authenticate itself to another machine (e.g., AWS API servers) without storing any secrets in plain text?

The solution here heavily depends on the type of machines involved: that is, the machine you're authenticating *from* and the machine you're authenticating *to*. Let's go through three examples:

- CircleCI as a CI server, with stored secrets
- EC2 Instance running Jenkins as a CI server, with IAM roles
- GitHub Actions as a CI server, with OIDC

WARNING: SIMPLIFIED EXAMPLES

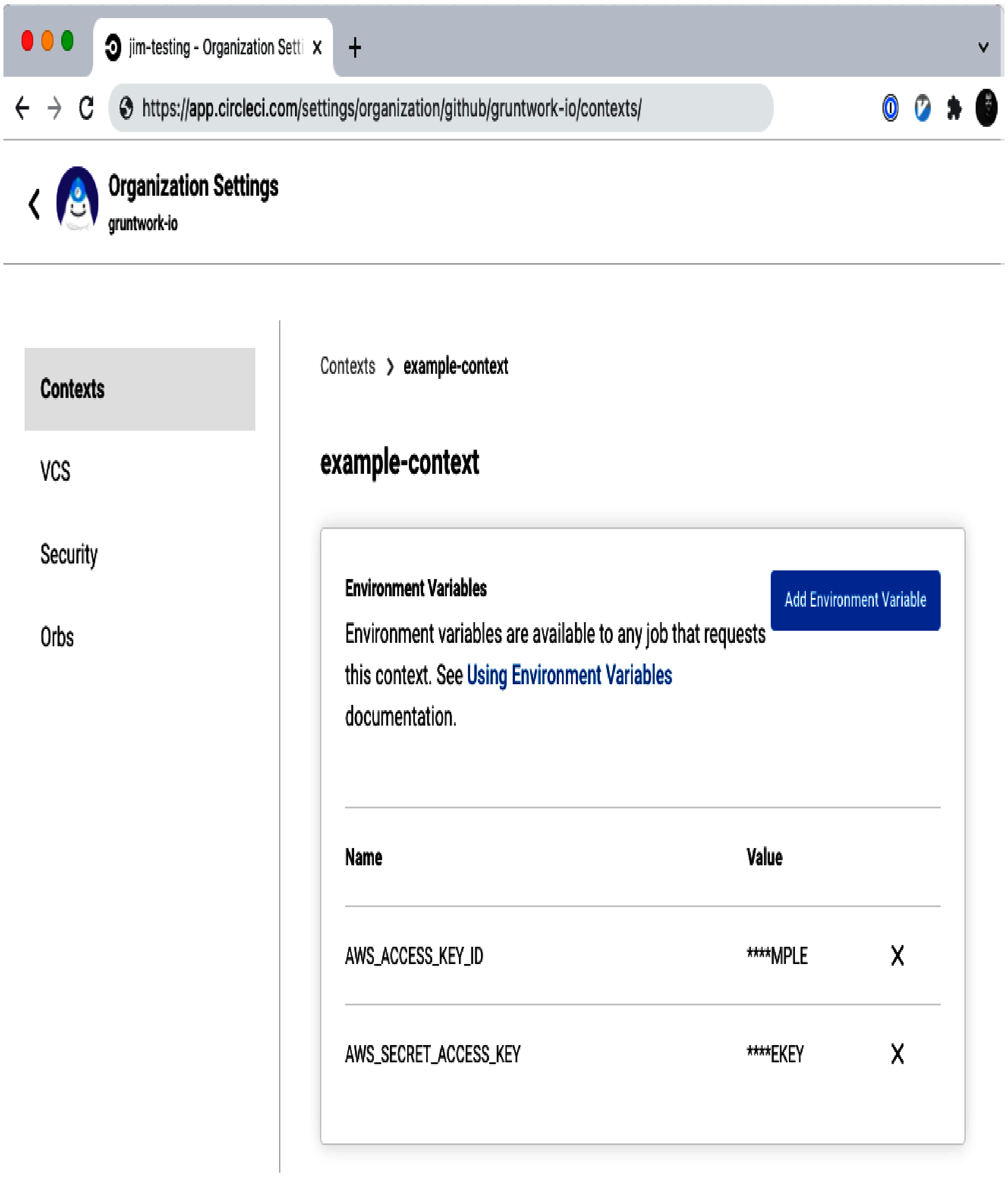
This section contains examples that fully flush out how to handle provider authentication in a CI/CD context, but all other aspects of the CI/CD workflow are highly simplified. You'll see more complete, end-to-end, production-ready CI/CD workflows in Chapter 9.

CircleCI as a CI server, with stored secrets

Let's imagine that you want to use CircleCI, a popular managed CI/CD platform, to run Terraform code. With CircleCI, you configure your build steps in a *.circleci/config.yml* file, where you might define a job to run `terraform apply` that looks like this:

```
version: '2.1'
orbs:
  # Install Terraform using a CircleCi Orb
  terraform: circleci/terraform@1.1.0
jobs:
  # Define a job to run 'terraform apply'
  terraform_apply:
    executor: terraform/default
    steps:
      - checkout            # git clone the code
      - terraform/init      # Run 'terraform init'
      - terraform/apply     # Run 'terraform apply'
workflows:
  # Create a workflow to run the 'terraform apply' job defined
above
  deploy:
    jobs:
      - terraform_apply
    # Only run this workflow on commits to the main branch
    filters:
      branches:
        only:
          - main
```

With a tool like CircleCI, the way to authenticate to a provider is to create a machine user in that provider (that is, a user solely used for automation, and not by any human), store the credentials for that machine user in CircleCI in what's called a *CircleCI Context*, and when your build runs, CircleCI will expose the credentials in that Context to your workflows as environment variables. For example, if your Terraform code needs to authenticate to AWS, you would create a new IAM user in AWS, give that IAM user the permissions it needs to deploy your Terraform changes, and manually copy that IAM user's access keys into a CircleCI Context, as shown in Figure 6-1.


jim-testing - Organization Setti
https://app.circleci.com/settings/organization/github/gruntwork-io/contexts/
Organization Settings
gruntwork-io
Contexts
VCS
Security
Orbs
Contexts › example-context
Add Environment Variable

example-context

Environment Variables

Environment variables are available to any job that requests this context. See Using Environment Variables documentation.

Name	Value	
AWS_ACCESS_KEY_ID	****MPLE	X
AWS_SECRET_ACCESS_KEY	****EKEY	X

Figure 6-1. A CircleCI Context with AWS credentials.

Finally, you update the `workflows` in your *.circleci/config.yml* file to use your CircleCI Context via the `context` parameter:

```
workflows:
  # Create a workflow to run the 'terraform apply' job defined
above
```