Hands-on Infrastructure Automation with Terraform on AWS

Alexander Savchuk

Working with Terraform as a team

Section 7

Managing secrets securely

Types of secrets

- **API keys (e.g. provider credentials)**
- Resource secrets (e.g. initial password for RDS instance)
- Application secrets (e.g. database password)

Types of secrets

- API keys (e.g. provider credentials)
- **Resource secrets (e.g. initial password for RDS instance)**
- Application secrets (e.g. database password)

Types of secrets

- API keys (e.g. provider credentials)
- Resource secrets (e.g. initial password for RDS instance)
- **Application secrets (e.g. database password)**

API keys

```
provider "aws" {
  access_key = "${var.aws_access_key}"
  secret_key = "${var.aws_secret_key}"
  region     = "us-east-1"
}
```

Storing API keys

- **Source control (don’t, seriously)**
- Command-line flags
- Gitignored variable files
- Shared credentials file (AWS)
- Environment variables

Storing API keys

- ~~Source control (don’t, seriously)~~
- **Command-line flags**
- Gitignored variable files
- Shared credentials file (AWS)
- Environment variables

Storing API keys

- ~~Source control (don’t, seriously)~~
- ~~Command-line flags~~
- **Gitignored variable files**
- Shared credentials file (AWS)
- Environment variables

Storing API keys

- ~~Source control (don’t, seriously)~~
- ~~Command-line flags~~
- Gitignored variable files
- **Shared credentials file (AWS)**
- Environment variables

Storing API keys

- ~~Source control (don’t, seriously)~~
- ~~Command-line flags~~
- Gitignored variable files
- Shared credentials file (AWS)
- **Environment variables**

Protecting the state file

```
terraform {
  backend "s3" {
    encrypt = "true"
    kms_key_id =
"arn:aws:kms:ap-southeast-2:111111111111:key/d2834912-402d-4951-
9956-9c0553ede985"
  }
}
```

Protecting the bucket

```
resource "aws_s3_bucket" "main" {
 server_side_encryption_configuration {
   rule {
     apply_server_side_encryption_by_default {
       sse_algorithm = "AES256"
     }
   }
 }
```

Protecting the bucket

```
resource "aws_s3_bucket_policy" "main" {
  policy = "..."
}
```

Keep secrets out of state

Resource secrets

```
resource "aws_db_instance" "default" {
  password = "${random_string.db_password.result}"
}
```

Creating resources securely

1. Seed the resource with initial secret, stored in the state file
2. Change the secret to a secure one, stored in a secret manager (AWS SSM Parameter Store, AWS Secrets Manager, Hashicorp Vault, etc.)

Application secrets (hard-coded)

```
"environment": [
  {
    "name": "PGPASSWORD",
    "value": "${pgpassword}"
  }
]
```

Fetching secrets from the secrets store

1. Store secrets in the secrets manager
2. Use a secrets helper to fetch them at application start-up time, decrypt and expose as environment variables
3. No changes to the application code needed
4. Still possible to set secrets directly for local development

Application secrets (dynamic)

```
"environment": [
  {
    "name": "PSTORE_PGPASSWORD",
    "value": "${parameter_store_pgpassword}"
  }
]
```

Secrets helpers

1. https://github.com/glassechidna/pstore
2. https://github.com/segmentio/chamber

Use secrets managers to store the secrets

Running Terraform in automation for CI/CD

Next video