




Ceci n'est pas une pipe.

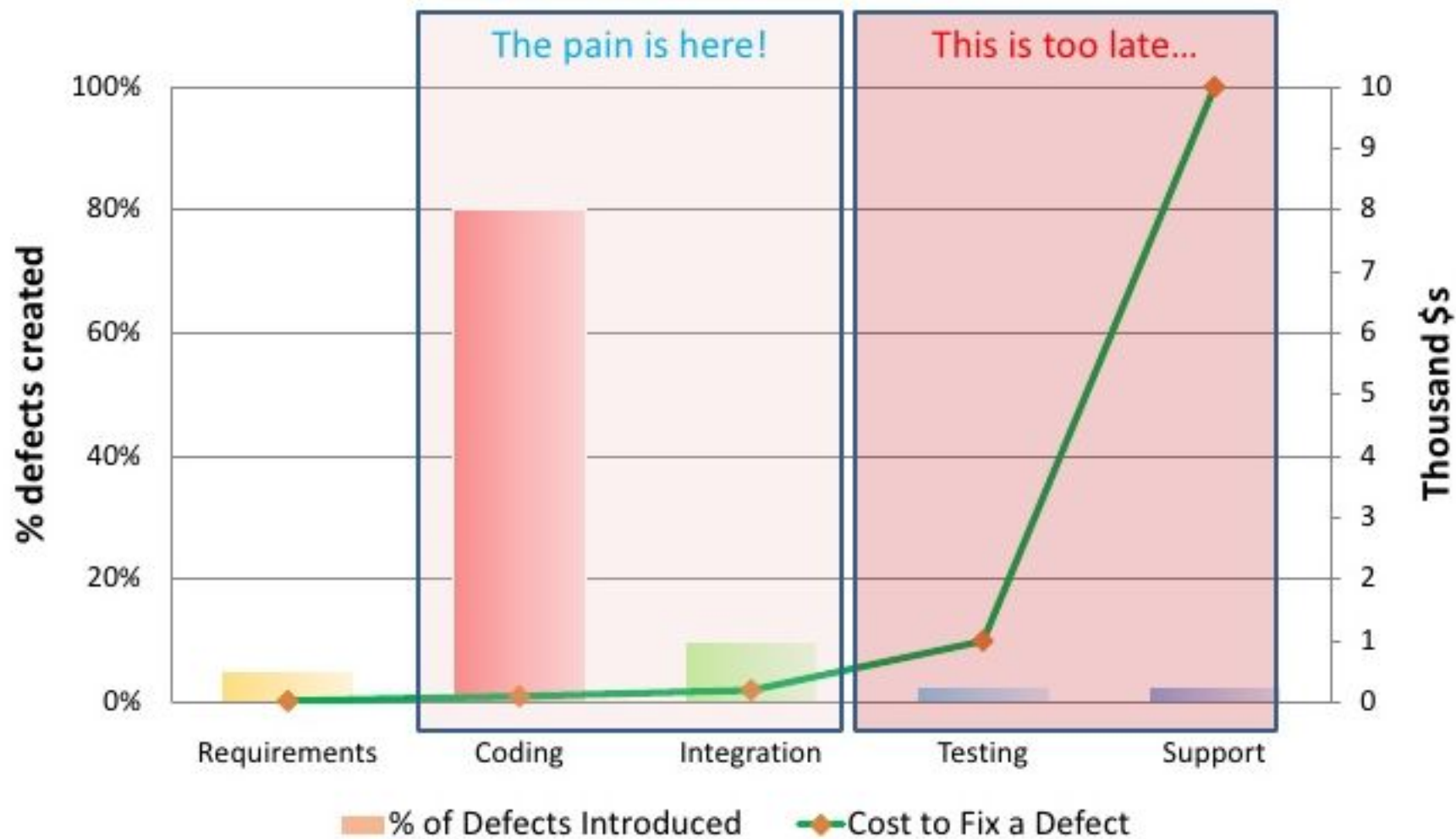


This is not a pipe
HTTP Mocking and more

Zoltán Domahidi
senior Java developer

- 
- A wooden easel with a white sign attached to it. The sign has a list of four items, each preceded by a red dot. The text is in a red, cursive font. The background is a solid light beige color.
- *There's no place like 127.0.0.1*
 - *HTTP Mocking*
 - *Wiremock*
 - *Mountebank*

There is no place like 127.0.0.1

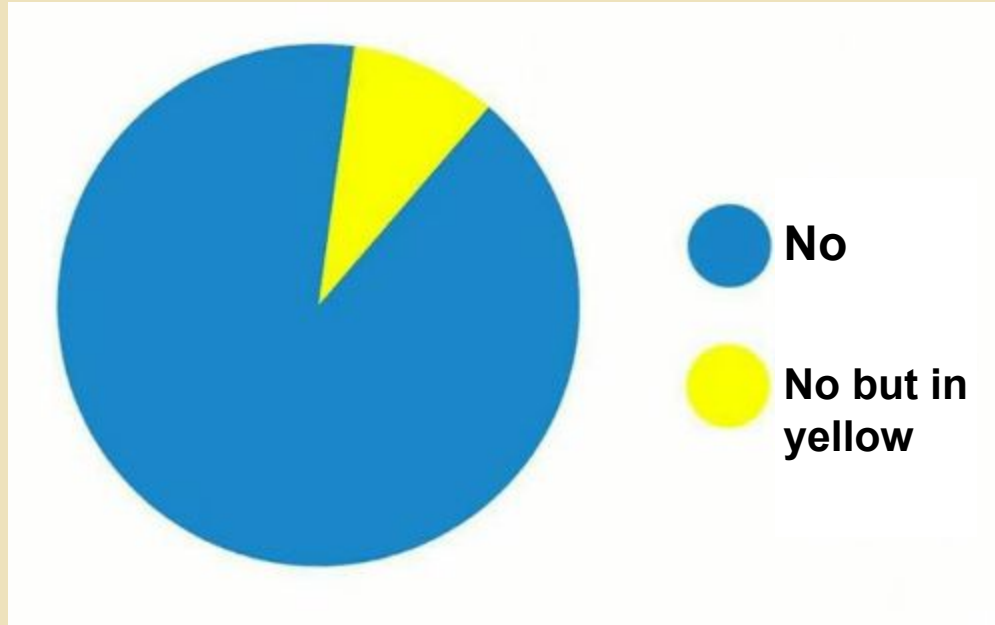


Reasons to stay Home...

- cheap
- control
- fast
- easy to set up
- independence



Leaving Home?



...but you have to!

- tests are not enough
- different versions
- different configurations
- permissions and authorizations
- different operating systems
- out-of-date upstream expectations



- how to use same versions team/project wide?
- how to keep them in sync?
- how to stay close to production environment?
- how to set up dependencies on each machine?
- how to minimize the time and effort?
- how to document?



The answer is...

42

The answer is...

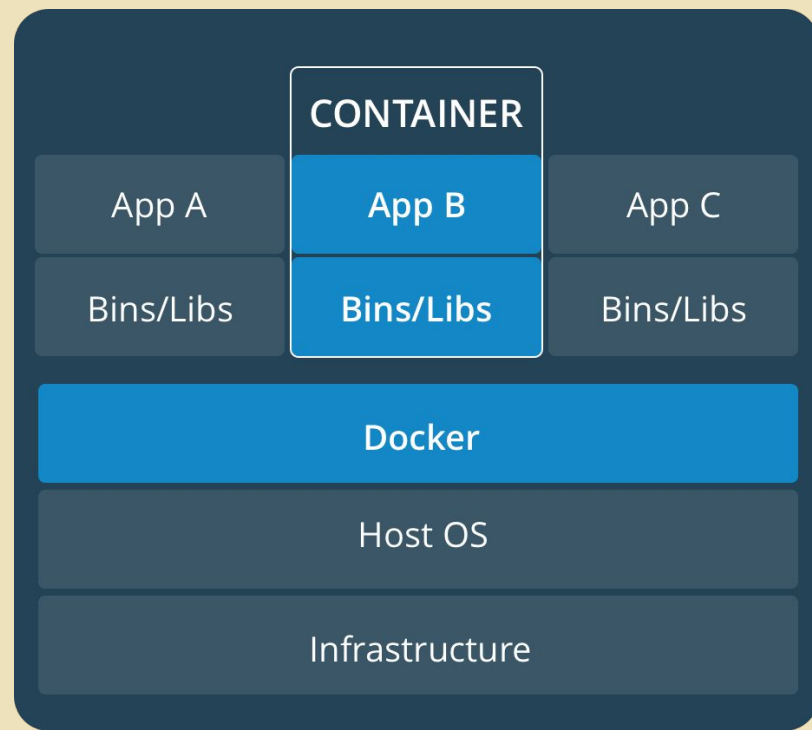
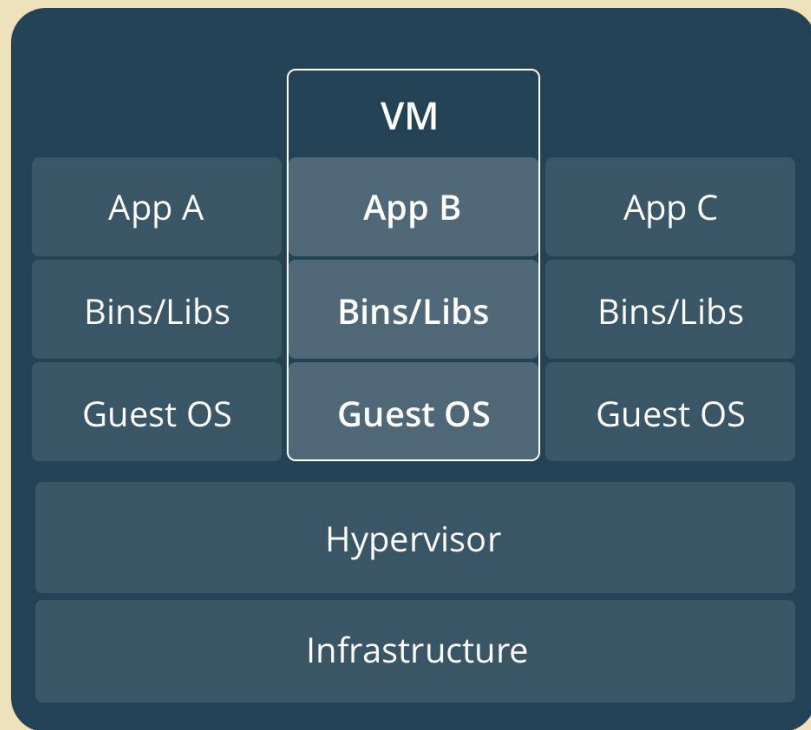
Version Control



Scripts and other text based files could be...

- version controlled
- used as documentation
- shared
- easily updated and diffed
- branched and merged
- automated
- used with command-line tools
- ...and they are lightweight

Docker



Docker

- containers
- lightweight compared to VMs
- don't need to install applications/services
- easy to change versions
- easy to run multiple instances
- same image runs on Linux/Mac/Win
- works on local machine/dev env./CI workflows
- use public images or create custom ones using Dockerfile
- cattle not pets



docker

Docker

● docker
Search term

● vmware
Search term

+ Add comparison

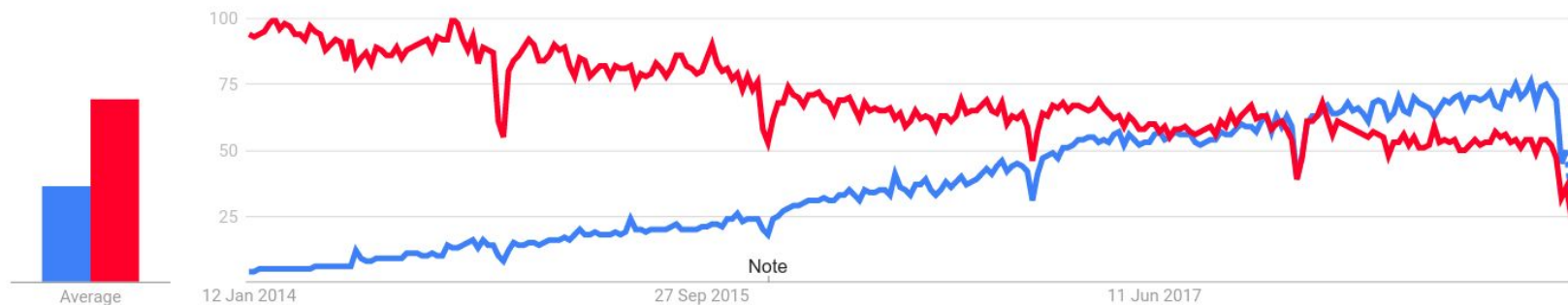
Worldwide ▼

Past 5 years ▼

All categories ▼

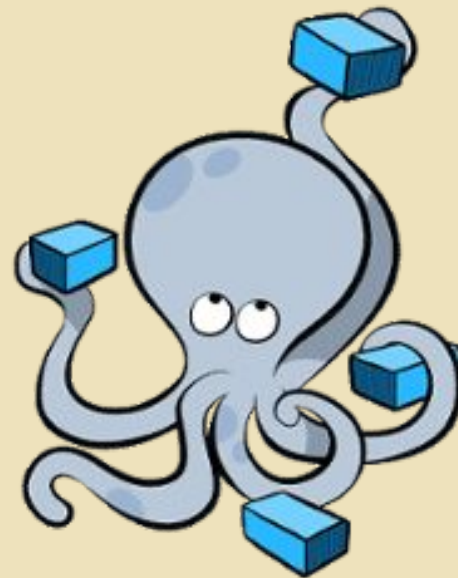
Web Search ▼

Interest over time ⓘ



Docker Compose

- container orchestration
- fast and easy
- no installers/uninstallers
- YAML descriptor:
 - document application dependencies
 - easy version update
 - easy project-wide synchronization
 - easy diff



docker-compose.yml

```
version: '3.6'
services:
  db:
    image: mysql:${MYSQL_VERSION}
    restart: on-failure
    container_name: ${SERVICE_NAME}-db
    environment:
      MYSQL_ROOT_PASSWORD: pass
      MYSQL_DATABASE: test-database
    ports:
      - 3306:3306
    volumes:
      - ./db/schema.sql:/docker-entrypoint-initdb.d/00_schema.sql
      - ./db/data.sql:/docker-entrypoint-initdb.d/01_data.sql
      - ./localdata/mysql:/var/lib/mysql
    env_file:
      - ./env
```

docker-compose.yml

version: '3.6'

services:

db:

image: mysql:\${MYSQL_VERSION}

restart: on-failure

container_name: \${SERVICE_NAME}-db

environment:

MYSQL_ROOT_PASSWORD: pass

MYSQL_DATABASE: test-database

ports:

- 3306:3306

volumes:

- ./db/schema.sql:/docker-entrypoint-initdb.d/00_schema.sql

- ./db/data.sql:/docker-entrypoint-initdb.d/01_data.sql

- ./localdata/mysql:/var/lib/mysql

env_file:

- ./env

docker-compose.yml

```
version: '3.6'
services:
  db:
    image: mysql:${MYSQL_VERSION}
    restart: on-failure
    container_name: ${SERVICE_NAME}-db
    environment:
      MYSQL_ROOT_PASSWORD: pass
      MYSQL_DATABASE: test-database
    ports:
      - 3306:3306
    volumes:
      - ./db/schema.sql:/docker-entrypoint-initdb.d/00_schema.sql
      - ./db/data.sql:/docker-entrypoint-initdb.d/01_data.sql
      - ./localdata/mysql:/var/lib/mysql
  env_file:
    - ./env
```


docker-compose.yml

```
version: '3.6'
services:
  db:
    image: mysql:${MYSQL_VERSION}
    restart: on-failure
    container_name: ${SERVICE_NAME}-db
    environment:
      MYSQL_ROOT_PASSWORD: pass
      MYSQL_DATABASE: test-database
    ports:
      - 3306:3306
    volumes:
      - ./db/schema.sql:/docker-entrypoint-initdb.d/00_schema.sql
      - ./db/data.sql:/docker-entrypoint-initdb.d/01_data.sql
      - ./localdata/mysql:/var/lib/mysql
    env_file:
      - ./env
```

docker-compose.yml

```
version: '3.6'
services:
  db:
    image: mysql:${MYSQL_VERSION}
    restart: on-failure
    container_name: ${SERVICE_NAME}-db
    environment:
      MYSQL_ROOT_PASSWORD: pass
      MYSQL_DATABASE: test-database
    ports:
      - 3306:3306
    volumes:
      - ./db/schema.sql:/docker-entrypoint-initdb.d/00_schema.sql
      - ./db/data.sql:/docker-entrypoint-initdb.d/01_data.sql
      - ./localdata/mysql:/var/lib/mysql
    env_file:
      - ./env
```

docker-compose.yml

```
version: '3.6'
services:
  db:
    image: mysql:${MYSQL_VERSION}
    restart: on-failure
    container_name: ${SERVICE_NAME}-db
    environment:
      MYSQL_ROOT_PASSWORD: pass
      MYSQL_DATABASE: test-database
    ports:
      - 3306:3306
    volumes:
      - ./db/schema.sql:/docker-entrypoint-initdb.d/00_schema.sql
      - ./db/data.sql:/docker-entrypoint-initdb.d/01_data.sql
      - ./localdata/mysql:/var/lib/mysql
    env_file:
      - ./env
```

Scripts To Rule Them All

<https://github.com/github/scripts-to-rule-them-all>

- script pattern used at GitHub
- all projects have the same scripts
- contributors can bootstrap the project by knowing only the pattern
- the scripts are maintained by the project authors having the knowledge
- each script is a unit of work, so they can be called from other scripts

Scripts To Rule Them All

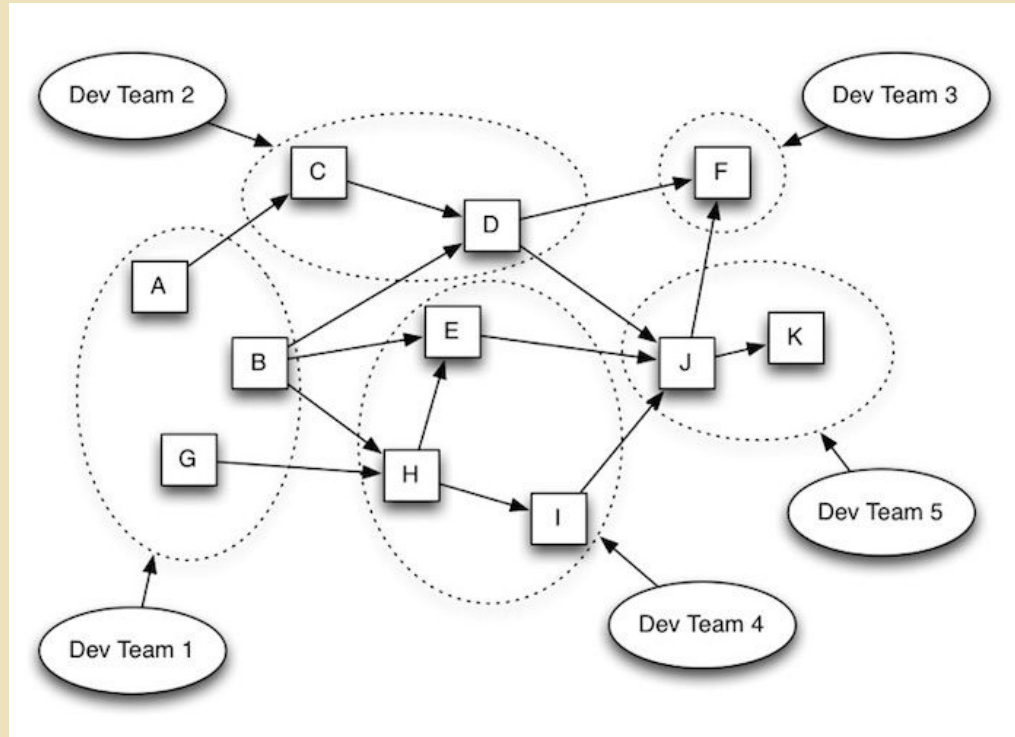
- `script/bootstrap` install required dependencies
- `script/setup` set up an initial state (after clone/reset)
- `script/update` ensure everything is up to date (after a pull)
- `script/server` starts the application (include update)
- `script/test` run tests/test (included in update or cibuild)
- `script/cibuild` used by your CI server
- `script/console` open a console to your application

Now you live in a bubble



HTTP Mocking

Microservices calling REST APIs



What this means for developers?

- each application calling several others (dependencies)
- you need to test HTTP calls:
 - using remote environment
 - make a dummy application
 - write unit tests / contract tests
 - use HTTP mocking tools:
 - easy to create simple requests
 - no language knowledge needed
 - request verification
 - extensible
 - CORS, HTTPS, faults
 - test library, standalone or docker container

Name [▲]	FOSS [↕]	Free [↕]	Supported protocols and APIs [↕]	Has a GUI [↕]	Scriptable/Programmable [↕]	Docker support [↕]	Cloud offering [↕]	Quick start guide [↕]
Hoverfly	Yes; Apache 2 ^[10]	Yes	HTTP(S)	Yes	Yes	Yes	Yes ^[11]	Hoverfly introduction [🔗]
Mountebank	Yes; MIT ^[15]	Yes	HTTP(S), TCP, SMTP ^[16]	No	Yes ^[17]	Yes ^[18]	No	Mountebank Getting Started [🔗]
Parasoft Virtualize	No; Proprietary	Yes (Community Edition) ^[19]	AMQP, FIX, FTP, HTTP(S), ISO 8583, JMS, JDBC, MQ, MQTT, .NET WCF, RabbitMQ, SAP, TCP/IP, etc. ^[20]	Yes	Yes	Yes ^[21]	Yes	
sMockin	Yes; Apache 2 ^[22]	Yes	HTTP, Websockets, Server Side Events, JMS (Queues and Topics), FTP	Yes	No	No	No	[2] [🔗]
SoapUI MockServer	Yes; EURL ^[23]	Yes	HTTP(S) ^[24]	Yes ^[24]	No ^[24]	No ^[24]	No ^[24]	MockServer quick start [🔗]
Traffic Parrot	Partial ^[25] , Proprietary ^[26]	No	HTTP(S), JMS (Queues and Topics), IBM® MQ, File transfers, gRPC ^[26] In beta ^[27] : FIX, FAST, FIXatdl, SWIFT, AMQP, MQTT, RabbitMQ, SonicMQ, CORBA, FTP, SFTP, .NET WCF, RMI, MTP, TIBCO EMS, CICS, SAP RFC, JDBC, Mongo, Databases, OFX, IFX, RIXML, Amazon Simple Queue Service (SQS), Amazon Simple Notification Service (SNS), AWS IoT Message Broker, XMPP, Google Cloud Messaging (GCM), Azure Event Grid, Azure Event Hubs, Azure Service Bus, STOMP, Thrift and Avro	Yes ^[26]	Yes ^[28]	Yes ^[26]	Yes ^[29] ^[26]	Traffic Parrot Quick Start [🔗]
Wiremock	Yes; Apache 2 ^[33]	Yes	HTTP(S)	No	Yes	Yes	Yes ^[34]	Wiremock Getting Started [🔗]



HTTP testing demo

Wiremock

Wiremock

- Java based HTTP mock server:
 - clients: PHP, Ruby, Python, Groovy, .NET
- stub configuration over REST API or JSON descriptors
- extensions and customization writing Java code
- build dependency, Junit rule, standalone Java process or docker container
- cloud based services (import mocks or swagger definitions, etc)



Wiremock: Features

- verify matched, unmatched, near miss and tagged requests
- request matchers:
 - url, headers, query parameters, request body, cookies
 - operators: equality, regex, contains, json, json path, xml, xpath, absence, etc
- response templating (Handlebars):
 - extract parts of requests, date and time, random values, strings, conditions, etc
- faults:
 - fixed and random delays, chunked dribble delay, bad responses
- proxy with record and playback



Wiremock: Features

- stateful behaviour
- HTTPS with certificate verification
- customizations:
 - transforming responses, custom request matchers, post-serve actions, request listeners
- Android
- Spring Cloud Contract integration





Wiremock demo

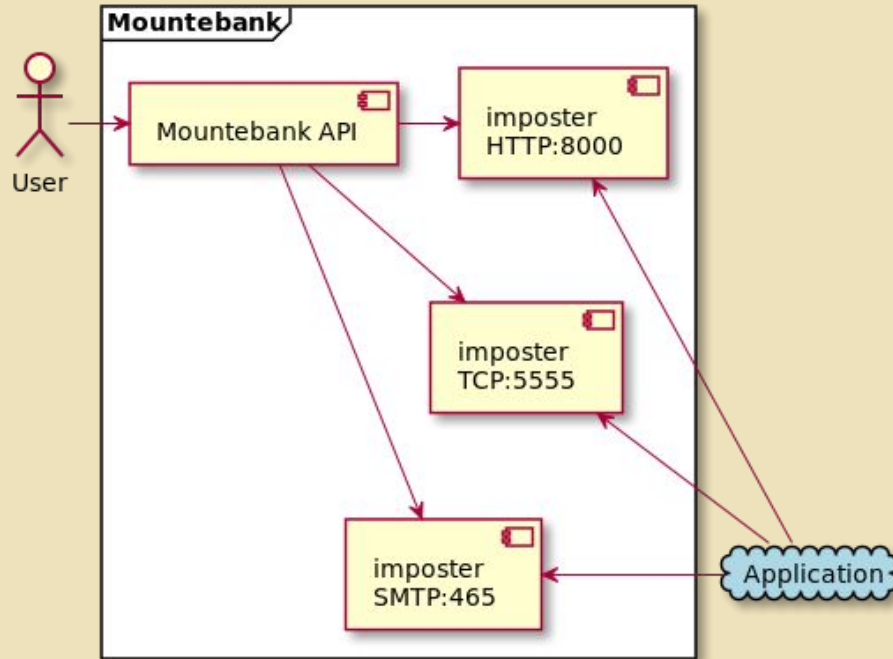
Mountebank

Mountebank

- Node.js based HTTP mock server:
 - clients: Java, Go, JavaScript, TypeScript, PHP, Python, C#, F#, Ruby, Clojure, Delphi, Perl, Shell
- also supports TCP and SMTP mocking
- build tool integrations:
 - Grunt, Gradle, Maven
- stub configuration over REST API or JSON descriptors
- extensions and customization writing JavaScript code (Node.js, EJS)
- build dependency, service or docker container
- built in graphical control-pane
- imposters



Mountebank: Imposters



Mountebank: Features

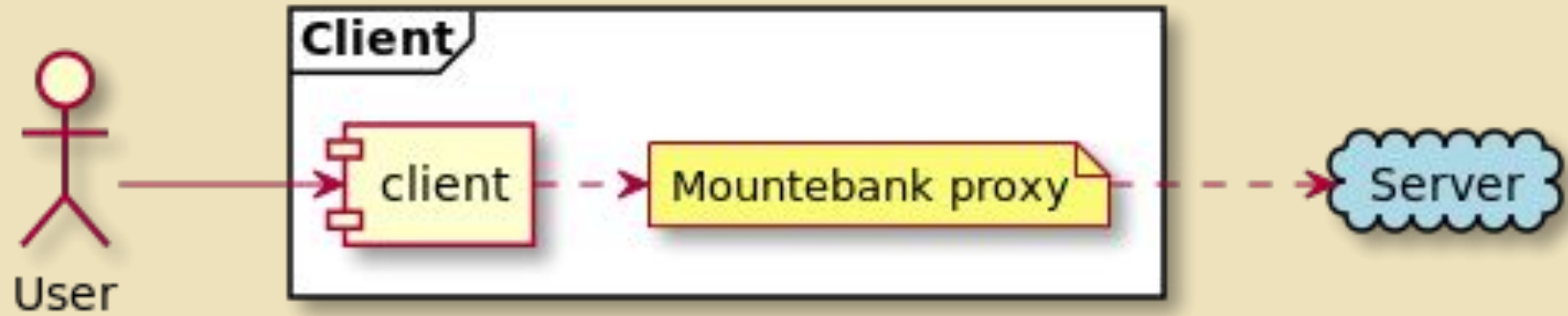
- verify requests
- pretty similar matchers (predicates) to WireMock
- stub behaviors to post-process responses:
 - wait, repeat, copy, lookup, decorate, shellTransform
 - with lookup and shellTransform we can use external datasets
- most non-static behaviors requires `--allowInjection`
- circular response arrays
- proxy with record and playback
- HTTP/HTTPS (with CORS), TCP, SMTP





Mountebank demo

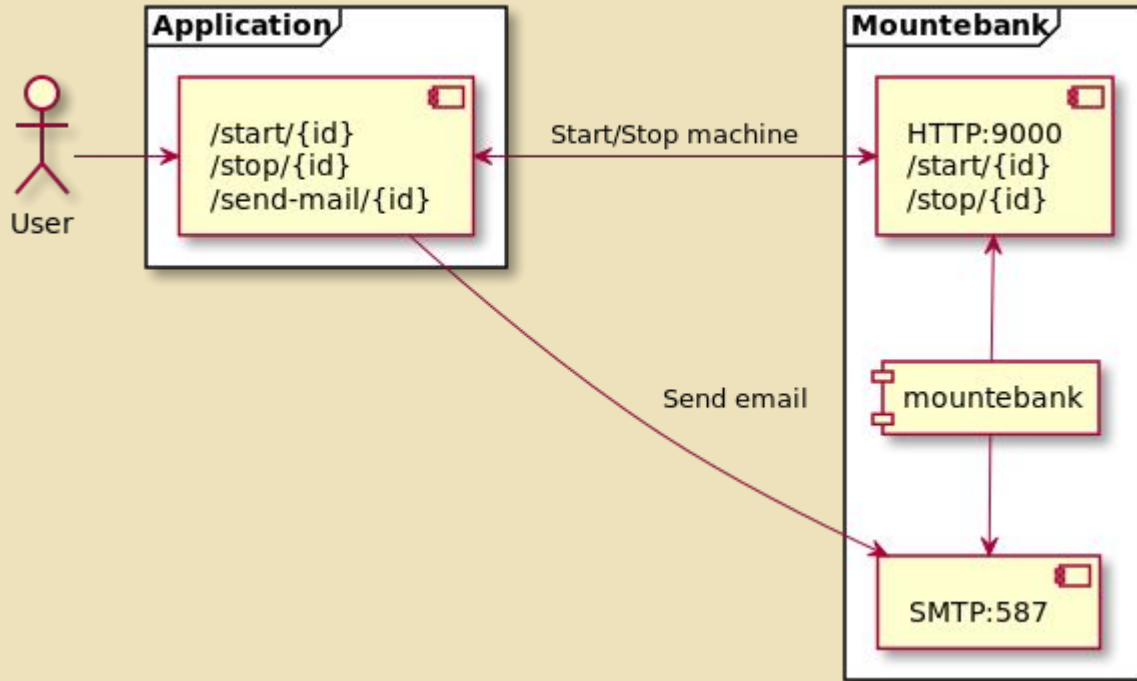
Mountebank demo



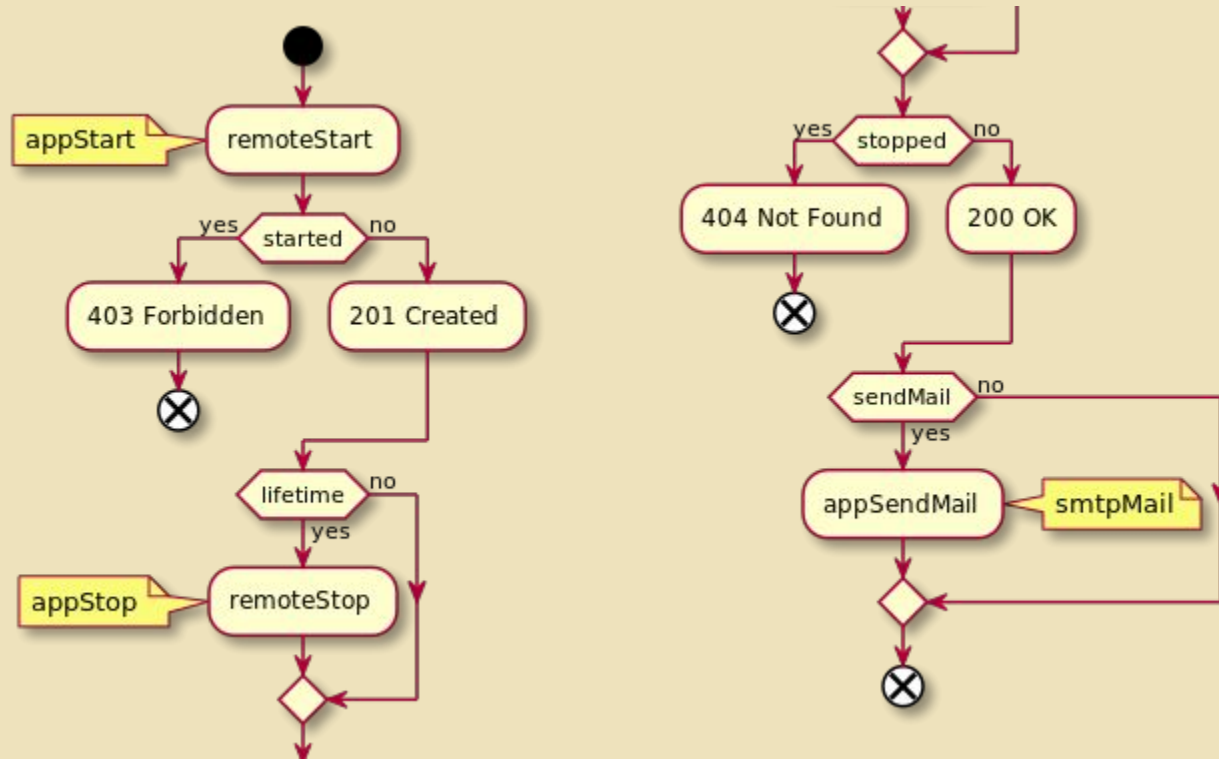


State machine demo

State machine demo



State machine demo





Swagger bank demo

Resources

- MockServer
- Comparison of API simulation tools
- Docker
- Docker Compose
- Scripts to rule them all
- WireMock
- Mountebank
- Demo: <https://github.com/domahidizoltan/presentation-this-is-not-a-pipe>



Thank you!