on a schedule

```
$ tofu plan -detailed-exitcode
0  no changes
1  error
2  drift → alert
```

HOW WE RUN IT

We use Atlantis

- ✓ Plans posted on every PR
- ✓ No apply without an approval
- ✓ Runs inside our network
- ✓ OpenTofu via `--tf-distribution`

```
repos:
  - id: /.*/
    apply_requirements:
      [approved, mergeable]
```

Mapping to PCI DSS Requirement 10

Req.	What the logs must capture	Our evidence
`10.2.1.1`	Individual user access to cardholder data	pgaudit object audit on card tables
`10.2.1.2`	All actions by administrators	pgaudit ddl + role classes
`10.2.1.5`	Changes to accounts and credentials	pgaudit role class + PR history of every role
`10.2.1.6`	Changes to audit logging itself	Settings in code + nightly drift check

What doesn’t belong in IaC

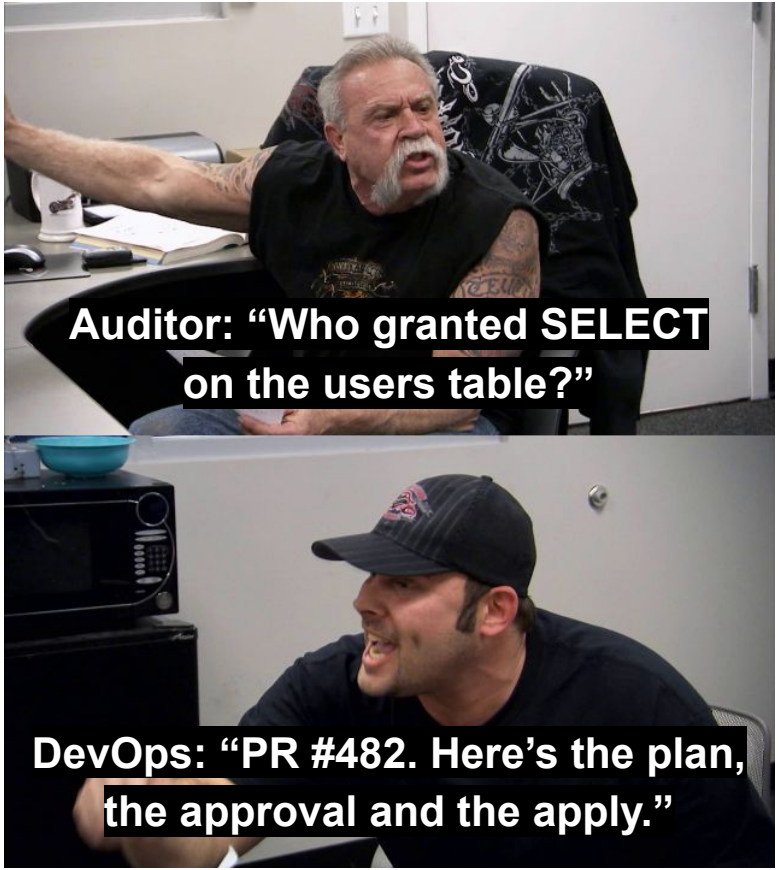
In OpenTofu

- ✓ pgaudit settings
- ✓ Roles and grants
- ✓ Audited tables (object audit)
- ✓ Log export and retention config

Keep out

- ✕ Data and schema migrations
- ✕ Break-glass access (time-boxed, logged)
- ✕ The audit logs themselves
- ✕ Individual human passwords

With OpenTofu during an Audit





Quick Takeaways

1

Put audit config and access in code

pgaudit settings, roles, grants and audited tables.

2

Make the workflow your evidence

PR history + plan + nightly drift check

3

Know where IaC stops

Data, migrations, break-glass access and the logs.



OpenTofu Day
NORTH AMERICA

Thank you!

@skysingh04



linkedin.in/skysingh04