



HALE AND HEARTY MIT HALE UND DEEGREE

Praxisbericht eines INSPIRE-Stacks mit OpenSource

Oliver Schmidt, LVermGeo RP



Kurze Vorstellung...

Oliver Schmidt (Dipl.-Geograph)

- Schon im Studium viele Kenntnisse in der Verarbeitung von zahlreichen Geodaten (Fernerkundung, Meteorologie)
- Seit 2014 beim LVermGeo RP in Koblenz
- Betreuung eines Serversystems zur Bereitstellung von WMS, WFS und INSPIRE-Diensten seit 2015

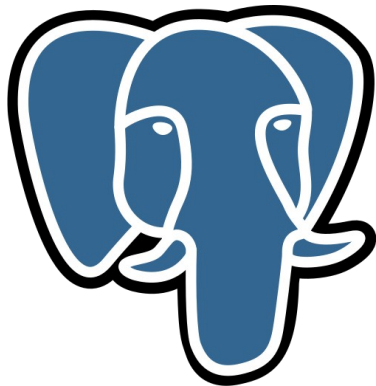


Ausgangspunkt

- Datenimport, -haltung und -bereitstellung sollen ausschließlich mit OpenSource-Software geschehen
- Basistechnologie soll keine Kosten verursachen
- Hohe Performanz und Verfügbarkeit unabdingbar
- System muss einfach zu skalieren sein
- INSPIRE-Vorgaben müssen voll unterstützt werden
- Möglichst keine Eigenentwicklung von Software
- Volle Kontrolle über die Prozesse und eigenes Hosting

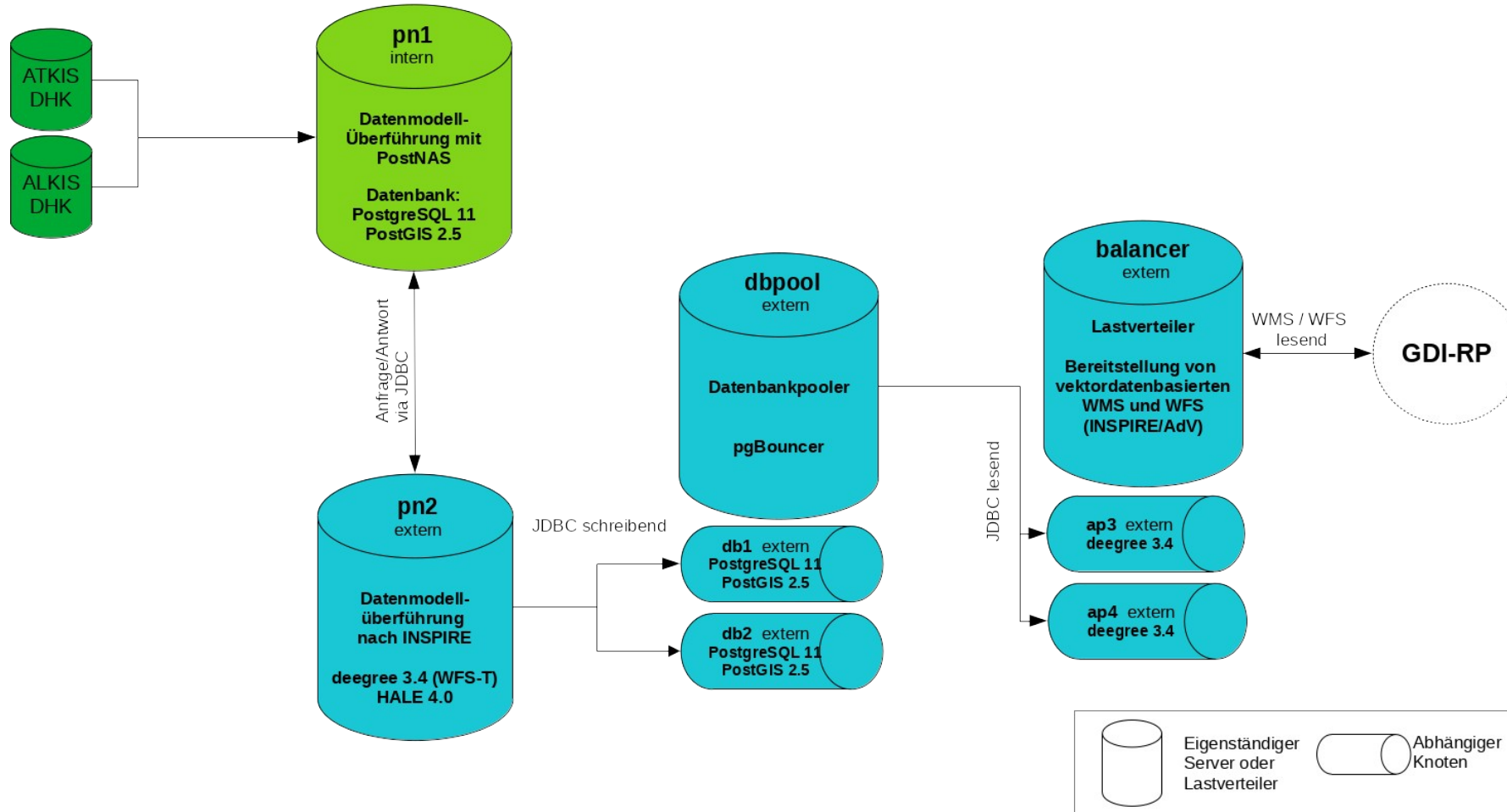
Hardware und Basistechnologie

- Alle Server sind virtualisiert (VMs): Verschieben, Klonen, Erweiterung von Ressourcen, ...
- Debian GNU/Linux 10 „Buster“ ohne grafische Oberfläche
- Datenhaltung in PostgreSQL 11 mit PostGIS 2.5
- Nutzung des norBIT ALKIS-Import (auch für BasisDLM, DLM50)



PostNAS

Hardware und Basistechnologie



hale studio Open Source

- ETL-Software (vgl. FME) zum Mapping von AAA- bzw. PostNAS-Schemata auf INSPIRE-Schemata mit anschließender Datentransformation in GML
- Mappingtabellen der AdV für INSPIRE wurden in hale-Projekte überführt (AAA nach INSPIRE)
- Weg von AAA nach INSPIRE und AAA nach PostNAS war somit bekannt
- Mapping von PostNAS nach INSPIRE

hale studio Open Source



Rheinland-Pfalz
LANDESAMT FÜR VERMESSUNG
UND GEOBASISINFORMATION

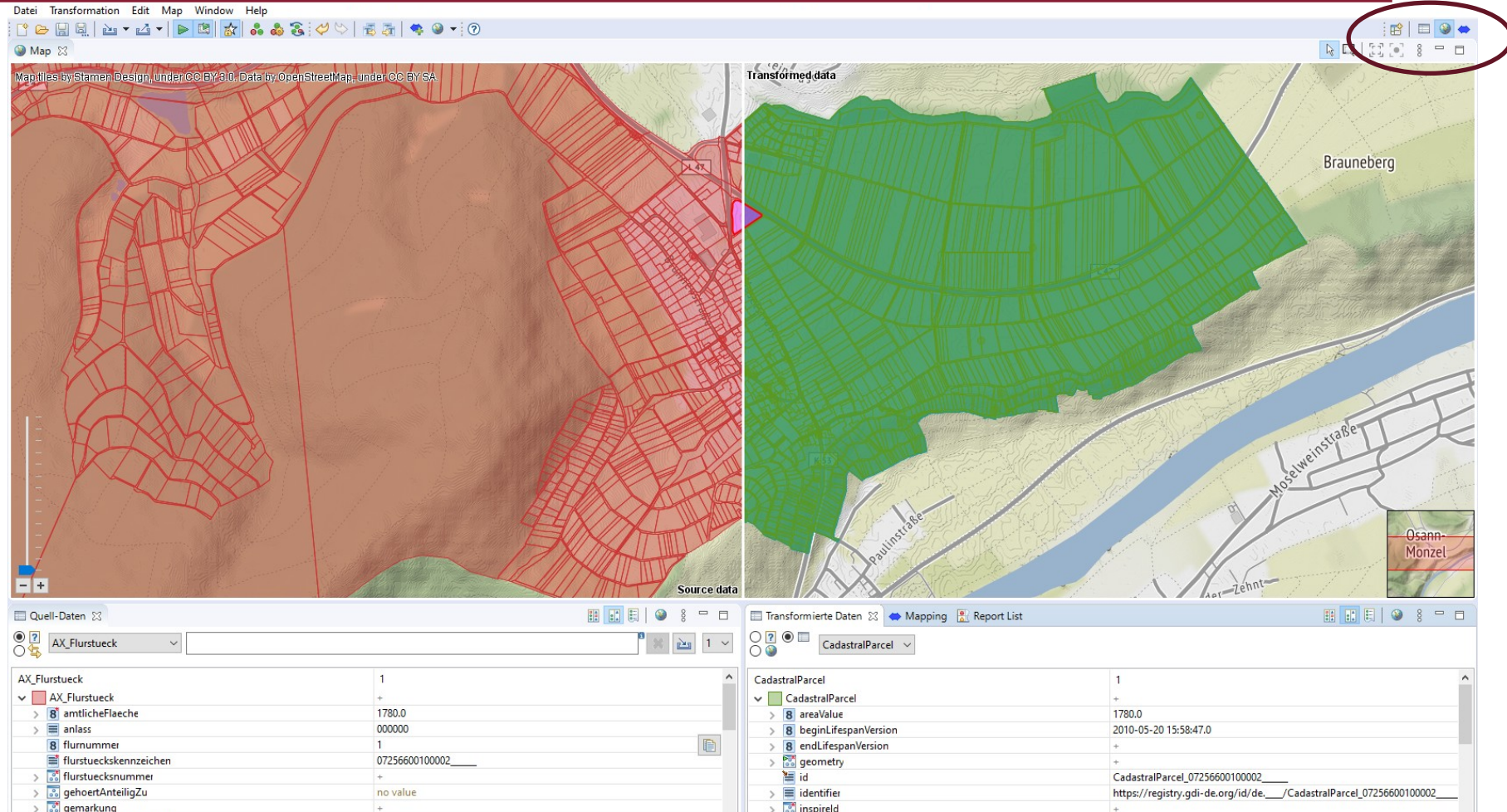
The screenshot displays the hale studio Open Source interface. The top menu bar includes Datei, Transformation, Edit, Window, and Help. Below the menu is a toolbar with various icons. The main workspace is divided into several panels:

- Source:** A tree view showing a hierarchy of data types. The selected item is **AX_Flurstueck × 2369**, which includes properties like **location (0..1)**, **abweichenderRechtszustand (0..1)**, **amtlicheFlaeche × 2369**, **anlass (0..2) × 2369**, **beziehtSichAufFlurstueck (0..n)**, **boundedBy (0..1)**, **description (0..1)**, **descriptionReference (0..1)**, **flurnummer (0..1) × 2369**, **flurstuecksfolge (0..1)**, **flurstueckskennzeichen × 2369**, **flurstuecksnummer × 2369**, **gehörtAnteiligZu (0..n) × 31 (1)**, **gemarkung × 2369**, **gemeindezugehörigkeit (0..1) × 2369**, **hatDirektUnten (0..n)**, **id × 2369**, **identifizier (0..1) × 2369**, **inversZu_beziehtSichAuf (0..n)**, **inversZu_dientZurDarstellungVon_AP_Darstellung (0..n)**, **inversZu_dientZurDarstellungVon_AP_FPO (0..n)**, **inversZu_dientZurDarstellungVon_AP_KPO_3D (0..n)**, **inversZu_dientZurDarstellungVon_AP_LPO (0..n)**, **inversZu_dientZurDarstellungVon_AP_LTO (0..n)**, **inversZu_dientZurDarstellungVon_AP_PPO (0..n)**, **inversZu_dientZurDarstellungVon_AP_PTO (0..n)**, **inversZu_hatDirektUnten (0..n)**, **inversZu_verweistAuf (0..n)**, and **istAboeleitetAus (0..n)**.
- Target:** A tree view showing a hierarchy of data types. The selected item is **CadastralParcel × 2369**, which includes properties like **location (0..1)**, **administrativeUnit (0..1)**, **areaValue (0..1) × 2369**, **basicPropertyUnit (0..n)**, **beginLifespanVersion × 2369**, **boundedBy (0..1)**, **description (0..1)**, **descriptionReference (0..1)**, **endLifespanVersion (0..1) × 2369**, **geometry × 2369**, **id (0..1) × 2369**, **identifizier (0..1) × 2369**, **inspireId × 2369**, **label × 2369**, **metaDataProperty (0..n)**, **name (0..n)**, **nationalCadastralReference × 2369**, **referencePoint (0..1) × 2369**, **validFrom (0..1) × 2369**, **validTo (0..1) × 2369**, **zoning (0..1) × 2369**, and **CadastralZoning**.
- Alignment:** A diagram showing the transformation between the Source and Target data types. It includes various transformation rules such as **Retype**, **Rename**, **AdV Maßinheit zu UCUM**, **AdV INSPIRE localId**, **AdV INSPIRE identifizier (URI)**, **AdV INSPIRE localId**, **Rename**, **Groovy script**, **zoning.href**, **label**, **Rename**, **geometry**, **referencePoint.Point**, **validFrom**, **endLifespanVersion.nilReason**, **inspireId.identifizier.namespace**, and **validTo.nilReason**.
- Instance validation:** A section at the bottom left showing the validation status. It includes a **Report** table with columns for **Success**, **Summary**, and **Time**. The **Success** column shows **true**, the **Summary** column shows **Finished successfully**, and the **Time** column shows **Mon May 02 15:32:06 CEST 2022**.
- Log:** A section at the bottom right showing a list of log entries. The log entries include **Instance validation**, **Instance transformation**, **Load data into database**, **GZipped XML file import**, and **Load data into database**, with timestamps ranging from 15:31:01 to 15:31:57.

hale studio Open Source

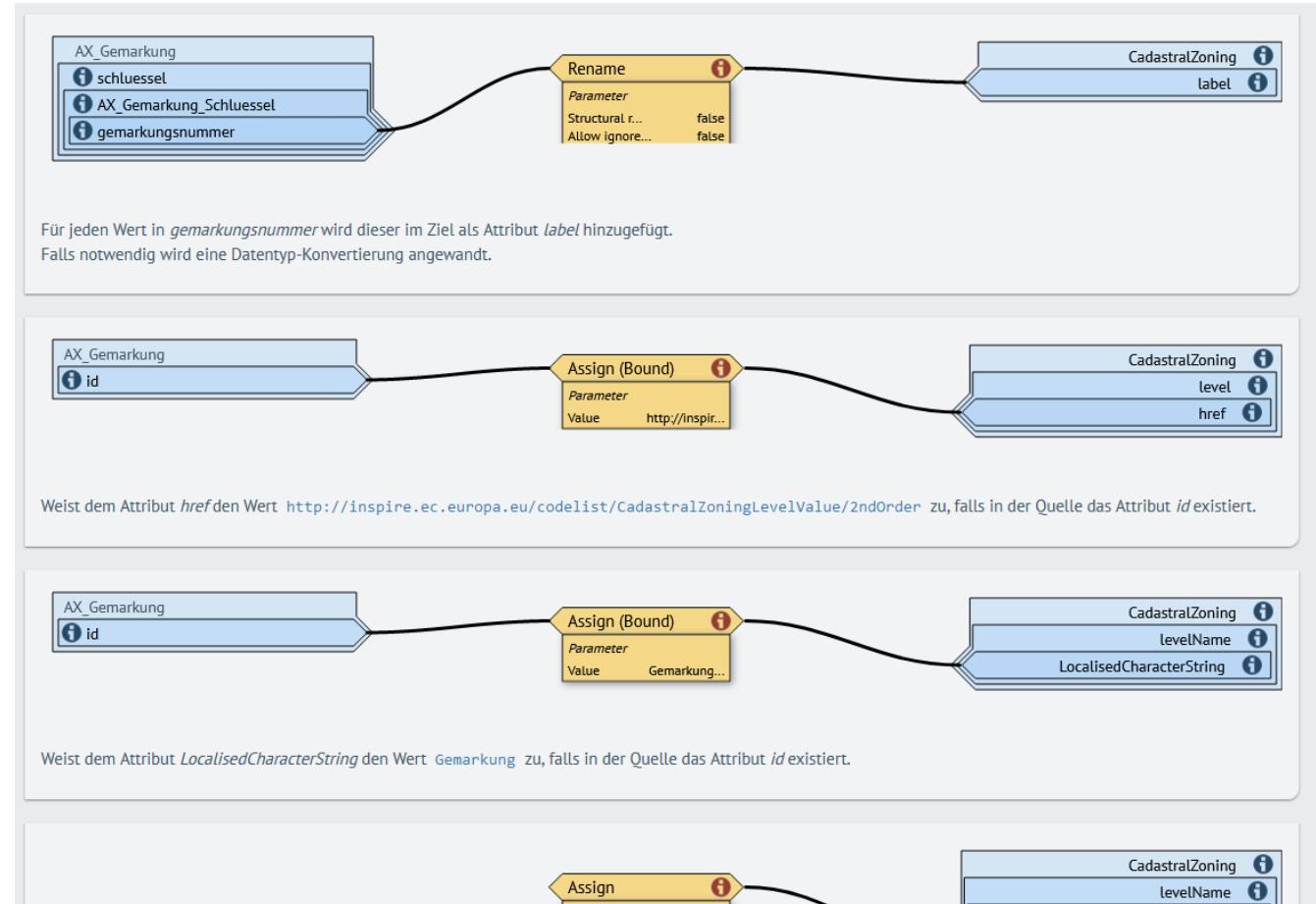


Rheinland-Pfalz
LANDESAMT FÜR VERMESSUNG
UND GEOBASISINFORMATION



hale studio Open Source

- Doku der Alignments als HTML exportierbar
- Sehr gute Übersicht über die Mappings und Eigenschaften der Attribute
- Doku auch ohne Daten erstellbar



hale studio Open Source

- hale kann auch per Shell/Kommandozeile ausgeführt werden
- Keine grafische Oberfläche zum Ausführen nötig, alle hale-Projekte können somit auf Headless-Servern genutzt werden
- Fertiges .deb-Paket für Debian GNU/Linux
- Aufrufbeispiele in hale-Doku oder direkt in der Hilfe via Shell
- Wichtig: hale läuft aktuell noch mit JAVA 8 → adoptopenjdk-8-hotspot

hale studio Open Source

Transformation mittels hale auf der Kommandozeile

```
### Ausführen der HALE-Commandline ###
```

```
{HALE} transform -project ~/contrib/alignments/${dbname}/postnas-inspire-alignments/projects/${annex_thema}-db.halex \  
-source jdbc:postgresql://${server}/${dbname} -providerId eu.esdihumboldt.hale.io.jdbc.instance.reader \  
-SdefaultSrs code:EPSG:25832 \  
-Sjdbc.user postgres \  
-Sjdbc.password ${password} \  
-target ${output_source}/${modellart}_${annex_thema}.gml -providerId eu.esdihumboldt.hale.io.wfs.fc.write-2.0 \  
-Scrs.epsg.prefix http://www.opengis.net/def/crs/EPSG/0/ \  
-Scrs code:EPSG:4258 \  
-SskipFeatureCount true \  
-reportsOut ${output_source}/reports/${modellart}_${annex_thema}-${report}.txt \  
-statisticsOut ${output_source}/reports/${modellart}_${annex_thema}-${report}.json \  
-stacktrace \  
-trustGroovy
```

hale studio Open Source

Upload der transformierten GML durch hale nach deegree3

```
###Ausführen der HALE-Commandline###
hale data rewrite \
--data ${output_source}/${modellart}_${annex_thema}.gml \
--data-setting suppressParsingGeometry=true \
--schema-setting relevantMode=featureTypes \
--target http://localhost:8080/services/${modellart}_${annex_thema} \
--target-writer eu.esdihumboldt.hale.io.wfs.write.partitioned \
--target-setting partition.mode=cut \
--target-setting wfsVersion=2.0.0 \
--target-setting instancesPerRequest=10000 \
--reportsOut ${output_source}/reports/${modellart}_${annex_thema}-upload.txt \
-stacktrace
```

hale studio Open Source

- Aufwändiger Prozess, da alle Schritte in der Kommandozeile selbst erstellt werden müssen → Orchestrierung notwendig!
- Transformation von allen Annex-Themen, von denen BasisDLM, DLM50 und DLKM betroffen sind
- Transformation und Upload nach deegree3 per WFS-T in zwei Schritte aufgeteilt → Transformationsfehler landen nicht automatisch in den INSPIRE-Diensten
- Partitionierter WFS-T wegen zu großer Datenmengen → Anpassung des deegree3-Quelltexts in unserer Installation



deegree3 Community Edition



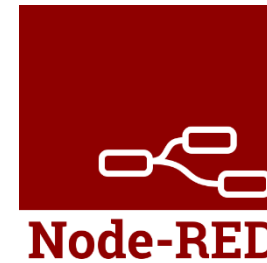
- Betrieb im Docker-Container, für einen schnellen Austausch
- INSPIRE-Workspace wird per Dockerfile in den Container gemountet, damit dieser persistent bleibt und bei Bedarf einfach ausgetauscht werden kann
- Dienste-Konfigurationen für internen deegree3 unterscheidet sich von den extern verfügbaren deegree3-Instanzen: WFS-T und kein WMS vs. WFS und WMS
- Deaktivierung der Prüfung auf interne Referenzen, da es sonst zu Problemen beim partitionierten WFS-T kommt



deegree3 Community Edition



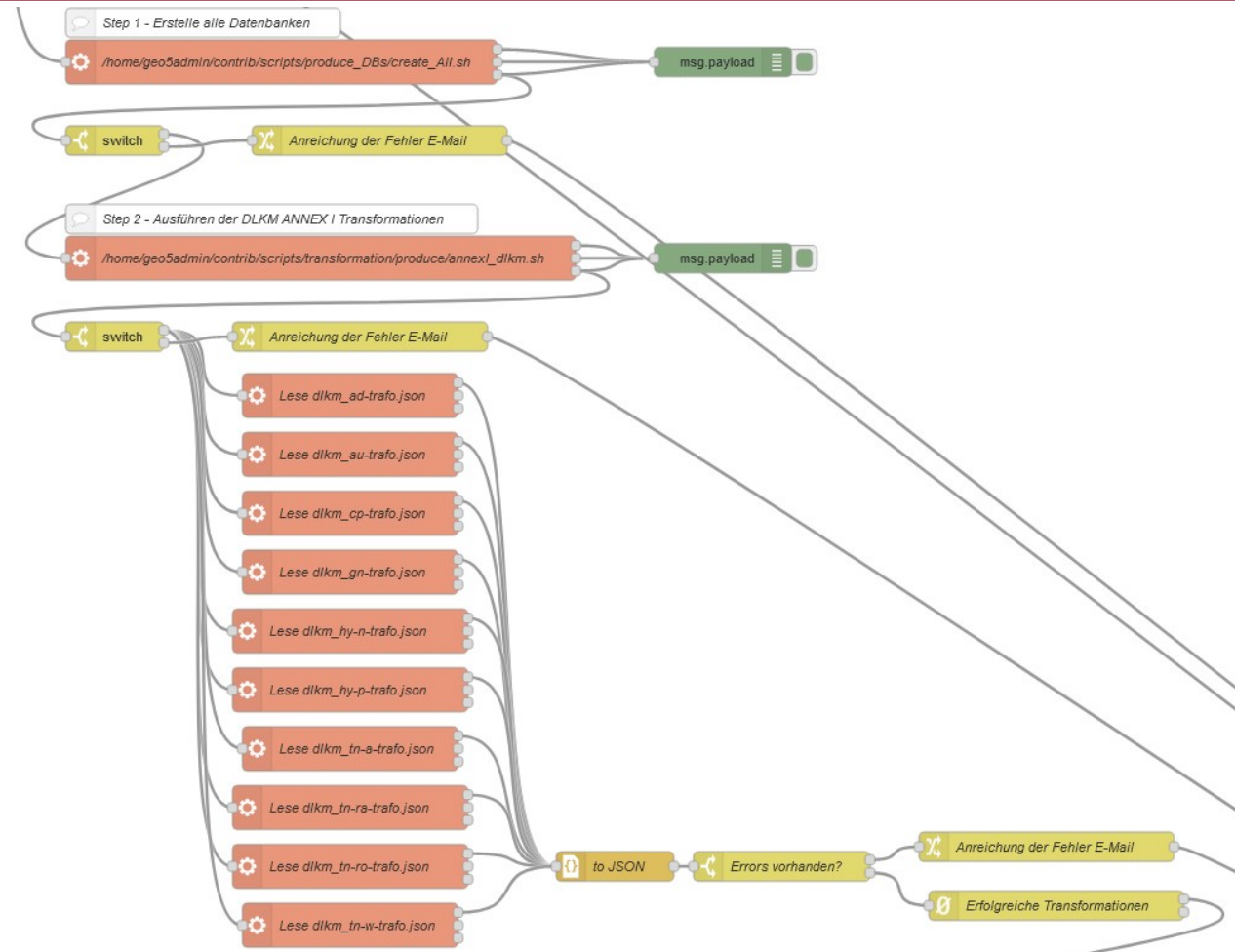
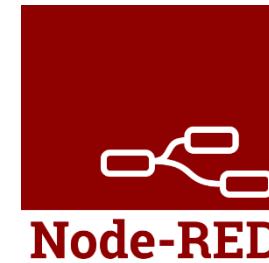
- Upload erfolgt zunächst in temporäre Datenbanken, da sonst der „Live-Betrieb“ gestört wäre → nach erfolgreichem Upload werden die Datenbanken umbenannt
- Reload des deegree3-Workspaces bei externen Eingriffen in die Datenbankverbindungen nötig, sonst kommt es zu Fehlern
- Reload auch notwendig, wenn Konfigurationsdateien des deegree3 extern verändert werden



Orchestrierung mit Node-RED

- Datenflussorientierte Programmierung
- Ursprünglich zur Automatisierung von IoT-Komponenten
- Steuerung von Shellskripten
- Prüfung von Logdateien und Mailversand des aktuellen Status
- Datenflüsse können miteinander verknüpft werden
- Grafische Oberfläche im Browser, Betrieb im Docker-Container

Orchestrierung mit Node-RED



Aktuelle und zukünftige Entwicklungen

hale studio

- Anpassung hale-Alignments an GeoInfoDok 7.1.1
- Erstellung eines „referenceDataset“ für deegree3

deegree3

- Performanz-Verbesserungen des WMS-Renderings (LBG) mittels „referenceDataset“
- Umstellung von Java 8 auf Java 11 OpenJDK (LVermGeo)
- Implementierung PROJ6 (LDBV)



Rheinland-Pfalz

LANDESAMT FÜR VERMESSUNG
UND GEOBASISINFORMATION

Vielen Dank für Ihre Aufmerksamkeit

Oliver Schmidt, oliver.schmidt@vermkv.rlp.de

Landesamt für Vermessung
und Geobasisinformation Rheinland-Pfalz

Von-Kuhl-Straße 49

56070 Koblenz

www.lvermgeo.rlp.de

Wir liefern die GeoBasis.

UTM-Koordinate:
32 U 399363 5581262

Geografische Koordinate:
7° 35' 5,0" L 50° 22' 29,3" B

