# Using AWS for Remote State

**Ned Bellavance**
FOUNDER, NED IN THE CLOUD

@ned1313 | nedinthecloud.com

# Overview

Terraform remote state

AWS for remote state

Migrating the current state

# Terraform State Data

**State is local by default**

**Safeguard the state data**

**Enable team collaboration**

**Multiple supported backends**

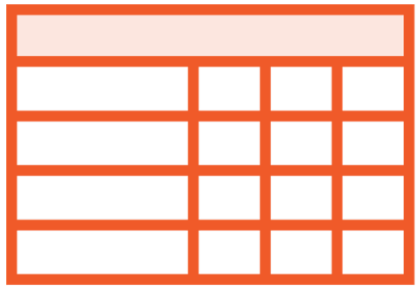
**Standard and enhanced**

**Special features**

- Locking
- Workspaces

# AWS Storage for Remote State

Simple Storage Service (S3)

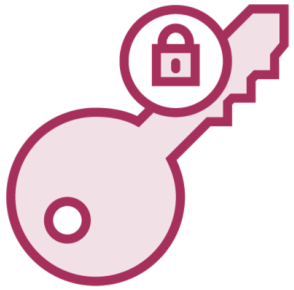


DynamoDB

Supports locking

Supports workspaces

Granular control



Supports encryption

# Authentication Methods

Instance profile

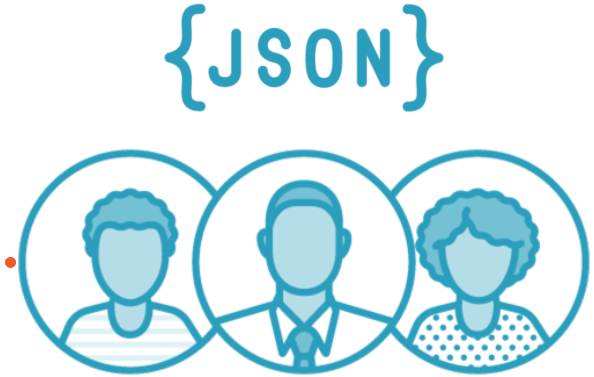
Access & secret keys

Credentials file & profile

Session token

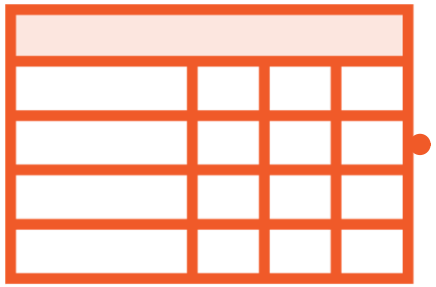
# Remote State Storage

Infra bucket




Full access

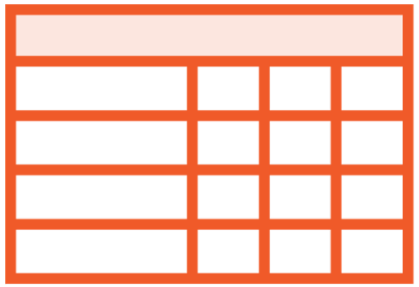
Infra table



# Remote State Storage

Infra bucket

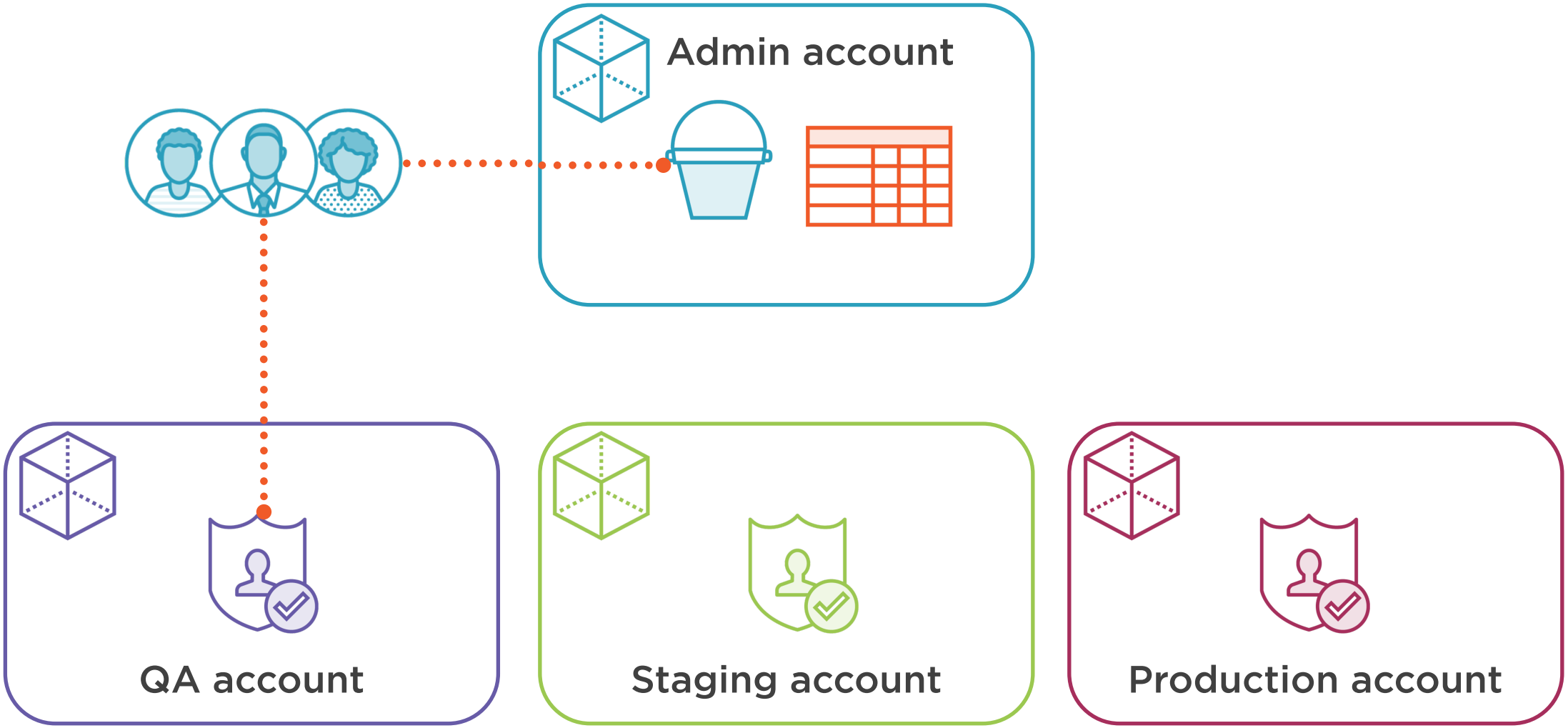
Infra table



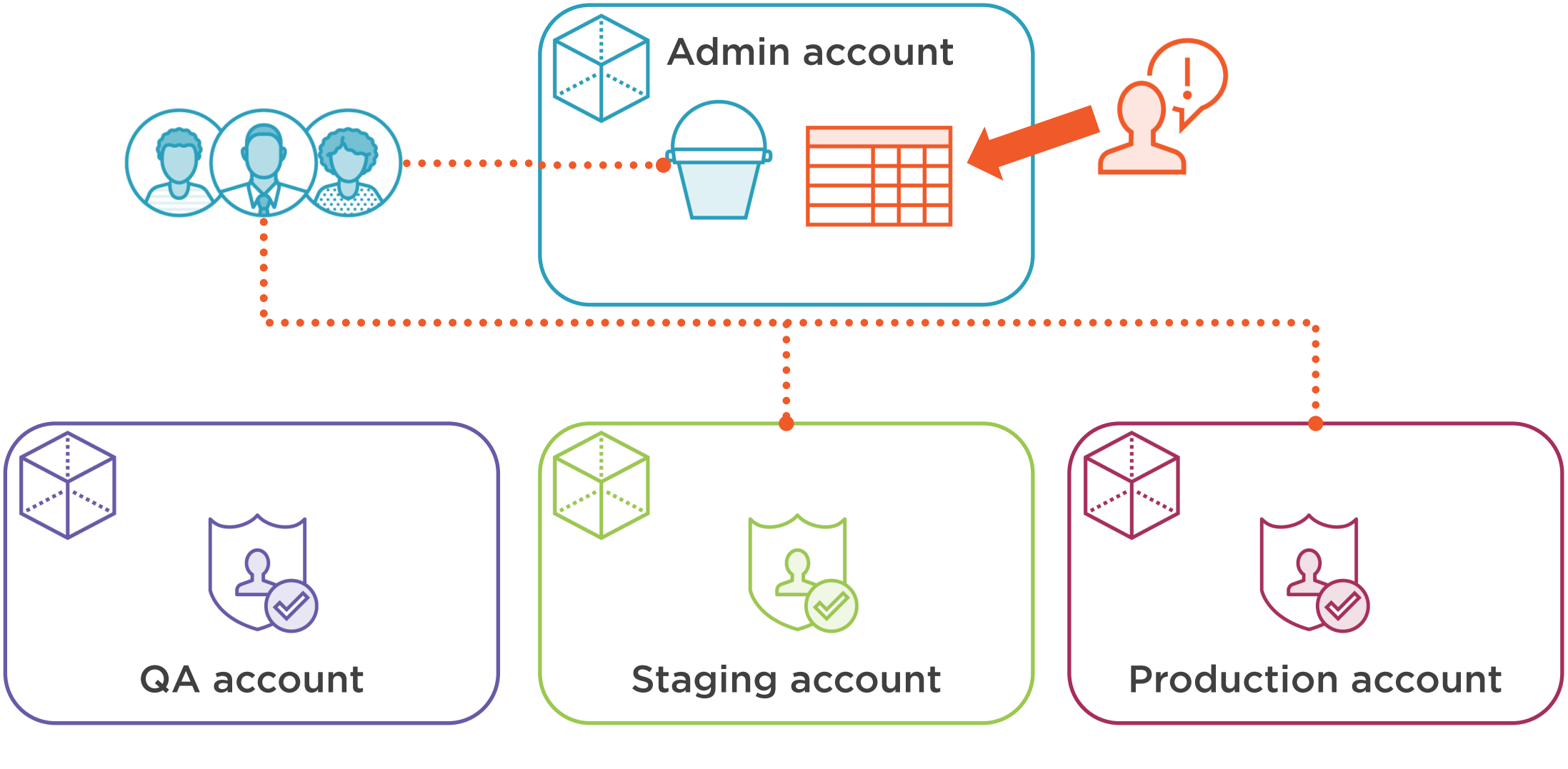

Full access


Read-only

# Multiple Environments



# Multiple Environments



# Migrating Terraform State

**Update backend configuration**

**Run terraform init**

**Confirm state migration**

# Backend Configuration

```
terraform {
  backend "s3" {
    bucket = "globo-infra-12345"
    key    = "terraform-state"
    region = "us-east-1"
  }
}
terraform init –backend-config="profile=infra"
```

# Summary

**Use remote state by default**

**AWS S3 and DynamoDB supports locking and workspaces**

**Migration is super simple**

**Coming up:**

- Adding in automation
- Adopting AWS Code tools