PLATFORM · TEMPLATE

Detecting Infrastructure Drift *Without* Drowning in Alerts

KAANSYSTEMS.COM/LIBRARY/INFRASTRUCTURE-DRIFT-DETECTION · MAY 30, 2026

ABOUT THIS TEMPLATE

How to tier infrastructure drift by risk so you fix what matters and ignore what doesn't, instead of paging engineers about EBS tag mismatches.

THE TEMPLATE

Below is the GitHub Actions workflow that runs the hourly Tier 1 check against an estate of AWS accounts. It uses an OIDC-federated role to assume into each account, runs `terraform plan -detailed-exitcode`, and routes anything matching the Tier 1 pattern to a webhook.

```yaml
# .github/workflows/drift-check.yml
name: Infrastructure drift · Tier 1
on:
  schedule:
    - cron: '0 * * * *'   # hourly
  workflow_dispatch:

permissions:
  id-token: write          # for OIDC role assumption
  contents: read

jobs:
  detect:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        account: [prod-us-east-1, prod-us-west-2, log-archive]
      fail-fast: false
    steps:
      - uses: actions/checkout@v4

      - uses: aws-actions/configure-aws-credentials@v4
        with:
          role-to-assume: arn:aws:iam::${{ secrets.ACCOUNT_ID }}:role/DriftDetector
          aws-region: us-east-1

      - uses: hashicorp/setup-terraform@v3
        with: { terraform_version: 1.9.x }

      - name: terraform plan
        id: plan
        working-directory: ./environments/${{ matrix.account }}
        run: |
          terraform init -input=false
          terraform plan -input=false -lock=false \
            -out=plan.bin -detailed-exitcode || exit_code=$?
          terraform show -json plan.bin > plan.json
          echo "exit_code=${exit_code:-0}" >> $GITHUB_OUTPUT

      - name: Classify and route
        if: steps.plan.outputs.exit_code == '2'   # changes detected
        run: |
          python3 .ci/classify-drift.py plan.json > tier1.json
          if [ -s tier1.json ]; then
            curl -X POST -H 'Content-Type: application/json' \
              --data @tier1.json \
              ${{ secrets.PAGERDUTY_WEBHOOK }}
          fi
```

The `classify-drift.py` script is a 30-line filter over the JSON plan that emits Tier 1 entries only:

```
import json, sys, re

TIER1_RESOURCES = (
    r'^aws_iam_',
    r'^aws_security_group',
    r'^aws_kms_key',
    r'^aws_cloudtrail',
    r'^aws_s3_bucket_policy$',
    r'^aws_s3_bucket_public_access_block$',
    r'^aws_backup_(plan|vault)',
)

plan = json.load(open(sys.argv[1]))
tier1 = []

for change in plan.get('resource_changes', []):
    if change['change']['actions'] == ['no-op']:
        continue
    if any(re.match(p, change['type']) for p in TIER1_RESOURCES):
        tier1.append({
            'resource': change['address'],
            'type': change['type'],
            'actions': change['change']['actions'],
        })

if tier1:
    json.dump({'alert': 'Tier-1 drift', 'changes': tier1}, sys.stdout)
```

HOW TO USE IT

Run the digest review on Mondays. Anything in Tier 2 that's been there three consecutive weeks becomes a Tier 1 candidate, or gets accepted and moved to Tier 3 with a comment explaining why. The tiers aren't static. They're the team's working classification of what matters, and they should evolve as you learn what your environment actually does.

The number to watch isn't drift incidents. It's how often Tier 1 fires. More than once a week on average and your change-control process has a gap. Less than once a quarter and the classifier is too lenient. Someone is changing security-relevant settings without it surfacing.