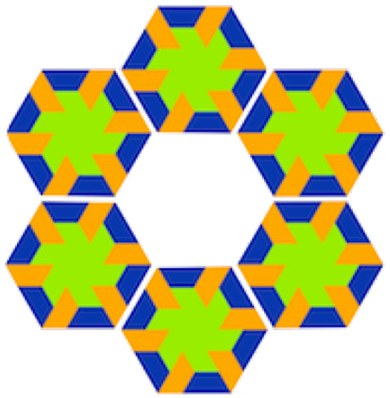
REST API

testing Go with Docker and Go

Ernest Micklei

devFest 2016, Amsterdam

bol.com



Ernest Micklei

Software Architect

Apollo @ bol.com

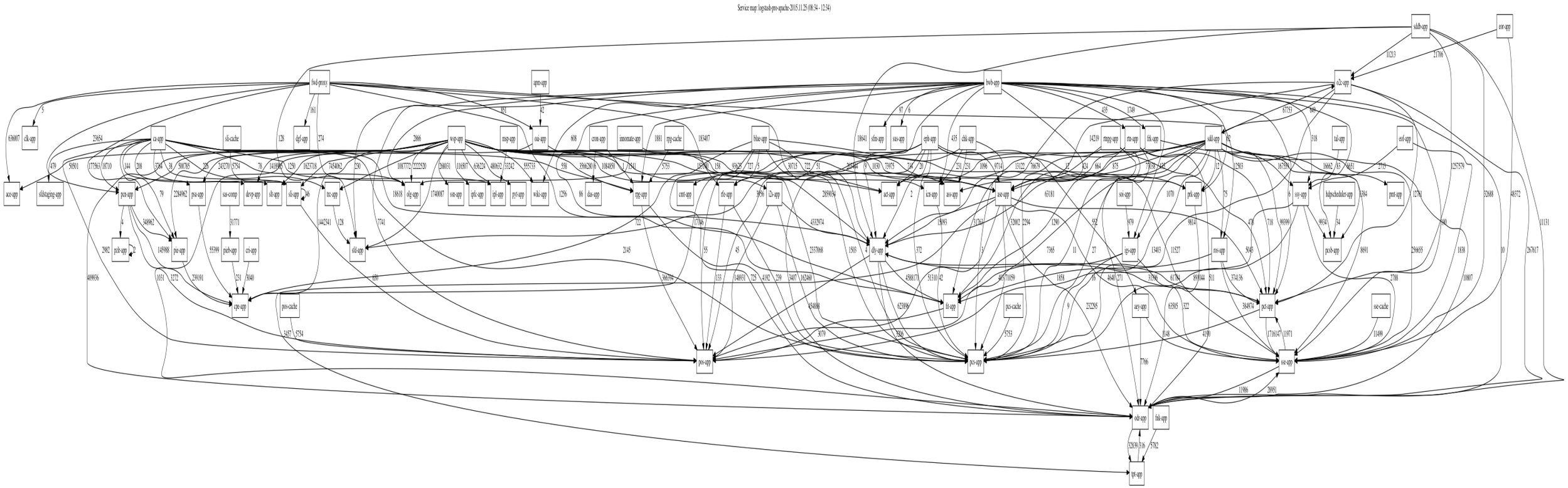
platform services

We are bol.com

Technology focus

- 100+ microservices
- REST (98% Java, 2% Go)
- ProtocolBuffers, JSON (XML for testing only)
- Oracle, PostgresSQL, MongoDB, Cassandra, HBase, etcd, consul, ElasticSearch

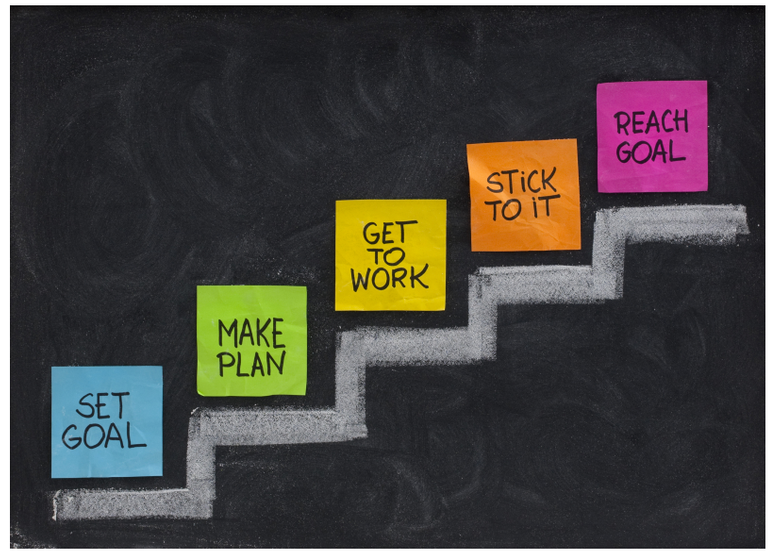
Services

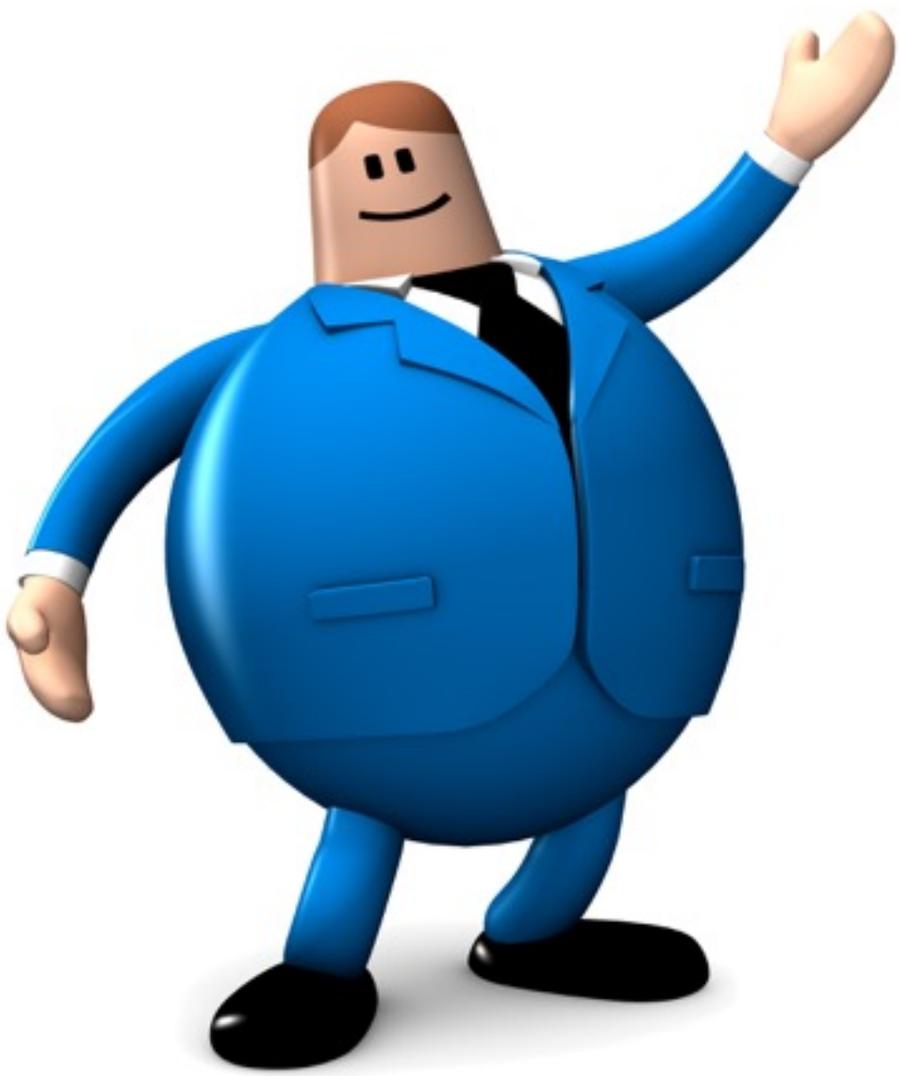


the new monolith?

Testing goals

- fast feedback loop
- high code coverage
- minimal external dependencies
- cheap to maintain tests
- local runnable & CD pipeline





code
coverage

Coverage

- happy execution paths
- simulate failure conditions for the other paths
- measure the coverage

Real testing

- more confident about quality
- higher coverage on external access
- focus shift + less unit testing code (no mocks)

requires

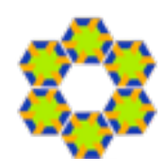
- incorporate external dependencies

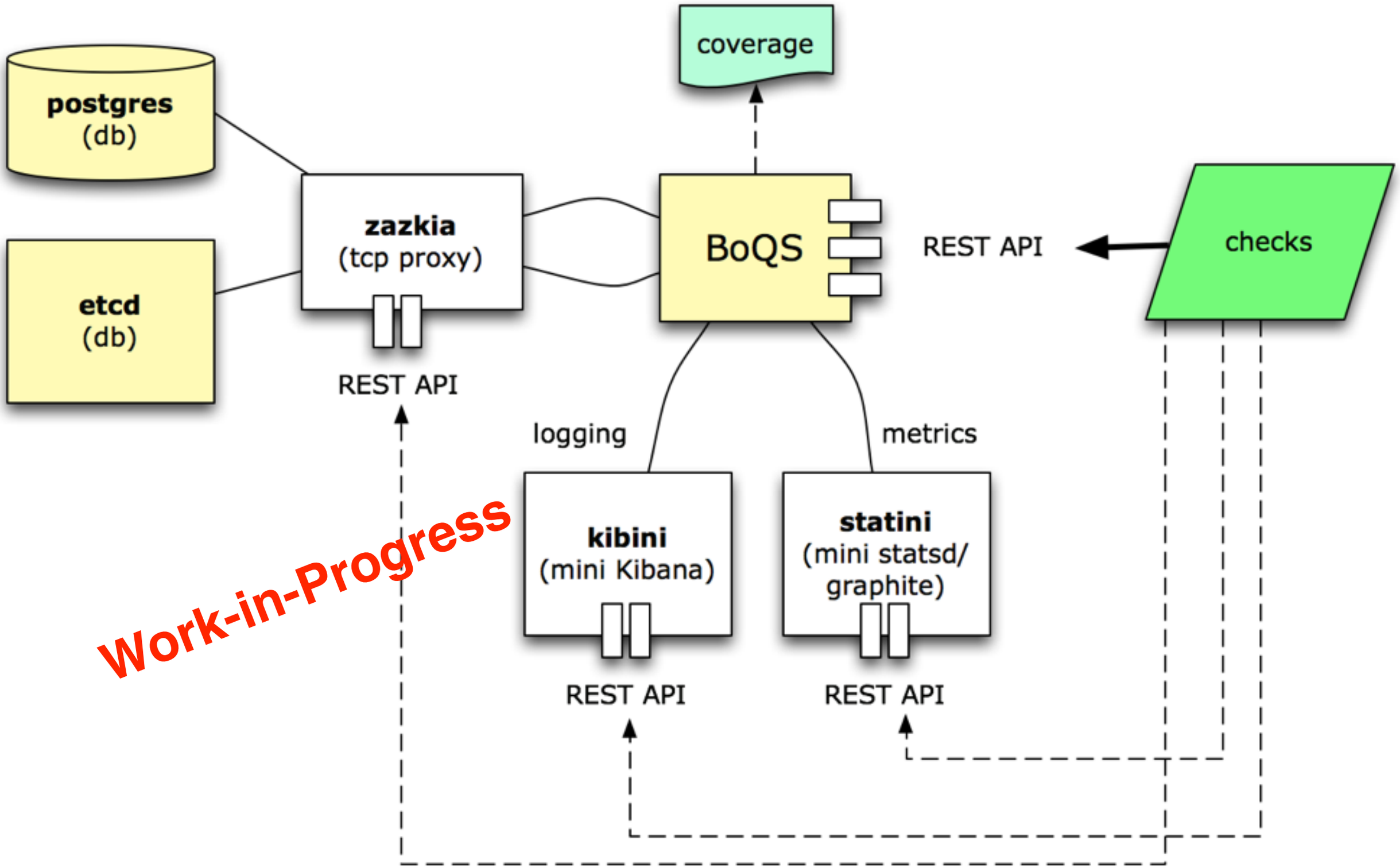
Explore limits

- numeric value ranges
- text length
- character encodings
- payload size
- wrong | missing HTTP headers

Inject failure

- backend connectivity (timeouts, unreachable)
- backend errors (HTTP 500)
- testing resilience



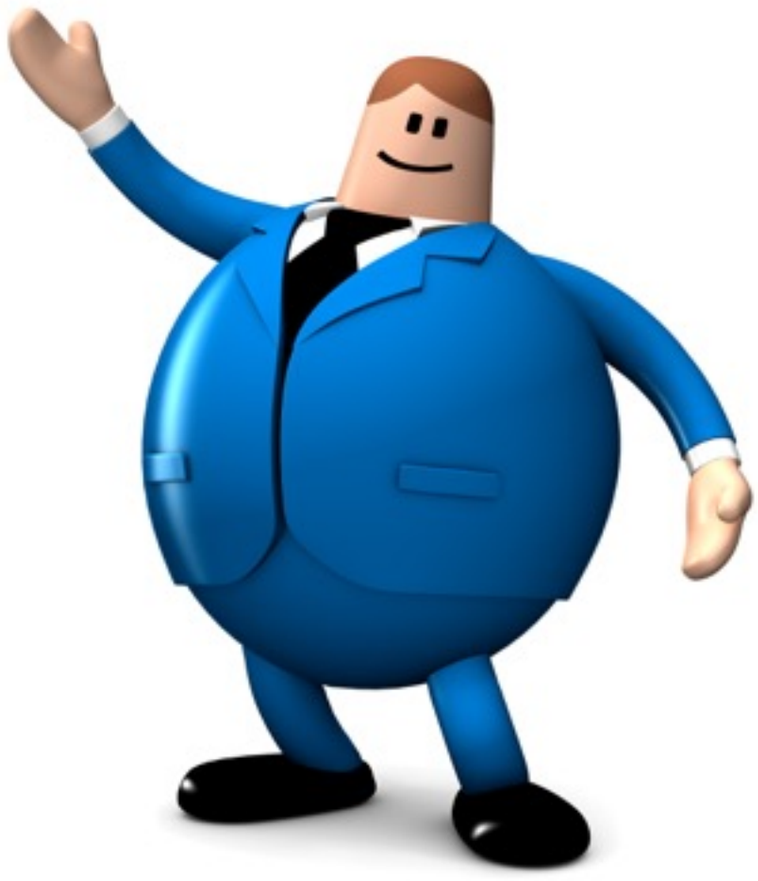

coverage
postgres
(db)
etcd
(db)
zazkia
(tcp proxy)
REST API
BoQS
REST API
checks
logging
metrics
kibini
(mini Kibana)
statini
(mini statsd/
graphite)
REST API
REST API
Work-in-Progress

Measure

- Many test frameworks support code coverage
- Go has support for coverage annotated binaries
- Running API checks -> coverage report
  - https://www.elastic.co/blog/code-coverage-for-your-golang-system-tests

```go
package main

// This file is mandatory as otherwise the filebeat.test binary
// is not generated correctly.
import (
    "flag"
    "testing"
)

var systemTest *bool

func init() {
    systemTest = flag.Bool("systemTest", false,
        "Set to true when running system tests")
}

// Test started when the test binary is started. Only calls main.
func TestSystem(t *testing.T) {
    if *systemTest {
        main()
    }
}
```


system & infra dependencies



no more TAXP

- run checks against your system in an isolated environment
- manage your own test data
- temporary environment using containers

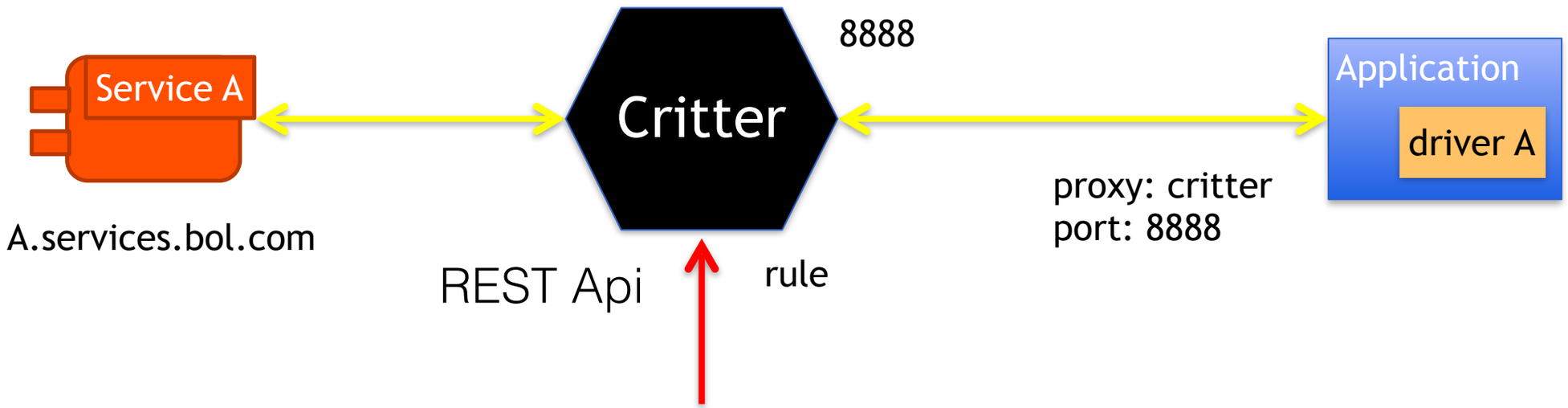
- run dependent systems in their own container
- run system under test in container
- docker compose
- Go scripts for complex setups

if not dockerizable

- start application with Go/Java stubs
- connect to a staging environment…
- stubbing as a service (e.g. Critter)

Critter

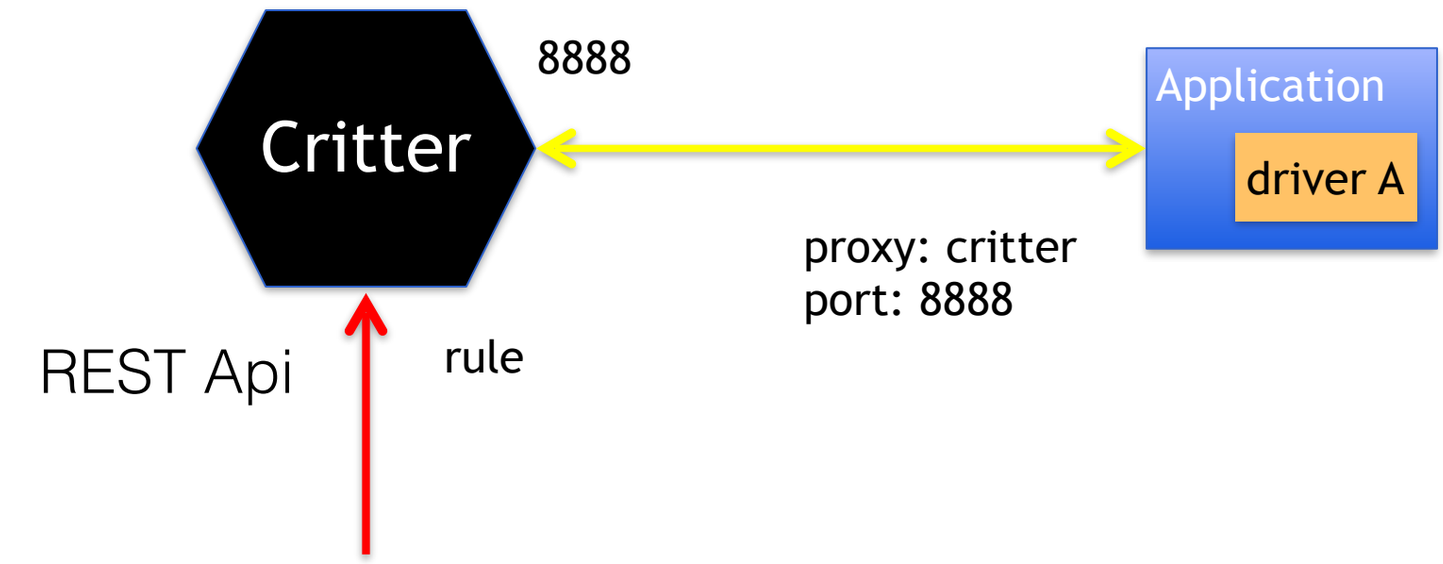
HTTP testing proxy for testing



IF
request for service A
THEN
return Http Status 500

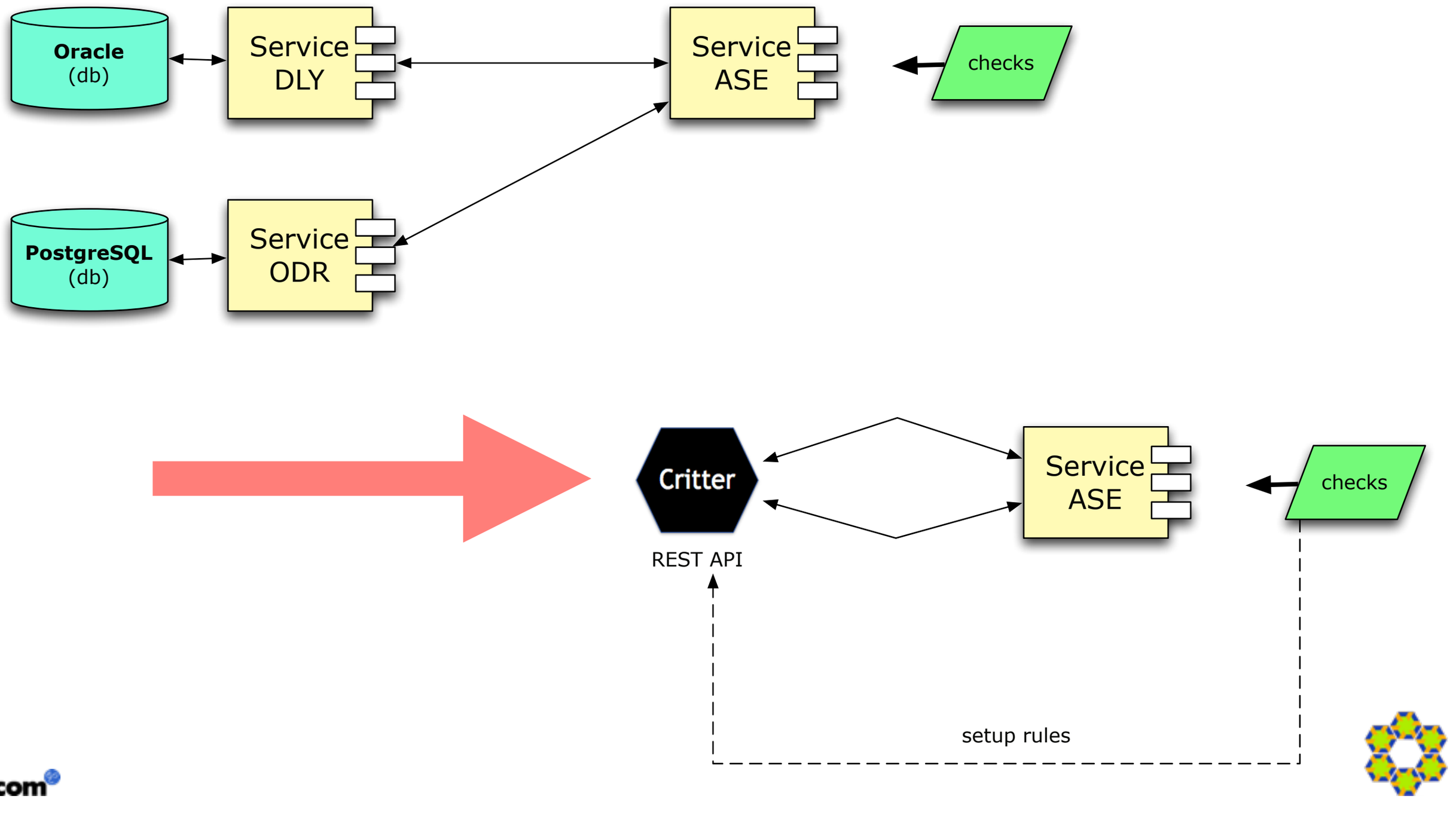
Critter

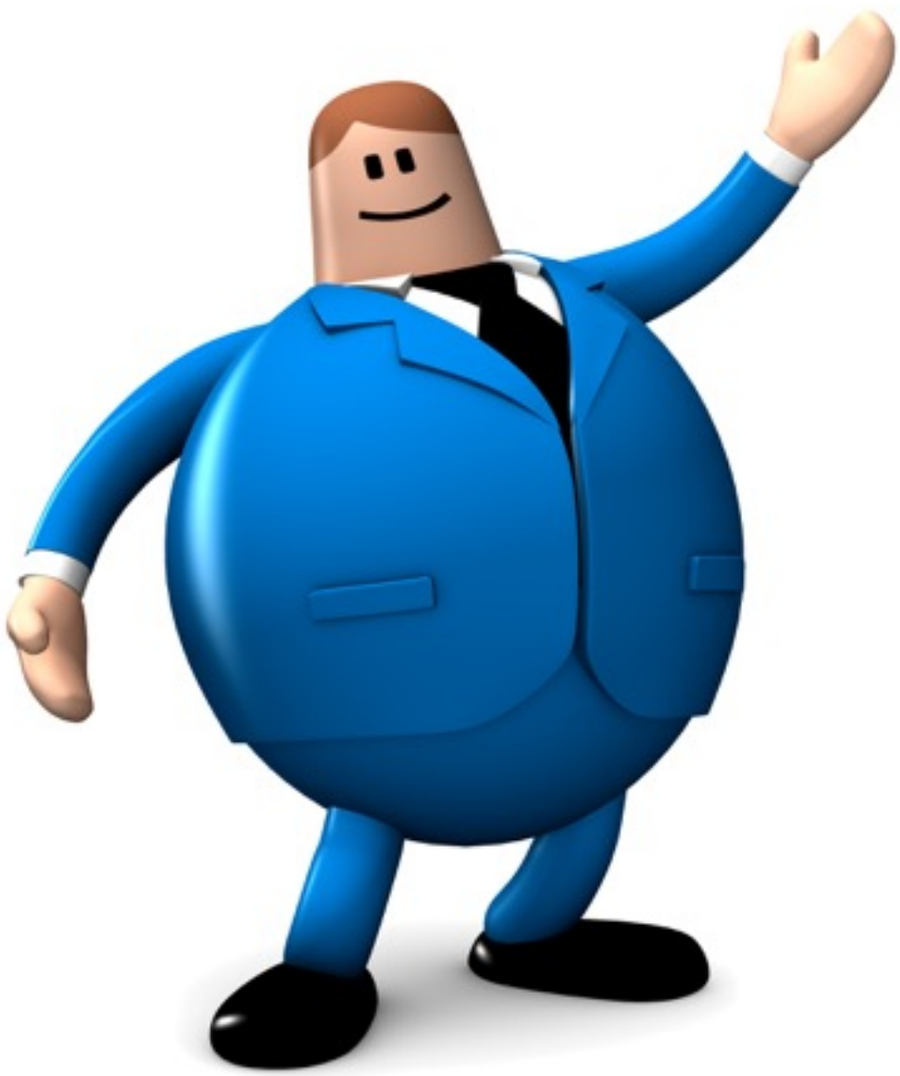
HTTP testing proxy for testing




```
IF
        request for service A
THEN
        return <some-prepared-payload/>
```

Stub service





test tools

Tooling @ bol

- JUnit with Grizzly JAX-RS container
- Fitnesse (web, Java fixtures)
- Cucumber
- **forest (go test)**
- JMeter (desktop tool, XML)
- SoapUI
- Gatling (Scala, DSL)

Program your checks

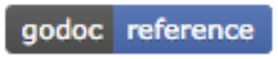
- Prefer same language
- Create abstractions to improve readability
- Provide detailed logging
- Run API tests as part of build

POST with http pkg

```
client := new(http.Client)
payload := strings.NewReader("key=value")
req , err := http.NewRequest("POST",
    "http://localhost:12346/v1/properties",
    payload)
if err != nil {
    log.Fatal("invalid url:%v",err)
}
req.Header.Add("Content-Type","text/plain")
req.Header.Add("Accept","application/json")
resp, err := client.Do(req)
if err != nil {
    log.Fatal("failed to send request:%v",err)
}
defer resp.Body.Close()
body, err := ioutil.ReadAll(resp.Body)
if err != nil {
    log.Fatal("failed to read response:%v",err)
}
if resp.StatusCode != http.StatusNoContent {
    var serviceError ServiceError
    err = json.Unmarshal(body, &serviceError )
}
...
```


forest - for testing REST api-s in Go

This package provides a few simple helper types and functions to create functional tests that call a running REST based WebService.



Introduction Blog Post

forest , Go package

```go
package main

import (
    "net/http"
    "testing"

    . "github.com/emicklei/forest"
)

var github = NewClient("https://api.github.com", new(http.Client))

func TestForestProjectExists(t *testing.T) {
    cfg := NewConfig("/repos/emicklei/{repo}", "forest").Header("Accept", "application/json")
    r := github.GET(t, cfg)
    ExpectStatus(t, r, 200)
}
```

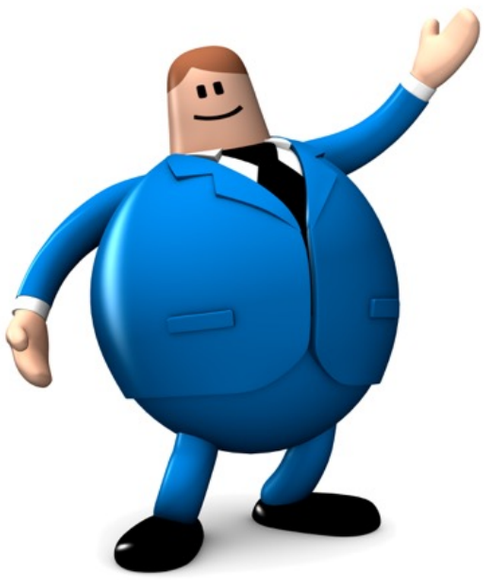

forest

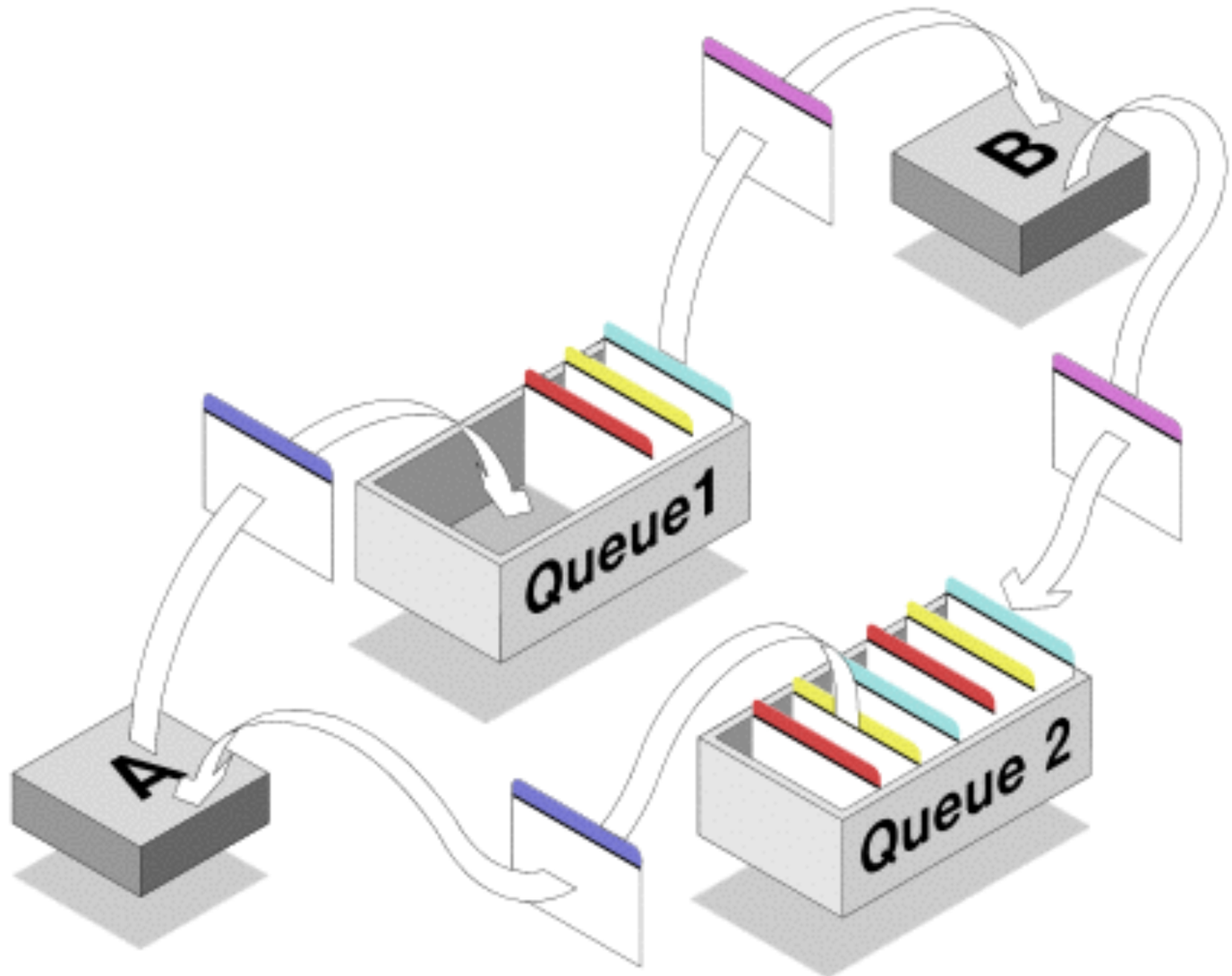
```
func Test_create_a_public_topic(t *testing.T) {
    SkipUnless(t, "apitest")
    topicName := "test_topic_new_public"

    When(t, "I create a new topic")
    defer cleanUpTopic(t, topicName, defaultToken)
    r := createTopic(t, topicName, defaultToken, IsPublic)

    Then(t, "I expect this was successful")
    ExpectStatus(t, r, http.StatusNoContent)
}
```


```
=== RUN   Test_create_a_public_topic
--- PASS: Test_create_a_public_topic (0.00s)
        bddl.go:21:  When I create a new topic
        api_testing.go:58:
                PUT http://localhost:8060/v1/topics/test_
wMjY2MDAsImlhdCI6MTQ3NTY2NjYwMCwiaWRlbnRpdHkiOiJkZWZhdWx0
cept:[application/json]]
        bddl.go:21:  Then I expect this was successful
        api_testing.go:28:
                GET http://localhost:8060/v1/topics map[C
IkpXVCJ9.eyJleHAiOjE3OTEwMjY2MDAsImlhdCI6MTQ3NTY2NjYwMCwi
        api_testing.go:73:
                DELETE http://localhost:8060/v1/topics/te
zation:[eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3
YUxzLN6eZP0jC8x_iWf1yk8c]]
        utils.go:128: delete topic
PASS
run teardown
ok      github.com/bolcom/boqs/systemtest       0.025s
```


BQS

platform service
at bol.com





API per stakeholder

- Applications use the **public API**
- Testers and checks use the **testing API**
- DevOps use the **admin API**

Swagger UI

boqs.services.test2.bol.com/internal/apidocs/

BoQS

topics : Create topics and publish messages with a topic.

Show/Hide | List Operations | Expand Operations | Raw

GET	/v1/topics	Return all topics
PUT	/v1/topics/{name}	Create a topic. Max length = 255
DELETE	/v1/topics/{name}	Delete a topic.
POST	/v1/topics/{name}/messages	Publish a message with a topic. All query parameters are handled as message properties. Maximum length of the **name** is 255 chars. Maximum length of the **value** is 255 chars. Maximum number of properties is 25. Maximum size of the content (compressed or not) is 10 MiB.

subscriptions : Subscribe to topics.

Show/Hide | List Operations | Expand Operations | Raw

GET	/v1/subscriptions	Return all subscriptions
PUT	/v1/subscriptions/{topic}/to/{queue}	Create a subscription (topic -> queue)
DELETE	/v1/subscriptions/{topic}/to/{queue}	Delete a subscription (topic -> queue)

queues : Consuming messages from queues.

Show/Hide | List Operations | Expand Operations | Raw

GET	/v1/queues	Return all queues
GET	/v1/queues/{name}/messages	Lease a set of messages from the queue
DELETE	/v1/queues/{name}/messages	Purge all messages from the queue
POST	/v1/queues/{name}/messages/{id}/receipts/{handle}	Handle the result of processing a leaseed message from the queue. Sending this request without a body (MessageProcessError in JSON) then the message will deleted ; it has been consumed. If a MessageProcessError is send then the message will be transferred to the associated failures queue.
PUT	/v1/queues/{name}	Create an queue. Max length = 255
DELETE	/v1/queues/{name}	Delete a queue

failures : Process failed messages from queues.

Show/Hide | List Operations | Expand Operations | Raw

Test API

queues : [test] Queue inspection and troubleshooting.

Show/Hide | List Operations | Expand Operations | Raw

GET	/v1/test/queues/{name}/traced-messages/{trace_id}	Find the messages with a given trace_id
GET	/v1/test/queues/{name}/messages	Return the next available set of messages on the queue without leasing them ; no handle is set
GET	/v1/test/queues/{name}/messages/counts	Number of messages stored in the queue per store.
GET	/v1/test/queues/{name}/messages/counts/total	Number of messages stored in the queue.
GET	/v1/test/queues/{name}/messages/longest-waiting	Wait duration of the oldest message
GET	/v1/test/queues/{name}/inspected-messages/{id}	Inspect a message including its actual payload. It will not be leased.

failures : [test] Failures inspection and troubleshooting.

Show/Hide | List Operations | Expand Operations | Raw

GET	/v1/test/failures/{name}/messages/counts	Number of failed messages stored in the queue per store
GET	/v1/test/failures/{name}/messages/counts/total	Number of failed messages stored in the queue
GET	/v1/test/failures/{name}/traced-messages/{trace_id}	Find the failed messages with a given trace_id

topics : [test] Topics inspection and troubleshooting.

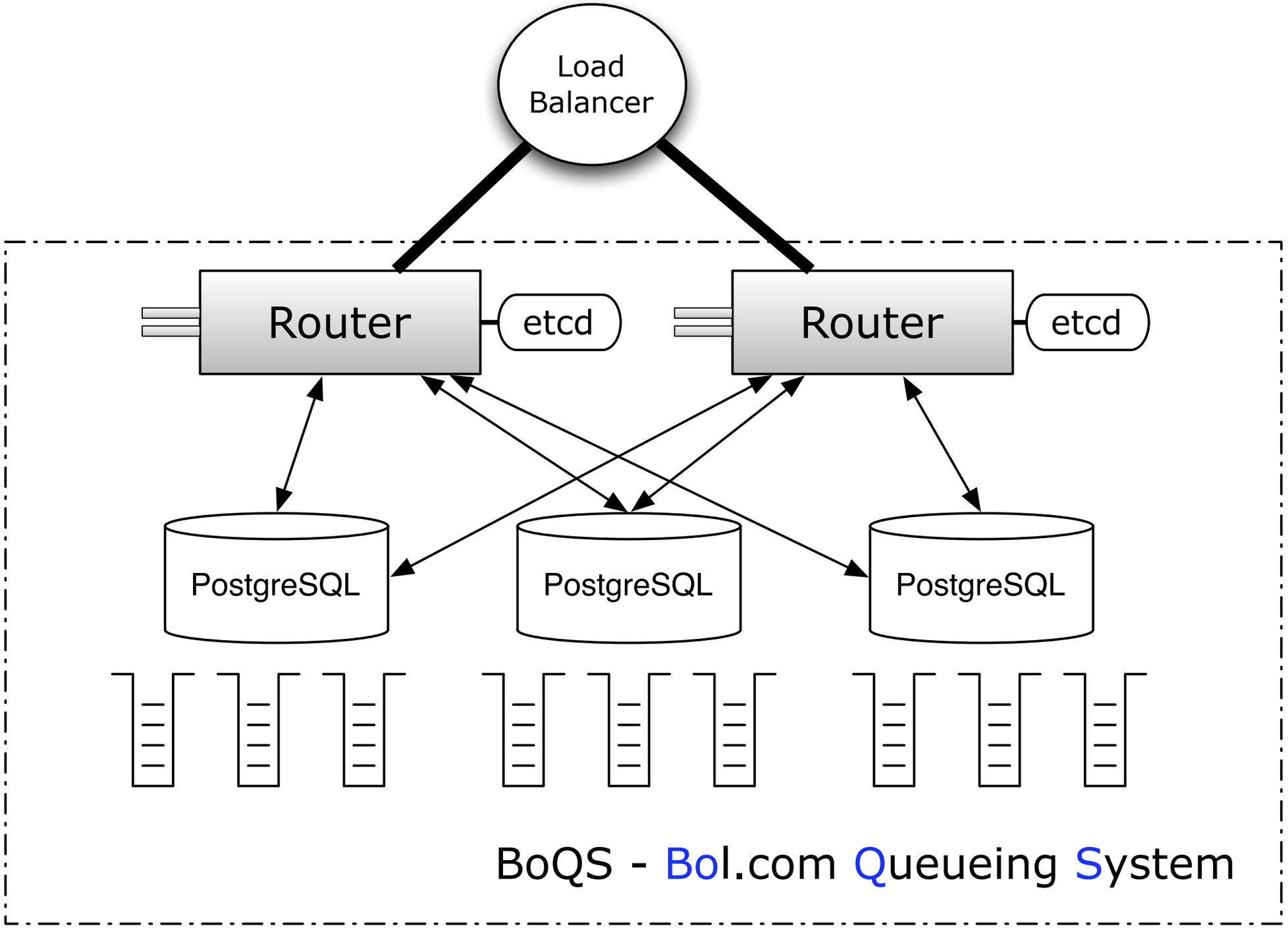
Show/Hide | List Operations | Expand Operations | Raw

GET	/v1/test/topics/{name}/messages/counts	Number of messages stored in queues as a result of publishing on this topic.

API driven tests

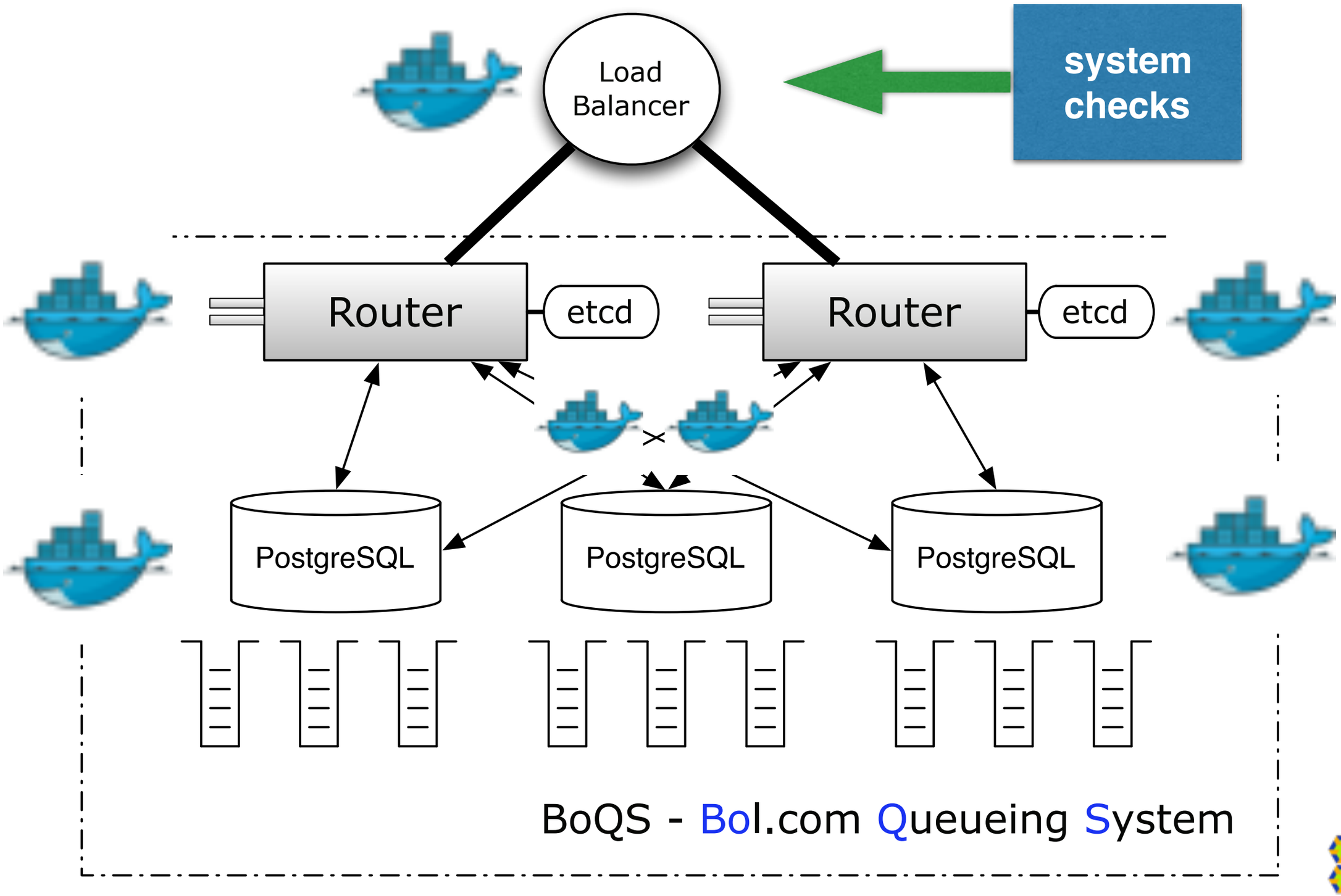
- encapsulates the persistent store(s)
- setup and teardown is controlled by the checks
- test data control
- applications can use the same system for their own integration tests where this system is a dependent.

Architecture



BoQS

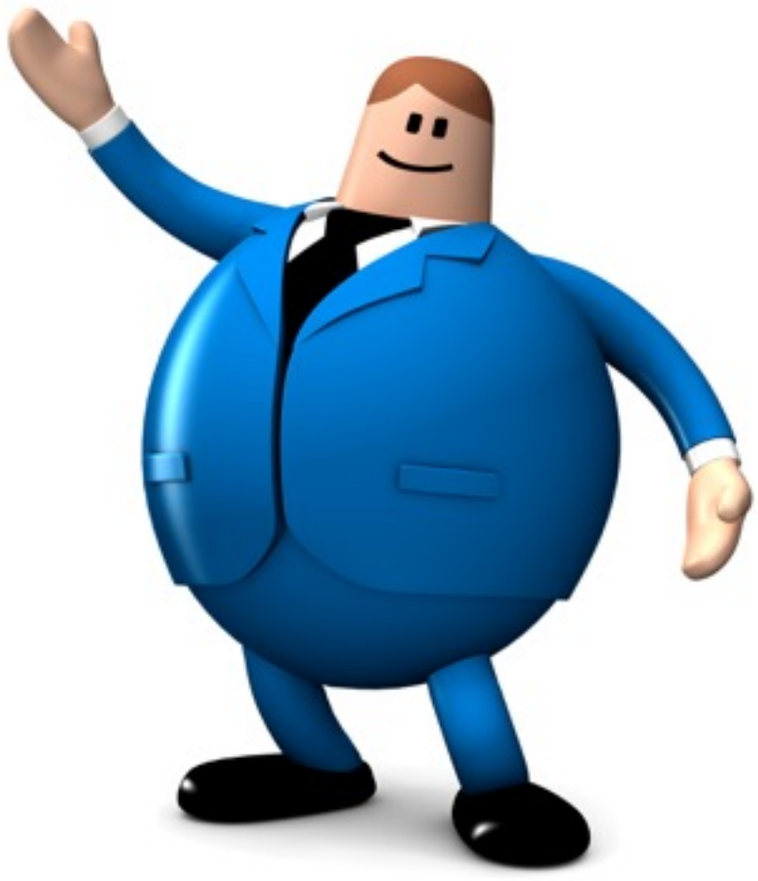
dockerize it



bol.com

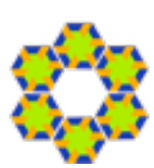
DEMO

time



Testing

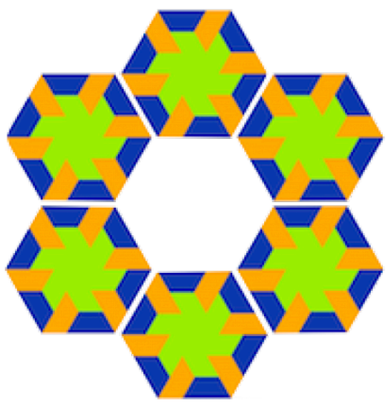
- fast feedback loop **Go test**
- code coverage **Go cov**
- minimal external dependencies  Critter
- cheap to maintain tests **Go**
- local runnable & CD pipeline 



all tests, all the time

- Setting up the environment
- Running the tests
- Detailed reporting on failures

Thanks



ernestmicklei.com

github.com/emicklei/critter

github.com/emicklei/zazkia

github.com/emicklei/forest