Wüstenrot & Württembergische 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

w&w gruppe

Wüstenrot & Württembergische AG
Stand: 27. Januar 2025

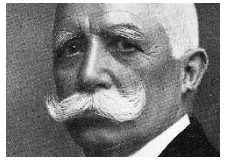
Der Vorsorge verbunden.
Das sind wir:
die W&W-Gruppe.

1828

Gründung der
„Württembergischen Privat-Feuer-
Versicherungs-Gesellschaft“

1921

Gründung des Vereins
„Gemeinschaft der
Freunde“ in Wüstenrot



1999

wüstenrot

Württembergische

Fusion von Wüstenrot
und der Württembergischen

2007

Gemeinsame
Identität

wüstenrot
württembergische

2017

Adam Riese

Gründung der
Adam Riese GmbH,
ein Corporate Startup
der W&W-Gruppe

2019

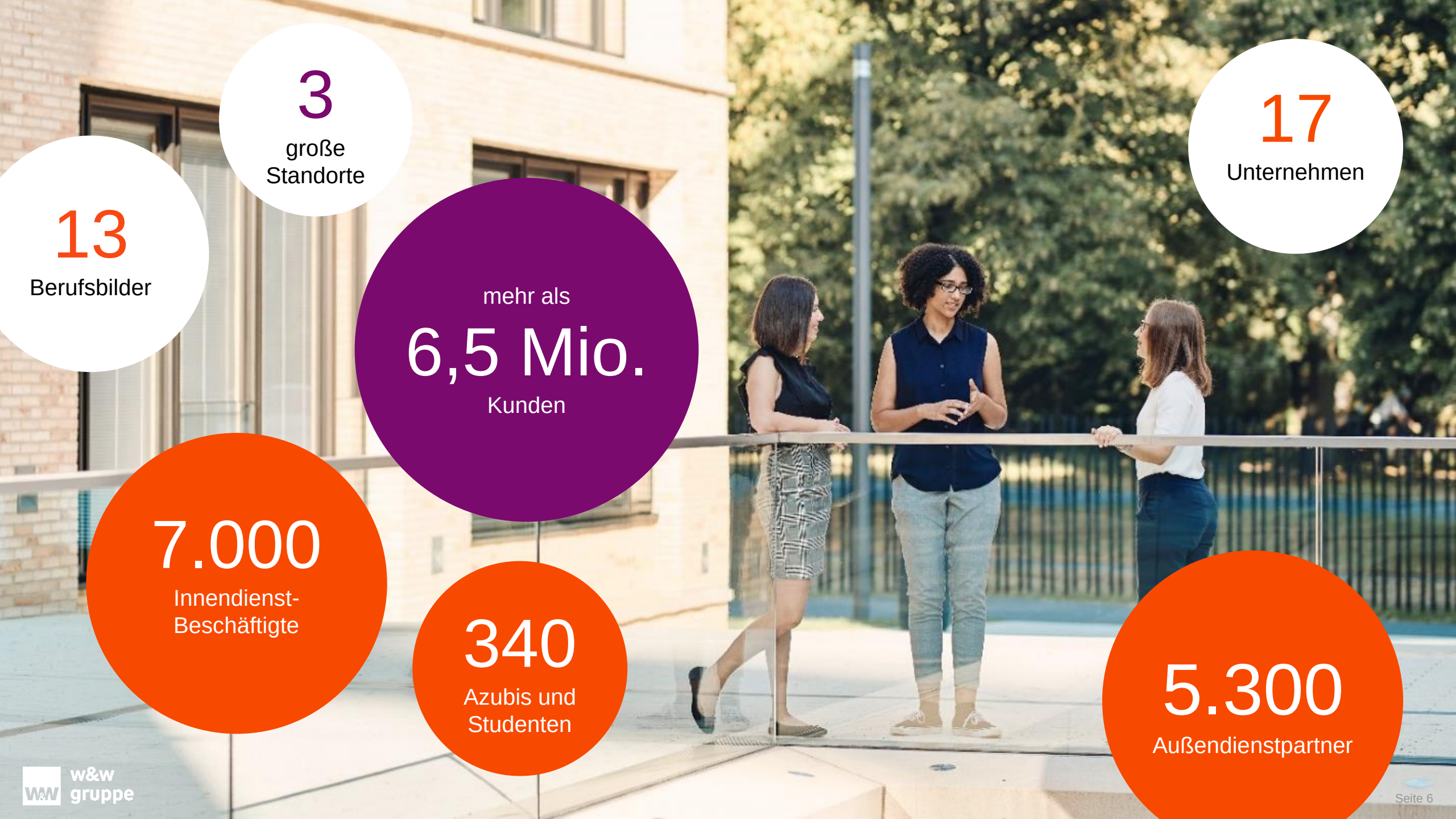
Offizieller Auftritt als
W&W-Gruppe

w&w
gruppe

2023

Gemeinsamer Einzug
auf den W&W-Campus

w&w
gruppe


3
große Standorte
17
Unternehmen
13
Berufsbilder
mehr als
6,5 Mio.
Kunden
7.000
Innendienst-Beschäftigte
340
Azubis und Studenten
5.300
Außendienstpartner

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. €

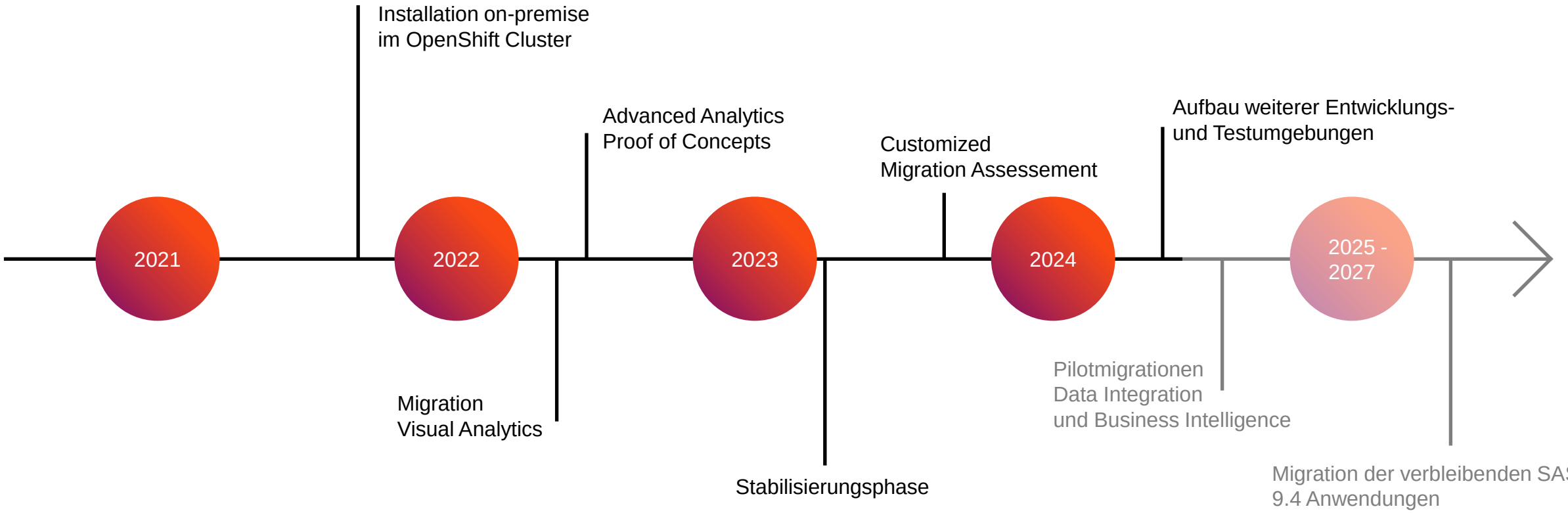
Quelle: Geschäftsbericht Wüstenrot und Württembergische AG 2023, Werte gerundet.

w&w gruppe

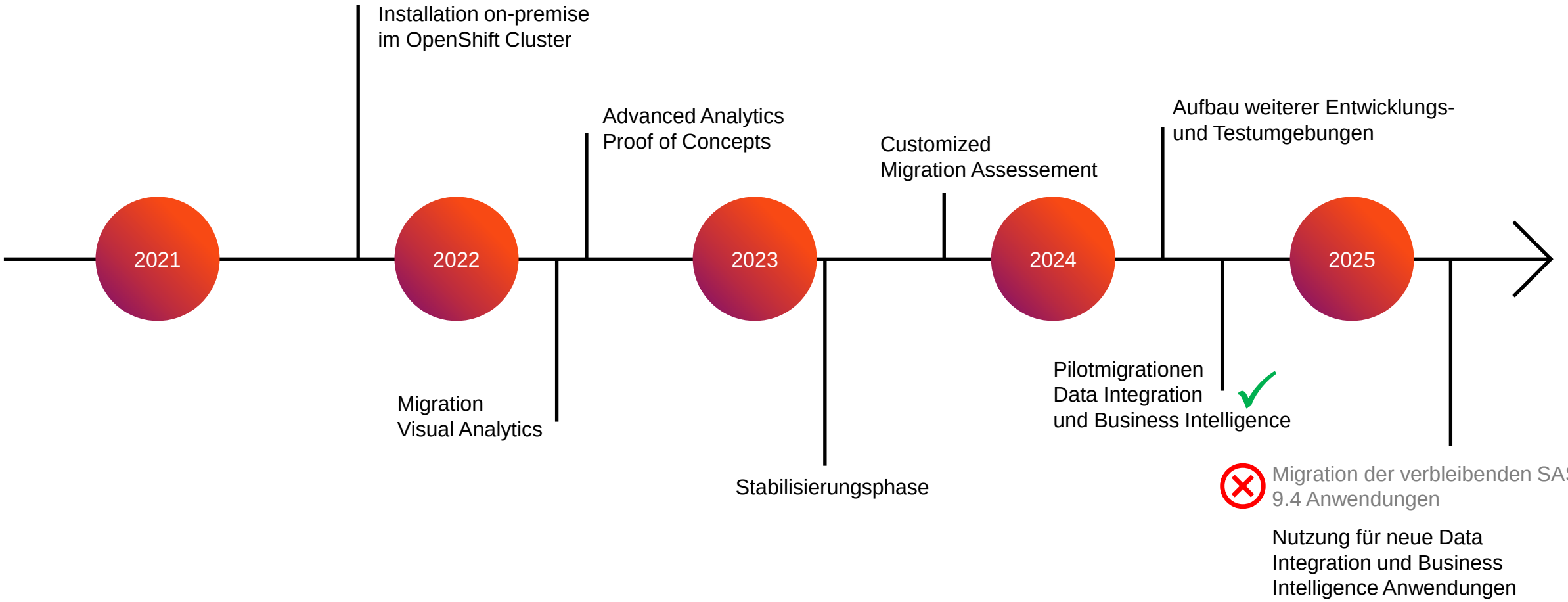
SAS Viya in der W&W

w&w gruppe

SAS Viya in der W&W – Rückblende PNT 2024



SAS Viya in der W&W



w&w gruppe

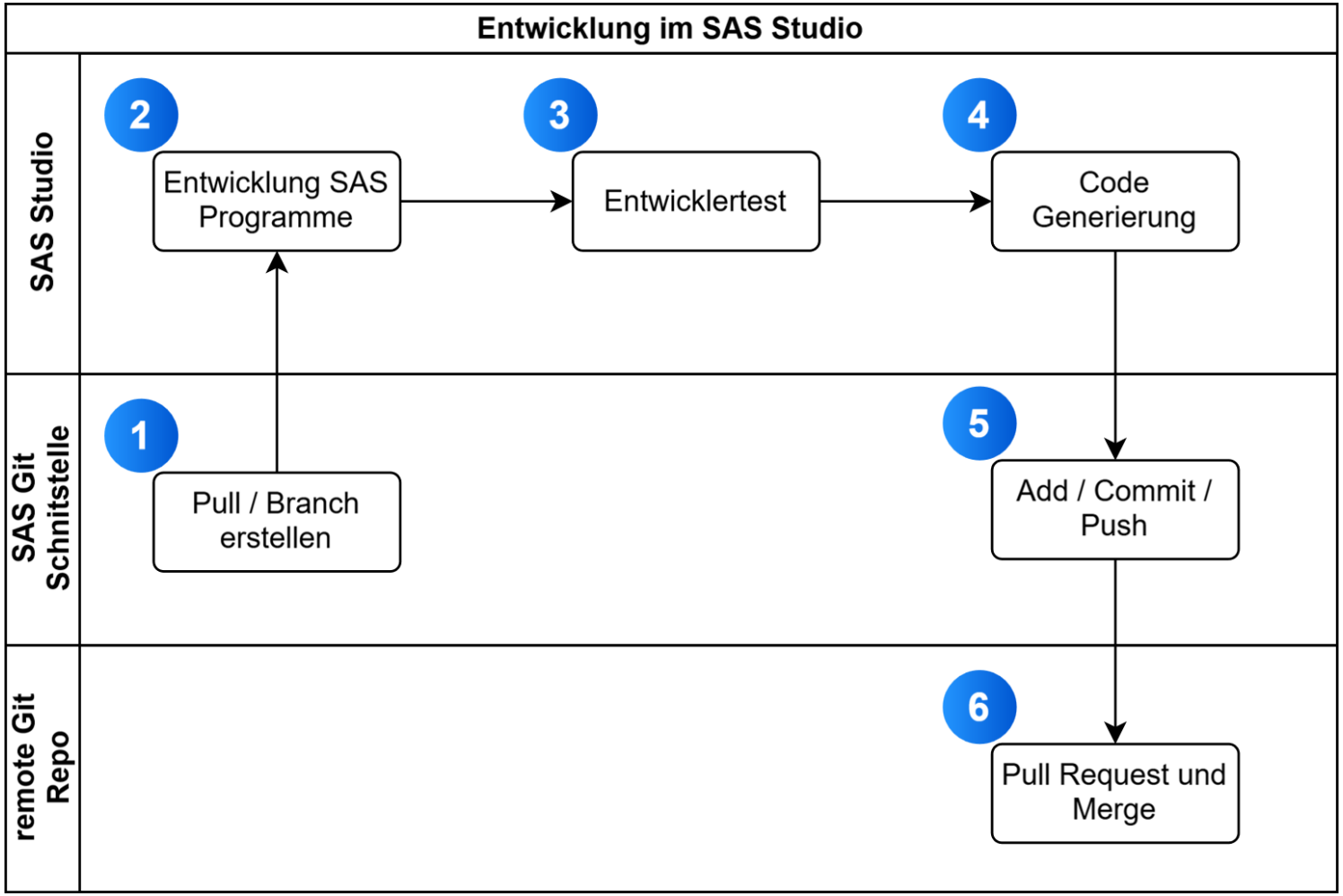
Entwicklungsprozess

Entwicklungsprozess

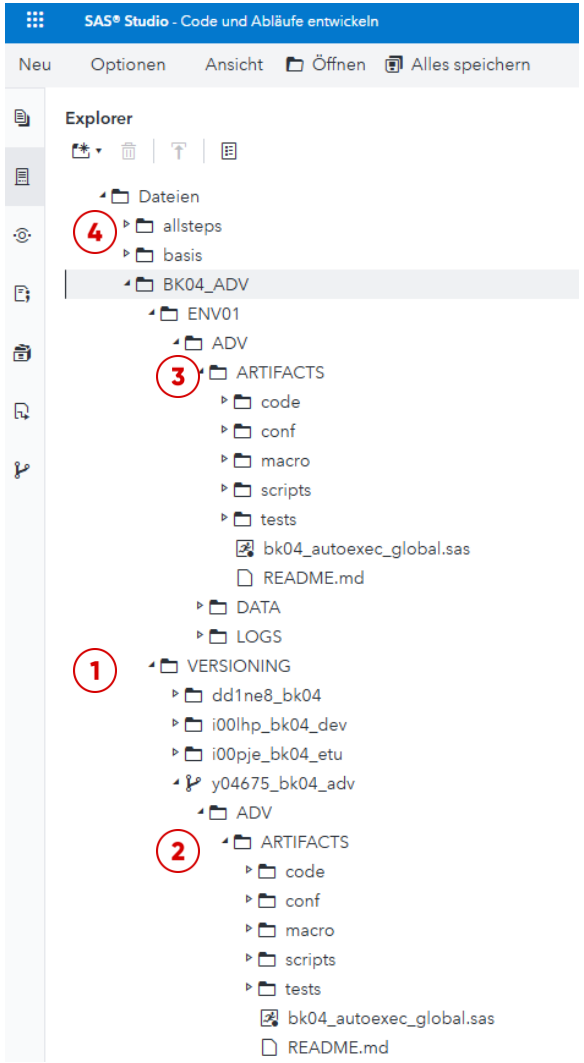
1. Prozessfluss
2. Ablagestruktur
3. Versionierung / Git Integration
4. Code-Generierung

w&w gruppe

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)

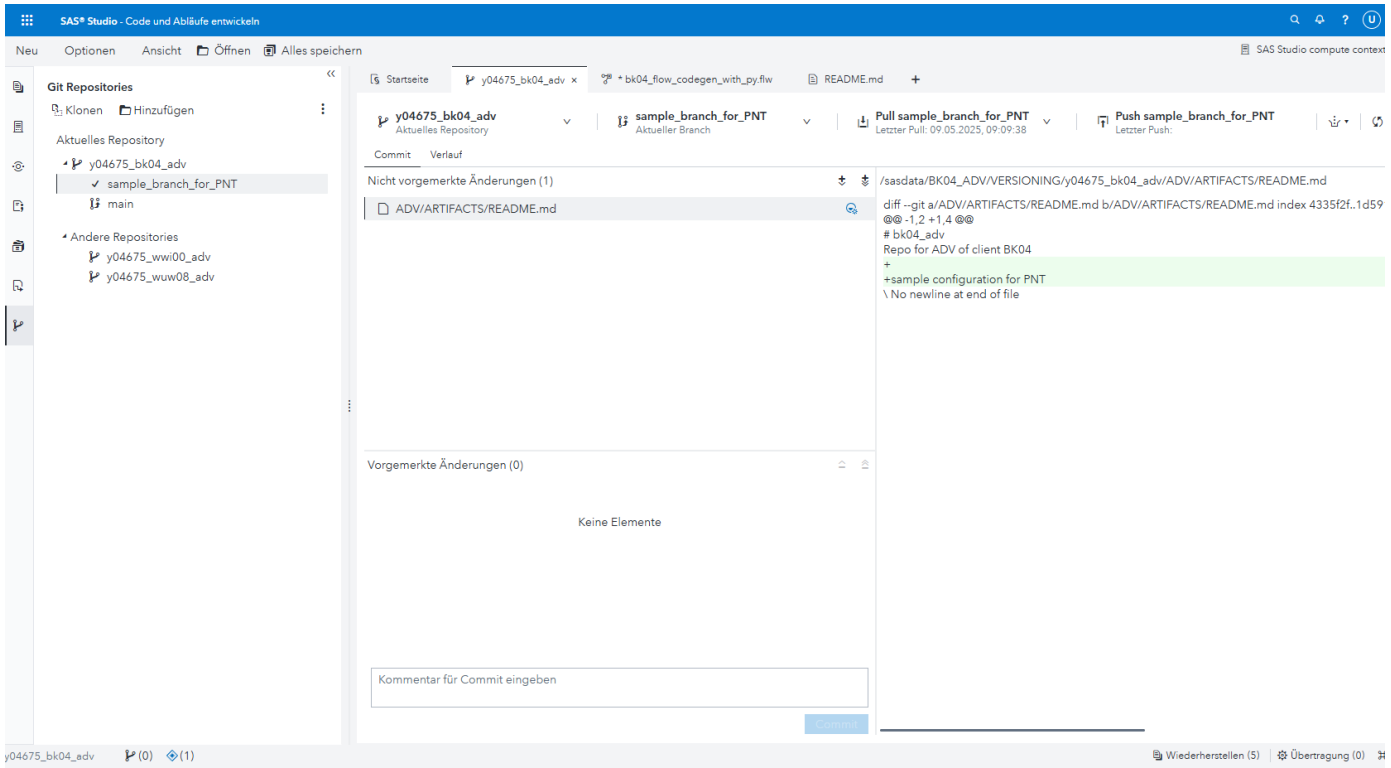
❗ Flows enthalten absolute Pfadreferenzen von Custom Steps im Filesystem

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

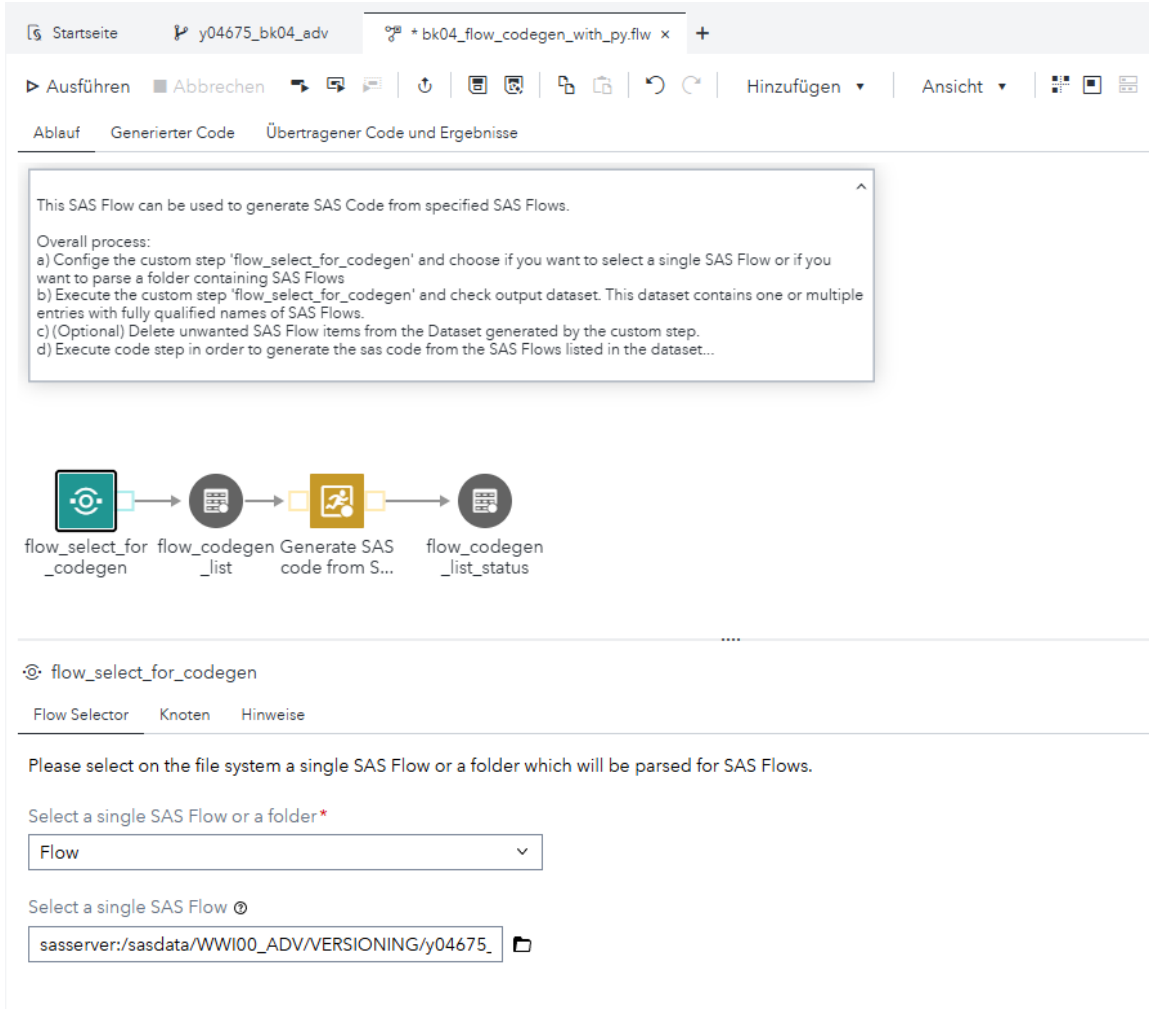
Statusabfrage für SAS Funktionen:

```
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/n1mlc3f9w9zh9fn13qswiq6hrta0.htm

w&w gruppe

Code-Generierung



- ❗ 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;
```

https://communities.sas.com/t5/SAS-Communities-Library/How-To-Get-the-Code-From-a-SAS-Studio-Flow-and-Why-It-Matters/ta-p/859324

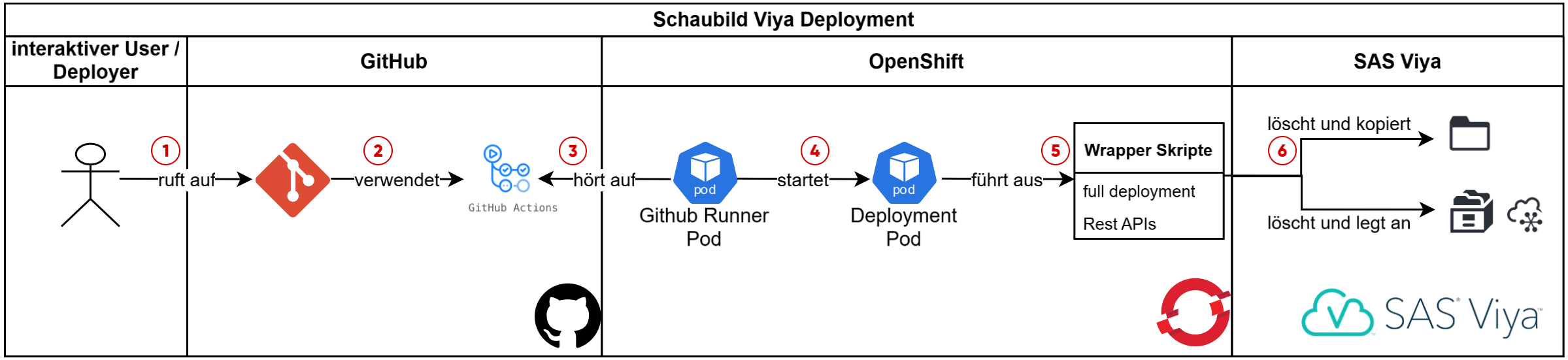
Deploymentverfahren

Deploymentverfahren

- Technische Umsetzung
- Live Demo
- Erreichte Ziele
- Ausblick

Technische Umsetzung

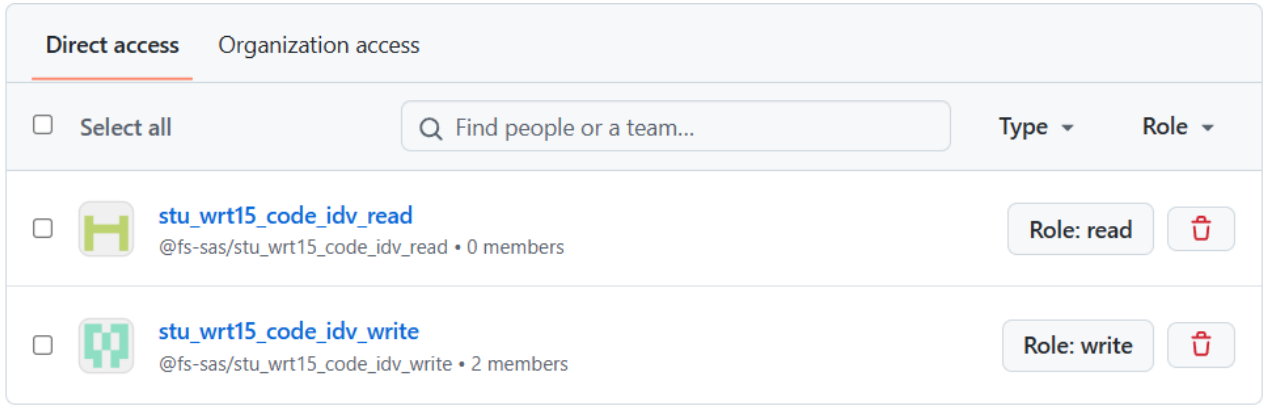
Architektur



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

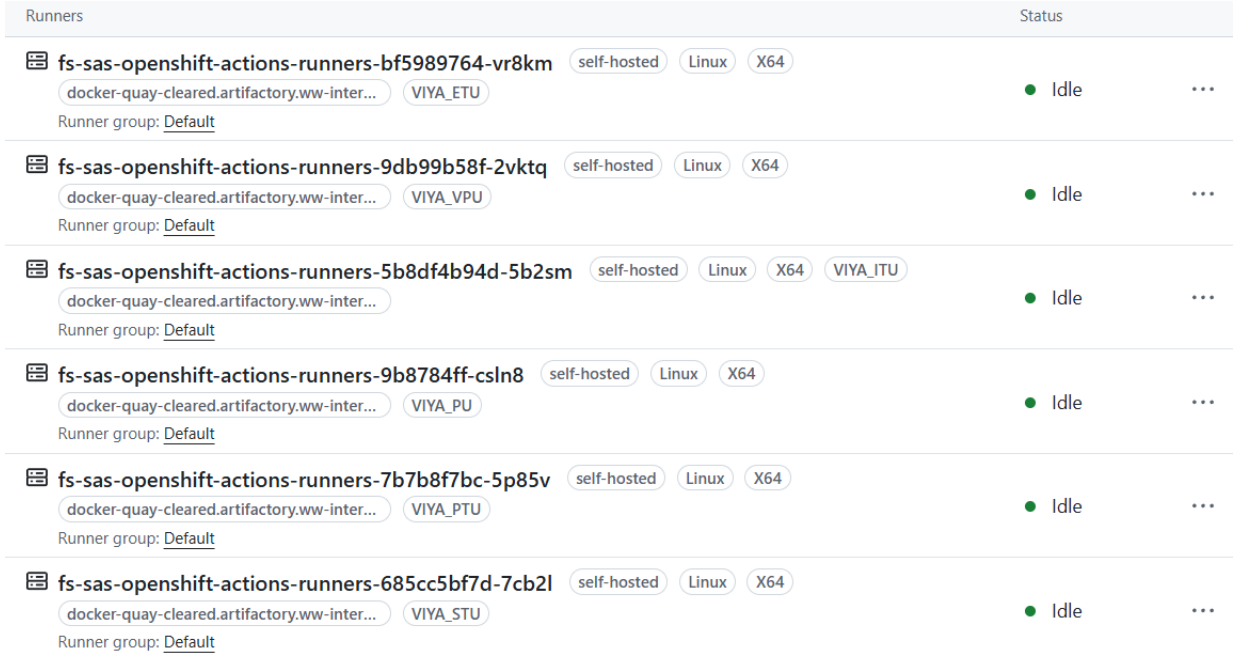
Berechtigungen GitHub

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



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

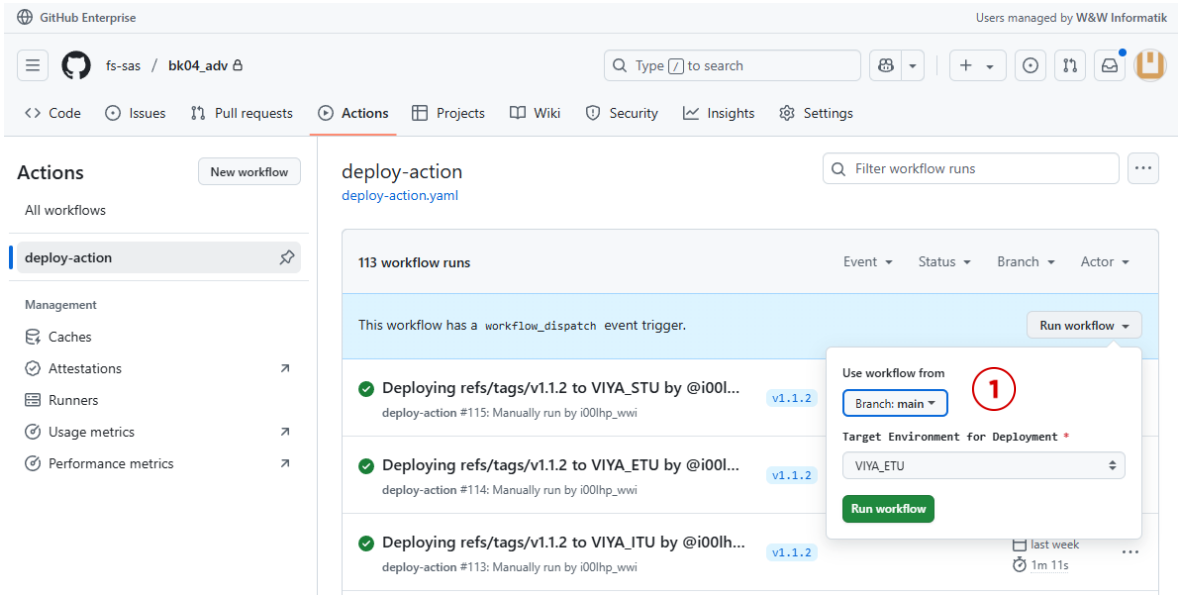
Runner



- 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




```
74          - name: Check out repository code
75            uses: actions/checkout@v4
76            with:
77              path: ${{ env.REPO_NAME }}
78              ref: "${{ github.ref }}"

188         - name: apply of deployment.yaml
189           run:  |
190                 oc apply -f ${{ env.base_repo_path }}/.deploy_yaml/wwi-deploy.yaml
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)…

```yaml
kind: Job
apiVersion: batch/v1
metadata:
  name: wwi-viya-deploy
spec:
  backoffLimit: 0
  template:
    spec:
      volumes:
        - name: sasdata
          persistentVolumeClaim:
            claimName: sas-sasdata
        - name: access-clients
          persistentVolumeClaim:
            claimName: sas-access-clients
      containers:
        - resources:
            limits:
              cpu: '1'
              memory: 2Gi
            requests:
              cpu: 100m
              memory: 100Mi
          name: wwi-viya-deploy
          args: ["deploy_client","base_repo_path"]
          imagePullPolicy: Always
          image: 'docker-releases.artifactory.ww-intern.de/de_wwag_sas_viya_admin/container-deployment:0.14'
          envFrom:
          - secretRef:
              name: wwi-client
          - configMapRef:
              name: wwi-environment
          volumeMounts:
            - name: sasdata
              mountPath: /sasdata
            - name: access-clients
              mountPath: /opt/access-clients
      tolerations:
        - key: workload.sas.com/class
          operator: Equal
          value: compute
          effect: NoSchedule
      restartPolicy: Never
      serviceAccountName: sas-programming-environment
      securityContext:
        runAsUser: to_modified_runAsUser
        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)

w&w gruppe

Live Demo

Erreichte Ziele

w&w gruppe

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

w&w gruppe

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

w&w gruppe

Vielen Dank! Fragen?

w&w gruppe