Using Terraform in the context of AWS multi-account setups

Jan.Nabbefeld@kreuzwerker.de

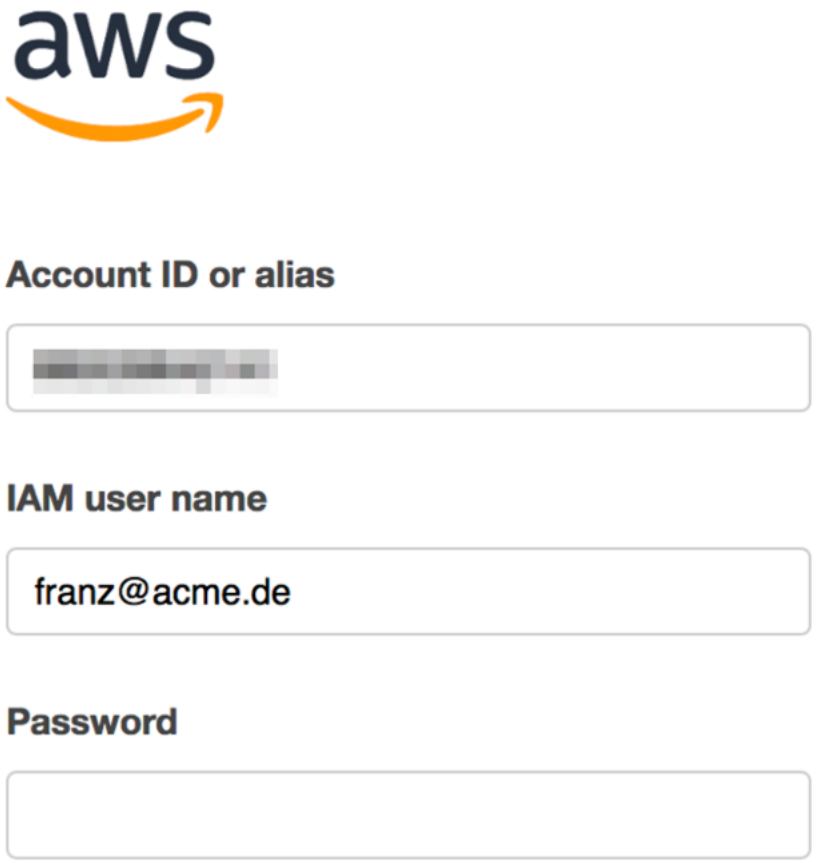
AWS accounts for security purposes

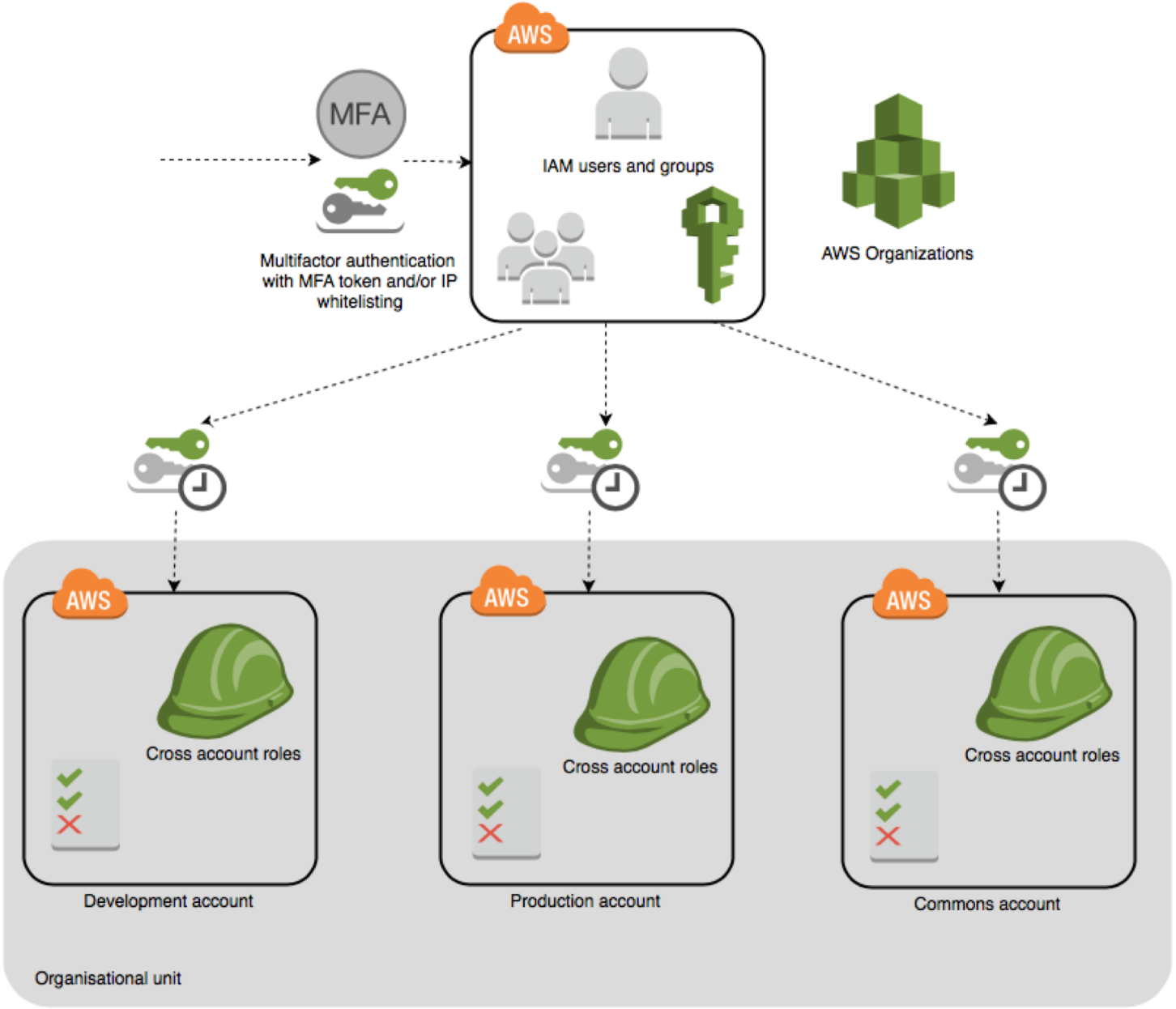
- strong isolation of recovery and/or auditing data
- administrative isolation between workloads
- limited visibility and discoverability of workloads
- isolation to minimize blast radius

AWS multi-account setups alias AWS Organizations

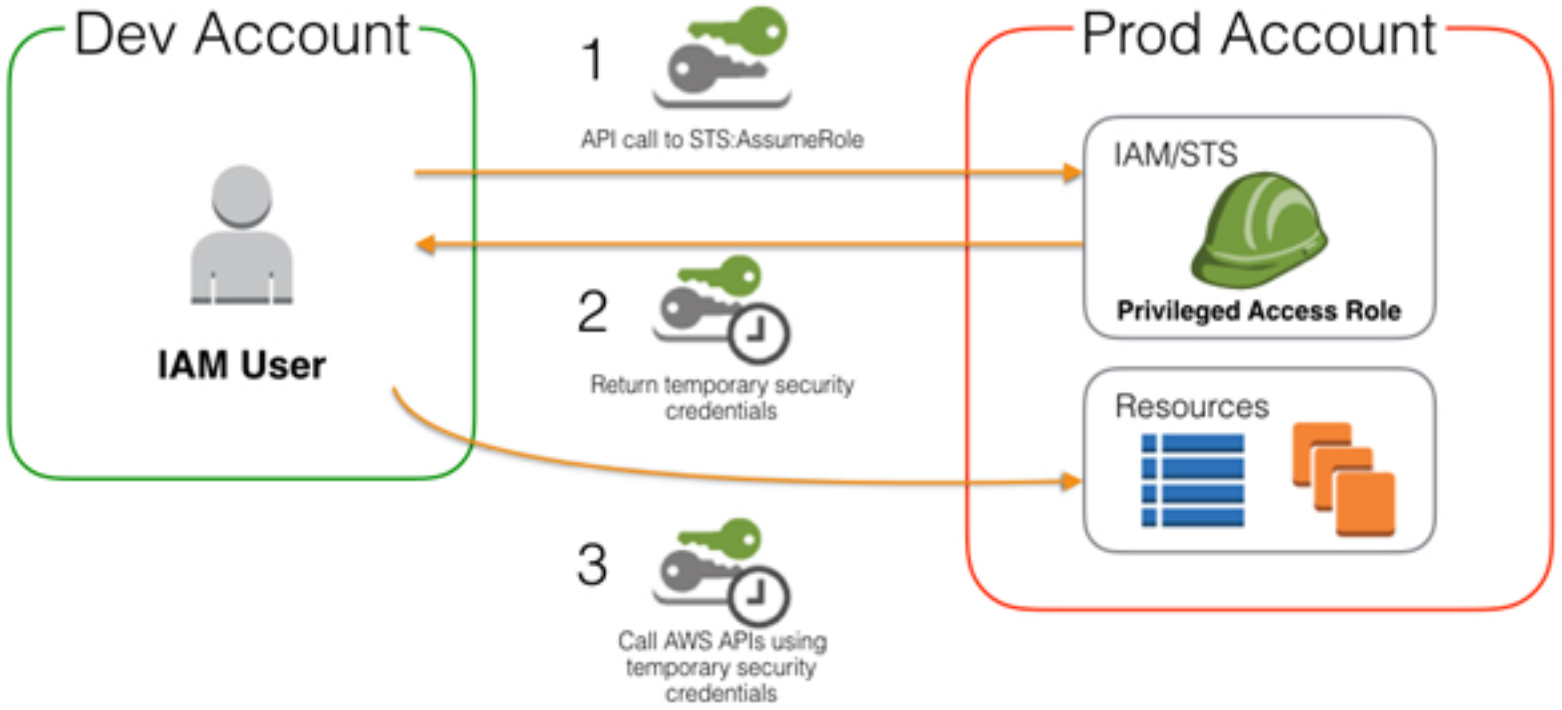
- Service launched in February 2017
- Enables you to consolidate multiple AWS accounts into an organization
- Implements account hierarchy among other features
- Defines the AWS account management standard today

AWS Console Example




AWS
MFA
IAM users and groups
Multifactor authentication with MFA token and/or IP whitelisting
AWS Organizations
AWS
Cross account roles
Development account
AWS
Cross account roles
Production account
AWS
Cross account roles
Commons account
Organisational unit

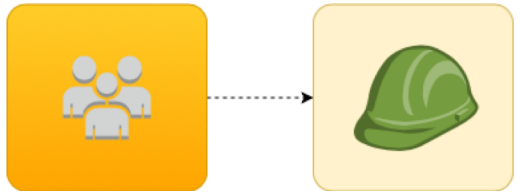
STS:AssumeRole



https://aws.amazon.com/blogs/security/how-to-use-a-single-iam-user-to-easily-access-all-your-accounts-by-using-the-aws-cli/

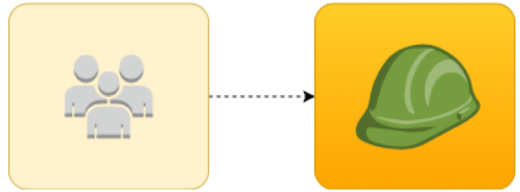
IAM policy to assume a cross account role

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::<TARGET ACCOUNT ID>:role/admin-cross-account"
     }
  ]
}
```



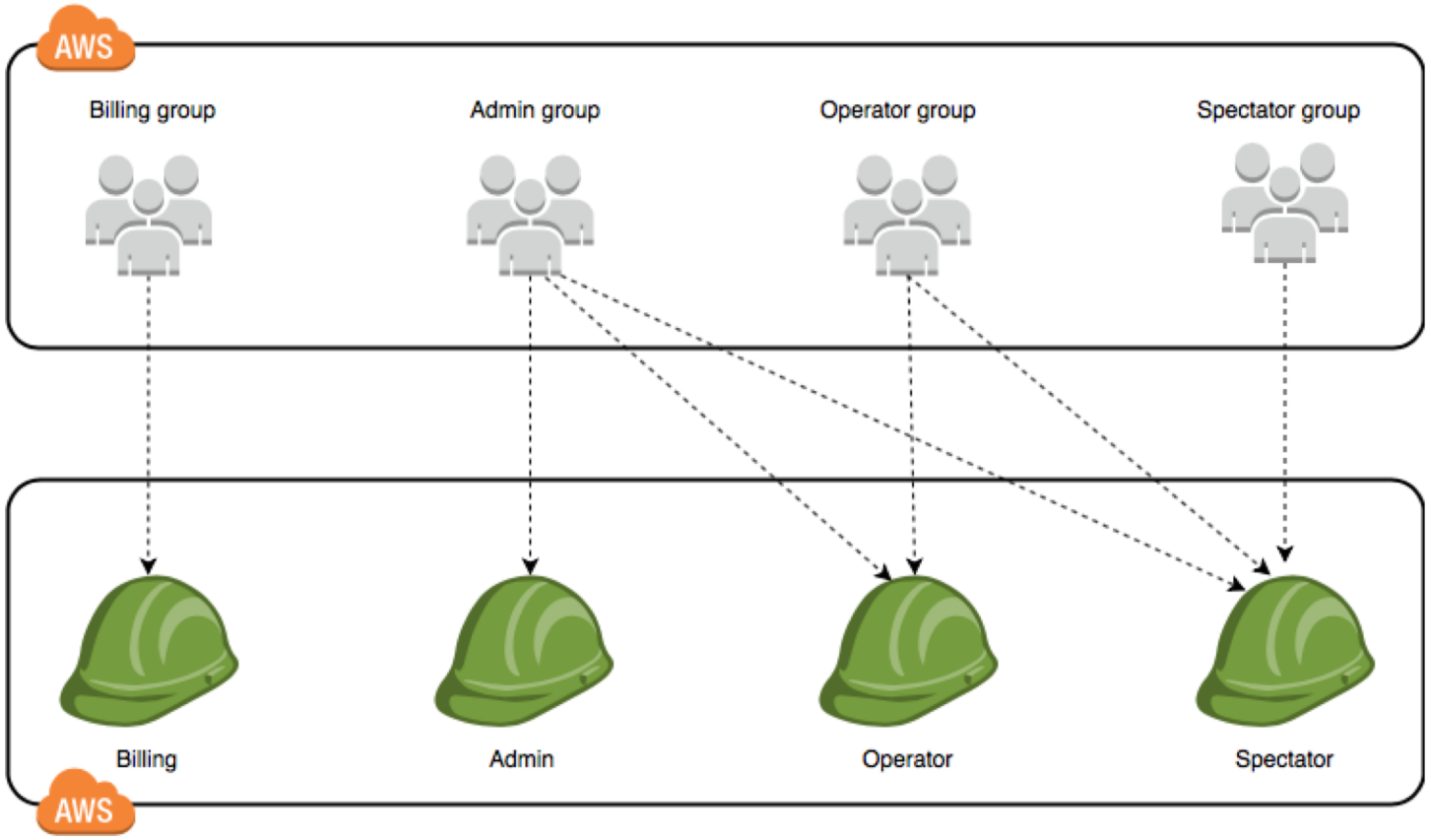
IAM trust relationship of a cross account role

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<IAM USER ACCOUNT ID>:root"
      },
      "Action": "sts:AssumeRole",
    }
  ]
}
```



Permission schema

- Admin: full access
- Operator: limited IAM, expensive operations like CloudHsm, RI, AWS Organizations are denied
- Spectator: ReadOnly resp. ViewOnly



Terraform

Cross Account Deployments

HashiCorp Terraform

DevOpsCon

Terraform project structure

- Separate Terraform modules and stacks
- Use terraform remote state (e.g. S3)
- Keep your Terraform small and lightweight
- Reference external resources via data sources like `$data.terraform_remote_state.state_name.output`
- Work with locals to use interpolation for conditions and simple math (`locals.tf`)

HashiCorp Terraform

Terraform global configuration

```
└── stacks
    ├── mystack
    │   ├── main.tf
    │   ├── outputs.tf
    │   ├── config.tf-> ../config.tf
    │   ├── terraform.tfvars -> ../terraform.tfvars
    │   └── variables.tf -> ../variables.tf
    ├── config.tf
    ├── terraform.tfvars
    └── variables.tf
```

HashiCorp Terraform

Global variables for multiple AWS accounts

```
# terraform.tfvars
customer_tag = "acme"
remote_state_bucket = "eu-central-1-kreuzwerker-default-accountmgmt-state"

user_account_id = "123456789012"
user_account_role_arn = "arn:aws:iam::123456789012:role/acme-user-admin"

sandbox_account_id = "09876543212"
sandbox_account_role_arn = "arn:aws:iam::09876543212:role/OrganizationAccountAccessRole"
```

HashiCorp Terraform

Stack related configuration (locals)

```
locals {
  billing_role_name = "billing-cross-account"

  admin_role_name      = "${var.customer_tag}-admin-cross-account"
  operator_role_name   = "${var.customer_tag}-operator-cross-account"
  spectator_role_name = "${var.customer_tag}-spectator-cross-account"
}
```

HashiCorp Terraform

DevOpsCon

Terraform AWS provider with alias

```
# config.tf
provider "aws" {
  allowed_account_ids = [
    "${var.user_account_id}",
  ]

  assume_role {
    role_arn = "${var.user_account_role_arn}"
  }

  alias   = "${var.sandbox_account_id}"
  region  = "${var.aws_region}"
  version = "~> 1.17"
}
```

HashiCorp Terraform

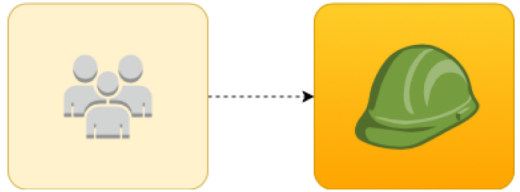
DevOpsCon

Terraform AWS provider usage (IAM roles)

```
module "sandox_account_roles" {
  providers = {
    aws = "aws.${var.sandbox_account_id}"
  }

  admin_role_name      = "${local.admin_role_name}"
  operator_role_name   = "${local.operator_role_name}"
  spectator_role_name = "${local.spectator_role_name}"
  user_account_id      = "${var.user_account_id}"

  source = "../../modules/account-iam-roles"
}
```

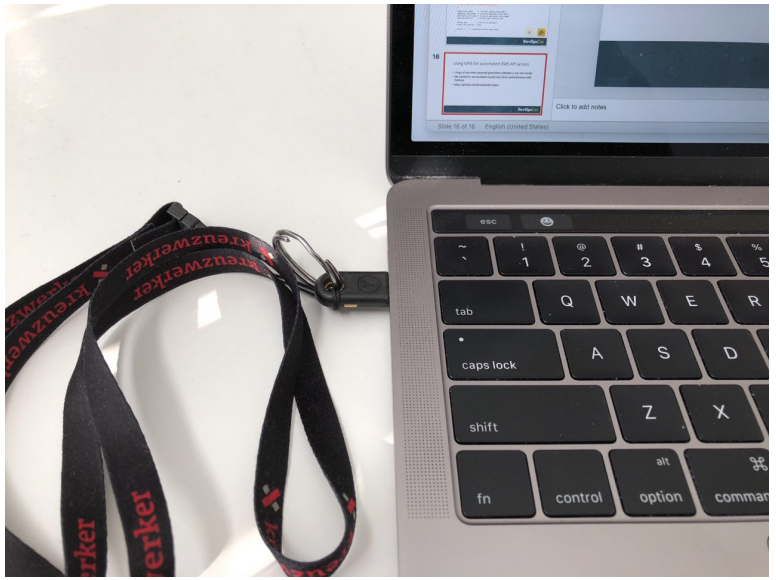


Security Considerations

Multi Factor Authentication / IP whitelisting

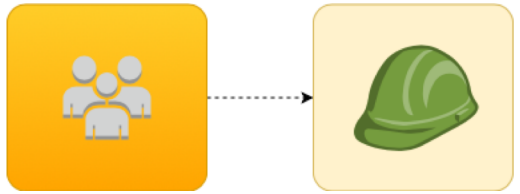
Using MFA for automated AWS API access

- Usage of one-time passcode generators software is not very handy
- We wanted to use hardware-based two-factor authentication with Yubikeys
- https://github.com/kreuzwerker/awsu



IAM policy to assume a cross account role

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::<TARGET ACCOUNT ID>:role/admin-cross-account",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": “110.1.1.1/32"
        },
        "BoolIfExists": {
          “aws:MultiFactorAuthPresent": true
        }
      }
    }
  ]
}
```



AWS CLI basics

```
AWS_ACCESS_KEY_ID = AKIABLAHBLAHBLAHBLAH

AWS_SECRET_ACCESS_KEY = 4d8Dz0/JoxcgggvcdYHQTTIgSJbT

AWS_SESSION_TOKEN =
FQoDYXdzX49wcGYQtnKViKvASw+1UjY98xbSyGrPkFU+NksK4OtUu+fGhwEbdE0i7e9
cerNEKtJIbbljMIVxpzK9Rwibmwel//fzzzJPvsdasfsdf8+BsvU69Ammkh+g6Gi2k9
Ivj7F4766lbe0jF7nH5IALD5kQsMwaCzWL4HoaCuH7jIuD9uLHYZ9BhhF4/0FnsYqNQ
QrBpl1L0Ru+vX1+uKuuv/dSW0m7uglN1cko4tmb2AU=

AWS_DEFAULT_REGION = eu-central-1

AWS_SHARED_CREDENTIALS_FILE = ~/code/TF-project/.config
```

AWS_SHARED_CREDENTIALS_FILE

```
[franz@acme.de]
aws_access_key_id = AKIABLAHBLAHBLAHBLAH
aws_secret_access_key = <blah>
mfa_serial = arn:aws:iam::123456789123:mfa/franz@acme.de

[franz+admin@acme.de]
role_arn = arn:aws:iam::123456789123:role/acme-user-admin
source_profile = franz@acme.de
mfa_serial = arn:aws:iam::123456789123:mfa/franz@acme.de
```

Example usage

```
$ awsu -p acme-user-admin –v -- terraform plan
using aquirer "long_term" (cache: false) for profile "acme-user-admin”
using "acme-user-admin" for MFA serial
using aquirer "session_token" (cache: true) for profile "acme-user-admin"
failed to load cached profile "acme-user-admin": existing credentials are
invalid
using "acme-user-admin" for MFA serial
getting session token for profile "acme-user-admin" and serial
"arn:aws:iam::123456789123:mfa/franz@acme.de"
asking for yubikey OATH slot with issuer "aws/iam/123456789012" and name
"franz@acme.de"
received "626104" as code
running "/usr/bin/terraform" with args [”terraform" ”plan"]
```

Questions

Nerdy backup slides

Terraform AWS provider usage (default)

```
module "internal_groups" {
  customer_tag = "${var.customer_tag}"
  group_prefix = "${var.customer_tag}-internal"

  admin_role_name     = "${local.admin_role_name}"
  operator_role_name  = "${local.operator_role_name}"
  spectator_role_name = "${local.spectator_role_name}"

  accounts = [
    "${var.sandbox_account_id}",
    "${var.testing_account_id}",
    "${var.commons_account_id}",
  ]

  enable_mfa_device         = true
  source_ips                = ["110.1.1.1/32"]
  inherit_role_permissions = true

  source = "../../modules/aos-groups"
}
```