IaC Meets Reality

Engineering for the Cloud You Actually Have

SANS

CloudSecNext SUMMIT 2025

sans.org/CloudSecNextSummit

unicrons.cloud

Our Context

- Small team
- Different environments
  - Different teams
  - Different tech stacks
  - Different time zones
- Compliance-heavy environment
  - SOC2, HIPAA, SOX, PCI, ...
- Embracing Infra as Code

- Small team
- Different environments
  - Different teams
  - Different tech stacks
  - Different time zones
- Compliance-heavy environment
  - SOC2, HIPAA, SOX, PCI, ...
- **Embracing Infra as Code**

$ aws sts get-caller-identity --profile unicrons_cloud

Samuel

Cloud Security Engineer @ flywire

sbldevnet

Andoni

Cloud Security Engineer @ PROWLER

andoniaf

Is IaC the path to go?

Embracing IaC

IaC Benefits:

- Version Control & Auditability
- Consistency & Standardization
- Automation & Speed
- Documentation That Never Lies
- Cost & Resource Management

The Pain of ClickOps

ClickOps in highly regulated environments

- **Audit trails?** What audit trails?
- **Compliance?** "Trust me, I configured it right"
- **Security consistency?**
- **Scale?** Good luck with 50+ cloud accounts

Auditors don't accept "I clicked it in the console" as documentation


ClickOps
IaC
Compliance
Security
Easy audits
Less toil


IaC
ClickOps
Us (and probably you too)

The Reality

Not Everything can be IaC

Not Everything can be IaC

- Legacy systems
- Unsupported resources*
- Manual changes
  - Emergency fixes
  - Tests

Not Everything can be IaC

- Legacy systems
- Unsupported resources*
- Manual changes
  - Emergency fixes
  - Tests
- Drift

The Reality

IaC Maintenance

IaC Maintenance: Upgrades

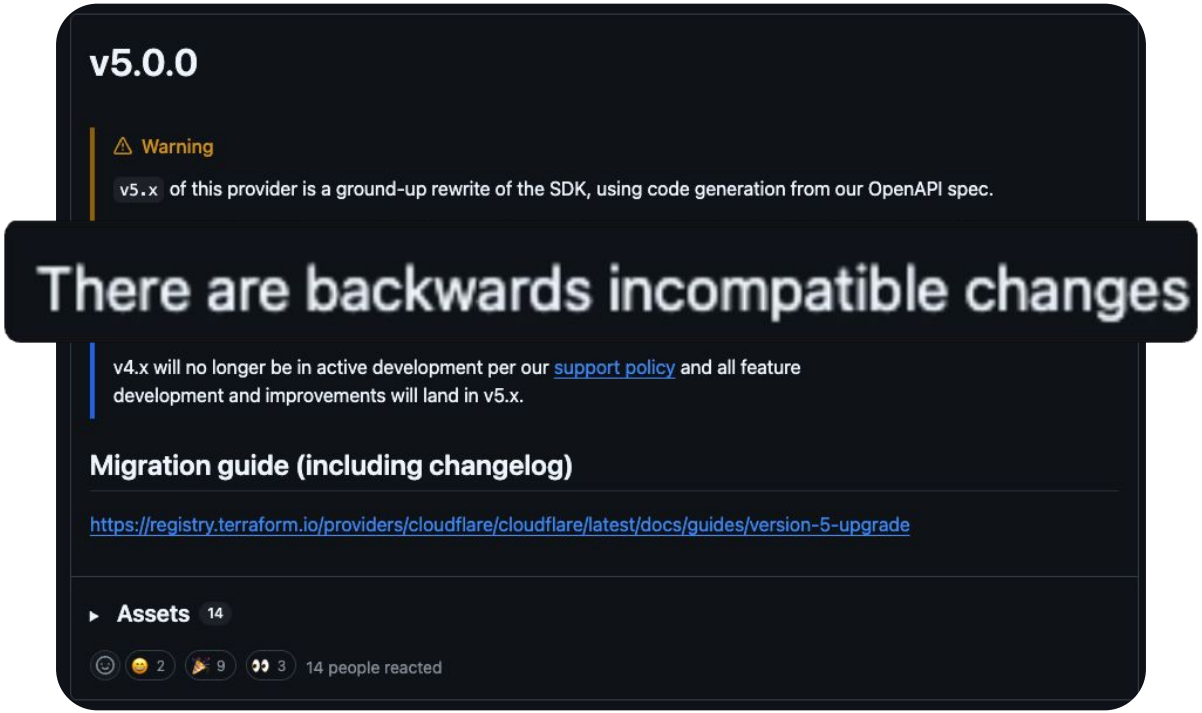
Upgrade Terraform 0.11 to 0.12 - by Kraig McFadden

Having a hard time getting off **Terraform 0.11**? Getting to **0.12** can be a challenge, but here are a few tips to get you on your way.

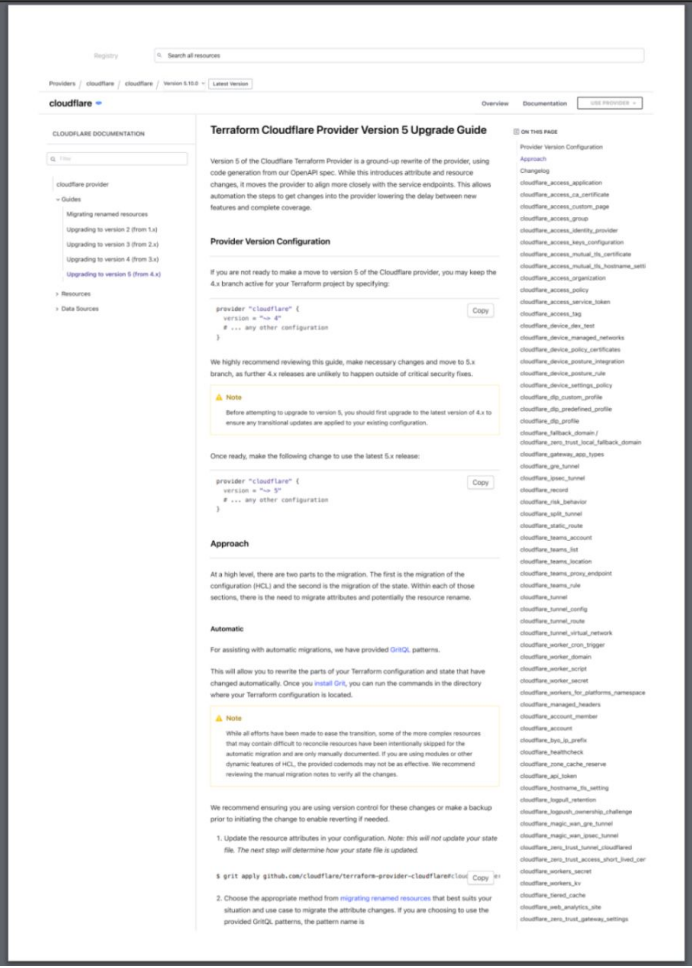
Upgrade Terraform : Upgrade Terraform code from 0.11 to 0.12

Upgrading **Terraform** code from version **0.11 to 0.12** can be a daunting task for DevOps teams and cloud engineers looking to leverage the latest features and improvements in **Terraform**. In this comprehensive guide, we will walk you through the steps to update your **Terraform** code from version...

IaC Maintenance: Provider Upgrades



registry.terraform.io/providers/cloudflare/cloudflare/latest/docs/guides/version-5-upgrade



Terraform Cloudflare Provider Version 5 Upgrade Guide

Version 5 of the Cloudflare Terraform Provider is a ground-up rewrite of the provider, using code generation from our OpenAPI spec. While this introduces attribute and resource changes, it moves the provider to align more closely with the service endpoints. This allows automation the steps to get changes into the provider lowering the delay between new features and complete coverage.

Provider Version Configuration

If you are not ready to make a move to version 5 of the Cloudflare provider, you may keep the 4.x branch active for your Terraform project by specifying:

```
provider "cloudflare" {
  version = "~> 4"
  # ... any other configuration
}
```

We highly recommend reviewing this guide, make necessary changes and move to 5.x branch, as further 4.x releases are unlikely to happen outside of critical security fixes.

> Note
> Before attempting to upgrade to version 5, you should first upgrade to the latest version of 4.x to ensure any transitional updates are applied to your existing configuration.

Once ready, make the following change to use the latest 5.x release:

```
provider "cloudflare" {
  version = "~> 5"
  # ... any other configuration
}
```

Approach

At a high level, there are two parts to the migration. The first is the migration of the configuration (HCL) and the second is the migration of the state. Within each of those sections, there is the need to migrate attributes and potentially the resource rename.

Automatic

For assisting with automatic migrations, we have provided GritQL patterns.

This will allow you to rewrite the parts of your Terraform configuration and state that have changed automatically. Once you install Grit, you can run the commands in the directory where your Terraform configuration is located.

> Note
> While all efforts have been made to ease the transition, some of the more complex resources that may contain difficult to reconcile resources have been intentionally skipped for the automatic migration and are only manually documented. If you are using modules or other dynamic features of HCL, the provided codemods may not be as effective. We recommend reviewing the manual migration notes to verify all the changes.

We recommend ensuring you are using version control for these changes or make a backup prior to initiating the change to enable reverting if needed.

1. Update the resource attributes in your configuration. *Note: this will not update your state file. The next step will determine how your state file is updated.*

```
$ grit apply github.com/cloudflare/terraform-provider-cloudflare#clou
```

2. Choose the appropriate method from migrating renamed resources that best suits your situation and use case to migrate the attribute changes. If you are choosing to use the provided GritQL patterns, the pattern name is

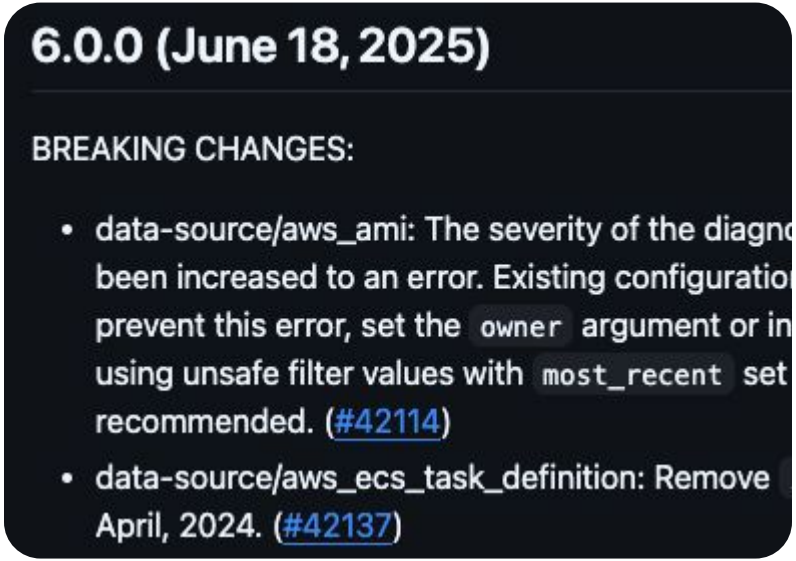
Print 22 pages

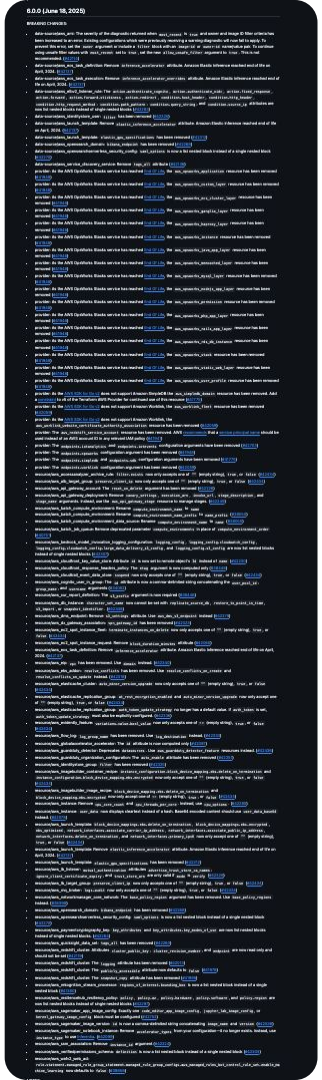
Destination Save as PDF

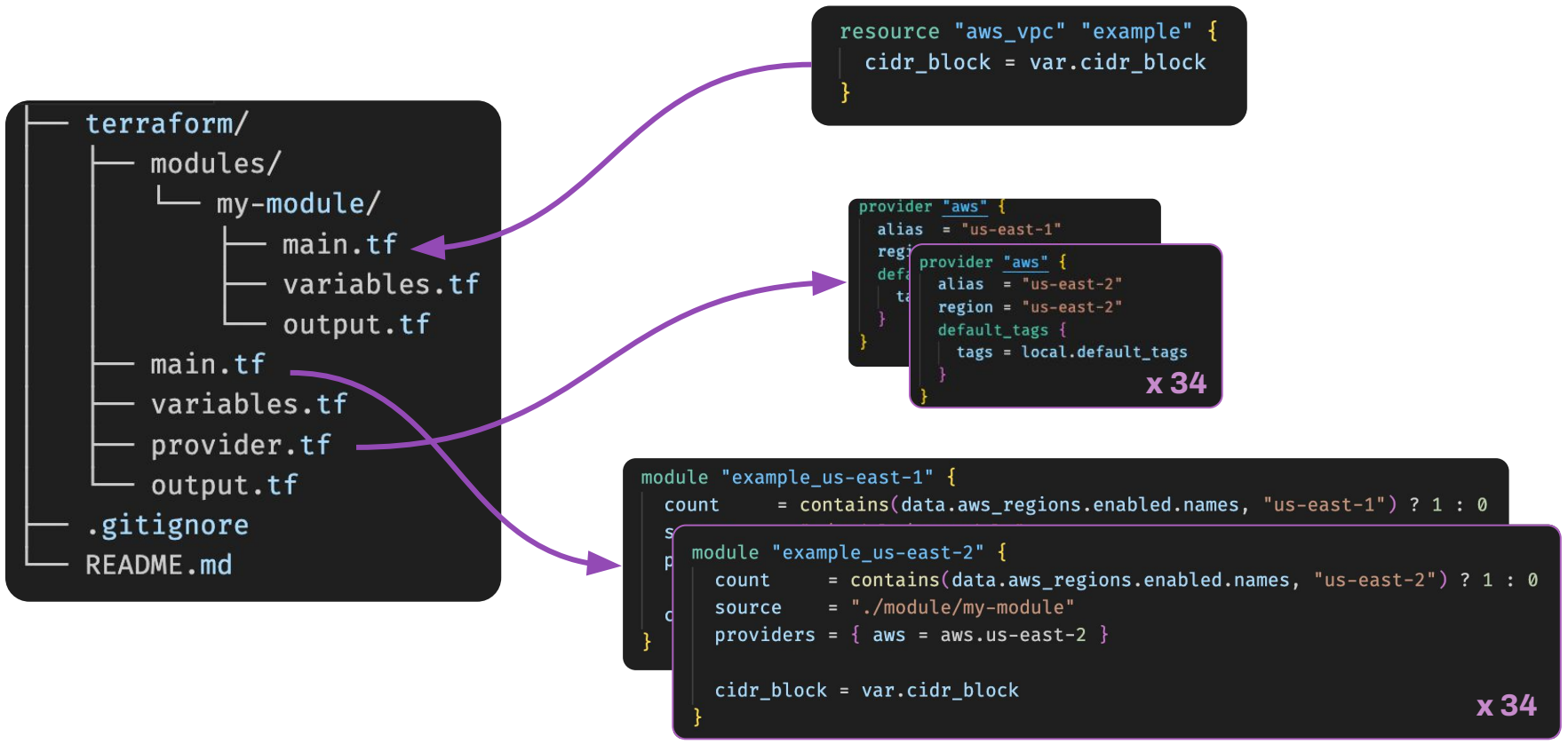
22 pages

Cancel Save

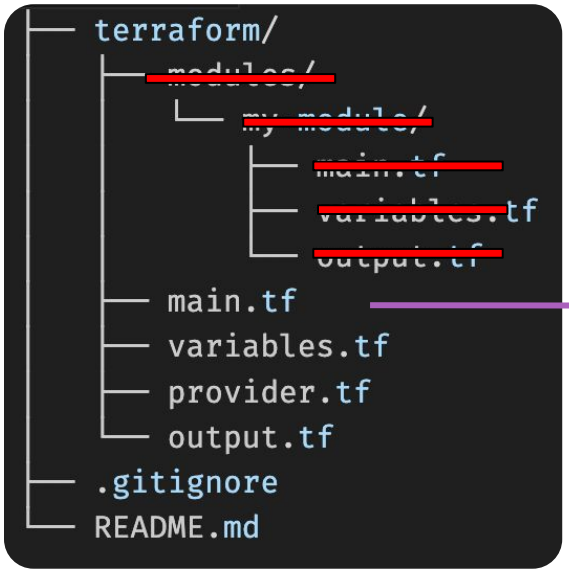
IaC Maintenance: Provider Upgrades







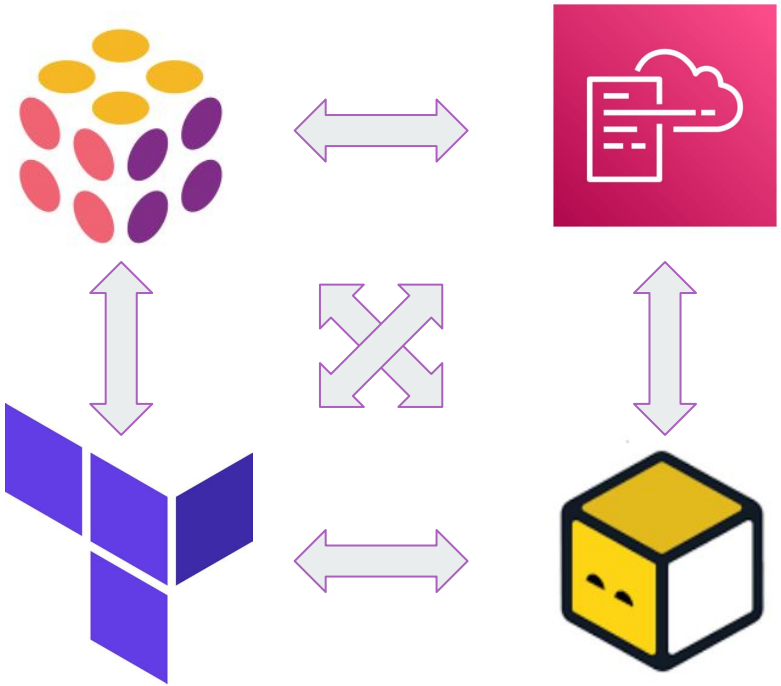
terraform-provider-aws < 6.0




```hcl
resource "aws_vpc" "example" {
  for_each = toset(data.aws_regions.enabled.names)
  region   = each.value

  cidr_block = var.cidr_block
}
```

terraform-provider-aws ≥ 6.0

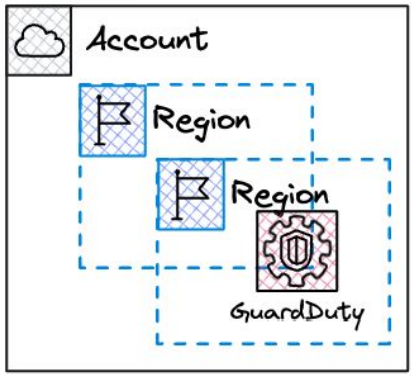
IaC Maintenance: Migrations

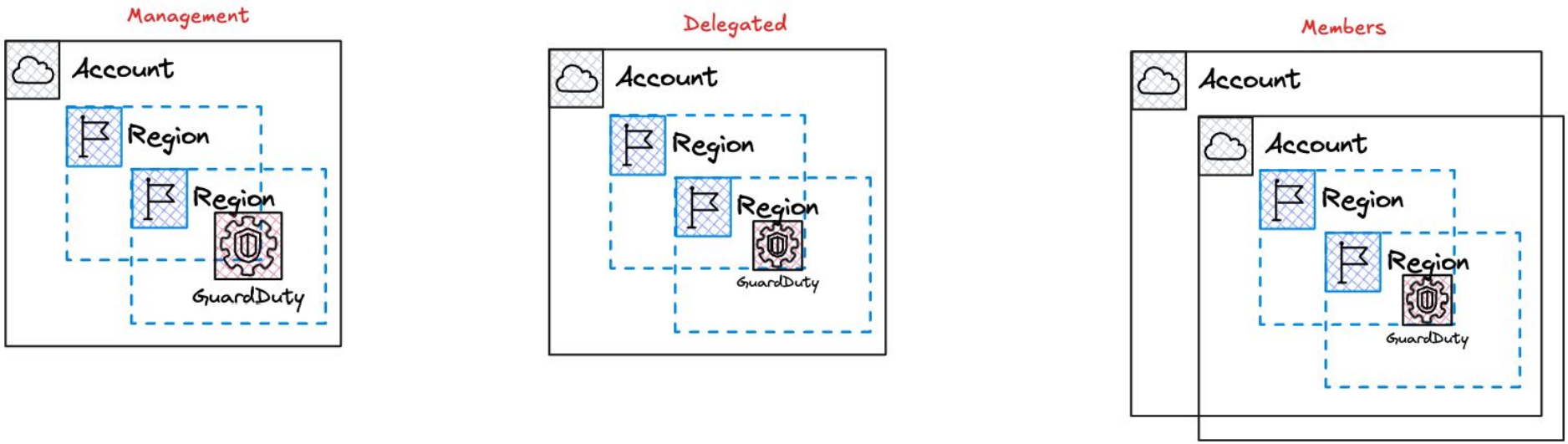


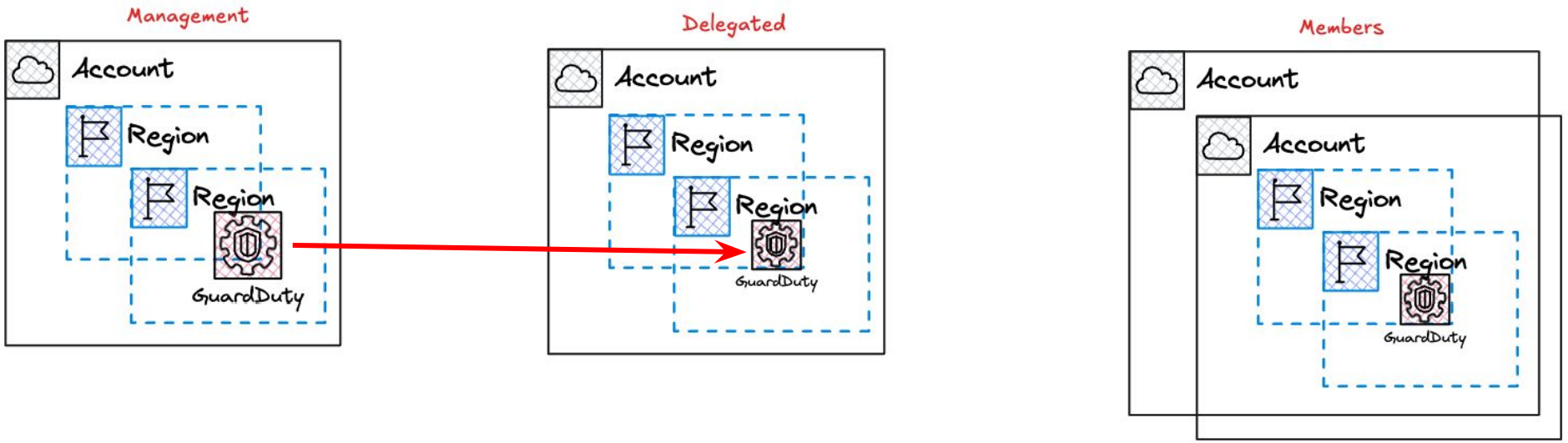
The Reality

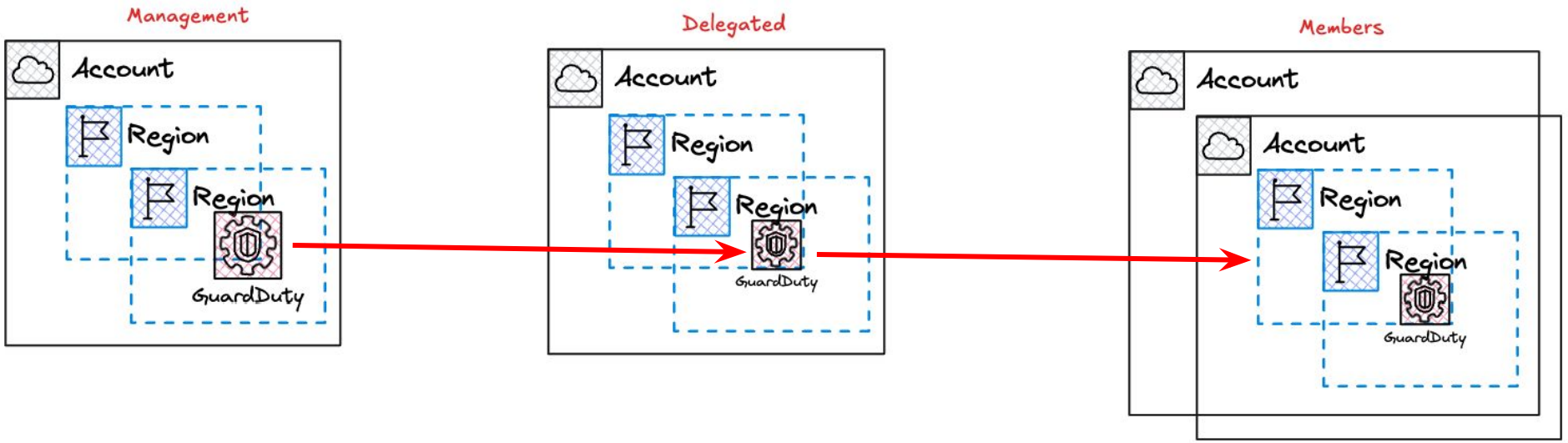
Implementation Complexity


GuardDuty


Account
Region
Region
GuardDuty


Management
Account
Region
Region
GuardDuty
Delegated
Account
Region
Region
GuardDuty
Members
Account
Account
Region
Region
GuardDuty


Management
Account
Region
Region
GuardDuty
Delegated
Account
Region
Region
GuardDuty
Members
Account
Account
Region
Region
GuardDuty


Management
Account
Region
Region
GuardDuty
Delegated
Account
Region
Region
GuardDuty
Members
Account
Account
Region
Region
GuardDuty

```
resource "aws_guardduty_detector" "example" {
  enable = true
}
```

Management Account

```
resource "aws_guardduty_organization_admin_account" "example" {
  admin_account_id = var.delegated_account_id
}
```

```
resource "aws_guardduty_detector" "example" {
  enable = true
}
```

Delegated Account

Management Account

```
resource "aws_guardduty_organization_admin_account" "example" {
  admin_account_id = var.delegated_account_id
```

```
Error: creating GuardDuty Detector: operation error GuardDuty: CreateDetector, https response error StatusCode: 400,
BadRequestException: The request is rejected because a detector already exists for the current account.

  with aws_guardduty_detector.example,
  on delegated.tf line 3, in resource "aws_guardduty_detector" "example":
   3: resource "aws_guardduty_detector" "example" {
```

```
}
```

Delegated Account

Management Account

```
resource "aws_guardduty_organization_admin_account" "example" {
  admin_account_id = var.delegated_account_id
}
```

```
data "aws_guardduty_detector" "example" {}
```

Delegated Account

Management Account

```
data "aws_regions" "current" {}

resource "aws_guardduty_organization_admin_account" "example" {
  for_each = data.aws_regions.current.names
  region   = each.value

  admin_account_id = var.delegated_account_id
}
```

```
data "aws_regions" "current" {}

data "aws_guardduty_detector" "example" {
  for_each = data.aws_regions.current.names
  region   = each.value
}
```

Delegated Account

```
data "aws_regions" "current" {}

data "aws_guardduty_detector" "example" {
  for_each = data.aws_regions.current.names
  region   = each.value
}
```

```
data "aws_regions" "current" {}

data "aws_guardduty_detector" "example" {
  for_each = data.aws_regions.current.names
  region   = each.value
}
```

```
resource "terraform_data" "aws_guardduty_finding_publishing_frequency" {
  for_each = data.aws_regions.current.names
  triggers_replace = [
    data.aws_guardduty_detector.example.id
  ]

  provisioner "local-exec" {
    command = "aws guardduty update-detector --detector-id ${data.aws_guardduty_detector.example[each.value].id} --finding-publishing-frequency ONE_HOUR --region ${each.value}"
  }
}
```

```
data "aws_regions" "current" {}

data "aws_guardduty_detector" "example" {
  for_each = data.aws_regions.current.names
  region   = each.value
}
```

```
resource "aws_guardduty_detector" "example" {
  for_each = data.aws_regions.current.names
  region   = each.value

  enable = true
  finding_publishing_frequency = "ONE_HOUR"
}

import {
  for_each = data.aws_regions.current.names

  to = aws_guardduty_detector.example[each.value]
  id = "${data.aws_guardduty_detector.example[each.value].id}@${each.value}"
}
```

```
data "aws_regions" "current" {}

data "aws_guardduty
  for_each = data.
  region   = each.
}
```

```
resource "aws_guardduty_organization_configuration" "example" {
  for_each = data.aws_regions.current.names
  region   = each.value

  detector_id = data.aws_guardduty_detector.example[each.value].id
  auto_enable_organization_members = "ALL"
}
```

```
resource "aws_guardduty_dete
  for_each = data.aws_region
  region   = each.value

  enable = true
  finding_publishing_frequency = "ONE_HOUR"
}

import {
  for_each = data.aws_regions.current.names

  to = aws_guardduty_detector.example[each.value]
  id = "${data.aws_guardduty_detector.example[each.value].id}@${each.value}"
}
```

```
locals {
  current_region       = data.aws_region.this.region
  default_regions      = data.aws_regions.current.names
  member_accounts      = data.aws_organizations_organization.this.accounts
  delegated_account_id = data.aws_caller_identity.this.account_id
```

```
data "aws_regions" "current" {}

data "aws_guarddut
  for_each = data.
  region   = each.
}
```

```
resource "aws_guardduty_organization_configuration" "example" {
  for_each = data.aws_regions.current.names
  region   = each.value

  detector_id = data.aws_guardduty_detector.example[each.value].id
  auto_enable_organization_members = "ALL"
}
```

```
resource "aws_guardduty_dete
  for_each = data.aws_region
  region   = each.value

  enable = true
  finding_publishing_frequency = "ONE_HOUR"
}

import {
  for_each = data.aws_regions.current.names

  to = aws_guardduty_detector.example[each.value]
  id = "${data.aws_guardduty_detector.example[each.value].id}@${each.value}"
}
```

```
locals {
  current_region       = data.aws_region.this.region
  default_regions      = data.aws_regions.current.names
  member_accounts      = data.aws_organizations_organization.this.accounts
  delegated_account_id = data.aws_caller_identity.this.account_id

  active_members = { for account in local.member_accounts : account.id => account
    if account.id != local.delegated_account_id && account.status == "ACTIVE"
  }
```

```
data "aws_regions" "current" {}

data "aws_guardduty
  for_each = data.
  region   = each.
}
```

```
resource "aws_guardduty_organization_configuration" "example" {
  for_each = data.aws_regions.current.names
  region   = each.value

  detector_id = data.aws_guardduty_detector.example[each.value].id
  auto_enable_organization_members = "ALL"
}
```

```
resource "aws_guardduty_dete
  for_each = data.aws_region
  region   = each.value

  enable = true
  finding_publishing_frequency = "ONE_HOUR"
}

import {
  for_each = data.aws_regions.current.names

  to = aws_guardduty_detector.example[each.value]
  id = "${data.aws_guardduty_detector.example[each.value].id}@${each.value}"
}
```

```
locals {
  current_region       = data.aws_region.this.region
  default_regions      = data.aws_regions.current.names
  member_accounts      = data.aws_organizations_organization.this.accounts
  delegated_account_id = data.aws_caller_identity.this.account_id

  active_members = { for account in local.member_accounts : account.id => account
    if account.id != local.delegated_account_id && account.status == "ACTIVE"
  }

  default_region_members = merge([
    for region in local.default_regions : {
      for account_id, account in local.active_members : "${region}:${account_id}" => {
        region     = region
        account_id = account_id
        account    = account
      }
    }
  ]... )
```

```
data "aws_reg

data "aws_gua
  for_each =
  region   =
}
```

```
{


e].id
```

```
resource "aws_guardduty_dete
  for_each = data.aws_region
  region   = each.value

  enable = true
  finding_publishing_frequency = "ONE_HOUR"
}

import {
  for_each = data.aws_regions.current.names

  to = aws_guardduty_detector.example[each.value]
  id = "${data.aws_guardduty_detector.example[each.value].id}@${each.value}"
}
```

```
locals {
  current_region       = data.aws_region.this.region
  default_regions      = data.aws_regions.current.names
  member_accounts      = data.aws_organizations_organization.this.accounts
  delegated_account_id = data.aws_caller_identity.this.account_id

  active_members = { for account in local.member_accounts : account.id => account
    if account.id != local.delegated_account_id && account.status == "ACTIVE"
  }

  default_region_members = merge([
    for region in local.default_regions : {
      for account_id, account in local.active_members : "${region}:${account_id}" => {
        region     = region
        account_id = account_id
        account    = account
      }
    }
  ]... )
```

```
data "aws_reg

data "aws_gua
  for_each =
  region   =
}
```

```
{

e].id
```

```
resource "aws_guardduty
  for_each = data.aws_
  region   = each.val

  enable = true
  finding_publishing_

import {
  for_each = data.aws_

  to = aws_guardduty_
  id = "${data.aws_gu
}
```

```
  opt_in_region_members = merge([
    for account_id, regions in var.aws_accounts_opt_in_region : {
      for region in regions : "${region}:${account_id}" => {
        region     = region
        account_id = account_id
        account    = local.active_members[account_id]
      }
      if contains(keys(local.active_members), account_id)
    }
  ]... )
```

```
locals {
  current_region       = data.aws_region.this.region
  default_regions      = data.aws_regions.current.names
  member_accounts      = data.aws_organizations_organization.this.accounts
  delegated_account_id = data.aws_caller_identity.this.account_id

  active_members = { for account in local.member_accounts : account.id => account
    if account.id != local.delegated_account_id && account.status == "ACTIVE"
  }

  default_region_members = merge([
    for region in local.default_regions : {
      for account_id, account in local.active_members : "${region}:${account_id}" => {
        region     = region
        account_id = account_id
        account    = account
      }
    }
  ]...)

  opt_in_region_members = merge([
    for account_id, regions in var.aws_accounts_opt_in_region : {
      for region in regions : "${region}:${account_id}" => {
        region     = region
        account_id = account_id
        account    = local.active_members[account_id]
      }
      if contains(keys(local.active_members), account_id)
    }
  ]...)

  members = merge(local.opt_in_region_members, local.default_region_members)
}
```

```
data "aws_reg

data "aws_gua
  for_each = (
  region   = (
}
```

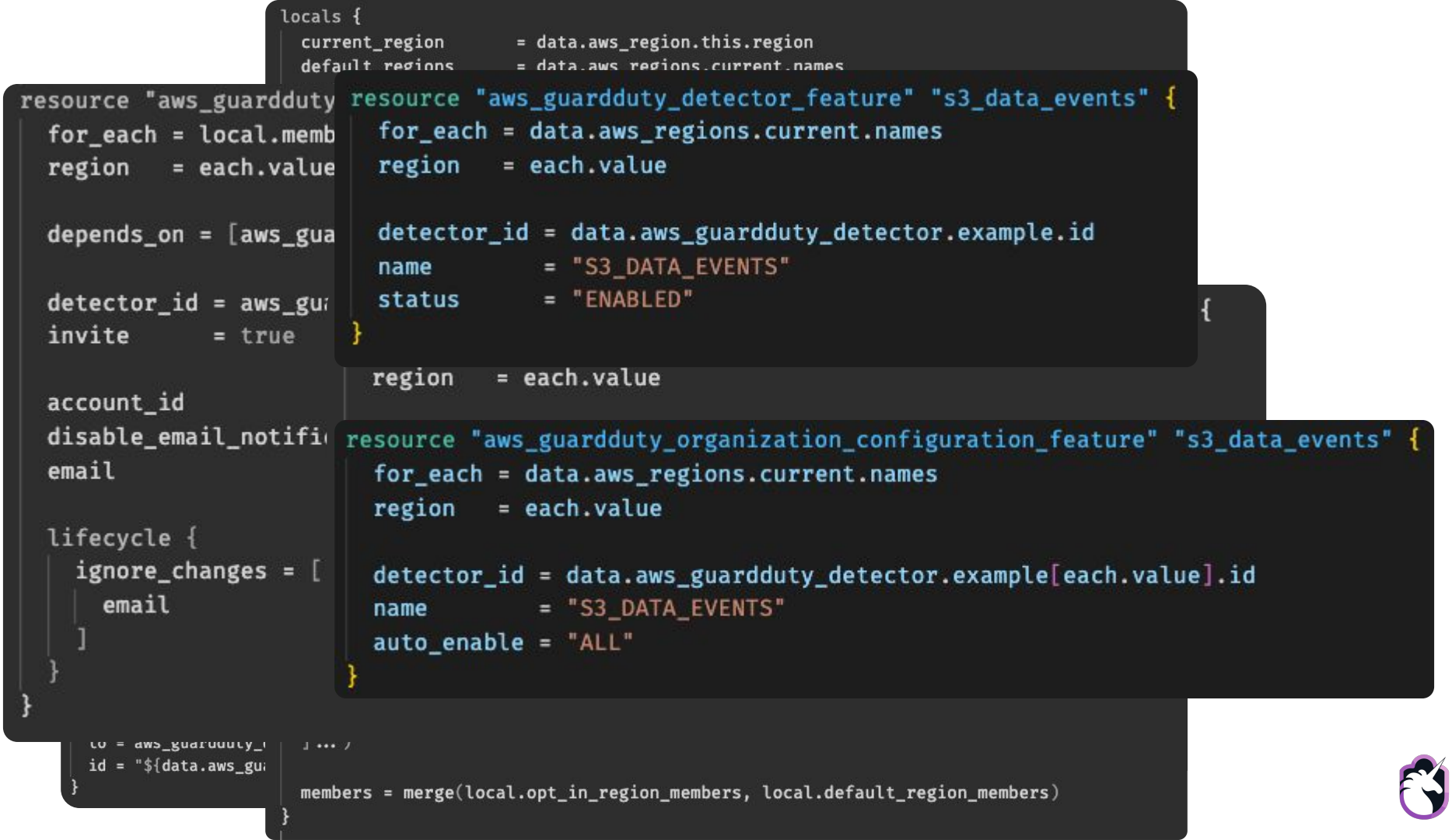
```
resource "aws_guardduty
  for_each = data.aws_
  region   = each.val

  enable = true
  finding_publishing_

import {
  for_each = data.aws_

  to = aws_guardduty_(
  id = "${data.aws_gu
}
```

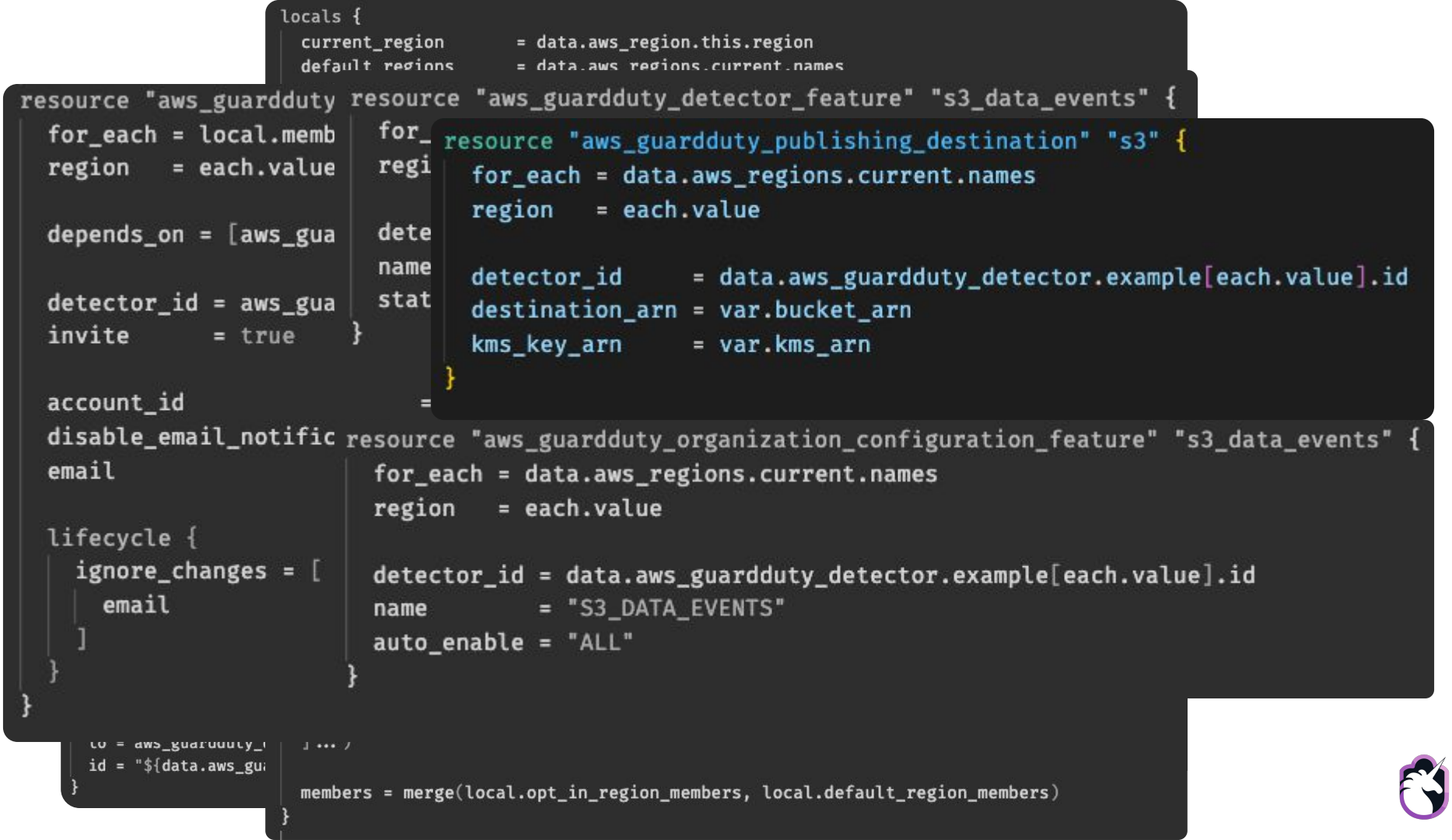
```
{

e].id
```

```
resource "aws_guardduty_member" "members" {
  for_each = local.members
  region   = each.value.region

  depends_on = [aws_guardduty_organization_configuration.example]

  detector_id = aws_guardduty_detector.example[each.value.region].id
  invite      = true

  account_id                 = each.value.account_id
  disable_email_notification = true
  email                      = each.value.account.email

  lifecycle {
    ignore_changes = [
      email
    ]
  }
}
```

```
locals {
  current_region       = data.aws_region.this.region
  default_regions      = data.aws_regions.current.names
```

```
  members = merge(local.opt_in_region_members, local.default_region_members)
}
```

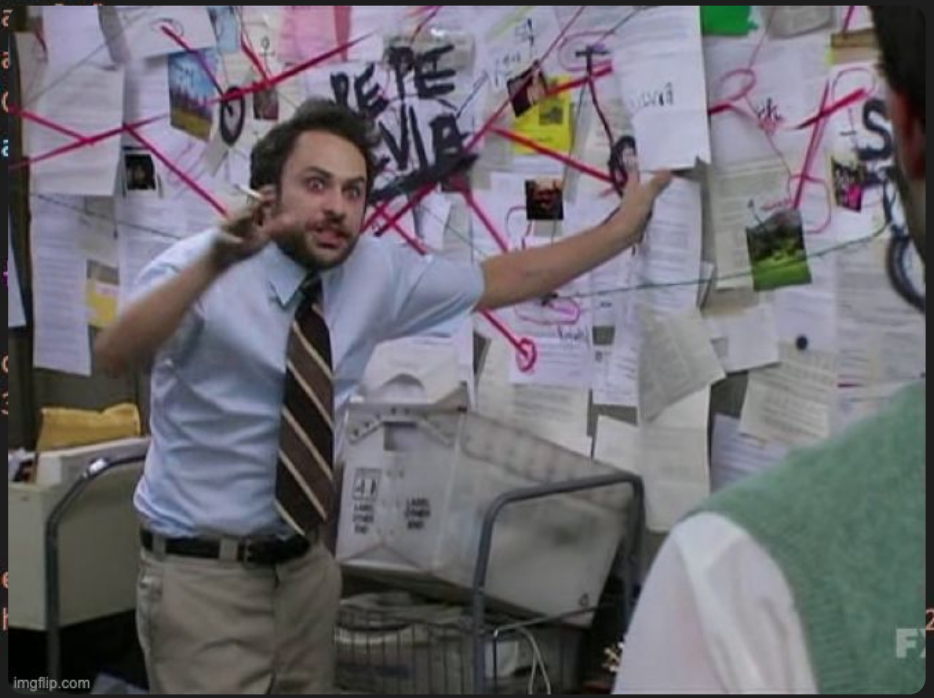

locals {
current_region = data.aws_region.this.region
default_regions = data.aws_regions.current.names
resource "aws_guardduty
for_each = local.memb
region = each.value
depends_on = [aws_gua
detector_id = aws_gu
invite = true
account_id
disable_email_notifi
email
lifecycle {
ignore_changes = [
email
]
}
}
resource "aws_guardduty_detector_feature" "s3_data_events" {
for_each = data.aws_regions.current.names
region = each.value
detector_id = data.aws_guardduty_detector.example.id
name = "S3_DATA_EVENTS"
status = "ENABLED"
}
region = each.value
resource "aws_guardduty_organization_configuration_feature" "s3_data_events" {
for_each = data.aws_regions.current.names
region = each.value
detector_id = data.aws_guardduty_detector.example[each.value].id
name = "S3_DATA_EVENTS"
auto_enable = "ALL"
}
id = "${data.aws_gu
}
members = merge(local.opt_in_region_members, local.default_region_members)
}


locals {
current_region = data.aws_region.this.region
default_regions = data.aws_regions.current.names
resource "aws_guardduty
for_each = local.memb
region = each.value
depends_on = [aws_gua
detector_id = aws_gua
invite = true
account_id =
disable_email_notific
email
lifecycle {
ignore_changes = [
email
]
}
}
resource "aws_guardduty_detector_feature" "s3_data_events" {
for_
regi
dete
name
stat
}
resource "aws_guardduty_publishing_destination" "s3" {
for_each = data.aws_regions.current.names
region = each.value
detector_id = data.aws_guardduty_detector.example[each.value].id
destination_arn = var.bucket_arn
kms_key_arn = var.kms_arn
}
resource "aws_guardduty_organization_configuration_feature" "s3_data_events" {
for_each = data.aws_regions.current.names
region = each.value
detector_id = data.aws_guardduty_detector.example[each.value].id
name = "S3_DATA_EVENTS"
auto_enable = "ALL"
}
id = "${data.aws_gu
}
members = merge(local.opt_in_region_members, local.default_region_members)
}

```hcl
resource "aws_guardduty_filter" "example" {
  for_each = data.aws_regions.current.names
  region   = each.value

  name        = "example"
  description = "example"
  action      = "ARCHIVE"
  detector_id = data.aws_guardduty_detector.example[each.value].id
  rank        = 1

  finding_criteria {
    criterion {
      field  = "accountId"
      equals = ["123456789012"]
    }

    criterion {
      field  = "type"
      equals = ["Behavior:EC2/TrafficVolumeUnusual", "CryptoCurrency:EC2/BitcoinTool.B!DNS"]
    }
  }
}
```

```
locals {
  current_region       = data.aws_region.this.region
```

```
resource "aws_guardduty_filter" "example" {
  for_each = data.aws_regions.current.names
  region   = each.value

  name        = "exa
  description = "exa
  action      = "AR
  detector_id = data
  rank        = 1

  finding_criteria {
    criterion {
      field  = "acc
      equals = ["12
    }

    criterion {
      field  = "type
      equals = ["Be
    }
  }
}
```




```
2/BitcoinTool.B!DNS"]
```

```
}
```

Bugs + Few people using it = Learn Go?

AWS IAM Access Analyzer de

Closed

sbldevnet opened on May 11, 2021

Community Note

- Please vote on this issue by adding a request

Created a pull request in hashicorp/terraform-provider-aws that received 21 comments Apr 16

Use trail `arn` instead of `name` to get organization trails from a delegated account

Description Getting the list of existing CloudTrail organization trails from a delegated account only works using the `arn`. This change will use t...

+565 −354 lines changed • 21 comments

Created a pull request in hash

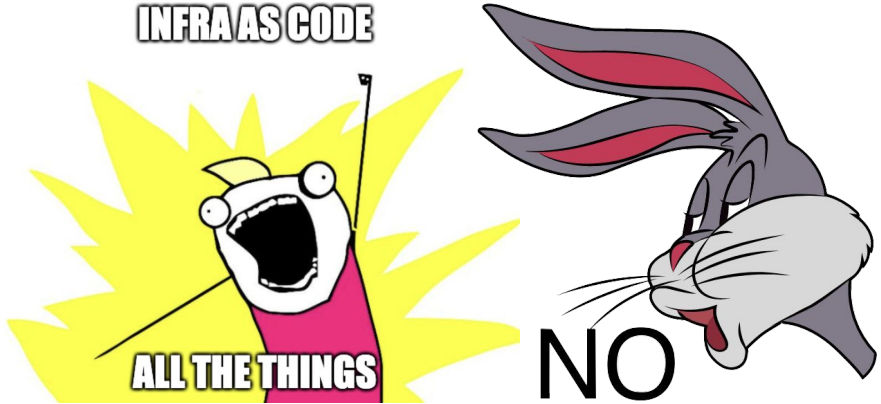
fix: add enabled_standard_arns if service_enabled is true in aws_securityhub_configuration_policy

Description If `enabled_standard_arns` is used, and `service_enabled` is set to `false`, it will be included in the AWS API call. Now it's only added if ...

+13 −10 lines changed • 6 comments

riginal issue to help the

The Hybrid Approach



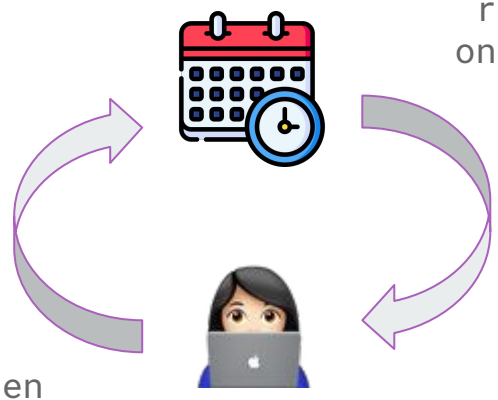
Manual changes, the good way

- Principle Of Least Privilege
  - Oncall
  - Just-In-Time (JIT)

- Think about traceability:
  - Tagging → **Who?**
  - Notifying → **Why?**
  - Audit logs → **What?**

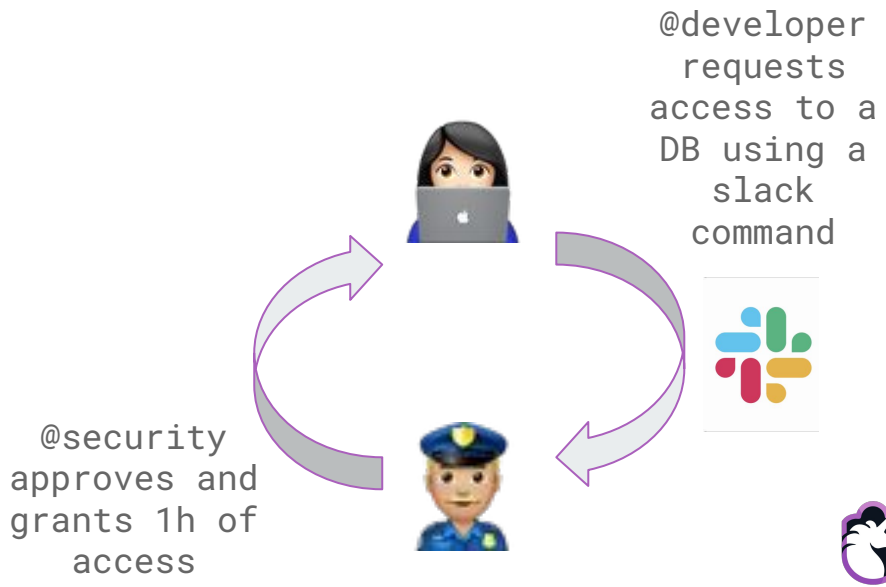
Manual changes, the good way

- **Principle Of Least Privilege**
  - Oncall
  - Just-In-Time (JIT)
- Think about traceability:
  - Tagging → **Who?**
  - Notifying → **Why?**
  - Audit logs → **What?**



Manual changes, the good way

- **Principle Of Least Privilege**
  - Oncall
  - Just-In-Time (JIT)
- Think about traceability:
  - Tagging → **Who?**
  - Notifying → **Why?**
  - Audit logs → **What?**



Manual changes, the good way

- Principle Of Least Privilege
  - Oncall
  - Just-In-Time (JIT)
- **Think about traceability**
  - Tagging → **Who?**
  - Notifying → **Why?**
  - Audit logs → **What?**

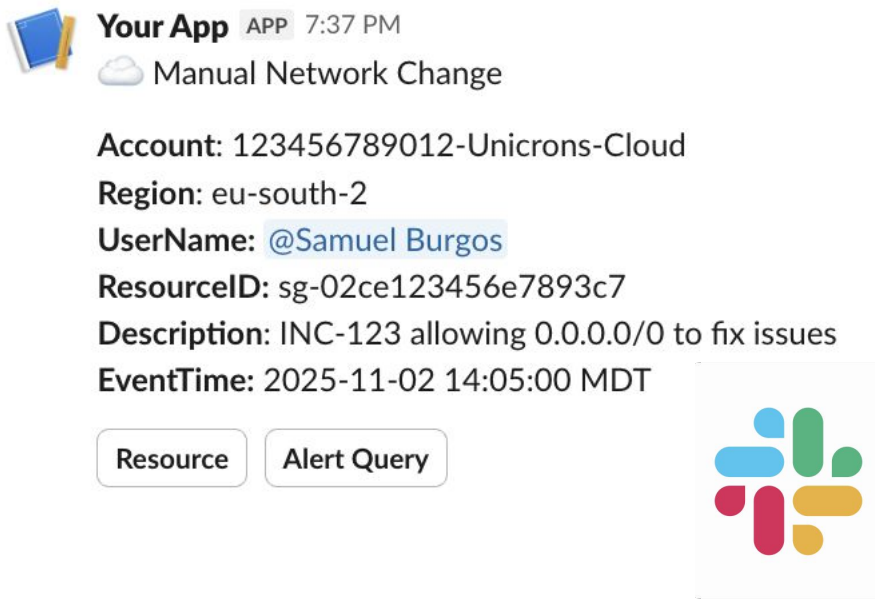
Manual changes, the good way

- Principle Of Least Privilege
  - Oncall
  - Just-In-Time (JIT)

- **Think about traceability**
  - Tagging → **Who?**
  - Notifying → **Why?**
  - Audit logs → **What?**

Manual changes, the good way

- Principle Of Least Privilege
  - Oncall
  - Just-In-Time (JIT)
- **Think about traceability**
  - Tagging → **Who?**
  - Notifying → **Why?**
  - Audit logs → **What?**



Scripts. Isn't this IaC too?

```
az keyvault key rotation-policy update \
  -n <keyName> \
  --vault-name <vaultName> \
  --value <path/to/policy.json>
```

```
aws kms enable-key-rotation \
  --key-id <kms_key_id>
```

Tools. Isn't this IaC too?

README Apache-2.0 license

AWS Root Manager

A CLI tool for easily manage AWS Centralized Root Access.

aws-root-manager Public — Edit Pins — Watch 0

main — 1 Branch — 2 Tags — Go to file t — Add file — Code

sbldevnet [dep][go](deps): Bump the golang-dependencies group with 7 updates (#33) c1e5a37 · 3 hours ago — 63 Commits

.github	[dep][actions](deps): Bump actions/setup-go from 5 to 6	last week
cmd	add recovery command	6 months ago
img	add example images	6 months ago
pkg	add recovery command	6 months ago
.gitignore	add scaffolding	6 months ago
.goreleaser.yaml	add version	6 months ago
LICENSE	fix: remove license boilerplate	2 months ago
README.md	reference security section in requirements	5 months ago
go.mod	[dep][go](deps): Bump the golang-dependencies group ...	last week
go.sum	[dep][go](deps): Bump the golang-dependencies group ...	last week
main.go	add scaffolding	6 months ago

https://github.com/unicrons/aws-root-manager

Tools. Isn’t this IaC too?

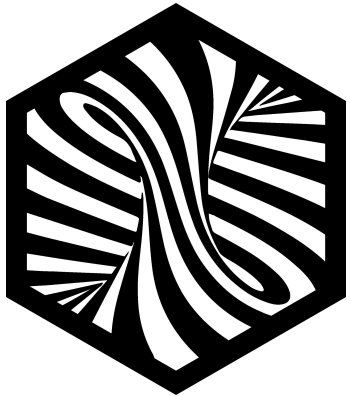
```
prowler aws -c ec2_ebs_default_encryption --fixer
```

```
Executing 1 check, please wait...

Check ID: ec2_ebs_default_encryption - ec2 [medium]
        FAIL eu-west-1: EBS Default Encryption is not activated.
        FAIL us-east-1: EBS Default Encryption is not activated.
-> Scan completed! |████████████████████████████████████████| 1/1 [100%] in 32.6s

Running Prowler Fixer, please wait...

Fixing fails for check ec2_ebs_default_encryption...
        FIXING eu-west-1...
        DONE
        FIXING us-east-1...
        DONE

2 findings fixed!
```

https://hub.prowler.com/check/ec2_ebs_default_encryption

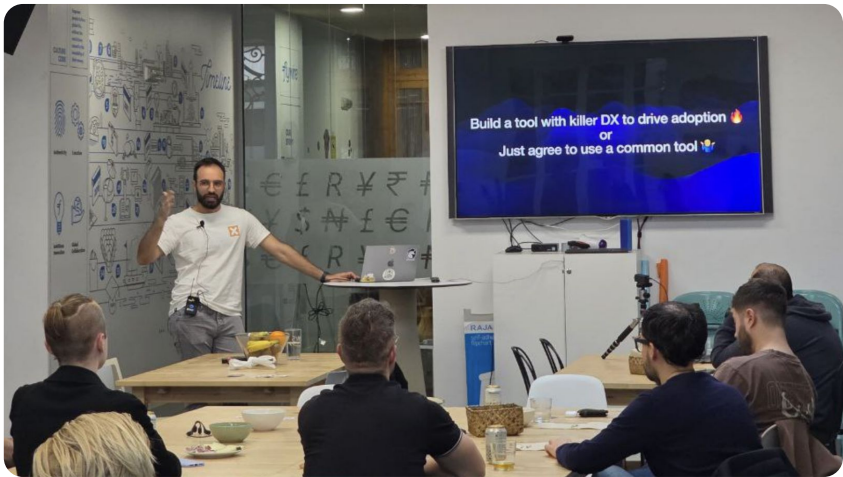
https://docs.prowler.com/projects/prowler-open-source/en/latest/tutorials/fixer/

Tools. Isn’t this IaC too?



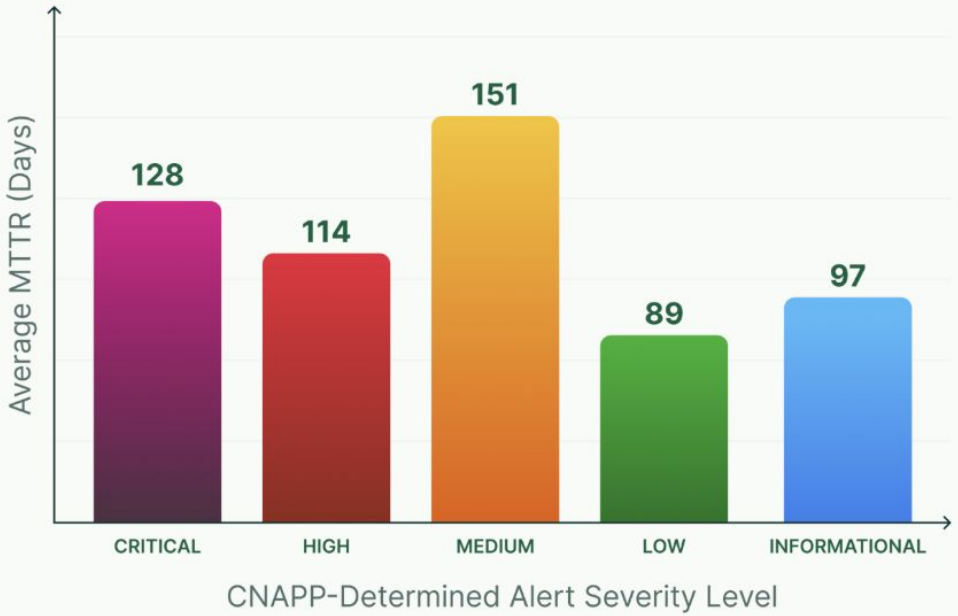
Internal Tooling

“**InnerSource** takes the lessons learned from developing open source software and applies them to the way companies develop software internally.”



innersourcecommons.org

Cloud Alert Resolution Time: Severity Matters



2025 State of Cloud Remediation Report

All Alerts Dataset, Closed Alerts

https://tamnoon.io/state-of-cloud-remediation/

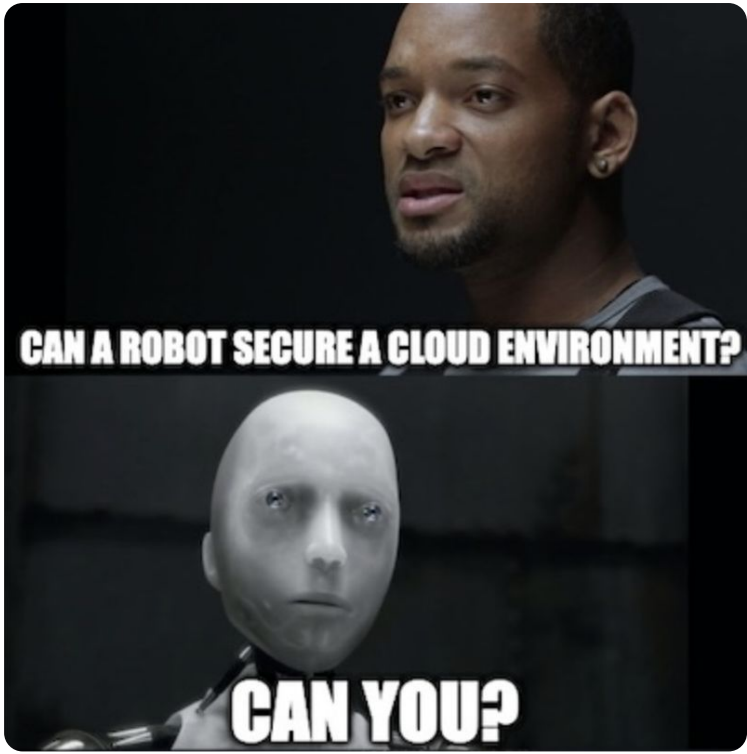
The hybrid approach

Auto Remediation

Auto Remediation

Farris's Three Laws of Cloud Security Auto Remediation:

- A bot must never harm stateful data or allow stateful data to come to harm.
- A bot must act with utmost haste so functionality doesn't become dependent on a misconfiguration.
- A bot must announce its existence and tell a carbon-based life form what it did and why.

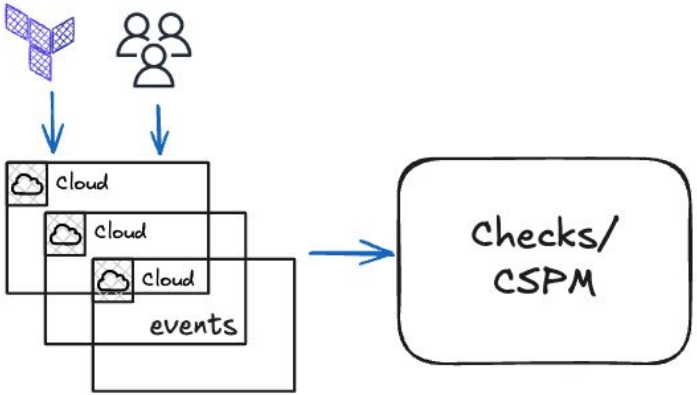


(Stolen meme)

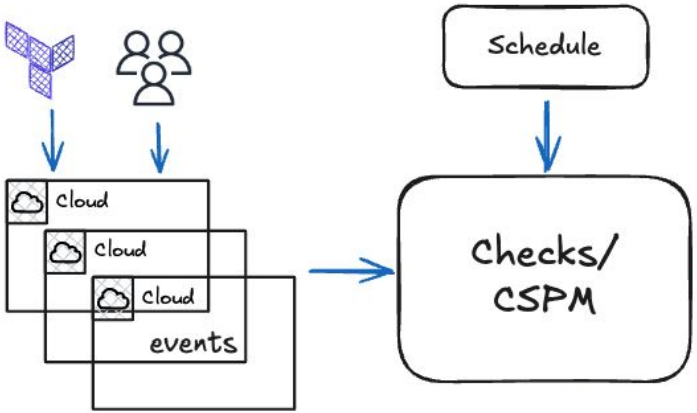
https://www.chrisfarris.com/post/three-laws/

Auto Remediation

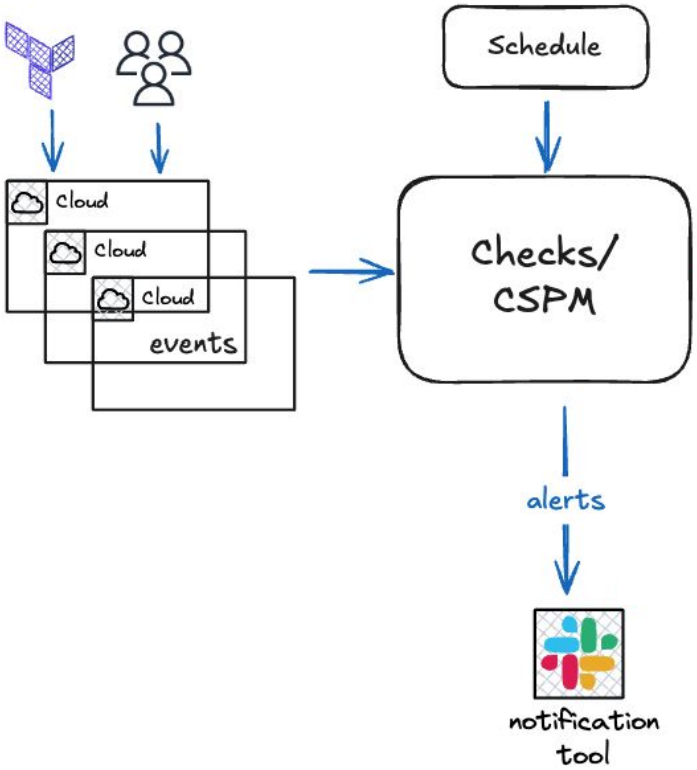
Auto Remediation



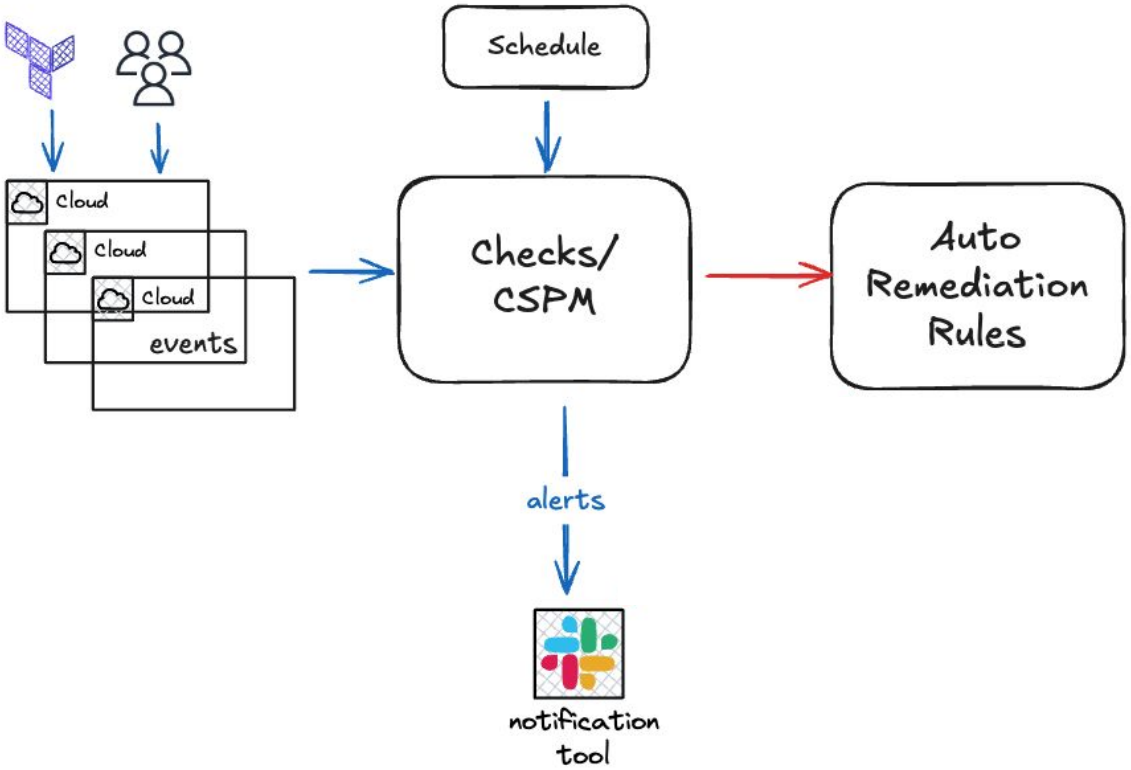
Auto Remediation



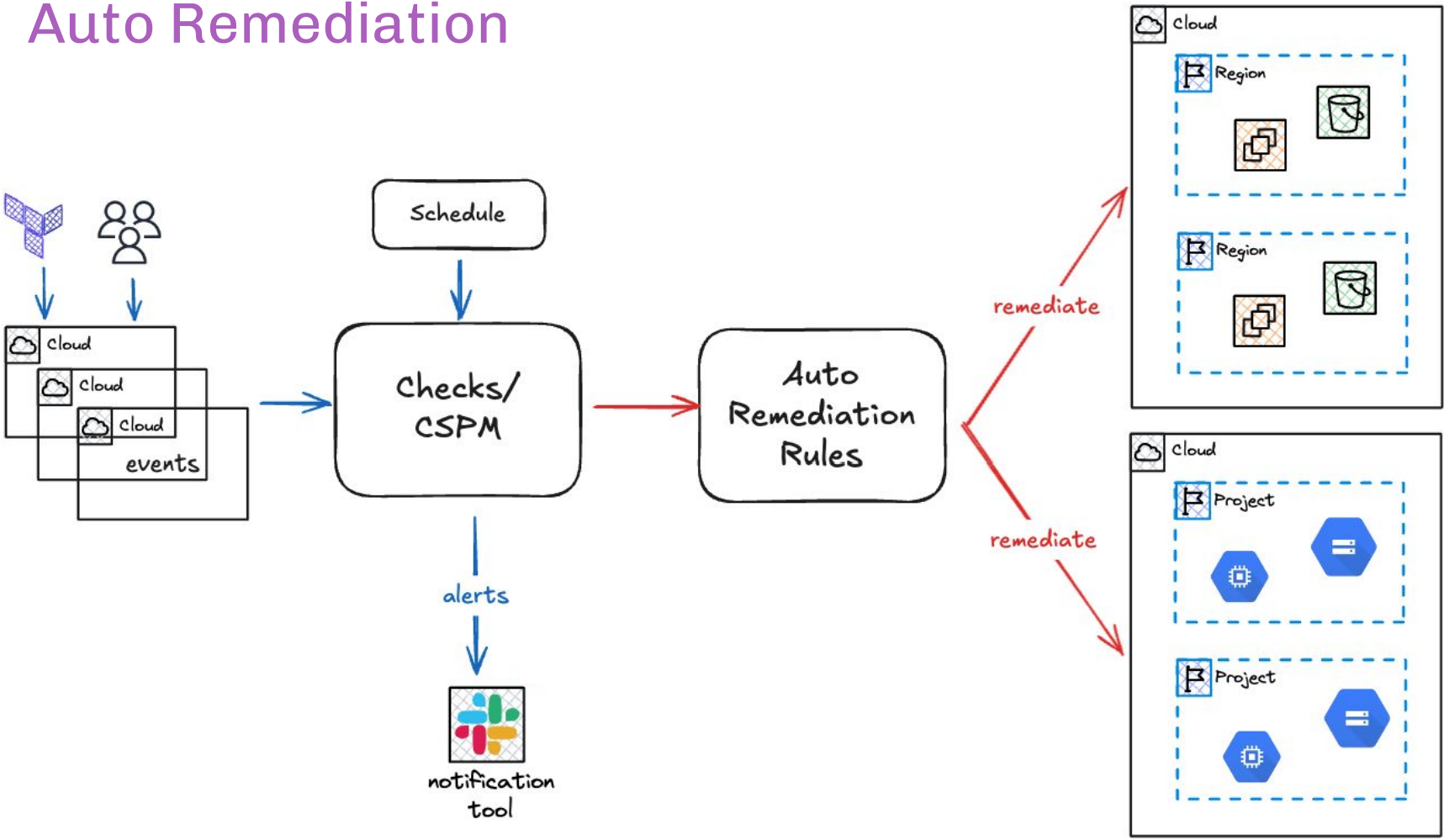
Auto Remediation



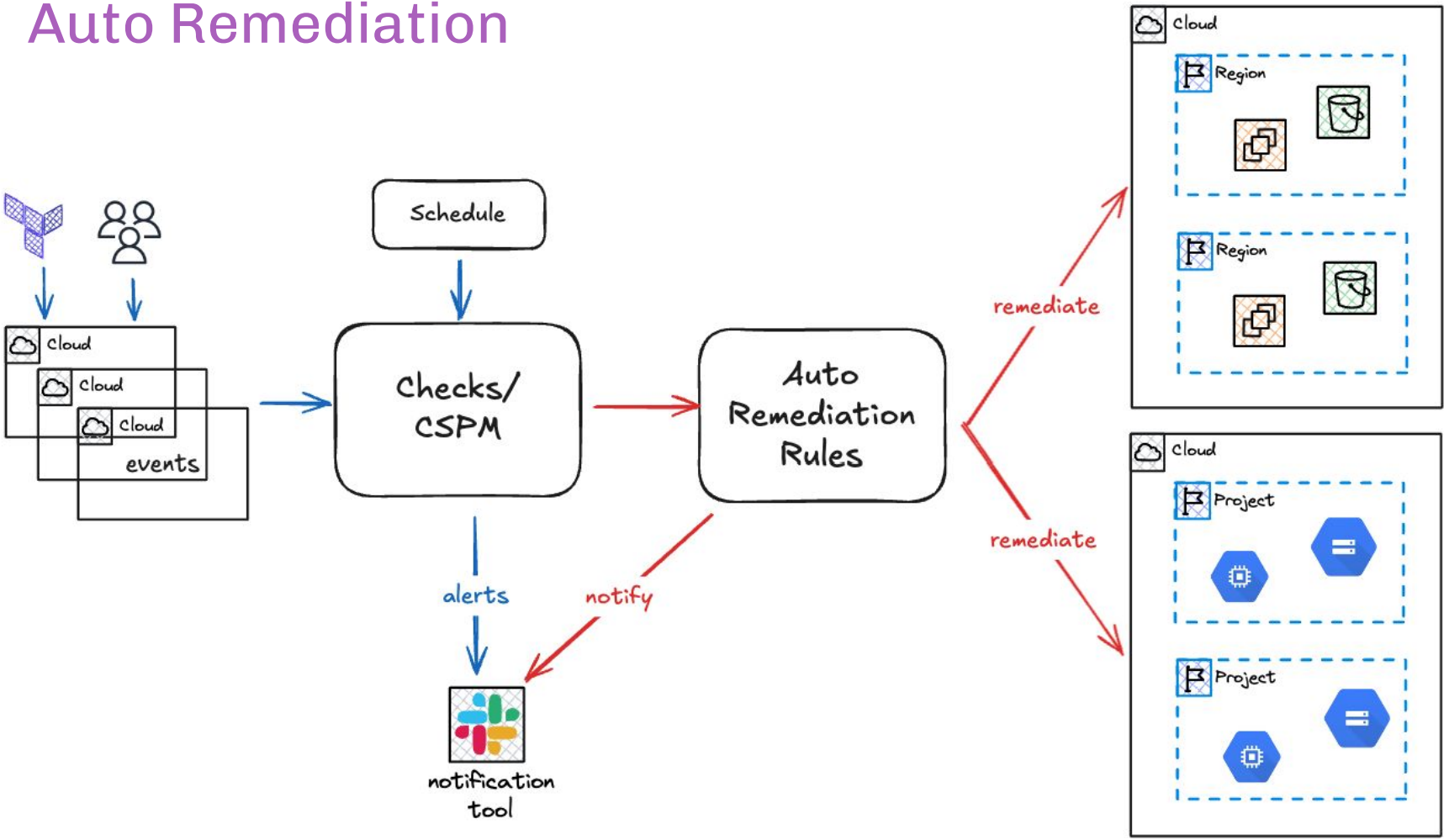
Auto Remediation



Auto Remediation



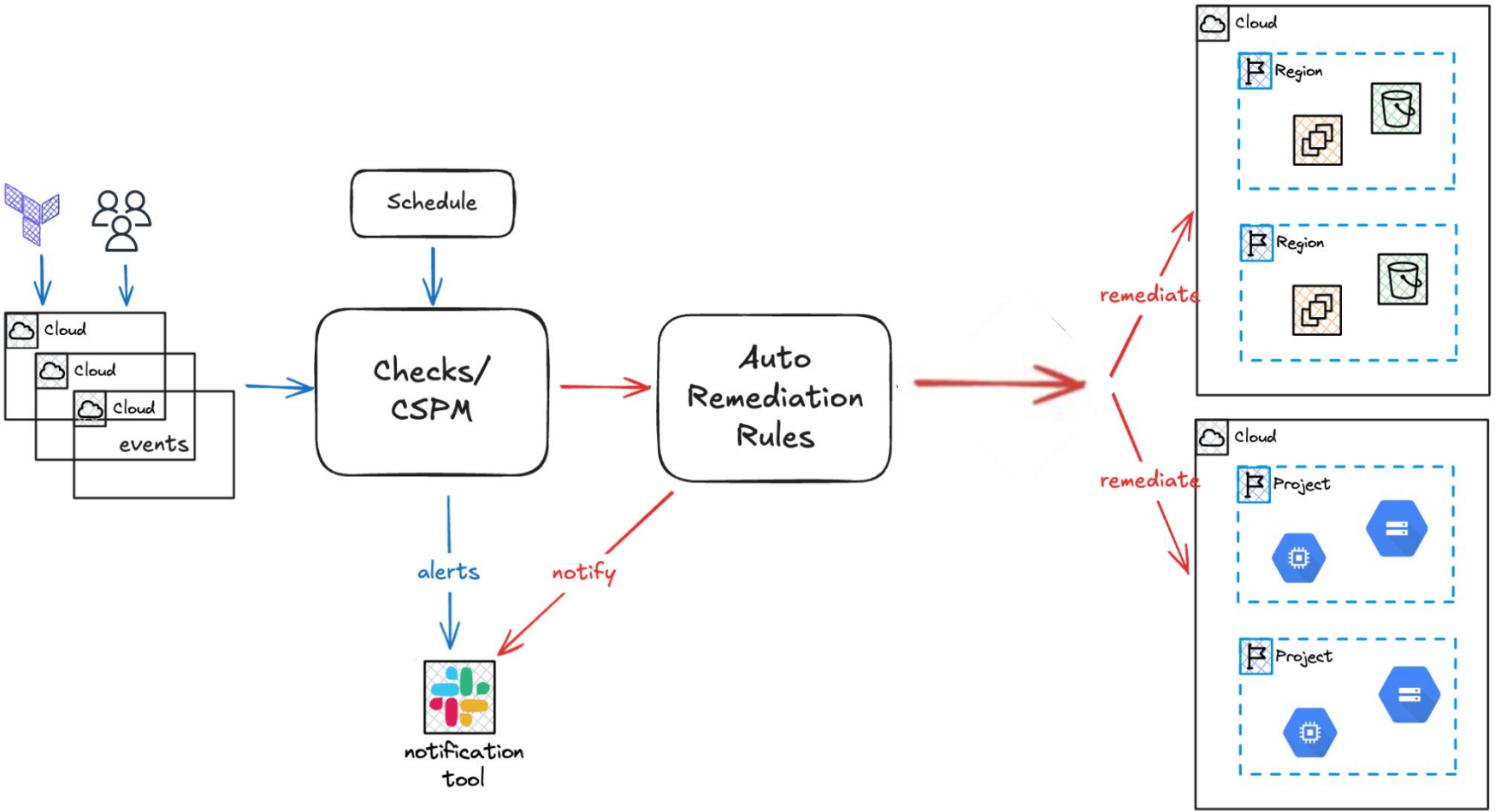
Auto Remediation

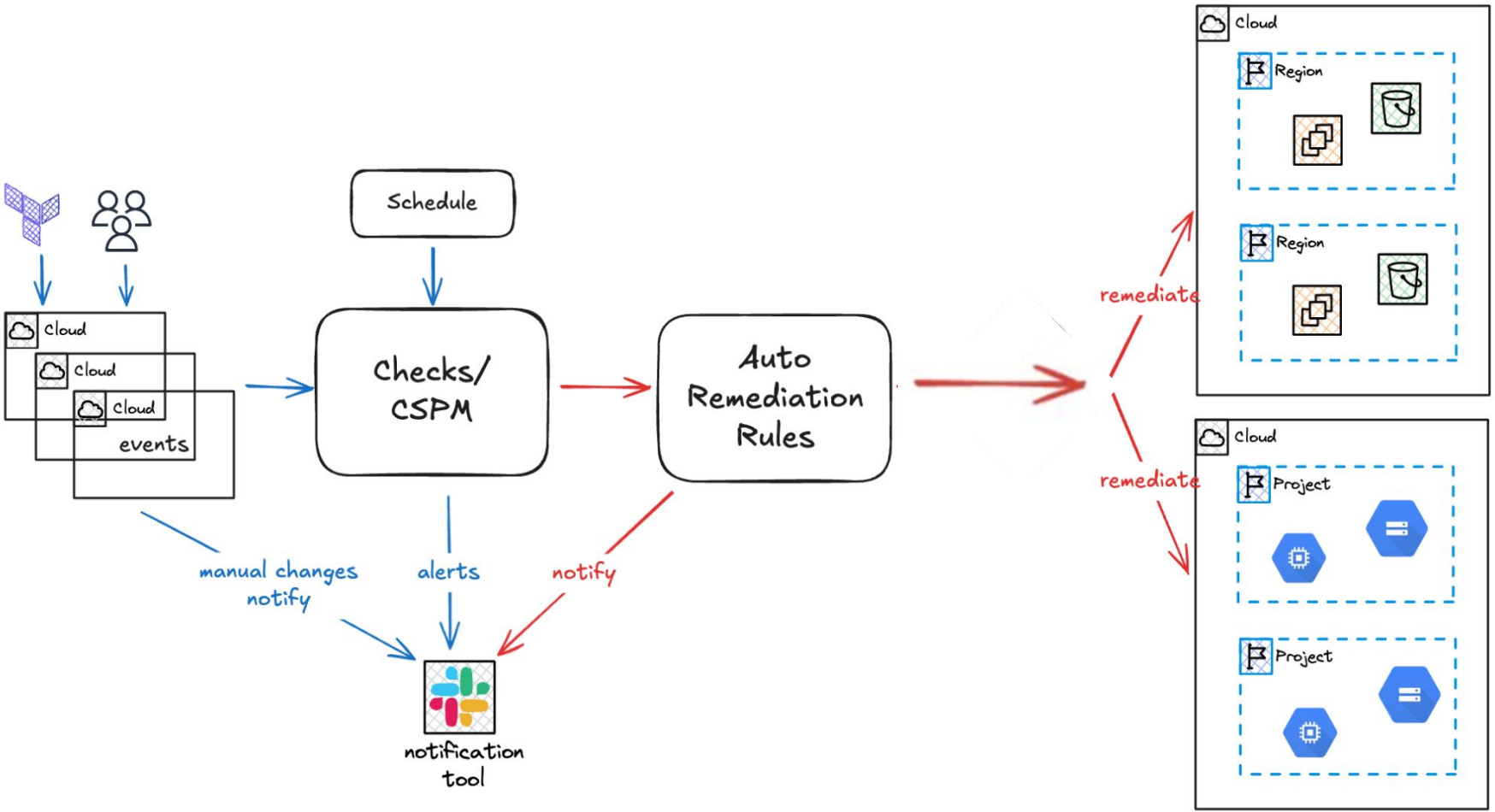


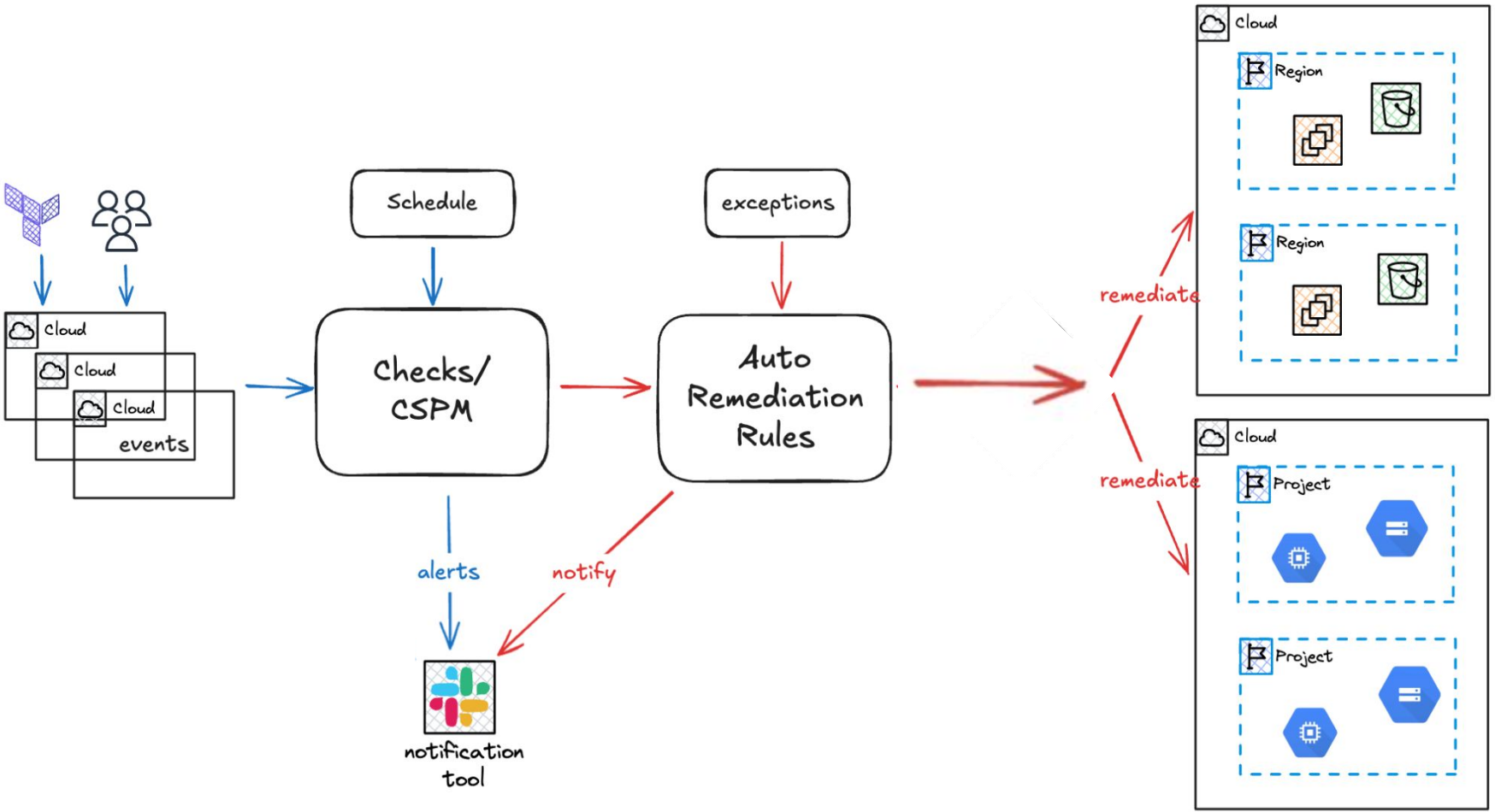
IaC + Manual Changes + Scripts + Tools = 🔥

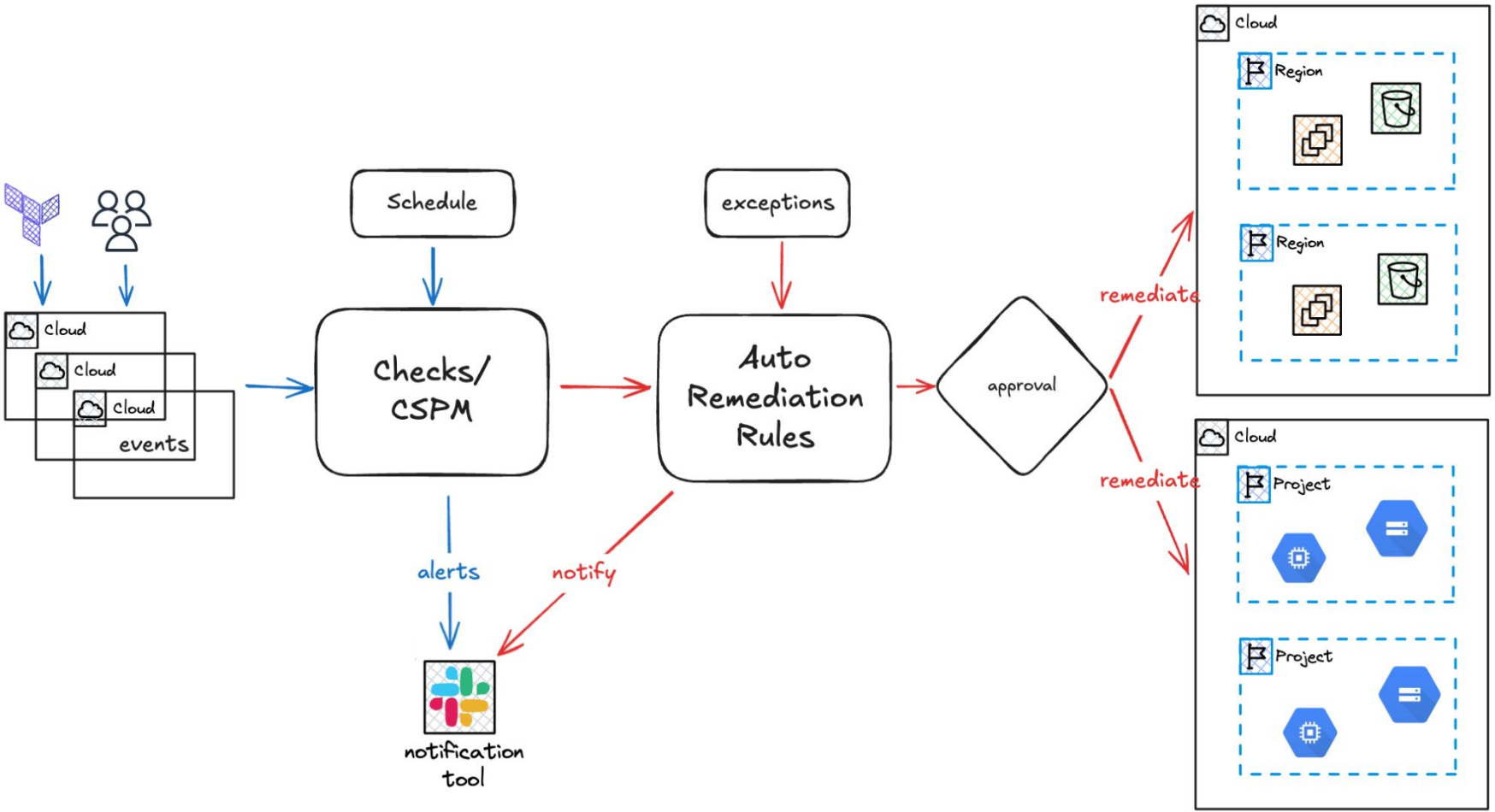
Can we put everything together?

Yes, but be careful


Cloud
Cloud
Cloud
events
Schedule
Checks/
CSPM
Auto
Remediation
Rules
alerts
notify
notification
tool
remediate
remediate
Cloud
Region
Region
Cloud
Project
Project


Schedule
Cloud
Cloud
Cloud
events
Checks/
CSPM
Auto
Remediation
Rules
remediate
remediate
Cloud
Region
Region
Cloud
Project
Project
manual changes
notify
alerts
notify
notification
tool


Cloud
Cloud
Cloud
events
Schedule
Checks/
CSPM
exceptions
Auto
Remediation
Rules
alerts
notify
notification
tool
remediate
remediate
Cloud
Region
Region
Cloud
Project
Project


Cloud
Cloud
Cloud
events
Schedule
Checks/
CSPM
exceptions
Auto
Remediation
Rules
approval
remediate
remediate
alerts
notify
notification
tool
Cloud
Region
Region
Cloud
Project
Project

New public bucket using IaC

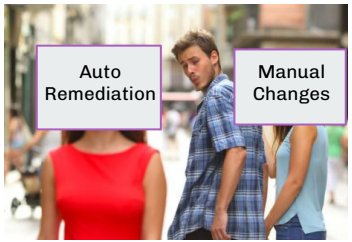
Auto Remediation

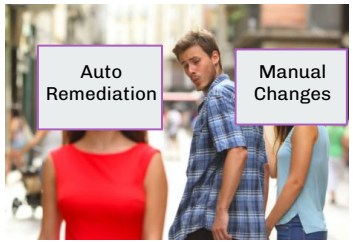
Admin change it to public

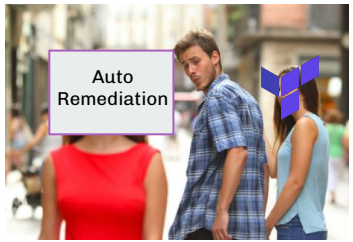
Auto Remediation

Dev triggers a new CI

Auto Remediation







Where to start?

Where to start?

Do you need a state?

Is too complex to implement it using IaC?

One time configuration?

Can it be auto remediated?

Where to start? Our first steps...

“Safe” changes

- Password Policy
- Delete Root Access Keys
- Enable Security Services
- Enable KMS Rotation

Potential Impact Changes

- Public AMI / EBS
- EBS Default Encryption
- Public Buckets
- Public Security Groups

Thanks for coming!

@sbldevnet
sbldevnet
sbldevnet.com

@andoniaf_
andoniaf
andoniaf.unicrons.cloud

unicrons
@unicrons_cloud
unicrons.cloud

Questions?