# Lift-Off from APL2 Mainframe to Dyalog in the Cloud

Migration of APL2 system from mainframe to Dyalog APL on Linux

Gilgamesh Athoraya

TIAMATICA

## The system

CAPP – Computer Aided Process Planning

Sandvik AB

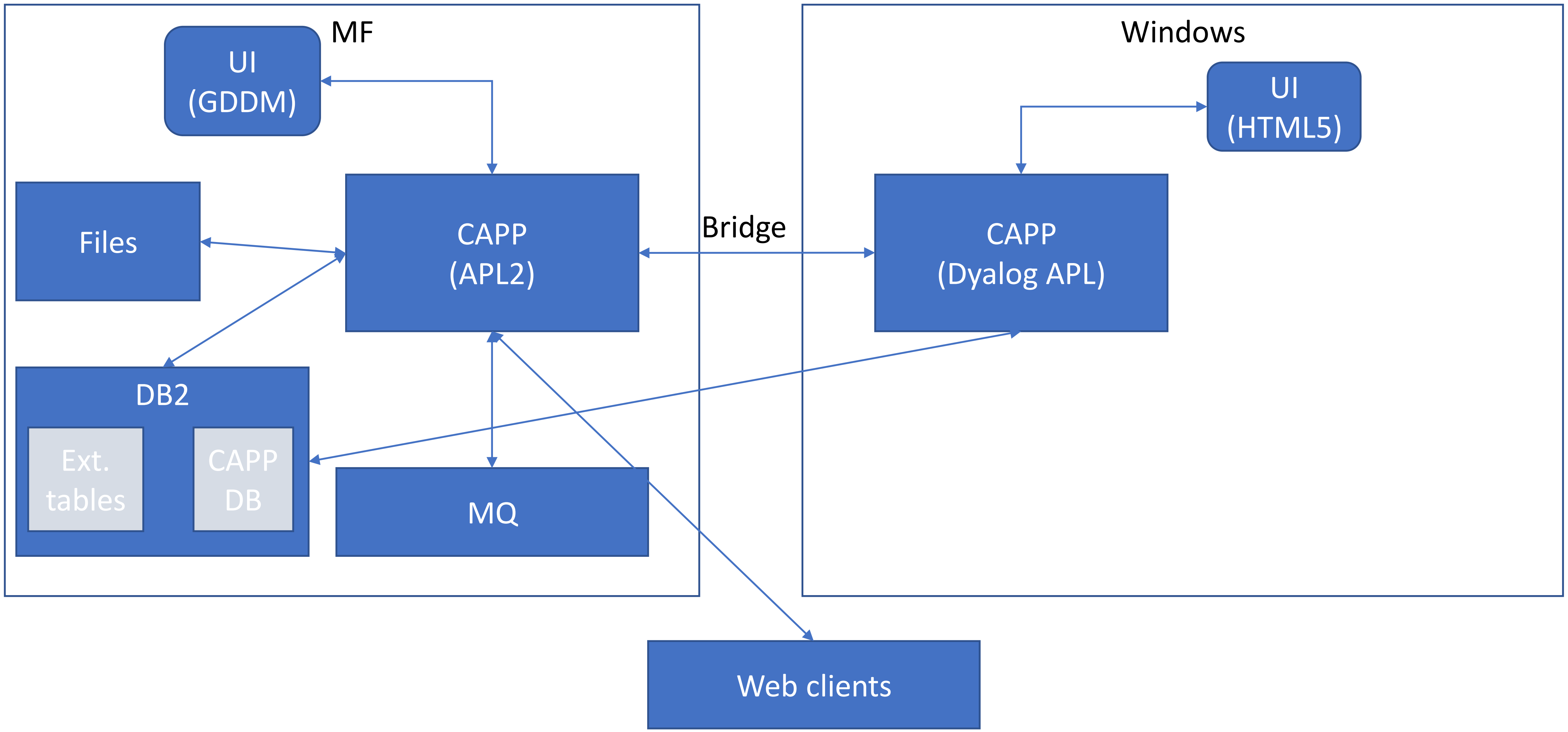
- APL2 v3.0.0 (LogOn)
- z/OS 2.3

## Project objective

Migrate the CAPP system off of mainframe and APL2 onto Dyalog APL on Windows and Linux.

- Language differences
- User interface
- Database and files
- Network communication (DB2, webserver, MQ, etc.)

# System overview – Current



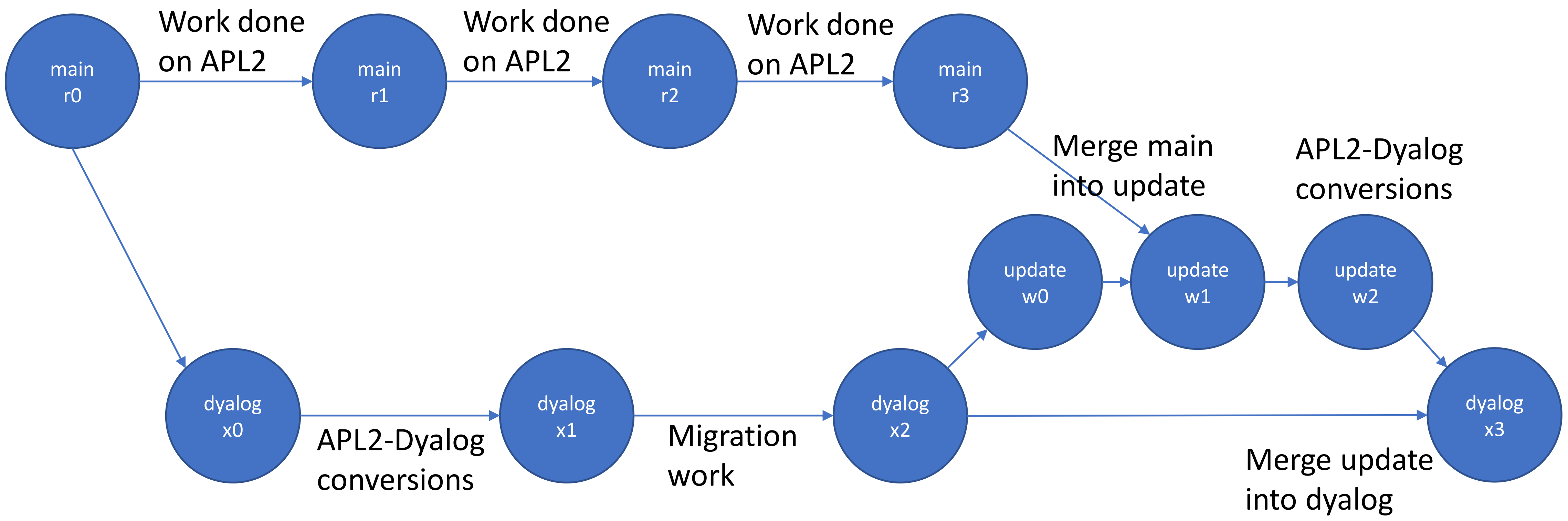
## Source control

- Import APL2 source into Git (main branch)
  - Use backups as source for individual commits to add some history
- Create “dyalog” branch for migration work
- Apply migration conversion to dyalog branch
- Carry on migration work in dyalog branch

## Source control

- Update “main” branch (import from APL2)
- Create new “update” branch from “dyalog” for merge
- Merge “main” into “update”
  - Git recognises conflicts, even after re-structuring in “dyalog” branch
  - Apply migration conversions to modified files (identified using git diff)
  - Review and test
  - Merge “update” into “dyalog” branch

# Git graph on development



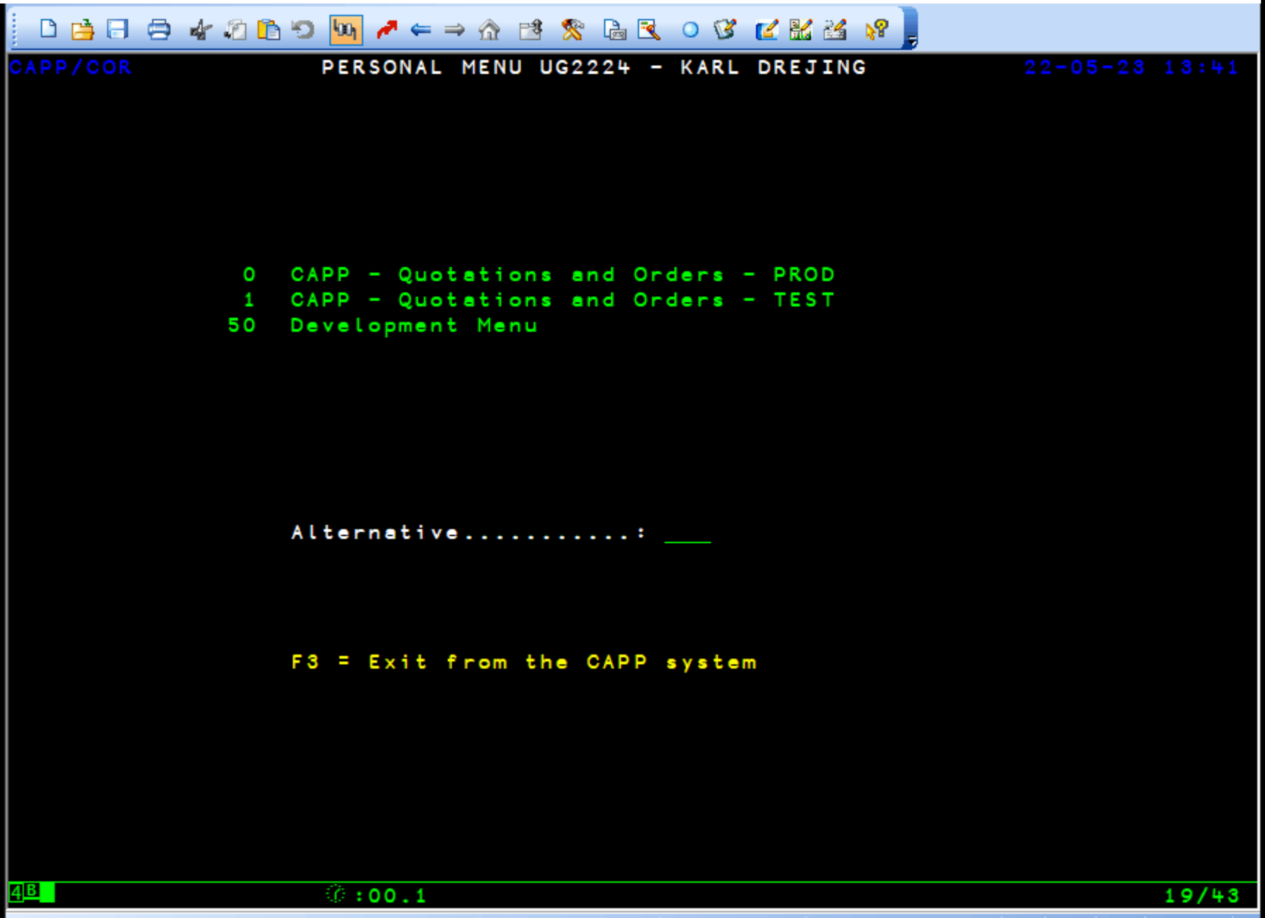
# Language differences

- Legal APL names (high minus)
- Ambivalent functions
- Replicate each
- Control structures (conditional branch/execute)
- System variables (⎕TZ, ⎕ET, ⎕FC, ⎕PR)
- System functions (⎕EA, ⎕EM, ⎕ES, ⎕TS)
- Format by example (‘0000’⍕)
- Each operator (prototype on empty)
- Bracket indexing
  - A B C[index]
  - A B (C[index])
- Assign to single name
  - A B C←1 2 3
  - A B (C←1 2 3)
- APL2 namespace/package

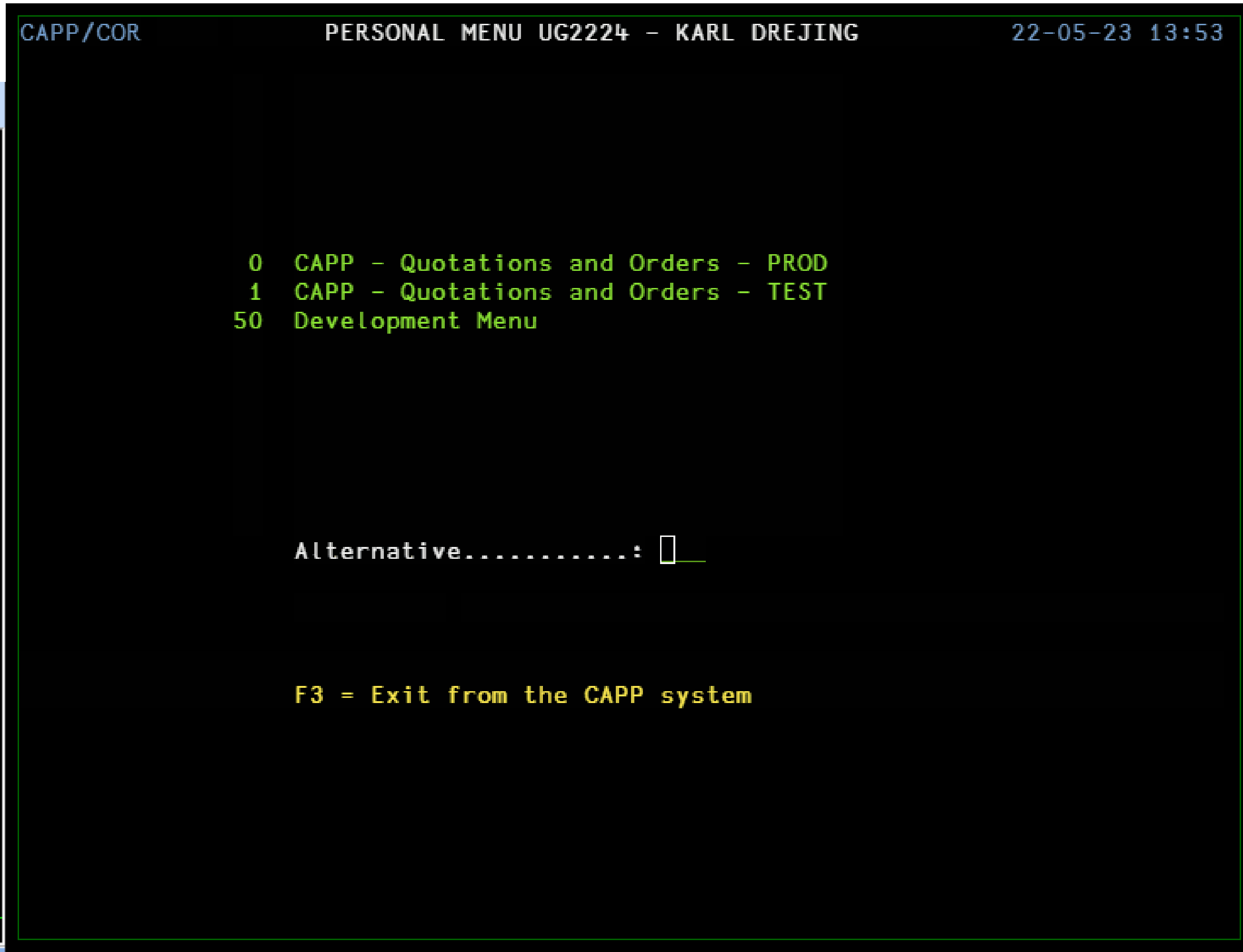
## User interface (GDDM)

- Keep UI code unchanged (GDDM control messages)
- GDDM emulator in javascript (frontend)
  - xterm for terminal emulation
  - Svg.js for graphics
  - Support both browser (thin client) and HTMLRenderer (fat client)
- WebSocket server (backend)
  - Bridge between shared variable access and web client
  - Manage async communication with client

# GDDM – Menus and panels

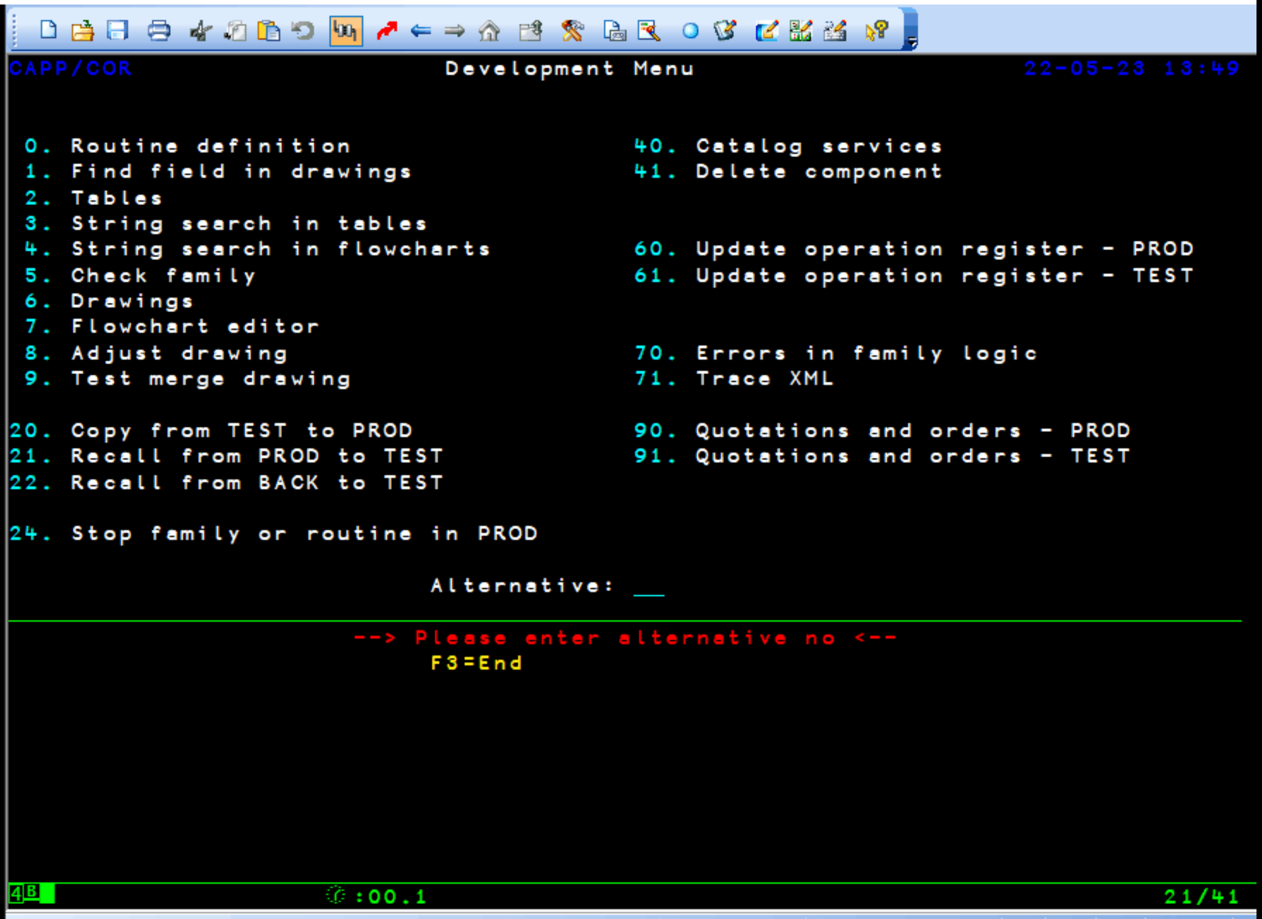


```
CAPP/COR                PERSONAL MENU UG2224 - KARL DREJING            22-05-23 13:41




                 0   CAPP - Quotations and Orders - PROD
                 1   CAPP - Quotations and Orders - TEST
                50   Development Menu





                     Alternative..........: ___



                     F3 = Exit from the CAPP system






                          :00.1                                          19/43
```

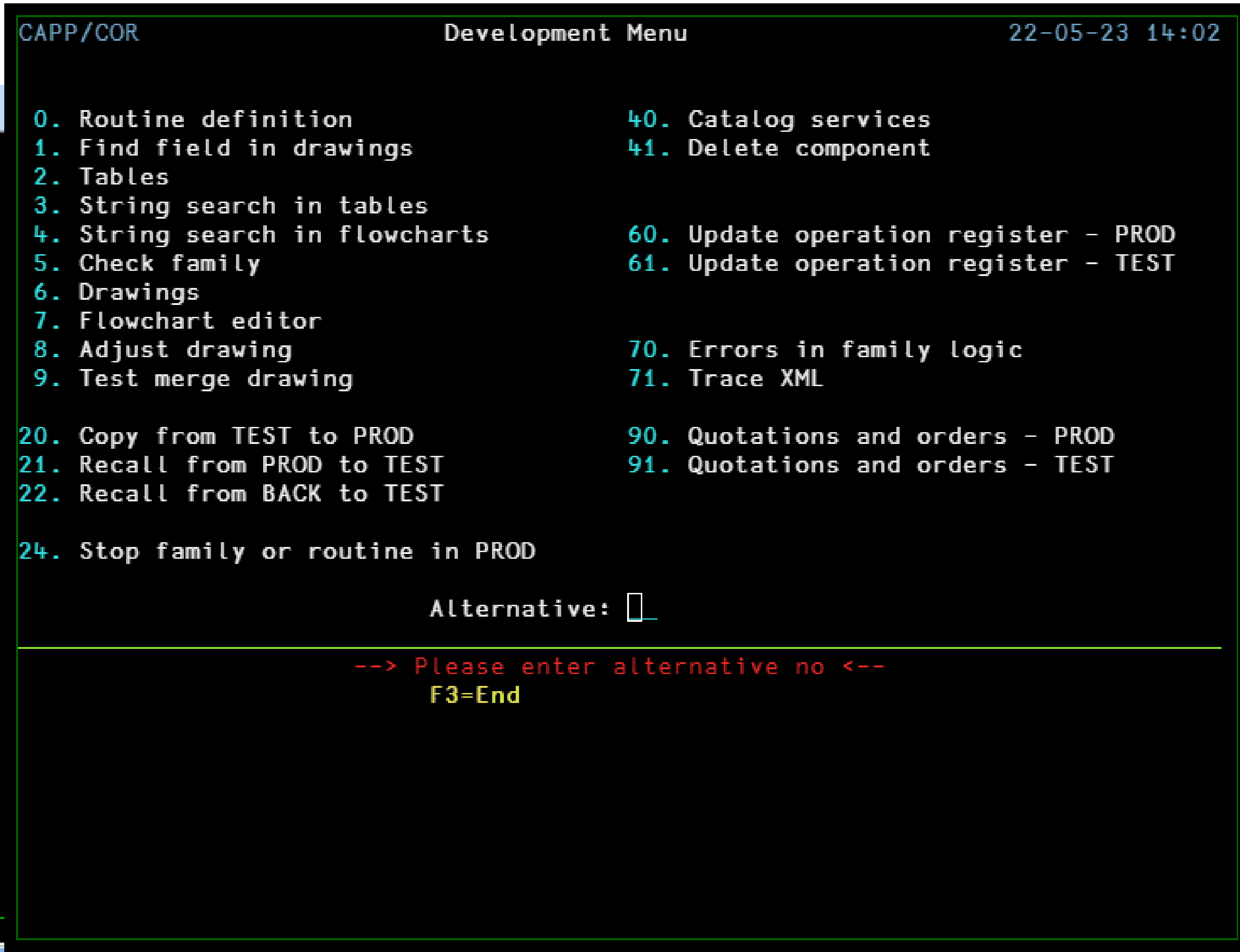


```
CAPP/COR                PERSONAL MENU UG2224 - KARL DREJING            22-05-23 13:53




                 0   CAPP - Quotations and Orders - PROD
                 1   CAPP - Quotations and Orders - TEST
                50   Development Menu





                     Alternative..........: ___



                     F3 = Exit from the CAPP system
```

# GDDM – Flowchart editor



```
CAPP/COR                         Development Menu                    22-05-23 13:49


 0. Routine definition                          40. Catalog services
 1. Find field in drawings                      41. Delete component
 2. Tables
 3. String search in tables
 4. String search in flowcharts                 60. Update operation register - PROD
 5. Check family                                61. Update operation register - TEST
 6. Drawings
 7. Flowchart editor
 8. Adjust drawing                              70. Errors in family logic
 9. Test merge drawing                          71. Trace XML

20. Copy from TEST to PROD                      90. Quotations and orders - PROD
21. Recall from PROD to TEST                    91. Quotations and orders - TEST
22. Recall from BACK to TEST

24. Stop family or routine in PROD

                           Alternative: __

                      --> Please enter alternative no <--
                           F3=End

4B                    :00.1                                              21/41
```



```
CAPP/COR                         Development Menu                    22-05-23 14:02


 0. Routine definition                          40. Catalog services
 1. Find field in drawings                      41. Delete component
 2. Tables
 3. String search in tables
 4. String search in flowcharts                 60. Update operation register - PROD
 5. Check family                                61. Update operation register - TEST
 6. Drawings
 7. Flowchart editor
 8. Adjust drawing                              70. Errors in family logic
 9. Test merge drawing                          71. Trace XML

20. Copy from TEST to PROD                      90. Quotations and orders - PROD
21. Recall from PROD to TEST                    91. Quotations and orders - TEST
22. Recall from BACK to TEST

24. Stop family or routine in PROD

                           Alternative: _

                      --> Please enter alternative no <--
                           F3=End
```

## Database

- DB2 database shared with other systems
- CAPP requires access to tables owned by other systems
- APL2 tables serialised with ATR (array to record, IBM serializer) and stored in CLOB columns
- DB2 columns use EBCDIC 278 (Swedish/Finnish) but APL2 ⎕AV implied

## Database

- Keep DB2 on mainframe during development of Dyalog version (to reduce performance hit for current users).
- Replace ATR format with SCAR (to allow both for Dyalog and APL2)
- Access DB2 from Dyalog via ODBC

## Network comms

Encrypt all TCP/IP connections to allow secure communication between cloud and mainframe.

- No native way to create secure TCP connections in APL2
- AT-TLS
  Application Transparent - TLS
  Policy controlled upgrade of TCP connections to use TLS. No change required to APL code.
- DB2 Connect server – add support for secure clients
- MQ manager – encrypted channels

## Cloud solution

How to build, test and deploy APL code in the cloud?

- Azure Repos
- Azure Pipelines
- Azure Container Registry
- Azure Kubernetes

## Docker

Different requirements for build and deployment

Build/test:

- Tatin

Deployment:

- .NET runtime
- MQ client
- ODBC + DB2 driver

# Docker architecture – Dyalog Images

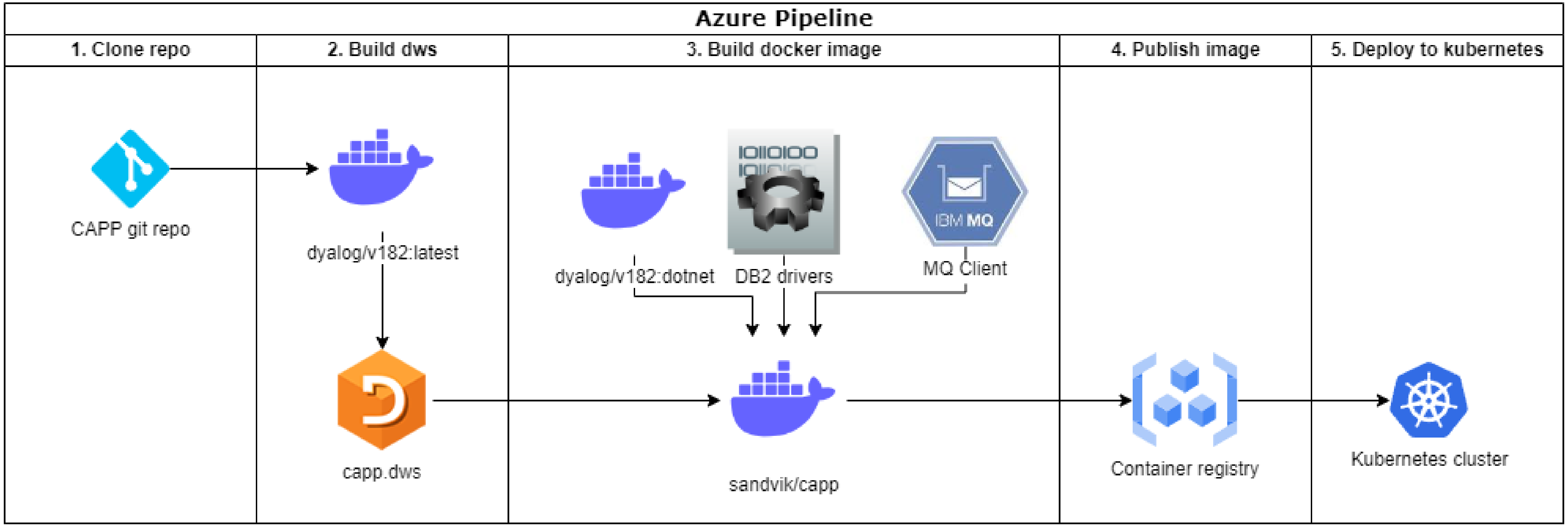
| Base Image | Instructions | Resulting image | Size |
|---|---|---|---|
| debian :buster-slim | Add: Dyalog ODBC Tatin | dyalog/v182 :latest | 240 MB |
| dyalog/v182 :latest | Add .NET Runtime | dyalog/v182 :dotnet | 430 MB |

## Docker architecture – Build and Deploy .dws

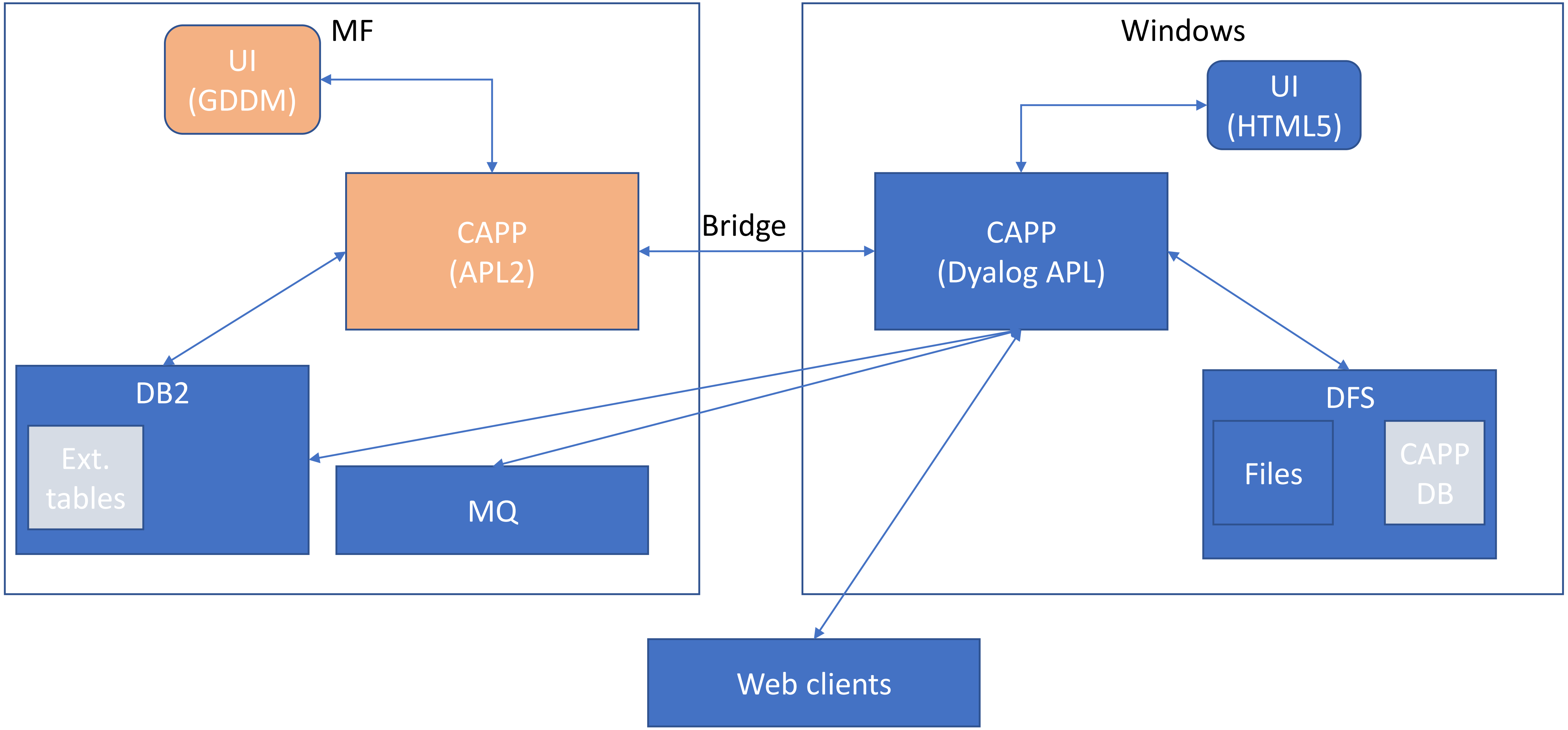
In pipeline:

1. Use base image to build dws
2. Build Docker Image for service
   1. Base on dyalog with or w/o dotnet
   2. Add other dependencies:
      - DB2 drivers
      - MQ Client
   3. Add dws
3. Push image to Container Registry
4. Deploy to Kubernetes

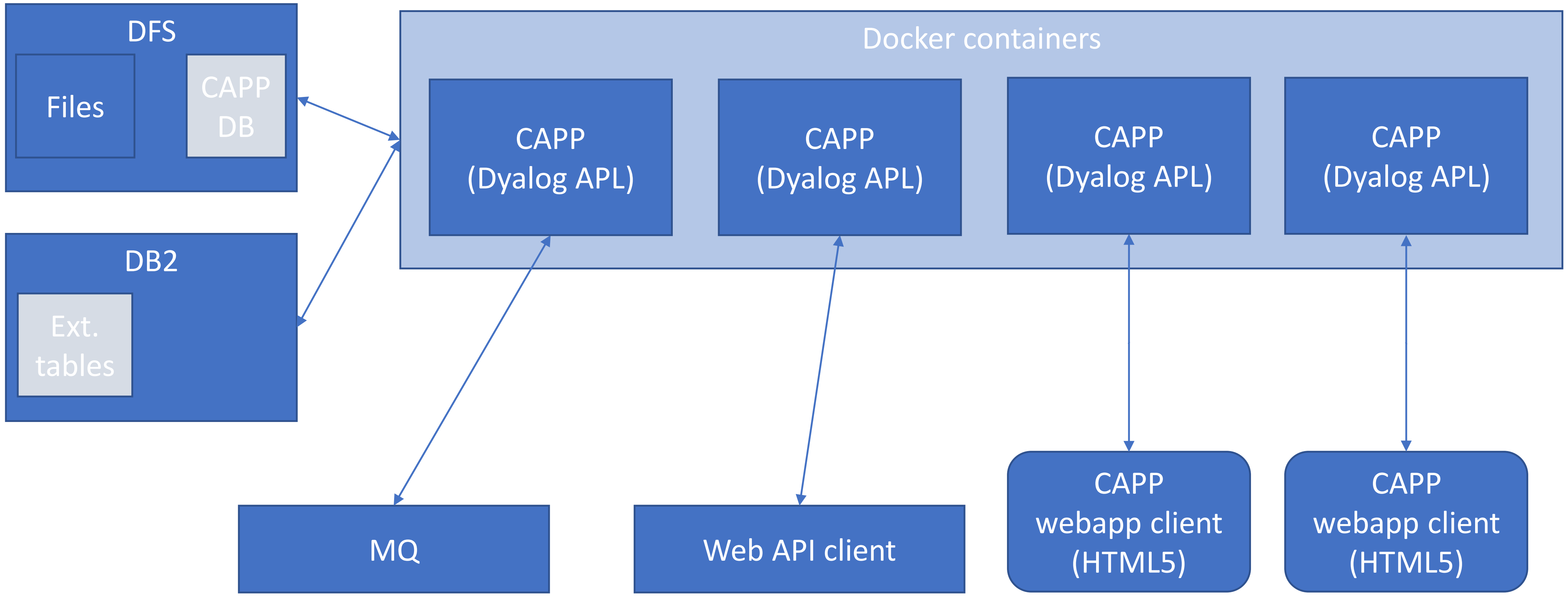
# Azure pipeline



# System overview – Next step



# System overview – End goal



Thanks for listening

TIAMATICA