

From I developed and tested it to I developed and my CI tested it

Automate and structure tedious but important tasks

Lars Stampe Villadsen

SimCorp A/S

Dyalog User Conference - Olhão

October 9-13th 2022



Me

Working with APL since 1992 where I started in SimCorp -
“Whenever possible ‘steal’ code”

System Architect in “Engineering Platform” responsible for
the qualification pipeline for the SimCorp Dimension Solution.

“Current” challenge:

Moving SimCorp Dimension from own developed repositories
(and pipelines) to GitHub based on files for all APL (and other
programming languages) components – while everything still
works.



The world's leading provider bringing integrated solutions to investment managers

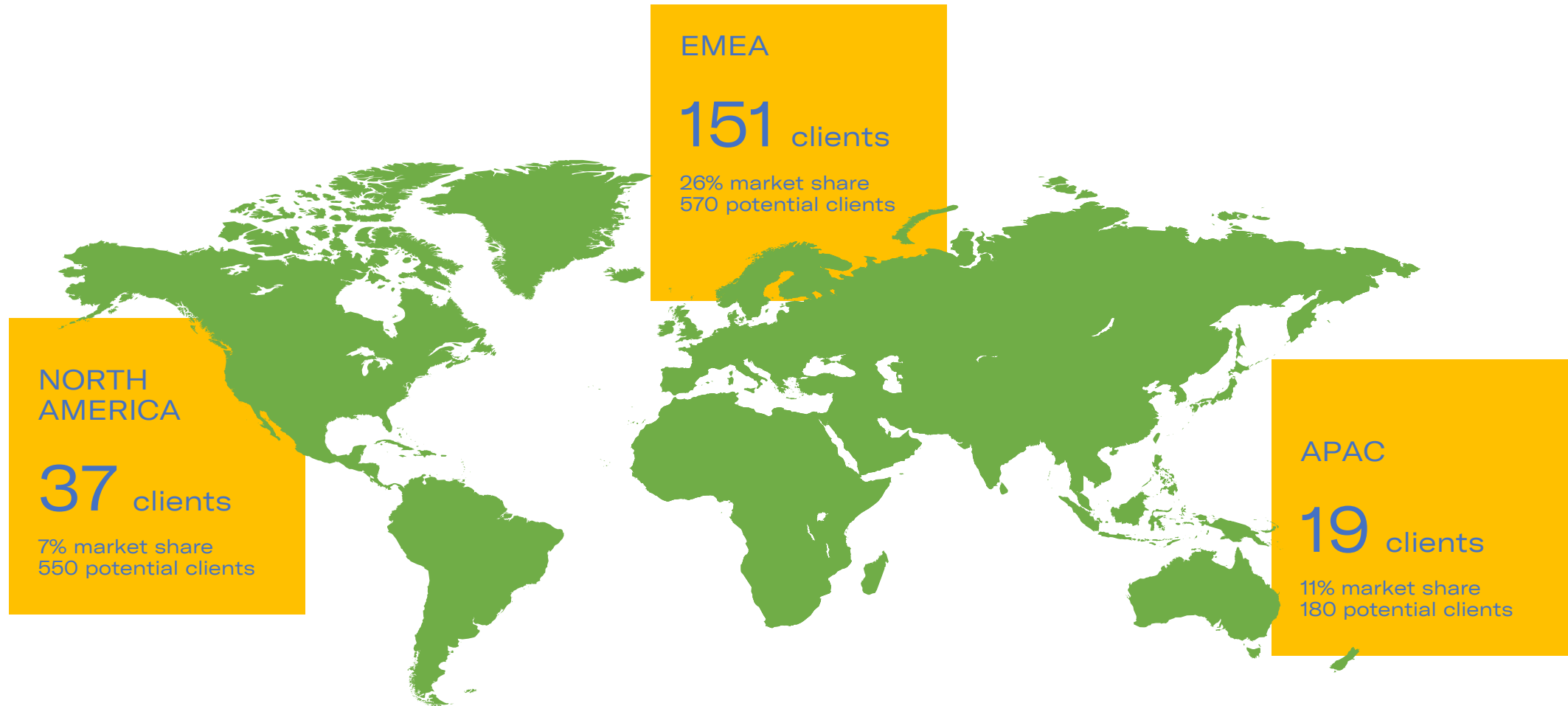
- Servicing the global buy-side market
- Established in 1971
- Headquartered in Copenhagen, Denmark
- Listed on Nasdaq Copenhagen
- Offices across EMEA, North America, and APAC
- 2,000 employees





The world's leading provider

SimCorp Dimension[®] usage around the world



Note: SimCorp Dimension clients (as of end-Q4, 2021)



Motivation

“All” developers test their solutions

If they are seasoned (or young), they build functions that test those solutions for them

If they are "modern" they automatically run their test functions using Continuous Integration (CI).

We will demonstrate how everyone can move from the initial step to the last step using standard (free) tools for APL where the code is maintained in files.



Motivation (where will we end)

GitHub navigation bar: Search or jump to... Pull requests Issues Marketplace Explore

Repository: stampes / CISimple Public

Actions: Pin Unwatch 1 Fork 0

Navigation: Code Issues Pull requests **Actions** Projects Wiki Security Insights Settings

Workflows: **All workflows** (New workflow)

Build and Execute UT Runner

All workflows
Showing runs from all workflows

Filter workflow runs

6 workflow runs

Event	Status
Removed debug	main
Build and Execute UT Runner #29: Commit a3ab848 pushed by stampes	



Motivation (where will we end)

✓ Removed debug Build and Execute UT Runner #29

Summary

Jobs

✓ build

```
build
succeeded 3 hours ago in 28s
Search logs

> ✓ Set up job
> ✓ Run actions/checkout@v3
v ✓ Build the Docker image and run tests

1 ▶ Run docker run -v /home/runner/work/CISimple/CISimple/src/Main:/main -v /home/runner/work/CISimple/CISimple/src/Test:/test $(docker build -q .)
4 _____
5 Dyalog APL/S-64 Version 18.2.45405
6 | _ \ \ / // \ | | / _ \ / _ |
7 Serial number: UNREGISTERED - not for commercial use
8 |_| | \ \ / // \ | | | | | | |
9 | | \ // \ \ | | | | | | _
10 _ | | | | / / \ \ | | | | | |
11 | _ / | / / \ \ _ \ / \ \ |
12
13 https://www.dyalog.com
14
15 LOAD, CONFIGFILE or DYAPP was set, not loading any defaults
16 +-----+
17 | Dyalog is free for non-commercial use but is not free software. |
18 | A basic licence can be used for experiments and proof of |
19 | concept until the point in time that it is of value. |
20 | For further information visit |
21 | https://www.dyalog.com/prices-and-licences.htm |
22 +-----+
23 Mon Oct 10 07:02:35 2022
24 Loaded: #.UT_Runner from "/tools/UT_Runner/UT_Runner.aplf"
25 2022 10 10 7 2 36 136 Executing 4 test(s) in container
26 2022 10 10 7 2 36 136 testInputBundleDiscount_BuyOne_GetAnother Executing
27 2022 10 10 7 2 36 136 testInputBundleDiscount_BuyOne_GetPart_Fail Executing
28 2022 10 10 7 2 36 136 testInputBundleDiscount_Buy_Get Executing
29 2022 10 10 7 2 36 136 testInputBundleDiscount_Member Executing
30 2022 10 10 7 2 36 136 4 test(s) OK
```



Let us define an Application

Assumptions (you can ease these for own purpose):

- All application source files are placed in a `.\src\` folder (and sub-folders) – All source files are `*.aplf` files
- No initialization of application code is needed
- All test source files are placed in a `.\src\` folder (and sub-folders) - All test files are `*.aplt` files
- All test functions will be niladic and throw a signal if the fail



Let us make a simple application

Be able to parse and validate a set of discount statements:

Buy one, get one free

Buy two get one free

Buy shirt, get pants at 50%

Get 10% off if you're a member

Products:

Pants

Shirt

Socks

Shoes

Hat



Let us make a simple application

```
ok=InputBundleDiscount bundlediscount;_m;inputtest;numbers;_skip;foundnumber;_d;buynumber;getnumber;parts;foundpart;buypart;getpart;match
A ... Assume that bundle discount either is in the form
A 'Buy <number-as-text>[.]* get <number-as-text> free'
A 'Buy <part>[.]* get <part> at <pct>%'
A
A Return ok = true for valid string

_m={a#(a|p|w)+}
_d={{(a|p|w)}+}
_skip={{(v|w|e|'|')/w}

ok=0

inputtest=DC bundlediscount
numbers='one' 'two' 'three' 'four'
:If 'buy'_m inputtest
    inputtest=_skip'buy'_d inputtest
    foundnumber=numbers _m"cinputtest
:AndIf v/foundnumber
A
-----
A ... Validating buy / get pattern:
A-----
    buynumber=>foundnumber/numbers
    inputtest=_skip buynumber _d inputtest
:If 'get'_m inputtest
    inputtest=_skip'get'_d inputtest
    foundnumber=numbers _m"cinputtest
:AndIf v/foundnumber
    getnumber=>foundnumber/numbers
    inputtest=_skip getnumber _d inputtest
:AndIf (0=>pinputtest)vinputtest='free'
    ok=1
:EndIf
:Else
A
-----
A ... Validating buy one get another pattern:
A-----
    parts=DC GetParts
    foundpart=parts _m"cinputtest
:If v/foundpart
    buypart=>foundpart/parts
    inputtest=_skip buypart _d inputtest
:AndIf 'get'_m inputtest
    inputtest=_skip'get'_d inputtest
    foundpart=parts _m"cinputtest
:AndIf v/foundpart
    getpart=>foundpart/parts
    inputtest=_skip getpart _d inputtest
:If 0=>pinputtest
    ok=1
:Else
    ok=0=>p('at [0-9]+%'S'\0')inputtest
:EndIf
:EndIf
:EndIf
```



Save tests for simple application

```
testInputBundleDiscount_Buy_Get  
A Validate input format for Discount specification
```

```
specification+ 'Buy one, get one free'
```

```
ok+InputBundleDiscount specification
```

```
±(-ok)/'[] SIGNAL 99'
```

```
specification+ 'Buy two get one free'
```

```
ok+InputBundleDiscount specification
```

```
±(-ok)/'[] SIGNAL 99'
```

```
testInputBundleDiscount_BuyOne_GetAnother  
A Validate input format for Discount specification
```

```
specification+ 'Buy shirt, get pants at 50%'
```

```
ok+InputBundleDiscount specification
```

```
±(-ok)/'[] SIGNAL 99'
```

```
testInputBundleDiscount_BuyOne_GetPart_Fail  
A Validate input format for Discount specification
```

```
specification+ 'Buy one, get pants'
```

```
ok+InputBundleDiscount specification
```

```
±(ok)/'[] SIGNAL 99'
```



Run tests automatic in session

```
UT_Runner;_fixfiles;_loadfiles;mainfiles;mainfolder;mainfunctions;recurse;testfiles;testfolder;testfunctio
A0: Simple Unit Test runner written in pure APL no extras allowed.
A
=====
A ... Function expects environment variables pointing to
A ...   MainFolder: Location of main (application) files - all files with extension
A ...   '*.aplf' will be loaded from that folder (and all subfolders)
A ...   TestFolder: Load all test functions (with the extension) '*.aplt'.
A ...   Folder and subfolders are scanned
A
=====

:Section Utilities

A
-----
A ... Load main (application) code
A
mainfiles←mainfolder _loadfiles'aplf'
mainfunctions←_fixfiles mainfiles

A
-----
A ... Load test code:
A
testfiles←testfolder _loadfiles'aplt'
testfunctions←_fixfiles testfiles

failedtests←0p''0

_out'Executing ',(≡p)testfunctions),' test(s)'
:For testfunction :In testfunctions
  testfunction _out'Executing'
  :Trap 0
    testfunction
  :Else
    dmx←DMX
    testfunction _out'Failed'
    testfunction _out dmx
    failedtests,←testfunction dmx
  :EndTrap
:EndFor
:If 0≠pfailedtests
  _out(≡pfailedtests),' test(s) FAILED'
  _off 11
:Else
  _out(≡p)testfunctions),' test(s) OK'
  _off 0
:EndIf
_out'Run Ended'
```



Run tests automatic in session

```
UT_Runner
2022 10 7 15 42 34 606      Executing 4 test(s)
2022 10 7 15 42 34 607 testInputBundleDiscount_BuyOne_GetAnother Executing
2022 10 7 15 42 34 610 testInputBundleDiscount_BuyOne_GetPart_Fail Executing
2022 10 7 15 42 34 611 testInputBundleDiscount_Buy_Get Executing
2022 10 7 15 42 34 614 testInputBundleDiscount_Member Executing
2022 10 7 15 42 34 614      4 test(s) OK
2022 10 7 15 42 34 617      Run Ended
```

Debugger

Ready...



Run tests automatic w/o session

DOCKERFILE

```
FROM dyalog/dyalog
```

```
USER root
```

```
RUN mkdir /main /test /tools && \  
    chmod 777 /main /test /tools
```

```
RUN apt-get update && apt-get install -y git
```

```
RUN git clone https://github.com/stampes/CISimple /tools
```

```
VOLUME [ "/main", "/test" ]
```

```
ENV MainFolder /main
```

```
ENV TestFolder /test
```

```
ENV LOAD /tools/UT_Runner/UT_Runner.aplf
```

```
USER dyalog
```



Run tests automatic w/o session

```
PS C:\repos\CISimple> docker build -t ut_runner .
[+] Building 0.2s (8/8) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 32B                                              0.0s
=> [internal] load .dockerignore                                                0.1s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/dyalog/dyalog:latest                 0.0s
=> [1/4] FROM docker.io/dyalog/dyalog                                          0.0s
=> CACHED [2/4] RUN mkdir /main /test /tools &&      chmod 777 /main /test /tools  0.0s
=> CACHED [3/4] RUN apt-get update && apt-get install -y git                    0.0s
=> CACHED [4/4] RUN git clone https://github.com/stampes/CISimple /tools        0.0s
=> exporting to image                                                           0.1s
=> => exporting layers                                                           0.0s
=> => writing image sha256:f5c5ffff1fa9cfc9e1e3b89f0c9a0a7ddfc70de2a66dd137623b27b00fb38b25 0.0s
=> => naming to docker.io/library/ut_runner                                    0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS C:\repos\CISimple> |
```




Just “Run tests”

name: Docker Image CI

on:

push:

branches: ["main"]

pull_request:

branches: ["main"]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: Build the Docker image and run tests

run: docker run -v \${github.workspace}/src/Main:/main -v

\${github.workspace}/src/Test:/test \$(docker build -q .)



Just “Run tests”

stampes / CISimple Public

<> Code Issues Pull requests **Actions** Projects Wiki Security Insights Settings

Workflows

[New workflow](#)


All workflows

 **Docker Image CI**

Docker Image CI

ut-docker-image.yml

18 workflow runs

-  **Reorder so code before tests**
Docker Image CI #18: Commit 9bdb2c0 pushed by stampes
-  **Load source code own way**
Docker Image CI #17: Commit 7c14840 pushed by stampes



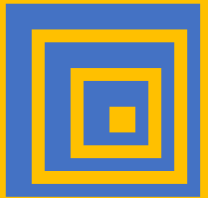
Just “Run tests”

<https://github.com/stampes/CISimple>



And it just “Run tests”

Thank you



SimCorp

Legal notice

The contents of this publication are for general information and illustrative purposes only and are used at the reader's own risk. SimCorp uses all reasonable endeavors to ensure the accuracy of the information. However, SimCorp does not guarantee or warrant the accuracy, completeness, factual correctness, or reliability of any information in this publication and does not accept liability for errors, omissions, inaccuracies, or typographical errors. The views and opinions expressed in this publication are not necessarily those of SimCorp. © 2022 SimCorp A/S. All rights reserved. Without limiting rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form, by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose without the express written permission of SimCorp A/S. SimCorp, the SimCorp logo, SimCorp®, and SimCorp Services are either registered trademarks or trademarks of SimCorp A/S in Denmark and/or other countries. Refer to www.simcorp.com/trademarks for a full list of SimCorp A/S trademarks. Other trademarks referred to in this document are the property of their respective owners.