

Wüstenrot & Würtembergische AG

SAS Viya Deploymentverfahren

Markus Ullrich, Andre Zieher, 15.05.2025

Inhalt

1. Die W&W Gruppe
2. SAS Viya in der W&W
3. Entwicklungsprozess
4. Deploymentverfahren
 - Technische Umsetzung
 - Live Demo
 - Erreichte Ziele
 - Ausblick

Die W&W Gruppe

Der Vorsorge verbunden.
Das sind wir:
die W&W-Gruppe.







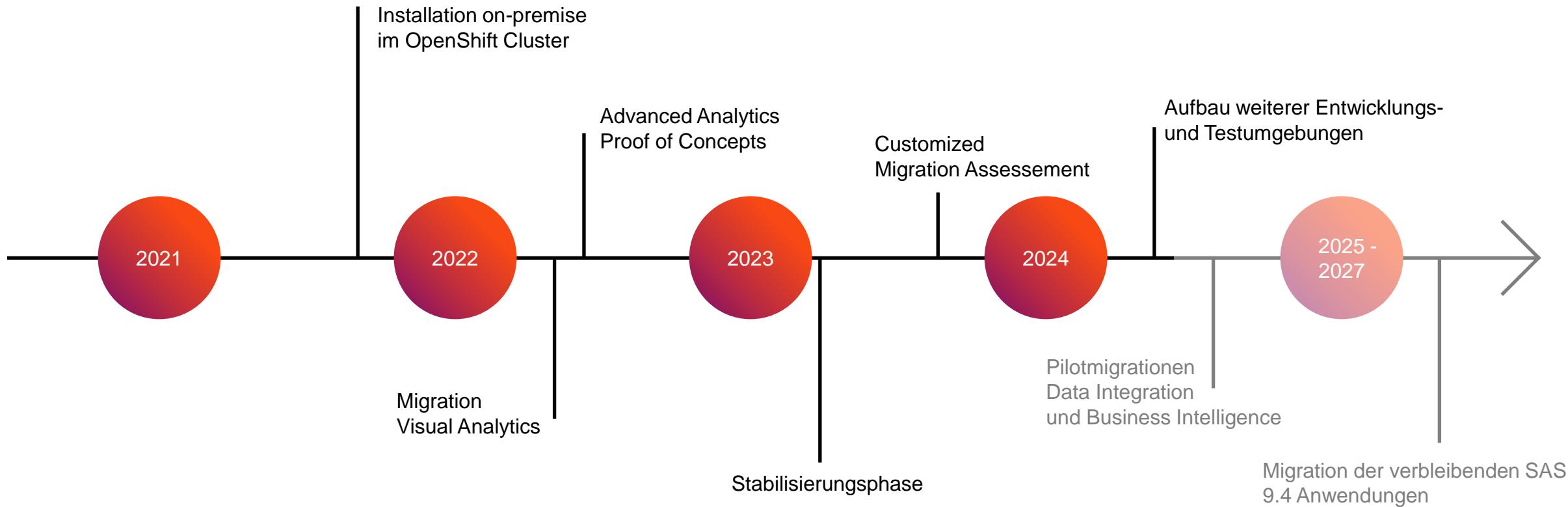
Unsere Unternehmenserfolge 2023

Neugeschäftsvolumen Wohnen (brutto)	21,5 Mrd. €
Versicherungsumsatz (brutto nach IFRS)	3,8 Mrd. €
Kapitalanlagen der W&W (nach IFRS)	38,9 Mrd. €
Konzernüberschuss (nach IFRS)	140,5 Mio. €
Eigenkapital (nach IFRS)	5 Mrd. €

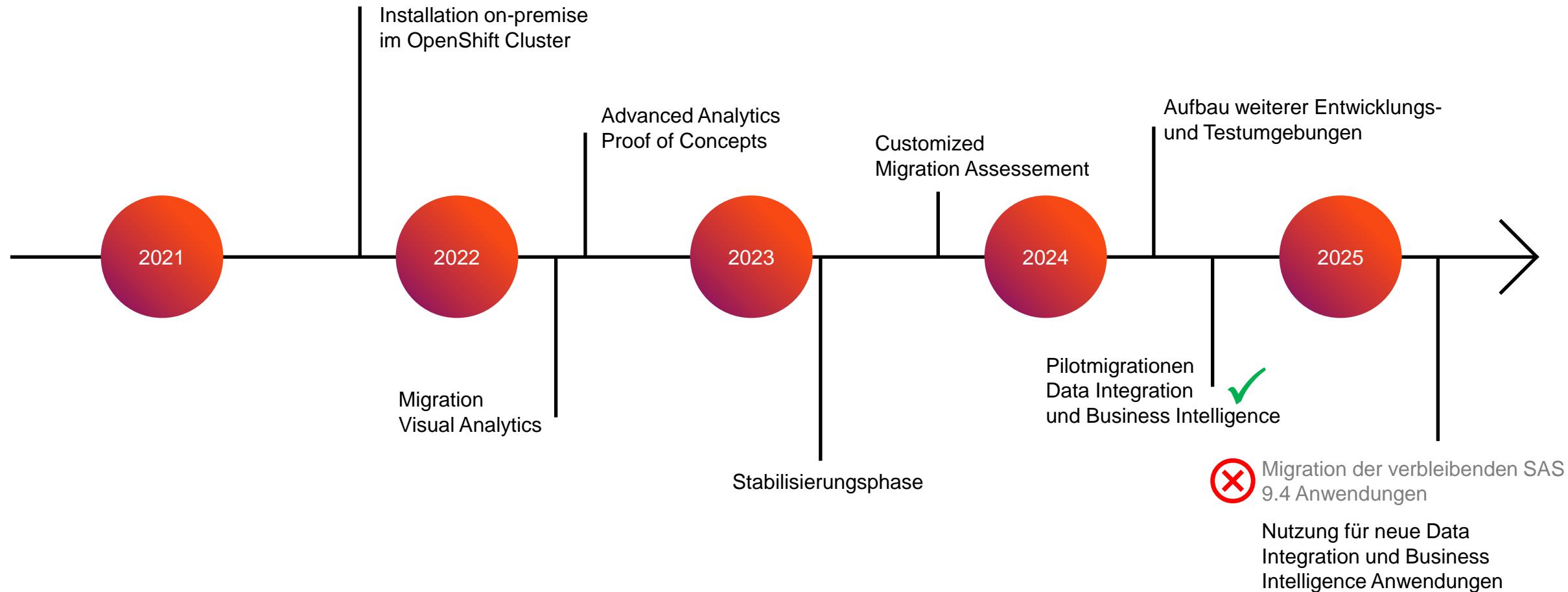
Quelle: Geschäftsbericht Wüstenrot und Württembergische AG 2023, Werte gerundet.

SAS Viya in der W&W

SAS Viya in der W&W – Rückblende PNT 2024



SAS Viya in der W&W

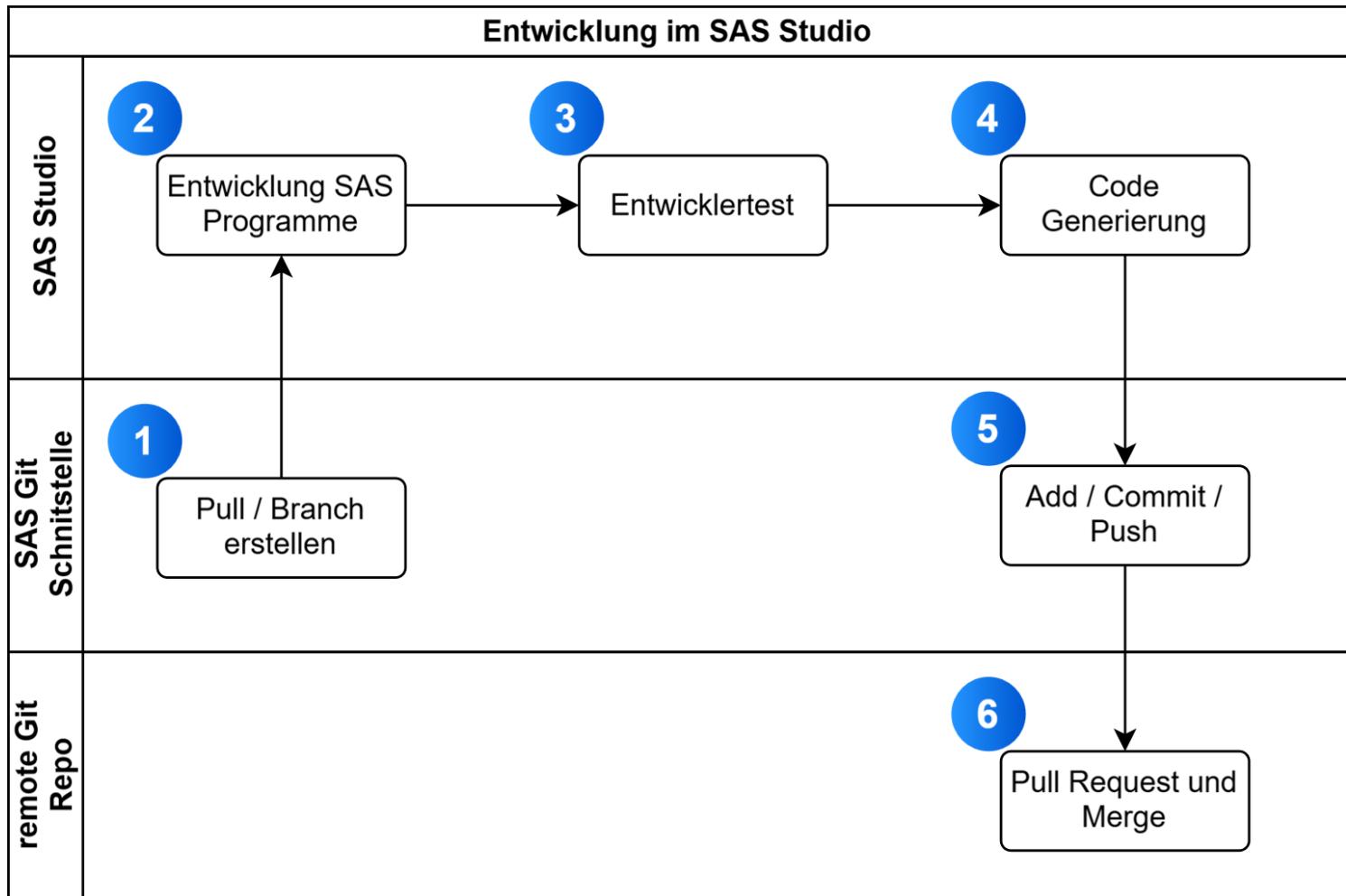


Entwicklungsprozess

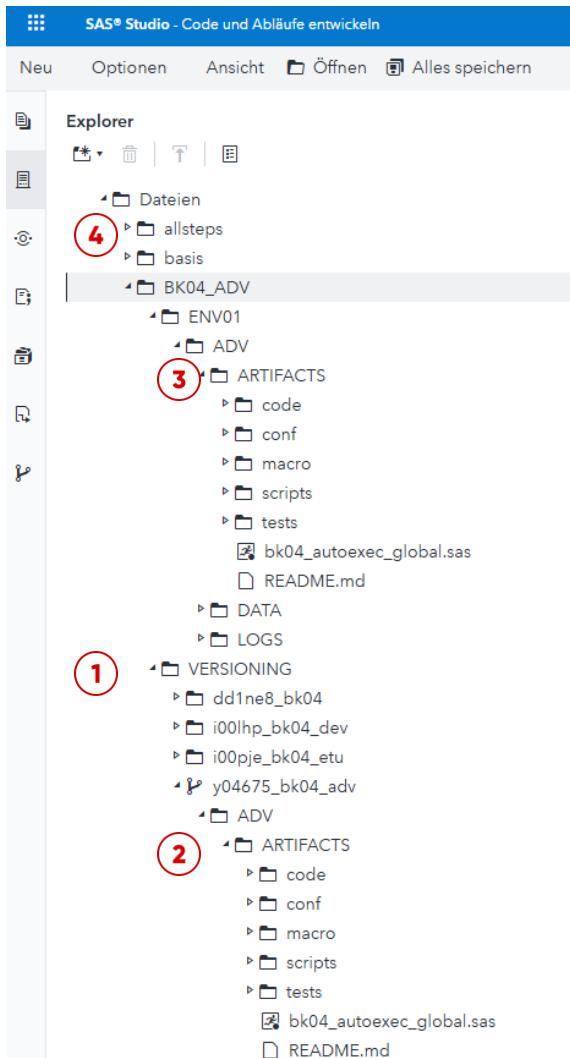
Entwicklungsprozess

1. Prozessfluss
2. Ablagestruktur
3. Versionierung / Git Integration
4. Code-Generierung

Prozessfluss



Ablagestruktur

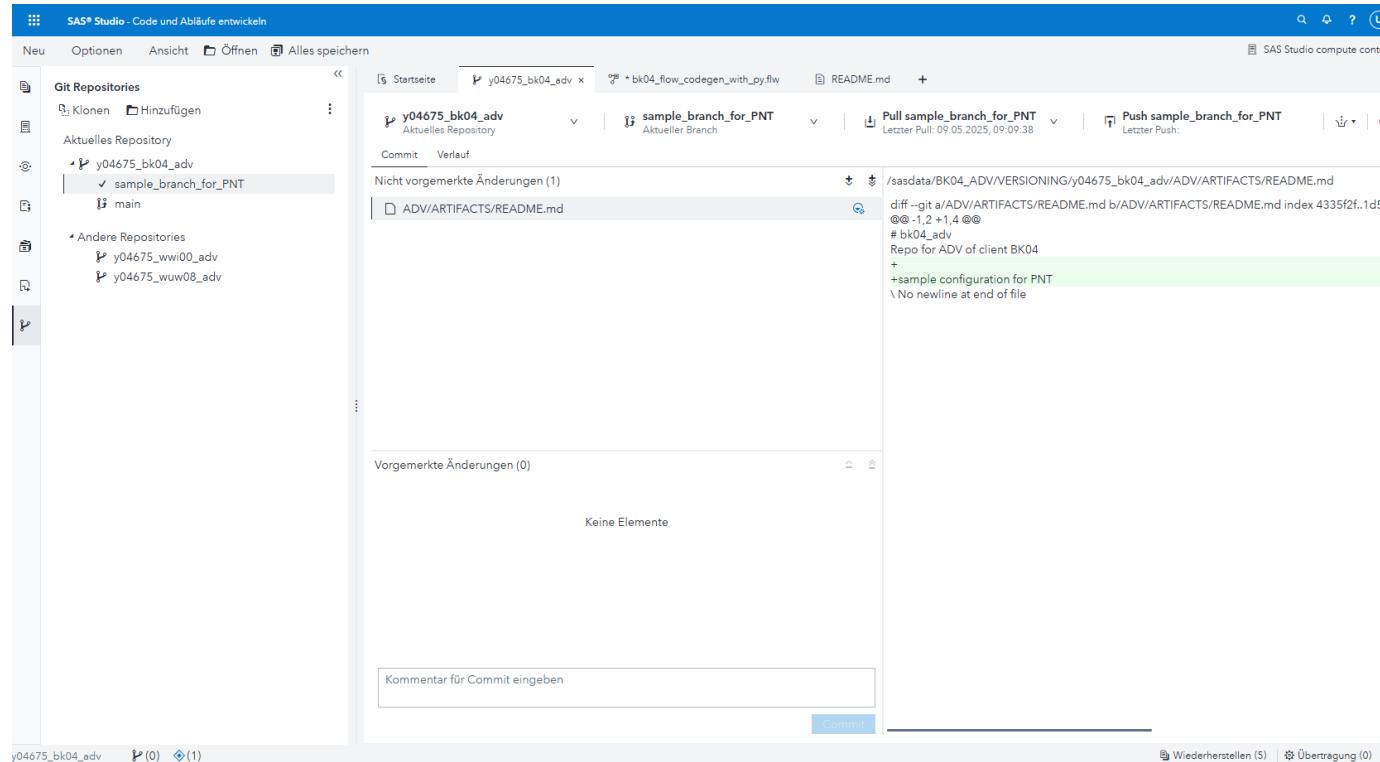


- Anbindung des Git Repositories nur im Filesystem möglich
 - Ablage von SAS Code und Flows im Filesystem

 - Jeder Entwickler arbeitet im geclonten Repository unter „VERSIONING“ **1**

 - ! Flows enthalten absolute Pfadreferenzen von Custom Steps im Filesystem
- Auszug JSON Struktur von test.flw:
- ```
"stepReference": {
 "type": "filesystem",
 "path": "/sasdata/ENV01/WWI00_SAS_PLATTFORM/VERSIONING/y04675/test/test.step"
```
- Probleme bei der Übertragung von Flows vom Entwicklungs- **2** in den Mandantenordner **3**
  - Kopieren aller Custom Steps bei Sitzungsstart nach „allsteps“ über ein Macro **4**

# Versionierung / Git Integration



- Versionierung kann vollständig über die SAS Git Schnittstelle abgebildet werden

! Keine Rückgabe von Fehlern über die Oberfläche

- Fehlermeldungen z.B. beim PUSH werden nicht angezeigt
- mit SAS Git Funktionen gelöst

Statusabfrage für SAS Funktionen:

```
data _null_;
rc= git_status("your-local-repository");
put rc=;
run;
```

<https://communities.sas.com/t5/SAS-Communities-Library/Demystifying-Git-Using-SAS-Studio-in-SAS-Viya-with-GitHub/ta-p/962687>  
[https://documentation.sas.com/doc/en/pgmsascdc/v\\_062/lefunctionsref/n1mlc3f9w9zh9fn13gswiq6hrt0.htm](https://documentation.sas.com/doc/en/pgmsascdc/v_062/lefunctionsref/n1mlc3f9w9zh9fn13gswiq6hrt0.htm)

# Code-Generierung

This SAS Flow can be used to generate SAS Code from specified SAS Flows.

Overall process:

- Configure the custom step 'flow\_select\_for\_codegen' and choose if you want to select a single SAS Flow or if you want to parse a folder containing SAS Flows.
- Execute the custom step 'flow\_select\_for\_codegen' and check output dataset. This dataset contains one or multiple entries with fully qualified names of SAS Flows.
- (Optional) Delete unwanted SAS Flow items from the Dataset generated by the custom step.
- Execute code step in order to generate the sas code from the SAS Flows listed in the dataset...

```

graph LR
 A((flow_select_for_codegen_list)) --> B[Generate SAS code from S...]
 B --> C((flow_codegen_list_status))

```

**flow\_select\_for\_codegen**

Flow Selector   Knoten   Hinweise

Please select on the file system a single SAS Flow or a folder which will be parsed for SAS Flows.

Select a single SAS Flow or a folder\*

Flow

Select a single SAS Flow ⓘ

sasserver:/sasdata/WWI00\_ADV/VERSIONING/y04675\_

! Kein “out of the box” Button zur Code-Generierung wie unter SAS Studio 3.8 / oder DI Studio

- Codestand für Versionierung und Deployment benötigt
  - Code-Generierung von Flows über Rest-API
  - Umsetzung über Step zur Erzeugung von Code aus
    - einzelnen Flows
    - Flows innerhalb von ganzen Verzeichnissen
- Performanter REST-Call mit COMPUTE\_SESSION\_ID

```

/* call code generation api */
filename fn_respj temp;
proc http url = "&viya_url./studioDevelopment/code?sessionId=&SYS_COMPUTE_SESSION_ID"
 in = fn_input
 out = fn_respj
 method = "POST"
 OAUTH_BEARER=SAS_SERVICES_VERBOSE;
 headers
 "Content-Type" = "application/json;charset=utf-8"
 "Accept" = "application/json"
 ;
run;

```

- Verwendung von Python zur Verarbeitung der Rückgabe benötigt

```

/* Extract the sas code from the json response. Use python to circumvent the limit of libname json. */
%put DEBUG: Executing python code to retrieve JSON response (SAS Code) ...;
proc python;
%include fn_pycod;
run;

```

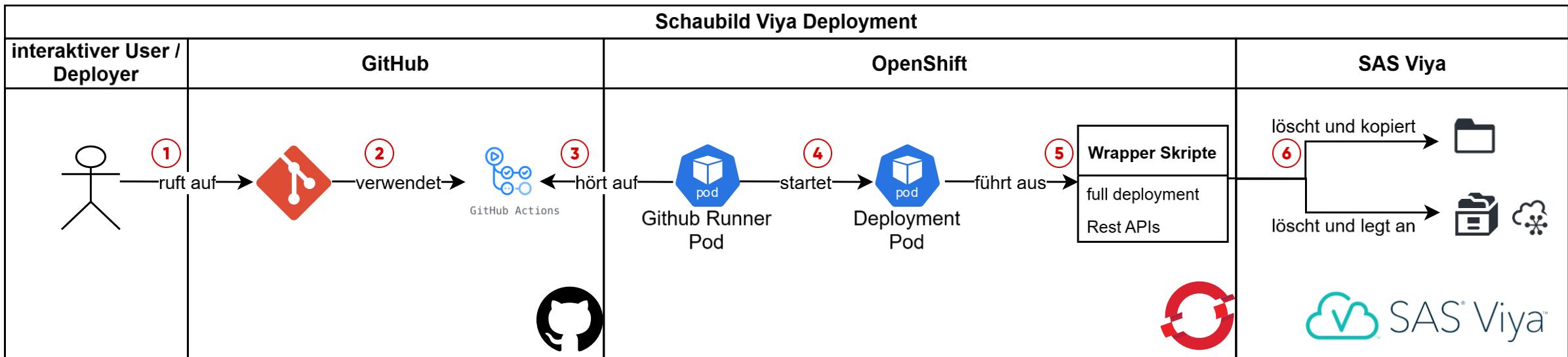
# Deploymentverfahren

# Deploymentverfahren

- Technische Umsetzung
- Live Demo
- Erreichte Ziele
- Ausblick

# Technische Umsetzung

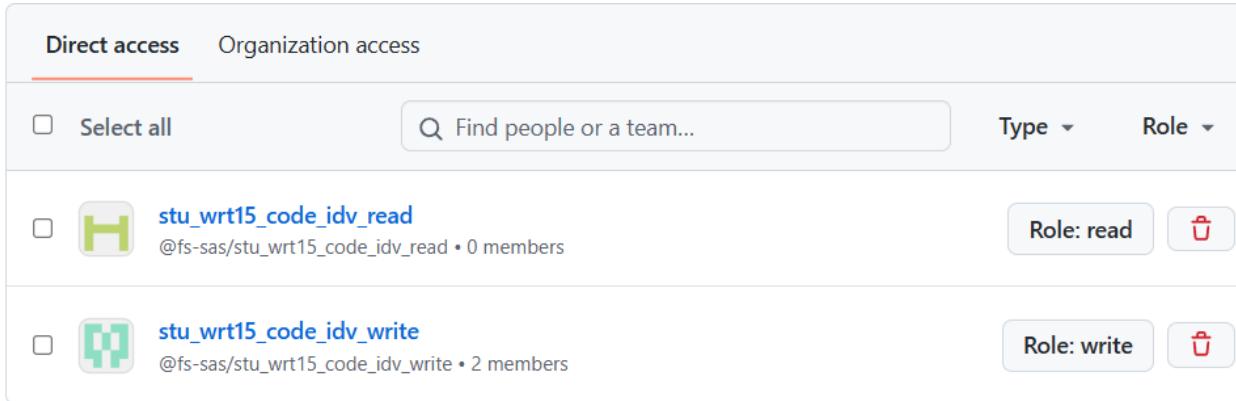
# Architektur



1. Deployer ruft das GitHub Repository auf
2. Repository verwendet die hinterlegte GitHub Action zum Starten des Deployments
3. GitHub Runner prüft alle 10 Sekunden, ob eine Action zur Verarbeitung gestartet wurde
4. GitHub Runner startet den Deployment Pod und übergibt notwendige Informationen (Mandant, Release, etc.)
5. Über den Deployment Pod werden Wrapper Skripte mit dem technischen User ausgeführt
6. Diese deployen das ausgewählte Release und ggf. Bibliotheken auf die Zielumgebung

## Berechtigungen GitHub

- SAS Viya Berechtigungsgruppen werden nach GitHub synchronisiert und automatisiert auf Teams übertragen
- Entwickler besitzen das „Write“ Recht in GitHub, alle anderen „Read“



The screenshot shows the GitHub organization access settings for the 'stu' organization. It displays two teams: 'stu\_wrt15\_code\_idv\_read' and 'stu\_wrt15\_code\_idv\_write'. The 'stu\_wrt15\_code\_idv\_read' team has a 'Role: read' and 0 members. The 'stu\_wrt15\_code\_idv\_write' team has a 'Role: write' and 2 members. There are buttons to 'Select all', 'Find people or a team...', 'Type', and 'Role'.

| Team                     | Role        | Members   |
|--------------------------|-------------|-----------|
| stu_wrt15_code_idv_read  | Role: read  | 0 members |
| stu_wrt15_code_idv_write | Role: write | 2 members |

- Einspielen von Branches auf Entwicklungsumgebungen erlaubt
- Ab Test- und Abnahmumgebung nur noch Releases
- Einspielen auf Produktion nur noch durch das Releasemanagement und Admins

# Runner

| Runners                                             |                                             |          |     | Status |     |
|-----------------------------------------------------|---------------------------------------------|----------|-----|--------|-----|
| fs-sas Openshift Actions Runners - bf5989764-vr8km  | self-hosted                                 | Linux    | X64 | Idle   | ... |
| Runner group: Default                               | docker-quay-cleared.artifactory.ww-inter... | VIYA_ETU |     |        |     |
| fs-sas Openshift Actions Runners - 9db99b58f-2vktq  | self-hosted                                 | Linux    | X64 | Idle   | ... |
| Runner group: Default                               | docker-quay-cleared.artifactory.ww-inter... | VIYA_VPU |     |        |     |
| fs-sas Openshift Actions Runners - 5b8df4b94d-5b2sm | self-hosted                                 | Linux    | X64 | Idle   | ... |
| Runner group: Default                               | docker-quay-cleared.artifactory.ww-inter... | VIYA_ITU |     |        |     |
| fs-sas Openshift Actions Runners - 9b8784ff-csln8   | self-hosted                                 | Linux    | X64 | Idle   | ... |
| Runner group: Default                               | docker-quay-cleared.artifactory.ww-inter... | VIYA_PU  |     |        |     |
| fs-sas Openshift Actions Runners - 7b7b8f7bc-5p85v  | self-hosted                                 | Linux    | X64 | Idle   | ... |
| Runner group: Default                               | docker-quay-cleared.artifactory.ww-inter... | VIYA_PTU |     |        |     |
| fs-sas Openshift Actions Runners - 685cc5bf7d-7cb2l | self-hosted                                 | Linux    | X64 | Idle   | ... |
| Runner group: Default                               | docker-quay-cleared.artifactory.ww-inter... | VIYA_STU |     |        |     |

- Ein Runner pro Viya Umgebung
  - wird über Label (z.B. VIYA\_ETU) angesprochen
- Runner wurde auf Basis des Standard redhat-github-actions Image gebaut

<https://github.com/redhat-actions/openshift-actions-runner-chart>  
<https://quay.io/organization/redhat-github-actions>

# GitHub Actions

The screenshot shows the GitHub Actions interface for a repository named 'fs-sas/bk04\_adv'. The left sidebar is collapsed, and the main area displays the 'Actions' tab for a workflow named 'deploy-action'. A modal window is open over the list of workflow runs, titled 'Run workflow'. It contains fields for 'Use workflow from' (set to 'Branch: main' with a red circle 1), 'Target Environment for Deployment' (set to 'VIYA\_ETU'), and a 'Run workflow' button. Below the modal, the list of workflow runs shows three entries, each with a green checkmark and a timestamp.

```

74 - name: Check out repository code ②
75 uses: actions/checkout@v4
76 with:
77 path: ${{ env.REPO_NAME }}
78 ref: "${{ github.ref }}"
79
80
81
82
83
84
85
86
87
88 - name: apply of deployment.yaml ③
89 run: |
90 oc apply -f ${{ env.base_repo_path }}/.deploy_yaml/wwi-deploy.yaml
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
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
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191

```

## deploy-action.yaml

- Ablage unter `.github/workflows` in jedem Repository
- Parameter: Branch und Umgebung ①
- Funktionsumfang
  - Auschecken des Repository ②
  - Berechtigungsprüfungen
  - Erzeugung eines Pods ③

# GitHub Actions

base\_configuration / wwi-deploy.yaml

y04675\_wwi Update wwi-deploy.yaml

Code Blame 47 lines (47 loc)...

```

1 kind: Job
2 apiVersion: batch/v1
3 metadata:
4 name: wwi-viya-deploy
5 spec:
6 backoffLimit: 0
7 template:
8 spec:
9 volumes:
10 - name: sasdata
11 persistentVolumeClaim:
12 claimName: sas-sasdata
13 - name: access-clients
14 persistentVolumeClaim:
15 claimName: sas-access-clients
16 containers:
17 - resources:
18 limits:
19 cpu: '1'
20 memory: 2Gi
21 requests:
22 cpu: 100m
23 memory: 100Mi
24 name: wwi-viya-deploy
25 args: ["deploy_client", "base_repo_path"]
26 imagePullPolicy: Always
27 image: 'docker-releases.artifactory.ww-intern.de/de_wtag_sas_viya_admin/container-deployment:0.14'
28 envFrom:
29 - secretRef:
30 name: wwi-client
31 - configMapRef:
32 name: wwi-environment
33 volumeMounts:
34 - name: sasdata
35 mountPath: /sasdata
36 - name: access-clients
37 mountPath: /opt/access-clients
38 tolerations:
39 - key: workload.sas.com/class
40 operator: Equal
41 value: compute
42 effect: NoSchedule
43 restartPolicy: Never
44 serviceAccountName: sas-programming-environment
45 securityContext:
46 runAsUser: to_modified_runAsUser
47 runAsNonRoot: true

```

① ② ③ ④

## wwi-deploy.yaml

- Basis .yaml für die Erstellung des Deployment-Pods
- Ausführung eines selbst gebauten Docker Images ①
- Im Pod wird ein Wrapper Shell Skript parametrisiert aufgerufen
- Der Pod läuft SAS-seitig mit dem technischen Schreibuser des jeweiligen Mandanten ②
- Anbindung benötigter Filesysteme ③
- Variablen und Passwörter in OpenShift Secrets und ConfigMaps hinterlegt ④

# Live Demo

# Erreichte Ziele

# Erreichte Ziele

- ✓ gute Integration in SAS Studio
- ✓ vollautomatisiertes Deployment ohne manuelle Eingriffe
- ✓ gute Performance
- ✓ Deployment von vollständigen Softwarepaketen
- ✓ Reduzierung auf ein einziges Tool
  - integrierter Freigabeprozess
  - Audit-Logs
  - Verwendung der SAS Viya Berechtigungsgruppen

# Ausblick

# Ausblick

- Deployment von SAS Content Objekten (VA Reports, Job Definitions)
- Performance-Optimierungen (Berechtigung CAS Libraries)
- Prüfung Entwicklerrichtlinien
- Dashboard mit Codeständen (Git Branches / Tags) je Umgebung
- Verbessertes Error Handling



Vielen Dank! Fragen?