

Docker-basierte Selenium/Playwright UI Tests in Azure Pipelines

Marc Müller
Principal Consultant



marc.mueller@4tecture.ch
[@muellermarc](https://twitter.com/muellermarc)
www.4tecture.ch

4tecture®
empower your software solutions

A black and white portrait of Marc Müller, a middle-aged man with short, dark hair, wearing glasses and a dark polo shirt under a light-colored jacket. He is smiling and looking towards the camera.

About me:

Marc Müller
Principal Consultant
@muellermarc



4tecture[©]
empower your software solutions

Our Products:

Multi-Tenant OpenID
Connect Identity Provider



ProAuth

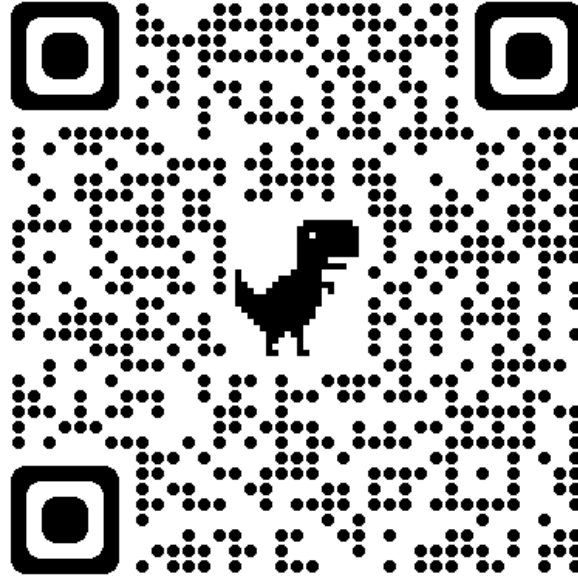
www.proauth.net

Enterprise Application
Framework for .NET



www.reafx.net

Slide Download



<https://www.4tecture.ch/events/dwx24seleniumdockerpipelines>

Agenda

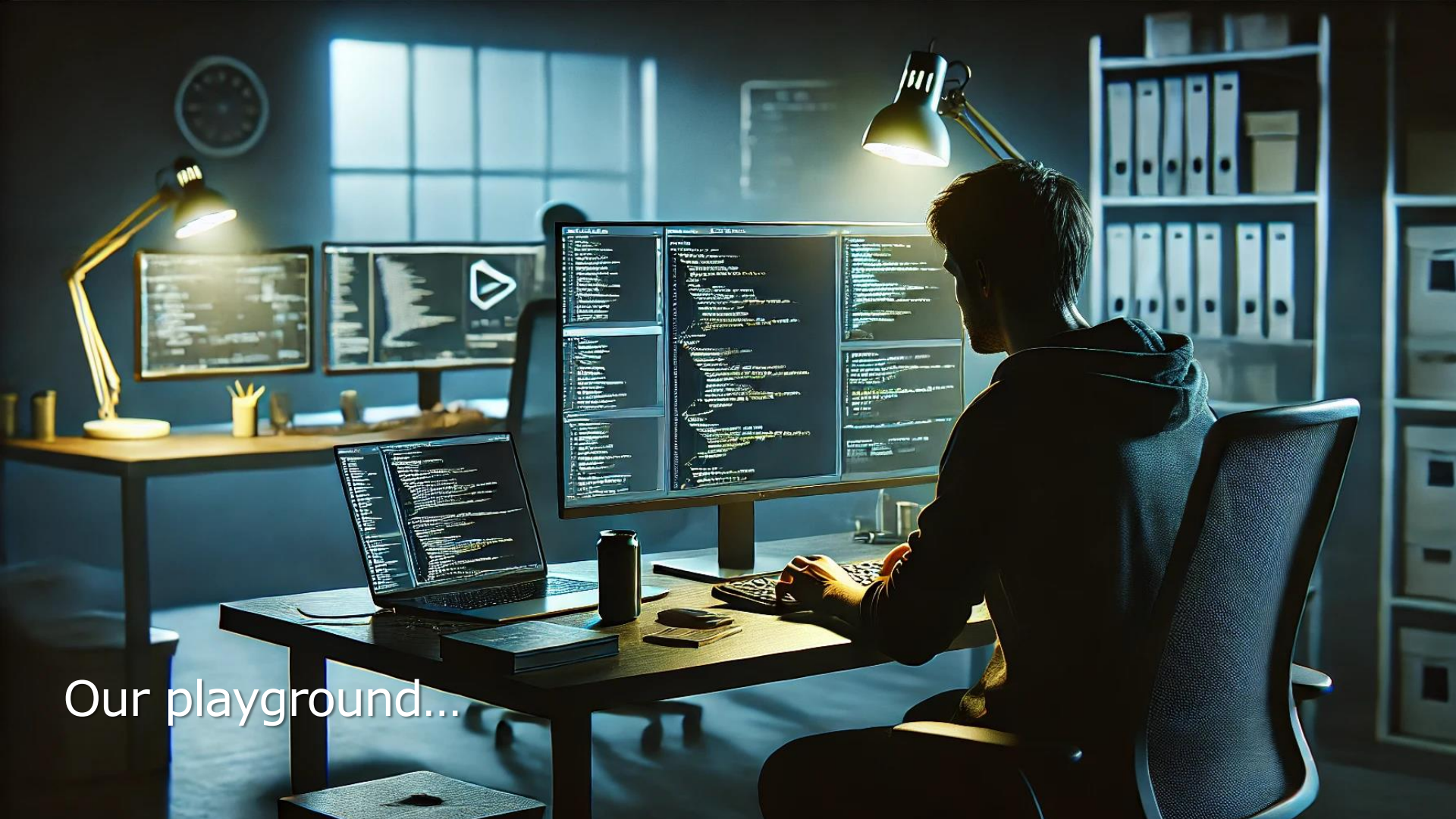
- Intro
- Selenium / Page Objects
- UI Tests in Containers / Setup
- Running tests in container jobs
- Conclusion





Intro

4tecture[®]
empower your software solutions

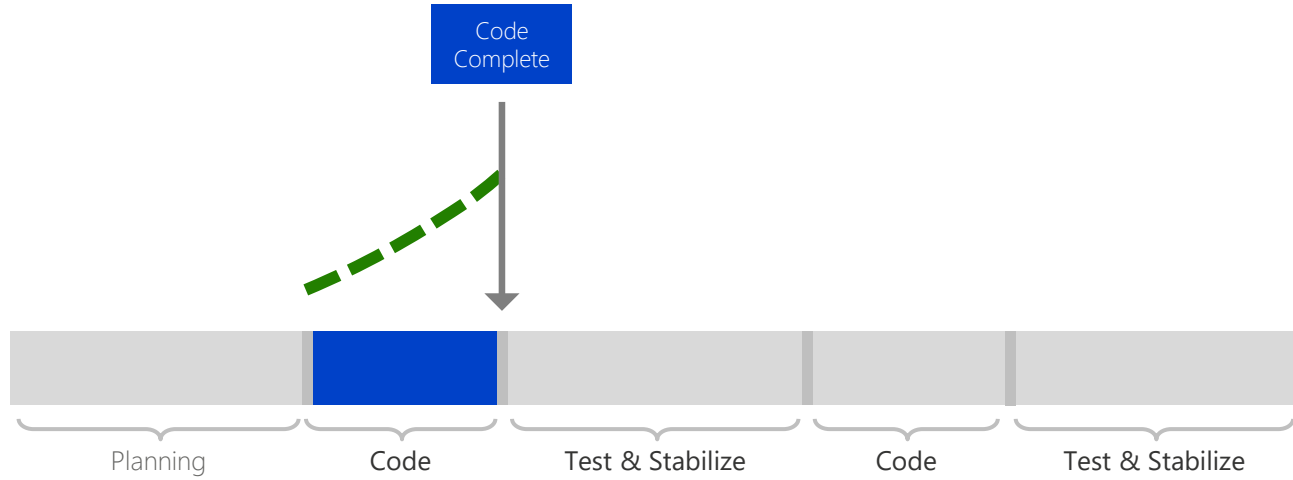


Our playground...

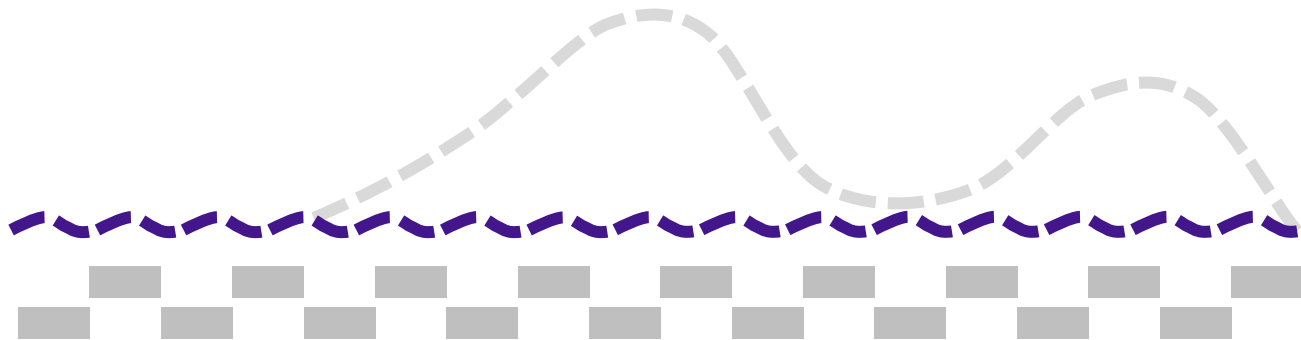


Fail fast!

Before



After



Shift left

Move the testing process to the left

- Integrate testing into the sprint / pull request
- Fast detection and fix
- Testers are part of the team

Continuous Testing

- Effective and continuous integration
- No Bulks of tests / bug-fixing

Without shift left context-switching is expensive / lowers throughput drastically

Shift Left Benefits

- Reduced costs involved in dev/test
- Early bug detection – better quality
- Effective resolution of bugs
- Massive time and effort saved





If it
hurts, do
it more
often!

Test Automation

- Reduce test time
- Have regression tests
- Focus on test design and management, rather than manual repetitive tasks

Conclusion

- No “shift left” only strategy
Combination of shift left and system testing on target system recommended
- Reducing or avoiding long circle times is crucial
- “There is no place like production”

Options for running UI tests

Headless

- No real UI
- Easy and fast

Containers

- Sweet spot between headless and VM
- Fast, reproducible
- Easy pipeline integration

VMs

- Cumbersome to maintain
- Slow, complex pipelines

Real Device

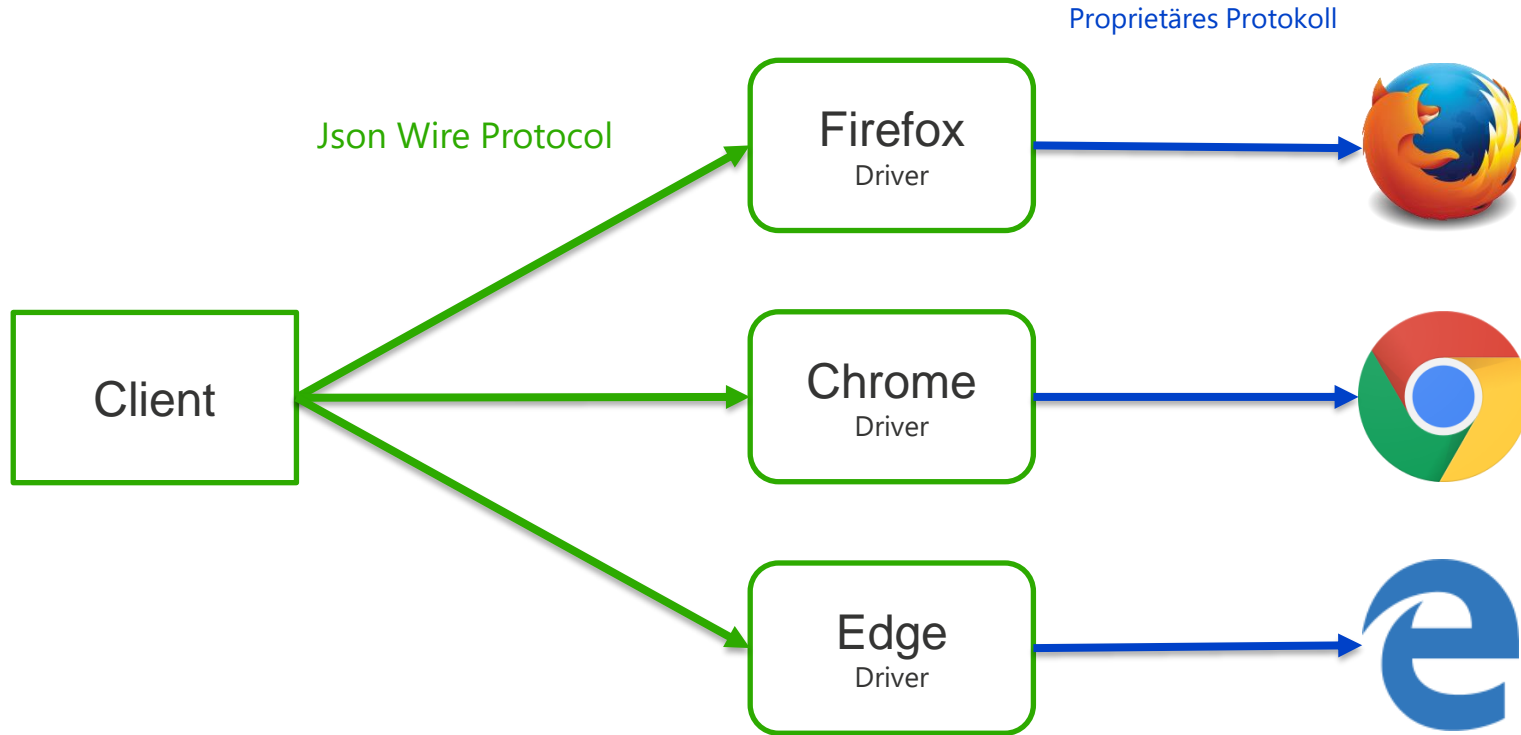
- Best for testing
- Expensive
- “not” scalable



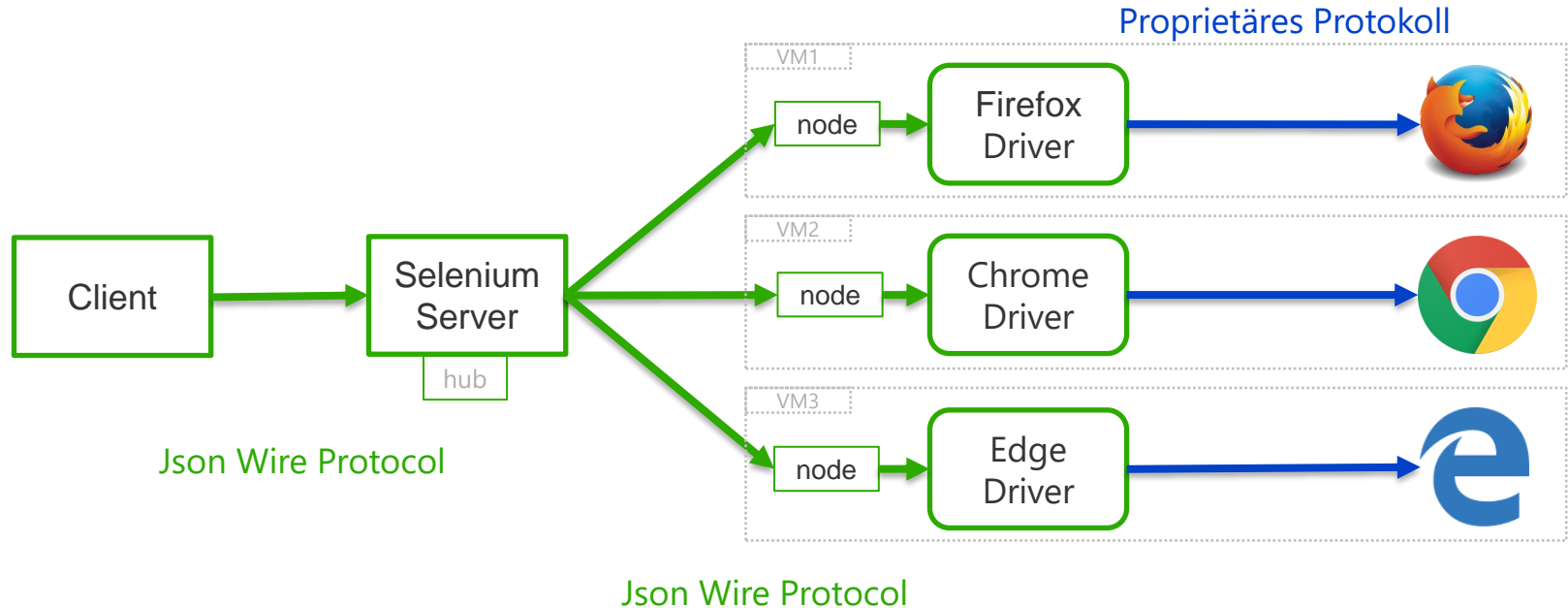
Selenium

4tecture[®]
empower your software solutions

Selenium How-To



Selenium Server How-To



Example: JSON Wire Protocol

POST /session/{session id}/element/{element id}/click

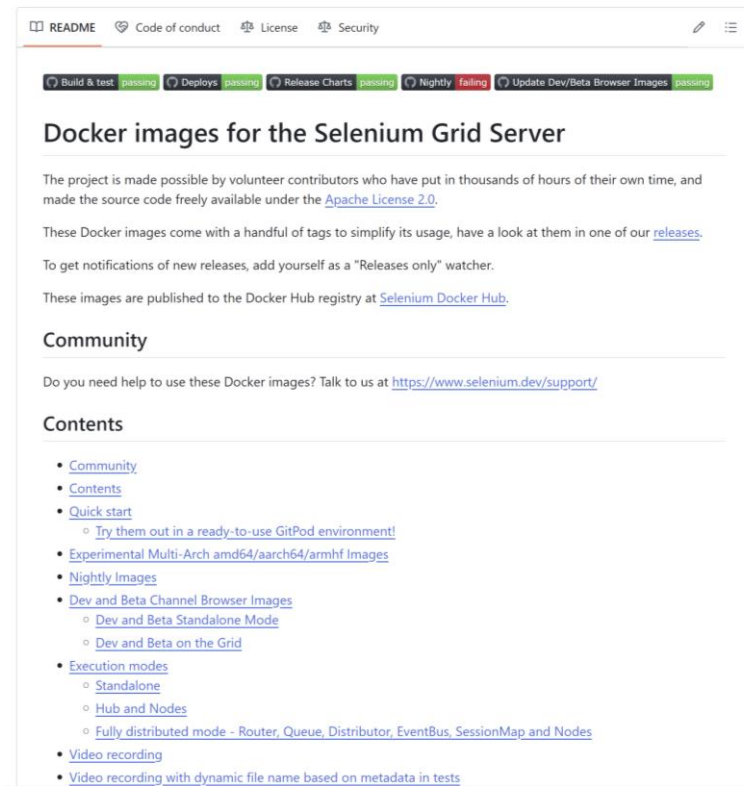
```
{
  "SessionId": "a433343ec6e678b1bc17a93bbbf6aea7",
  "Status": 0,
  "Value": { "AcceptSslCerts": true,
    "ApplicationCacheEnabled": false,
    "BrowserConnectionEnabled": false,
    "BrowserName": "chrome",
    "Chrome": { "UserDataDir":
      "/var/folders/p6/ll1grbcs4jv_k7675qv47l6m0000gn/T/.org.chromium.Chromium.wEZRL6" },
    "CssSelectorsEnabled": true,
    "DatabaseEnabled": false,
    "HandlesAlerts": true,
    "JavascriptEnabled": true,
    "LocationContextEnabled": true,
    "NativeEvents": true,
    "Platform": "Mac OS X",
    "Rotatable": false,
    "TakesHeapSnapshot": true,
    "TakesScreenshot": true,
    "Version": "38.0.2125.111",
    "WebStorageEnabled": true }
}
```

Mehr Informationen:

<https://w3c.github.io/webdriver/>

Selenium Containers

<https://github.com/SeleniumHQ/docker-selenium>



The screenshot shows the GitHub repository page for SeleniumHQ/docker-selenium. At the top, there are navigation links: README, Code of conduct, License, and Security. Below these are several status bars indicating the success of various CI/CD pipelines: Build & test (passing), Deploys (passing), Release Charts (passing), Nightly (failing), and Update Dev/Beta Browser Images (passing). The main heading is "Docker images for the Selenium Grid Server". The text describes the project as being made possible by volunteer contributors and made available under the Apache License 2.0. It mentions that the Docker images come with a handful of tags to simplify usage and that users can get notifications of new releases by adding themselves as a "Releases only" watcher. It also states that the images are published to the Docker Hub registry at Selenium Docker Hub. Below this is a "Community" section with a link to the Selenium dev support page. The "Contents" section lists several links: Community, Contents, Quick start (with a sub-link to try them out in a ready-to-use GitPod environment), Experimental Multi-Arch amd64/aarch64/armhf Images, Nightly Images, Dev and Beta Channel Browser Images (with sub-links for Dev and Beta Standalone Mode and Dev and Beta on the Grid), Execution modes (with sub-links for Standalone, Hub and Nodes, and Fully distributed mode - Router, Queue, Distributor, EventBus, SessionMap and Nodes), Video recording, and Video recording with dynamic file name based on metadata in tests.

README Code of conduct License Security

Build & test **passing** Deploys **passing** Release Charts **passing** Nightly **failing** Update Dev/Beta Browser Images **passing**

Docker images for the Selenium Grid Server

The project is made possible by volunteer contributors who have put in thousands of hours of their own time, and made the source code freely available under the [Apache License 2.0](#).

These Docker images come with a handful of tags to simplify its usage, have a look at them in one of our [releases](#).

To get notifications of new releases, add yourself as a "Releases only" watcher.

These images are published to the Docker Hub registry at [Selenium Docker Hub](#).

Community

Do you need help to use these Docker images? Talk to us at <https://www.selenium.dev/support/>

Contents

- [Community](#)
- [Contents](#)
- [Quick start](#)
 - [Try them out in a ready-to-use GitPod environment!](#)
- [Experimental Multi-Arch amd64/aarch64/armhf Images](#)
- [Nightly Images](#)
- [Dev and Beta Channel Browser Images](#)
 - [Dev and Beta Standalone Mode](#)
 - [Dev and Beta on the Grid](#)
- [Execution modes](#)
 - [Standalone](#)
 - [Hub and Nodes](#)
 - [Fully distributed mode - Router, Queue, Distributor, EventBus, SessionMap and Nodes](#)
- [Video recording](#)
- [Video recording with dynamic file name based on metadata in tests](#)



Page Object Pattern

Page Object Pattern

Purpose

- Represents a View / Web Page
- Encapsulates all functions provided by the View / Web Page

Methods

- Abstraction to interact with UI elements
- Returns additional Page Objects
 - UI navigation/hierarchy is represented by other Page Objects
- Includes query methods to check the state of the View / Web Page
 - HasErrors()
 - IsCustomerInList(string customerName)

Beispiel Page Object

Shared Actions (base class)

- HomePage NavigateHome()
- CustomerList NavigateCustomers()
- OrderList NavigateOrders()

CustomerList Actions

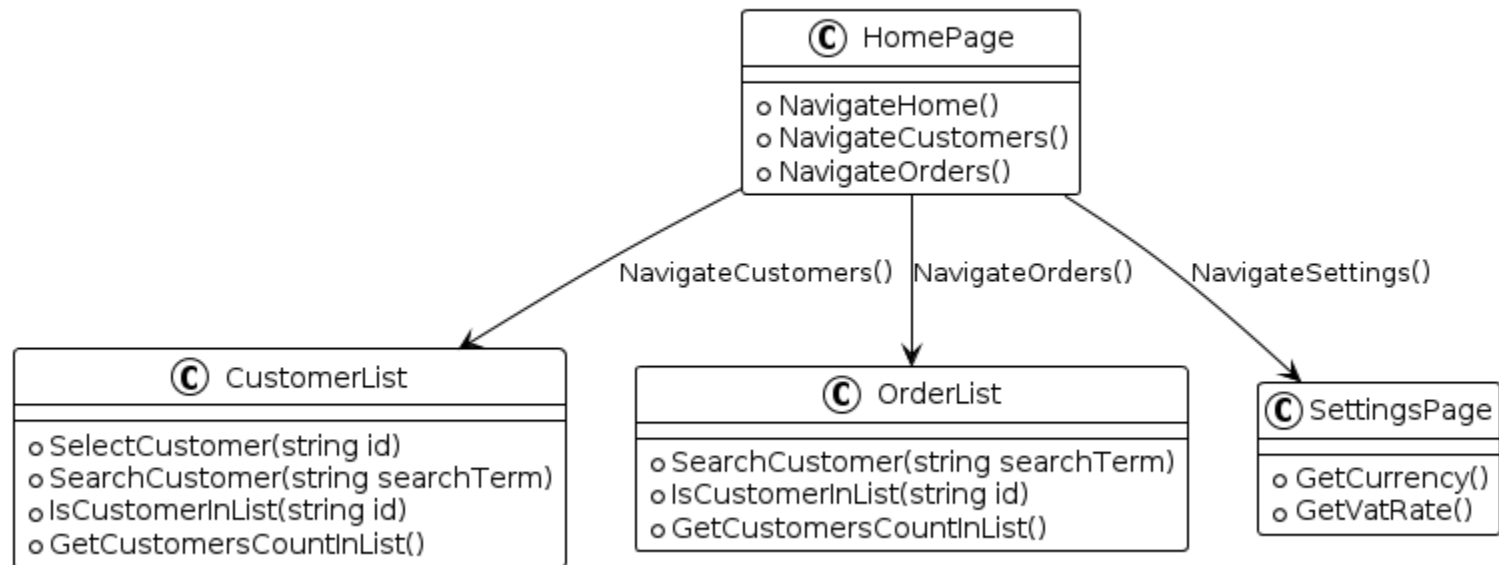
- CustomerDetail SelectCustomer(string id)
- CustomerList SearchCustomer(string searchterm)

```
Assert.IsTrue(home.NavigateCustomers()  
    .SearchCustomer («Meier»)  
    .IsCustomerInList («1»),  
    «Customer not found!»);
```

CustomerList Queries

- IsCustomerInList(string id)
- GetCustomersCountInList()

Page Objects



Advantages of Page Object Pattern

Application of software engineering principles

- Maintainable tests
- SOLID, DRY, ...

Easy readability of scenarios

Test focus on interaction, not plumbing

- Separation of Concerns
- UI interaction is abstracted



Run a UI in a Container

Run UI Tests in a Container

- UI Options

- Headless
- Xvfb (X virtual frame buffer)

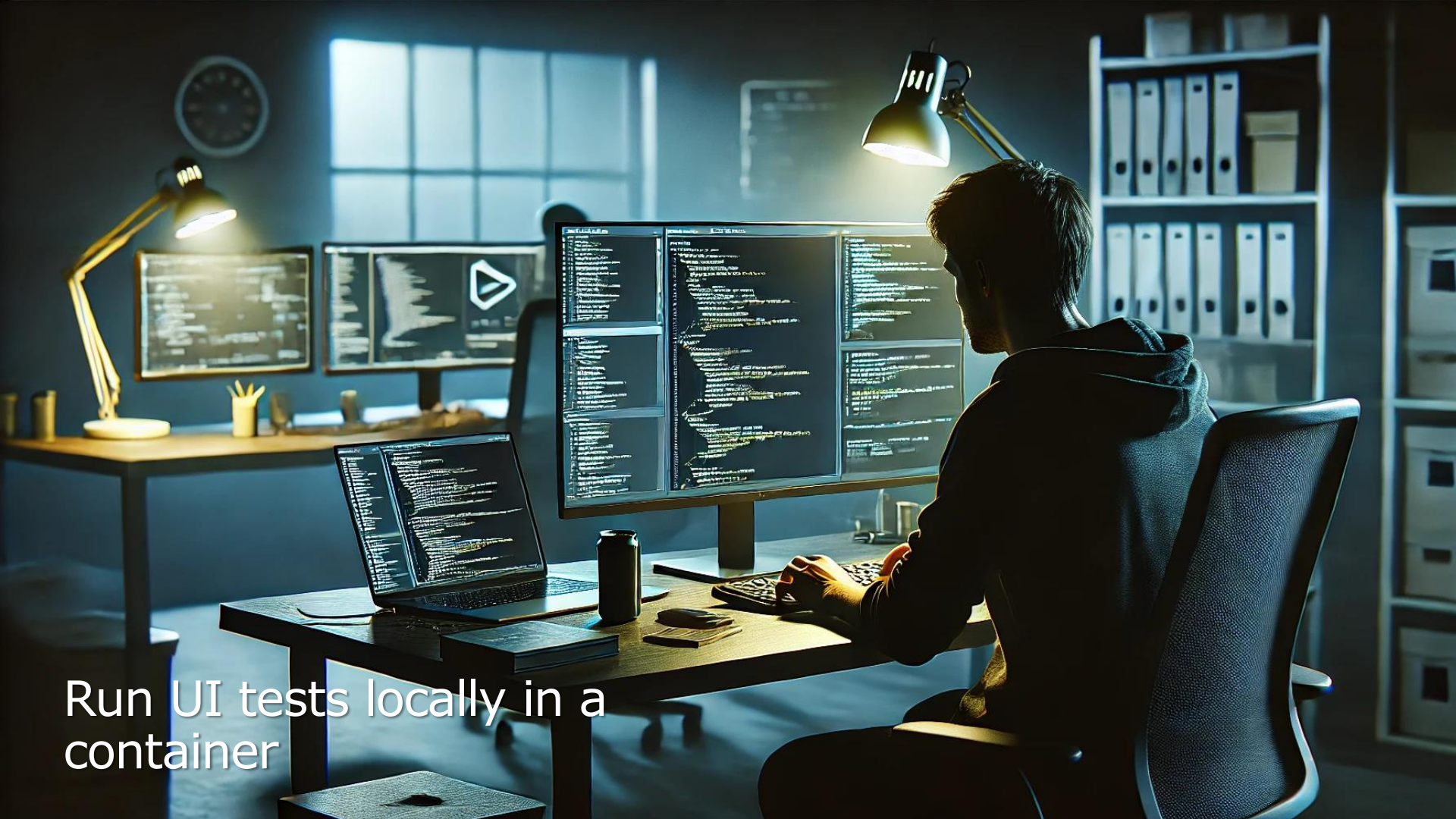
- Install the software

- Dockerfile
- Versioned container images

- Use IPC to execute tests

- Selenium Wire Protocol / Grid Server
- Playwright Remoting
-

- Connect as VNC client



Run UI tests locally in a
container



Docker in Pipelines

4tecture[®]
empower your software solutions



IT WORKS ON MY MACHINE



THEN WE'LL SHIP YOUR MACHINE



AND THAT IS HOW DOCKER WAS BORN

Containers

- Reduce the complexity of maintaining build servers
- Use a «cook book» Dockerfile to build images → Configuration as Code
- Layering-System → re-use and separation of concerns
- Fast (instant) start
- Low footprint

CI Environment Challenges

- Storing all VM images need a lot of storage
- Build VM snapshots not integrated in standard process
- Where to store the VM configuration?



Container Jobs



- Work with hosted pools and private pools
- Pipeline Job runs inside the container
- Each step runs inside the container

Container Jobs

```
trigger:  
- master
```

```
pool:  
  #vmImage: 'windows-latest'  
  default
```

```
container: mcr.microsoft.com/dotnet/framework/sdk:4.7.2-windowsservercore-ltsc2019
```

```
variables:  
  solution: '**/*.sln'  
  buildPlatform: 'Any CPU'  
  buildConfiguration: 'Release'
```

```
steps:  
- task: NuGetToolInstaller@1  
  
- task: NuGetCommand@2  
  inputs:  
    restoreSolution: '$(solution)'
```

```
- task: VSBUILD@1  
  inputs:  
    solution: '$(solution)'  
    platform: '$(buildPlatform)'  
    configuration: '$(buildConfiguration)'
```

```
- task: VSTest@2  
  inputs:  
    testSelector: 'testAssemblies'  
    searchFolder: '$(System.DefaultWorkingDirectory)'  
    vstestLocationMethod: 'location'  
    vstestLocation: 'C:\Program Files (x86)\Microsoft Visual  
    platform: '$(buildPlatform)'  
    configuration: '$(buildConfiguration)'
```

```
- publish: $(System.DefaultWorkingDirectory)\ConsoleApp\bin\  
  artifact: ConsoleApp
```

```
container:  
  image: myprivate.azurecr.io/windowsservercore:1803  
  endpoint: my_acr_connection
```

```
container:  
  image: ubuntu:16.04  
  options: --hostname container-test --ip 192.168.0.1
```

```
resources:  
  containers:  
  - container: pycontainer  
    image: python:3.8
```

```
steps:  
- task: SampleTask@1  
  target: host  
- task: AnotherTask@1  
  target: pycontainer
```

```
resources:  
  containers:  
  - container: u14  
    image: ubuntu:14.04  
  
  - container: u16  
    image: ubuntu:16.04  
  
  - container: u18  
    image: ubuntu:18.04
```

```
jobs:  
- job: RunInContainer  
  pool:  
    vmImage: 'ubuntu-16.04'
```

```
strategy:  
  matrix:  
    ubuntu14:  
      containerResource: u14  
    ubuntu16:  
      containerResource: u16  
    ubuntu18:  
      containerResource: u18
```

```
container: $[ variables['containerResource'] ]
```


A background image showing a rowing team in a boat. The rowers are wearing blue and red uniforms and are captured in a synchronized rowing motion. The focus is on the oars and the rowers' hands, with the water visible in the background.

Container Jobs and Service Containers

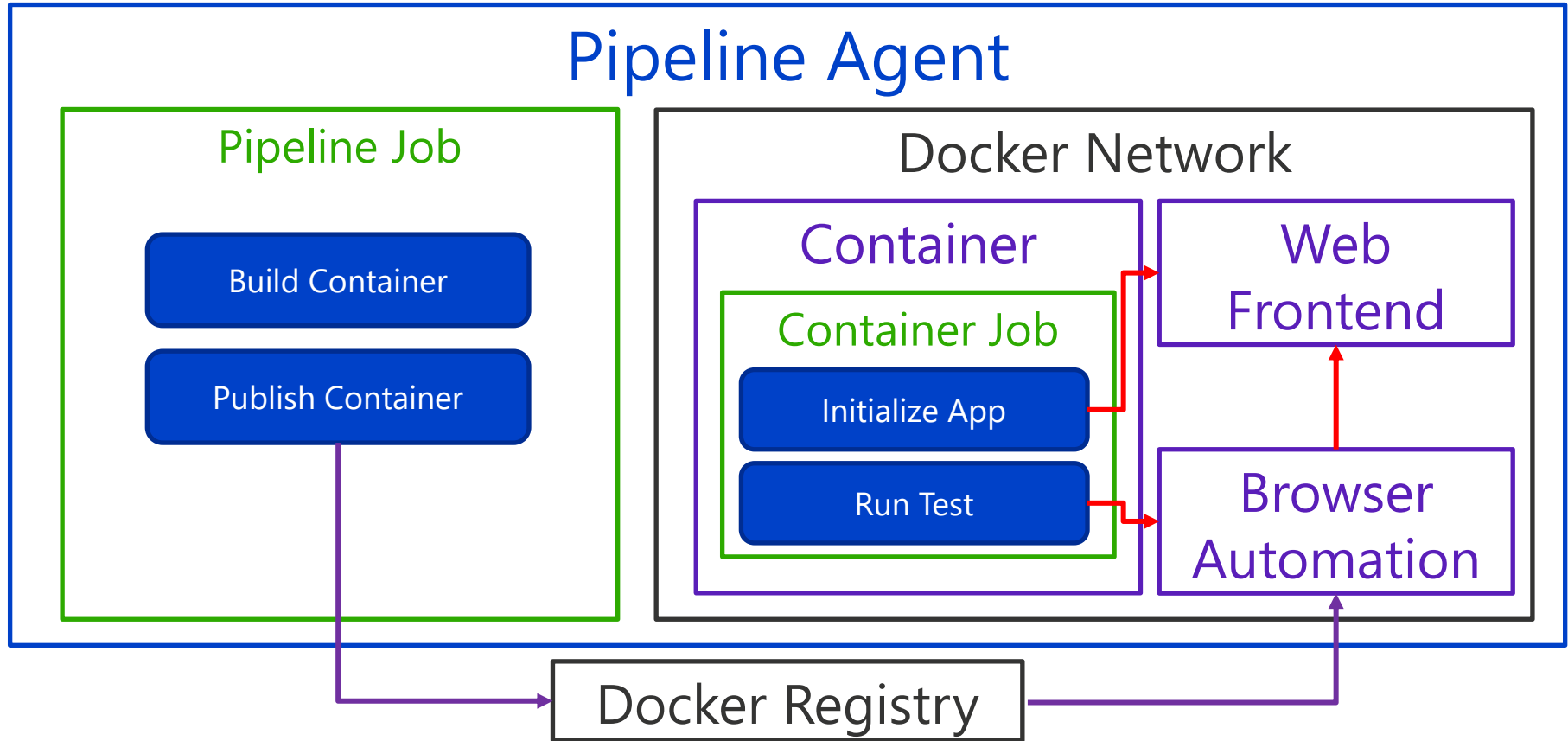
4tecture[®]
empower your software solutions

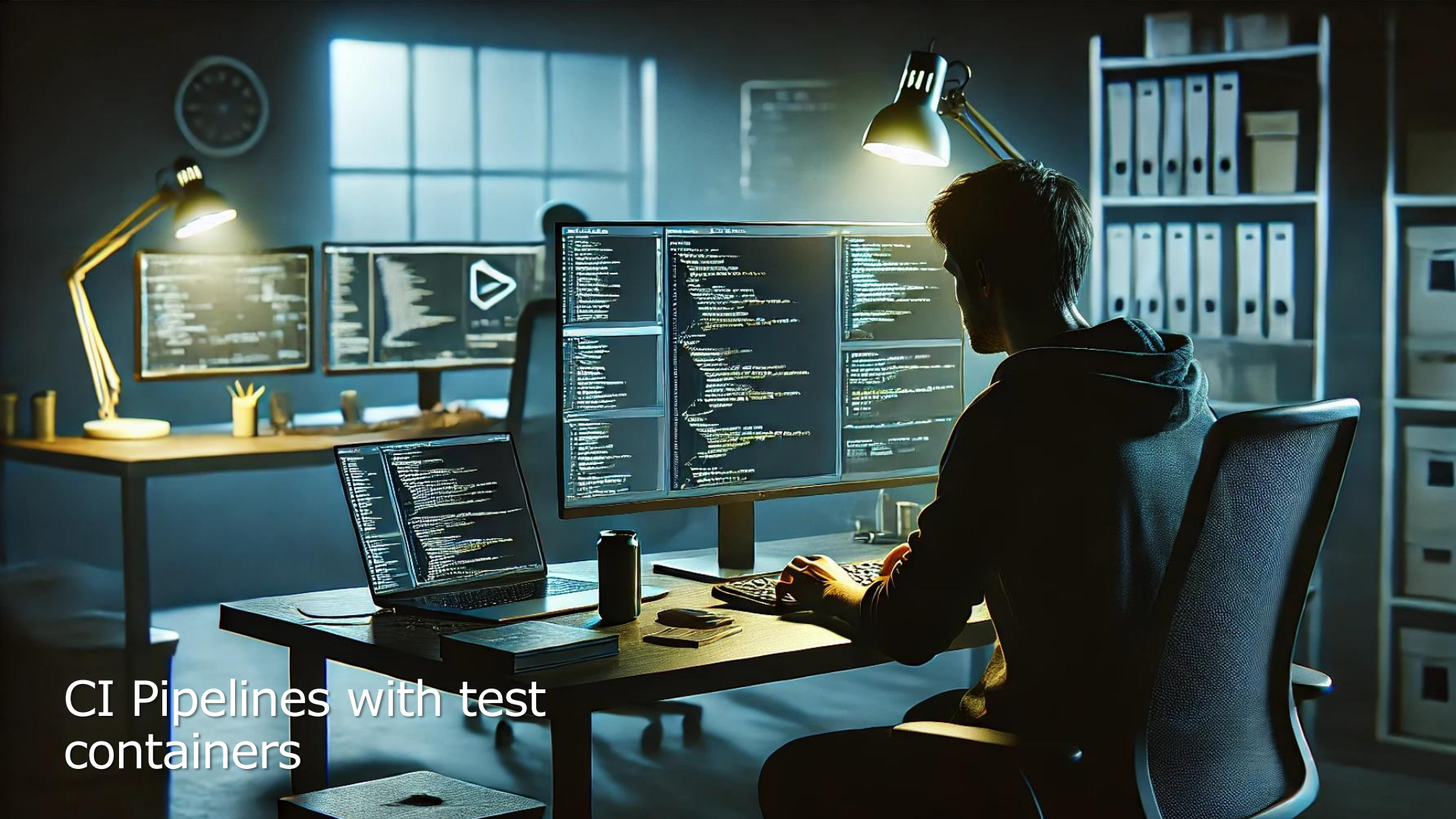
CI Challenges

- Complex System Testing Environment Setup
- Shift-Left / Fast-Feedback
- Target multiple versions



Bringing all together – Frontend





CI Pipelines with test
containers

DevFunWeb-CI

Variables

Run

Show assistant

master DevFun / .azure-pipelines/devfunweb-ci.yml

```
23 resources:
24   containers:
25     - container: web
26       image: $(AzureContainerRegistry)/$(DockerRepositoryName)/$(Build.BuildNumber)
27       endpoint: 4taksDemoAcr
28       env:
29         - devfunoptions_deploymentenvironment: "Development"
30         - connectionstrings_url: "${{ parameters.integrationTestBackendUrl }}"
31     - container: worker
32       image: $(AzureContainerRegistry)/linuxworker:latest
33       endpoint: 4taksDemoAcr
34     - container: seleniumChrome
35       image: selenium/node-chrome:4.22.0
36       volumes:
37         - /dev/shm:/dev/shm
38       env:
39         - SE_EVENT_BUS_HOST: seleniumhub
40         - SE_EVENT_BUS_PUBLISH_PORT: 4442
41         - SE_EVENT_BUS_SUBSCRIBE_PORT: 4443
42     - container: seleniumEdge
43       image: selenium/node-edge:4.22.0
44       volumes:
45         - /dev/shm:/dev/shm
46       env:
47         - SE_EVENT_BUS_HOST: seleniumhub
48         - SE_EVENT_BUS_PUBLISH_PORT: 4442
49         - SE_EVENT_BUS_SUBSCRIBE_PORT: 4443
50     - container: seleniumFirefox
51       image: selenium/node-firefox:4.22.0
52       volumes:
53         - /dev/shm:/dev/shm
54       env:
55         - SE_EVENT_BUS_HOST: seleniumhub
56         - SE_EVENT_BUS_PUBLISH_PORT: 4442
57         - SE_EVENT_BUS_SUBSCRIBE_PORT: 4443
58     - container: seleniumvideo
59       image: selenium/video:ffmpeg-6.1.1-20240621
60       volumes:
61         - /home/vsts/work/_temp/videos:/videos
62       env:
63         - DISPLAY_CONTAINER_NAME: selenium
64         - SE_NODE_GRID_URL: http://seleniumhub:4444
65         - SE_VIDEO_FILE_NAME: auto
66     - container: seleniumhub
67       image: selenium/hub:4.22.0
68       ports:
69         - 4442:4442
70         - 4443:4443
71         - 4444:4444
```

Container
Resources

< DevFunWeb-CI

Variables

Validate and save



master

DevFun / .azure-pipelines/devfunweb-ci.yml

Show assistant

```
139 -- stage: IntegrationTests
140 -- displayName: 'Run Integration Tests'
141 -- dependsOn: Build
142 -- jobs:
143 --   job: SystemTest
144 --     displayName: 'System tests with Selenium UI Tests'
145 --     pool:
146 --       vmImage: 'ubuntu-latest'
147 --       #name: Default
148 --       #name: LocalAgents
149 --     strategy:
150 --       matrix:
151 --         chrome:
152 --           containerResource: seleniumChrome
153 --           targetBrowser: Chrome
154 --         edge:
155 --           containerResource: seleniumEdge
156 --           targetBrowser: Edge
157 --         firefox:
158 --           containerResource: seleniumFirefox
159 --           targetBrowser: Firefox
160 --     services:
161 --       web: web
162 --       seleniumhub: seleniumhub
163 --       selenium: $[ variables['containerResource'] ]
164 --       seleniumvideo: seleniumvideo
165 --       container: worker
166 --     steps:
167 --       - checkout: none
168 --       - download: current
169 --       - artifact: SystemTests
170 --       - displayName: Download the SystemTests artifact
171 --       - View template
172 --       - template: templates/test_runseleniumtests.yml
173 --       - parameters:
174 --         - SystemTestArtifactLocation: $(Pipeline.Workspace)/SystemTests
175 --         - RunSettingsFileName: container.runsettings
176 --         - TargetBrowser: $(targetBrowser)
177 -- stage: PullRequest
178 -- displayName: 'Update Pull Request'
```

Matrix → run test in browser environments in parallel

Dynamic service container selection based on matrix variable

DevFunWeb-CI

Variables

Validate and save

master

DevFun / .azure-pipelines/devfunweb-ci.yml

Show assistant

```

141  - services:
142    - web: web
143    - selenium: $[ variables['containerResource'] ]
144    - seleniumVideo: seleniumVideo
145    - container: worker
146  - steps:
147    - checkout: none
148    - download: current
149    - artifact: SystemTests
150    - displayName: Download the SystemTests artifact
151      View template
152    - template: templates/test_runseleiumtests.yml
153      parameters:
154        SystemTestArtifactLocation: $(Pipeline.Workspace)/SystemTests
155        RunSettingsFileName: container.runsettings
156        TargetBrowser: $(targetBrowser)
157  - stage: PullRequest
158    - displayName: 'Update Pull Request'
159    - dependsOn: IntegrationTests
160    - condition: and(succeeded(), eq(variables['Build.Reason'], 'PullRequest'))
161    - pool:
162      - vmImage: 'ubuntu-latest'
163      - #name: Default
164      - #name: LocalAgents
165    - jobs:
166      - job: UpdatePullRequest
167        - displayName: 'Update the pull request with policies'
168        - steps:
169          - checkout: none
170          - Settings
171          - task: PullRequestStatus@0
172            - displayName: 'Initialize status'
173            - inputs:
174              - name: 'deploy-devfunweb'
175              - action: 'Create'
176              - state: 'pending'
177              - description: 'Deploy DevFun Web'
178  - stage: TagReleaseVersion
179    - displayName: 'Tag the release version'
  
```

Get the test assemblies

Run the UI tests

DevFunWeb-CI

Save

master

DevFun / .azure-pipelines/templates/test_runeleniumtests.yml Template

Show assistant

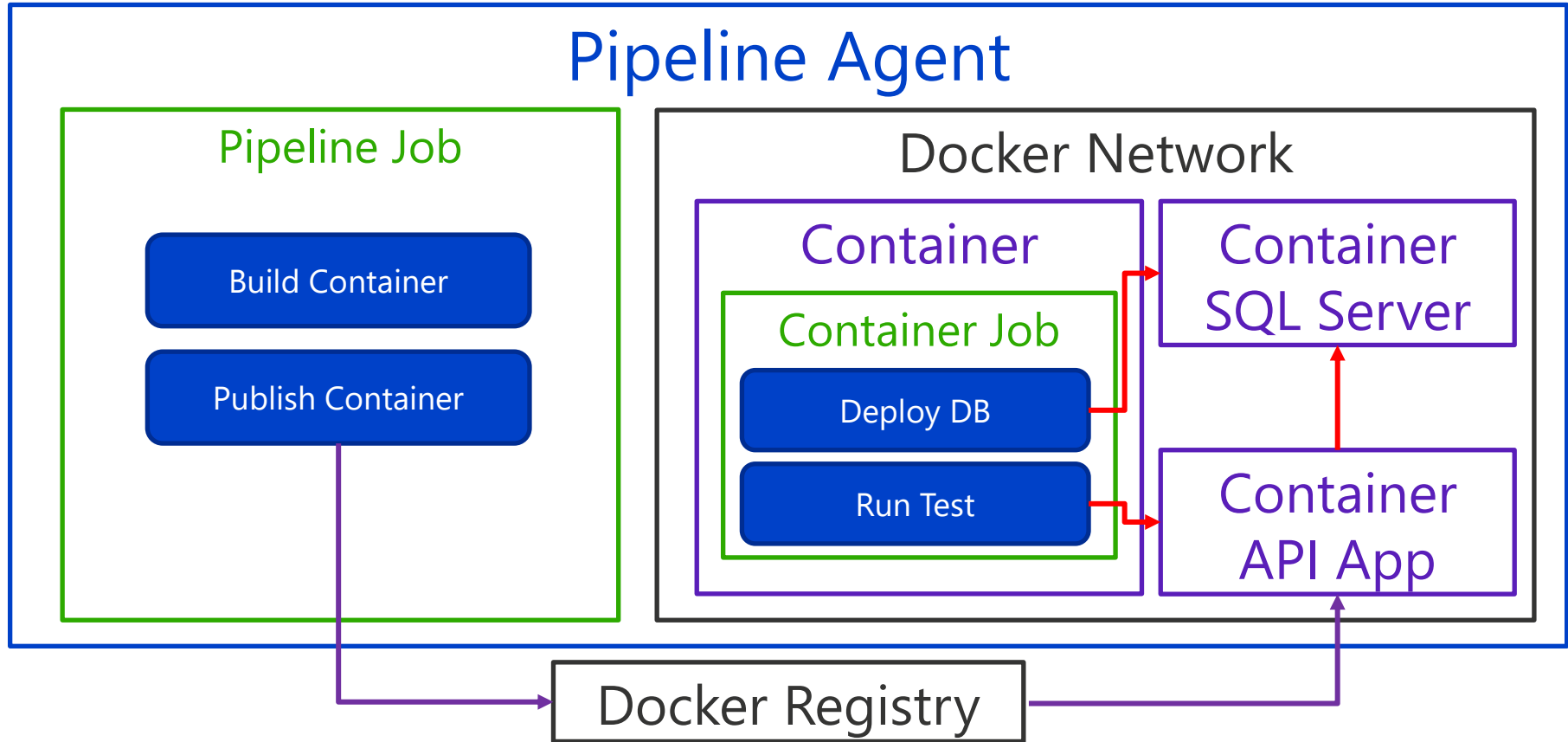
```
1 parameters:
2   - name: SystemTestArtifactLocation
3     - type: string
4     - default: "SystemTests"
5   - name: RunSettingsFileName
6     - type: string
7     - default: ""
8   - name: TargetBrowser
9     - type: string
10    - default: "Firefox"
11
12 steps: ...
13   - script: |
14     - mkdir -p /home/vsts/work/_temp/videos
15     - sudo chmod 777 /home/vsts/work/_temp/videos
16     - mkdir $(Build.ArtifactStagingDirectory)/TestResults
17     - displayName: 'Create and set permissions for /home/vsts/work/_temp/videos'
18     - target: host
19
20   - task: Tokenizer@0
21     - displayName: 'Tokenizer to set correct Urls'
22     - inputs:
23       - sourceFilesPattern: '**/*.runsettings'
24       - sourcePath: ${ parameters.SystemTestArtifactLocation }
25
26   - task: DotNetCoreCLI@2
27     - displayName: "Run UI Tests"
28     - inputs:
29       - command: test
30       - projects: "${ parameters.SystemTestArtifactLocation }/**/*Ui.Tests.dll"
31       - arguments: "--logger:trx --settings ${ parameters.SystemTestArtifactLocation }/${ parameters.RunSettingsFileName } --targetBrowser=${ parameters.Target
32
33   - task: CopyFiles@2
34     - displayName: 'Copy test result video files to: $(Build.ArtifactStagingDirectory)/TestResults/videos'
35     - inputs:
36       - SourceFolder: $(Agent.TempDirectory)/videos
37       - Contents: '**/*.mp4'
38       - TargetFolder: $(Build.ArtifactStagingDirectory)/TestResults/videos
39       - condition: always()
40
41   - task: PublishPipelineArtifact@1
42     - displayName: 'Publish test result artifacts'
```

Configure test execution

Run the tests

Get the videos

Bringing all together – Backend



← DevFunApi-CI

Variables

Save

master

DevFun / .azure-pipelines/devfunapi-ci.yml *

Show assistant

```

26
27 resources:
28   containers:
29     - container: sql2022
30       image: mcr.microsoft.com/mssql/server:2022-latest
31       env:
32         SA_PASSWORD: "$(containerdbpassword)"
33         ACCEPT_EULA: "Y"
34     - container: sql2019
35       image: mcr.microsoft.com/mssql/server:2019-latest
36       env:
37         SA_PASSWORD: "$(containerdbpassword)"
38         ACCEPT_EULA: "Y"
39     - container: sql2017
40       image: mcr.microsoft.com/mssql/server:2017-latest
41       env:
42         SA_PASSWORD: "$(containerdbpassword)"
43         ACCEPT_EULA: "Y"
44     - container: api
45       image: '$(ImageName):$(Build.BuildNumber)'
46       endpoint: 4taksDemoAcr
47       env:
48         DEVFUNCTIONS__DEPLOYMENTENVIRONMENT: "Development"
49         CONNECTIONSTRINGS__DEVFUNDATABASE: 'Server=tcp:$(containerdbserver),1433;Database=$(containerdbname);User ID=$(containerdbuser);Password=$(containerdbpassword);'
50         LICENSE__LICENSEDATA: "$(LicenseData)"
51     - container: worker
52       image: 4taksdemoacr.azurecr.io/linuxworker:latest
53       endpoint: 4taksDemoAcr
54
55 stages:
56   - stage: Build
57     jobs:
58       - job: CI_Build...
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110   - job: DataInitializer_Build...
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136

```

Container Resources

DevFunApi-CI

Variables

Save

master

DevFun / .azure-pipelines/devfunapi-ci.yml *

Show assistant

```

168 - stage: IntegrationTests
169   displayName: 'Run Integration Tests'
170   dependsOn: Build
171   jobs:
172   - job: SystemTest
173     displayName: 'System tests with SQL Server'
174     pool:
175       vmImage: 'ubuntu-latest'
176       #name: Default
177     strategy:
178       matrix:
179         sql2017:
180           containerResource: sql2017
181         sql2019:
182           containerResource: sql2019
183         sql2022:
184           containerResource: sql2022
185     services:
186       sql: $[ variables['containerResource'] ]
187       api: api
188     container: worker
189     steps:
190     - download: current
191       artifact: SystemTests
192       displayName: Download the SystemTests artifact
193     - download: current
194       artifact: TestDataInitializer
195       displayName: Download the TestDataInitializer artifact
196     - download: current
197       artifact: dacpacs
198       displayName: Download the Database artifact
199     View template
200     - template: templates/sql_DeployDacPac.yml
201       parameters:
202         dbServernameFqdn: sql
203         dbName: $(containerdbname)

```

Matrix → run test in different environments in parallel

Dynamic service container selection based on matrix variable

DevFunApi-CI

Variables

Save

master

DevFun / .azure-pipelines/devfunapi-ci.yml *

Show assistant

```

181     sql2019:
182     containerResource: sql2019
183     sql2022:
184     containerResource: sql2022
185     services:
186     sql: $[ variables['containerResource'] ]
187     api
188     container: worker
189     steps:
190     - download: current
191       artifact: SystemTests
192       displayName: Download the SystemTests artifact
193     - download: current
194       artifact: TestDataInitializer
195       displayName: Download the TestDataInitializer artifact
196     - download: current
197       artifact: dacpacs
198       displayName: Download the Database artifact
199     View template
200     template: templates/sql_DeployDacPac.yml
201     parameters:
202     dbServerNameFqdn: sql
203     dbName: $(containerdbname)
204     dbUser: $(containerdbuser)
205     dbPassword: $(containerdbpassword)
206     dacpacFile: $(DacpacFile)
207     trustServerCertificate: true
208     View template
209     template: templates/test_runapitests.yml
210     parameters:
211     SystemTestArtifactLocation: $(Pipeline.Workspace)/SystemTests
212     DataInitializerArtifactLocation: $(Pipeline.Workspace)/TestDataInitializer
213     ApiUrl: http://api
214     RunSettingsFileName: container.runsettings
215     -- stage: PullRequest
    
```

DB schema deployment

Run integration / system tests

K8sDemo



Overview

Boards

Repos

Pipelines

Pipelines

Environments

Releases

Library

Task groups

Deployment groups

Test Plans

Artifacts

Project settings



← DevFunApi-CI

Variables

Save

master

DevFun / .azure-pipelines/devfunapi-ci.yml *

Show assistant

```
181 .....sql2019:
182 .....containerResource: sql2019
183 .....sql2022:
184 .....containerResource: sql2022
185 .....services:
186 .....sql: $[ variables['containerResource'] ]
187 .....api: api
188 .....container: worker
189 .....steps:
190 .....- download: current
191 .....  artifact: SystemTests
192 .....  displayName: Download the SystemTests artifact
193 .....- download: current
194 .....  artifact: TestDataInitializer
195 .....  displayName: Download the TestDataInitializer artifact
196 .....- download: current
197 .....  artifact: dacpacs
198 .....  displayName: Download the Database artifact
```

View template

```
199 .....- template: templates/sql_DeployDacPac.yml
200 .....  parameters:
201 .....    dbServerNameFqdn: sql
202 .....    dbName: $(containerdbname)
203 .....    dbUser: $(containerdbuser)
204 .....    dbPassword: $(containerdbpassword)
205 .....    dacpacFile: $(DacpacFile)
206 .....    trustServerCertificate: true
```

View template

```
207 .....- template: templates/test_runapitests.yml
208 .....  parameters:
209 .....    SystemTestArtifactLocation: $(Pipeline.Workspace)/SystemTests
210 .....    DataInitializerArtifactLocation: $(Pipeline.Workspace)/TestDataInitializer
211 .....    ApiUrl: http://api
212 .....    RunSettingsFilename: container.runsettings
213
214 .....- stage: PullRequest
```

Docker network container alias

Docker network container alias

A background image showing a rowing team in a boat. The rowers are wearing blue and red uniforms and are captured in a synchronized rowing motion. The focus is on the oars and the rowers' hands. The word "Conclusion" is overlaid in a large, blue, sans-serif font.

Conclusion

4tecture[®]
empower your software solutions

Conclusion

- Continue with your container strategy also for UI testing
- Shift-left into CI, later full deployment
- Tests run within agent (service containers), simpler setup, no additional resources

A close-up, low-angle shot of several rowers in a boat, wearing blue and red uniforms, pulling their oars. The oars are black with yellow handles and are connected to a complex mechanical system. The background is a bright, overexposed sky.

Q & A

4tecture[®]
empower your software solutions

Thank you for your attention!

If you have any questions do not hesitate to contact us:

4tecture GmbH
Industriestrasse 25
CH-8604 Volketswil

+41 44 508 37 00
info@4tecture.ch
www.4tecture.ch

Marc Müller
Principal Consultant

marc.mueller@4tecture.ch
@muellermarc
www.powerofdevops.com



A background image showing several hands of different skin tones reaching towards the center, where they are assembling a small cluster of interlocking wooden puzzle pieces. The pieces are colored in a variety of shades including light brown, white, green, and reddish-brown. The overall scene is softly blurred, focusing attention on the hands and the puzzle pieces.

4tecture[©]
empower your software solutions