Testcontainers

Are you writing test codes? Especially when writing tests that involve database operations, it's cumbersome to insert test data into the database before executing the test and remove the test data after the test. In this article, we introduce how to write test codes using a library called `testcontainers`.

What is testcontainers?

`testcontainers` is a library that provides containers for testing. You can programmatically define containers that should be run as part of a test, and clean up those resources when the test is done.

Official Documentation

Installing Testcontainers for Go

To install:

```
go get github.com/testcontainers/testcontainers-go
```

Writing Tests with testcontainers-go

Container Definition

Here's a snippet on how to define a test database container:

```
func NewTestDatabase(t *testing.T) testcontainers.Container {
    req := testcontainers.ContainerRequest{
                Hostname:     "postgres-server",
                Image:        "postgres:15.4",
                ExposedPorts: []string{"5432/tcp"},
                HostConfigModifier: func(hostConfig *container.HostConfig) {
                        hostConfig.AutoRemove = true
                },
                Env: map[string]string{
                        "POSTGRES_USER":     "user",
                        "POSTGRES_PASSWORD": "password",
                        "POSTGRES_DB":       "testdb",
                },
                HostConfigModifier: func(hostConfig *container.HostConfig) {
                        hostConfig.AutoRemove = true
                },
                Mounts: testcontainers.ContainerMounts{
                        testcontainers.BindMount(testDataPath, "/docker-entrypoint-initdb.d"),
                },
                WaitingFor: wait.ForSQL(nat.Port("5432/tcp"), "pgx", dbURL).WithStartupTimeout(time.Minute * 5),
    }
```

Starting the Container

Here's how you can start the defined container:

```
        postgres, err := testcontainers.GenericContainer(
                ctx,
                testcontainers.GenericContainerRequest{
                        ContainerRequest: req,
                        Started:          true,
                },
        )
        if err != nil {
                log.Printf("err: %v", err)
        }
        t.Cleanup(func() {
                require.NoError(t, postgres.Terminate(ctx))
        })
```

Executing Tests Using testcontainers-go

```
$ make test
```

Side Note

This slide was created using a tool called marp. Marp is a tool that allows you to create slides using Markdown.