TestContainers in Go

Overview

> *Testcontainers-Go is a Go package that makes it simple to create and clean up container-based dependencies for automated integration/smoke tests. The clean, easy-to-use API enables developers to programmatically define containers that should be run as part of a test and clean up those resources when the test is done.*

Sample Test

component_test.go

```
package component_tests

import (
    "context"
    "fmt"
    "github.com/sirupsen/logrus"
    "github.com/stretchr/testify/assert"
    "github.com/testcontainers/testcontainers-go"
    "kafka-stream-operator/component-tests/containers"
    "kafka-stream-operator/component-tests/services"
    "os"
    "testing"
    "time"
)

var networkName = "test"

var inTopic = "ncyd_test_in"
var outTopic = "ncyd_test_out"
var kafkaStreamOperatorVersion = "22.0.0-SNAPSHOT"


var compose *testcontainers.LocalDockerCompose
var producer *services.Producer
var consumer *services.Consumer
var kafkaContainer *containers.KafkaContainer
var zookeeperContainer *containers.ZookeeperContainer
var kafkaStreamOperatorContainer *containers.KafkaStreamOperatorContainer
var network testcontainers.Network
var ctx context.Context

func TestMain(m *testing.M) {
        defer teardown()
    setup()
        m.Run()
}

func setup(){
        logrus.Info("Setup")

    //read version
    v := os.Getenv("VERSION")
    if v !="" {
        kafkaStreamOperatorVersion = v
        logrus.Infof("Setting Kafka Stream Operator Image Version to: %v", kafkaStreamOperatorVersion)
    }
```

```
   var err error
   ctx = context.Background()

   network, err = testcontainers.GenericNetwork(ctx, testcontainers.GenericNetworkRequest{
            NetworkRequest: testcontainers.NetworkRequest{
            Name:           networkName,
            CheckDuplicate: true,
            },
        })
        if err != nil {
                logrus.Fatalf("Failed to create network: %v",err)
        }

   zookeeperContainer, err = containers.NewZookeeperContainer(ctx,networkName)
    if err != nil{
       logrus.Fatalf("Failed to start zookeeper: %v", err)
    }

    kafkaContainer, err = containers.NewKafkaContainer(ctx,networkName)
    if err != nil{
       logrus.Fatalf("Failed to start kafka: %v", err)
    }

    kafkaStreamOperatorContainer, err = containers.NewKafkaStreamOperatorContainer(ctx,networkName,
kafkaStreamOperatorVersion)
    if err != nil{
       logrus.Fatalf("Failed to start kafka stream operator: %v", err)
    }

    var brokers= []string{ "localhost:9092"}

    logrus.Info("Creating Producer")
        producer, _ = services.NewProducer(brokers,inTopic)

    logrus.Info("Creating Consumer")
    consumer,_ = services.NewConsumer(brokers,inTopic)
    go consumer.Start()
    logrus.Info("Consumer Started")

}

func teardown(){
   logrus.Info("Teardown")
   kafkaStreamOperatorContainer.Terminate()
   kafkaContainer.Terminate()
   zookeeperContainer.Terminate()
   network.Remove(ctx)
   //time.Sleep(1 * time.Second)
}


func TestSuccess(t *testing.T) {
   logrus.Info("testSuccess running")
   expectedMessage := "{\"log\":\"9 - testing log for user test\", \"kubernetes\":{\"host\":\"docker-
desktop\"}}"

   //send 10 messages
   logrus.Info("Sending 10 Messages")
   for i:=0;i<10;i++ {
     message := fmt.Sprintf("{\"log\":\"%d - testing log for user test\", \"kubernetes\":{\"host\":\"docker-
desktop\"}}",i)
     err := producer.Send("",message)
     if err != nil {
       t.Errorf("Failed to send message: %v", err)
       t.Fail()
     }
   }
  logrus.Info("Messages sent")
   time.Sleep(30 * time.Second )

   logrus.Info("Reading Messages")
```

```
   outputMessage, err := consumer.Read()
   if err !=nil {
       t.Errorf("Failed to receive message: %v", err)
       t.Fail()
   }
   logrus.Infof("Received Message %v",outputMessage)

   assert.Equal(t,expectedMessage,outputMessage,"Failed to retrieve expected output")

}
```

kafka_container.go

```
package containers

import (
    "context"
    "github.com/testcontainers/testcontainers-go"
    "github.com/testcontainers/testcontainers-go/wait"
)

type KafkaContainer struct {
    Container testcontainers.Container
    context context.Context
    dockerId string

}

func NewKafkaContainer(ctx context.Context, networkName string ) (*KafkaContainer, error){

   kafkaContainer := &KafkaContainer{
       Container: nil,
       context: ctx,
   }


   req := testcontainers.ContainerRequest{

       Image:           "bitnami/kafka:3" ,
       Name:            "kafka",
       Hostname:        "kafka",
       Networks:        []string{ networkName},
       WaitingFor:      wait.ForLog("Ready to serve as the new controller with epoch 1"),
       ExposedPorts:    []string{"9092:9092"},
       Env:             map[string]string{
                           "KAFKA_CFG_ZOOKEEPER_CONNECT": "zookeeper:2181",
                           "ALLOW_PLAINTEXT_LISTENER": "yes",
                           "KAFKA_CFG_LISTENERS": "INTERNAL://:9093,EXTERNAL://:9092",
                           "KAFKA_CFG_ADVERTISED_LISTENERS": "INTERNAL://:9093,EXTERNAL://localhost:9092",
                           "KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP": "INTERNAL:PLAINTEXT,EXTERNAL:PLAINTEXT",
                           "KAFKA_CFG_INTER_BROKER_LISTENER_NAME": "INTERNAL",
                       },
   }
   container, err := testcontainers.GenericContainer(ctx, testcontainers.GenericContainerRequest{
       ContainerRequest: req,
       Started:          true,
   })
   if err != nil {
       return nil, err
   }

   kafkaContainer.Container =  container

   return kafkaContainer,nil
}

func (k KafkaContainer) Terminate() {
        k.Container.Terminate(k.context)
        //time.Sleep(1 * time.Second)
}
```

zookeeper_container.go

```
package containers

import (
    "context"
    "github.com/testcontainers/testcontainers-go"
    "github.com/testcontainers/testcontainers-go/wait"
)

type ZookeeperContainer struct {
    container testcontainers.Container
    context context.Context
    dockerId string

}

func NewZookeeperContainer(ctx context.Context, networkName string) (*ZookeeperContainer, error){

    zookeeperContainer := &ZookeeperContainer{
        container: nil,
        context: ctx,
    }


    req := testcontainers.ContainerRequest{

        Image:              "bitnami/zookeeper:3.7",
        Name:               "zookeeper",
        Hostname:           "zookeeper",
        Networks:           []string{ networkName},
        WaitingFor:         wait.ForLog("Started AdminServer on address"),
        ExposedPorts:       []string{"2181/tcp"},
        Env:                map[string]string{
                                "ALLOW_ANONYMOUS_LOGIN": "yes",
                            },
    }
    container, err := testcontainers.GenericContainer(ctx, testcontainers.GenericContainerRequest{
        ContainerRequest: req,
        Started:          true,
    })
    if err != nil {
        return nil, err
    }

    zookeeperContainer.container =  container

    return zookeeperContainer,nil
}

func (z ZookeeperContainer) Terminate() {
         z.container.Terminate(z.context)
         //time.Sleep(1 * time.Second)
}
```

Sample Test Case using Docker-Compose

```
package test

import (
        "github.com/google/uuid"
```

```
        "github.com/sirupsen/logrus"
        "github.com/testcontainers/testcontainers-go"
        "github.com/testcontainers/testcontainers-go/wait"
        "kafka-stream-operator/src/internal/gateways/kafka/publisher"
        "log"
        "strings"
        "testing"
        "time"
)

var kafkaPublisher         *publisher.Publisher
var bootstrapURL = "localhost:9092"
var compose *testcontainers.LocalDockerCompose
var testTopic = "test-in"

func TestMain(m *testing.M) {
        defer teardown()
    setup()
        m.Run()
}

func setup(){
        log.Println("Setup")
    compose = testcontainers.NewLocalDockerCompose([]string{"../../docker-compose.yml"}
                , strings.ToLower(uuid.New().String()))
        compose.WaitForService("kafka_1",wait.ForLog("Ready to serve as the new controller with epoch 1"))
        compose.WithCommand([]string{"up", "-d"}).Invoke()
    logrus.Info("Containers started! ")

        var err error
        kafkaPublisher, err = publisher.NewPublisher(bootstrapURL, testTopic)
        if err !=nil {
                logrus.Errorf("Failed to start publisher: %v", err)
        }

}

func teardown(){
        log.Println("Teardown")
   compose.Down()
   time.Sleep(1 * time.Second)
}


func sendMessage(message string) error{

   err := kafkaPublisher.Publish("", message)
        if err !=nil {
                logrus.Errorf("Failed to send mesaage: %v", message)
       return err
        }
   return nil
}


func TestSuccess(t *testing.T) {
   log.Println("testSuccess running")

   message := `{"log":"testing log for user test", "kubernetes":{"host":"docker-desktop"}}`

   err := sendMessage(message)
   if err != nil {
       t.Errorf("Failed to send message: %v", err)
       t.Fail()
   }
   err = sendMessage(message)
   if err != nil {
       t.Errorf("Failed to send message: %v", err)
       t.Fail()
   }
```

```
}
```

Docker-Compose

```
version: "2"

services:
  zookeeper:
    image: bitnami/zookeeper:3.7
    ports:
      - "2181:2181"
    environment:
      - ALLOW_ANONYMOUS_LOGIN=yes

  kafka:
    image: docker.io/bitnami/kafka:3
    ports:
      - "9092:9092"
    environment:
      - KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181
      - ALLOW_PLAINTEXT_LISTENER=yes
      - KAFKA_CFG_LISTENERS=INTERNAL://:9093,EXTERNAL://:9092
      - KAFKA_CFG_ADVERTISED_LISTENERS=INTERNAL://:9093,EXTERNAL://localhost:9092
      - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=INTERNAL:PLAINTEXT,EXTERNAL:PLAINTEXT
      - KAFKA_CFG_INTER_BROKER_LISTENER_NAME=INTERNAL
    depends_on:
      - zookeeper

  kowl:
    image: rsmnarts/kowl:latest
    restart: on-failure
    ports:
      - "8080:8080"
    environment:
      - KAFKA_BROKERS=kafka:9093
    depends_on:
      - kafka
```

References

Reference	URL
Testcontainers-Go	https://golang.testcontainers.org/
Using Docker Compose	https://golang.testcontainers.org/features/docker_compose/
Kafka Test Containers with GoLang	https://medium.com/trendyol-tech/kafka-test-containers-with-golang-b85e4b2469db